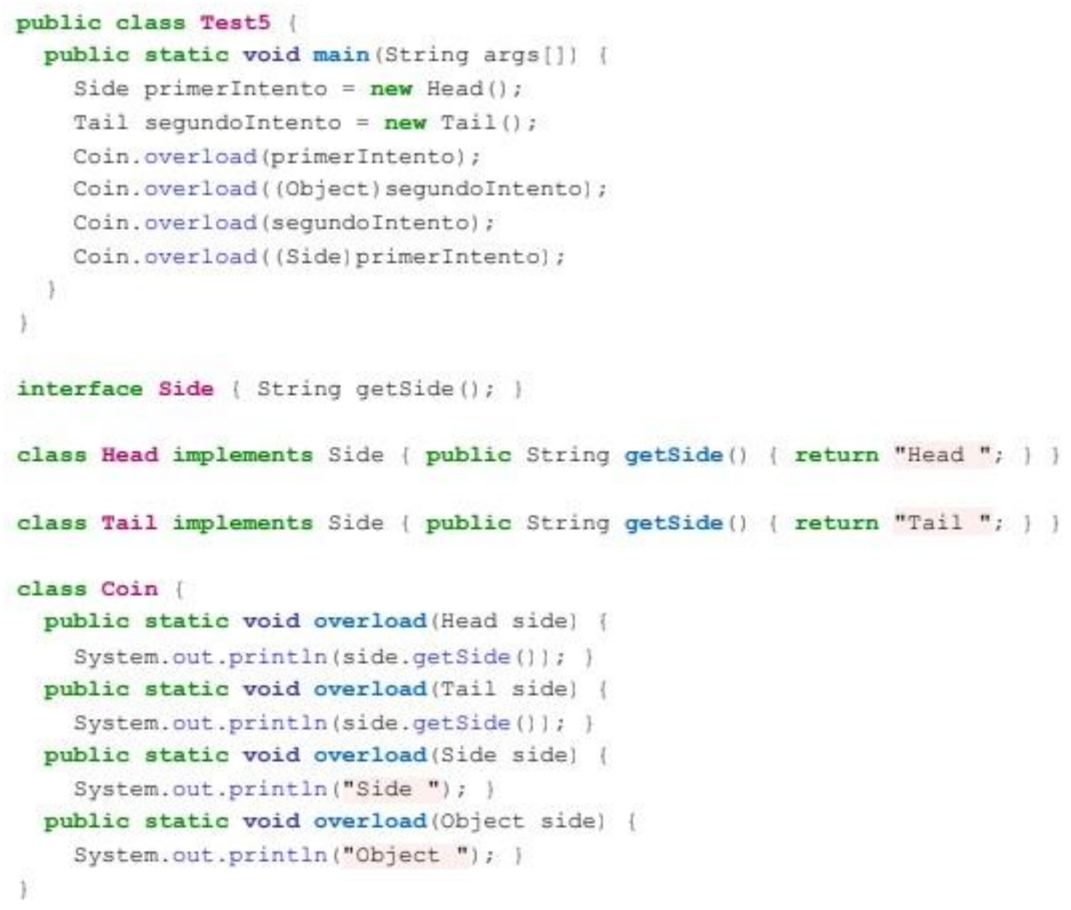
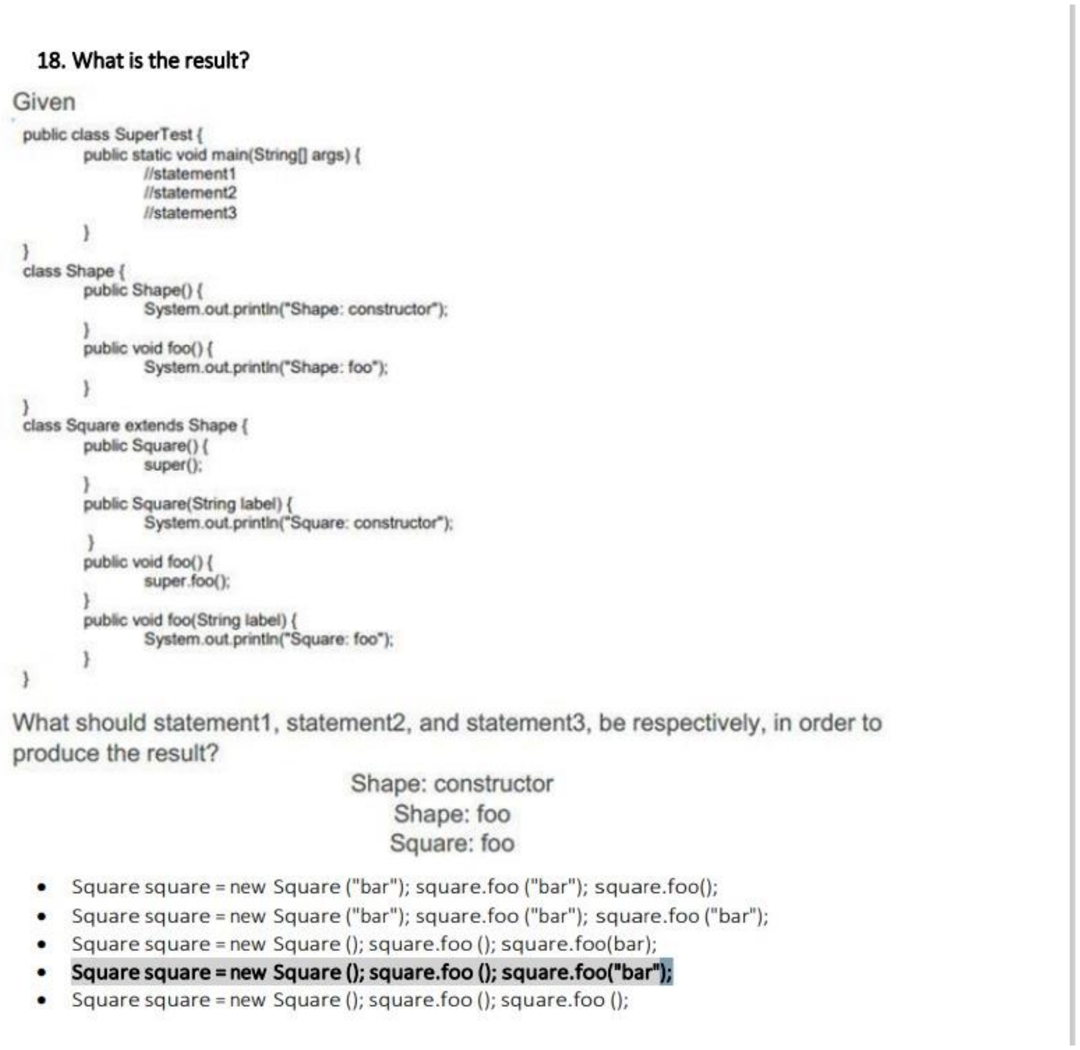
70. Selecciona la respuesta correcta con respecto al resultado del bloque de código.



