```java
interface Side {
   String getSide();
}

class Head implements Side {
   public String getSide() {
       return "Head ";
   }
}

class Tail implements Side {
   public String getSide() {
       return "Tail ";
   }
}

class Coin {
   public static void overload(Head side) {
       System.out.print(side.getSide());
   }

   public static void overload(Tail side) {
       System.out.print(side.getSide());
   }

   public static void overload(Side side) {
       System.out.print("Side ");
   }

   public static void overload(Object side) {
       System.out.print("Object ");
   }

   public static void main(String[] args) {
       Side firstAttempt = new Head();
       Tail secondAttempt = new Tail();
       overload(firstAttempt);
       overload((Object) firstAttempt);
       overload(secondAttempt);
       overload((Side) secondAttempt);
   }
}
```

Side Object Tail Side

---

abstract class Vehicle {}
interface Drivable {}
class Car extends Vehicle implements Drivable {}
class SUV extends Car {}
Which of the following options will compile?

a)ArrayList<Vehicle> all = new ArrayList<>();
SUV suv = all.get(0);

b)ArrayList<Drivable> al2 = new ArrayList<>();
Car car = al2.get(0);

c)ArrayList<SUV> al3 = new ArrayList<>();
Drivable drivable = al3.get(0);

d)ArrayList<SUV> al4 = new ArrayList<>();
Car car2 = al4.get(0);

e)ArrayList<Vehicle> al5 = new ArrayList<>();
Drivable drivable2 = al5.get(0);

What will be the result of attempting to compile and run the following program?

```
public class TestClass {
    public static void main(String args[]) {
        int x = 0;
        labelA:
        for (int i = 10; i > 0; i--) {
            int j = 0;
            labelB:
            while (j < 10) {
                if (j > i) break labelB;
                if (i == j) {
                    x++;
                    continue labelA;
                }
                j++;
            }
            x--;
        }
        System.out.println(x);
}
}
```

8

```
public class PromotionTest {
    public static void main(String args[]) {
        int i = 5;
        float f = 5.5f;
        double d = 3.8;
        char c = 'a';

        if (i == f) c++;
        if (((int) (f + d)) ((int) f + (int) d)) c += 2;
        System.out.println(c);
    }
}
```

Compilation error (Second if syntax is wrong)

```
public class ForSwitch {
    public static void main(String args[]) {
        char i;
        LOOP: for (i = 0; i < 5; i++) {
            switch (i++) {
                case '0': System.out.println("A");
                case 1: System.out.println("B"); break LOOP;
                case 2: System.out.println("C"); break;
                case 3: System.out.println("D"); break;
                case 4: System.out.println("E");
                case 'E': System.out.println("F");
            }
        }
    }
}
```

C, E, F

```
public class Test {
    public static void main(String[] args) {
        if (args[0].equals("open")) {
            if (args[1].equals("someone")) {
                System.out.println("Hello!");
            } else {
                System.out.println("Go away " + args[1]);
            }
        }
    }
}
```

Which of the following statements are true if the above program is run with the command line: java Test closed

a. It will throw ArrayIndexOutOfBoundsException at runtime
b. It will end without exceptions and will print nothing.

c. It will print Go away
d. It will print Go away and then throw ArrayIndexOutOfBoundsException
e. None of the above

How many objects have been created by the time the main method reaches its end on the following code?

```
public class Noobs {
    public Noobs() {
        try {
            throw new MyException();
        } catch (Exception e) {
        }
    }

    public static void main(String[] args) {
        Noobs a = new Noobs();
        Noobs b = new Noobs();
        Noobs c = a;
    }

    class MyException extends Exception {}
}
```

4

What will the following code print?

```
public class TestClass {
    static char ch;
    static float f;
    static boolean bool;

    public static void main(String[] args) {
        System.out.print(f);
        System.out.print(" ");
        System.out.print(ch);
        System.out.print(" ");
        System.out.print(bool);
    }
}
```

0.0 [null character] false

Which statements can be inserted at line 1 in the following code to make the program write x on the standard output when run?

```
public class AccessTest {
    static char b = 'x';
    String a = "y";
    class Inner {
        String c = "y";
        String get() {
            String temp = "temp";
            // Line 1
            return c;
        }
    }
    AccessTest() {
        System.out.println(new Inner().get());
    }
    public static void main(String args[]) {
        new AccessTest();
    }
}
```

a. c = "temp";
b. c = this.a;
c. c = "" + AccessTest.b;
d. c = AccessTest.this.a; ????????
e. c = "" + b;

// 18. ¿Cuál es el resultado?

```
public class SuperTest {
    public static void main(String[] args) {
        // Aquí deben ir las instrucciones para obtener el resultado deseado
        // statement1;
        // statement2;
        // statement3;
    }
}

class Shape {
    public Shape() {
        System.out.println("Shape: constructor");
    }

    public void foo() {
        System.out.println("Shape: foo");
    }
}

class Square extends Shape {
    public Square(String label) {
        super(); // Llama al constructor de la clase base (Shape)
        System.out.println("Square: constructor");
    }

    public void foo() {
        super.foo(); // Llama al método foo() de la clase base (Shape)
    }

    public void foo(String label) {
        System.out.println("Square: foo");
    }
}
```

What should statement1, statement2, and statement3, be respectively, in order to produce the result:
Shape: constructor
Square: foo
Shape: foo

