

Certified Java Programmer Exam

1.4 Mock Exam

[Type Conversions]

Question 1

```
class A {  
    public static void main(String[] args) {  
        short s1 = 1;           //1  
        char c1 = 1;            //2  
        byte b1 = s1;           //3  
        byte b2 = c1;           //4  
        final short s2 = 1;     //5  
        final char c2 = 1;      //6  
        byte b3 = s2;           //7  
        byte b4 = c2;           //8  
    }  
}
```

What is the result of attempting to compile the program ?

- a. Compiler error at 1
- b. Compiler error at 2
- c. Compiler error at 3
- d. Compiler error at 4
- e. Compiler error at 5
- f. Compiler error at 6
- g. Compiler error at 7
- h. Compiler error at 8
- i. Runtime error.
- j. None of the Above

Question 2

```
1. interface I1 {}  
2. interface I2 {}  
3. class Base implements I1 {}  
4. class Sub extends Base implements I2 {}  
5.  
6. class Red {  
7.     public static void main(String args[]) {  
8.         Sub s1 = new Sub();  
9.         I2 i2 = s1;  
10.        I1 i1 = s1;  
11.        Base base = s1;  
12.        Sub s2 = (Sub)base;  
13.    }  
14. }
```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 9.
- b. Compiler error at line 10.
- c. Compiler error at line 11.
- d. Compiler error at line 12.
- e. Runtime error.

f. Compiles and runs without error.

Question 3

```
1. class Teal {  
2.     public static void main (String[] args) {  
3.         byte b=1;  
4.         long l=1000;  
5.         b += l;  
6.     }  
7. }
```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 3.
- b. Runtime error at line 3.
- c. Compiler error at line 4.
- d. Runtime error at line 4.
- e. Compiler error at line 5.
- f. Runtime error at line 5.
- g. Compiles and runs without error.

Question 4

```
class UltraViolet {  
    public static void main (String[] args) {  
        char a = '\u002a'; // Asterisk  
        char b = '\u0024'; // Dollar Sign  
        System.out.print(a + b);  
        System.out.print(" ABC" + a + b);  
    }  
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 78 ABC*\$
- b. Prints: *\$ ABC*\$
- c. Prints: 78 ABC78
- d. Compiler error.
- e. Runtime error.
- f. None of the Above

Question 5

```
1. class Purple {  
2.     public static void main (String []args) {  
3.         int[] i = {1,2,3};  
4.         Object obj = i;  
5.         i = obj;  
6.     }  
7. }
```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 2.
- b. Runtime error at line 2.
- c. Compiler error at line 3.
- d. Runtime error at line 3.
- e. Compiler error at line 4.
- f. Runtime error at line 4.
- g. Compiler error at line 5.

- h. Runtime error at line 5.
- i. Compiles and runs without error.

Question 6

```
class Sienna {
    static double a;
    static float b;
    static int c;
    static char d;
    public static void main(String[] args) {
        a = b = c = d = 'a';
        System.out.println(a+b+c+d == 4 * 'a');
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: true
- b. Prints: false
- c. Compiler error.
- d. Run time error.
- e. None of the above.

Question 7

```
1. class Black {
2.     public static void main(String args[]) {
3.         int[] i = {1,2,3,4,5};
4.         long[] l = new long[5];
5.         for (int j=0; j < l.length(); j++) {
6.             l[j] = i[j];
7.         }
8.     }
9. }
```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 3.
- b. Runtime error at line 3.
- c. Compiler error at line 4.
- d. Runtime error at line 4.
- e. Compiler error at line 5.
- f. Runtime error at line 5.
- g. Compiler error at line 6.
- h. Runtime error at line 6.
- i. Compiles and runs without error.

Question 8

```
class A {}
class B extends A {}
class C extends B {
    static void m(A x, A y) {System.out.print("AA");}
    static void m(A x, B y) {System.out.print("AB");}
    static void m(B x, A y) {System.out.print("BA");}
    static void m(B x, B y) {System.out.print("BB");}
    public static void main(String[] args) {
        C c = new C();
        m(c, c);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: AA
- b. Prints: AB
- c. Prints: BA
- d. Prints: BB
- e. Compiler error.
- f. Runtime error.
- g. None of the Above

Question 9

```
class A {}
class B extends A {}
class C extends B {
    static void m(A x, A y) {System.out.print("AA");}
    static void m(A x, B y) {System.out.print("AB");}
    static void m(B x, A y) {System.out.print("BA");}
    static void m(B x, B y) {System.out.print("BB");}
    public static void main(String[] args) {
        A a;
        B b;
        m(null,null);
        m(a=null,b=null);
        m(b, a);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: BBABAB
- b. Prints: BBABBA
- c. Prints: BBBBAB
- d. Prints: BBBBBA
- e. Prints: BBBBBB
- f. Compiler error.
- g. Runtime error.
- h. None of the Above

Question 10

```
1. class Amber {
2.     public static void main(String[] args) {
3.         int[][] a =
4.         {{1,2},{ 'a'-'a','b'-'a','c'-'a'},{~0,2&3,1<<1}};
5.         Object[] obj = (Object[])a.clone();
6.         for(int i = 0;i<obj.length; i++) {
7.             int[] ia = (int[])obj[i];
8.             System.out.print(ia[i]);
9.         }
10. }
```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 3.
- b. Compiler error at line 4.
- c. Compiler error at line 5.
- d. Compiler error at line 6.
- e. Compiler error at line 7.
- f. Run time error.
- g. None of the above.

No.

정답 - 해설

1 c d

Compiler error at 3 Compiler error at 4

The compiler will implicitly do a narrowing conversion for an assignment statement if the right hand operand is a compile time constant of type byte, short, char, or int and the value falls within the range of the variable on the left and if the variable is of type byte, short, or char. In this case, variables s1 and c1 are not compile time constants. However, variables s2 and c2 are compile time constants.

2 f

Compiles and runs without error.

Although line 12 appears to be the source of a compiler error it is not. In line 12, the reference named base actually refers to an object of type Sub. Therefore, the reference to the object may be cast to type Sub.

3 g

Compiles and runs without error.

The compound assignment operators include an implicit cast to the type of the left hand operand. Therefore, an explicit cast is not necessary.

4 a

Prints: 78 ABC*\$

When char variables a and b are promoted to String types they are printed as *\$. When not promoted to String types they are printed as the numeric sum of 0x2a and 0x24. Note: 0x2a is decimal 42 and 0x24 is decimal 36.

5 g

Compiler error at line 5.

Although the referenced object is indeed an array of type int, an explicit cast is necessary to cast the obj reference to an int array.

6 a

Prints: true

A char type literal or variable can be cast to a numeric value.

7 e

Compiler error at line 5.

The length attribute of the array is accessed as though it were a method.

8 d

Prints: BB

Section 15.12.2.2 of the Java Language Specification states the following.

If more than one method declaration is both accessible and applicable to a method invocation, it is necessary to choose one to provide the descriptor for the run-time method dispatch.

The Java programming language uses the rule that the most specific method is chosen.

The informal intuition is that one method declaration is

more specific than another

if any invocation handled by the first method could be passed on to the other one without a compile-time type error.

End of quote. In this case, m(B b, B b) is more specific than the others.

9 b

Prints: BBABBA

The first method call, m(null,null), calls method m with two null literals that have type null. Both null literals are promoted to type B because B is more specific than type A. As a result, m(B x, B y) is invoked. The second method call, m(a=null,b=null), calls m with a first parameter of type A and a second parameter of type B. As a result, m(A x, B y) is invoked because the parameter types are matched exactly. The third method call invokes m(B x, A y) because the parameter types match exactly.

10 g

None of the above.

The program compiles and runs without error and prints 112. (a[0][0]=1, a[1][1]='b'-'a'=1, a[2][2]=1<<1=2) It is necessary to remember that all arrays are objects and therefore are clonable. Furthermore, a two dimensional array is also a single dimensional array of single dimensional arrays.

Question 11

class A {

public static void main(String[] args) {

final short s1 = 1; // 1

final char c1 = 1; // 2

byte b1 = s1; // 3

byte b2 = c1; // 4

byte b3 = 1; // 5

byte b4 = 1L; // 6

byte b5 = 1.0; // 7

byte b6 = 1.0d; // 8

}

}

What is the result of attempting to compile the program ?

- a. Compiler error at 1
- b. Compiler error at 2
- c. Compiler error at 3
- d. Compiler error at 4
- e. Compiler error at 5
- f. Compiler error at 6
- g. Compiler error at 7
- h. Compiler error at 8
- i. Runtime error.
- j. None of the Above

Question 12

class A {}

class B extends A {}

class C extends B {

static void m(A x, A y) {System.out.print("AA");}

static void m(A x, B y) {System.out.print("AB");}

```

static void m(B x, A y) {System.out.print("BA");}
static void m(B x, B y) {System.out.print("BB");}
static void m(A x, C y) {System.out.print("AC");}
public static void main(String[] args) {
    C c = new C();
    m(c, c);
}
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|-------------------|----------------------|
| a. Prints: AA | b. Prints: AB |
| c. Prints: BA | d. Prints: BB |
| e. Prints: AC | f. Compiler error. |
| g. Runtime error. | h. None of the Above |

Question 13

```

class Fuchsia {
    public static void main (String[] args) {
        System.out.println((short)Integer.MIN_VALUE +
            ", " + (short)Integer.MAX_VALUE +
            ", " + (int)(char)Integer.MIN_VALUE +
            ", " + (int)(char)Integer.MAX_VALUE);
    }
}

```

What is the result of attempting to compile and run the above program ?

- Prints: -32768, 32767, -32768, 32767
- Prints: -32768, 32767, 0, 65535
- Prints: 0, 0, 0, 0
- Prints: 0, -1, 0, 65535
- Compiler error.
- Run time error.

Question 14

```

1. class Green {
2.     public static void main (String args[]) {
3.         int[] i = null;
4.         Cloneable c = i;
5.         i = (int [])c;
6.     }
7. }

```

What is the result of attempting to compile and run the above program ?

- Compiler error at line 3.
- Runtime error at line 3.
- Compiler error at line 4.
- Runtime error at line 4.
- Compiler error at line 5.
- Runtime error at line 5.
- Compiles and runs without error.

Question 15

```

1. class White {
2.     public static void main(String args[]) {
3.         int[] i = {1,2,3,4,5};
4.         long[] l1 = new long[5];
5.         long []l2 = l1;
6.         long l3[] = (long[])i;
7.         long l4[] = new long[5];

```

```

8.         l4[1] = i[1];
9.     }
10. }

```

What is the result of attempting to compile and run the above program ?

- Compiler error at line 3.
- Runtime error at line 3.
- Compiler error at line 4.
- Runtime error at line 4.
- Compiler error at line 5.
- Runtime error at line 5.
- Compiler error at line 6.
- Runtime error at line 6.
- Compiler error at line 7.
- Runtime error at line 7.
- Compiler error at line 8.
- Runtime error at line 8.
- Compiles and runs without error.

Question 16

```

class Chartreuse {
    public static void main (String[] args) {
        short i = (short)Float.NEGATIVE_INFINITY;
        short j = (short)Float.POSITIVE_INFINITY;
        short k = (short)Float.NaN;
        System.out.println((i == Short.MIN_VALUE) +
            ", " + (j == Short.MAX_VALUE) +
            ", " + (k == 0));
    }
}

```

What is the result of attempting to compile and run the above program ?

- Prints: false, false, false
- Prints: false, false, true
- Prints: true, true, false
- Prints: true, true, true
- Compiler error.
- Run time error.

Question 17

```

import java.io.Serializable;
1. class Blue {
2.     public static void main (String args[]) {
3.         int[] i = {1,2,3};
4.         Serializable s = i;
5.         i = (int [])s;
6.     }
7. }

```

What is the result of attempting to compile and run the above program ?

- Compiler error at line 3.
- Runtime error at line 3.
- Compiler error at line 4.
- Runtime error at line 4.
- Compiler error at line 5.
- Runtime error at line 5.
- Compiles and runs without error.

Question 18

```

class Blue {
    public static void main (String[] args) {
        int i1 = (Integer.MAX_VALUE + 2);
        int i2 = (Integer.MIN_VALUE + 1);
        int i3 = (Integer.MIN_VALUE + 2);
        int i4 = (Integer.MIN_VALUE + 3);
        System.out.println((i1==i2)+","+(i1      ==i3)+","+(i1
==i4));
    }
}

```

What is the result of attempting to compile and run the above program ?

- Prints: false,false,false
- Prints: false,false,true
- Prints: false,true,false
- Prints: true,false,false
- Compiler error.
- Runtime error.
- None of the Above

Question 19

```

1. interface I1 {}
2. interface I2 {}
3. class Base implements I1 {}
4. class Sub extends Base implements I2 {}
5. class Gray {
6.     public static void main(String []args) {
7.         Base[] base = {new Base()};
8.         Sub sub[] = {new Sub()};
9.         Object obj = sub;
10.        base = obj;
11.    }
12. }

```

What is the result of attempting to compile and run the above program ?

- Compiler error at line 7.
- Runtime error at line 7.
- Compiler error at line 8.
- Runtime error at line 8.
- Compiler error at line 9.
- Runtime error at line 9.
- Compiler error at line 10.
- Runtime error at line 10.
- Compiles and runs without error.

Question 20

```

1. class Primitives {
2.         static void printFloat(float f)
{System.out.println(f);}
3.         static void printDouble(double d)
{System.out.println(d);}
4.     public static void main(String[] args) {
5.         byte b = 1;
6.         short s = b;
7.         char c = s;
8.         int i = c;
9.         long l = i;
10.        float f = l;
11.        printFloat(i);
12.        printFloat(l);

```

```

13.        printDouble(l);
14.    }
15. }

```

What is the result of attempting to compile and run the above program ?

- Compiler error at line 3.
- Compiler error at line 4.
- Compiler error at line 5.
- Compiler error at line 6.
- Compiler error at line 7.
- Compiler error at line 8.
- Compiler error at line 9.
- Compiler error at line 10.
- Compiler error at line 11.
- Compiler error at line 12.
- Compiler error at line 13.
- Compiles and runs without error.

정답 - 해설

11 f g h

Compiler error at 6 Compiler error at 7 Compiler error at 8

The compiler will implicitly do a narrowing conversion for an assignment statement if the right hand operand is a compile time constant of type byte, short, char, or int and the value falls within the range of the variable on the left and if the variable is of type byte, short, or char.

12 f

Compiler error.

Section 15.12.2.2 of the Java Language Specification states the following. If more than one method declaration is both accessible and applicable to a method invocation, it is necessary to choose one to provide the descriptor for the run-time method dispatch. The Java programming language uses the rule that the most specific method is chosen. The informal intuition is that one method declaration is more specific than another if any invocation handled by the first method could be passed on to the other one without a compile-time type error. End of quote. In this case, no method is clearly more specific than all of the others.

13 d

Prints: 0, -1, 0, 65535

The most negative value of type int is represented as 0x80000000 and becomes 0x0000 when converted to a short or char. The most positive value of type int is represented as 0x7FFFFFFF and becomes 0xFFFF when converted to a short or char.

14 g

Compiles and runs without error.

The null literal is converted to an int array type. All array types implement the Cloneable interface and may therefore be assigned to a reference of type Cloneable. The int array object referenced by the Cloneable reference c can be assigned to a reference

of the int array type.

15 g

Compiler error at line 6.

An array of primitive types can not be cast to an array of a different primitive type.

16 b

Prints: false, false, true

Float values are converted to an integer before being converted to a short. The most negative value of type int is represented as 0x80000000 and becomes 0x0000 when converted to a short. The most positive value of type int is represented as 0x7FFFFFFF and becomes 0xFFFF when converted to a short. NaN is converted to zero.

17 g

Compiles and runs without error.

All array types implement the Serializable interface and may be assigned to a reference of type Serializable.

18 d

Prints: true,false,false

When two is added to Integer.MAX_VALUE the result is an integer with the most significant bit set to one and the least significant bit set to one and all the bits in between set to zero. The same is true for the two's complement representation of the most negative integer plus one.

19 g

Compiler error at line 10.

The reference "base" is of type "Base[]", therefore an explicit cast to type "Base[]" is required.

20 e

Compiler error at line 7.

Short is signed and char is not signed so an explicit cast is necessary when a short is assigned to a char and vice versa.

Question 21

```
1. interface I1 {}
2. interface I2 {}
3. class Base implements I1 {}
4. class Sub extends Base implements I2 {}
5. class Silver {
6.     public static void main(String []args) {
7.         Base[] base = {new Base()};
8.         Sub sub[] = new Sub[1];
9.         Object obj = base;
10.        sub = (Sub[])obj;
11.        I1 []i1 = (I1[])obj;
12.    }
13. }
```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 8.
- b. Runtime error at line 8.

- c. Compiler error at line 9.
- d. Runtime error at line 9.
- e. Compiler error at line 10.
- f. Runtime error at line 10.
- g. Compiler error at line 11.
- h. Runtime error at line 11.
- i. Compiles and runs without error.

Question 22

```
class A {}
class B {
    static void m(Object x) {System.out.print("Object");}
    static void m(String x) {System.out.print("String");}
    public static void main(String[] args) {
        m(null);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: Object
- b. Prints: String
- c. Compiler error.
- d. Runtime error.
- e. None of the Above

Question 23

```
1. interface I1 {}
2. interface I2 {}
3. class Base implements I1 {}
4. class Sub extends Base implements I2 {}
5. class Yellow {
6.     public static void main(String args[]) {
7.         Base base = new Sub();
8.         I1 i1 = base;
9.         Sub sub = (Sub)base;
10.        I2 i2 = (Sub)base;
11.    }
12. }
```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 7.
- b. Runtime error at line 7.
- c. Compiler error at line 8.
- d. Runtime error at line 8.
- e. Compiler error at line 9.
- f. Runtime error at line 9.
- g. Compiler error at line 10.
- h. Runtime error at line 10.
- i. Compiles and runs without error.

Question 24

```
class A {}
class B extends A {}
class C {
    void m(A x) {System.out.print("A");}
    void m(B x) {System.out.print("B");}
    public static void main(String[] args) {
        C c = new C();
        c.m(null);
    }
}
```

```
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: A
- b. Prints: B
- c. Compiler error.
- d. Runtime error.
- e. None of the Above

Question 25

```
class Blue {
    public static void main (String[] args) {
        byte b1 = (byte)(2 * Byte.MAX_VALUE);
        byte b2 = -1;
        byte b3 = -2;
        byte b4 = -3;

        System.out.println((b1==b2)+" "+(b1==b3)+" "+(b1==b4))
        ;
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: true,false,false
- e. Compiler error.
- f. Runtime error.
- g. None of the Above

Question 26

```
1. interface I1 {}
2. interface I2 {}
3. class Base implements I1 {}
4. class Sub extends Base implements I2 {}
5.
6. class Orange {
7.     public static void main(String args[]) {
8.         Base base = new Base();
9.         I1 i1 = base;
10.        Sub sub = (Sub)base;
11.    }
12. }
```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 9.
- b. Runtime error at line 9.
- c. Compiler error at line 10.
- d. Runtime error at line 10.
- e. Compiles and runs without error.

Question 27

```
class Beige {
    public static void main (String[] args) {
        int i = (int)Float.NEGATIVE_INFINITY;
        int j = (int)Float.POSITIVE_INFINITY;
        int k = (int)Float.NaN;
        System.out.println((i == Integer.MIN_VALUE) +
            ", " + (j == Integer.MAX_VALUE) +
            ", " + (k == 0));
    }
}
```

```
}
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: false, false, false
- b. Prints: false, false, true
- c. Prints: true, true, false
- d. Prints: true, true, true
- e. Compiler error.
- f. Run time error.

Question 28

```
class A {}
class B {
    static void m(Object x) {System.out.print("Object");}
    static void m(String x) {System.out.print("String");}
    static void m(A x) {System.out.print("A");}
    public static void main(String[] args) {
        m(null);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: Object
- b. Prints: String
- c. Compiler error.
- d. Runtime error.
- e. None of the Above

Question 29

```
1. class Maroon {
2.     public static void main (String[] args) {
3.         int i=1;
4.         short s = 1;
5.         long l=1,m=2;
6.         i = l + i;
7.         l = s + i;
8.     }
9. }
```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 4.
- b. Runtime error at line 4.
- c. Compiler error at line 5.
- d. Runtime error at line 5.
- e. Compiler error at line 6.
- f. Runtime error at line 6.
- g. Compiler error at line 7.
- h. Runtime error at line 7.
- i. Compiles and runs without error.

No.

정답 - 해설

21 f

Runtime error at line 10.

The object referenced by obj is of type "Base[]", and can not be cast to type "Sub[]".

22 b

Prints: String

Section 15.12.2.2 of the Java Language Specification states the following. If more than one method declaration is both accessible and applicable to a method invocation, it is necessary to choose one to provide the descriptor for the run-time method dispatch. The Java programming language uses the rule that the most specific method is chosen. The informal intuition is that one method declaration is more specific than another if any invocation handled by the first method could be passed on to the other one without a compile-time type error. End of quote. In this case, the String type is more specific than the Object type.

23 i

Compiles and runs without error.

Although the reference named base is of type Base, the object that it refers to is of type Sub. Therefore, the object may be cast to type Sub. Since objects of type Sub implement both interfaces, I1 and I2, the Sub type objects can be assigned to references of type I1 and I2 without an explicit cast.

24 b

Prints: B

Section 15.12.2.2 of the Java Language Specification states the following. If more than one method declaration is both accessible and applicable to a method invocation, it is necessary to choose one to provide the descriptor for the run-time method dispatch. The Java programming language uses the rule that the most specific method is chosen. The informal intuition is that one method declaration is more specific than another if any invocation handled by the first method could be passed on to the other one without a compile-time type error. End of quote. In this case, type B is more specific than type A.

25 c

Prints: false,true,false

When Byte.MAX_VALUE is multiplied by two, the result is a byte with the most significant seven bits set to one and the least significant bit set to zero. The two's complement representation of negative two is the same.

26 d

Runtime error at line 10.

The compiler accepts the explicit cast at line 12. However, Base is not a subclass of Sub; therefore, a runtime exception is thrown.

27 d

Prints: true, true, true

Negative infinity is rounded to the most negative value of type int or long. Positive infinity is rounded to the most positive value of type int or long. NaN is converted to zero.

28 c

Compiler error.

Section 15.12.2.2 of the Java Language Specification states the following. If more than one method declaration is both accessible and applicable to a method invocation, it is necessary to choose one to provide the descriptor for the run-time method dispatch. The Java programming language uses the rule that the most specific method is chosen. The informal intuition is that one method declaration is more specific than another if any invocation handled by the first method could be passed on to the other one without a compile-time type error. End of quote. In this case, both the String class and class A are more specific than type Object. The ambiguity generates a compiler error.

29 e

Compiler error at line 6.

An explicit cast to type int is required.

[CONSTRUCTORS]

Question 1

```
class C {
    {System.out.print("1");}
    static {System.out.print("2");}
    C() {System.out.print("3");}
    String m() {return "4";}
}
class D {
    {System.out.print("5");}
    static {System.out.print("6");}
    D() {System.out.print("7");}
    public static void main(String[] args) {
        System.out.print("8");
        System.out.print(new C().m());
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 65782134
- b. Prints: 681234
- c. Prints: 682134
- d. Prints: 68572134
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 2

```
class D {
    {System.out.print("1");}
    static {System.out.print("2");}
    D() {System.out.print("3");}
    D(String s) {System.out.print("4");}
}
class E extends D {
    {System.out.print("5");}
    static {System.out.print("6");}
    E(String s) {System.out.print("7");super(s);}
}
class F {
    public static void main(String[] args) {
```



```

        new E("e");
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 261457 b. Prints: 261357
- c. Prints: 267315 d. Prints: 267415
- e. Compiler Error f. Runtime Error
- g. None of the Above

Question 3

```

class D {
    {System.out.print("1");}
    static {System.out.print("2");}
    D() {System.out.print("3");}
    D(String s) {System.out.print("4");}
}
class E extends D {
    {System.out.print("5");}
    static {System.out.print("6");}
    E() {System.out.print("7");this("e");}
    E(String s) {System.out.print("8");}
}
class F {
    public static void main(String[] args) {
        new E();
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 2613587 b. Prints: 2614587
- c. Prints: 2671358 d. Prints: 2678315
- e. Compiler Error f. Runtime Error
- g. None of the Above

Question 4

```

class A {
    static {System.out.print("1");}
    {System.out.print("2");}
    A() {System.out.print("3");}
    static String m() {return "4";}
}
class B {
    static {System.out.print("5");}
    {System.out.print("6");}
    B() {System.out.print("7");}
    public static void main(String[] args) {
        System.out.print("8");
        System.out.print(A.m());
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 5814 b. Prints: 15814
- c. Prints: 581234 d. Prints: 58123674
- e. Compiler Error f. Runtime Error
- g. None of the Above

Question 5

```

class P {void m1() throws Exception {}}
class Q {
    {System.out.print("1");}
    static {System.out.print("2");}
    Q() {System.out.print("3");}
}
class R extends Q {
    {System.out.print("4"); new P().m1();}
    static {System.out.print("5");}
    R() {System.out.print("6");}
}
class S {
    public static void main(String[] args) {
        new R();
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 251346 b. Prints: 256314
- c. Prints: 123456 d. Prints: 142356
- e. Compiler Error f. Runtime Error
- g. None of the Above

Question 6

```

class Q {
    int i = 1;
    {System.out.print("1");}
    static {System.out.print("2");}
    Q() {System.out.print("3");}
    Q(int x) {System.out.print("4");}
}
class R extends Q {
    int j = 2;
    {System.out.print("5");}
    static {System.out.print("6");}
    R() {super(j);System.out.print("7");}
}
class S {
    public static void main(String[] args) {
        new R();
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 261357 b. Prints: 261457
- c. Prints: 267315 d. Prints: 267415
- e. Compiler Error f. Runtime Error
- g. None of the Above

Question 7

```

class T {
    {System.out.print("1");}
    static {System.out.print("2");}
    T() {System.out.print("3");}
}
class U extends T {
    {System.out.print("4");}
    static {System.out.print("5");}
    U() {System.out.print("6");}
}

```

```
class V {
    public static void main(String[] args) {
        new U();
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 36
- b. Prints: 1346
- c. Prints: 123456
- d. Prints: 213546
- e. Prints: 251346
- f. Prints: 251436
- g. Compiler Error
- h. Runtime Error
- i. None of the Above

Question 8

```
class E {
    {System.out.print("1");}
    static {System.out.print("2");}
    E() {System.out.print("3");}
    String m() {return "4";}
}
class F {
    {System.out.print("5");}
    static {System.out.print("6");}
    F() {System.out.print("7");}
    public static void main(String[] args) {
        System.out.print("8");
        System.out.print(new E().m());
        System.out.print(new E().m());
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 82134134
- b. Prints: 682134134
- c. Prints: 6821342134
- d. Prints: 657821342134
- e. Prints:
- f. Compiler Error
- g. Runtime Error
- h. None of the Above

Question 9

```
class Q {
    {System.out.print("1");}
    static {System.out.print("2");}
    Q() {System.out.print("3");}
}
class R extends Q {
    {System.out.print("4");}
    static {System.out.print("5");}
    R() {System.out.print("6");}
    static String m1() {return "7";}
}
class S {
    public static void main(String[] args) {
        System.out.print(R.m1());
    }
}
```

What is the result of attempting to compile and run

the program ?

- a. Prints: 7
- b. Prints: 57
- c. Prints: 257
- d. Prints: 527
- e. Prints: 2135467
- f. Prints: 2514367
- g. Prints: 2513467
- h. Compiler Error
- i. Runtime Error
- j. None of the Above

Question 10

```
class A {
    {System.out.print("1");}
    static {System.out.print("2");}
    A(String s) {System.out.print("3");}
}
class B extends A {
    {System.out.print("4");}
    static {System.out.print("5");}
    B(String s) {System.out.print("6");}
}
class C {
    public static void main(String[] args) {
        new B("b");
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 251346
- b. Prints: 256314
- c. Prints: 142346
- d. Prints: 123456
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 11

```
class A {
    {System.out.print("1");}
    static {System.out.print("2");}
    A(String s) {System.out.print("3");}
}
class B extends A {
    {System.out.print("4");}
    static {System.out.print("5");}
    B(String s) {super(s);System.out.print("6");}
}
class C {
    public static void main(String[] args) {
        new B("b");
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 251346
- b. Prints: 256314
- c. Prints: 142346
- d. Prints: 123456
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 12

```
class M {
    {System.out.print("1");}
    static {System.out.print("2");}
    M() {System.out.print("3");}
    M(String s) {System.out.print("4");}
}
```

```

class N extends M {
    {System.out.print("5");}
    static {System.out.print("6");}
    N() {this("@"); System.out.print("7");}
    N(String s) {System.out.print("8");}
}
class P {
    public static void main(String[] args) {
        new N();
    }
}

```

What is the result of attempting to compile and run the program ?

- | | |
|--------------------|----------------------|
| a. Prints: 2613578 | b. Prints: 2613587 |
| c. Prints: 261357 | d. Prints: 2614587 |
| e. Prints: 2678315 | f. Compiler Error |
| g. Runtime Error | h. None of the Above |

Question 13

```

class J {
    {System.out.print("1");}
    static {System.out.print("2");}
    J() {System.out.print("3");}
    J(String s) {System.out.print("4");}
}
class K extends J {
    {System.out.print("5");}
    static {System.out.print("6");}
    K() {System.out.print("7");}
    K(String s) {System.out.print("8");}
}
class L {
    public static void main(String[] args) {
        new K("!");
    }
}

```

What is the result of attempting to compile and run the program ?

- | | |
|-------------------|----------------------|
| a. Prints: 261458 | b. Prints: 261358 |
| c. Prints: 261357 | d. Prints: 268135 |
| e. Prints: 268145 | f. Compiler Error |
| g. Runtime Error | h. None of the Above |

Question 14

```

class G {
    {System.out.print("1");}
    static {System.out.print("2");}
    G() {System.out.print("3");}
    String m1() {return "4";}
    static String m2() {return "5";}
}
class H extends G {
    {System.out.print("6");}
    static {System.out.print("7");}
    H() {System.out.print("8");}
    String m1() {return "9";}
    static String m2() {return "A";}
}
class I {
    public static void main(String[] args) {

```

```

        System.out.print(H.m2());
        System.out.print(new H().m1());
    }
}

```

What is the result of attempting to compile and run the program ?

- | | |
|----------------------|-----------------------|
| a. Prints: 7A689 | b. Prints: 271368A9 |
| c. Prints: 27A13689 | d. Prints: 27A2713689 |
| e. Compiler Error | f. Runtime Error |
| g. None of the Above | |

No.

정답 - 해설

Remark

1 c

Prints: 682134

Class D is loaded and the static initializer is processed before the main method begins to run. No instance of class D is created, therefore, the instance initializer and constructor do not run. The main method invokes method "m" of class C. As a result, C is loaded and the static initializer runs. A new instance of C is being created, so the constructor for class C is invoked. However, the instance initializer runs to completion before the body of the constructor runs.

2 e

Compiler Error

The constructor for class E contains an explicit call to the super class constructor. Since the super class constructor call is not the first statement of the constructor a compiler error is generated.

3 e

Compiler Error

The no parameter constructor for class E contains an explicit call to the single parameter constructor. A compiler error is generated because the explicit constructor call is not the first statement of the constructor.

4 a

Prints: 5814

Class B is loaded and the static initializer is processed before the main method begins to run. The use of the A.m() method causes Class A to be loaded and the static initializer to be processed before method A.m() is processed. No instance of either class is created, so no constructor or instance initializer runs.

5 e

Compiler Error

The instance initializer of class R calls P.m1. Since P.m1 may throw an exception, that exception or a super class of the exception must be declared in the throws clause of every constructor of class R. Note: Unlike a named class, an instance initializer of an anonymous class may throw any exception.

6 e

Compiler Error

The constructor of R has an explicit call to the super class constructor that includes an instance variable as a parameter. A compiler error is generated since the instance of class R has not yet been fully initialized at the point in time when the super class constructor is invoked. The instance variable R.j still contains the default value of zero rather than the assigned value of 2.

7 e

Prints: 251346

The main method of class V begins to run and calls for the creation of a new instance of U. Before the new instance can be created, class U and its super class T are loaded and the static initializers of the super class and the sub class are processed. If any static variables had been declared, they would also have been initialized. After the classes are loaded and initialized, the constructor of class U begins to run. The first statement is an implicit call to the super class constructor. The call to the super class constructor has been added by the compiler, since it does not appear in the source code. Before the body of the super class constructor is processed, the instance initializer will run to completion. If any instance variables had been declared, then they would also be initialized before the body of the super class constructor is processed. Control then returns to the sub class constructor and the sub class instance initializer runs to completion before the body of the constructor is processed.

8 b

Prints: 682134134

Class F is loaded and the static initializer is processed before the main method begins to run. No instance of class F is created, therefore, the instance initializer and constructor do not run. The main method invokes method "m" of class E. As a result, E is loaded and the static initializer runs. A new instance of E is being created, so the constructor for class E is invoked causing the instance initializer to run to completion before the body of the constructor is processed. Method E.m then runs. A second instance of Class E is created causing the constructor for class E and the instance initializer for class E to run again followed by method m. Of course, the constructor of the Object class is always called each time a new instance of any Object is created.