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

La clase Coin tiene 4 métodos overload que imprimen diferentes cadenas de texto en la consola dependiendo del tipo de parámetro que reciben:

1. overload(Head side): Imprime "side.getSide()"
2. overload(Tail side): Imprime "side.getSide()"
3. overload(Side side): Imprime "Side"
4. overload(Object side): Imprime "Object"



18. What is the result?

Given

```
public class SuperTest {
        public static void main(String[] args) {
                //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() {
                super();
        }
        public Square(String label) {
                System.out.println("Square: constructor");
        }
        public void foo() {
                super.foo();
        }
        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
Shape: foo
Square: foo
```

- Square square = new Square ("bar"); square.foo ("bar"); square.foo();
- Square square = new Square ("bar"); square.foo ("bar"); square.foo ("bar");
- Square square = new Square (); square.foo (); square.foo(bar);
- **Square square = new Square (); square.foo (); square.foo("bar");**
- Square square = new Square (); square.foo (); square.foo ();

Square square =new Square ("bar"); square.foo("bar"); square.foo();
Esto se debe a que:

1. Crea una nueva instancia de la clase Square con el parámetro "bar".
2. Llama al método foo() de la instancia de Square, pasando "bar" como parámetro.
3. Llama al método foo() de la instancia de Square sin parámetros.

Estas tres instrucciones juntas producirán el resultado deseado:

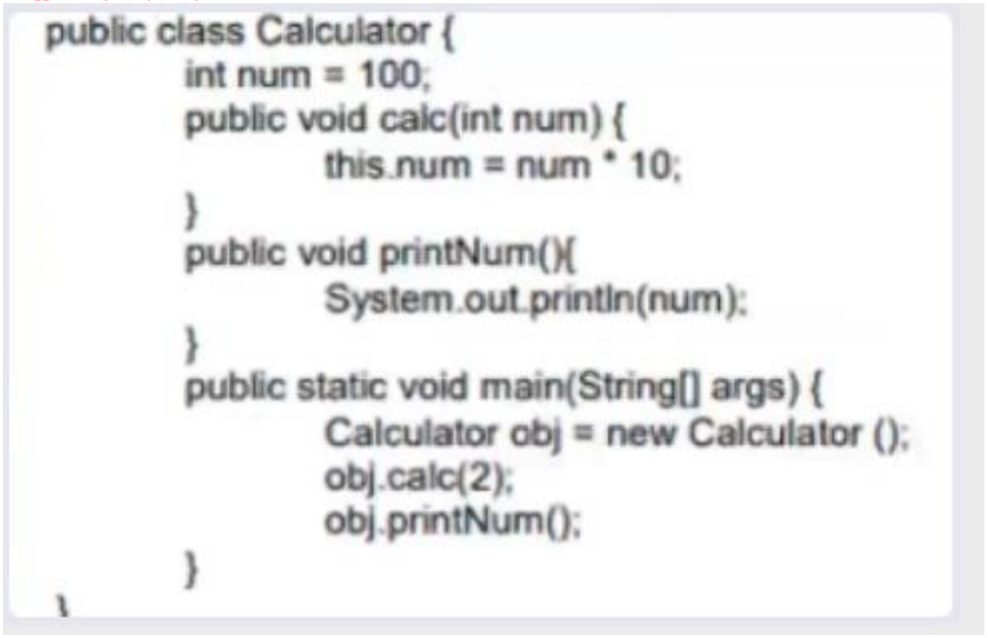
- Shape: constructor
- Shape: foo
- Square: foo

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

```
A. int[][] array2D = {{0, 1, 2, 4}{5, 6)};

B. int[][] array2D = new int[][2];
   array2D[0][0] = 1;
   array2D[0][1] = 2;
   array2D[1][0] = 3;
   array2D[1][1] = 4;
   int[] array3D = new int[2][2][2];

C. int[][][] array3D = {{{0, 1}, {2, 3}, {4, 5}}};
   int[] array = {0, 1};

D. array3D[0][0] = array;
   array3D[0][1] = array;
   array3D[1][0] = array;
   array3D[1][1] = array;
```

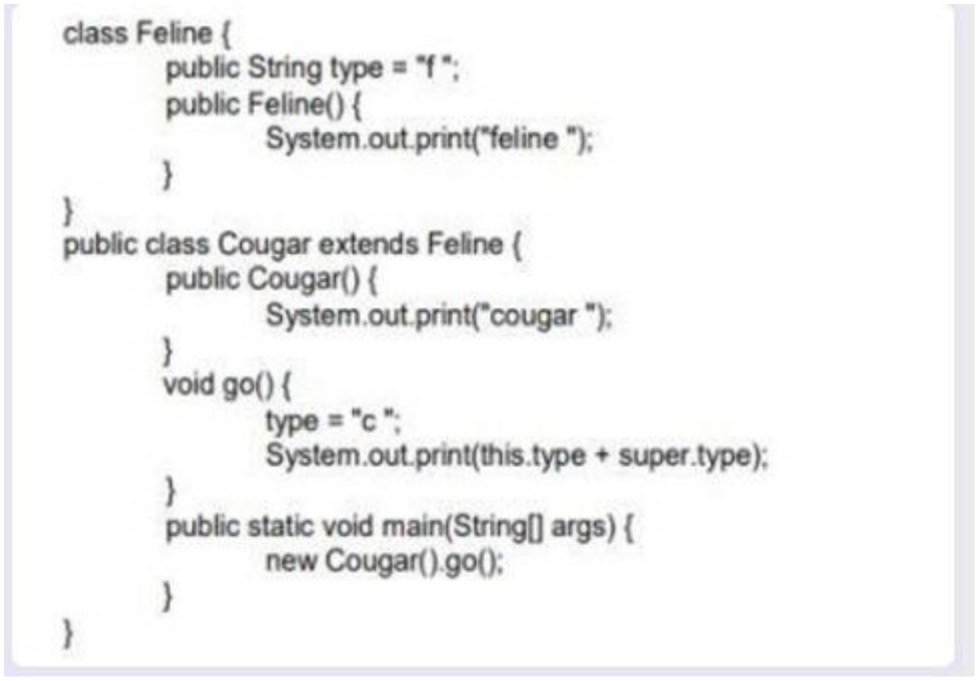
B. int[][] array2D = new int[2][2];
array2D[0][0] = 1;
array2D[0][1] = 2;
array2D[1][0] = 3;
array2D[1][1] = 4;
int[][] array3D = new int[2][2][2];