A. Square square = new Square ("bar");
square.foo ("bar");
square.foo();
B. Square square = new Square ("bar");
square.foo ("bar");
square.foo ("bar");
C. Square square = new Square ();
square.foo ();
square.foo(bar);
D. Square square = new Square ();
square.foo ();
square.foo("bar");
E. Square square = new Square ();
square.foo ();
square.foo ();
F. Square square = new Square ();
square.foo("bar");
square.foo ();

//21. ¿Cuáles de las siguientes opciones son instanciaciones e inicializaciones válidas de un arreglo multidimensional? Elige dos

```
// Opción A: Correcta
int[][] array2D = {{0, 1, 2, 4}, {5, 6}}; // Inicialización directa de un arreglo bidimensional
```

```
// Opción B: Incompleta (pero válida hasta el punto de la declaración)
int[][] array2D = new int[2][2]; // Crea un arreglo bidimensional de 2x2
array2D[0][0] = 1;
array2D[0][1] = 2;
array2D[1][0] = 3;
array2D[1][1] = 4;

// Opción C: Incorrecta (error de sintaxis)
// int[][][] array3D = {{{0, 1}, {2, 3}, {4, 5}}}; // Faltan comas entre los elementos internos

// Opción D: Incorrecta
// array3D[0][0] = array; // No se puede asignar un arreglo completo a un elemento
// array3D[0][1] = array;
// array3D[1][0] = array;
// array3D[1][1] = array;
```

Which three are valid?

```
class ClassA {}
class ClassB extends ClassA {}
class ClassC extends ClassA {}

// Y las siguientes instancias de objetos:

ClassA p0 = new ClassA();
ClassB p1 = new ClassB();
ClassC p2 = new ClassC();
ClassA p3 = new ClassB();
ClassA p4 = new ClassC();

p0 = p1;
p1 = p2;
p2 = p4;
p2 = (ClassC)p1;
p1 = (ClassB)p3;
p2 = (ClassC)p4;
```

// 70. Selecciona la respuesta correcta con respecto al resultado del bloque de código.

```
public class Test5 {
   public static void main(String args[]) {
      Side primerIntento = new Head();
```

```
        Tail segundoIntento = new Tail();
        Coin.overload(primerIntento);
        Coin.overload((Object) segundoIntento);
        Coin.overload(segundoIntento);
        Coin.overload((Side) primerIntento);
    }
}

    interface Side {
        String getSide();
    }

    class Head implements Side {
        public String getSide() {
            return "Head";
        }
    }

    class Tail implements Side {
        public String getSide() {
            return "Tail";
        }
    }

    class Coin {
        public static void overload(Head side) {
            System.out.println(side.getSide());

        }

        public static void overload(Tail side) {
            System.out.println(side.getSide());
        }

        public static void overload(Side side) {
            System.out.println(side.getSide());

        }

        public static void overload(Object side) {
            System.out.println("Object");
        }
    }
```

Head Object Tail Side

```
public class Calculator {
    int num = 100;

    public void calc(int num) {
        this.num = num * 10;
    }

    public void printNum() {
        System.out.println(num);
    }

    public static void main(String[] args) {
        Calculator obj = new Calculator();
        obj.calc(2);
        obj.printNum();

    }
}
```

20

```
class Feline {
    public String type = "f";
    public Feline() {
        System.out.print("feline ");
    }
}

public class Cougar extends Feline {
    public Cougar() {
        System.out.print("cougar ");
    }

    void go() {
        type = "c";
        System.out.print(this.type + super.type);
    }

    public static void main(String[] args) {
        new Cougar().go();

    }
}
```

feline cougar cc

What is the result?

```
interface Rideable {
    String getGait(); //is public abstract
}

public class Camel implements Rideable {
    int weight = 2;
    String getGait() //should be public {
        return " mph, lope";
    }

    void go(int speed) {
        ++speed;
        weight++;
        int walkrate = speed * weight;
        System.out.print(walkrate + getGait());
    }

    public static void main(String[] args) {
        new Camel().go(8);
    }
}

// 16 mph, lope
// 24 mph, lope.
// 27 mph, lope.
// Compilation fails
```

Which class has a default constructor?

```
class X {}

class Y {
    Y() {}
}

class Z {
    Z(int i) {}
}
```

Opciones de respuesta:

Z only.
X only.
X, Y and Z.

[✓ X and Y.]
X and Z.
Y only.
Y and Z.

Which of the following implementations of a max() method will correctly return the largest value?

```
int max(int x, int y) {
    return (if (x > y) { x; } else { y; });
}

---------------------------------------------------------------------

int max(int x, int y) {
    return(if (x > y) { return x; } else (return y; } );
}

---------------------------------------------------------------------

int max(int x, int y) {
    switch (x < y) {
        case true:
            return y;
        default:
            return x;
    }
}

---------------------------------------------------------------------
```

[✓
```
int max(int x, int y) {
    if (x > y)
        return x;
    return y;
}
```
]

What will the following code print when run without any arguments?

```
public class TestClass {
    public static int ml(int i) {
        return i++;
    }
    public static void main(String[] args) {
        int k = ml(args.length);
        k += 3 + ++k;
        System.out.println(k);

    }
}
```

It Will throw ArrayIndexPutOfBoundsException
It Will throw NullPointerException
6
5
7
2

What Will the following code print?

```
int i = 1;
int j = i++;
if ((i == ++j) | (i++ == j)) {
    i += j;
}
System.out.println(i);
```

5