9 c

Prints: 257

The main method of class S begins to run and calls R.m1. Before m1 can run, class R and its super class Q are loaded and the static initializers of the super class and the sub class are processed. If any static variables had been declared, they would also have been initialized. After the classes are loaded and initialized, R.m1 runs. Since no new instance of class Q or R were created, the constructors and the

instance initializers were not processed.

10 e

Compiler Error

The single parameter constructor for class B does not have an explicit call to the super class constructor. Therefore, the compiler tries to add an implicit call to the no parameter constructor of the super class. Since the super class has a declaration for a single parameter constructor, the compiler does not implicitly create a no parameter construct. A compiler error is the result.

11 a

Prints: 251346

The main method of class C begins to run and calls for the creation of a new instance of class B. Before the new instance can be created, class B and its super class A must be loaded and the static initializers of the super class and the sub class are processed. If any static variables had been declared, they would also have been initialized. After the classes are loaded and initialized, the instance creation process begins. The single parameter constructor has as its first statement a call to the single parameter super class constructor. Similarly, the super class constructor has as its first statement an implicit call to the no parameter constructor of the Object class. Once the Object class constructor runs to completion, control returns to the constructor of class A. Before the body of the class A constructor is processed, the instance initializer of class A runs to completion. After the body of the class A constructor runs to completion control returns to the constructor of class B. The instance initializer of class B runs to completion before the body of the constructor is processed.

12 b

Prints: 2613587

The main method of class P begins to run and calls for the creation of a new instance of class N. Before the new instance can be created, class N and its super class M are loaded and the static initializers of the super class and the sub class are processed. If any static variables had been declared, they would also have been initialized. After the classes are loaded and initialized, the instance creation process begins. The no parameter constructor has as its first statement a call to the single parameter constructor. The single parameter constructor has as its first statement an implicit call to the no parameter superclass constructor. Since the call to the super class constructor does not appear in the source code, the compiler adds it implicitly. Before the body of the super class constructor is processed the constructor of the Object class is called and runs to completion. Afterward, the instance initializer of class M runs to completion. If any instance variables had been declared, then they would also be initialized. After the body of the super class constructor runs to completion control returns to the sub class constructor. Before the body of the sub class

constructor runs, the instance initializer runs to completion.

13 b

Prints: 261358

The main method of class L begins to run and calls for the creation of a new instance of class K. Before the new instance can be created, class K and its super class J must be loaded and the static initializers of the super class and the sub class are processed. If any static variables had been declared, they would also have been initialized. After the classes are loaded and initialized, the instance creation process begins. The single parameter constructor has as its first implicit statement a call to the no parameter super class constructor. The implicit statement is added by the compiler even if it does not appear in the source code. Another implicit feature of a constructor is the fact that all instance variables are initialized and all instance initializer blocks are processed before the body of the constructor is processed.

14 c

Prints: 27A13689

The main method of class I begins to run and calls H.m2. Before the m2 method can run, class H and its super class G are loaded and the static initializers of the super class and the sub class are processed. If any static variables had been declared, they would also have been initialized. After the classes are loaded and initialized, H.m2 runs. Since no new instance of class H or G were created, the constructors and the instance initializers were not yet processed. The next line of I.main creates a new instance of H and calls H.m1. The instance creation process begins with a call to the constructor of H. The first line of the construct is an implicit call to the super class construct. The call to the super class constructor does not appear in the source code, so the compiler added it implicitly. Before the body of the super class constructor runs the instance initializer is processed. If any instance variables had been declared, then they would also have been initialized. The body of the super class constructor is then processed before control returns to the sub class constructor. The instance initializer of the sub class is processed before the body of the sub class. Following the instance creation process, the method H.m1 runs.

[Control Flow]

Question 1

```
class Red {
    public static void main(String args[]) {
        for (int i = 0; i<5 ;i++) {
            switch(i) {
                case 0: System.out.print("v ");break;
                case 1: System.out.print("w ");
                case 2: System.out.print("x ");break;
                case 3: System.out.print("y ");
                case 4: System.out.print("z ");break;
```

```
                default: System.out.print("d ");
            }
        }
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: v w x y z
- b. Prints: v w x y z d
- c. Prints: v w x x y z z
- d. Prints: v w w x y y z d
- e. Prints: d d d d d d
- f. Runtime Exception
- g. Compiler Error
- h. None of the Above

Question 2

```
class Purple {
    public static void main(String args[]) {
        int x = 6;
        int success = 0;
        do {
            switch(x) {
                case 0: System.out.print("0");
                    x += 5; break;
                case 1: System.out.print("1");
                    x += 3; break;
                case 2: System.out.print("2");
                    x += 1; break;
                case 3: System.out.print("3");
                    success++; break;
                case 4: System.out.print("4");
                    x -= 1; break;
                case 5: System.out.print("5");
                    x -= 4; break;
                case 6: System.out.print("6");
                    x -= 5; break;
                default: System.out.print("Default ");
            }
        } while ((x != 3) || (success < 2));
    }
}
```

What is the result of attempting to compile and run the above program ?

- | | |
|--------------------|----------------------|
| a. Prints: 0123456 | b. Prints: 60514233 |
| c. Prints: 6152433 | d. Prints: 61433 |
| e. Prints: 6143 | f. Prints: 6 |
| g. Prints: Default | h. Runtime Exception |
| i. Compiler Error | j. None of the Above |

Question 3

```
class Yellow {
    public static void main (String args[]) {
        char c = 'c';
        switch(c) {
            case 'a':
                System.out.print("1 ");
            case 'b':
                System.out.print("2 ");
            case 'c':
```

```

        System.out.print("3 ");
    default:
        System.out.print("4 ");
    }
}
}

```

What is the result of attempting to compile and run the above program ? Select only one Answer.

- a. Prints: 1
- b. Prints: 2
- c. Prints: 3
- d. Prints: 4
- e. Runtime error
- f. Compiler error
- g. None of the above

Question 4

```

class Blue {
    public static void main(String args[]) {
        long[] l = {1, -1, 3, 4};
        for (int i = 0; i<l.length;i++) {
            switch(l[i]) {
                case 2-1: System.out.print("v ");
                break;
                case 'd'-'a': System.out.print("w ");
                break;
                case 0: System.out.print("x ");
                break;
                case ~0: System.out.print("y ");
                break;
                case 4&5: System.out.print("z ");
                break;
                default: System.out.print("Default ");
            }
        }
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: v w x y
- b. Prints: v w x y z Default
- c. Prints: v y w z
- d. Prints: Default Default Default Default
- e. Runtime Exception
- f. Compiler Error
- g. None of the Above

Question 5

```

class Violet {
    public static void main(String args[]) {
        int x = -5;
        int success = 0;
        do {
            switch(x) {
                case 0: System.out.print("0");
                x += 5; break;
                case 1: System.out.print("1");
                x += 3; break;
                case 2: System.out.print("2");
                x += 1; break;
                case 3: System.out.print("3");
                success++; break;
                case 4: System.out.print("4");

```

```

                x -= 1; break;
            case 5: System.out.print("5");
                x -= 4; break;
            case 6: System.out.print("6");
                x -= 5; break;
            default:
                x += x < 0 ? 2 : -2;
        }
    } while ((x != 3) || (success < 2));
}
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: 0123456
- b. Prints: 60514233
- c. Prints: 1433
- d. Prints: 61433
- e. Prints: 051433
- f. Prints: 33
- g. Runtime Exception
- h. Compiler Error
- i. None of the Above

Question 6

```

class Green {
    public static void main(String args[]) {
        short[] s = {1, -1, 3, 4};
        for (int i = 0; i<s.length;i++) {
            switch(s[i]) {
                case 2-1: System.out.print("v ");
                break;
                case 'd'-'a': System.out.print("w ");
                break;
                case 0: System.out.print("x ");
                break;
                case ~0: System.out.print("y ");
                break;
                case 4&5: System.out.print("z ");
                break;
                default: System.out.print("Default ");
            }
        }
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: v w x y
- b. Prints: v w x y z Default
- c. Prints: v y w z
- d. Prints: Default Default Default Default
- e. Runtime Exception
- f. Compiler Error
- g. None of the Above

Question 7

```

class Orange {
    public static void main(String args[]) {
        byte b = -1;
        switch(b) {
            case 0:
                System.out.print("zero ");
                break;
            case 100:
                System.out.print("100 ");

```

```

        break;
    case 1000:
        System.out.print("1000 ");
        break;
    default: System.out.print("Default ");
    }
}
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|----------------------|--------------------|
| a. Prints: zero | b. Prints: 100 |
| c. Prints: 1000 | d. Prints: Default |
| e. Runtime error | f. Compiler error |
| g. None of the above | |

Question 8

```

class Brown {
    public static void main (String args[]) {
        int i = 0;
        int j = 0;
label1:
        while (1) {
label2:
            for (;;) {
label3:
                do {
                    System.out.print(i + j);
                    switch (i+j++) {
                        case 0: continue label3;
                        case 1: continue label2;
                        case 2: continue label1;
                        case 3: break;
                        case 4: break label3;
                        case 5: break label2;
                        case 6: break label1;
                        default: break label1;
                    }
                } while (++j<5);
            }
        }
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|-------------------|----------------------|
| a. Prints: 1357 | b. Prints: 02 |
| c. Prints: 02356 | d. Prints: 01346 |
| e. Prints: 0236 | f. Runtime Exception |
| g. Compiler Error | h. None of the Above |

Question 9

```

class Black {
    public static void main (String args[]) {
        int i = 0;
        int j = 0;
label1:
        while (i++<5) {
label2:
            for (;;) {
label3:
                do {

```

```

        System.out.print(i + j);
        switch (i+j++) {
            case 0: continue label3;
            case 1: continue label2;
            case 2: continue label1;
            case 3: break;
            case 4: break label3;
            case 5: break label2;
            case 6: break label1;
            default: break label1;
        }
    } while (++j<5);
}
}
}
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|-------------------|----------------------|
| a. Prints: 12457 | b. Prints: 02357 |
| c. Prints: 02356 | d. Prints: 1357 |
| e. Prints: 1356 | f. Runtime Exception |
| g. Compiler Error | h. None of the Above |

Question 10

```

class A {
    public static void main (String[] args) {
        int i = 0,j = 0,m = 0,n = 0,p;
label1:
        try {
            do {
                m++;
                try {
                    p = i + j + m + n;
                    if (p>=5) break label1;
                    n++;
                } finally {i++;}
            } while (m++ < 2);
        } finally {j++;}
        System.out.print(i + "," + j + "," + m + "," + n);
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|-----------------------|-----------------------|
| a. Prints: 1, 1, 2, 3 | b. Prints: 1, 1, 2, 2 |
| c. Prints: 1, 1, 3, 1 | d. Prints: 1, 0, 3, 1 |
| e. Prints: 2, 1, 3, 1 | f. Prints: 2, 1, 4, 1 |
| g. Prints: 0, 0, 3, 1 | h. Runtime Exception |
| i. Compiler Error | j. None of the Above |

Question 11

```

class Gray {
    public static void main (String args[]) {
        int i = 0;
        int j = 0;
        int k = 0;
label1:
        do {
            k++;
label2:
            do {

```

```

j++;
label3:
do {
    i++;
    System.out.print((i + j + k + ", "));
    switch ((i+j+k)) {
        default: break label1;
        case 3: break label3;
        case 4: break label3;
        case 5: break label2;
        case 6: continue label3;
        case 7: continue label2;
        case 8: continue label2;
        case 9: continue label1;
    }
} while (i++ < 2);
} while (j++ < 2);
} while (k++ < 2);
}
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: 3, 4, 6, 12 b. Prints: 3, 4, 7, 10
- c. Prints: 3, 5, 8, 12, d. Prints: 3, 6, 12,
- e. Prints: 3, 6, 8, 12, f. Runtime Exception
- g. Compiler Error h. None of the Above

Question 12

```

class Gray {
    public static void main (String args[]) {
        int i = 0;
        int j = 0;
        int k = 0;
label1:
        do {
            k++;
label2:
            do {
                j++;
label3:
                do {
                    i++;
                    System.out.print((i + j + k + ", "));
                    switch ((i+j+k)) {
                        case 3: continue label3;
                        case 4: continue label2;
                        case 5: continue label1;
                        case 6: break;
                        case 7: break label3;
                        case 8: break label2;
                        default: break label1;
                    }
                } while (i++ < 2);
            } while (j++ < 2);
        } while (k++ < 2);
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: 3, 4, 6, 8, 12 b. Prints: 3, 4, 7, 10

- c. Prints: 3, 5
- e. Prints: 3, 6, 8, 12
- g. Compiler Error
- d. Prints: 3, 5, 9
- f. Runtime Exception
- h. None of the Above

No.

정답 - 해설

Remark

1 c

Prints: v w x x y z z

Cases one and three have no break statement, so the next case is also executed and x and z are printed twice.

2 d

Prints: 61433

Since each iteration of the loop prints the value of the switch expression, it should be possible to figure out what is going on here.

3 g

None of the above

There is no break statement after the case statement with the constant expression c. Consequently, the next case is also processed so both 3 and 4 are printed. The question specifies that only one answer may be selected, so none of the above is the only single answer that is correct.

4 f

Compiler Error

The switch expression can not be of type long. The legal types for the switch expression are byte, short, char, and int.

5 c

Prints: 1433

Since each iteration of the loop prints the value of the switch expression, it should be possible to figure out what is going on here.

6 c

Prints: v y w z

The legal types for the switch expression are byte, short, char, and int. The constant case expressions can be any expression that is assignable to the type of the switch expression. 'd'-'a'=3. ~0=-1. 4&5=4.

7 f

Compiler error

The Java Language Specification states that "Every case constant expression associated with a switch statement must be assignable to the type of the switch expression." The constant expression 1000 is beyond the range of type byte.

8 g

Compiler Error

In the C programming language, the boolean value of false is represented by zero and true is any non zero value. In the Java programming language, numeric

values are not assignable to a boolean value. The use of the numeric value 1 as the boolean expression in the while loop causes a compiler error.

9 a

Prints: 12457

Since each iteration of the loop prints the value of the switch expression, it should be possible to figure out what is going on here. If more information is needed, then change the print statement so that the values of i and j are printed separately in columns.

10 e

Prints: 2, 1, 3, 1

A break statement with a label will transfer control outside of the enclosing statement with the matching label. Furthermore, the enclosing statements complete abruptly. In the case of a do-while loop, the boolean expression that controls the loop is not evaluated as control transfers out of the loop. However, if the break statement transfers control out of a try-catch-finally statement, then the finally clause is processed.

11 d

Prints: 3, 6, 12,

A break statement with a label will transfer control outside of the enclosing statement with the matching label. Furthermore, the enclosing statements complete abruptly. In the case of a do-while loop, the boolean expression that controls the loop is not evaluated as control transfers out of the loop. However, if the break statement transfers control out of a try-catch-finally block, then the finally clause is processed. The first iteration of this code example executes the case statement with the value 3. As a result, the statement "break label3" is executed causing the inner most loop to complete abruptly and control to be transferred out of the inner most loop to the second nested loop. The boolean expression of the second loop is evaluated and variable j is incremented as a side effect. The boolean expression evaluates as true and a second iteration begins. Variable j is incremented again at the top of the loop and variable i is incremented at the top of the inner most loop. The switch expression evaluates as six. A continue statement without a label will cause the current iteration of the enclosing loop to end and the next iteration to begin. As a result, the boolean expression that controls the loop is evaluated as the next iteration begins. Similarly, a continue statement that has a label transfers control to the next iteration of the enclosing loop that has the same label. If the continue statement causes control to break out of a nested loop to transfer control to an outer loop with a matching label, then execution of the nested loop is completed abruptly as control transfers to the outer loop. Furthermore, the boolean expression that controls the nested loop is not evaluated as control is abruptly transferred to the labeled outer loop.

12 d

Prints: 3, 5, 9,

A continue statement without a label will cause the current iteration of the enclosing loop to end and the next iteration to begin. As a result, the boolean expression that controls the loop is evaluated as the next iteration begins. Similarly, a continue statement that has a label transfers control to the next iteration of the enclosing loop that has the same label. If the continue statement causes control to break out of a nested loop to transfer control to an outer loop with a matching label, then execution of the nested loop is completed abruptly as control transfers to the outer loop. Furthermore, the boolean expression that controls the nested loop is not evaluated as control is abruptly transferred to the labeled outer loop.

[Exception Handling]

Question 1

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
    public static void main(String args[]) {
        int a, b, c, d, f;
        a = b = c = d = f = 0;
        int x = 4;

        try {
            switch (x) {
                case 1: throw new Level1Exception();
                case 2: throw new Level2Exception();
                case 3: throw new Level3Exception();
            }
            a++;
        }
        catch (Level3Exception e) { b++;}
        catch (Level2Exception e) { c++;}
        catch (Level1Exception e) { d++;}
        finally {f++;}
        System.out.print(a+","+b+","+c+","+d+","+f);
    }
}
```

What is the result of attempting to compile and run the program ?

- | | |
|----------------------|----------------------|
| a. Prints: 0,0,0,1,1 | b. Prints: 0,0,1,1,1 |
| c. Prints: 0,1,1,1,1 | d. Prints: 1,1,1,1,1 |
| e. Prints: 0,0,1,0,1 | f. Prints: 0,1,0,0,1 |
| g. Prints: 1,0,0,0,1 | h. Compiler Error |
| i. Run Time Error | j. None of the Above |

Question 2

```
1. class A {
2.     void m1() {throw new ArithmeticException();}
3.     void m2() {throw new ClassCastException();}
4.     void m3() {throw new IllegalArgumentException();}
5.         void m4() {throw new
IndexOutOfBoundsException();}
6.     void m5() {throw new NullPointerException();}
7.     void m6() {throw new SecurityException();}
```

8. }

What is the result of attempting to compile the program ?

- a. Compiler error at line 2.
- b. Compiler error at line 3.
- c. Compiler error at line 4.
- d. Compiler error at line 5.
- e. Compiler error at line 6.
- f. Compiler error at line 7.
- g. None of the Above

Question 3

```
class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
    void m1() throws ColorException {throw new
WhiteException();}
    void m2() throws WhiteException {}
    public static void main (String[] args) {
        White white = new White();
        int a,b,d,f;
        a = b = d = f = 0;
        try {white.m1():a++;} catch (ColorException e)
{b++;}
        try {white.m2():d++;} catch (WhiteException e)
{f++;}
        System.out.print(a+","+b+","+d+","+f);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,0
- c. Prints: 0,1,1,0
- d. Prints: 1,1,1,0
- e. Prints: 1,1,1,1
- f. Compiler Error
- g. Run Time Error
- h. None of the Above

Question 4

```
1. class A {
2.     void m1() {throw new ClassNotFoundException();}
3.     void m2() {throw new ArithmeticException();}
4.     void m3() {throw new ClassCastException();}
5.     void m4() {throw new IllegalArgumentException();}
6.         void m5() {throw new
CloneNotSupportedException();}
7.     void m6() {throw new NoSuchFieldException();}
8. }
```

What is the result of attempting to compile the program ?

- a. Compiler error at line 2.
- b. Compiler error at line 3.
- c. Compiler error at line 4.
- d. Compiler error at line 5.
- e. Compiler error at line 6.
- f. Compiler error at line 7.
- g. None of the Above

Question 5

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
```

```
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 1;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                }
                a++;
            }
            catch (Level2Exception e) {b++;}
            finally{c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
        finally {g++;}
        System.out.print(a+","+b+","+c+","+d+","+f+","+g);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,0,0,1,0,0
- b. Prints: 0,0,1,1,0,1
- c. Prints: 0,1,1,1,0,1
- d. Prints: 1,0,1,1,0,1
- e. Prints: 1,1,1,1,0,1
- f. Compiler Error
- g. Run Time Error
- h. sNone of the Above

Question 6

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 5;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                    case 4: throw new Exception();
                }
                a++;
            }
            catch (Level2Exception e) {b++;}
            finally{c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
        finally {g++;}
        System.out.print(a+","+b+","+c+","+d+","+f+","+g);
    }
}
```

What is the result of attempting to compile and run

the program ?

- a. Prints: 1,0,0,0,0,0
- b. Prints: 1,0,1,0,0,1
- c. Prints: 0,0,1,0,0,1
- d. Prints: 1,1,1,1,1,1
- e. Compiler Error
- f. Run Time Error
- g. None of the Above

Question 7

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
    public static void main(String args[]) {
        int a, b, c, d, f;
        a = b = c = d = f = 0;
        int x = 1;

        try {
            switch (x) {
                case 1: throw new Level1Exception();
                case 2: throw new Level2Exception();
                case 3: throw new Level3Exception();
            }
            a++;
        }
        catch (Level3Exception e) { b++;}
        catch (Level2Exception e) { c++;}
        catch (Level1Exception e) { d++;}
        finally {f++;}
        System.out.print(a+","+b+","+c+","+d+","+f);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compiler Error
- i. Run Time Error
- j. None of the Above

Question 8

```
class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
    void m1() throws ColorException {throw new
ColorException();}
    void m2() throws WhiteException {throw new
ColorException();}
    public static void main (String[] args) {
        White white = new White();
        int a,b,d,f;
        a = b = d = f =0;
        try {white.m1():a++;} catch (ColorException e)
{b++;}
        try {white.m2():d++;} catch (WhiteException e)
{f++;}
        System.out.print(a+","+b+","+d+","+f);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,1
- c. Prints: 0,1,0,1
- d. Prints: 0,1,1,1
- e. Prints: 1,1,1,1
- f. Compiler Error
- g. Run Time Error
- h. None of the Above

Question 9

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 3;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                }
                a++;
            }
            catch (Level2Exception e) {b++;}
            finally{c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
        finally {g++;}
        System.out.print(a+","+b+","+c+","+d+","+f+","+g);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 1,1,1,0,0,1
- b. Prints: 0,1,1,0,0,1
- c. Prints: 0,1,0,0,0,0
- d. Prints: 0,1,0,0,0,1
- e. Prints: 0,0,1,0,0,1
- f. Compiler Error
- g. Run Time Error
- h. None of the Above

No.

정답 - 해설

Remark

1 g

Prints: 1,0,0,0,1

No exception is thrown so variable a is incremented. The finally block is also executed.

2 g

None of the Above

Methods m1(), m2(), m3(), m4(), m5(), and m6() throw subclasses of RuntimeException. Any exception that is a direct subclass of RuntimeException should not be caught and should not be declared in the throws clause of a method.

3 c

Prints: 0,1,1,0

White.m1 throws an exception so variable a is not incremented. White.m2 does not throw an exception

so variable f is not incremented. The throws clause of m1 declares a ColorException, so the body may throw a ColorException or any subclass of ColorException. Although the throws clause of m2 declares a WhiteException, there is no requirement to throw any exception.

4 a e f

Compiler error at line 2. Compiler error at line 6. Compiler error at line 7.

All of the exceptions in this question are a subclass of Exception and three are a subclass of RuntimeException. Any exception that is a subclass of RuntimeException should not be caught. Instead, it should be allowed to bubble to the top of the program where the runtime system will display a stack trace. RuntimeExceptions are a result of program bugs. A RuntimeException should not be listed in the throws clause of a program. Methods m1, m5, and m6 throw subclasses of Exception that are not subclasses of RuntimeException. Any Exception that is a direct subclass of Exception must either be caught or must be declared in the throws clause of the method.

5 b

Prints: 0,0,1,1,0,1

The nested catch block is able to catch a Level2Exception or any subclass of it. The Level1Exception is not a subclass of Level2Exception so it is caught by the second of the two outer catch blocks. Both finally blocks are executed.

6 b

Prints: 1,0,1,0,0,1

No exception is thrown so a is incremented. Both of the finally blocks are then executed.

7 a

Prints: 0,0,0,1,1

The first catch block is able to catch a Level3Exception or any subclass of Level3Exception. The second catch block is able to catch a Level2Exception or any subclass of Level2Exception. The third catch block is the only one that is able to catch a Level1Exception. The finally block is also executed.

8 f

Compiler Error

The throws clause of White.m2 declares a WhiteException, so the body of m2 may throw a WhiteException or any subclass of WhiteException. Instead, the body of m2 throws a superclass of WhiteException. The result is a compiler error.

9 b

Prints: 0,1,1,0,0,1

The nested catch block is able to catch a Level2Exception or any subclass of it causing b to be incremented. Both of the finally blocks are then executed.

Question 10

```
1. class A {
2.         void m1() {throw new
   IndexOutOfBoundsException();}
3. void m2() {throw new InstantiationException();}
4. void m3() {throw new IllegalAccessException();}
5. void m4() {throw new NullPointerException();}
6. void m5() {throw new NoSuchMethodException();}
7. void m6() {throw new SecurityException();}
8. }
```

What is the result of attempting to compile the program ?

- a. Compiler error at line 2.
- b. Compiler error at line 3.
- c. Compiler error at line 4.
- d. Compiler error at line 5.
- e. Compiler error at line 6.
- f. Compiler error at line 7.
- g. None of the Above

Question 11

```
class ColorException extends Exception {}
class WhiteException extends ColorException {}
abstract class Color {
    abstract void m1() throws ColorException;
}
class White extends Color {
    void m1() throws WhiteException {throw new
WhiteException();}
    public static void main (String[] args) {
        White white = new White();
        int a,b,c,d;
        a = b = c = 0;
        try {
            white.m1();
            a++;
        } catch (WhiteException e) {b++;}
        finally {c++;}
        System.out.print(a+","+b+","+c);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,0,0
- b. Prints: 0,0,1
- c. Prints: 0,1,0
- d. Prints: 0,1,1
- e. Prints: 1,0,0
- f. Prints: 1,0,1
- g. Prints: 1,1,0
- h. Prints: 1,1,1
- i. Compiler Error
- j. Run Time Error
- k. None of the Above

Question 12

```
1. class A {
2. void m1() {throw new ClassNotFoundException();}
3. void m2() {throw new
CloneNotSupportedException();}
4. void m3() {throw new IllegalAccessException();}
5. void m4() {throw new InstantiationException();}
6. void m5() {throw new InterruptedException();}
```

```

7. void m6() {throw new NoSuchMethodException();}
8. }

```

What is the result of attempting to compile the program ?

- a. Compiler error at line 2.
- b. Compiler error at line 3.
- c. Compiler error at line 4.
- d. Compiler error at line 5.
- e. Compiler error at line 6.
- f. Compiler error at line 7.
- g. None of the Above

Question 13

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 4;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                    case 4: throw new Exception();
                }
                a++;
            }
            catch (Level2Exception e) {b++;}
            finally{c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
        finally {g++;}
        System.out.print(a+","+b+","+c+","+d+","+f+","+g);
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,0,0,0,0,1
- b. Prints: 0,0,0,0,1,0
- c. Prints: 0,0,1,0,0,1
- d. Prints: 0,0,1,0,1,1
- e. Prints: 0,1,1,1,1,1
- f. Prints: 1,1,1,1,1,1
- g. Compiler Error
- h. Run Time Error
- i. None of the Above

Question 14

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
    public static void main(String args[]) {
        int a, b, c, d, f;
        a = b = c = d = f = 0;
        int x = 2;

        try {
            switch (x) {

```

```

                case 1: throw new Level1Exception();
                case 2: throw new Level2Exception();
                case 3: throw new Level3Exception();
            }
            a++;
        }
        catch (Level3Exception e) { b++;}
        catch (Level2Exception e) { c++;}
        catch (Level1Exception e) { d++;}
        finally {f++;}
        System.out.print(a+","+b+","+c+","+d+","+f);
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compiler Error
- i. Run Time Error
- j. None of the Above

Question 15

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 2;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                }
                a++;
            }
            catch (Level2Exception e) {b++;}
            finally{c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
        finally {g++;}
        System.out.print(a+","+b+","+c+","+d+","+f+","+g);
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,0,1,0,0,1
- b. Prints: 0,1,0,0,0,0
- c. Prints: 0,1,1,0,0,1
- d. Prints: 0,1,0,0,0,1
- e. Prints: 1,1,1,0,0,1
- f. Compiler Error
- g. Run Time Error
- h. None of the Above

Question 16

```

class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
    void m1() throws ColorException {throw new

```

```

ColorException();}
void m2() throws WhiteException {throw new
WhiteException();}
public static void main (String[] args) {
    White white = new White();
    int a,b,d,f;
    a = b = d = f = 0;
    try {white.m1():a++;} catch (WhiteException e)
{b++;}
    try {white.m2():d++;} catch (WhiteException e)
{f++;}
    System.out.print(a+","+b+","+d+","+f);
}
}

```

What is the result of attempting to compile and run the program ?

- | | |
|--------------------|----------------------|
| a. Prints: 0,1,0,0 | b. Prints: 1,1,0,1 |
| c. Prints: 0,1,0,1 | d. Prints: 0,1,1,1 |
| e. Prints: 1,1,1,1 | f. Compiler Error |
| g. Run Time Error | h. None of the Above |

Question 17

```

class A {
    public static void main (String[] args) {
        Object error = new Error();
        Object runtimeException = new
RuntimeException();
        System.out.print((error instanceof Exception) + ",");
        System.out.print(runtimeException instanceof
Exception);
    }
}

```

What is the result of attempting to compile and run the program ?

- | | |
|------------------------|-----------------------|
| a. Prints: false,false | b. Prints: false,true |
| c. Prints: true,false | d. Prints: true,true |
| e. Compiler Error | f. Run Time Error |
| g. None of the Above | |

Question 18

```

class A {
    public static void main (String[] args) {
        Error error = new Error();
        Exception exception = new Exception();
        System.out.print((exception instanceof Throwable)
+ ",");
        System.out.print(error instanceof Throwable);
    }
}

```

What is the result of attempting to compile and run the program ?

- | | |
|------------------------|-----------------------|
| a. Prints: false,false | b. Prints: false,true |
| c. Prints: true,false | d. Prints: true,true |
| e. Compiler Error | f. Run Time Error |
| g. None of the Above | |

No.

정답 - 해설

Remark

10 b c e

Compiler error at line 3. Compiler error at line 4. Compiler error at line 6.

All of the exceptions in this question are a subclass of Exception and three are a subclass of RuntimeException. Any exception that is a subclass of RuntimeException should not be caught. Instead, it should be allowed to bubble to the top of the program where the runtime system will display a stack trace. RuntimeExceptions are a result of program bugs. A RuntimeException should not be listed in the throws clause of a program. Methods m2, m3, and m5 throw subclasses of Exception that are not subclasses of RuntimeException. Any Exception that is a direct subclass of Exception must either be caught or must be declared in the throws clause of the method.

11 d

Prints: 0,1,1

An exception is thrown so variable a is not incremented. Although Color.m1 declares a ColorException in the throws clause, a subclass of Color is free to declare only a subclass of Color in the throws clause of the overriding method.

12 a b c d e f

Compiler error at line 2. Compiler error at line 3. Compiler error at line 4. Compiler error at line 5. Compiler error at line 6. Compiler error at line 7.

Methods m1(), m2(), m3(), m4(), m5(), and m6() throw subclasses of Exception that are not subclasses of RuntimeException. Any Exception that is a direct subclass of Exception must either be caught or must be declared in the throws clause of the method.

13 d

Prints: 0,0,1,0,1,1

The nested catch block is able to catch a Level2Exception or any subclass of it, but Exception is not a subclass of Level2Exception. The Exception is caught by the second of the two outer catch blocks causing f to be incremented. Both of the finally blocks are then executed.

14 e

Prints: 0,0,1,0,1

The first catch block is able to catch a Level3Exception or any subclass of Level3Exception. The second catch block is able to catch a Level2Exception or any subclass of Level2Exception, so variable c is incremented. The finally block is also executed.

15 c

Prints: 0,1,1,0,0,1

The nested catch block is able to catch a Level2Exception or any subclass of it causing b to be incremented. Both of the finally blocks are then executed.

16 f

Compiler Error

The throws clause of White.m1 declares a ColorException, but the catch clause in the main method catches only a subclass of ColorException. The result is a compiler error.

17 b

Prints: false,true

Error is a direct subclass of Throwable. RuntimeException is a direct subclass of Exception.

18 d

Prints: true,true

Both Error and Exception are subclasses of Throwable.

[Inheritance]

Question 1

```
class G {
    String s1 = "G.s1";
    void printS1(){System.out.print("G.printS1," + s1);}
    G() { printS1();}
}
```

```
class H extends G {
    String s1 = "H.s1";
    void printS1(){System.out.print("H.printS1," + s1);}
    public static void main(String[] args) {
        H h = new H();
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: G.printS1,G.s1
- b. Prints: G.printS1,H.s1
- c. Prints: G.printS1,null
- d. Prints: H.printS1,G.s1
- e. Prints: H.printS1,H.s1
- f. Prints: H.printS1,null
- g. Runtime Exception
- h. Compiler Error
- i. None of the Above