C. int[][][] array3D = {{{0, 1}, {2, 3}, {4, 5}}};
int[] array = {0, 1};



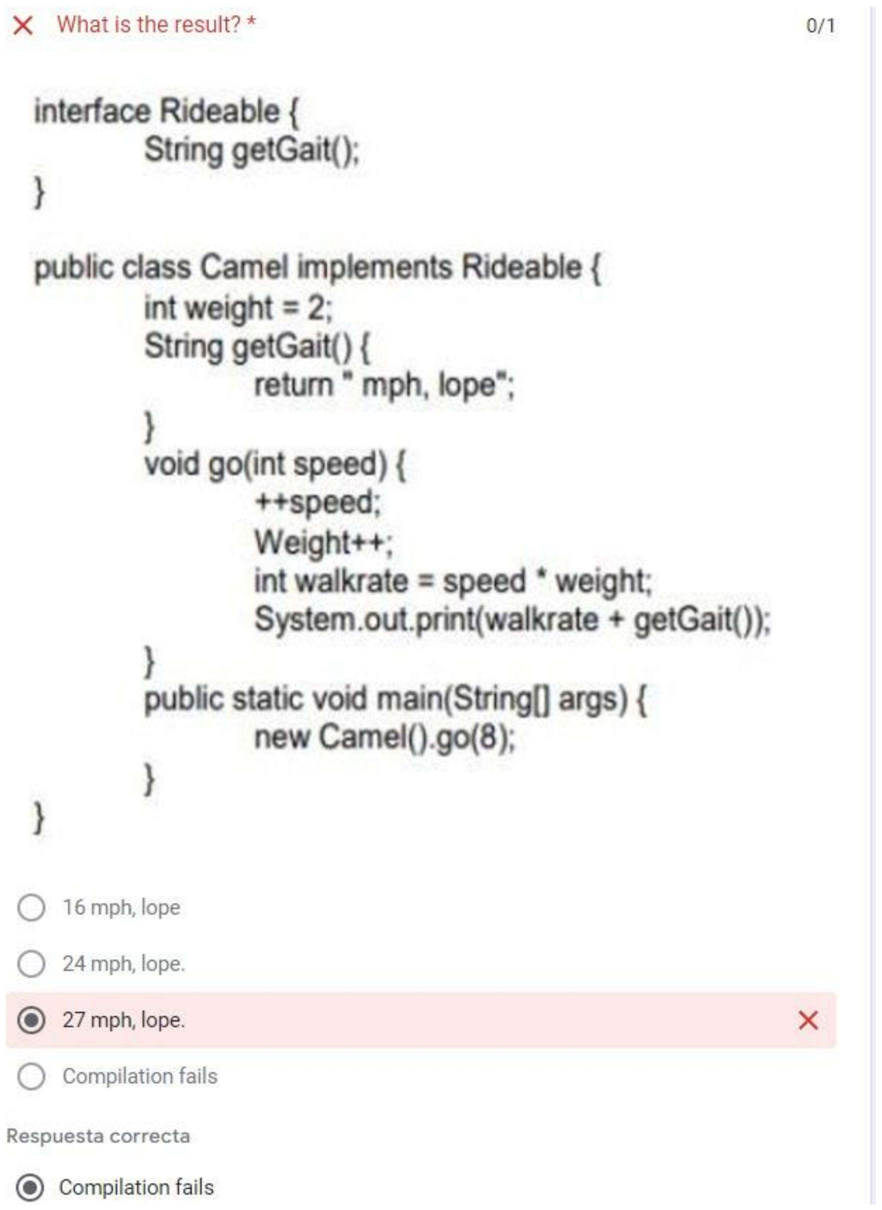
```
public class Calculator {
        int num = 100;
        public void calc(int num) {
                this.num = num * 10;
        }
        public void printNum(){
                System.out.println(num);
        }
        public static void main(String[] args) {
                Calculator obj = new Calculator ();
                obj.calc(2);
                obj.printNum();
        }
}
```

Este código crea una calculadora simple que multiplica un número por 10 y luego lo imprime a la consola.



```
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();
        }
}
```

Este código define una jerarquía de clases donde "Cougar" hereda de "Feline". El método "go()" de "Cougar" modifica el valor de la variable "type" y luego imprime los valores de "this.type" y "super.type".



What is the result? *

0/1

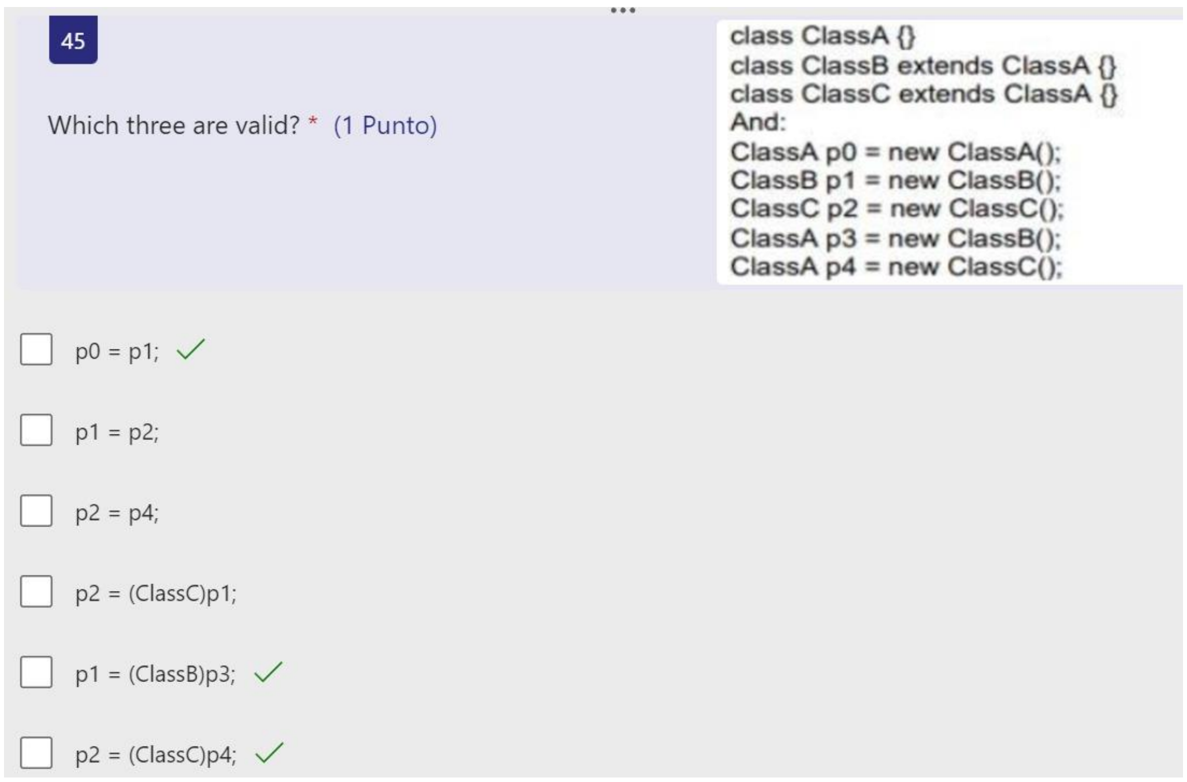
```
interface Rideable {
        String getGait();
}

public class Camel implements Rideable {
        int weight = 2;
        String getGait() {
                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

Respuesta correcta

- [✓ Compilation fails]

En el método "go()", la línea "++speed;" incrementa la velocidad, pero no se asigna el resultado de vuelta a la variable "speed". Esto causaría un error de compilación.



45

Which three are valid? * (1 Punto)

```
class ClassA {}
class ClassB extends ClassA {}
class ClassC extends ClassA {}
And:
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;

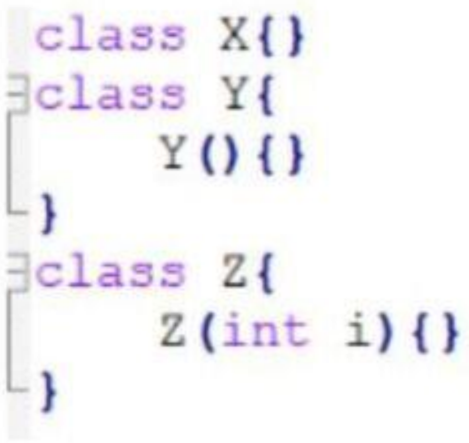
☐ p2 = (ClassC)p1;

☐ p1 = (ClassB)p3; ✓

☐ p2 = (ClassC)p4; ✓

Estas opciones son válidas porque cumplen con las asignaciones definidas en la clase "ClassA", "ClassB" y "ClassC" que se muestran en la parte superior de la imagen.

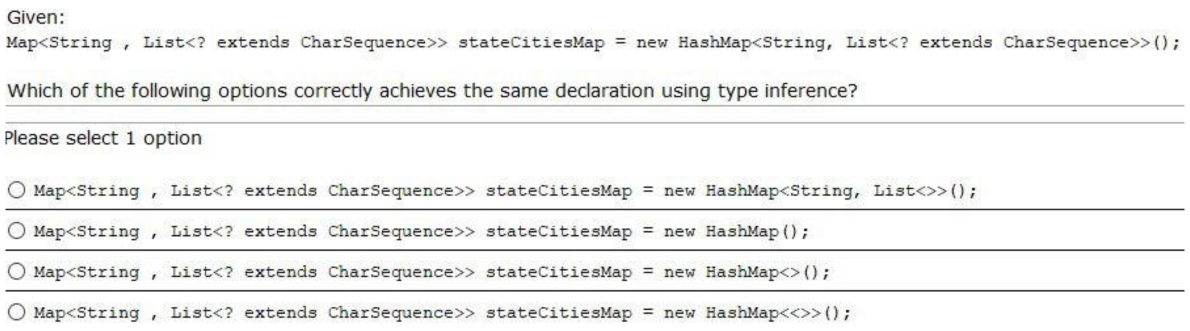
69. Which class has a default constructor?



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

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

La clase que tiene un constructor por defecto es la clase "X".



Given:

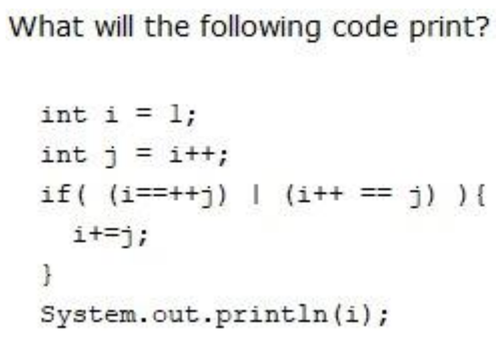
```
Map<String , List<? extends CharSequence>> stateCitiesMap = new HashMap<String, List<? extends CharSequence>>();
```

Which of the following options correctly achieves the same declaration using type inference?

Please select 1 option

- ○ `Map<String , List<? extends CharSequence>> stateCitiesMap = new HashMap<String, List<>>();`
- ○ `Map<String , List<? extends CharSequence>> stateCitiesMap = new HashMap();`
- ○ `Map<String , List<? extends CharSequence>> stateCitiesMap = new HashMap<>();`
- ○ `Map<String , List<? extends CharSequence>> stateCitiesMap = new HashMap<<>>();`