Question 2

```
interface I {String s1 = "I";}
class A implements I {String s1 = "A";}
class B extends A {String s1 = "B";}
class C extends B {
    String s1 = "C";
    void printIt() {
        System.out.print(((A)this).s1 + ((B)this).s1 +
                          ((C)this).s1 + ((I)this).s1);
    }
    public static void main (String[] args) {
        new C().printIt();
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: ABCI
- b. Runtime Exception

c. Compiler Error

d. None of the Above

Question 3

```
class P {
    static void printS1(){System.out.print("P.printS1 ");}
    void printS2() {System.out.print("P.printS2 ");}
    void printS1S2(){printS1();printS2();}
}
```

```
class Q extends P {
    static void printS1(){System.out.print("Q.printS1 ");}
    void printS2(){System.out.print("Q.printS2 ");}
    public static void main(String[] args) {
        new Q().printS1S2();
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: P.printS1 P.printS2
- b. Prints: P.printS1 Q.printS2
- c. Prints: Q.printS1 P.printS2
- d. Prints: Q.printS1 Q.printS2
- e. Runtime Exception
- f. Compiler Error
- g. None of the Above

Question 4

```
class M {
    static String s1 = "M.s1";
    static void printS1(){System.out.print("M.printS1,"+s1+" ");}
    M() { printS1();}
}
```

```
class N extends M {
    static String s1 = "N.s1";
    static void printS1(){System.out.print("N.printS1,"+s1+" ");}
    N() { printS1();}
    public static void main(String[] args) {
        N n = new N();
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: M.printS1,M.s1 N.printS1,N.s1
- b. Prints: N.printS1,N.s1 N.printS1,N.s1
- c. Prints: N.printS1,N.s1 M.printS1,M.s1
- d. Prints: N.printS1,M.s1 M.printS1,M.s1
- e. Prints: N.printS1,N.s1
- f. Runtime Exception
- g. Compiler Error
- h. None of the Above

Question 5

```
class R {
    private void printS1(){System.out.print("R.printS1 ");}
    protected void printS2() {System.out.print("R.printS2 ");}
    protected void printS1S2(){printS1();printS2();}
```

```

}

class S extends R {
    private void printS1(){System.out.print("S.printS1 ");}
    protected void printS2(){System.out.print("S.printS2
");}
    public static void main(String[] args) {
        new S().printS1S2();
    }
}

```

What is the result of attempting to compile and run the above program ?

- Prints: R.printS1 R.printS2
- Prints: R.printS1 S.printS2
- Prints: S.printS1 R.printS2
- Prints: S.printS1 S.printS2
- Runtime Exception
- Compiler Error
- None of the Above

Question 6

```

1. interface I10 {
2.     String name = "I10";
3.     String s10 = "I10.s10";
4. }
5. interface I20 {
6.     String name = "I20";
7.     String s20 = "I20.s20";
8. }
9. class C10 implements I10, I20 {
10.     public static void main(String[] args) {
11.         System.out.print(s10+",");
12.         System.out.print(s20+",");
13.         System.out.print(name);
14.     }
15. }

```

What is the result of attempting to compile and run the above program ?

- Prints: I10.s10,I20.s20,I10
- Prints: I10.s10,I20.s20,I20
- Prints: I10.s10,I20.s20,
- Prints: I10.s10,I20.s20,null
- Compiler error at line 9.
- Compiler error at line 11.
- Compiler error at line 12.
- Compiler error at line 13.
- Runtime Error
- None of the Above

Question 7

```

class T {
    private int i1, i2;
    void printI1I2() {System.out.print("T: i1=" + i1 + ",
i2=" + i2);}
    T(int i1, int i2) {
        this.i1=i1;this.i2=i2;
    }
}

```

```

class U extends T {

```

```

    private int i1, i2;
    void printI1I2() {System.out.print("U: i1=" + i1 + ",
i2=" + i2);}
    U(int i1, int i2) {
        this.i1=i1;this.i2=i2;
    }
    public static void main(String[] args) {
        T t = new U(1,2);
        t.printI1I2();
    }
}

```

What is the result of attempting to compile and run the above program ?

- Prints: U: i1=1, i2=2
- Prints: T: i1=1, i2=2
- Prints: U: i1=null, i2=null
- Prints: T: i1=null, i2=null
- Runtime Exception
- Compiler Error
- None of the Above

Question 8

```

class A {static void m() {System.out.print("A");}}
class B extends A {static void m()
{System.out.print("B");}}
class C extends B {static void m()
{System.out.print("C");}}
class D {
    public static void main(String[] args) {
        C c = new C();
        c.m();
        B b = c;
        b.m();
        A a = b;
        a.m();
    }
}

```

What is the result of attempting to compile and run the above program ?

- Prints: AAA
- Prints: ABC
- Prints: CBA
- Prints: CCC
- Compiler Error
- Runtime Error
- None of the Above

No.

정답 - 해설

Remark

1 f

Prints: H.printS1,null

The default constructor of H calls the constructor of G. The constructor of G calls the overloaded method printS1 causing the invocation of H.printS1. The instance variable H.s1 hides G.s1. Although G.s1 has already been initialized, the initialization of the instance variables of H won't begin until the initialization and construction of G is complete.

2 a

Prints: ABCI

The field access expression ((X)this).identifier can be used to access hidden fields that are visible within the super class or interface X.

3 b

Prints: P.printS1 Q.printS2

Static method Q.printS1 hides the static method P.printS1 in the super class P. Instance method Q.printS2 overrides the instance method P.printS2. Due to the differences between the hiding of static methods and the overriding of instance methods the invocation of the two methods in P.printS1S2 produces different results. The method invocation expression printS1 results in the invocation of the hidden super class method P.printS1. The method invocation expression printS2 results in the invocation of the overriding sub class method Q.printS2.

4 a

Prints: M.printS1,M.s1 N.printS1,N.s1

A static method may hide a method in the super class, but a static method does not override a super class method.

5 b

Prints: R.printS1 S.printS2

Method R.printS1 is private and therefore is not inherited by class S and is not overridden by method S.printS1. In contrast, method R.printS2 is protected and therefore is inherited by class S and is overridden by method S.printS2. R.printS1S2 therefore calls the R.printS1 and S.printS2 methods.

6 h

Compiler error at line 13.

Class C10 inherits ambiguous declarations of the name field. As long as the field is not referenced as a member of class C10, then no compiler error occurs. Line 13 generates the compiler error, because it is the first to access the name field as a member of class C10.

7 f

Compiler Error

The two-parameter constructor of U does not invoke the two-parameter constructor of T. Therefore, the constructor of U will try to invoke a no-parameter constructor of T, but none exists.

8 c

Prints: CBA

The method, m, is static; therefore, the method that is actually invoked at runtime is based on the type of the reference and not based on the type of the object instance at runtime. When the object reference is of type A, then the method from A is invoked even though the type of the object instance at runtime is C.

Question 9

1. class A {void m1(String s1) {}}

```
2. class B extends A {
3.     void m1(String s1) {}
4.     void m1(boolean b) {}
5.     void m1(byte b) throws Exception {}
6.     String m1(short s) {return new String();}
7.     private void m1(char c) {}
8.     protected void m1(int i) {}
9. }
```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 3.
- b. Compiler error at line 4.
- c. Compiler error at line 5.
- d. Compiler error at line 6.
- e. Compiler error at line 7.
- f. Compiler error at line 8.
- g. Compiler error at line 9.
- h. None of the Above

Question 10

```
class A {
    void m1() {System.out.print("A.m1");}
}
class B extends A {
    void m1() {System.out.print("B.m1");}
    static void m1(String s) {System.out.print(s+",");}
}
class C {
    public static void main (String[] args) {
        B.m1("main");
        new B().m1();
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: main,B.m1
- b. Compiler Error
- c. Runtime Error
- d. None of the Above

Question 11

```
1. interface I10 {
2.     String name = "I10";
3.     String s10 = "I10.s10";
4. }
5. interface I20 {
6.     String name = "I20";
7.     String s20 = "I20.s20";
8. }
9. class C20 implements I10, I20 {
10.     public static void main(String[] args) {
11.         System.out.print(I10.s10+",");
12.         System.out.print(I20.s20+",");
13.         System.out.print(I20.name);
14.     }
15. }
```

What is the result of attempting to compile and run the above program ?

- a. Prints: I10.s10,I20.s20,I10
- b. Prints: I10.s10,I20.s20,I20

- c. Prints: I10.s10,I20.s20,
- d. Prints: I10.s10,I20.s20,null
- e. Compiler error at line 9.
- f. Compiler error at line 11.
- g. Compiler error at line 12.
- h. Compiler error at line 13.
- i. Runtime Error
- j. None of the Above

Question 12

```
class A {
    String s1 = "A.s1";
    String s2 = "A.s2";
}

class B extends A {
    String s1 = "B.s1";
    public static void main(String args[]) {
        B x = new B();
        A y = (A)x;
        System.out.println(x.s1+" "+x.s2+" "+y.s1+" "+y.s2);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: B.s1 A.s2 B.s1 A.s2
- b. Prints: B.s1 A.s2 A.s1 A.s2
- c. Prints: A.s1 A.s2 B.s1 A.s2
- d. Prints: A.s1 A.s2 A.s1 A.s2
- e. Runtime error.
- f. Compiler error.
- g. None of the Above

Question 13

```
abstract class D {
    String s1 = "D";
    abstract String getS1();
}

class E extends D {
    String s1 = "E";
    String getS1() {return s1;}
}

class F {
    public static void main (String[] s) {
        D x = new E();
        System.out.print(x.s1 + x.getS1());
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: DD
- b. Prints: DE
- c. Prints: ED
- d. Prints: EE
- e. Runtime Error
- f. Compiler Error
- g. None of the Above

Question 14

```
class A {String s1 = "A";}
class B extends A {String s1 = "B";}
class C extends B {String s1 = "C";}
class D {
```

```
static void m1(A x) {System.out.print(x.s1);}
static void m1(B x) {System.out.print(x.s1);}
static void m1(C x) {System.out.print(x.s1);}
public static void main(String[] args) {
    A a; B b; C c;
    a = b = c = new C();
    m1(a);m1(b);m1(c);
}
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CBA
- d. Prints: CCC
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 15

```
class E {
    void printS1(){System.out.print("E.printS1 ");}
    static void printS2() {System.out.print("E.printS2");}
}

class F extends E {
    void printS1(){System.out.print("F.printS1 ");}
    static void printS2() {System.out.print("F.printS2");}
    public static void main (String args[]) {
        E x = new F();
        x.printS1(); x.printS2();
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: E.printS1 E.printS2
- b. Prints: E.printS1 F.printS2
- c. Prints: F.printS1 E.printS2
- d. Prints: F.printS1 F.printS2
- e. Runtime Exception
- f. Compiler Error
- g. None of the Above

Question 16

```
class C {
    void printS1() {System.out.print("C.printS1 ");}
    static void printS2() {System.out.print("C.printS2 ");}
}

class D extends C {
    void printS1(){System.out.print("D.printS1 ");}
    void printS2() {System.out.print("D.printS2 ");}
    public static void main (String args[]) {
        C c = new D();
        c.printS1(); c.printS2();
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: C.printS1 C.printS2
- b. Prints: C.printS1 D.printS2
- c. Prints: D.printS1 C.printS2

- d. Prints: D.printS1 D.printS2
- e. Runtime error
- f. Compiler error
- g. None of the above

No.
정답 - 해설
Remark

9 h
None of the Above
Overloaded methods may have different return types and a different throws clause.

10 a
Prints: main,B.m1
The no parameter version of B.m1 overrides A.m1. Therefore, B.m1 must comply with the rules of method overriding. The single parameter version of B.m1 overloads the m1 method and therefore must comply with the rules of method overloading. For example, one version of the overloaded method may be an instance method while the other is a static method.

11 b
Prints: I10.s10,I20.s20,I20
This is a trick question. Class C10 inherits ambiguous declarations of the name field. As long as the field is not referenced as a member of class C10, then no compiler error occurs. Although line 13 may appear to generate the compiler error it does not, because name is access directly as a member of interface I20. Therefore, the compiler does not encounter an ambiguity.

12 b
Prints: B.s1 A.s2 A.s1 A.s2
The variables of a subclass can hide the variables of a superclass or interface. The variable that is accessed depends on the type of the reference.

13 b
Prints: DE
The instance variable referred to by x.s1 depends on the type of the reference at compile time. The instance method referred to by x.getS1() depends on the type of the object at runtime.

14 b
Prints: ABC
In all three cases, the object passed to method m1 is an instance of class C; however, the type of the reference is different for each method. Since member variables are accessed based on the type of the reference, the value printed by each method is different even though the same object instance is used for each method invocation.

15 c
Prints: F.printS1 E.printS2
An instance method can override a superclass

method, but a static method does not override a superclass method. Instead, a static method can hide a superclass method. The static method that is accessed will depend on the type of the reference.

16 f
Compiler error
An instance method can not override a static method.

[Initialization]

Question 1
class Magenta {
 public static void main(String[] args) {
 int[][] a =
 {{1,2},{'a'-'a','b'-'a','c'-'a'}},{~0,2&3,1<<1,3^2},{}};
 for (int i = 0; i < a.length; i++) {
 System.out.print(a[i].length+",");
 }
 }
}

What is the result of attempting to compile and run the above program ?

- a. Prints: 2,3,4,0
- b. Prints: 1,2,5,0
- c. Compiler Error.
- d. Runtime Error.
- e. None of the Above.

Question 2
class Black {
 static byte a;
 static short b;
 static char c;
 static int d;
 static long e;
 static String s;
 public static void main(String[] args) {
 System.out.println(a+b+c+d+e+s);
 }
}

What is the result of attempting to compile and run the above program ?

- a. Prints: 00000null
- b. Prints: 00000
- c. Prints: 0null
- d. Prints: 0
- e. Prints: null
- f. Compiler Error.
- g. Runtime Exception.
- h. None of the Above.

Question 3
interface C {
 char w = DD.echo('w');
 char x = DD.echo('x');
}
interface D extends C {
 char y = DD.echo('y');
 char z = DD.echo('z');
 char a = DD.echo(w);
}

class DD implements D {
 static char echo(char c) {

```

        System.out.print(c);
        return c;
    }
    public static void main (String[] args) {
        System.out.print("Main");
        DD dd = new DD();
        System.out.println(a);
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: yzwMain
- b. Prints: Mainyzw
- c. Prints: wxyzwMain
- d. Prints: Mainwxyzw
- e. Prints: yzwxwwMain
- f. Prints: Mainyzwxww
- g. Runtime Exception
- h. Compiler Error
- i. None of the Above

Question 4

```

class Blue {
    Blue() {System.out.print("Constructor ");}
    static {
        System.out.print("Static ");
    }
    public static void main(String args[]) {
        System.out.print("Main ");
        Blue blue = new Blue();
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: Constructor Static Main
- b. Prints: Static Main Constructor
- c. Prints: Main Constructor Static
- d. Prints: Main Static Constructor
- e. Prints: Static Constructor Main
- f. Prints: Constructor Main Static
- g. Runtime error.
- h. Compiler error.
- i. None of the Above

Question 5

```

class Green {
    static {
        System.out.print("Static ");
    }
    String s1 = echo("s1 ");
    static String s2 = echo("s2 ");
    static String echo (String s) {
        System.out.print(s);
        return s;
    }
    public static void main(String args[]) {
        Green g = new Green();
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: Static s1 s2
- b. Prints: s1 s2 Static
- c. Prints: s2 Static s1
- d. Prints: s2 s1 Static

- e. Prints: s1 Static s2
- f. Prints: Static s2 s1
- g. Runtime Exception
- h. Compiler Error
- i. None of the Above

Question 6

```

class Red {
    Red() {System.out.print("Constructor ");}
    static {
        System.out.print("Static ");
    }
    {
        System.out.print("Instance ");
    }
    public static void main(String args[]) {
        Red red = new Red();
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: Instance Static Constructor
- b. Prints: Static Constructor Instance
- c. Prints: Constructor Instance Static
- d. Prints: Constructor Static Instance
- e. Prints: Static Instance Constructor
- f. Prints: Instance Constructor Static
- g. Runtime error
- h. Compiler error
- i. None of the above

Question 7

```

interface C {
    char w = DD.echo('w');
    char x = DD.echo('x');
}
interface D extends C {
    char y = DD.echo('y');
    char z = DD.echo('z');
    char a = DD.echo(w);
}

```

```

class DD implements D {
    static char echo(char c) {
        System.out.print(c);
        return c;
    }
    public static void main (String[] args) {
        System.out.print("Main");
        DD dd = new DD();
        System.out.println(w);
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: wxwMain
- b. Prints: Mainwxw
- c. Prints: wxyzwMain
- d. Prints: Mainwxyzw
- e. Prints: yzwwxMain
- f. Prints: Mainyzwwx
- g. Runtime Exception
- h. Compiler Error
- i. None of the Above

Question 8

```

class White {
    static byte a;
    static short b;
    static char c;
    static int d;
    static long e;
    static String s;
    public static void main(String[] args) {

System.out.println(a+","+b+","+ (int)c+","+d+","+e+","+s);
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|---------------------------|-----------------------|
| a. Prints: 0,0,0,0,0,null | b. Prints: 0,0,0,0,0, |
| c. Prints: 0,0, ,0,0, | d. Compiler Error. |
| e. Runtime Exception. | f. None of the Above. |

Question 9

```

interface E {
    char v = 'v';
    char w = G.echo('w');
}
interface F extends E {
    char x = G.echo('x');
    char y = G.echo('y');
    char z = G.echo('z');
}

```

```

class G implements F {
    static char echo(char c) {
        System.out.print(c);
        return c;
    }
    public static void main (String[] args) {
        System.out.print("Main");
        G g = new G();
        System.out.println(v);
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|----------------------|----------------------|
| a. Prints: Mainv | b. Prints: wMainv |
| c. Prints: Mainwv | d. Prints: wxyzMain |
| e. Prints: Mainwxyzv | f. Prints: xyzwMainv |
| g. Prints: Mainxyzwv | h. Runtime Exception |
| i. Compiler Error | j. None of the Above |

No.
정답 - 해설
Remark

1 a
Prints: 2,3,4,0
a[0][1]=2, a[1][1]='b'-'a'=1, a[2][2]=1<<1=2

2 c
Prints: 0 null
The numeric sum of variables a, b, c, d, and e is

zero. The zero is converted to a string and concatenated with s.

3 f
Prints: Mainyzwxww

A variable in interface D is needed so all of the variables defined in D are initialized. One variable declared in D is defined in terms of a variable in C so all of the variables in C are initialized.

4 b
Prints: Static Main Constructor
Static initializers run before main.

5 f
Prints: Static s2 s1
Static variables and initializers are evaluated before instance variables.

6 e
Prints: Static Instance Constructor
Static initializers run before instance initializers.

7 b
Prints: Mainwxw
One of the variables defined in interface C is needed so all of the variables defined in C are initialized. None of the variables defined in D are needed, so none are initialized.

8 a
Prints: 0,0,0,0,0,null

9 a
Prints: Mainv
Field v of interface E is a compile time constant. Use of v does not cause interface E to be initialized at run time; because the value is known at compile time. For more information, please see section 12.4.1 of the Java Language Specification.

Question 1

```

class Orange {
    static {
        System.out.print("Static1 ");
    }
    static String string1 = echo("string1 ");
    static {
        System.out.print("Static2 ");
    }
    static String echo (String s) {
        System.out.print(s);
        return s;
    }
    public static void main(String args[]) {
        Orange o = new Orange();
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: Static1 string1 Static2

- b. Prints: string1 Static2 Static1
- c. Prints: Static2 Static1 string1
- d. Prints: Static2 string1 Static1
- e. Prints: string1 Static1 Static2
- f. Prints: Static1 Static2 string1
- g. Runtime Exception
- h. Compiler Error
- i. None of the Above

Question 2

```
class E {
    static byte a = (byte)127;
    static byte b = (byte)128;
    static byte c = (byte)255;
    static byte d = (byte)256;
    public static void main(String args[]) {
        System.out.print(a + " " + b + " " + c + " " + d);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 127 128 255 256
- b. Prints: 127 128 255 0
- c. Prints: 127 -1 -127 0
- d. Prints: 127 -128 -1 0
- e. Runtime Exception
- f. Compiler Error
- g. None of the Above

Question 3

```
class A {
    A() {System.out.print("CA ");}
    {System.out.print("IA ");}
    static {System.out.print("SA ");}
}
class B extends A {
    B() {System.out.print("CB ");}
    {System.out.print("IB ");}
    static {System.out.print("SB ");}
    public static void main (String[] args) {
        B b = new B();
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: SA IA CA SB IB CB
- b. Prints: SA SB IA CA IB CB
- c. Prints: SA SB IA IB CA CB
- d. Prints: SB IB SA IA CA CB
- e. Prints: SB IB CB SA IA CA
- f. Runtime Exception
- g. Compiler Error
- h. None of the Above

Question 4

```
class Maroon {
    public static void main(String[] args) {
        int[][] a = {{5,7,11},{ 'a'-'a','b'-'a','c'-'a'},{~0,2&3,1<<2,3^2}};
        System.out.print(a[0][2]+" "+a[1][0]+" "+a[2][1]);
    }
}
```

```
}
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 11,0,2
- b. Prints: -1,7,2
- c. Compiler Error.
- d. Runtime Error.
- e. None of the Above.

Question 5

```
class Teal {
    static boolean b1;
    public static void main(String[] args) {
        boolean[] array = new boolean[1];
        boolean b2;
        System.out.print(b1+"");
        System.out.print(array[0]+"");
        System.out.print(b2);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: true,true,true
- b. Prints: false,false,false
- c. Prints: null,null,null
- d. Prints: false,true,false
- e. Compiler Error.
- f. Runtime Error.
- g. None of the Above.

Question 6

```
class Gray {
    static int a;
    static float b;
    static double c;
    static boolean d;
    static String s;
    public static void main(String[] args) {
        System.out.println(a+" "+b+" "+c+" "+d+" "+s);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints 0,0,0,false,null
- b. Prints 0,0,0,false,
- c. Prints 0,0,0,0,0,false,null
- d. Prints 0,0,0,0,0,false,
- e. Prints 0,0,0,0,0,true,null
- f. Prints 0,0,0,0,0,true,
- g. Prints 0,0,0,true,null
- h. Prints 0,0,0,true,
- i. Compiler Error.
- j. Runtime Error.
- k. None of the Above.

Question 7

```
interface C {
    char w = DD.echo('w');
    char x = DD.echo('x');
}
interface D extends C {
    char y = DD.echo('y');
    char z = DD.echo('z');
```

```

char a = DD.echo(w);
}

class DD implements D {
    static char echo(char c) {
        System.out.print(c);
        return c;
    }
    public static void main (String[] args) {
        System.out.print("Main DD");
        DD dd = new DD();
    }
}

```

What is the result of attempting to compile and run the above program ?

- Prints: wxyzwMain DD
- Prints: Main DDwxyzw
- Prints: yzwwxMain DD
- Prints: Main DDyzwwx
- Runtime Exception
- Compiler Error
- None of the Above

Question 8

```

1. class Violet {
2.     int x;
3.     public static void main(String[] args) {
4.         int y;
5.         System.out.print("x="+x);
6.         System.out.print(", y="+y);
7.     }
8. }

```

What is the result of attempting to compile and run the above program ?

- Compiler error at line 1.
- Compiler error at line 2.
- Compiler error at line 3.
- Compiler error at line 4.
- Compiler error at line 5.
- Compiler error at line 6.
- Runtime Exception
- None of the Above

No.

정답 - 해설

Remark

1 a

Prints: Static1 string1 Static2

Static variables and initializers are evaluated in the order they are defined.

2 d

Prints: 127 -128 -1 0

Bytes are stored as 8 bit two's complement signed integers. When an int primitive is cast to a byte, the three most significant bytes are discarded and only the least significant byte remains. The most significant bit of the remaining byte becomes the new sign bit.

3 b

Prints: SA SB IA CA IB CB

Static initialization of the super class occurs first followed by static initialization of the sub class. Instance initialization of the super class occurs before instance initialization of the sub class.

4 a

Prints: 11,0,2

a[0][2]=11, a[1][0]='a'-'a'=0, a[2][1]=2&3=2

5 e

Compiler Error.

Teal.b1 is always initialized because it is a class member. Arrays are always initialized even if declared locally. Local variable b2 is not initialized because it is local. Consequentially, a compiler error is generated when the attempt is made to print the value of b2.

6 c

Prints 0,0,0,0,0,false,null

Generally speaking, numeric type variables are initialized to zero. Booleans are initialized to false. Reference type variables are initialized to null.

7 g

None of the Above

As a result of lazy initialization, the variables declared in the interfaces are not initialized because they are not used in the program.

8 e f

Compiler error at line 5. Compiler error at line 6.

The non-static field x can not be referenced from within a static method. The local variable y has not been initialized.

[Nested Classes]

Question 1

```

1. class F {
2.     public void m1() {Z.m1();}
3.     private static class Y {
4.         private static void m1() {
5.             System.out.print("Y.m1 ");
6.         }
7.     }
8.     private static class Z {
9.         private static void m1(){
10.             System.out.print("Z.m1 ");
11.             Y.m1();
12.         }
13.     }
14.     public static void main(String[] args) {
15.         new F().m1();
16.     }
17. }

```

What is the result of attempting to compile and run the above program ?

- a. Compiler error at line 2
- b. Compiler error at line 11
- c. Runtime error at line 2
- d. Runtime error at line 11
- e. Prints: Z.m1 Y.m1
- f. None of the Above

Question 2

```
class A {
    private static String s1 = "s1";
    final String s2 = "s2";
    A () { new Z("s5","s6");}
    class Z {
        final String s3 = "s3";
        String s4 = "s4";
        Z (final String s5, String s6) {
            System.out.print( );
        }
    }
    public static void main(String args[]) {new A();}
}
```

Which variables can be substituted for _____ without causing a compiler error ?

- a. s1
- b. s2
- c. s3
- d. s4
- e. s5
- f. s6
- g. Runtime error.
- h. Compiler error.
- i. None of the Above

Question 3

```
1. class Outer {
2.     static class StaticNested {
3.         static final int a = 25;
4.         static final int b;
5.         static int c;
6.         int d;
7.         static {b = 42;}
8.     }
9.     class NonStaticInner {
10.        static final int e = 25;
11.        static final int f;
12.        static int g;
13.        int h;
14.        static {f = 42;}
15.    }
16. }
```

What is the result of attempting to compile the program ?

- a. Compiler error at line 2.
- b. Compiler error at line 3.
- c. Compiler error at line 4.
- d. Compiler error at line 5.
- e. Compiler error at line 6.
- f. Compiler error at line 7.
- g. Compiler error at line 9.
- h. Compiler error at line 10.
- i. Compiler error at line 11.
- j. Compiler error at line 12.
- k. Compiler error at line 13.
- l. Compiler error at line 14.

- m. None of the Above

Question 4

```
class C {
    private static String s1 = "s1";
    String s2 = "s2";
    C() {m1("s5","s6");}
    void m1(final String s5, String s6) {
        final String s3 = "s3"; String s4 = "s4";
        class Z {
            Z() {
                System.out.print( );
            }
        }
        new Z();
    }
    public static void main(String args[]) {new C();}
}
```

Which variables can be substituted for _____ without causing a compiler error ?

- a. s1
- b. s2
- c. s3
- d. s4
- e. s5
- f. s6
- g. Runtime error
- h. Compiler error
- i. None of the above

Question 5

```
class B {
    private static String s1 = "s1";
    final String s2 = "s2";
    B () { new Z("s5","s6");}
    static class Z {
        final String s3 = "s3";
        static String s4 = "s4";
        Z (final String s5, String s6) {
            System.out.print( );
        }
    }
    public static void main(String args[]) {new B();}
}
```

Which variables can be substituted for _____ without causing a compiler error ?

- a. s1
- b. s2
- c. s3
- d. s4
- e. s5
- f. s6
- g. Runtime error
- h. Compiler error
- i. None of the above

Question 6

```
1. class Red {
2.     static class StaticNested {
3.         interface ABC {}
4.     }
5.     class NonStaticInner {
6.         interface DEF {}
7.     }
8.     interface GHI {}
9. }
```

What is the result of attempting to compile the

program ?

- a. Compiler error at line 2.
- b. Compiler error at line 3.
- c. Compiler error at line 5.
- d. Compiler error at line 6.
- e. Compiler error at line 8.
- f. None of the Above

Question 7

```
class G {  
    final String s1 = "G.s1";  
    class Z {  
        String s1;  
        void m1() {System.out.println(    );}  
    }  
    public static void main(String args[]) {  
        G g = new G();  
        g.new Z().m1();  
    }  
}
```

What field access expression could be used in place of above to cause the program to print "G.s1" ?

- a. s1
- b. G.s1
- c. ((G)this).s1
- d. G.this.s1
- e. G.super.s1
- f. None of the Above

Question 8

```
class E {  
    E() {System.out.print("E");}  
    static class Z {Z(){System.out.print("Z");}}  
    public static void main(String args[]) {  
        new E.Z();  
    }  
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: E
- b. Prints: Z
- c. Prints: EZ
- d. Prints: ZE
- e. Runtime Exception
- f. Compiler Error
- g. None of the Above

Question 9

```
1. class Red {  
2.     private static final int a = 10;  
3.     protected static int b = 20;  
4.     int c = 30;  
5.     static class StaticNested {  
6.         int d = a;  
7.         int e = b;  
8.         int f = c;  
9.     }  
10.    class NonStaticInner {  
11.        int g = a;  
12.        int h = b;  
13.        int i = c;  
14.    }  
15. }
```

What is the result of attempting to compile the program ?

- a. Compiler error at line 2.
- b. Compiler error at line 3.
- c. Compiler error at line 4.
- d. Compiler error at line 5.
- e. Compiler error at line 6.
- f. Compiler error at line 7.
- g. Compiler error at line 8.
- h. Compiler error at line 10.
- i. Compiler error at line 11.
- j. Compiler error at line 12.
- k. Compiler error at line 13.
- l. None of the Above

Question 10

```
class D {  
    D() {System.out.print("D");}  
    class Z {Z(){System.out.print("Z");}}  
    public static void main(String args[]) {  
        new D.Z();  
    }  
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: D
- b. Prints: Z
- c. Prints: DZ
- d. Prints: ZD
- e. Runtime Exception
- f. Compiler Error
- g. None of the Above

No.

정답 - 해설

Remark

1 e

Prints: Z.m1 Y.m1

Member classes have access to the private fields and methods of the enclosing class and other member classes. Furthermore, the enclosing class has access to the private fields and methods of the member classes.

2 a b c d e f

s1 s2 s3 s4 s5 s6

Class Z is a member class of class A and therefore has access to all of the fields and methods (static, non-static, and private) of class A.

3 i j l

Compiler error at line 11. Compiler error at line 12. Compiler error at line 14.

A non-static nested class is called an inner class. An inner class can not declare a static member unless it is a compile time constant. Even though f is final, it does not have a value at compile time and therefore it causes a compilation error. Member variable g also causes a compiler error because it is static. The static initializer of NonStaticInner causes a compiler error because inner classes (i.e. non-static nested classes) can not declare static initializers.

4 a b c e

s1 s2 s3 s5

Class Z is a local class defined within the code block of method m1 of class C. Class Z therefore has access to all fields and methods (static, non-static, and private) of class C. Additionally, class Z has access to the local variables and parameters of method m1 that are declared final. Consequently, class Z does not have access to parameter s6 or local variable s4.

5 a c d e f

s1 s3 s4 s5 s6

Class Z is a static member class of class B and therefore has access to all of the static fields and methods of class B. However, a static member class does not have access to non-static fields and methods of the enclosing class. Consequently, class Z does not have access to the s2 field of class B.

6 d

Compiler error at line 6.

An interface is implicitly static and therefore can not be declared as a member of a non-static nested class.

7 d

G.this.s1

The qualified this expression, ClassName.this.name, can be used to access hidden variables within an enclosing class.

8 b

Prints: Z

Class Z is a static member class of class E. Static member classes are similar to ordinary top-level classes with the added advantage of having access to all of the static fields and methods (including those that are private) of the enclosing class. Since Z can be instantiated without first creating an instance of the enclosing class, only the letter "Z" is printed.

9 g

Compiler error at line 8.

A static nested class does not have access to non-static members of the enclosing class.

10 f

Compiler Error

Class Z is a non-static member class of class D. Since Z is non-static, an instance of Z can not be instantiated without first creating an instance of class D. Therefore, the correct syntax for creating an instance of class Z would be as follows: "new D().new Z()".