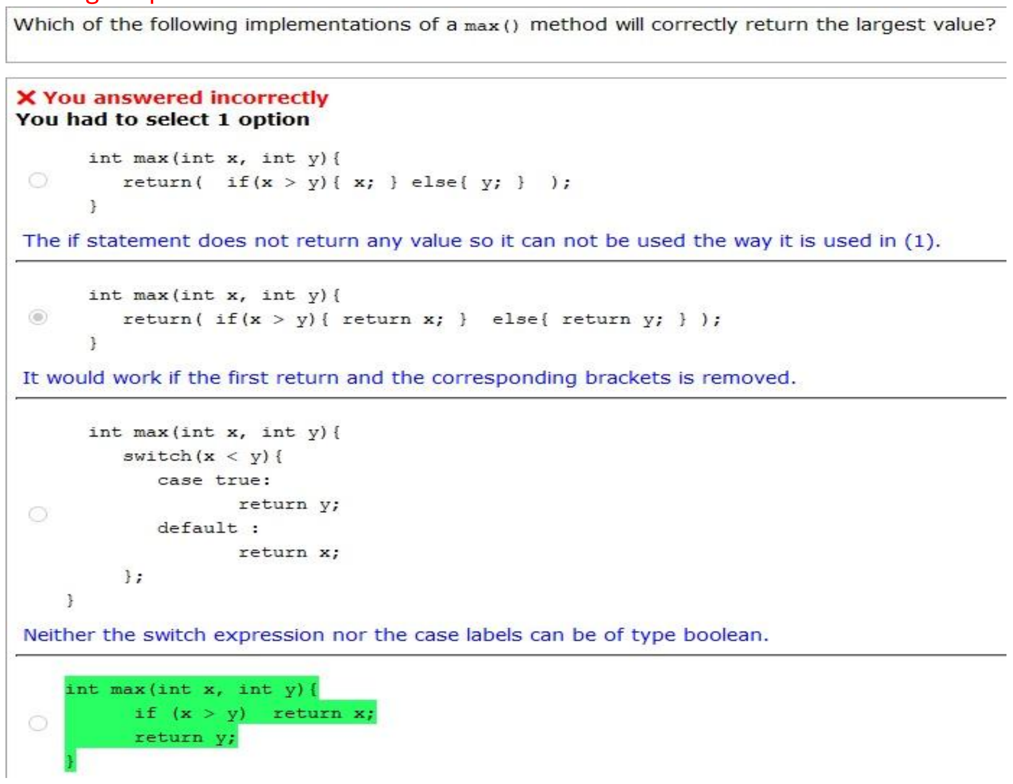
Map<String, List<?extends CharSequence>> stateCitiesMap = new HashMap<>();



What will the following code print?

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

El código imprimirá "4".



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

✗ You answered incorrectly
You had to select 1 option

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

The if statement does not return any value so it can not be used the way it is used in (1).

[✓]

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

It would work if the first return and the corresponding brackets is removed.

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

Neither the switch expression nor the case labels can be of type boolean.

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

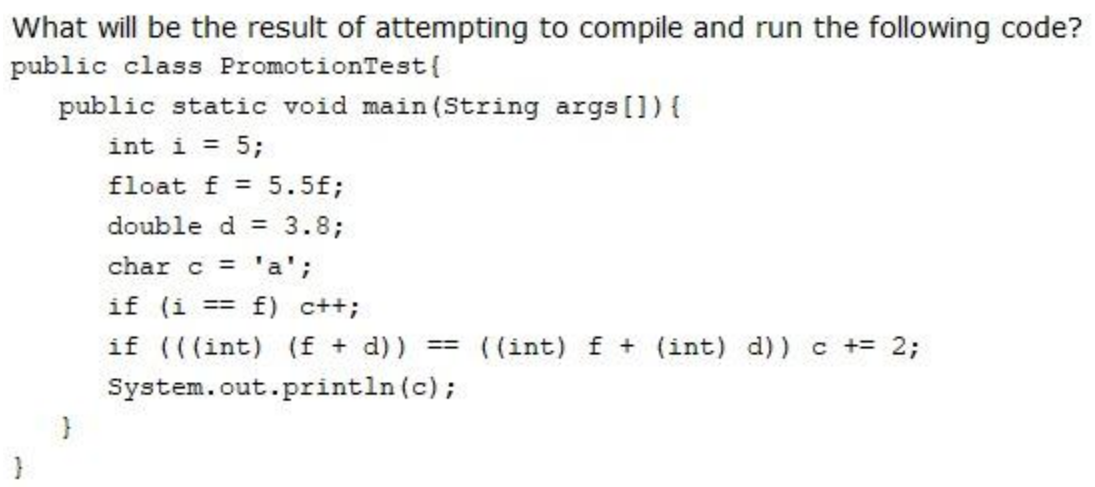
What will the following code print when run without any arguments ...

```
public class TestClass {

    public static int m1(int i){
        return ++i;
    }

    public static void main(String[] args) {

        int k = m1(args.length);
        k += 3 + ++k;
        System.out.println(k);
    }

}
```

✓ You answered correctly

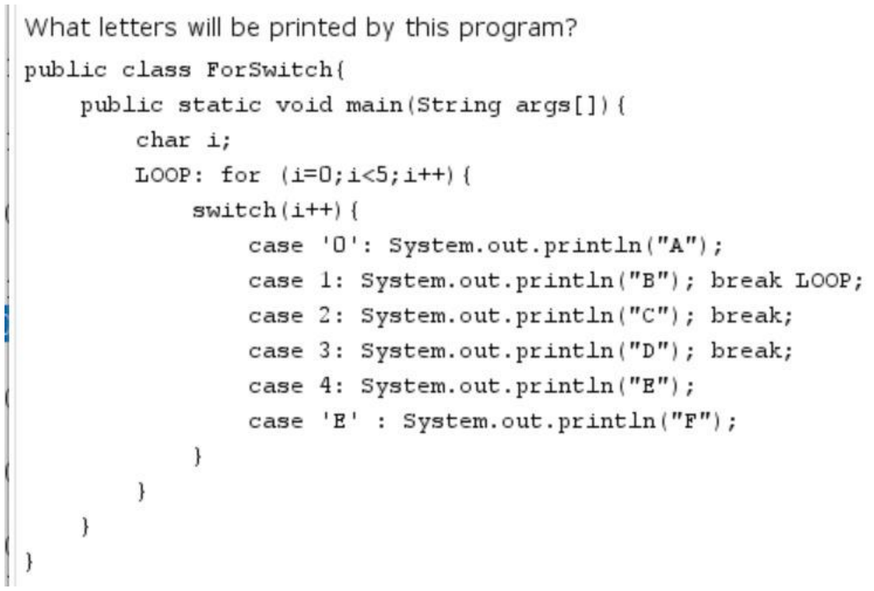
You had to select 1 option

- ○ It will throw `ArrayIndexOutOfBoundsException`.
- ○ It will throw `NullPointerException`.
- [✓ 6]
- ○ 5
- ○ 7
- ○ 2


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

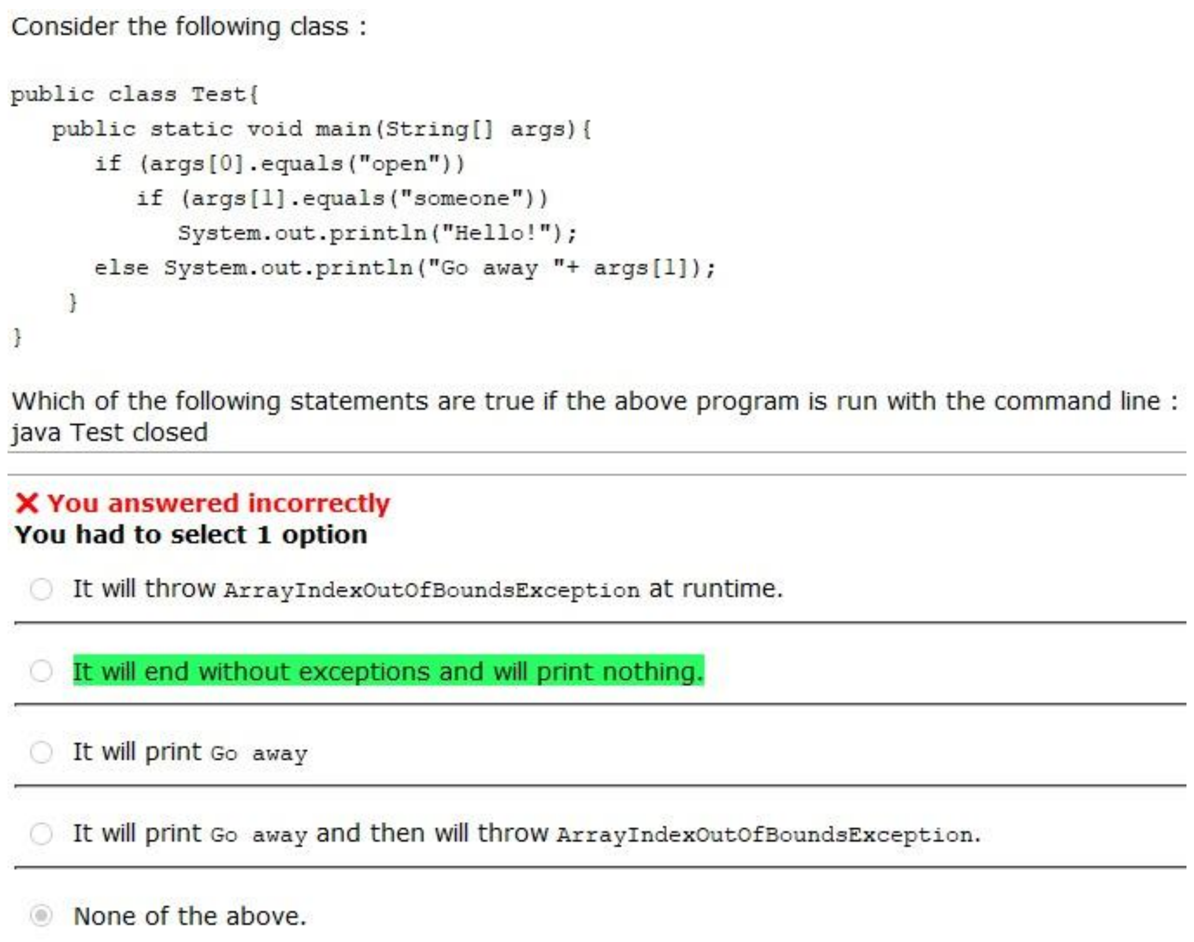
```
public class PromotionTest{
   public static void main(String args[]){
      int i = 5;
      float f = 5.5f;
      double d = 3.8;
      char c = 'a';
      if (i == f) c++;
      if (((int) (f + d)) == ((int) f + (int) d)) c += 2;
      System.out.println(c);
   }
}
```

El código tiene varios problemas de compilación relacionados con la conversión de tipos de datos y la comparación de tipos incompatibles. Antes de poder ejecutar este código.


What letters will be printed by this program?

```
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");
            }
        }
    }
}
```

A
B
C
D
E
F



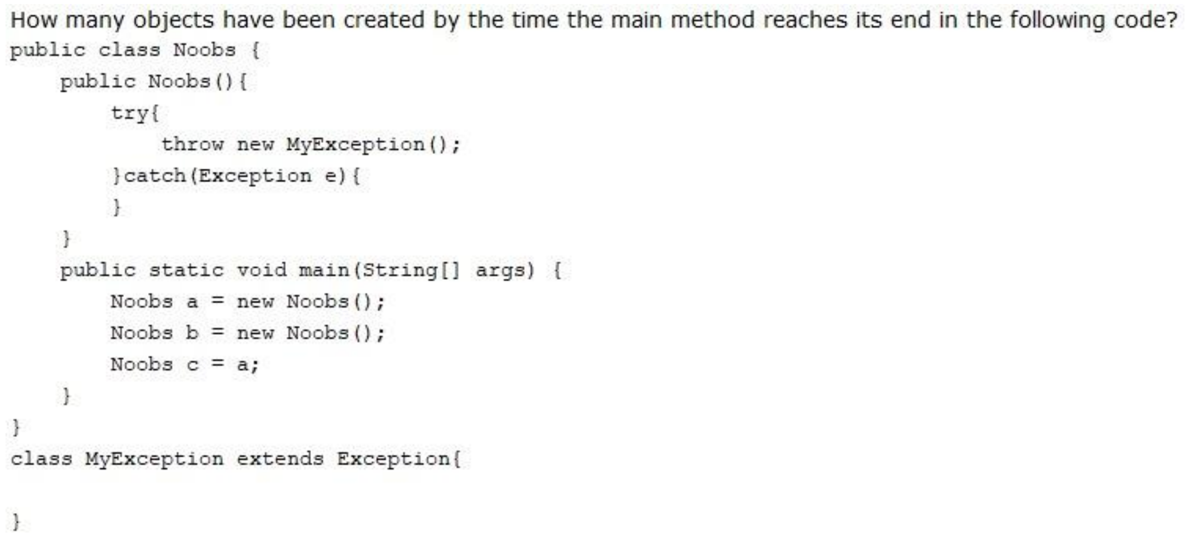
Consider the following class :

```
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

✕ You answered incorrectly
You had to select 1 option

- It will throw `ArrayIndexOutOfBoundsException` at runtime.
- It will end without exceptions and will print nothing.
- It will print `Go away`
- It will print `Go away` and then will throw `ArrayIndexOutOfBoundsException`.
- [✓ None of the above.]

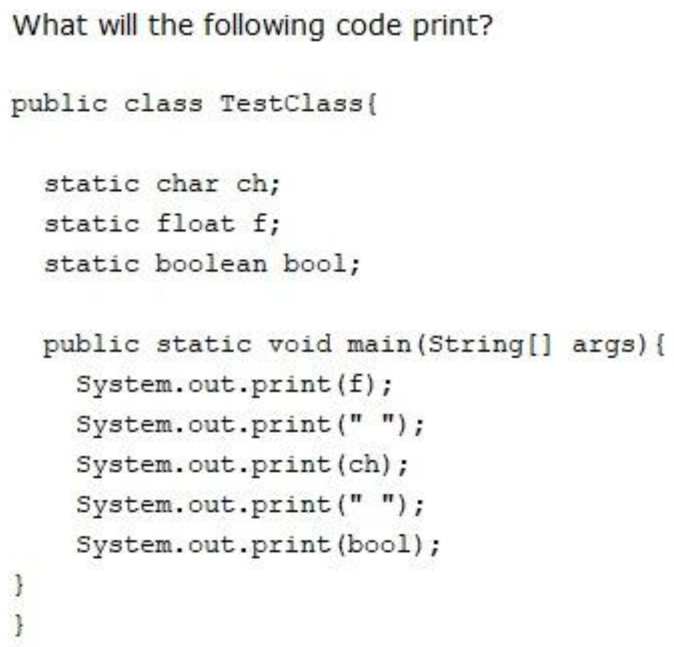


How many objects have been created by the time the main method reaches its end in the following code?