[java.lang.Math]

Question 1

```
class B {
    public static void main(String[] args) {
        double d1 = Math rint(1.5);
    }
}
```

```
double d2 = Math rint(2.5);
System.out.print(d1 + "," + d2);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 1.0,2.0
- b. Prints: 1.0,3.0
- c. Prints: 2.0,2.0
- d. Prints: 2.0,3.0
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 2

```
class A {
    public static void main (String[] args) {
        System.out.print(Integer.MIN_VALUE ==
Math.abs(Integer.MIN_VALUE));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false
- b. Prints: true
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 3

```
class J {
    public static void main (String[] args) {
        float f1 = -0.0f;
        float f2 = +0.0f;
        float f3 = Math.min(f1,f2);
        float f4 = Math.max(f1,f2);
        System.out.print(f3+","+f4);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0.0,0.0
- b. Prints: 0.0,+0.0
- c. Prints: -0.0,0.0
- d. Prints: -0.0,+0.0
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 4

```
class J {
    static String m(int i) {return "int";}
    static String m(long i) {return "long";}
    static String m(float i) {return "float";}
    static String m(double i) {return "double";}
    public static void main (String[] args) {
        System.out.print(m(Math.random()));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: int
- b. Prints: long
- c. Prints: float
- d. Prints: double
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 5

```
class C {
    public static void main (String[] args) {
        double d = Long.MAX_VALUE;
        long l = Math.round(d + 1.0);
        System.out.print((l == 0) + ",");
        System.out.print((l == Long.MIN_VALUE) + ",");
        System.out.print(l == Long.MAX_VALUE);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: true,false,false
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 6

```
class E {
    public static void main (String[] args) {
        System.out.print(Math.sqrt(Math.exp(2)) == Math.E);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: true
- b. Prints: false
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 7

```
class J {
    public static void main (String[] args) {
        int a = Math.min(-0,+0);
        long b = Math.min(-0L,+0L);
        float c = Math.min(-0.0f,+0.0f);
        double d = Math.min(-0.0,+0.0);
        System.out.print(a+","+b+","+c+","+d);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,0,0.0,0.0
- b. Prints: -0,-0,-0.0,-0.0
- c. Prints: 0,0,-0.0,-0.0
- d. Compiler Error
- e. Runtime Error
- f. None of the Above

Question 8

```
class L {
    public static void main (String[] args) {
        System.out.print((Math.random() >= 0.0) + ",");
        System.out.print(Math.random() < 1.0);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true

- c. Prints: true,false
- d. Prints: true,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 9

```
class J {
    public static void main (String[] args) {
        byte x1 = 0;
        byte x2 = 1;
        byte a = Math.min(x1,x2); // 1
        short x3 = 0;
        short x4 = 1;
        short b = Math.min(x3,x4); // 2
        int c = Math.min(0,1); // 3
        long d = Math.min(0L,+1L); // 4
        System.out.print(a+","+b+","+c+","+d);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,0,0,0
- b. Compiler Error at 1
- c. Compiler Error at 2
- d. Compiler Error at 3
- e. Compiler Error at 4
- f. Runtime Error
- g. None of the Above

No.

정답 - 해설

Remark

1 c

Prints: 2.0,2.0

Math rint rounds the argument to the closest whole number. If the argument is equally close to two whole numbers then the even whole number is selected. The return type is a primitive double.

2 b

Prints: true

The magnitude of Integer.MIN_VALUE is one greater than the magnitude of Integer.MAX_VALUE. Therefore, it is not possible to use an int variable to correctly represent the absolute value of Integer.MIN_VALUE. The binary representation of Integer.MIN_VALUE is a single bit set to one followed by 31 bits set to zero. To calculate the two's complement of Integer.MIN_VALUE all of the bits must be inverted and then one must be added. When all of the bits are inverted the result is a single zero followed by 31 bits set to one. That is also the binary representation of Integer.MAX_VALUE. The final step of calculating the two's complement is to add one. The result is a single bit set to one followed by 31 bits set to zero. Once again, that is the binary representation of Integer.MIN_VALUE.

3 c

Prints: -0.0,0.0

The Math.min and Math.max methods are overloaded. This version of the Math.min method accepts two parameters of type float and returns the value that is

closer to negative infinity. This version of the Math.max method accepts two input parameters of type float and returns the value that is closer to positive infinity. Both Math.min and Math.max view -0.0 as being smaller than +0.0.

4 d

Prints: double

The Math.random method returns a value of type double. The value is always positive. The range of values is greater than or equal to zero and less than one.

5 b

Prints: false,false,true

The Math.round method is overloaded. If the argument is a float, then the return value is an int. If the argument is a double, then the return value is a long. If the argument is a double and is greater than or equal to Long.MAX_VALUE then the return value is Long.MAX_VALUE.

6 a

Prints: true

The Math.sqrt method calculates the square root of a number. The Math.exp method raises e to the power specified by the argument. Math.E is the value of e.

7 c

Prints: 0,0,-0.0,-0.0

The Math.min method is overloaded. There are versions that accept parameters of types int, long, float, and double. The float and double versions support the concept of negative and positive zero, and they view negative zero as being smaller than positive zero. The int and long versions ignore the sign of zero.

8 d

Prints: true,true

The Math.random method returns a primitive of type double. The value is equal to or greater than zero and less than 1.0.

9 b c

Compiler Error at 1 Compiler Error at 2

The Math.min method is overloaded. There are versions that accept parameters of types int, long, float, and double. The type of the return value is the same as the type of the input parameter. If a byte or a short is passed to the Math.min method as an input parameter, then it will be promoted to a type that is accepted by Math.min. The type that it is promoted to will depend on the type of the other parameter.

Question 1

```
class C {
    public static void main (String[] args) {
        int i1 = Math.round(0.5f);
        int i2 = Math.round(1.5f);
        System.out.print(i1 + "," + i2);
    }
}
```

```
}
```

What is the result of attempting to compile and run the program ?

- | | |
|----------------------|------------------|
| a. Prints: 0,1 | b. Prints: 0,2 |
| c. Prints: 1,1 | d. Prints: 1,2 |
| e. Compiler Error | f. Runtime Error |
| g. None of the Above | |

Question 2

```
class K {
    static String m(int i) {return "int";}
    static String m(long i) {return "long";}
    static String m(float i) {return "float";}
    static String m(double i) {return "double";}
    public static void main (String[] args) {
        System.out.print(m(Math.sin(0.0))+",");
        System.out.print(m(Math.cos(0.0))+",");
        System.out.print(m(Math.tan(0.0)));
    }
}
```

What is the result of attempting to compile and run the program ?

- | |
|---------------------------------|
| a. Prints: int,int,int |
| b. Prints: long,long,long |
| c. Prints: float,float,float |
| d. Prints: double,double,double |
| e. Compiler Error |
| f. Runtime Error |
| g. None of the Above |

Question 3

```
class J {
    public static void main (String[] args) {
        double d1 = -0.0;
        double d2 = +0.0;

        System.out.print(Math.min(d1,d2)+" "+Math.max(d1,d2));
    }
}
```

What is the result of attempting to compile and run the program ?

- | | |
|----------------------|----------------------|
| a. Prints: 0.0,0.0 | b. Prints: 0.0,+0.0 |
| c. Prints: -0.0,0.0 | d. Prints: -0.0,+0.0 |
| e. Compiler Error | f. Runtime Error |
| g. None of the Above | |

Question 4

```
class B {
    public static void main(String[] args) {
        double d1 = Math rint(0.5);
        double d2 = Math rint(1.5);
        System.out.print(d1 + "," + d2);
    }
}
```

What is the result of attempting to compile and run the program ?

- | | |
|--------------------|--------------------|
| a. Prints: 0.0,1.0 | b. Prints: 0.0,2.0 |
| c. Prints: 1.0,1.0 | d. Prints: 1.0,2.0 |

- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 5

```
class J {
    static String m(byte i) {return "byte";}
    static String m(int i) {return "int";}
    static String m(long i) {return "long";}
    static String m(double i) {return "double";}
    public static void main (String[] args) {
        byte b = 0;
        System.out.print(m(Math.min(b,b))+",");
        System.out.print(m(Math.min(b,1))+",");
        System.out.print(m(Math.min(b,1L)));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: byte,byte,byte
- b. Prints: byte,int,long
- c. Prints: int,int,int
- d. Prints: int,int,long
- e. Prints: long,long,long
- f. Compiler Error
- g. Runtime Error
- h. None of the Above

Question 6

```
class J {
    static String m(int i) {return "int";}
    static String m(long i) {return "long";}
    static String m(float i) {return "float";}
    static String m(double i) {return "double";}
    public static void main (String[] args) {
        long seed = 1L;
        System.out.print(m(Math.random(seed)));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: int
- b. Prints: long
- c. Prints: float
- d. Prints: double
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 7

```
class C {
    public static void main (String[] args) {
        float f = (float)(Integer.MIN_VALUE - 1.0);
        int i = Math.round(f);
        System.out.print((i == 0) + ",");
        System.out.print((i == Integer.MIN_VALUE) + ",");
        System.out.print(i == Integer.MAX_VALUE);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true

- c. Prints: false,true,false
- d. Prints: true,false,false
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 8

```
class C {
    public static void main (String[] args) {
        int i1 = Math.round(0.5f);
        int i2 = Math.round(1.5d);
        System.out.print(i1 + "," + i2);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,1
- b. Prints: 0,2
- c. Prints: 1,1
- d. Prints: 1,2
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

No.

정답 - 해설

Remark

1 d

Prints: 1,2

The result is calculated by adding 0.5 to the argument and then taking the Math.floor of the result. In this case, the input parameter is a float so the return value is an int. If the input parameter had been a double, then the return value would have been a long.

2 d

Prints: double,double,double

The Math.sin, Math.cos, and Math.tan methods return a value of type double.

3 c

Prints: -0.0,0.0

The Math.min and Math.max methods are overloaded. This version of the Math.min method accepts two parameters of type double and returns the value that is closer to negative infinity. This version of the Math.max method accepts two input parameters of type double and returns the value that is closer to positive infinity. Both Math.min and Math.max view -0.0 as being smaller than +0.0.

4 b

Prints: 0.0,2.0

Math rint rounds the argument to the closest whole number. If the argument is equally close to two whole numbers then the even whole number is selected. The return type is a primitive double.

5 d

Prints: int,int,long

The Math.min method is overloaded. There are versions that accept parameters of types int, long,

float, and double. The type of the return value is the same as the type of the input parameter. If a byte or a short is passed to the Math.min method as an input parameter, then it will be promoted to a type that is accepted by Math.min. The type that it is promoted to will depend on the type of the other parameter. If both parameters are of type byte, then both are promoted to an int. If one is of type byte and the other is of type int, then the byte is promoted to an int. If one is of type byte and the other is of type long, then the byte is promoted to a long.

6 e

Compiler Error

The Math.random method returns a value of type double. The value is always positive. The range of values is greater than or equal to zero and less than one. The Math.random method does not accept a seed value. If you need to use a seed, then use java.util.Random.

7 c

Prints: false,true,false

The Math.round method is overloaded. If the argument is a float, then the return value is an int. If the argument is a double, then the return value is a long. If the argument is a float and is less than or equal to Integer.MIN_VALUE then the return value is Integer.MIN_VALUE.

8 e

Compiler Error

There are two versions of the Math.round method. One accepts a float parameter and returns an int. The other accepts a double parameter and returns a long. A compiler error is generated here due to the attempt to assign the long return value to an int parameter.

Question 1

```
class D {
    public static void main(String[] args) {
        double d1 = Math.ceil(0.5);
        double d2 = Math.ceil(1.5);
        System.out.print(d1 + "," + d2);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0.0,1.0
- b. Prints: 0.0,2.0
- c. Prints: 1.0,1.0
- d. Prints: 1.0,2.0
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 2

```
class J {
    public static void main (String[] args) {
        float f1 = Math.min(-0.0,+0.0);
        float f2 = Math.max(-0.0,+0.0);
        System.out.print(f1+", "+f2);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0.0,0.0
- b. Prints: 0.0,+0.0
- c. Prints: -0.0,0.0
- d. Prints: -0.0,+0.0
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 3

```
class K {
    public static void main (String[] args) {
        double radians = Math.PI/2;
        double degrees = 90;
        System.out.print((Math.sin(radians)==1.0)+",");
        System.out.print((Math.sin(degrees)==1.0));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 4

```
class J {
    static String m(byte i) {return "byte";}
    static String m(short i) {return "short";}
    static String m(int i) {return "int";}
    static String m(long i) {return "long";}
    static String m(double i) {return "double";}
    public static void main (String[] args) {
        byte b = 0;
        short s = 0;
        System.out.print(m(Math.min(b,b))+",");
        System.out.print(m(Math.min(s,s))+",");
        System.out.print(m(Math.min(b,1)));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: byte,byte,byte
- b. Prints: short,short,short
- c. Prints: int,int,int
- d. Prints: byte,short,int
- e. Prints: short,short,int
- f. Compiler Error
- g. Runtime Error
- h. None of the Above

Question 5

```
class D {
    public static void main(String[] args) {
        double d1 = Math.floor(0.5);
        double d2 = Math.floor(1.5);
        System.out.print(d1 + "," + d2);
    }
}
```

What is the result of attempting to compile and run

the program ?

- a. Prints: 0.0,1.0
- b. Prints: 0.0,2.0
- c. Prints: 1.0,1.0
- d. Prints: 1.0,2.0
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 6

```
class B {  
    public static void main(String[] args) {  
        float f = Math rint(0.9);  
        System.out.print(f);  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0
- b. Prints: 1
- c. Prints: 1.0
- d. Compiler Error
- e. Runtime Error
- f. None of the Above

Question 7

```
class C {  
    public static void main (String[] args) {  
        System.out.print(Math.round(Float.NaN));  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: NaN
- b. Prints: 0.0
- c. Prints: 0
- d. Compiler Error
- e. Runtime Error
- f. None of the Above

Question 8

```
class E {  
    public static void main (String[] args) {  
        int a = -1;  
        long b = -2;  
        float c = -3.0f;  
        double d = -4.0;  
        a = Math.abs(a);  
        b = Math.abs(b);  
        c = Math.abs(c);  
        d = Math.abs(d);  
        System.out.print(a+b+c+d);  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 10.0
- b. Compiler Error
- c. Runtime Error
- d. None of the Above

No.
정답 - 해설
Remark

1 d
Prints: 1.0,2.0

The Math.ceil method accepts an argument of type double and returns a double. The returned value is the

smallest whole number that is greater than or equal to the argument.

2 e
Compiler Error

The Math.min and Math.max methods are overloaded. This version of the Math.min method accepts two parameters of type double and returns the value that is closer to negative infinity. The type of the return value is double. The attempt to assign the double return value to a variable of type float generates a compiler error. Similarly, This version of the Math.max method accepts two parameters of type double and returns the value that is closer to positive infinity. The type of the return value is double. The attempt to assign the double return value to a variable of type float generates a compiler error.

3 c
Prints: true,false

The Math.sin, Math.cos, and Math.tan methods all accept an input parameter of type double that represents an angle measured in radians. The return value of each method is of type double.

4 c
Prints: int,int,int

The Math.min method is overloaded. There are versions that accept parameters of types int, long, float, and double. The type of the return value is the same as the type of the input parameter. If a byte or a short is passed to the Math.min method as an input parameter, then it will be promoted to a type that is accepted by Math.min. The type that it is promoted to will depend on the type of the other parameter. If both parameters are of type byte, then both are promoted to an int. Similarly, if both parameters are of type short, then both are promoted to int. If one is of type byte or short and the other is of type int, then the byte or short is promoted to an int.

5 a
Prints: 0.0,1.0

The Math.floor method accepts an argument of type double and returns a double. The returned value is the largest whole number that is smaller than or equal to the argument.

6 d
Compiler Error

Math rint returns a double, so the attempt to assign the result to float causes a compiler error due to a possible loss of precision.

7 c
Prints: 0
If the input parameter is NaN, then the result is zero.

8 a
Prints: 10.0
The Math.abs method is overloaded. If the argument is of type int, then the return value is an int. If the

argument is of type long, then the return value is a long. If the argument is of type float, then the return value is a float. If the argument is of type double, then the return value is a double.

[Operators]

Question 1

```
class D {
    public static void main (String args[]) {
        int i1 = ~1;
        int i2 = -1;
        int i3 = -2;
        System.out.print(Integer.toHexString(i1) + ",");
        System.out.print(Integer.toHexString(i2) + ",");
        System.out.print(Integer.toHexString(i3));
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: ffffffff,fffffffe,fffffffe
- b. Prints: ffffffff,fffffffe,fffffffe
- c. Prints: ffffffff,fffffffe,fffffffe
- d. Prints: ffffffff,fffffffe,fffffffe
- e. Prints: ffffffffe,fffffffe,fffffffe
- f. Prints: ffffffffe,fffffffe,fffffffe
- g. Prints: ffffffffe,fffffffe,fffffffe
- h. Prints: ffffffffe,fffffffe,fffffffe
- i. Runtime error
- j. Compiler error
- k. None of the above

Question 2

```
class V {
    public static void main (String[] args) {
        System.out.print((0.0 % -1)+"", " + (-0.0 % 1)+"", " + (-0.0 % -1));
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 0.0,0.0,0.0
- b. Prints: 0.0,0.0,-0.0
- c. Prints: 0.0,-0.0,0.0
- d. Prints: 0.0,-0.0,-0.0
- e. Prints: -0.0,0.0,0.0
- f. Prints: -0.0,0.0,-0.0
- g. Prints: -0.0,-0.0,0.0
- h. Prints: -0.0,-0.0,-0.0
- i. Runtime error
- j. Compiler error
- k. None of the above

Question 3

```
class V {
    public static void main (String[] args) {
        System.out.print(Float.POSITIVE_INFINITY % 2 + ",");
        System.out.print(Float.NEGATIVE_INFINITY % 2 + ",");
    }
}
```

```
    },");
    System.out.print(2 % Float.NEGATIVE_INFINITY);
}
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: NaN,NaN,NaN
- b. Prints: NaN,NaN,2.0
- c. Prints: NaN,NaN,2
- d. Prints: 2.0,2.0,2.0
- e. Prints: 2,2,2
- f. Runtime error
- g. Compiler error
- h. None of the above

Question 4

```
class A {
    static int m(int i, String s) {
        System.out.print("["+s+", "+i+"]"+"",");
        return i;
    }
    public static void main (String[] args) {
        int i = 0;
        int a[] = new int[3];
        a[m(i++, "a")] = m(i++, "b");
        System.out.print(a[0]+"", "+a[1]"+", "+a[2] +", "+ i);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: [a,0],[b,1],1,0,0,2
- b. Prints: [a,1],[b,2],0,2,0,2
- c. Prints: [a,1],[b,0],0,0,0,2
- d. Prints: [a,2],[b,1],0,0,1,2
- e. Runtime error
- f. Compiler error
- g. None of the above

Question 5

```
class X {
    public static void main(String args[]) {
        double a = 1.0f / 3.0f;
        double b = 1.0d / 3.0f;
        double c = 1.0d / 3.0d;
        System.out.print((a==b) + ",");
        System.out.print((a==c) + ",");
        System.out.print(b==c);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true

- i. Runtime error
- j. Compiler error
- k. None of the above

Question 6

```
class L {
    static int m(int i) {
        System.out.print(i + " ");
        return i;
    }

    public static void main(String s[]) {
        m(m(1) - m(2) + m(3) * m(4));
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 1, 2, 3, 4, 8,
- b. Prints: 1, 2, 3, 4, 11,
- c. Prints: 3, 4, 1, 2, 11,
- d. Runtime error
- e. Compiler error
- f. None of the above

Question 7

```
class J {
    public static void main (String[] s) {
        byte b = 0;
        b += ~b >>> 1;
        System.out.println(b);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: -128
- b. Prints: -127
- c. Prints: -1
- d. Prints: 0
- e. Prints: 1
- f. Prints: 127
- g. Prints: 128
- h. Runtime Exception
- i. Compiler Error
- j. None of the Above

Question 8

```
class A {
    static int m(int i, String s) {
        System.out.print("[s+, "+i+" "+",");
        return i;
    }

    public static void main (String[] args) {
        int i = 0;
        int a[] = {0,1,2};
        a[m(i++, "a")] = a[m(i++, "b")] = a[m(i++, "c")];
        System.out.print(a[0]+", "+a[1]+", "+a[2]);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: [a,0],[b,1],[c,2],2,2,2
- b. Prints: [a,1],[b,2],0,2,0,2
- c. Prints: [a,1],[b,0],0,0,0,2
- d. Prints: [a,2],[b,1],0,0,1,2

- e. Runtime error
- f. Compiler error
- g. None of the above

Question 9

```
class Red {
    public static void main (String[] args) {
        byte a = 1, b = 2, c, d, e;
        c = (byte)a++;
        d = (byte)++b;
        e = (byte)a + b;
        System.out.print(c + d + e);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 1 2 3
- b. Prints: 6
- c. Prints: 2 3 5
- d. Prints: 10
- e. Prints: 1 3 4
- f. Prints: 8
- g. Prints: 1 3 5
- h. Prints: 9
- i. Runtime error.
- j. Compiler error.
- k. None of the Above

Question 10

```
class N {
    public static void main (String[] s) {
        byte b = 5;
        System.out.println(b<<32);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: -1
- b. Prints: 0
- c. Prints: 1
- d. Prints: 5
- e. Runtime error
- f. Compiler error
- g. None of the above

No.

정답 - 해설

Remark

1 f

Prints: ffffffff,ffffffff,ffffffff

2 d

Prints: 0.0,-0.0,-0.0

If the result is not NaN, then the sign of the result is equal to the sign of the dividend.

3 b

Prints: NaN,NaN,2.0

If the dividend is infinity then the result is NaN. If the dividend is finite and the divisor is infinity, then the result is the dividend.

4 a

Prints: [a,0],[b,1],1,0,0,2

The array index expression is evaluated before the right hand operand of the simple assignment operator.

Evaluation of the array index expression causes method `m` to be invoked. The input parameter is the postfix increment expression with the operand `i`. Since the postfix increment expression returns the original value of the operand, the value zero is passed to method `m` along with a `String` containing the character `a`. Method `m` returns the value that was passed in as an input parameter. In this case, the return value is zero. As a side effect of the postfix increment expression, the value of variable `i` is one after the array index expression is evaluated. The right hand operand of the simple assignment operator is the return value of method `m`. The input parameter to method `m` is the postfix increment expression with variable `i` as the operand. The value of variable `i` is now one and that is the value that is passed into method `m` and that is the value returned by method `m` and assigned to element zero of the array `a`. As a side effect of the postfix increment expression, the value of variable `i` is two after method `m` is invoked.

5 b

Prints: false,false,true

The result of dividing two float values is a float. When the float is converted to a double a rounding error is introduced. For that reason, value of variable `a` is different from the value of variables `b` and `c`.

6 b

Prints: 1, 2, 3, 4, 11,

Java evaluates expressions from left to right. The first operator encountered is the subtraction operator. First, the left operand is evaluated as one and then the right operand is evaluated as two and then the subtraction is performed resulting in a value of negative one. The next operand encountered is the addition operator. The value of the left operand is the result of the previous operation, `-1`. The value of the right operand will be the result of the expression `m(3) * m(4)` which is evaluated as 12. The addition operator adds the left hand operand, `-1`, to the right hand operand, 12, to produce a value of 11.

7 c

Prints: -1

If the left hand operand of the shift operator is of type `byte`, `short`, or `char` then the left operand is promoted to a 32 bit `int` and all four bytes are shifted. The result is of type `int`. The compound assignment operator includes an implicit cast to the type of the left hand operand. Therefore, the result of the shift operation, an `int`, is cast to a `byte` before the assignment to variable `b`. The result of the cast is the contents of the least significant byte. In this case, all eight bits are set to one so the result is a negative one in two's complement format.

8 a

Prints: [a,0],[b,1],[c,2],2,2,2

The question contains a series of simple assignment operators where all of the operands are array access expressions. Starting from the left, the array index

expression is evaluated before the right hand operand of the simple assignment operator. Evaluation of the array index expression causes method `m` to be invoked. The input parameter is the postfix increment expression with the operand `i`. Since the postfix increment expression returns the original value of the operand, the value zero is passed to method `m` along with a `String` containing the character `a`. Method `m` returns the value that was passed in as an input parameter. In this case, the return value is zero and method `m` prints `[a,0]`. As a side effect of the postfix increment expression, the value of variable `i` is one after the array index expression is evaluated. The right hand operand of the first simple assignment operator is evaluated next. The right hand operand is another array access expression similar to the first. The only difference is that the value of variable `i` is now one greater than it was at the time when the previous array access expression was evaluated. This time, method `m` returns one and prints `[b,1]`. The final array access expression is evaluated next. Method `m` returns 2 and prints `[c,2]`.

9 j

Compiler error.

The result of the addition of `a` and `b` is an `int`. The attempt to assign the `int` result to a `byte` variable `e` generates a possible loss of precision error. The precedence of the cast operator is higher than the precedence of the addition operator. Therefore, the cast applies only to variable `a` and not to the result of the addition.

10 d

Prints: 5

If the left hand operand of the shift operator is of type `byte`, `short`, or `char` then the left operand is promoted to an `int`. If the promoted type of the left hand operand is of type `int`, then the shift operator only uses the least significant five bits of the right hand operand and the actual shift distance is always less than 32 bits. Therefore, if the right hand operand is 32 then the shift distance is zero because all five of the least significant bits are zero. Note: If the type of the left hand operand is long, then the least significant six bits of the right hand operand are used.

Question 1

```
class E {
    public static void main (String args[]) {
        int i1 = 0xffffffff;
        int i2 = i1 << 1;
        int i3 = i1 >> 1;
        int i4 = i1 >>> 1;
        System.out.print(Integer.toHexString(i2) + ",");
        System.out.print(Integer.toHexString(i3) + ",");
        System.out.print(Integer.toHexString(i4));
    }
}
```

What is the result of attempting to compile and run

the above program ?

- a. Prints: ffffffff,ffffffff,ffffffff
- b. Prints: ffffffff,ffffffff,7fffffff
- c. Prints: ffffffff,7fffffff,ffffffff
- d. Prints: ffffffff,7fffffff,7fffffff
- e. Prints: ffffffff,ffffffff,ffffffff
- f. Prints: ffffffff,ffffffff,7fffffff
- g. Prints: ffffffff,7fffffff,ffffffff
- h. Prints: ffffffff,7fffffff,7fffffff
- i. Runtime error
- j. Compiler error
- k. None of the above

Question 2

```
class V {
    public static void main (String[] args) {
        float a = Float.POSITIVE_INFINITY;
        double b = Double.POSITIVE_INFINITY;
        double c = Double.NaN;
        System.out.print((a == b)+"",(c == c)+"",(c !=
c));
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Runtime error
- j. Compiler error
- k. None of the above

Question 3

```
class L {
    public static void main (String s[]) {
        int i = 1 | 2 ^ 3 >> 2 & 14 + 2;
        System.out.println(i);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 1
- b. Prints: 2
- c. Prints: 3
- d. Prints: 4
- e. Prints: 5
- f. Runtime error
- g. Compiler error
- h. None of the above

Question 4

```
class Color {}
class Red extends Color {}
class Blue extends Color {}
class A {
    public static void main (String[] args) {
        Color c1 = new Red();
        Color c2 = new Blue();
        Red r1 = new Red();
```

```
        boolean b1 = c1 instanceof Color;
        boolean b2 = c1 instanceof Blue;
        boolean b3 = r1 instanceof Blue;
        System.out.print(b1+" "+b2+" "+b3);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. false,false,false
- b. false,false,true
- c. false,true,false
- d. false,true,true
- e. true,false,false
- f. true,false,true
- g. true,true,false
- h. true,true,true
- i. Runtime error
- j. Compiler error
- k. None of the above

Question 5

```
class L {
    public static void main (String s[]) {
        int i = 1 | 2 ^ 3 * 2 & 13 | 2;
        System.out.println(i%5);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 1
- b. Prints: 2
- c. Prints: 3
- d. Prints: 4
- e. Prints: 5
- f. Runtime error
- g. Compiler error
- h. None of the above

Question 6

```
class M {
    static int m(int i) {
        System.out.print(i + " ");
        return i;
    }
    public static void main(String s[]) {
        int i=0;
        int j=0;
        System.out.print(m(++i) | m(++i) & m(++i) >>
m(++j) / m(++j) + m(++j));
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 1,2,3,4,5,6,1
- b. Prints: 1,2,3,1,2,3,1
- c. Prints: 1,2,3,4,5,6,2
- d. Prints: 1,2,3,1,2,3,2
- e. Prints: 1,2,3,4,5,6,3
- f. Prints: 1,2,3,1,2,3,3
- g. Runtime error
- h. Compiler error
- i. None of the above

Question 7

```
class U {
    public static void main(String args[]) {
        int a = 1;
        a += ++a + a++;
        System.out.print(a);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 3
- b. Prints: 4
- c. Prints: 5
- d. Prints: 6
- e. Prints: 7
- f. Runtime error
- g. Compiler error
- h. None of the above

Question 8

```
class B {
    static boolean m(boolean b) {
        System.out.print(b + " ");
        return b;
    }
    public static void main(String[] args) {
        boolean a = false;
        boolean b = false;
        boolean c = true;
        boolean d = false;

        m(m(a | b == c & d) == m((a | b) == (c & d)));
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: true false false
- b. Prints: false true false
- c. Prints: true true true
- d. Prints: false false true
- e. Runtime error.
- f. Compiler error.
- g. None of the Above

Question 9

```
class D {
    static int m1(int i, String s) {
        System.out.print(""+s+" "+i+" "+",");
        return i;
    }

    static int m2(int i, String s) {
        System.out.print(""+s+" "+i+" "+",");
        return 0;
    }
    public static void main (String[] args) {
        int i = 0;
        i = m1(++i,"A") + m2(i,"B") + m1(i++, "C") +
        m2(i,"D");
        System.out.print(i);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: [A,0],[B,1],[C,1],[D,2],1
- b. Prints: [A,1],[B,1],[C,1],[D,2],2
- c. Prints: [A,1],[B,1],[C,1],[D,2],3
- d. Prints: [A,1],[B,1],[C,1],[D,2],5
- e. Runtime error
- f. Compiler error
- g. None of the above

Question 10

```
class Q {
    static int m(int i) {
        System.out.print(i + ", ");
        return i;
    }

    public static void main(String s[]) {
        int i = 1;
        m(m(++i) + m(i++) + m(-i) + m(i++));
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 1, 2, 3, 4, 10,
- b. Prints: 1, 2, -3, 4, 4,
- c. Prints: 2, 2, -3, -3, -2,
- d. Prints: 2, 2, -3, 3, 4,
- e. Prints: 2, 3, -3, -2, 0,
- f. Prints: 2, 3, -3, 4, 6,
- g. Prints: 2, 3, 4, 5, 14,
- h. Runtime error
- i. Compiler error
- j. None of the above

No.

정답 - 해설

Remark

1 f

Prints: fffffffe,ffffff,7ffffff

The left shift operator << shifts each bit of the left operand to the left a distance specified by the right operand. The right operand specifies the shift distance. For each bit position of the shift distance a bit that is set to zero is shifted into the least significant bit. The least significant bit is the right most bit. The right shift operator >> shifts each bit of the left operand to the right a distance specified by the right operand. The right operand specifies the shift distance. For each bit position of the shift distance a copy of the sign bit is shifted to the right as each bit of the left operand is shifted to the right. As a result, the sign of the left operand remains unchanged by the shift operation. The sign bit is the left most bit of the left operand. The unsigned right shift operator >>> shifts each bit of the left operand to the right a distance specified by the right operand. The right operand specifies the shift distance. For each bit position of the shift distance a bit that is equal to zero is shifted into the sign bit as each bit of the left operand is shifted to the right. Therefore, the sign of the result is always positive.

2 f

Prints: true,false,true

Positive infinity is equal to positive infinity. NaN is not equal to anything including itself.

3 c

Prints: 3

Java evaluates expressions from left to right while respecting operator precedence. $(1 \mid 2 \wedge 3 \gg 2 \& 14 + 2)$, $(1 \mid (2 \wedge ((3 \gg 2) \& (14 + 2))))$, $(1 \mid (2 \wedge ((12) \& (16))))$, $(1 \mid (2 \wedge (0)))$, $(1 \mid 2)$, (3) ,

4 j

Compiler error

The expression used to set the value of variable b3, "r1 instanceof Blue" causes a compiler error because the type of variable r1, Red, can not be converted to type Blue. The expressions used to set variables b1 and b2 are legal. The first of the two would produce a value of true and the second would produce a value of false.

5 b

Prints: 2

Java evaluates expressions from left to right while respecting operator precedence. $(1 \mid 2 \wedge 3 * 2 \& 13 \mid 2)$, $(1 \mid (2 \wedge ((3 * 2) \& 13)) \mid 2)$, $(1 \mid (2 \wedge ((6) \& 13)) \mid 2)$, $(1 \mid (2 \wedge (4)) \mid 2)$, $(1 \mid 6 \mid 2 = 7)$, $(7\%5=2)$

6 b

Prints: 1,2,3,1,2,3,1

Java evaluates expressions from left to right while respecting operator precedence. $(1 \mid (2 \& (3 \gg ((1 / 2) + 3))))$, $(1 \mid (2 \& (3 \gg ((0) + 3))))$, $(1 \mid (2 \& (3 \gg (3))))$, $(1 \mid (2 \& (24)))$, $(1 \mid (0))=1$

7 c

Prints: 5

A compound assignment expression of the form $E1 \text{ op} = E2$ can be rewritten as $E1 = (T)((E1) \text{ op} (E2))$ where T is the type of E1. Therefore, the statements `int a=1; a+=++a+a++;` can be rewritten as `a=(int)((1)+(++a + a++))`. With further evaluation we have `a=(int)((1)+(2 + 2))`.

8 b

Prints: false true false

The equality operator has higher precedence than the logical operators; therefore, the left side, $(a \mid b == c \& d)$, is evaluated as $(a \mid (b == c) \& d)$. Substituting the actual boolean values for the variable names produces $(\text{false} \mid (\text{false} == \text{true}) \& \text{false})$ and evaluates to false.

9 b

Prints: [A,1],[B,1],[C,1],[D,2],2

The statement contains a series of addition operations. The left operand of each addition operator is evaluated completely before the right hand operand. The left hand operand of the first addition operator is the return value of method m1. A prefix increment expression is passed to m1 as an input parameter. The result of the prefix expression is the original value of i plus one. The right hand operand of the addition operator is the return value of method m2. Method m2 prints the current value of i which is one and then m2 always returns zero. The result of the

first addition operation becomes the left operand of the second addition operator. The right operand is the return value of method m1. The input parameter of m1 is a postfix increment expression. The result of the postfix increment expression is the existing value of i which is one and that is the value that is passed to m1 as the input parameter. The side effect of the postfix increment expression is the addition of one to the value of i. Therefore, parameter i has the value of two after method m1 is invoked. The result of the second addition operation is the value 2. The result of the second addition operation is the left hand operand of the third addition operation. Since method m2 always returns zero, the third addition operation does not change the final result which is two.

10 d

Prints: 2, 2, -3, 3, 4,

Question 1

```
class A {
    static int m(int i) {
        System.out.print(i + ",");
        return 0;
    }
    public static void main (String[] args) {
        int i = 0;
        i = i++ + m(i);
        System.out.print(i);
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 0,0
- b. Prints: 1,0
- c. Prints: 0,1
- d. Prints: 1,1
- e. Runtime error
- f. Compiler error
- g. None of the above

Question 2

```
class W {
    public static void main (String[] args) {
        byte x = 3;
        byte y = 5;
        System.out.print((-x == ~x + 1)+" "+(-y == ~y + 1));
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Runtime error
- f. Compiler error
- g. None of the above

Question 3

```
class B {
    static int x;
    static int m(int i) {
        System.out.print(i + ",");
    }
}
```

```

    x += i;
    return 0;
}
public static void main (String[] args) {
    int i = 0;
    int j = 0;
    int k = 0;
    i = j = k = i++ + m(i) + j++ + m(j) + k++ + m(k);
    System.out.print(i + j + k + "," + x);
}
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|----------------------|----------------------|
| a. Prints: 0,0,0,3,0 | b. Prints: 1,1,1,0,3 |
| c. Prints: 1,1,1,3,3 | d. Prints: 1,1,1,6,3 |
| e. Prints: 1,1,1,9,3 | f. Prints: 1,1 |
| g. Runtime error | h. Compiler error |
| i. None of the above | |

Question 4

```

class P{
    public static void main (String[] s) {
        byte b = 5;
        System.out.println(b<<33);
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|-------------------|----------------------|
| a. Prints: -1 | b. Prints: 0 |
| c. Prints: 1 | d. Prints: 5 |
| e. Prints: 10 | f. Runtime error |
| g. Compiler error | h. None of the above |

Question 5

```

class A {
    public static void main (String[] args) {
        System.out.print((new Object() instanceof Object)+",");
        System.out.print((new Object() instanceof String)+",");
        System.out.print((new String() instanceof Object));
    }
}

```

What is the result of attempting to compile and run the above program ?

- Prints: false,false,false
- Prints: false,false,true
- Prints: false,true,false
- Prints: false,true,true
- Prints: true,false,false
- Prints: true,false,true
- Prints: true,true,false
- Prints: true,true,true
- Runtime error
- Compiler error
- None of the above

Question 6

```

class K {

```

```

    public static void main (String[] s) {
        byte b = 127;
        b <<= 2;
        System.out.println(b);
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|-------------------|----------------------|
| a. Prints: -4 | b. Prints: -3 |
| c. Prints: -2 | d. Prints: 0 |
| e. Prints: 1 | f. Prints: 127 |
| g. Prints: 508 | h. Runtime Exception |
| i. Compiler Error | j. None of the Above |

Question 7

```

class M {
    static int m(int i) {
        System.out.print(i + ", ");
        return i;
    }

    public static void main(String s[]) {
        int i=0;
        System.out.print(m(++i) | m(++i) & m(++i) +
m(++i) % m(++i));
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|------------------------|------------------------|
| a. Prints: 1,2,3,4,5,1 | b. Prints: 1,2,3,4,5,2 |
| c. Prints: 1,2,3,4,5,3 | d. Prints: 1,2,3,4,5,4 |
| e. Prints: 1,2,3,4,5,5 | f. Prints: 1,2,3,4,5,6 |
| g. Runtime error | h. Compiler error |
| i. None of the above | |

Question 8

```

class T {
    public static void main(String args[]) {
        String a = "1";
        byte b = 2;
        short c = 3;
        char d = 4;
        int e = 5;
        float f = 2;
        a += b *= c += d *= e += f;
        System.out.print(a);
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|-------------------|----------------------|
| a. Prints: 57 | b. Prints: 63 |
| c. Prints: 162 | d. Runtime error |
| e. Compiler error | f. None of the above |

Question 9

```

class F {
    public static void main (String []s) {
        int i = 1;
        i += ~i - -i * ++i + i-- % ++i * i++;
    }
}

```

```

        System.out.print(i);
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: -1
- b. Prints: 0
- c. Prints: 1
- d. Prints: 2
- e. Prints: 3
- f. Prints: 4
- g. Runtime error
- h. Compiler error
- i. None of the above

Question 10

```

class R {
    static int m(int i) {
        System.out.print(i + ", ");
        return i;
    }

    public static void main(String s[]) {
        m(m(~1) + m(1|2) + m(1&2) + m(1^3) +
m(1<<1));
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: -2, 3, 0, 3, 0, 6
- b. Prints: -2, 3, 0, 2, 1, 4
- c. Prints: -2, 3, 0, 2, 2, 5
- d. Prints: -2, 3, 0, 3, 2, 6
- e. Prints: -1, 3, 0, 3, 2, 7
- f. Prints: -2, 0, 3, 3, 0, 6
- g. Prints: -1, 0, 3, 2, 1, 4
- h. Prints: -2, 0, 3, 2, 2, 5
- i. Prints: -2, 0, 3, 3, 2, 6
- j. Prints: -1, 0, 3, 3, 2, 7
- k. Runtime error
- l. Compiler error
- m. None of the above

No.

정답 - 해설

Remark

1 b

Prints: 1,0

The statement contains an addition operation. The left operand is a postfix operation and it is evaluated first. The result of the postfix operation is zero, but variable i is incremented as a side effect. The right hand operand of the addition operation is the result of method m. As a side effect, method m prints the current value of variable i which is one. Method m then returns the value zero. The result of the addition operation is zero and that is the value that is assigned to variable i.

2 d

Prints: true,true

The sign of a number is changed by inverting all of

the bits and by adding one.

3 b

Prints: 1,1,1,0,3

The statement contains a series of addition operations. The left operand of each addition operator is evaluated completely before the right hand operand. The left hand operand of the first addition operator is a postfix increment expression that increments variable i. The result of the postfix expression is the original value of i. The side effect of the postfix operation is the addition of one to the value of variable i. The right hand operand of the addition operator is the return value of method m. Method m prints the current value of i which is one. Method m then adds the value of i to the value of x. The result is x equals one. Then m returns zero. The result of the addition operation is the sum of the original value of i which is zero and the return value of m which is also zero. The left hand operand of the next addition operator is the result of the first addition. The value of the right operand is the result of the postfix expression that increments the value of variable j. The result of the addition operation is then the sum of zero and the original value of j which is also zero. The rest of the expression is evaluated similarly. The final result of the statement is that the value of zero is assigned to variables i, j, and k. The value of x is sum of the side effects of each postfix expression.

4 e

Prints: 10

If the left hand operand of the shift operator is of type byte, short, or char then the left operand is promoted to an int. If the promoted type of the left hand operand is of type int, then the shift operator only uses the least significant five bits of the right hand operand and the actual shift distance is always less than 32 bits. Therefore, if the right hand operand is 33 then the shift distance is one bit because the five least significant bits are 00001. Note: If the type of the left hand operand is long, then the least significant six bits of the right hand operand are used.

5 f

Prints: true,false,true

The left operand of the "instanceof" operator must be a reference to an instance of an Object or subclass of Object. The right operand of the "instanceof" operator must be an Object type. If the left operand is a reference to an instance of the type specified by the right operand or if the left operand is a reference to an instance of a subclass of the type specified by the right operand, then instanceof returns true.

6 a

Prints: -4

If the left hand operand of the shift operator is of type byte, short, or char then the left operand is promoted to a 32 bit int. When a variable of type int with a value of 127 is shifted to the left two bits, the result is 508. However, the compound assignment

operator includes an implicit cast to the type of the left hand operand. $E1 \text{ op} = E2$ is equivalent to $E1 = (T)((E1) \text{ op } (E2))$, where T is the type of the left hand operand. Therefore, when 508 is cast to an eight bit byte, the result is negative four.

7 c

Prints: 1,2,3,4,5,3

Java evaluates expressions from left to right while respecting operator precedence. $(1 \mid (2 \& (3 + (4 \% 5))))$, $(1 \mid (2 \& (3 + (4))))$, $(1 \mid (2 \& (7))))$, $(1 \mid 2) = 3$

8 c

Prints: 162

The compound assignment operator is right associative so the expression is evaluated from right to left.

9 c

Prints: 1

10 c

Prints: -2, 3, 0, 2, 2, 5

Question 1

```
class L {
    public static void main (String s[]) {
        int i = 1 + 2 | 1 << 2 ^ 1 | 1;
        System.out.println(i%5);
    }
}
```

What is the result of attempting to compile and run the above program ?

- | | |
|-------------------|----------------------|
| a. Prints: 1 | b. Prints: 2 |
| c. Prints: 3 | d. Prints: 4 |
| e. Prints: 5 | f. Runtime error |
| g. Compiler error | h. None of the above |

Question 2

```
class V {
    public static void main (String[] args) {
        System.out.print(Float.NaN % 2 + ",");
        System.out.print(2 % Float.NaN);
    }
}
```

What is the result of attempting to compile and run the above program ?

- | | |
|----------------------|--------------------|
| a. Prints: NaN,NaN | b. Prints: NaN,2.0 |
| c. Prints: 2.0,NaN | d. Prints: 2.0,2.0 |
| e. Runtime error | f. Compiler error |
| g. None of the above | |

Question 3

```
class M {
    static int m(int i) {
        System.out.print(i + ", ");
        return i;
    }

    public static void main(String s[]) {
```

```
        int i = 1;
        int j = m(i++) + m(i++) * m(i++) + m(i++);
        System.out.print(j % 5);
    }
}
```

What is the result of attempting to compile and run the above program ?

- | | |
|----------------------|----------------------|
| a. Prints: 1,2,3,4,1 | b. Prints: 1,2,3,4,2 |
| c. Prints: 1,2,3,4,3 | d. Prints: 1,2,3,4,4 |
| e. Prints: 1,2,3,4,5 | f. Runtime error |
| g. Compiler error | h. None of the above |

Question 4

```
class V {
    public static void main (String[] args) {
        System.out.print((0.0 * -0.0)+"", " + (-0.0 * -0.0) + ",");
        System.out.print((0.0 + -0.0));
    }
}
```

What is the result of attempting to compile and run the above program ?

- | |
|---------------------------|
| a. Prints: 0.0,0.0,0.0 |
| b. Prints: 0.0,0.0,-0.0 |
| c. Prints: 0.0,-0.0,0.0 |
| d. Prints: 0.0,-0.0,-0.0 |
| e. Prints: -0.0,0.0,0.0 |
| f. Prints: -0.0,0.0,-0.0 |
| g. Prints: -0.0,-0.0,0.0 |
| h. Prints: -0.0,-0.0,-0.0 |
| i. Runtime error |
| j. Compiler error |
| k. None of the above |

Question 5

```
class L {
    public static void main (String s[]) {
        int i = 1 | 2 ^ 15 & 7 ^ 13 | 2;
        System.out.println(i % 5);
    }
}
```

What is the result of attempting to compile and run the above program ?

- | | |
|-------------------|----------------------|
| a. Prints: 1 | b. Prints: 2 |
| c. Prints: 3 | d. Prints: 4 |
| e. Prints: 5 | f. Runtime error |
| g. Compiler error | h. None of the above |

Question 6

```
class M {
    static int m(int i) {
        System.out.print(i + ", ");
        return i;
    }

    public static void main(String s[]) {
        m(m(1) + m(2) % m(3) * m(4));
    }
}
```


What is the result of attempting to compile and run the above program ?

- a. Prints: 1, 2, 3, 4, 0,
- b. Prints: 1, 2, 3, 4, 3,
- c. Prints: 1, 2, 3, 4, 9,
- d. Prints: 1, 2, 3, 4, 12,
- e. Prints: 2, 3, 4, 1, 9,
- f. Prints: 2, 3, 4, 1, 3,
- g. Runtime error
- h. Compiler error
- i. None of the above

Question 7

```
class S {  
    public static void main(String args[]) {  
        byte b = -1,c,d;  
        c = (byte)(b >>> 1);  
        d = (byte)(b >>> 25);  
        System.out.print(c+", "+d);  
    }  
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 127,0
- b. Prints: 0,127
- c. Prints: -1,0
- d. Prints: -1,127
- e. Prints: -1,-1
- f. Prints: 127,127
- g. Runtime error
- h. Compiler error
- i. None of the above

Question 8

```
class M {  
    static int m(int i) {  
        System.out.print(i + " , ");  
        return i;  
    }  
  
    public static void main(String s[]) {  
        int i=0;  
        int j = m(++i) + m(++i) * m(++i) % m(++i) +  
m(++i);  
        System.out.print(j%5);  
    }  
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: 1,2,3,4,5,1
- b. Prints: 1,2,3,4,5,2
- c. Prints: 1,2,3,4,5,3
- d. Prints: 1,2,3,4,5,4
- e. Prints: 1,2,3,4,5,5
- f. Runtime error
- g. Compiler error
- h. None of the above

Question 9

```
class C {  
    static int m(int i) {  
        System.out.print(i + " , ");  
        return i;  
    }  
  
    public static void main(String s[]) {  
        int i = 1;  
        m(m(++i) - m(i++) + ~m(-i) * -m(~i));  
    }  
}
```

What is the result of attempting to compile and run

the above program ?

- a. Prints: 2, 2, -3, -4, 8,
- b. Prints: 2, 3, -3, -4, 7,
- c. Prints: 1, 1, 1, 1, 0,
- d. Prints: 2, 2, -2, -2, 4,
- e. Prints: 2, 3, -2, -2, 3,
- f. Prints: -1, -2, 2, 2, 0,
- g. Runtime error
- h. Compiler error
- i. None of the above

No.

정답 - 해설

Remark

1 b

Prints: 2

Java evaluates expressions from left to right while respecting operator precedence. (1 + 2 | 1 << 2 ^ 1 | 1), ((1 + 2) | ((1 << 2) ^ 1) | 1), ((3) | ((1 << 2) ^ 1) | 1), ((3) | ((4) ^ 1) | 1), ((3) | (5) | 1 == 7), (7%5=2)

2 a

Prints: NaN,NaN

If either operand is NaN then the result is NaN.

3 a

Prints: 1,2,3,4,1

Java evaluates expressions from left to right while respecting operator precedence. j = 1 + (2 * 3) + 4 == 11

4 e

Prints: -0.0,0.0,0.0

5 a

Prints: 1

Java evaluates expressions from left to right while respecting operator precedence. (1 | ((2 ^ (15 & 7)) ^ 13) | 2), (1 | ((2 ^ (7)) ^ 13) | 2), (1 | ((5) ^ 13) | 2), ((1 | 8 | 2)==11), (11%5=1)

6 c

Prints: 1, 2, 3, 4, 9,

Java evaluates expressions from left to right. The first operator encountered is the addition operator. First, the left operand is evaluated as 1. The value of the right operand will be the result of the expression m(2) % m(3) * m(4). When Java encounters the remainder operator, "%", it first evaluates the left hand operand, 2, and then the right, 3. The result is 2. The next operator is the multiplication operator. The left operand is the result of the previous operation, 2. The right operand is 4. The result is 8. Java then adds the left operand, 1, of the addition operator to the value of the right operand, 8. The result is 9.

7 d

Prints: -1,127

If the left hand operand of the shift operator is of

type byte, short, or char then the left operand is promoted to an int. Therefore, the result of shifting a byte to the right less than $32 - 7 = 25$ bits will not be obvious if the result is viewed as a byte.

8 c

Prints: 1,2,3,4,5,3

Java evaluates expressions from left to right while respecting operator precedence. $j = 1 + (((2 * 3) \% 4) + 5)$

9 a

Prints: 2, 2, -3, -4, 8,

Integral primitives are stored in two's complement format. To change a positive value into a negative value, invert each bit and add one to the result. The "~" operator is the bitwise complement operator. It inverts the value of each bit. For example, $\sim 0 = -1$.

[String]

Question 1

```
class B {
    public static void main(String[] args) {
        String s1 = "ABCDE";
        System.out.print(s1.indexOf("B")+",");
        System.out.print(s1.indexOf("F")+",");
        System.out.print(s1.indexOf(""));
    }
}
```

What is the result of attempting to compile the program ?

- a. Prints: 1,0,0
- b. Prints: 1,0,-1
- c. Prints: 1,-1,0
- d. Prints: 1,-1,-1
- e. Prints: 2,0,0
- f. Prints: 2,0,-1
- g. Prints: 2,-1,0
- h. Prints: 2,1,-1
- i. Compiler error
- j. Run time error
- k. None of the Above

Question 2

```
class Black {
    public static void main(String args[]) {
        String a = "A";
        String b = "B";
        String c = a+b;
        String d = a+b;
        System.out.print(((a+b)==(a+b)) + ",");
        System.out.print((c==d) + ",");
        System.out.print(c.equals(d));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false

h. Prints: true,true,true

Question 3

```
class A {
    public static void main(String[] args) {
        String s1 = "A";
        String s2 = "a";
        String s3 = "b";
        s1.toLowerCase();
        s3.replace('b','a');
        System.out.print((s1.equals(s2)) + " " +
            (s2.equals(s3)));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 4

```
class Green {
    public static void main(String args[]) {
        final String a = "A";
        final String b = "B";
        String c = a+b;
        String d = a+b;
        System.out.print((c==c) + ",");
        System.out.print(((a+b)==(a+b)) + ",");
        System.out.print(c==d);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Compiler Error
- j. Runtime Error
- k. None of the Above

Question 5

```
class B {
    public static void main(String[] args) {
        String s1 = " B C D E ";
        System.out.print('A' + s1.trim() + 'F');
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: ABCDEF
- b. Prints: A B C D E F
- c. Prints: A BCDEF
- d. Prints: ABCDE F
- e. Prints: AB C D EF
- f. Compiler Error

g. Runtime Error

h. None of the Above

Question 6

```
class D {
    public static void main (String[] args) {
        System.out.print("CC".compareTo("C") + ",");
        System.out.print("CC".compareTo("CC") + ",");
        System.out.print("CC".compareTo("CCC"));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,true,false
- b. Prints: 1,0,-1
- c. Prints: -1,0,1
- d. Compiler Error
- e. Runtime Error
- f. None of the Above

Question 7

```
class Blue {
    public static void main(String args[]) {
        String a = "A";
        String b = "B";
        final String c = a+b;
        final String d = a+b;
        System.out.print((c==c) + ",");
        System.out.print(((a+b)==(a+b)) + ",");
        System.out.print(c==d);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Compiler Error
- j. Runtime Error
- k. None of the Above

Question 8

```
class G {
    public static void main (String[] args) {
        char[] c = {'a','b','c','d'};
        String s1 = String.valueOf(c);
        boolean b = false;
        for (int i = 0; i < s1.length; i++) {
            b |= c[i] == s1.charAt(i);
        }
        System.out.print(b);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false
- b. Prints: true
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 9

```
class Yellow {
    public static void main(String args[]) {
        String a = "A";
        String b = "B";
        String c = a+b;
        String d = a+b;
        System.out.print((c==c) + ",");
        System.out.print(((a+b)==(a+b)) + ",");
        System.out.print(c==d);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Compiler Error
- j. Runtime Error
- k. None of the Above

No.

정답 - 해설

Remark

1 c

Prints: 1,-1,0

String.indexOf returns negative one if the argument is not found in the String. Zero is returned if the argument is an empty String.

2 b

Prints: false,false,true

The expression (a+b) is evaluated at runtime and produces a new instance of a String Object. Even though the expression (a+b) produces the same result each time it is evaluated, each evaluation of the expression produces a new instance of the String. Since the equality operator compares references and not content values, the equality operator returns false. The equals method compares content values so it returns true.

3 a

Prints: false,false

Strings are immutable. String methods such as String.toLowerCase and String.replace create and return a new immutable String. In this case, the new String Object was ignored.

4 h

Prints: true,true,true

The key to understanding this program is understanding the impact of declaring variables "a" and "b" with the "final" modifier. Since "a" and "b" are final, all of the String expressions become compile-time constant expressions that evaluate to "AB". As a result, all are represented by a single String Object at runtime. Therefore, the equality

operator finds that the references are equal and returns the value true.

5 e

Prints: AB C D EF

The trim method creates a new String Object with the leading and trailing white space removed.

6 b

Prints: 1,0,-1

The String.compareTo method returns an integer. The returned value is zero if both strings are equal. Otherwise, a negative or positive value is returned depending on which string precedes the other when the two are compared character-by-character according to Unicode ordering.

7 e

Prints: true,false,false

The expression (a+b) is evaluated at runtime and produces a new instance of a String Object. Even though the expression (a+b) produces the same result each time it is evaluated, each evaluation of the expression produces a new instance of the String. Since the equality operator compares the references to objects rather than the contents of the objects, the equality operator returns the value false. The declaration of variables "c" and "d" with the final modifier does not transform the expressions into compile time constants. Instead, it just prevents variables "c" and "d" from being changed after initialization at runtime. However, if variables "a" and "b" had been declared final then the result would be different.

8 c

Compiler Error

A compiler error is generated due to the attempt to access the length method of the String class as though it were a variable.

9 e

Prints: true,false,false

The expression (a+b) is evaluated at runtime and produces a new instance of a String Object. Even though the expression (a+b) produces the same result each time it is evaluated, each evaluation of the expression produces a new instance of the String. Since the equality operator compares the references to objects rather than the contents of the objects, the equality operator returns the value false.

Question 1

```
1. class A {
2.     public static void main(String[] args) {
3.         byte[] b = {'a', 'b', 'c'};
4.         char[] c = {'a', 'b', 'c'};
5.         String s = "abc";
6.         StringBuffer sb = new StringBuffer("abc");
7.         byte b2 = 'a';
8.         char c2 = 'a';
9.         String s1 = new String(b);
```

```
10.        String s2 = new String(c);
11.        String s3 = new String(s);
12.        String s4 = new String(sb);
13.        String s5 = new String(b2);
14.        String s6 = new String(c2);
15.    }
16. }
```

What is the result of attempting to compile the program ?

- a. Compiler error at line 9.
- b. Compiler error at line 10.
- c. Compiler error at line 11.
- d. Compiler error at line 12.
- e. Compiler error at line 13.
- f. Compiler error at line 14.
- g. None of the Above

Question 2

```
class B {
    public static void main(String[] args) {
        String s1 = "ABCDE";
        System.out.print(s1.indexOf("F")+","");
        System.out.print(s1.indexOf(""));
        System.out.print(s1.indexOf(''));
    }
}
```

What is the result of attempting to compile the program ?

- a. Prints: 0,0,0
- b. Prints: 0,0,-1
- c. Prints: 0,-1,0
- d. Prints: 0,-1,-1
- e. Prints: -1,0,0
- f. Prints: -1,0,-1
- g. Prints: -1,-1,0
- h. Prints: -1,1,-1
- i. Compiler error
- j. Run time error
- k. None of the Above

Question 3

```
class Orange {
    public static void main(String args[]) {
        String a = "A";
        String b = "B";
        String c = "AB";
        System.out.print(((("A"+"B")=="AB") + ","));
        System.out.print(("A"+"B")==c);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 4

```
class B {
    public static void main(String[] args) {
        String s1 = "ABCDE";
        System.out.print(s1.indexOf('A')+","");
        System.out.print(s1.indexOf("A")+","");
        System.out.print(s1.indexOf("A",1));
```

```

    }
}

```

What is the result of attempting to compile the program ?

- a. Prints: -1,-1,-1
- b. Prints: 0,0,0
- c. Prints: 0,0,-1
- d. Prints: 1,1,0
- e. Prints: 1,1,-1
- f. Prints: 1,1,1
- g. Compiler error
- h. Run time error
- i. None of the Above

Question 5

```

class White {
    public static void main(String args[]) {
        String a = "A";
        String b = "B";
        String c = a+b;
        System.out.print(((a+b)==(a+b)) + ",");
        System.out.print(c.intern()=="A"+"B");
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 6

```

class Red {
    public static void main(String args[]) {
        String a = "A";
        String b = "B";
        System.out.print(((a+b)=="AB") + ",");
        System.out.print(("A"+"B")==(a+b));
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 7

```

class Violet {
    public static void main(String args[]) {
        String a = "A";
        String b = "B";
        String c = a+b;
        String d = a+b;
        System.out.print(((a+b)==(a+b)) + ",");
        System.out.print(c.intern()==d.intern());
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true

- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 8

```

class E {
    public static void main (String[] args) {
        String s1 = "ABCDE";
        System.out.print(s1.substring(1,2)+s1.substring(3));
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: AABC
- b. Prints: ACDE
- c. Prints: ABABC
- d. Prints: ABCDE
- e. Prints: BABCD
- f. Prints: BDE
- g. Prints: BCABCD
- h. Prints: BCDE
- i. Compiler Error
- j. Runtime Error
- k. None of the Above

Question 9

```

class F {
    public static void main (String[] args) {
        System.out.print(String.valueOf(1) +
        String.valueOf(2));
        String s1 = "S1";
        String s2 = s1.toString();
        System.out.print(", " + (s1==s2));
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 3,false
- b. Prints: 3,true
- c. Prints: 12,false
- d. Prints: 12,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

No.

정답 - 해설

Remark

1 e f

Compiler error at line 13. Compiler error at line 14.

The overloaded constructors for the String class accept a parameter of type String or StringBuffer or an array of byte or char. There is no constructor that will accept a primitive byte or char. Please note that if you want to convert a primitive to a String then you can use the static String.valueOf method.

2 i

Compiler error

The String.indexOf method will accept an empty String, but not an empty char literal.

3 d

Prints: true,true

The String literals ("A"+"B") and ("AB") both refer to the same String Object in memory; therefore, the equality operator returns true when the literals are

compared. The variable `c` refers to the same String object in memory.

4 c

Prints: 0,0,-1

The overloaded `String.indexOf` methods accept either `char` or `String` parameters. An optional offset parameter may be used to specify the starting index for the search. In this case, the starting index is one, but the index of the `A` character is zero. As a result, negative one is returned.

5 b

Prints: false,true

The expression `(a+b)` is evaluated at runtime and produces a new instance of a String Object. Even though the expression `(a+b)` produces the same result each time it is evaluated, each evaluation of the expression produces a new instance of the String. The `String.intern()` method causes the JVM to share a previously existing String object if one is found that contains the same value. The expression `("A"+"B")` is a compile-time constant and therefore its String object can be shared with other compile-time constant Strings with the same value or with Strings that have had their `intern` method invoked.

6 c

Prints: true,false

The String literals `("A"+"B")` and `("AB")` are evaluated at compile time and found to be equal. Therefore, both share the same String Object at runtime. When the two literals are compared at runtime using the equality operator, they are found to refer to the same object. Therefore, the equality operator returns the value `true` because the references are identical. In contrast, the expression `(a+b)` is evaluated at runtime and produces a new instance of a String Object that contains `"AB"`. Therefore, the equality operator returns `false` because the object references are not identical.

7 b

Prints: false,true

The expression `(a+b)` is evaluated at runtime and produces a new instance of a String Object. Even though the expression `(a+b)` produces the same result each time it is evaluated, each evaluation of the expression produces a new instance of the String. The `String.intern()` method causes the JVM to share a previously existing String object if one is found that contains the same value.

8 f

Prints: BDE

The `substring` method returns a new string that is a substring of the original string. The single parameter form creates a new string that begins at the index specified by the parameter. The two parameter form creates a substring that starts at the index of the first parameter and ends at the index of the second parameter. The character at the start index is included in the substring but the character at the end index is

not.

9 d

Prints: 12,true

The `valueOf` method returns a String representation of the input parameter. The input parameter may be of type `Object`, `char` array, or any primitive type. The `toString` method returns a reference to the String object. It does not create a new instance of the String.

[String Buffer]

Question 1

```
class R{
    public static void main (String[] args) {
        StringBuffer s1 = new StringBuffer("ABCDEFGH");
        s1.setCharAt(7,'H');
        System.out.print(s1 + "," + s1.length());
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: ABCDEFGH
- b. Prints: ABCDEFH
- c. Compiler Error
- d. Run time Error
- e. None of the Above

Question 2

```
class H {
    public static void main (String[] args) {
        String s1 = "ABC";
        StringBuffer s2 = new StringBuffer(s1);
        System.out.print(s2.equals(s1) + "," +
s1.equals(s2));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compiler Error
- f. Run time Error
- g. None of the Above

Question 3

```
class I {
    public static void main (String[] args) {
        StringBuffer s1 = new StringBuffer();
        System.out.print(s1.length() + "," + s1.capacity());
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,0
- b. Prints: 0,8
- c. Prints: 8,0
- d. Prints: 8,8
- e. Prints: 0,16
- f. Prints: 16,0
- g. Prints: 16,16
- h. Compiler Error
- i. Run time Error
- j. None of the Above

Question 4

```
class K {  
    public static void main (String[] args) {  
        StringBuffer s1 = new StringBuffer("ABCDEFGF");  
        System.out.print(s1.capacity());  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 7
- b. Prints: 8
- c. Prints: 16
- d. Prints: 32
- e. Compiler Error
- f. Run time Error
- g. None of the Above

Question 5

```
class Q{  
    public static void main (String[] args) {  
        StringBuffer s1 = new StringBuffer();  
        s1.append(new Boolean(true));  
        s1.insert(1, new Boolean(false));  
        System.out.print(s1);  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: falsefalse
- b. Prints: falsetrue
- c. Prints: truefalse
- d. Prints: truetrue
- e. Compiler Error
- f. Run time Error
- g. None of the Above

Question 6

```
class L{  
    public static void main (String[] args) {  
        StringBuffer s1 = new StringBuffer();  
        s1.ensureCapacity(16);  
        s1.append("ABCDEFGF");  
        s1.setLength(5);  
        System.out.print(s1 + "," + s1.capacity());  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: ABCDEFG,16
- b. Prints: ABCDEFG,5
- c. Prints: ABCDE,16
- d. Prints: ABCDE,5
- e. Prints: 32
- f. Compiler Error
- g. Run time Error
- h. None of the Above

Question 7

```
class M {  
    public static void main (String[] args) {  
        StringBuffer s1 = new StringBuffer();  
        s1.ensureCapacity(16);  
        s1.append("ABCDEFGF");  
        s1.setLength(100);  
        System.out.print(s1.length() + "," + s1.capacity());  
    }  
}
```

What is the result of attempting to compile and run

the program ?

- a. Prints: 7,16
- b. Prints: 16,16
- c. Prints: 100,100
- d. Compiler Error
- e. Run time Error
- f. None of the Above

Question 8

```
class P{  
    public static void main (String[] args) {  
        StringBuffer s1 = new StringBuffer(null);  
        System.out.print(s1);  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints nothing
- b. Prints: null
- c. Compiler Error
- d. Run time Error
- e. None of the Above

Question 9

```
class J {  
    public static void main (String[] args) {  
        StringBuffer s1 = new StringBuffer(32);  
        System.out.print(s1.length() + "," + s1.capacity());  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0,0
- b. Prints: 0,32
- c. Prints: 32,0
- d. Prints: 32,32
- e. Compiler Error
- f. Run time Error
- g. None of the Above

Question 10

```
class N{  
    public static void main (String[] args) {  
        StringBuffer s1 = new StringBuffer();  
        System.out.print(s1.append(null));  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints nothing
- b. Prints: null
- c. Compiler Error
- d. Run time Error
- e. None of the Above

Question 11

```
class N{  
    public static void main (String[] args) {  
        String s1 = null;  
        StringBuffer s2 = new StringBuffer();  
        System.out.print(s2.append(s1));  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints nothing
- b. Prints: null
- c. Compiler Error
- d. Run time Error
- e. None of the Above

No.
정답 - 해설
Remark

1 d
Run time Error
A `StringIndexOutOfBoundsException` exception is thrown at run time in response to the attempt to set the value of a character beyond the current length of the `StringBuffer`.