```
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{

}
```

El número de objetos creados antes de que el método main termine es 3.

- Noobs a = new Noobs();
- Noobs b = new Noobs();
- Noobs c = a;



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
0.0
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{
   String a = "x";
   static char b = 'x';
   String  c = "x";
   class Inner{
      String  a = "y";
      String  get(){
         String c = "temp";
         // Line 1
         return c;
      }
   }

   AccessTest() {
     System.out.println(  new Inner().get()  );
   }

   public static void main(String args[]) {  new AccessTest();  }
}
```

✗ You answered incorrectly
You had to select 3 options

[✓ `c = c;`]

It will reassign 'temp' to c!

[✓ `c = this.a;`]

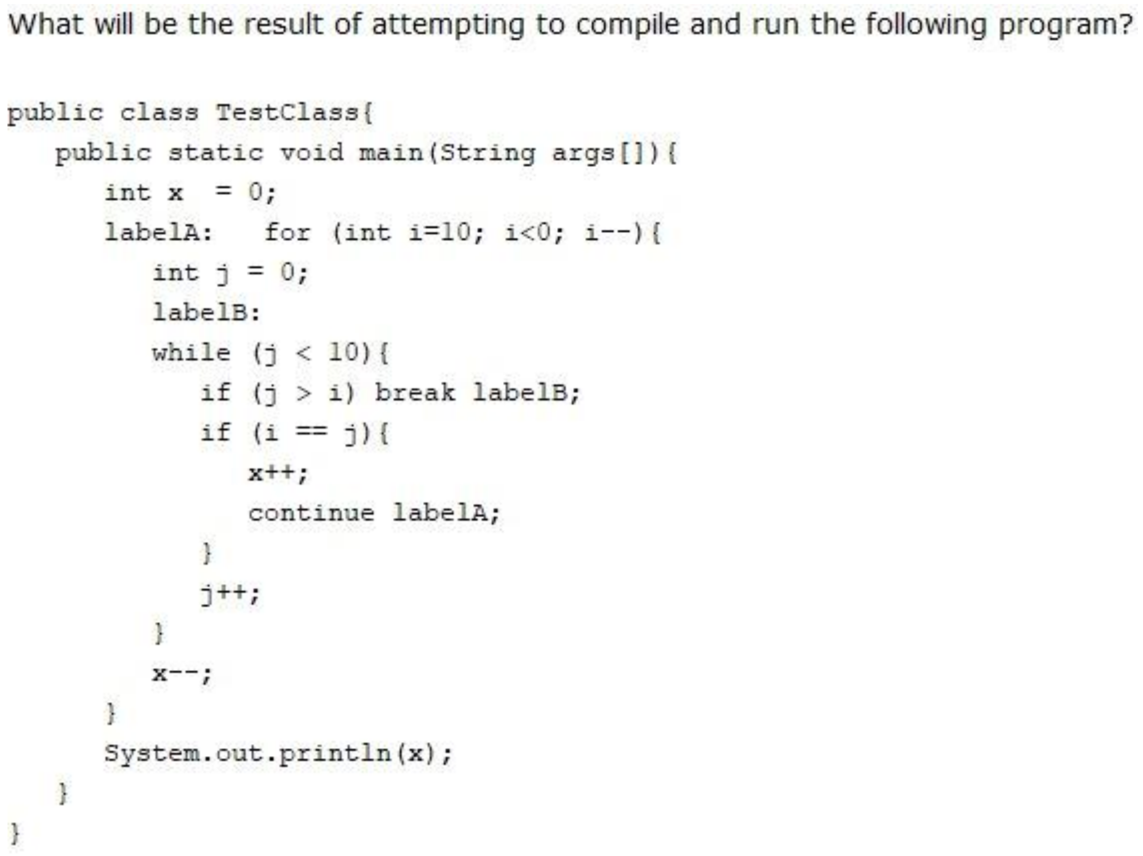
It will assign "y" to c.

[✓ `c = ""+AccessTest.b;`]

Because b is static.