2 a
Prints: false,false
`StringBuffer` does not override the `equals` method of the `Object` class, so `StringBuffer.equals` just compares reference values. The `equals` method of `s1` returns false in response to the attempt to compare the equality of a `String` and a `StringBuffer`. Since the two objects have different types, the result is false.

3 e
Prints: 0,16
The length is zero because the `StringBuffer` contains no characters. The default capacity of a `StringBuffer` is 16.

4 g
None of the Above
The capacity is the default capacity of a `StringBuffer` plus the length of the `String` used to initialize the `StringBuffer`. The default size is 16 and the length of the `String` is seven so the capacity is 23. This form of the `StringBuffer` constructor performs two steps. First, it instantiates the internal char array with the capacity described above. Second, it calls the `StringBuffer.append` method to place the `String` into the char array.

5 g
None of the Above
The `Boolean` objects are converted to `Strings` before being placed into the internal character array of the `StringBuffer`. After the first `Boolean` is appended to the `StringBuffer`, the `StringBuffer` has a length of four characters. The insert operation inserts the word false into the `StringBuffer` starting a character position one. The result is the character sequence `tfalserue`.

6 c
Prints: ABCDE,16
The `setLength` method has been used to reduce the length of the `StringBuffer` from seven down to five. As a result the last two characters are truncated. The capacity remains 16.

7 c
Prints: 100,100
The `setLength` method has been used to increase the length of the `StringBuffer` from seven up to 100. As a result the capacity is also increased to 100. If the

length had been set to a value less than 34, then the capacity would have been set to 34. The smallest capacity increase is defined by the following: $\text{newCapacity} = (\text{oldCapacity} + 1) * 2$. In this case, the old capacity was 16 so the smallest new capacity would be 34.

8 d
Run time Error
The constructor for the `StringBuffer` class does not verify that the input reference parameter is not null before attempting to invoke methods on the `String` object. As a result, passing a null reference to the constructor of the `StringBuffer` class results in an error at run time. Please note that a null `String` reference can safely be passed to the `StringBuffer.append` method, but a null literal may not be passed to `StringBuffer.append`.

9 b
Prints: 0,32
The capacity has been set to 32. The `StringBuffer` contains no characters so the length is zero.

10 c
Compiler Error
An attempt to pass a null literal to the `append` method will result in a compile time error due to an ambiguous reference type. However, if a null string reference is passed to the `append` method as the input parameter, then the null reference will be passed to the `String.valueOf` method and the result will be appended to the `StringBuffer`.

11 b
Prints: null
If a null `String` is passed to the `append` method as the input parameter, then the null reference will be passed to the `String.valueOf` method and the result will be appended to the `StringBuffer`. Please note that an attempt to pass a null literal to the `append` method will result in a compile time error due to an ambiguous reference type. Also, an attempt to pass a null `String` reference to the constructor of the `StringBuffer` class results in a run time error.

[Wrapper Classes]

Question 1
class C {
 public static void main (String args[]) {
 Long a = new Long(1);
 byte b = a.byteValue();
 short c = a.shortValue();
 int d = a.intValue();
 long e = a.longValue();
 float f = a.floatValue();
 double g = a.doubleValue();
 System.out.print(b+c+d+e+f+g);
 }
}

What is the result of attempting to compile and run the program ?

- a. Prints: 6
- b. Prints: 6L
- c. Prints: 6.0
- d. Compiler Error
- e. Runtime Error
- f. None of the Above

Question 2

```
class M {  
    public static void main(String s[]){  
        System.out.print((Long.MIN_VALUE < 0) + ",");  
        System.out.print((Double.MIN_VALUE < 0) + ",");  
        System.out.print(Double.MIN_VALUE < Long.MIN_VALUE);  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Compiler Error
- j. Runtime Error
- k. None of the Above

Question 3

```
class C {  
    public static void main (String args[]) {  
        String s1 = "#f";  
        String s2 = "0xf";  
        byte b1 = Byte.parseByte(s1);  
        byte b2 = Byte.parseByte(s2);  
        System.out.print(b1 + b2);  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0x1d
- b. Prints: 0x1f
- c. Prints: 0xff
- d. Prints: 30
- e. Prints: 32
- f. Compiler Error
- g. Run time Error
- h. None of the Above

Question 4

```
class D {  
    public static void main (String args[]) {  
        Byte b1 = new Byte("10");  
        Byte b2 = Byte.decode("#10");  
        byte b3 = Byte.parseByte("10",8);  
        Byte b4 = Byte.valueOf("10",8);  
        System.out.print(b1+"",b2+"",b3+"",b4);  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 10,10,8,8
- b. Prints: 10,10,10,10

c. Prints: 10,16,8,8

d. Compiler Error

e. Run time Error

f. None of the Above

Question 5

```
class A {  
    public static void main(String s[]){  
        System.out.print((Integer.MIN_VALUE < 0) + ",");  
        System.out.print((Double.MIN_VALUE < 0) + ",");  
        System.out.print(Double.MIN_VALUE < Integer.MIN_VALUE);  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Compiler Error
- j. Runtime Error
- k. None of the Above

Question 6

```
class A {  
    public static void main (String args[]) {  
        int primitiveInt = 1;  
        long primitiveLong = 1L;  
        float primitiveFloat = 1.0f;  
        String s = "1";  
        Integer i1 = new Integer(primitiveInt);  
        Integer i2 = new Integer(primitiveLong);  
        Integer i3 = new Integer(primitiveFloat);  
        Integer i4 = new Integer(s);  
        int i5 = i1.intValue() + i2.intValue() + i3.intValue() + i4.intValue();  
        System.out.print(i5);  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 4
- b. Prints: 4.0
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 7

```
class A {  
    public static void main (String args[]) {  
        byte primitiveByte = 1;  
        char primitiveChar = 1;  
        double primitiveDouble = 1;  
        String s = "1";  
        Double d1 = new Double(primitiveByte);  
        Double d2 = new Double(primitiveChar);  
        Double d3 = new Double(primitiveDouble);  
    }  
}
```

```

Double d4 = new Double(s);
double d5 = d1.doubleValue() + d2.doubleValue()
+
    d3.doubleValue() + d4.doubleValue();

    System.out.print(d5);
}
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 4
- b. Prints: 4.0
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 8

```

class D {
    static boolean m(double v) {
        return(v != v == Double.isNaN(v));
    }
    public static void main (String args[]) {
        double d1 = Double.NaN;
        double d2 = Double.POSITIVE_INFINITY;
        double d3 = Double.MAX_VALUE;
        System.out.print(m(d1) + "," + m(d2) + "," +
m(d3));
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Compiler Error
- j. Runtime Error
- k. None of the Above

Question 9

```

class A {
    public static void main (String args[]) {
        Double d1 = new Double(1.0);
        Double d2 = new Double(d1);
        System.out.print(d1.equals(d2));
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: false
- b. Prints: true
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 10

```

class C {
    public static void main (String args[]) {
        byte b1 = Byte.decode("1");

```

```

        byte b2 = Byte.decode("2");
        System.out.print(b1 + b2);
    }
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: 3
- b. Prints: 12
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

No.

정답 - 해설

Remark

1 c

Prints: 6.0

Long is a subclass of the abstract class Number and implements all of the methods of Number such as byteValue, shortValue, floatValue, etc. In this case, the Long object contains the value 1L. Please note that there is no charValue method.

2 e

Prints: true,false,false

Long.MIN_VALUE represents the most negative number that can be represented by an long value. Double.MIN_VALUE represents the smallest positive value that is greater than zero.

3 g

Run time Error

Byte.parseByte is overloaded. The first version accepts a String representation of an integer as the input parameter. The second version accepts both a String representation of an integer and also a radix. The code presented for this question generates a run time error as a result of the attempt to pass in a String that contains a prefix used to specify a hexadecimal integer. Byte.parseByte does not examine the format of the String to determine the radix. Instead, parseByte uses the radix argument to determine the radix.

4 c

Prints: 10,16,8,8

The Byte.decode method is the only method of the Byte class that determines the radix of the String by examining the format. An Octal or Hexadecimal radix is specified using a prefix. An octal radix is specified using a leading zero. A Hexadecimal radix is specified using either of two prefixes # or 0x. The parseByte and valueOf methods use a second parameter, radix, to specify the radix. The parseByte method is the only one that returns a primitive byte value.

5 e

Prints: true,false,false

Integer.MIN_VALUE represents the most negative number that can be represented by an int value. Double.MIN_VALUE represents the smallest positive value that is greater than zero.

6 c

Compiler Error

Integer has two constructors. The first accepts a parameter of type primitive int. The second accepts a parameter of type String. The attempt to pass primitiveLong and primitiveFloat to a constructor of the Integer class generates a compiler error because method invocation type conversions do not include implicit narrowing conversions such as the narrowing of a long or float to an int.

7 b

Prints: 4.0

Double has two constructors. The first accepts a parameter of type primitive double. The second accepts a parameter of type String.

8 h

Prints: true,true,true

NaN is the only value that is not equal to itself. The Double.isNaN method returns the result of the expression (v != v).

9 c

Compiler Error

Double has two constructors. The first accepts a parameter of type primitive double. The second accepts a parameter of type String. There is no constructor that accepts a reference to an object of type Double.

10 c

Compiler Error

The decode method returns a Byte Object. A compiler error is generated as a result of the attempt to assign the return value to a byte primitive.

Question 1

```
class C {
    public static void main (String args[]) {
        String s1 = "15";
        String s2 = "0xf";
        long i1 = Long.parseLong(s1);
        long i2 = Long.parseLong(s2);
        System.out.print(i1==i2);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: true
- b. Prints: false
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 2

```
class B {
    public static void main (String args[]) {
        Double a = new Double(0xFFFFF);
        byte b = a.byteValue();
        short c = a.shortValue();
        int e = a.intValue();
    }
}
```

```
long f = a.longValue();
float g = a.floatValue();
double h = a.doubleValue();
System.out.print(b+", "+c+", "+ (e+f+g+h == 4 *
0xFFFFF));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0xFFFF,0xFFFF,false
- b. Prints: 0xFFFF,0xFFFF,true
- c. Prints: -1,-1,false
- d. Prints: -1,-1,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 3

```
class D {
    public static void main (String args[]) {
        Byte a = new Byte("1");
        byte b = a.byteValue();
        short c = a.shortValue();
        char d = a.charValue();
        int e = a.intValue();
        long f = a.longValue();
        float g = a.floatValue();
        double h = a.doubleValue();
        System.out.print(b+c+d+e+f+g+h);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 7
- b. Prints: 7.0
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 4

```
class A {
    public static void main(String s[]){
        System.out.print((Integer.MIN_VALUE < 0) + ",");
        System.out.print((Float.MIN_VALUE < 0) + ",");
        System.out.print(Float.MIN_VALUE <
Integer.MIN_VALUE);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Compiler Error
- j. Runtime Error
- k. None of the Above

Question 5

```
class C {
    public static void main (String args[]) {
        String s1 = "0xf";
    }
}
```

```

String s2 = "010";
Byte b1 = Byte.decode(s1);
Byte b2 = Byte.decode(s2);
System.out.print(b1.byteValue() + b2.byteValue());
}
}

```

What is the result of attempting to compile and run the program ?

- | | |
|------------------|----------------------|
| a. Prints: 0x01f | b. Prints: 0x1f |
| c. Prints: 1f | d. Prints: 23 |
| e. Prints: 25 | f. Compiler Error |
| g. Runtime Error | h. None of the Above |

Question 6

```

class E {
    static String m(int i) {return "int primitive";}
    static String m(Integer i) {return "Integer Object";}
    static String m(double i) {return "double primitive";}
    static String m(Double i) {return "Double object";}
    static String m(String i) {return "String object";}
    public static void main (String[] args) {
        System.out.print(m(Integer.parseInt("1")));
    }
}

```

What is the result of attempting to compile and run the program ?

- | | |
|--------------------------|-----------------------------|
| a. Prints: int primitive | b. Prints: double primitive |
| c. Prints: Double object | d. Prints: String object |
| e. Compiler Error | f. Runtime Error |
| g. None of the Above | |

Question 7

```

class D {
    static String m(int i) {return "int primitive";}
    static String m(double i) {return "double primitive";}
    static String m(Double i) {return "Double object";}
    static String m(String i) {return "String object";}
    public static void main (String[] args) {
        System.out.print(m(Double.parseDouble("1")));
    }
}

```

What is the result of attempting to compile and run the program ?

- | | |
|--------------------------|-----------------------------|
| a. Prints: int primitive | b. Prints: double primitive |
| c. Prints: Double object | d. Prints: String object |
| e. Compiler Error | f. Runtime Error |
| g. None of the Above | |

Question 8

```

class C {
    public static void main (String args[]) {
        String s1 = "011";
        String s2 = "0xf";
        int i1 = Integer.parseInt(s1);
        int i2 = Integer.parseInt(s2);
        System.out.print(i1 + i2);
    }
}

```

What is the result of attempting to compile and run the program ?

- | | |
|------------------|----------------------|
| a. Prints: 24 | b. Prints: 25 |
| c. Prints: 26 | d. Compiler Error |
| e. Runtime Error | f. None of the Above |

Question 9

```

class F {
    static String m(long i) {return "long primitive";}
    static String m(Long i) {return "Long Object";}
    static String m(double i) {return "double primitive";}
    static String m(Double i) {return "Double object";}
    static String m(String i) {return "String object";}
    public static void main (String[] args) {
        System.out.print(m(Long.parseLong("1")));
    }
}

```

What is the result of attempting to compile and run the program ?

- | | |
|-----------------------------|--------------------------|
| a. Prints: long primitive | b. Prints: Long Object |
| c. Prints: double primitive | d. Prints: Double Object |
| e. Prints: String object | f. Compiler Error |
| g. Runtime Error | h. None of the Above |

Question 10

```

class H {
    public static void main (String args[]) {
        System.out.print(Long.toHexString(Long.MAX_VALUE));
    }
}

```

What is the result of attempting to compile and run the program ?

- | |
|-----------------------------|
| a. Prints: 8000000000000000 |
| b. Prints: 7FFFFFFFFFFFFFFF |
| c. Prints: FFFFFFFFFFFFFFFF |
| d. Prints: 80000000 |
| e. Prints: 7FFFFFFF |
| f. Prints: FFFFFFFF |
| g. Compiler Error |
| h. Runtime Error |
| i. None of the Above |

No.

정답 - 해설
Remark

1 d

Runtime Error

The Long.parseLong method is not able to determine the radix of the input String based on the format of the number. Long.parseLong assumes that the input String represents a decimal value unless a second parameter is provided to specify the radix. The code presented for this question produces a NumberFormatException at run time as a result of the attempt to pass in 0xf as an input parameter .

2 d

Prints: -1,-1,true

Double is a subclass of the abstract class Number and implements all of the methods of Number such as byteValue, shortValue, floatValue, etc. In this case, the Double object contains the value 0xFFFF. When that value is cast to a byte the result is 0xFF which is also the two's complement representation of a byte value that is equal to negative one. Similarly, 0xFFFF is the two's complement representation of a short value that is equal to negative one. Please note there is no charValue method.

3 c

Compiler Error

A compiler error is generated because the Byte class does not have a charValue method. The Byte class extends the Number class and implements all six of the Number methods. It is interesting to note that four of the Number methods are abstract while two, byteValue and shortValue, are not abstract. That's because there is a concrete implementation of byteValue and shortValue that just call the intValue method and then cast the result to the appropriate type.

4 e

Prints: true,false,false

Integer.MIN_VALUE represents the most negative number that can be represented by an int value. Float.MIN_VALUE represents the smallest positive value that is greater than zero.

5 d

Prints: 23

The decode method accepts a String representation of an 8-bit signed integer and parses it into a Byte object. The String may be in hexadecimal format, octal format, or in decimal format.

6 a

Prints: int primitive

The Integer.parseInt method returns an int primitive.

7 b

Prints: double primitive

The Double.parseDouble method returns a double primitive.

8 e

Runtime Error

The Integer.parseInt method is not able to determine the radix of the input String based on the format of the number. Integer.parseInt assumes that the input String represents a decimal value unless a second parameter is provided to specify the radix. The code presented for this question produces a NumberFormatException at run time.

9 a

Prints: long primitive

The Long.parseLong method returns a long primitive.

10 b

Prints: 7FFFFFFFFFFFFFFF

A primitive long is a 64 bit value. The maximum value is represented by a 64 bit value with the sign bit set to zero and all the other bits set to one.

Question 1

```
class A {
    public static void main (String args[]) {
        byte primitiveByte = 1; // 1
        Byte b1 = new Byte(primitiveByte); // 2
        Byte b2 = new Byte(1); // 3
        System.out.print(b1.byteValue() + b2.byteValue());
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 2
- b. Prints: 11
- c. Compiler Error at 1
- d. Compiler Error at 2
- e. Compiler Error at 3
- f. Runtime Error
- g. None of the Above

Question 2

```
class B {
    public static void main (String args[]) {
        Integer a = new Integer(0xFFFF);
        byte b = a.byteValue();
        short c = a.shortValue();
        int e = a.intValue();
        long f = a.longValue();
        float g = a.floatValue();
        double h = a.doubleValue();
        System.out.print(b+","+c+","+ (e+f+g+h == 4 *
0xFFFF));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 0xFF,0xFFFF,false
- b. Prints: 0xFF,0xFFFF,true
- c. Prints: 0xFFFF,0xFFFF,false
- d. Prints: 0xFFFF,0xFFFF,true
- e. Prints: -1,-1,false
- f. Prints: -1,-1,true
- g. Compiler Error
- h. Runtime Error
- i. None of the Above

Question 3

```
class C {
    public static void main (String args[]) {
        String s1 = "#f";
        String s2 = "0xf";
        Byte b1 = Byte.decode(s1);
        Byte b2 = Byte.decode(s2);
        System.out.print(b1.byteValue() + b2.byteValue());
    }
}
```

What is the result of attempting to compile and run

the program ?

- a. Prints: 0x1d
- b. Prints: 0x1f
- c. Prints: 0xff
- d. Prints: 30
- e. Prints: 32
- f. Compiler Error
- g. Runtime Error
- h. None of the Above

Question 4

```
class G {
    public static void main (String[] args) {
        System.out.print(Long.parseLong("11", 2)+" ", );
        System.out.print(Long.parseLong("11", 8)+" ", );
        System.out.print(Long.parseLong("11", 10)+" ", );
        System.out.print(Long.parseLong("11", 16));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 2,8,10,16
- b. Prints: 3,9,11,17
- c. Prints: 11,11,11,11
- d. Prints: 13,19,21,27
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 5

```
class A {
    public static void main (String args[]) {
        byte primitiveByte = 1;
        char primitiveChar = 'b'-'a';
        int primitiveInt = 1;
        short primitiveShort = 1;
        String s = "1";
        Integer i1 = new Integer(primitiveByte);
        Integer i2 = new Integer(primitiveChar);
        Integer i3 = new Integer(primitiveShort);
        Integer i4 = new Integer(primitiveInt);
        Integer i5 = new Integer(s);
        int i6 = i1.intValue() + i2.intValue() +
                i3.intValue() + i4.intValue() +
                i5.intValue();

        System.out.print(i6);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 5
- b. Prints: 5.0
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 6

```
class D {
    public static void main (String args[]) {
        byte b1 = Byte.valueOf("10",Character.MIN_RADIX);
        byte b2 = Byte.valueOf("10",8);
        byte b3 = Byte.valueOf("10",10);
        byte b4 = Byte.valueOf("10",16);
        byte b5 = Byte.valueOf("10",Character.MAX_RADIX);
        System.out.print(b1+","+b2+","+b3+","+b4+","+b5);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 2,8,10,16,36
- b. Compiler Error
- c. Run time Error
- d. None of the Above

Question 7

```
class D {
    public static void main (String args[]) {
        boolean b1 = Integer.MIN_VALUE == 0x80000000;
        boolean b2 = Integer.MAX_VALUE == 0x7FFFFFFF;
        System.out.print(b1 + "," + b2);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 8

```
class F {
    static String m(long i) {return "long primitive";}
    static String m(Long i) {return "Long Object";}
    static String m(double i) {return "double primitive";}
    static String m(Double i) {return "Double object";}
    static String m(String i) {return "String object";}
    public static void main (String[] args) {
        System.out.print(m(Long.parseLong("1")));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: long primitive
- b. Prints: Long Object
- c. Prints: double primitive
- d. Prints: Double Object
- e. Prints: String object
- f. Compiler Error
- g. Runtime Error
- h. None of the Above

Question 9

```
class A {
    public static void main (String args[]) {
        byte primitiveByte = 1;
        int primitiveInt = 1;
        long primitiveLong = 1L;
        float primitiveFloat = 1f;
        String s = "1";
        Long i1 = new Long(primitiveByte);
        Long i2 = new Long(primitiveInt);
        Long i3 = new Long(primitiveLong);
        Long i4 = new Long(primitiveFloat);
        Long i5 = new Long(s);
        int i6 = i1.intValue() + i2.intValue() +
                i3.intValue() + i4.intValue() +
                i5.intValue();

        System.out.print(i6);
    }
}
```

What is the result of attempting to compile and run

the program ?

- a. Prints: 5
- b. Prints: 5.0
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 10

```
class E {  
    public static void main (String[] args) {  
        Byte b1 = new Byte("1");  
        Byte b2 = new Byte("1");  
        System.out.print((b1==b2)+" "+(b1.equals(b2)));  
    }  
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

No.

정답 - 해설

Remark

1 e

Compiler Error at 3

The Byte class has only two constructors. One accepts a byte primitive and the other accepts a String. The literal primitive that appears in the constructor invocation statement at line 3 has type int and the compiler will not implicitly narrow the int to a byte. In line 1 a literal of type int is implicitly narrowed to a byte by the compiler. The compiler will implicitly do a narrowing conversion for an assignment statement if the right hand operand is a compile time constant of type byte, short, char, or int and the value falls within the range of the variable on the left and if the variable is of type byte, short, or char. For that reason, a constant int expression that is equal to one may be assigned to a byte without an explicit cast. The same is not true for the parameters of a method or constructor. The designers of the Java programming language felt that implicit narrowing conversions for method and constructor parameters would add unnecessary complexities to the process of resolving overloaded method calls.

2 f

Prints: -1,-1,true

Integer is a subclass of the abstract class Number and implements all of the methods of Number such as byteValue, shortValue, floatValue, etc. In this case, the Integer object contains the value 0xFFFF. When that value is cast to a byte the result is 0xFF which is also the two's complement representation of a byte value that is equal to negative one. Similarly, 0xFFFF is the two's complement representation of a short value that is equal to negative one. Please note there is no charValue method.

3 d

Prints: 30

The decode method accepts a String representation of an 8-bit signed integer and parses it into a Byte object. The String may be in hexadecimal format, octal format, or in decimal format. Hexadecimal format uses a prefix of 0x or #.

4 b

Prints: 3,9,11,17

The Long.parseLong method parses the input String parameter and returns a long primitive value that is equal to the value represented by the String. The second parameter, if present, is the radix. For example, the String value 11 with a radix of 2 is parsed as 2 plus 1 equals 3. Similarly, the String value 11 with a radix of 8 is an octal value that is parsed as 8 plus 1 equals 9. The String value 11 with a radix of 10 is a decimal value that is parsed as 10 plus 1 equals 11. The String value 11 with a radix of 16 is a hexadecimal value that is parsed as 16 plus 1 equals 17.

5 a

Prints: 5

Integer has two constructors. The first accepts a parameter of type primitive int. The second accepts a parameter of type String.

6 b

Compiler Error

A compiler error is generated as a result of the attempt to assign the return value of valueOf to a primitive byte variable. The Byte.valueOf method is overloaded. The return value of both versions is a Byte object.

7 d

Prints: true,true

An int is a 32 bit value. The left most bit is the sign bit. The sign bit is set to one for negative numbers and is set to zero for positive numbers.

8 a

Prints: long primitive

The Long.parseLong method accepts a String parameter that represents a numeric value. The parseLong method does not determine the type of the numeric value based on a suffix such as l, L, f, F, d, or D. Use of any such suffix generates a NumberFormatException at run time.

9 c

Compiler Error

Long has two constructors. The first accepts a parameter of type primitive long. The second accepts a parameter of type String. The attempt to pass primitiveFloat to a constructor generates a compiler error because method invocation type conversions do not include implicit narrowing conversions such as the narrowing of a float to a long.

10 b

Prints: false,true

The expression (b1==b2) compares the references of two instances of Byte objects. The result is false. The expression (b1.equals(b2)) compares the values of two instances of byte objects. Since both instances contain the value one the return value of the method call is true.

Question 1

```
class B {
    public static void main (String args[]) {
        Long i1 = new Long(1);
        Long i2 = new Long(i1);
        System.out.print(i1.longValue() + i2.longValue());
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false
- b. Prints: true
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 2

```
class D {
    public static void main (String args[]) {
        Byte b1 = Byte.valueOf("10",Character.MIN_RADIX);
        Byte b2 = Byte.valueOf("10",8);
        Byte b3 = Byte.valueOf("10",10);
        Byte b4 = Byte.valueOf("10",16);
        Byte b5 = Byte.valueOf("10",Character.MAX_RADIX);
        System.out.print(b1+","+b2+","+b3+","+b4+","+b5);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 2,8,10,16,36
- b. Compiler Error
- c. Run time Error
- d. None of the Above

Question 3

```
class H {
    public static void main (String args[]) {

System.out.print(Long.toHexString(Long.MIN_VALUE));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 8000000000000000
- b. Prints: 7FFFFFFFFFFFFFFF
- c. Prints: FFFFFFFFFFFFFFFF
- d. Prints: 80000000
- e. Prints: 7FFFFFFF
- f. Prints: FFFFFFFF
- g. Compiler Error
- h. Runtime Error
- i. None of the Above

Question 4

```
class C {
    public static void main (String args[]) {
        String s1 = "1";
        String s2 = "2";
        Byte b1 = Byte.parseByte(s1);
        Byte b2 = Byte.parseByte(s2);
        System.out.print(b1.byteValue() + b2.byteValue());
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 3
- b. Prints: 12
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 5

```
class A {
    public static void main (String args[]) {
        Integer i1 = new Integer(1);
        Integer i2 = new Integer(i1);
        System.out.print(i1.equals(i2));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false
- b. Prints: true
- c. Compiler Error
- d. Runtime Error
- e. None of the Above

Question 6

```
class E {
    public static void main (String[] args) {
        Byte b1 = new Byte("1");
        Byte b2 = new Byte("1");
        int a = b1.hashCode();
        int b = b2.hashCode();
        System.out.print((b1.equals(b2))+" "+(a==b));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compiler Error
- f. Runtime Error
- g. None of the Above

Question 7

```
class E {
    public static void main (String[] args) {
        System.out.print(Integer.parseInt("11", 2)+" ");
        System.out.print(Integer.parseInt("11", 8)+" ");
        System.out.print(Integer.parseInt("11", 10)+" ");
        System.out.print(Integer.parseInt("11", 16));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 2,8,10,16
- b. Prints: 3,9,11,17

- c. Prints: 11,11,11,11 d. Prints: 13,19,21,27
- e. Compiler Error f. Runtime Error
- g. None of the Above

Question 8

```
class F {
    public static void main (String[] args) {

System.out.print(Byte.MIN_VALUE+", "+Byte.MAX_VALUE);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: -128,127 b. Prints: -127,128
- c. Prints: 0,255 d. Compiler Error
- e. Runtime Error f. None of the Above

Question 9

```
1. class B {
2.     public static void main (String args[]) {
3.         Byte b1 = new Byte(1);
4.         Byte b2 = new Byte('2');
5.         Byte b3 = new Byte("3");
6.         byte i1 = b1.byteValue() + b2.byteValue() +
b3.byteValue();
7.         System.out.print(i1);
8.     }
9. }
```

What is the result of attempting to compile and run the program ?

- a. Prints: 6 b. Compiler Error at 3
- c. Compiler Error at 4 d. Compiler Error at 5
- e. Compiler Error at 6 f. Runtime Error
- g. None of the Above

No.

정답 - 해설

Remark

1 c

Compiler Error

Long has two constructors. The first accepts a parameter of type primitive long. The second accepts a parameter of type String. There is no constructor that accepts a reference to an object of type Long.

2 a

Prints: 2,8,10,16,36

The Byte.valueOf method is overloaded. The return value of both versions is a Byte object. The first version of valueOf accepts a single parameter that is a string representation of a byte. The valueOf method does not examine the format of the string parameter to determine the radix, so valueOf does not accept Strings that have a prefix such as the octal leading zero or the hexadecimal prefix 0x. The second version of the valueOf method accepts a second parameter which is a string that represents the radix used to parse the first parameter.

3 a

Prints: 8000000000000000

A primitive long is a 64 bit value. The minimum value is represented by a 64 bit value with the sign bit set to one and all the other bits set to zero.

4 c

Compiler Error

The Byte.parseByte method returns a primitive byte. A compiler error is generated as a result of the attempt to assign the return value to a Byte reference.

5 c

Compiler Error

Integer has two constructors. The first accepts a parameter of type primitive int. The second accepts a parameter of type String. There is no constructor that accepts a reference to an object of type Integer.

6 d

Prints: true,true

The expression (b1.equals(b2)) compares the values of two instances of byte objects. Since both instances contain the value one the return value of the method call is true. The expression (a==b) compares the hash codes of two instances of Byte objects. Since the two Byte objects contain the same value the result is true.

7 b

Prints: 3,9,11,17

The Integer.parseInt method parses the input String parameter and returns an int primitive value that is equal to the value represented by the String. The second parameter, if present, is the radix. For example, the String value 11 with a radix of 2 is parsed as 2 plus 1 equals 3. Similarly, the String value 11 with a radix of 8 is an octal value that is parsed as 8 plus 1 equals 9. The String value 11 with a radix of 10 is a decimal value that is parsed as 10 plus 1 equals 11. The String value 11 with a radix of 16 is a hexadecimal value that is parsed as 16 plus 1 equals 17.

8 a

Prints: -128,127

9 b c e

Compiler Error at 3 Compiler Error at 4 Compiler Error at 6

The Byte class has only two constructors. One accepts a byte primitive and the other accepts a String. The literal primitive that appears in the constructor invocation statement at line 3 has type int and the compiler will not implicitly narrow the int to a byte. Line 4 generates a compiler error due to the attempt to use an argument of type char. Line 6 generates a compiler error because the result of the expression is of type int but the variable is of type byte.

[Collections/Hashcodes]

Question 1

```
import java.util.*;
class H {
    public static void main (String args[]) {
        Object a = new HashSet();
        Object b = new HashMap();
        Object c = new Hashtable();
        System.out.print((a instanceof Map)+"");
        System.out.print((b instanceof Map)+"");
        System.out.print(c instanceof Map);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 2

- a. Stores key/value pairs.
- b. Allows null elements, keys, and values.
- c. Duplicate entries replace old entries.
- d. Entries are not sorted.

Which of these classes provides the specified features ?

- | | |
|---------------------|----------------------|
| a. LinkedList | b. TreeMap |
| c. TreeSet | d. HashMap |
| e. HashSet | f. Hashtable |
| g. All of the above | h. None of the above |

Question 3

```
import java.util.*;
class C {
    public static void main (String args[]) {
        AbstractList a = new Vector();
        AbstractMap b = new WeakHashMap();
        AbstractSet c = new HashSet();
        System.out.print((a instanceof Collection)+"");
        System.out.print((b instanceof Collection)+"");
        System.out.print(c instanceof Collection);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true

- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 4

```
import java.util.*;
class G {
    public static void main (String args[]) {
        Object a = new HashSet();
        Object b = new HashMap();
        Object c = new Hashtable();
        System.out.print((a instanceof
AbstractCollection)+"");
        System.out.print((b instanceof
AbstractCollection)+"");
        System.out.print(c instanceof AbstractCollection);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 5

Which implementation of the List interface provides for the slowest access to an element in the middle of the list by means of an index ?

- | | |
|----------------------|---------------------|
| a. Vector | b. ArrayList |
| c. LinkedList | d. All of the above |
| e. None of the above | |

Question 6

A class C contains a field of type int and a large array of primitives of type double. Programmer A suggests writing a hashCode method that saves time by calculating only the hashCode of the int field. Programmer B suggests writing a hashCode method that includes every element of the array in the calculation. As the manager of the software engineering department you must now decide which programmer receives the higher pay raise at the end of the year Which is it going to be ?

- a. Programmer A

b. Programmer B

Question 7

Which implementation of the List interface provides for the fastest insertion of a new element into the middle of the list ?

- a. Vector
- b. ArrayList
- c. LinkedList
- d. All of the above
- e. None of the above

Question 8

- a. Entries are not organized as key/value pairs.
- b. Duplicate entries are rejected.

Which interface of the java.util package offers the specified behavior ?

- a. List
- b. Map
- c. Set
- d. All of the above
- e. None of the above

Question 9

As long as the fields used for the equals comparison remain unchanged the hashCode method must consistently return the same integer value.

- a. false
- b. true

Question 10

An immutable class C contains a field of type int and a large array of primitives of type double. You must consider developing a hashCode method based one of these three options. Which of the three is most likely to optimize the performance of a Hashtable without violating any of the rules for coding a hashCode method ?

- a. Calculate the hashCode using only the int field.
- b. Calculate the hashCode using both the int field and the array.
- c. Calculate the hashCode using both the int field and the array, but only calculate the hashCode once and store the value for future use in an instance variable.

No.
정답 - 해설
Remark

1 d

Prints: false,true,true HashSet implements the Set interface but not the Map interface. HashMap extends AbstractMap and implements the Map interface. Hashtable extends Dictionary and implements the Map interface.

2 d

HashMap The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. The requirement to allow null elements is not satisfied by a Hashtable. TreeMap and TreeSet store elements in a sorted order based on the key.

3 f

Prints: true,false,true The List and Set interfaces extend the Collection interface but Map does not. WeakHashMap implements Map and therefore is not an instance of Collection.

4 e

Prints: true,false,false HashSet is a subclass of AbstractSet and AbstractCollection and therefore implements the Collection interface. HashMap extends AbstractMap and implements the Map interface. HashMap does not implement the Collection interface. Hashtable extends Dictionary and implements the Map interface. Hashtable does not implement Collection.

5 c

LinkedList The ArrayList and Vector both use an array to store the elements of the list so access to any element using an index is very fast. The LinkedList is implemented using a doubly linked list so access an element requires the list to be traversed using the links.

6 b

Programmer B Both implementations are technically legal. If two instances of class C are equal, then the field of type int in both classes will contain the same value and produce a hashCode that is equal. However, by ignoring the contents of the array it is possible that a very large number of unequal objects will produce the same hashCode causing the distribution of the objects across a hashtable to be less than optimal. For that reason, the algorithm suggested by programmer A is inferior to the algorithm suggested by programmer B. Therefore, programmer B gets the higher pay raise.

7 c

LinkedList The ArrayList and Vector both use an array to store the elements of the list. When an element is inserted into the middle of the list the elements that follow the insertion point must be shifted to make room for the new element. The LinkedList is implemented using a doubly linked list. An insertion at any point requires only the updating of the links. Therefore, the LinkedList allows for fast insertions and deletions.

8 c

Set The Map interface organizes entries as key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries.

9 b
true

10 c

Calculate the hashCode using both the int field and the array, but only calculate the hashCode once and store the value for future use in an instance variable. The value of an immutable object does not change so the value of the hashCode will not change either. Therefore, calculating the hashCode only once is acceptable.

Question 1

```
import java.util.*;
class B {
    public static void main (String args[]) {
        AbstractMap a = new HashMap();
        AbstractList b = new ArrayList();
        AbstractSet c = new HashSet();
        System.out.print((a instanceof Collection)+",");
        System.out.print((b instanceof Collection)+",");
        System.out.print(c instanceof Collection);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 2

```
import java.util.*;
class E {
    public static void main (String args[]) {
        AbstractSet a = new TreeSet();
        System.out.print((a instanceof Set)+",");
        System.out.print(a instanceof SortedSet);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. None of the above

Question 3

```
import java.util.*;
class F {
    public static void main (String args[]) {
        LinkedList a = new LinkedList();
        ArrayList b = new ArrayList();
        Vector c = new Vector();
        System.out.print((a instanceof List)+",");
        System.out.print((b instanceof List)+",");
        System.out.print(c instanceof List);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 4

- a. Duplicate elements are not accepted.
- b. Entries are sorted.

Which of these classes provides the specified features

- | | |
|---------------------|----------------------|
| a. LinkedList | b. TreeMap |
| c. TreeSet | d. HashMap |
| e. HashSet | f. Hashtable |
| g. All of the above | h. None of the above |

Question 5

- a. Entries are organized as key/value pairs.
- b. Duplicate entries replace old entries.

Which interface of the java.util package offers the specified behavior

- | | |
|----------------------|---------------------|
| a. List | b. Map |
| c. Set | d. All of the above |
| e. None of the above | |

Question 6

A class C has two fields of type String and both are used for the equals comparison. On Monday an instance of the class is created and the hashCode method is invoked. On Tuesday the program is loaded again and an instance of class C is created containing the same data that was loaded on Monday. If the hashCode method is invoked after restarting the program on Tuesday then the hashCode method must return the same integer value that was returned on Monday.

- a. false b. true

Question 7

If two instances of a class type are not equal according to the equals method, then the same integer value must not be returned by the hashCode method of the two objects.

- a. false b. true

Question 8

```
class A {
    private int[] val;
    private int hash;
    public int hashCode() {
        int h = hash;
        if (h == 0) {
            int off = 0;
            int len = val.length;
            for (int i = 0; i < len; i++) {
                h = 31*h + val[off++];
            }
            hash = h;
        }
        return h;
    }
    // The equals method has been omitted for clarity.
    A (int[] val) {this.val = val;}
    public static void main (String[] args) {
        A a = new A(new int[]{1,2,3});
        System.out.print(a.hashCode());
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 1026 b. Prints: 1091
c. Prints: 31806 d. Compiler error
e. Run time error f. None of the above

Question 9

```
import java.util.*;
class I {
    public static void main (String args[]) {
        Object a = new HashSet();
        Object b = new HashMap();
        Object c = new Hashtable();
        System.out.print((a instanceof Cloneable)+",");
        System.out.print((b instanceof Cloneable)+",");
        System.out.print((c instanceof Cloneable));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
b. Prints: false,false,true

- c. Prints: false,true,false
d. Prints: false,true,true
e. Prints: true,false,false
f. Prints: true,false,true
g. Prints: true,true,false
h. Prints: true,true,true
i. None of the above

Question 10

- a. Entries are not organized as key/value pairs.
b. Generally accepts duplicate elements.
c. Entries may be accessed by means of an index.

Which interface of the java.util package offers the specified behavior

- a. List b. Map
c. Set d. All of the above
e. None of the above

No.
정답 - 해설
Remark

1 d
Prints: false,true,true The List and Set interfaces extend the Collection interface but Map does not. HashMap implements Map and therefore is not an instance of Collection.

2 d
Prints: true,true TreeSet implements the Set interface and the SortedSet interface.

3 h
Prints: true,true,true Vector has been upgraded to implement the List interface.

4 c
TreeSet There is no requirement to store key/value pairs so a Map is not a good choice. A List generally accepts duplicate elements. A Set stores a collection of unique objects. Any attempt to store a duplicate object is rejected. TreeSet stores elements in an order that is determined either by a comparator or by the comparable interface.

5 b
Map The List and Set interfaces do not support key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries.

6 a
false

7 a
false If two objects are equal according to the equals method, then the hashcodes must also be equal. If two objects are not equal according to the equals method, then the hashcodes may or may not

be equal. It is preferable that unequal objects have different hashcodes, but that is not always possible. Since the hashCode value is a 32 bit primitive int it is not possible to produce a unique hashCode for each value of a primitive long.

8 a

Prints: 1026 The hashCode is calculated using all three array elements. Since the object is immutable, the hashCode is only calculated the first time hashCode is invoked.

9 h

Prints: true,true,true All three implement Cloneable.

10 a

List The Map interface organizes entries as key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries. A List allows entries to be accessed using an index.

Question 1

Which of the following classes allow unsynchronized read operations by multiple threads

- a. Vector
- b. Hashtable
- c. TreeMap
- d. TreeSet
- e. HashMap
- f. HashSet
- g. WeakHashMap
- h. All of the above
- i. None of the above

Question 2

- a. Stores key/value pairs.
- b. Allows null elements, keys, and values.
- c. Duplicate entries replace old entries.
- d. Entries are sorted using a comparator or the Comparable interface.

Which of these classes provides the specified features

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. Hashtable
- g. All of the above
- h. None of the above

Question 3

```
import java.util.*;
class A {
    public static void main (String args[]) {
        ArrayList a = new ArrayList();
        AbstractSet b = new TreeSet();
        AbstractMap c = new TreeMap();
        System.out.print((a instanceof Collection)+",");
        System.out.print((b instanceof Collection)+",");
        System.out.print((c instanceof Collection));
    }
}
```

What is the result of attempting to compile and run

the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 4

- a. Entries are organized as key/value pairs.
- b. Duplicate entries replace old entries.
- c. Entries are sorted using a comparator or the Comparable interface.

Which interface of the java.util package offers the specified behavior

- a. List
- b. Map
- c. Set
- d. SortedSet
- e. SortedMap
- f. All of the above
- g. None of the above

Question 5

```
import java.util.*;
import java.io.Serializable;
class J {
    public static void main (String args[]) {
        HashMap a = new HashMap();
        boolean b1, b2, b3;
        b1 = (a instanceof Cloneable) & (a instanceof
        Serializable);
        b2 = (a instanceof Map) & (a instanceof
        AbstractMap);
        b3 = a instanceof Collection;
        System.out.print(b1 + "," + b2 + "," + b3);
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 6

```
import java.util.*;
class D {
    public static void main (String args[]) {
```

```

AbstractSet a = new HashSet();
System.out.print((a instanceof Set)+"");
System.out.print(a instanceof SortedSet);
}
}

```

What is the result of attempting to compile and run the program ?

- a. Prints: false,false b. Prints: false,true
- c. Prints: true,false d. Prints: true,true
- e. None of the above

Question 7

- a. Entries are not organized as key/value pairs.
- b. Duplicate entries are rejected.
- c. Entries are sorted using a comparator or the Comparable interface.

Which interface of the java.util package offers the specified behavior

- a. List b. Map
- c. Set d. SortedSet
- e. SortedMap f. All of the above
- g. None of the above

Question 8

If two instances of a class type are equal according to the equals method, then the same integer value must be returned by the hashCode method of the two objects.

- a. false b. true

Question 9

- a. Each element must be unique.
- b. Duplicate elements must not replace old elements.
- c. Elements are not key/value pairs.

Which of these classes provides the specified features ?

- a. LinkedList b. TreeMap
- c. TreeSet d. HashMap
- e. HashSet f. Hashtable
- g. All of the above h. None of the above

Question 10

The presence of a mapping for a given key within this collection instance will not prevent the key from being recycled by the garbage collector.

Which concrete class provides the specified features

- a. Vector b. Hashtable
- c. TreeMap d. TreeSet
- e. HashMap f. HashSet

- g. WeakHashMap h. All of the above
- i. None of the above

No.

정답 - 해설

Remark

1 The Vector and Hashtable methods are synchronized and do not allow for simultaneous access by multiple threads. The concrete subclasses of the AbstractList, AbstractMap, and AbstractSet classes allow for unsynchronized read operations by multiple threads. Additionally, the synchronized wrapper methods of the Collections class allow for the instantiation of a Collection, List, Map, Set, SortedMap, or SortedSet with synchronized methods. If simultaneous read and write operations are necessary then a synchronized instance should be used.

2 b

TreeMap The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. The requirement to allow null elements is not satisfied by a Hashtable. TreeMap and TreeSet store elements in a sorted order based on the key, but the TreeSet does not support key/value pairs.

3 g

Prints: true,true,false The List and Set interfaces extend the Collection interface but Map does not. TreeMap implements Map and therefore is not an instance of Collection.

4 e

SortedMap The List and Set interfaces do not support key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries. A Map organizes the entries as key/value pairs. The SortedMap is similar to a Map except that the ordering of the elements is determined by a comparator or the Comparable interface.

5 g

Prints: true,true,false HashMap does not implement the Collection interface.

6 c

Prints: true,false HashSet implements the Set interface but not the SortedSet interface.

7 d

SortedSet The Map interface organizes entries as key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries. The SortedSet is similar to a Set except that the ordering of the elements is determined by a comparator or the Comparable interface.

8 b

true If two objects are equal according to the equals method, then the hashcodes must also be equal. If two objects are not equal according to the equals method, then the hashcodes may or may not be equal. It is preferable that unequal objects have different hashcodes, but that is not always possible. Since the hashCode value is a 32 bit primitive int it is not possible to produce a unique hashCode for each value of a primitive long.

9 e

HashSet The elements of a Map are key/value pairs so a Map is not a good choice. A List generally accepts duplicate elements. A Set stores a collection of unique elements. Any attempt to store a duplicate element in a Set is rejected. TreeSet stores elements in a sorted order based on the key. HashSet does not sort the elements based on the key.

10 g

WeakHashMap Objects become eligible for garbage collection when all references to the object are removed from active threads. With the exception of a WeakHashMap an entry stored in a Map is not garbage collected when the references to the key are removed. A WeakHashMap can be used to avoid the memory leaks generally associate with the use of other implementations of the Map interface.

[Assertions]

Question 1

```
class D {
    static {
        boolean assertsEnabled = false;
        assert assertsEnabled = true;
        if (!assertsEnabled) {throw new
RuntimeException();}
    }
    public static void main (String[] args) {}
}
```

Which statements are true ?

- a. If assertions are not enabled at run time it prints an error message.
- b. If assertions are not enabled at run time it prints nothing.
- c. With assertions enabled it prints an error message.
- d. With assertions enabled it prints nothing.
- e. The assert statement is being used to check a class invariant--something that must be true about each instance of the class.
- f. The assert statement is being used to prevent the class from being initialized if assertions are disabled.
- g. The assert statement is being used to check a postcondition--something that must be true when the method completes successfully.
- h. A compiler error is generated.
- i. None of the above.

Question 2

```
class D {
    private boolean b1;
    private boolean b2;
    public void setB1(boolean b) {b1 = b;}
    public void setB2(boolean b) {b2 = b;}
    public void m1 () {
        if (!b2 & !b1) {
            System.out.print("A");
        } else if (!b2 & b1) {
            System.out.print("B");
        } else if (b2 & !b1) {
            System.out.print("C");
        }
        assert false;
    }
    public static void main (String[] args) {
        D d = new D();
        d.setB1(true);
        d.setB2(true);
        d.m1();
    }
}
```

Which statements are true ?

- a. With assertions enabled it prints an AssertionError message.
- b. With assertions enabled it prints nothing.
- c. With assertions disabled it prints an AssertionError message.
- d. With assertions disabled it prints nothing.
- e. An assertion should not be placed at any location that the programmer believes will never be reached under normal operating conditions.
- f. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program.
- g. A compiler error is generated.
- h. None of the above.

Question 3

```
class B {
    public static void main(String[] args) {
        int i = 0;
        do {
            assert i++ < 5;
            System.out.print(i);
        } while (true);
    }
}
```

Which statements are true ?

- a. With assertions enabled it prints "01234" followed by an AssertionError message.
- b. With assertions enabled it prints "12345" followed by an AssertionError message.
- c. With assertions disabled it prints: 12345

- d. With assertions disabled it attempts to print an infinite sequence of zeros.
- e. With assertions disabled the variable `i` is incremented with each pass through the loop.
- f. As a rule, the boolean expression of an `assert` statement should not be used to perform actions necessary for normal operation of the program.
- g. A compiler error is generated.
- h. None of the above.

Question 4

Which statements are true ?

- a. Assertions should not be used to validate arguments passed to public methods.
- b. Assertions should not be used to validate arguments passed to nonpublic methods.
- c. Assertions should not be used within the default case of a switch statement.
- d. Assertions should not be used to perform processing that is required for the normal operation of the application.
- e. Assertions should never be placed at any point in the code that should not be reached under normal circumstances.
- f. The compiler will generate an error if an `assert` statement is "unreachable" as defined by the Java Language Specification.
- g. A try/catch block should not be used to catch an `AssertionError`.
- h. None of the above.

Question 5

```
class C {
    int a, b, c;
    public void setA(int i) {
        a = i;
        assert validateC() : c;
    }
    public void setB(int i) {
        b = i;
        assert validateC() : c;
    }
    private boolean validateC() {
        return c > a + 2 * b;
    }
    public int m1(int i) {
        c = a + b + i;
        assert validateC() : c;
        return c;
    }
    public C(int i) {
        c = i;
        assert validateC() : c;
    }
    public static void main(String[] args) {
        C c = new C(251);
        c.setA(50);
        c.setB(100);
    }
}
```

```
}
}
```

Which statements are true

- a. If assertions are not enabled at run time it prints an error message.
- b. If assertions are not enabled at run time it prints nothing.
- c. With assertions enabled it prints an error message.
- d. With assertions enabled it prints nothing.
- e. The `assert` statement is being used to check a class invariant--something that must be true about each instance of the class.
- f. The `assert` statements are being used to check a precondition--something that must be true when the method is invoked.
- g. A compiler error is generated.
- h. None of the above.

Question 6

```
class C {
    String m1(int i) {
        switch (i) {
            case 0: return "A";
            case 1: return "B";
            case 2: return "C";
            default:
                assert false;
        }
        return "E";
    }
    public static void main(String[] args) {
        C c = new C();
        for (int i = 0; i < 4; i++) {
            System.out.print(c.m1(i));
        }
    }
}
```

Which statements are true ?

- a. With assertions enabled it prints "ABC" followed by an `AssertionError` message.
- b. With assertions enabled it prints "ABCE" followed by an `AssertionError` message.
- c. With assertions disabled it prints "ABC"
- d. With assertions disabled it prints "ABCE"
- e. Assertions should not be used within the default case of a switch statement.
- f. A compiler error is generated.
- g. None of the above.

Question 7

```
class C {
    String m1(int i) {
        switch (i) {
            case 0: return "A";
            case 1: return "B";
            case 2: return "C";
        }
    }
}
```

```

        default:
            throw new AssertionError();
    }
}
public static void main(String[] args) {
    C c = new C();
    for (int i = 0; i < 4; i++) {
        System.out.print(c.m1(i));
    }
}
}

```

Which statements are true ?

- With assertions enabled it prints "ABC" followed by an AssertionError message.
- With assertions disabled it prints "ABC" followed by an AssertionError message.
- Assertions should not be used within the default case of a switch statement.
- In this code example an "assert" statement could not be used in place of the "throw" statement.
- A compiler error is generated.
- None of the above.

Question 8

```

class B {
    int a, b, c;
    private void setA(int i) {a = i;}
    private void setB(int i) {b = i;}
    private int m1 (int i) {
        c = a + b + i;
        assert c < 200 : c;
        return c;
    }
    public static void main (String[] args) {
        B b = new B();
        b.setA(50);
        b.setB(100);
        b.m1(50);
    }
}

```

Which statements are true ?

- If assertions are not enabled at run time it prints an error message.
- If assertions are not enabled at run time it prints nothing.
- With assertions enabled it prints an error message.
- With assertions enabled it prints nothing.
- The assert statement is being used to check a postcondition--something that must be true when the method completes successfully.
- A compiler error is generated.
- None of the above.

Question 9

```

class C {

```

```

String m1(int i) {
    switch (i) {
        case 0: return "A";
        case 1: return "B";
        case 2: return "C";
        default:
            assert false;
    }
}
}
public static void main(String[] args) {
    C c = new C();
    for (int i = 0; i < 4; i++) {
        System.out.print(c.m1(i));
    }
}
}

```

Which statements are true ?

- With assertions enabled it prints "ABC" followed by an AssertionError message.
- With assertions disabled it prints "ABC" followed by an AssertionError message.
- Assertions should not be used within the default case of a switch statement.
- In this code example a "throw" statement must be used in place of the "assert" statement.
- A compiler error is generated.
- None of the above.

Question 10

```

class E {
    private static final boolean assertsEnabled = false;
    private void m1(int i) {
        if (assertsEnabled) {
            assert i < 10;
        }
        // Do some interesting processing here.
    }
    public static void main (String[] args) {
        E e = new E();
        e.m1(11);
    }
}

```

Which statements are true ?

- The assert statement is being used to check an control flow invariant.
- The assert statement is being used to check a precondition.
- Method m1 is an example of an improper use of an assert statement.
- The compiler may choose to remove all traces of the assertion code from method m1.
- A compiler error is generated.
- None of the above.

Question 11

```

class A {

```

```

private void m1 (int i) {
    assert i < 10 : i;
    System.out.print(i);
}
public void m2 (int i) {
    assert i < 10 : i;
    System.out.print(i);
}
public static void main (String[] args) {
    A a = new A();
    a.m1(11);
    a.m2(12);
}
}

```

Which statements are true ?

- If assertions are enabled at run time it prints an error message.
- With assertions enabled it prints nothing.
- With assertions disabled it prints an error message.
- With assertions disabled it prints "1112".
- With assertions disabled it prints nothing.
- The assert statements are being used to check a precondition--something that must be true when the method is invoked.
- Method m1 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a non-public method.
- Method m2 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a public method.
- A compiler error is generated.
- None of the above.

Question 12

```

class A {
    void m1(int i) {
        int j = i % 3;
        switch (j) {
            case 0:
                System.out.print("0");
                break;
            case 1:
                System.out.print("1");
                break;
            default:
                assert j == 2;
                System.out.print(j);
        }
    }
}
public static void main (String[] args) {
    A a = new A();
    for (int i=5; i >= -1; i--) {
        a.m1(i);
    }
}
}

```

Which statements are true ?

- With assertions enabled it prints "210210-1" followed by an AssertionError message.
- With assertions enabled it prints "210210" followed by an AssertionError message.
- With assertions enabled it prints only "210210".
- With assertions enabled it prints nothing.
- With assertions disabled it prints "210210-1".
- With assertions disabled it prints only "210210".
- Assertions should not be used within the default case of a switch statement.
- A compiler error is generated.
- None of the above.

Question 13

An assert statement can be used to check a control-flow invariant to verify which of the following

- A particular assumption is true when the flow of control enters a method.
- The flow of control continues to flow.
- The flow of control does not reach a particular point in the program.
- The normal flow of control has reached a particular point in the program.
- The normal flow of control has reached the end of a method.
- The default case of a switch statement is not reached.
- The "else" block of an if/else statement is not reached.
- An assumption stated in a comment is correct.
- A compiler error is generated.
- None of the above.

Question 14

```

class F {
    private boolean b1;
    private boolean b2;
    private boolean b3;
    public void setB1(boolean b) {b1 = b;}
    public void setB2(boolean b) {b2 = b;}
    public void setB3(boolean b) {b3 = b;}
    public String m1 (int i) {
        b2 = i % 2 == 0;
        if (!b3 & !b2 & !b1) {
            return "None are true.";
        } else if (!b3 & !b2 & b1) {
            return "Only b1 is true.";
        } else if (!b3 & b2 & !b1) {
            return "Only b2 is true.";
        } else if (b3 & !b2 & !b1) {
            return "Only b3 is true.";
        } else {
            throw new AssertionError();
        }
    }
}
public static void main (String[] args) {
    F f = new F();
}
}

```

```

        f.setB1(true);
        System.out.println(f.m1(5));
        System.out.println(f.m1(6));
    }
}

```

Which statements are true ?

- a. Prints "Only b1 is true" followed by an error message.
- b. An assertion should not be placed at any location that the programmer believes will never be reached under normal operating conditions.
- c. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2, or b3, to be true.
- d. The assert statement is being used to check a precondition--something that must be true when the method begins.
- e. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program.
- f. The throw statement could be replaced by an assert statement.
- g. A compiler error is generated.
- h. None of the above.

Question 15

```

class E {
    private boolean b1;
    private boolean b2;
    private boolean b3;
    public void setB1(boolean b) {b1 = b;}
    public void setB2(boolean b) {b2 = b;}
    public void setB3(boolean b) {b3 = b;}
    public void m1 (int i) {
        b2 = i % 2 == 0;
        if (!b3 & !b2 & !b1) {
            System.out.print("A");
        } else if (!b3 & !b2 & b1) {
            System.out.print("B");
        } else if (!b3 & b2 & !b1) {
            System.out.print("C");
        } else { // Only b3 is true.
            assert b3 & !b2 & !b1;
        }
        System.out.print(b1 + "," + b2 + "," + b3);
        b1 = b2 = b3 = false;
    }
    public static void main (String[] args) {
        E e = new E();
        e.setB1(true);
        e.m1(2);
    }
}

```

Which statements are true ?

- a. With assertions enabled it prints an error message.
- b. With assertions enabled it prints: true,true,false
- c. With assertions enabled it prints nothing.

- d. With assertions disabled it prints an error message.
- e. With assertions disabled it prints: true,true,false
- f. With assertions disabled it prints nothing.
- g. An assertion should not be placed at any location that the programmer believes will never be reached under normal operating conditions.
- h. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2, or b3, to be true.
- i. The assert statement is being used to check a precondition--something that must be true when the method begins.
- j. The assert statement is being used to check an internal invariant--something that the programmer assumes to be true at a particular point in the program.
- k. The assert statement is being used to check a postcondition--something that must be true when the method is done.
- l. A compiler error is generated.
- m. None of the above.

No.

정답 - 해설

Remark

1 a d f

If assertions are not enabled at run time it prints an error message. With assertions enabled it prints nothing. The assert statement is being used to prevent the class from being initialized if assertions are disabled. This question is an example of the recommended implementation of a requirement to make use of a class only when assertions are enabled.

2 a d f

With assertions enabled it prints an AssertionError message. With assertions disabled it prints nothing. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program. The assert statement indicates that the programmer believes that b1 and b2 will never be true simultaneously and that the assert statement should not be reached under normal operating conditions.

3 b d f

With assertions enabled it prints "12345" followed by an AssertionError message. With assertions disabled it attempts to print an infinite sequence of zeros. As a rule, the boolean expression of an assert statement should not be used to perform actions necessary for normal operation of the program. An assert statement should not be used as demonstrated in this question. When assertions are disabled the variable i is not incremented so the loop is infinite.

4 a d f g

Assertions should not be used to validate arguments

passed to public methods. Assertions should not be used to perform processing that is required for the normal operation of the application. The compiler will generate an error if an assert statement is "unreachable" as defined by the Java Language Specification. A try/catch block should not be used to catch an AssertionError. Assertions may be enabled or disabled at run time. Since assertions are not always enabled, they should not be used to validate the parameters of public methods. Parameter checking is typically published in the API specification of a method and must be enforced even when assertions are not enabled. Rather than use an assertion, an appropriate runtime exception should be thrown such as IllegalArgumentException, IndexOutOfBoundsException, or NullPointerException. However, an assertion may be used to validate the parameters of a nonpublic method. Since assertions are not always enabled, an assertion should not be used to perform operations that are required for the normal operation of the program. For example, the boolean expression of an assertion should not be used to produce the side effect of incrementing a variable that controls a loop statement. If assertions are disabled then the loop is unlikely to function as intended. Section 14.20 of the Java Language Specification defines "unreachable" statements. If an assert statement is "unreachable" as defined by the JLS, then a compiler error is generated. In contrast, a programmer may believe that some points in the code will not be reached as a result of design assumptions. For example, a programmer may believe that the default case of a switch statement will never be reached. An assertion can be placed in the default case to verify the behavior of the switch statement.

5 b d e

If assertions are not enabled at run time it prints nothing. With assertions enabled it prints nothing. The assert statement is being used to check a class invariant--something that must be true about each instance of the class. This question is an example of using assertions to check a class invariant--something that must be true about each instance of the class. Although a class invariant must be true before and after the execution of each public method, the invariant is typically only checked at the end of each method and constructor.

6 a d

With assertions enabled it prints "ABC" followed by an AssertionError message. With assertions disabled it prints "ABCE" If the default case of a switch statement should not be reached under normal operating circumstances, then the default case becomes a good candidate for the use of an assert statement.

7 a b d

With assertions enabled it prints "ABC" followed by an AssertionError message. With assertions disabled it prints "ABC" followed by an AssertionError message.

In this code example an "assert" statement could not be used in place of the "throw" statement. If the default case of a switch statement should not be reached under normal operating circumstances, then the default case becomes a good candidate for the use of an assert statement. If a method is declared with a non void return type and if no return statement appears after the switch statement, then each case of the switch must have a return statement or a throw statement. The throw statement is used rather than an assert because the compiler knows that the assert statement is not functional when assertions are disabled.

8 b c e

If assertions are not enabled at run time it prints nothing. With assertions enabled it prints an error message. The assert statement is being used to check a postcondition--something that must be true when the method completes successfully. Variable c equals 200 when the assertion is checked.

9 d e

In this code example a "throw" statement must be used in place of the "assert" statement. A compiler error is generated. If the default case of a switch statement should not be reached under normal operating circumstances, then the default case becomes a good candidate for the use of an assert statement. If a method is declared with a non void return type and if no return statement appears after the switch statement, then each case of the switch must have a return statement or a throw statement. The throw statement is used rather than an assert because the compiler knows that the assert statement is not functional when assertions are disabled.

10 b d

The assert statement is being used to check a precondition. The compiler may choose to remove all traces of the assertion code from method m1. The "if" statement prevents the assert statement from ever being executed at run time. Section 14.20 of the Java Language Specification states that "an optimizing compiler may realize that the statement [in this case an assert statement] will never be executed and may choose to omit the code for that statement from the generated class file".

11 a d f h

If assertions are enabled at run time it prints an error message. With assertions disabled it prints "1112". The assert statements are being used to check a precondition--something that must be true when the method is invoked. Method m2 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a public method. Assertions may be enabled or disabled at run time. Since assertions are not always enabled, they should not be used to validate the parameters of public methods. Parameter checking is typically published in the API specification of a

method and must be enforced even when assertions are not enabled. Rather than use an assertion, an appropriate runtime exception should be thrown such as `IllegalArgumentException`, `IndexOutOfBoundsException`, or `NullPointerException`. However, an assertion may be used to validate the parameters of a nonpublic method.

12 b e

With assertions enabled it prints "210210" followed by an `AssertionError` message. With assertions disabled it prints "210210-1". If the default case of a switch statement should not be reached under normal operating circumstances, then an assert statement can be placed in the default case to verify the expected behavior of the program.

13 c f g h The flow of control does not reach a particular point in the program. The default case of a switch statement is not reached. The "else" block of an if/else statement is not reached. An assumption stated in a comment is correct. A control-flow invariant is placed at a point in the program that the programmer assumes will never be reached. Two examples are the default case of a switch statement or the "else" block of an if/else statement. It makes no sense to use an assert statement to verify that the flow of control does reach a particular point in the program because it is unlikely generating an assertion error is helpful when the program is found to be functioning correctly. An assert statement placed at the beginning of a method is generally used to check a precondition. An assert statement that is placed at the end of a method to check the state of some variables is generally said to be checking a post condition. However, it is also possible that an assert statement placed at the end of a method might also be checking a control-flow invariant. The correct term depends on the usage.

14 a c e

Prints "Only b1 is true" followed by an error message. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2, or b3, to be true. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program. Method m1 has a series of if/else statements. The first "if" statement is processed if none of the booleans are true. The second is processed if only b1 is true. The third is processed if only b2 is true. The fourth is processed if only b3 is true. A set of three booleans can exist in one of eight states. The first four "if" statements account for four of those states so four more states remain. The combination of the three "if" statements and the fact that an `AssertionError` is thrown from the final else block indicates that the programmer believes that no more than one of the booleans will be true when method m1 is being processed. That assumption is called an internal invariant. The fact that the invariant is tested by verifying that control never reaches a particular point

in the program means that it is a control-flow invariant. A throw statement is used in place of an assert statement because the throw statement can not be disabled and therefore the method is certain to generate an error once control passes beyond all of the return statements. The compilation would fail if an assert statement were used in place of the throw statement.

15 a e h j

With assertions enabled it prints an error message. With assertions disabled it prints: true,true,false. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2, or b3, to be true. The assert statement is being used to check an internal invariant--something that the programmer assumes to be true at a particular point in the program. Method m1 has a series of if/else statements. The first "if" statement is processed if none of the booleans are true. The second is processed if only b1 is true. The third is processed if only b2 is true. A set of three booleans can exist in eight states. The three "if" statements account for three of those states so five more states remain. The assert statement indicates that the programmer assumes that only one of those five remaining states is valid--that is the state where only b3 is true. The combination of the three "if" statements and the assert statement indicate that the programmer believes that no more than one of the booleans will be true at that point in the program. That assumption is called an internal invariant.

Question 1

```
class A {
    static byte m1() {
        final char c = 'b'-'a';
        return c; // 1
    }
    static byte m2() {
        final short s = 2;
        return s; // 2
    }
    static byte m3(final char c) {
        return c; // 3
    }
    static byte m4(final short s) {
        return s; // 4
    }
    public static void main(String[] args) {
        char c = 'd'-'a';
        short s = 4;
        System.out.print(""+m1()+m2()+m3(c)+m4(s));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 1234 b. Prints: 10

- c. Compiler Error at 1.
- e. Compiler Error at 3.
- g. Runtime Error
- d. Compiler Error at 2.
- f. Compiler Error at 4.
- h. None of the Above

Question 2

```
class B {
    static int m1(byte b) {;
        return b; // 1
    }
    static int m2(char c) {;
        return c; // 2
    }
    static int m3(short s) {
        return s; // 3
    }
    static int m4(long l) {
        return l; // 4
    }
    static int m5(float f) {
        return f; // 5
    }
    public static void main(String[] args) {
        byte b = 1;
        char c = 'c'-'a';
        short s = 3;
        long l = 4L;
        float f = 5.0f;

        System.out.print(""+m1(b)+m2(c)+m3(s)+m4(l)+m5(f));
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 12345
- c. Prints: 15
- e. Compiler Error at 1.
- g. Compiler Error at 3.
- i. Compiler Error at 5.
- k. None of the Above
- b. Prints: 12345.0
- d. Prints: 15.0
- f. Compiler Error at 2.
- h. Compiler Error at 4.
- j. Runtime Error

Question 3

```
class C {
    static void m1(byte b) {;
        return b; // 1
    }
    static void m2(char c) {;
        return c; // 2
    }
    static void m3(short s) {
        return s; // 3
    }
    public static void main(String[] args) {
        byte b = 1;
        char c = 'c'-'a';
        short s = 3;
        System.out.print(""+m1(b)+m2(c)+m3(s)); // 4
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 123
- c. Compiler Error at 1.
- e. Compiler Error at 3.
- g. Runtime Error
- b. Prints: 6
- d. Compiler Error at 2.
- f. Compiler Error at 4.
- h. None of the Above

Question 4

```
class D {
    static int m1(int i) {;
        return i; // 1
    }
    static void m2(int i) {;
        return i; // 2
    }
    static int m3(int i) {;
        return; // 3
    }
    public static void main(String[] args) {
        System.out.print(""+m1(1)+m2(2)+m3(3)); // 4
    }
}
```

What is the result of attempting to compile and run the program ?

- a. Prints: 123
- c. Compiler Error at 1.
- e. Compiler Error at 3.
- g. Runtime Error
- b. Prints: 6
- d. Compiler Error at 2.
- f. Compiler Error at 4.
- h. None of the Above

Question 5

```
class A {
    public static void main(String[] args) {
        boolean b = true;
        if (b = false) {
            System.out.print("A");
        } else if (b) {
            System.out.print("B");
        } else {
            System.out.print("C");
        }
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: A
- c. Prints: C
- e. Compiler Error
- b. Prints: B
- d. Runtime Exception
- f. None of the Above

Question 6

```
class A {
    static boolean b;
    public static void main(String[] args) {
        if (b) {
            System.out.print("A");
        }
    }
}
```

```

    } else if (b = false) {
        System.out.print("B");
    } else if (b) {
        System.out.print("C");
    } else if (!b) {
        System.out.print("D");
    } else {
        System.out.print("E");
    }
}
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|-------------------|----------------------|
| a. Prints: A | b. Prints: B |
| c. Prints: C | d. Prints: D |
| e. Prints: E | f. Runtime Exception |
| g. Compiler Error | h. None of the Above |

Question 7

```

class A {
    public static void main(String[] args) {
        boolean b;
        if (b = false) {
            System.out.print("A");
        } else if (b) {
            System.out.print("B");
        } else if (!b) {
            System.out.print("C");
        } else {
            System.out.print("D");
        }
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|-------------------|----------------------|
| a. Prints: A | b. Prints: B |
| c. Prints: C | d. Prints: D |
| e. Prints: E | f. Runtime Exception |
| g. Compiler Error | h. None of the Above |

Question 8

```

class A {
    public static void main (String[] args) {
        int a = 1 | 2 ^ 3 & 5;
        int b = ((1 | 2) ^ 3) & 5;
        int c = 1 | (2 ^ (3 & 5));
        System.out.print(a + "," + b + "," + c);
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|------------------|------------------|
| a. Prints: 0,0,0 | b. Prints: 0,0,3 |
| c. Prints: 0,3,0 | d. Prints: 0,3,3 |
| e. Prints: 3,0,0 | f. Prints: 3,0,3 |

- | | |
|----------------------|-------------------|
| g. Prints: 3,3,0 | h. Prints: 3,3,3 |
| i. Runtime error | j. Compiler error |
| k. None of the above | |

Question 9

```

class A {
    public static void main (String[] args) {
        int a = 1 || 2 ^ 3 && 5;
        int b = ((1 || 2) ^ 3) && 5;
        int c = 1 || (2 ^ (3 && 5));
        System.out.print(a + "," + b + "," + c);
    }
}

```

What is the result of attempting to compile and run the above program ?

- | | |
|----------------------|-------------------|
| a. Prints: 0,0,0 | b. Prints: 0,0,3 |
| c. Prints: 0,3,0 | d. Prints: 0,3,3 |
| e. Prints: 3,0,0 | f. Prints: 3,0,3 |
| g. Prints: 3,3,0 | h. Prints: 3,3,3 |
| i. Runtime error | j. Compiler error |
| k. None of the above | |

Question 10

```

class A {
    static boolean a;
    static boolean b;
    static boolean c;
    public static void main (String[] args) {
        boolean x = (a = true) || (b = true) && (c = true);
        System.out.print(a + "," + b + "," + c);
    }
}

```

What is the result of attempting to compile and run the above program ?

- | |
|------------------------------|
| a. Prints: false,false,false |
| b. Prints: false,false,true |
| c. Prints: false,true,false |
| d. Prints: false,true,true |
| e. Prints: true,false,false |
| f. Prints: true,false,true |
| g. Prints: true,true,false |
| h. Prints: true,true,true |
| i. Runtime error |
| j. Compiler error |
| k. None of the above |

Question 11

```

class A {
    static boolean a;
    static boolean b;
    static boolean c;
    public static void main (String[] args) {
        boolean x = a || (b = true) && (c = true);
        System.out.print(a + "," + b + "," + c);
    }
}

```



```

    }
}

```

What is the result of attempting to compile and run the above program ?

- Prints: false,false,false
- Prints: false,false,true
- Prints: false,true,false
- Prints: false,true,true
- Prints: true,false,false
- Prints: true,false,true
- Prints: true,true,false
- Prints: true,true,true
- Runtime error
- Compiler error
- None of the above

Question 12

```

class A {
    void m1(int i) {
        int j = i % 3;
        switch (j) {
            case 0:
                System.out.print("0");
                break;
            case 1:
                System.out.print("1");
                break;
            default:
                assert j == 2;
                System.out.print(j);
        }
    }
}

public static void main (String[] args) {
    A a = new A();
    for (int i=5; i >= -1; i--) {
        a.m1(i);
    }
}

```

Which statements are true ?

- With assertions enabled it prints "210210-1" followed by an AssertionError message.
- With assertions enabled it prints "210210" followed by an AssertionError message.
- With assertions enabled it prints only "210210".
- With assertions enabled it prints nothing.
- With assertions disabled it prints "210210-1"
- With assertions disabled it prints only "210210"
- Assertions should not be used within the default case of a switch statement.
- A compiler error is generated
- None of the above

Question 13

```

class B {
    public static void main(String[] args) {

```

```

        int i = 0;
        do {
            assert i++ < 5;
            System.out.print(i);
        } while (true);
    }
}

```

Which statements are true ?

- With assertions enabled it prints "01234" followed by an AssertionError message.
- With assertions enabled it prints "12345" followed by an AssertionError message.
- With assertions disabled it prints: 12345
- With assertions disabled it attempts to print an infinite sequence of zeros.
- With assertions disabled the variable i is incremented with each pass through the loop.
- As a rule, the boolean expression of an assert statement should not be used to perform actions necessary for normal operation of the program.
- A compiler error is generated.
- None of the above.

Question 14

Which statements are true ?

- Assertions should not be used to validate arguments passed to public methods.
- Assertions should not be used to validate arguments passed to nonpublic methods.
- Assertions should not be used within the default case of a switch statement.
- Assertions should not be used to perform processing that is required for the normal operation of the application.
- Assertions should never be placed at any point in the code that should not be reached under normal circumstances.
- The compiler will generate an error if an assert statement is "unreachable" as defined by the Java Language Specification.
- A try/catch block should not be used to catch an AssertionError.
- None of the above.

Question 15

```

class C {
    String m1(int i) {
        switch (i) {
            case 0: return "A";
            case 1: return "B";
            case 2: return "C";
            default:
                throw new AssertionError();
        }
    }
}

public static void main(String[] args) {

```

```

    C c = new C();
    for (int i = 0; i < 4; i++) {
        System.out.print(c.m1(i));
    }
}
}

```

Which statements are true ?

- With assertions enabled it prints "ABC" followed by an AssertionError message.
- With assertions disabled it prints "ABC" followed by an AssertionError message.
- Assertions should not be used within the default case of a switch statement.
- In this code example an "assert" statement could not be used in place of the "throw" statement.
- A compiler error is generated.
- None of the above.

Question 16

```

class C {
    String m1(int i) {
        switch (i) {
            case 0: return "A";
            case 1: return "B";
            case 2: return "C";
            default:
                assert false;
        }
        return "E";
    }
    public static void main(String[] args) {
        C c = new C();
        for (int i = 0; i < 4; i++) {
            System.out.print(c.m1(i));
        }
    }
}

```

Which statements are true ?

- With assertions enabled it prints "ABC" followed by an AssertionError message.
- With assertions enabled it prints "ABCE" followed by an AssertionError message.
- With assertions disabled it prints "ABC"
- With assertions disabled it prints "ABCE"
- Assertions should not be used within the default case of a switch statement.
- A compiler error is generated.
- None of the above.

Question 17

```

class C {
    String m1(int i) {
        switch (i) {
            case 0: return "A";
            case 1: return "B";
            case 2: return "C";

```

```

        default:
            assert false;
        }
    }
    public static void main(String[] args) {
        C c = new C();
        for (int i = 0; i < 4; i++) {
            System.out.print(c.m1(i));
        }
    }
}

```

Which statements are true ?

- With assertions enabled it prints "ABC" followed by an AssertionError message.
- With assertions disabled it prints "ABC" followed by an AssertionError message.
- Assertions should not be used within the default case of a switch statement.
- In this code example a "throw" statement must be used in place of the "assert" statement.
- A compiler error is generated.
- None of the above.

Question 18

```

class D {
    private boolean b1;
    private boolean b2;
    public void setB1(boolean b) {b1 = b;}
    public void setB2(boolean b) {b2 = b;}
    public void m1 () {
        if (!b2 & !b1) {
            System.out.print("A");
        } else if (!b2 & b1) {
            System.out.print("B");
        } else if (b2 & !b1) {
            System.out.print("C");
        }
        assert false;
    }
    public static void main (String[] args) {
        D d = new D();
        d.setB1(true);
        d.setB2(true);
        d.m1();
    }
}

```

Which statements are true ?

- With assertions enabled it prints an AssertionError message.
- With assertions enabled it prints nothing.
- With assertions disabled it prints an AssertionError message.
- With assertions disabled it prints nothing.
- An assertion should not be placed at any location that the programmer believes will never be reached under normal operating conditions.
- The assert statement is being used to check a

control-flow invariant to verify that the control flow never reaches a point in the program.

- g. A compiler error is generated.
- h. None of the above.

Question 19

```
class A {
    private void m1 (int i) {
        assert i < 10 : i;
        System.out.print(i);
    }
    public void m2 (int i) {
        assert i < 10 : i;
        System.out.print(i);
    }
    public static void main (String[] args) {
        A a = new A();
        a.m1(11);
        a.m2(12);
    }
}
```

Which statements are true ?

- a. If assertions are enabled at run time it prints an error message.
- b. With assertions enabled it prints nothing.
- c. With assertions disabled it prints an error message.
- d. With assertions disabled it prints "1112".
- e. With assertions disabled it prints nothing.
- f. The assert statements are being used to check a precondition--something that must be true when the method is invoked.
- g. Method m1 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a non-public method.
- h. Method m2 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a public method.
- i. A compiler error is generated.
- j. None of the above.

Question 20

```
class B {
    int a, b, c;
    private void setA(int i) {a = i;}
    private void setB(int i) {b = i;}
    private int m1 (int i) {
        c = a + b + i;
        assert c < 200 : c;
        return c;
    }
    public static void main (String[] args) {
        B b = new B();
        b.setA(50);
        b.setB(100);
        b.m1(50);
    }
}
```

```
}
```

Which statements are true ?

- a. If assertions are not enabled at run time it prints an error message.
- b. If assertions are not enabled at run time it prints nothing.
- c. With assertions enabled it prints an error message.
- d. With assertions enabled it prints nothing.
- e. The assert statement is being used to check a postcondition--something that must be true when the method completes successfully.
- f. A compiler error is generated.
- g. None of the above.

Question 21

```
class C {
    int a, b, c;
    public void setA(int i) {
        a = i;
        assert validateC() : c;
    }
    public void setB(int i) {
        b = i;
        assert validateC() : c;
    }
    private boolean validateC() {
        return c > a + 2 * b;
    }
    public int m1(int i) {
        c = a + b + i;
        assert validateC() : c;
        return c;
    }
    public C(int i) {
        c = i;
        assert validateC() : c;
    }
    public static void main(String[] args) {
        C c = new C(251);
        c.setA(50);
        c.setB(100);
    }
}
```

Which statements are true ?

- a. If assertions are not enabled at run time it prints an error message.
- b. If assertions are not enabled at run time it prints nothing.
- c. With assertions enabled it prints an error message.
- d. With assertions enabled it prints nothing.
- e. The assert statement is being used to check a class invariant--something that must be true about each instance of the class.
- f. The assert statements are being used to check a precondition--something that must be true when the method is invoked.
- g. A compiler error is generated.

h. None of the above.

Question 22

```
class D {
    static {
        boolean assertsEnabled = false;
        assert assertsEnabled = true;
        if (!assertsEnabled) {throw new
RuntimeException();}
    }
    public static void main (String[] args) {}
}
```

Which statements are true ?

- a. If assertions are not enabled at run time it prints an error message.
- b. If assertions are not enabled at run time it prints nothing.
- c. With assertions enabled it prints an error message.
- d. With assertions enabled it prints nothing.
- e. The assert statement is being used to check a class invariant--something that must be true about each instance of the class.
- f. The assert statement is being used to prevent the class from being initialized if assertions are disabled.
- g. The assert statement is being used to check a postcondition--something that must be true when the method completes successfully.
- h. A compiler error is generated.
- i. None of the above.

Question 23

```
class E {
    private static final boolean assertsEnabled = false;
    private void m1(int i) {
        if (assertsEnabled) {
            assert i < 10;
        }
        // Do some interesting processing here.
    }
    public static void main (String[] args) {
        E e = new E();
        e.m1(11);
    }
}
```

Which statements are true ?

- a. The assert statement is being used to check an control flow invariant.
- b. The assert statement is being used to check a precondition.
- c. Method m1 is an example of an improper use of an assert statement.
- d. The compiler may choose to remove all traces of the assertion code from method m1.
- e. A compiler error is generated.
- f. None of the above.

Question 24

```
class A {}
class B extends A {}
class C extends B {
    static void m1(A x) {System.out.print("m1A");}
    static void m2(B x) {System.out.print("m2B"); m1(x);}
    static void m2(A x) {System.out.print("m2A"); m1(x);}
    static void m3(C x) {System.out.print("m3C"); m2(x);}
    static void m3(B x) {System.out.print("m3B"); m2(x);}
    static void m3(A x) {System.out.print("m3A"); m2(x);}
    public static void main(String[] args) {
        m3(new C());
    }
}
```

What is the result of attempting to compile the program ?

- a. Prints: m3Am2Am1A
- b. Prints: m3Bm2Bm1A
- c. Prints: m3Cm2Bm1A
- d. Prints: m3Cm2Am1A
- e. Compiler error
- f. Run time error
- g. None of the Above

Question 25

```
class E {
    private boolean b1;
    private boolean b2;
    private boolean b3;
    public void setB1(boolean b) {b1 = b;}
    public void setB2(boolean b) {b2 = b;}
    public void setB3(boolean b) {b3 = b;}
    public void m1 (int i) {
        b2 = i % 2 == 0;
        if (!b3 & !b2 & !b1) {
            System.out.print("A");
        } else if (!b3 & !b2 & b1) {
            System.out.print("B");
        } else if (!b3 & b2 & !b1) {
            System.out.print("C");
        } else { // Only b3 is true.
            assert b3 & !b2 & !b1;
        }
        System.out.print(b1 + "," + b2 + "," + b3);
        b1 = b2 = b3 = false;
    }
    public static void main (String[] args) {
        E e = new E();
        e.setB1(true);
        e.m1(2);
    }
}
```

Which statements are true ?

- a. With assertions enabled it prints an error message.
- b. With assertions enabled it prints: true,true,false
- c. With assertions enabled it prints nothing.
- d. With assertions disabled it prints an error message.
- e. With assertions disabled it prints: true,true,false

- f. With assertions disabled it prints nothing.
- g. An assertion should not be placed at any location that the programmer believes will never be reached under normal operating conditions.
- h. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2, or b3, to be true.
- i. The assert statement is being used to check a precondition--something that must be true when the method begins.
- j. The assert statement is being used to check an internal invariant--something that the programmer assumes to be true at a particular point in the program.
- k. The assert statement is being used to check a postcondition--something that must be true when the method is done.
- l. A compiler error is generated.
- m. None of the above.

Question 26

```
class F {
    private boolean b1;
    private boolean b2;
    private boolean b3;
    public void setB1(boolean b) {b1 = b;}
    public void setB2(boolean b) {b2 = b;}
    public void setB3(boolean b) {b3 = b;}
    public String m1 (int i) {
        b2 = i % 2 == 0;
        if (!b3 & !b2 & !b1) {
            return "None are true.";
        } else if (!b3 & !b2 & b1) {
            return "Only b1 is true.";
        } else if (!b3 & b2 & !b1) {
            return "Only b2 is true.";
        } else if (b3 & !b2 & !b1) {
            return "Only b3 is true.";
        } else {
            throw new AssertionError();
        }
    }
    public static void main (String[] args) {
        F f = new F();
        f.setB1(true);
        System.out.println(f.m1(5));
        System.out.println(f.m1(6));
    }
}
```

Which statements are true ?

- a. Prints "Only b1 is true" followed by an error message.
- b. An assertion should not be placed at any location that the programmer believes will never be reached under normal operating conditions.
- c. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2, or b3, to be true.
- d. The assert statement is being used to check a

precondition--something that must be true when the method begins.

- e. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program.
- f. The throw statement could be replaced by an assert statement.
- g. A compiler error is generated.
- h. None of the above.

Question 27

An assert statement can be used to check a control-flow invariant to verify which of the following

- a. A particular assumption is true when the flow of control enters a method.
- b. The flow of control continues to flow.
- c. The flow of control does not reach a particular point in the program.
- d. The normal flow of control has reached a particular point in the program.
- e. The normal flow of control has reached the end of a method.
- f. The default case of a switch statement is not reached.
- g. The "else" block of an if/else statement is not reached.
- h. An assumption stated in a comment is correct.
- i. A compiler error is generated.
- j. None of the above.

Question 28

```
class A {String s1="A";}
class B extends A {String s1="B";}
class C extends B {String s1="C";}
class D extends C {
    String s1="D";
    void m1() {
        System.out.print(s1 + ",");
        System.out.print(((C)this).s1 + ","); // 1
        System.out.print(((B)this).s1 + ","); // 2
        System.out.print(((A)this).s1); // 3
    }
    public static void main (String[] args) {
        new D().m1(); // 4
    }
}
```

What is the result of attempting to compile and run the above program ?

- a. Prints: D,D,D,D
- b. Prints: D,C,B,A
- c. Compiler Error at 1.
- d. Compiler Error at 2.
- e. Compiler Error at 3.
- f. Compiler Error at 4.
- g. Runtime Error
- h. None of the Above

Question 29

```
class SuperA {String s1="SuperA";}
```

```

class SuperB {String s1="SuperB";}
class A extends SuperA {
    String s1="A";
    class B extends SuperB { // 1
        String s1="B";
        void m1() {
            System.out.print(this.s1 + ","); // 2
            System.out.print(super.s1 + ","); // 3
            System.out.print(A.this.s1 + ","); // 4
            System.out.print(A.super.s1); // 5
        }
    }
}
public static void main (String[] args) {
    new A().new B().m1(); // 6
}
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: B,SuperB,B,SuperB
- b. Prints: B,SuperB,A,SuperA
- c. Compiler Error at 1.
- d. Compiler Error at 2.
- e. Compiler Error at 3.
- f. Compiler Error at 4.
- g. Compiler Error at 5.
- h. Compiler Error at 6.
- i. Runtime Error
- j. None of the Above

Question 30

```

class A {void m1() {System.out.print("A");}}
class B extends A {void m1(){System.out.print("B");}}
class C extends B {void m1() {System.out.print("C");}}
class D extends C {
    void m1() {System.out.print("D");}
    void m2() {
        m1();
        ((C)this).m1(); // 1
        ((B)this).m1(); // 2
        ((A)this).m1(); // 3
    }
    public static void main (String[] args) {
        new D().m2(); // 4
    }
}

```

What is the result of attempting to compile and run the above program ?

- a. Prints: DCBA
- b. Prints: DDDD
- c. Compiler Error at 1.
- d. Compiler Error at 2.
- e. Compiler Error at 3.
- f. Compiler Error at 4.
- g. Runtime Error
- h. None of the Above

No.
정답 - 해설
Remark

1 e f

Compiler Error at 3. Compiler Error at 4. There are no compiler errors at 1 and 2 because the variables are compile time constants that are assignable to type byte. There are compiler errors at 3 and 4 because the variables are not compile time constants and are not assignable to type byte without an explicit cast. For more information, please see section 5.2 of the Java Language Specification for more information on assignment conversions.

2 h i

Compiler Error at 4. Compiler Error at 5. There are no compiler errors at 1, 2 or 3 because a widening conversion from type byte, char, or short to type int is legal. There are compiler errors at 4 and 5 because a narrowing conversion is not a legal assignment conversion without an explicit cast. For more information, please see section 5.2 of the Java Language Specification for more information on assignment conversions.

3 c d e f

Compiler Error at 1. Compiler Error at 2. Compiler Error at 3. Compiler Error at 4. There are compiler errors at 1, 2, 3 and 4 because all of the methods are declared with a void return type, therefore the return statement is not permitted to return a value. Please see section 14.16 of the Java Language Specification for more information on the return statement.

4 d e f

Compiler Error at 2. Compiler Error at 3. Compiler Error at 4. There is a compiler error at 2 because the method is declared with a void return type so the return statement is not permitted to return a value. There is a compiler error at 3 because the method is declared with an int return type but the return statement does not return a value. There is a compiler error at 4 because method m2 is declared with a void return type. Please see section 14.16 of the Java Language Specification for more information on the return statement.

5 c

Prints: C

This is a trick question. The boolean b is initialized to true, but the first if statement sets b to false. Notice that the equality operator has been replaced by the simple assignment operator. Anytime you think you see the equality operator on the exam make sure that it has not been replaced by the assignment operator.

6 d

Prints: D This is a trick question. The expression (b = false) appears to be testing the value of b, but it is really setting the value of b. Notice that the equality

operator has been replaced by the simple assignment operator. Anytime you think you see the equality operator on the exam make sure that it has not been replaced by the assignment operator.

7 c

Prints: C This is a trick question. It appears that a compiler error would be generated as a result of attempting to use the value of variable b before it is initialized. In reality, the first if statement does the initialization of b. The expression (b = false) appears to be testing the value of b, but it is really setting the value of b. Notice that the equality operator has been replaced by the simple assignment operator. Anytime you think you see the equality operator on the exam make sure that it has not been replaced by the assignment operator.

8 f

Prints: 3,0,3 Java evaluates expressions from left to right while respecting operator precedence. The order of operator precedence starting with the lowest is as follows: |, ^, &.

9 j

Compiler error Both operands of the conditional "and" operator and conditional "or" operator must be of type boolean.

10 e

Prints: true,false,false The right operand of the conditional "or" operator is not evaluated if the left hand operand is true. In this case the right hand operand of the conditional "or" operator is the result of the conditional "and" expression. Therefore, the conditional "and" expression is not evaluated.

11 d

Prints: false,true,true The right operand of the conditional "or" operator is evaluated only if the left hand operand is false. In this case, the left operand of the conditional "or" operator is false so the right operand must also be evaluated. The right operand of the conditional "and" operator is only evaluated if the left hand operand is true. In this case, the left hand operand is true so the right hand expression is evaluated.

12 b e

With assertions enabled it prints "210210" followed by an AssertionError message. With assertions disabled it prints "210210-1" If the default case of a switch statement should not be reached under normal operating circumstances, then an assert statement can be placed in the default case to verify the expected behavior of the program.

13 b d f

With assertions enabled it prints "12345" followed by an AssertionError message. With assertions disabled it attempts to print an infinite sequence of zeros. As a rule, the boolean expression of an assert statement

should not be used to perform actions necessary for normal operation of the program. An assert statement should not be used as demonstrated in this question. When assertions are disabled the variable i is not incremented so the loop is infinite.

14 a d f g

Assertions should not be used to validate arguments passed to public methods. Assertions should not be used to perform processing that is required for the normal operation of the application. The compiler will generate an error if an assert statement is "unreachable" as defined by the Java Language Specification. A try/catch block should not be used to catch an AssertionError. Assertions may be enabled or disabled at run time. Since assertions are not always enabled, they should not be used to validate the parameters of public methods. Parameter checking is typically published in the API specification of a method and must be enforced even when assertions are not enabled. Rather than use an assertion, an appropriate runtime exception should be thrown such as IllegalArgumentException, IndexOutOfBoundsException, or NullPointerException. However, an assertion may be used to validate the parameters of a nonpublic method. Since assertions are not always enabled, an assertion should not be used to perform operations that are required for the normal operation of the program. For example, the boolean expression of an assertion should not be used to produce the side effect of incrementing a variable that controls a loop statement. If assertions are disabled then the loop is unlikely to function as intended. Section 14.20 of the Java Language Specification defines "unreachable" statements. If an assert statement is "unreachable" as defined by the JLS, then a compiler error is generated. In contrast, a programmer may believe that some points in the code will not be reached as a result of design assumptions. For example, a programmer may believe that the default case of a switch statement will never be reached. An assertion can be placed in the default case to verify the behavior of the switch statement.

15 a b d

With assertions enabled it prints "ABC" followed by an AssertionError message. With assertions disabled it prints "ABC" followed by an AssertionError message. In this code example an "assert" statement could not be used in place of the "throw" statement. If the default case of a switch statement should not be reached under normal operating circumstances, then the default case becomes a good candidate for the use of an assert statement. If a method is declared with a non void return type and if no return statement appears after the switch statement, then each case of the switch must have a return statement or a throw statement. The throw statement is used rather than an assert because the compiler knows that the assert statement is not functional when assertions are disabled.

16 a d

With assertions enabled it prints "ABC" followed by an `AssertionError` message. With assertions disabled it prints "ABCE". If the default case of a switch statement should not be reached under normal operating circumstances, then the default case becomes a good candidate for the use of an `assert` statement.

17 d e

In this code example a "throw" statement must be used in place of the "assert" statement. A compiler error is generated. If the default case of a switch statement should not be reached under normal operating circumstances, then the default case becomes a good candidate for the use of an `assert` statement. If a method is declared with a non void return type and if no return statement appears after the switch statement, then each case of the switch must have a return statement or a throw statement. The throw statement is used rather than an `assert` because the compiler knows that the `assert` statement is not functional when assertions are disabled.

18 a d f

With assertions enabled it prints an `AssertionError` message. With assertions disabled it prints nothing. The `assert` statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program. The `assert` statement indicates that the programmer believes that `b1` and `b2` will never be true simultaneously and that the `assert` statement should not be reached under normal operating conditions.

19 a d f h

If assertions are enabled at run time it prints an error message. With assertions disabled it prints "1112". The `assert` statements are being used to check a precondition--something that must be true when the method is invoked. Method `m2` is an example of an improper use of an `assert` statement: an `assert` statement should not be used for argument checking in a public method. Assertions may be enabled or disabled at run time. Since assertions are not always enabled, they should not be used to validate the parameters of public methods. Parameter checking is typically published in the API specification of a method and must be enforced even when assertions are not enabled. Rather than use an `assert`, an appropriate runtime exception should be thrown such as `IllegalArgumentException`, `IndexOutOfBoundsException`, or `NullPointerException`. However, an `assert` may be used to validate the parameters of a nonpublic method.

20 b c e

If assertions are not enabled at run time it prints nothing. With assertions enabled it prints an error message. The `assert` statement is being used to check a postcondition--something that must be true when the method completes successfully. Variable `c`

equals 200 when the assertion is checked.

21 b d e

If assertions are not enabled at run time it prints nothing. With assertions enabled it prints nothing. The `assert` statement is being used to check a class invariant--something that must be true about each instance of the class. This question is an example of using assertions to check a class invariant--something that must be true about each instance of the class. Although a class invariant must be true before and after the execution of each public method, the invariant is typically only checked at the end of each method and constructor.

22 a d f

If assertions are not enabled at run time it prints an error message. With assertions enabled it prints nothing. The `assert` statement is being used to prevent the class from being initialized if assertions are disabled. This question is an example of the recommended implementation of a requirement to make use of a class only when assertions are enabled.

23 b d

The `assert` statement is being used to check a precondition. The compiler may choose to remove all traces of the assertion code from method `m1`. The "if" statement prevents the `assert` statement from ever being executed at run time. Section 14.20 of the Java Language Specification states that "an optimizing compiler may realize that the statement [in this case an `assert` statement] will never be executed and may choose to omit the code for that statement from the generated class file".

24 c

Prints: `m3Cm2Bm1A` Section 15.12.2.2 of the Java Language Specification states the following. If more than one method declaration is both accessible and applicable to a method invocation, it is necessary to choose one to provide the descriptor for the run-time method dispatch. The Java programming language uses the rule that the most specific method is chosen. The informal intuition is that one method declaration is more specific than another if any invocation handled by the first method could be passed on to the other one without a compile-time type error. End of quote. In this case, the most specific version of method `m3` is the one that declares a parameter of type `C`. The most specific version of `m2` is the one that declares a parameter of type `B`.

25 a e h j

With assertions enabled it prints an error message. With assertions disabled it prints: `true,true,false` The combination of the `if/else` statements and the `assert` statement indicate that the programmer expects no more than one boolean, `b1`, `b2`, or `b3`, to be true. The `assert` statement is being used to check an internal invariant--something that the programmer

assumes to be true at a particular point in the program. Method m1 has a series of if/else statements. The first "if" statement is processed if none of the booleans are true. The second is processed if only b1 is true. The third is processed if only b2 is true. A set of three booleans can exist in eight states. The three "if" statements account for three of those states so five more states remain. The assert statement indicates that the programmer assumes that only one of those five remaining states is valid--that is the state where only b3 is true. The combination of the three "if" statements and the assert statement indicate that the programmer believes that no more than one of the booleans will be true at that point in the program. That assumption is called an internal invariant.

26 a c e

Prints "Only b1 is true" followed by an error message. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2, or b3, to be true. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program. Method m1 has a series of if/else statements. The first "if" statement is processed if none of the booleans are true. The second is processed if only b1 is true. The third is processed if only b2 is true. The fourth is processed if only b3 is true. A set of three booleans can exist in one of eight states. The first four "if" statements account for four of those states so four more states remain. The combination of the three "if" statements and the fact that an AssertionError is thrown from the final else block indicates that the programmer believes that no more than one of the booleans will be true when method m1 is being processed. That assumption is called an internal invariant. The fact that the invariant is tested by verifying that control never reaches a particular point in the program means that it is a control-flow invariant. A throw statement is used in place of an assert statement because the throw statement can not be disabled and therefore the method is certain to generate an error once control passes beyond all of the return statements. The compilation would fail if an assert statement were used in place of the throw statement.

27 c f g h

The flow of control does not reach a particular point in the program. The default case of a switch statement is not reached. The "else" block of an if/else statement is not reached. An assumption stated in a comment is correct. A control-flow invariant is placed at a point in the program that the programmer assumes will never be reached. Two examples are the default case of a switch statement or the "else" block of an if/else statement. It makes no sense to use an assert statement to verify that the flow of control does reach a particular point in the program because it is unlikely generating an assertion

error is helpful when the program is found to be functioning correctly. An assert statement placed at the beginning of a method is generally used to check a precondition. An assert statement that is placed at the end of a method to check the state of some variables is generally said to be checking a post condition. However, it is also possible that an assert statement placed at the end of a method might also be checking a control-flow invariant. The correct term depends on the usage.

28 b

Prints: D,C,B,A Inherited fields hide super class fields but do not override super class fields. The field that is accessed at run time depends on the type of the reference.

29 b

Prints: B,SuperB,A,SuperA The expression A.this.s1 is an example of a "qualified this" expression. The expression A.super.s1 is the same as ((SuperA)A.this).s1.

30 b

Prints: DDDD Instance method D.m1 overrides the superclass methods and is therefore always invoked in place of the superclass methods even if the reference is cast to the superclass type. The instance method that is invoked depends on the type of the object and not the type of the reference.

-- 편집 후기 --

와. 처음에 이것을 사이트에서 찾아서 복사하면서 무척 많은 양에 280페이지에 가까운 양을 90페이지로 줄이고 줄였다는게 무척 다행스럽게 생각한다. 참엔 280페이지를 다 뽐올려구도 생각했지만(물론 SCJP 1.4를 준비하고 있는 입장에서) SCJP 1.4를 준비하는 사람들을 위해서 조금이나마 도움이 되었으면 하는 마음으로 이것을 준비합니다.

tandoo@empal.com로부터...