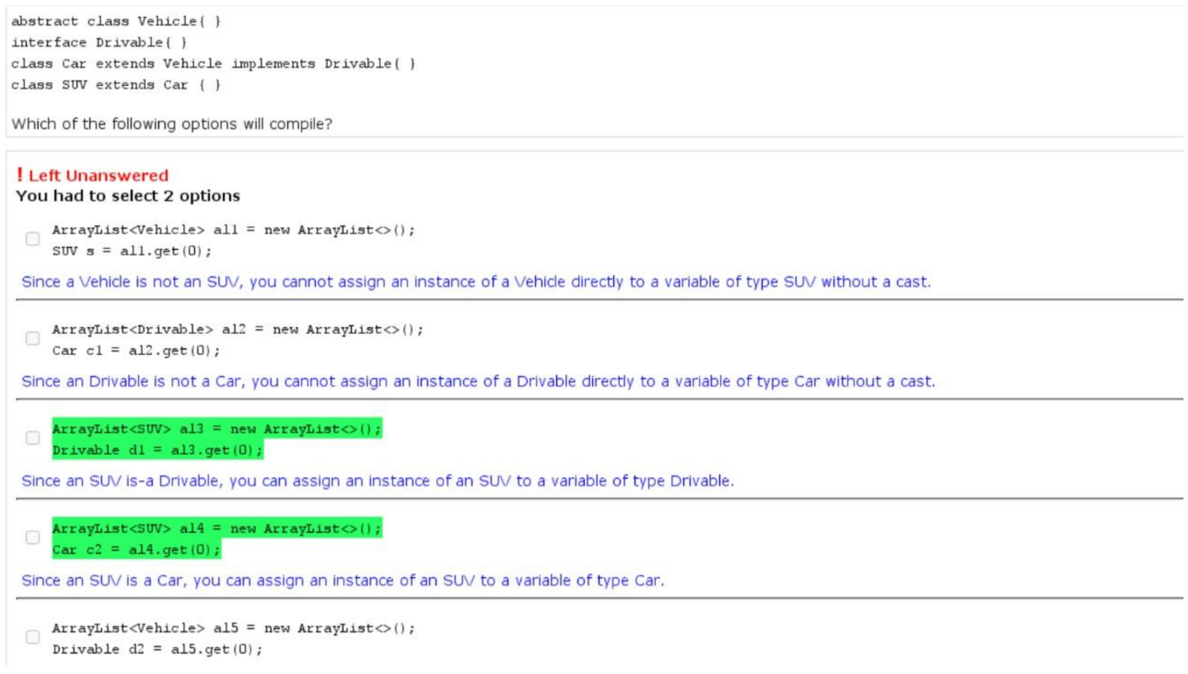
☐ `c = AccessTest.this.a;`

☐ `c = ""+b;`


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);
   }
}
```

Al intentar compilar y ejecutar este programa, se producirá un error de compilación. La razón es que la etiqueta labelB se declara pero no se utiliza.



```
abstract class Vehicle{ }
interface Drivable{ }
class Car extends Vehicle implements Drivable{ }
class SUV extends Car { }
```

Which of the following options will compile?

! Left Unanswered
You had to select 2 options

```
ArrayList<Vehicle> al1 = new ArrayList<>();
SUV s = al1.get(0);
```

Since a Vehicle is not an SUV, you cannot assign an instance of a Vehicle directly to a variable of type SUV without a cast.

```
ArrayList<Drivable> al2 = new ArrayList<>();
Car c1 = al2.get(0);
```

Since an Drivable is not a Car, you cannot assign an instance of a Drivable directly to a variable of type Car without a cast.

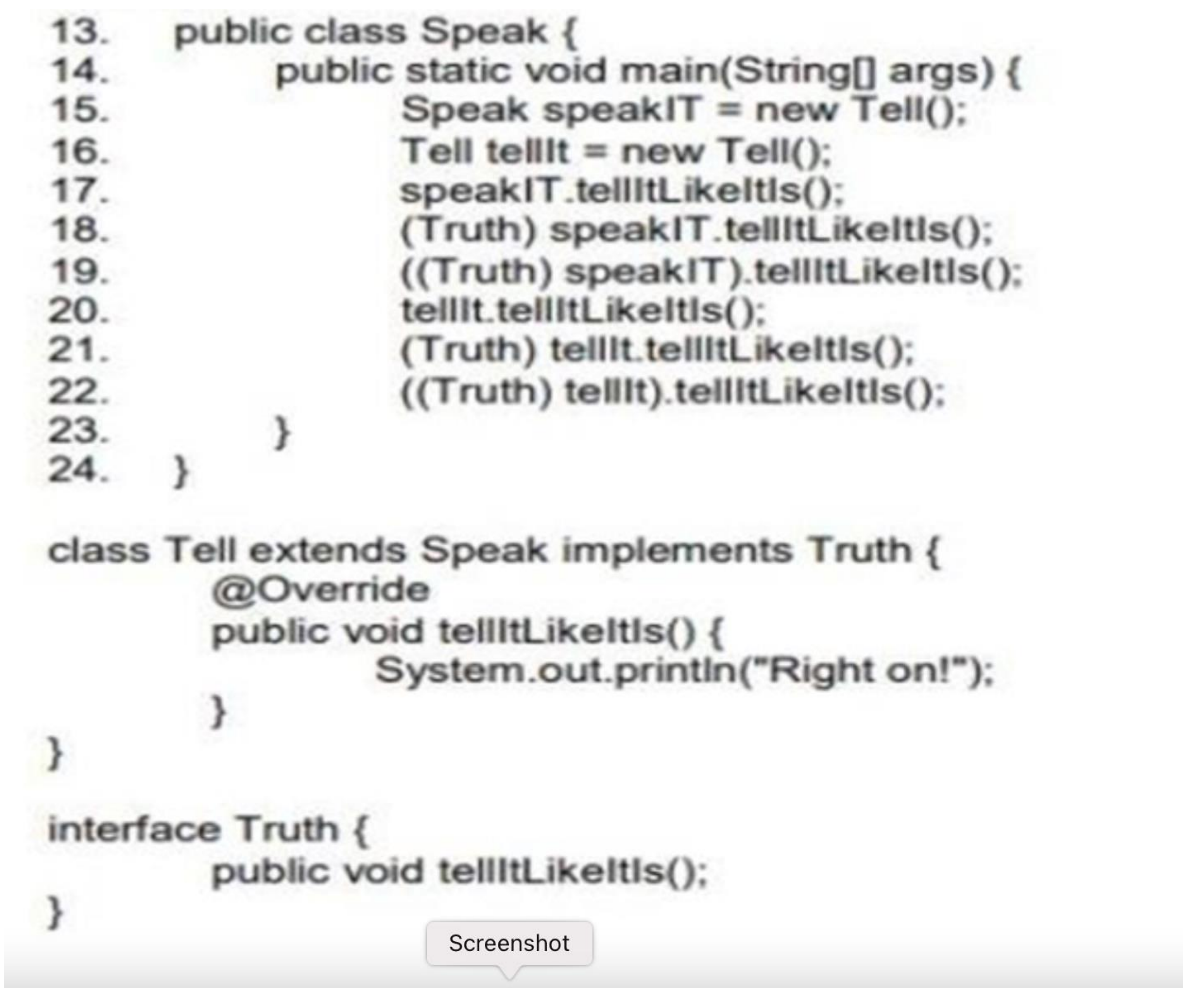
```
ArrayList<SUV> al3 = new ArrayList<>();
Drivable d1 = al3.get(0);
```

Since an SUV is-a Drivable, you can assign an instance of an SUV to a variable of type Drivable.

```
ArrayList<SUV> al4 = new ArrayList<>();
Car c2 = al4.get(0);
```

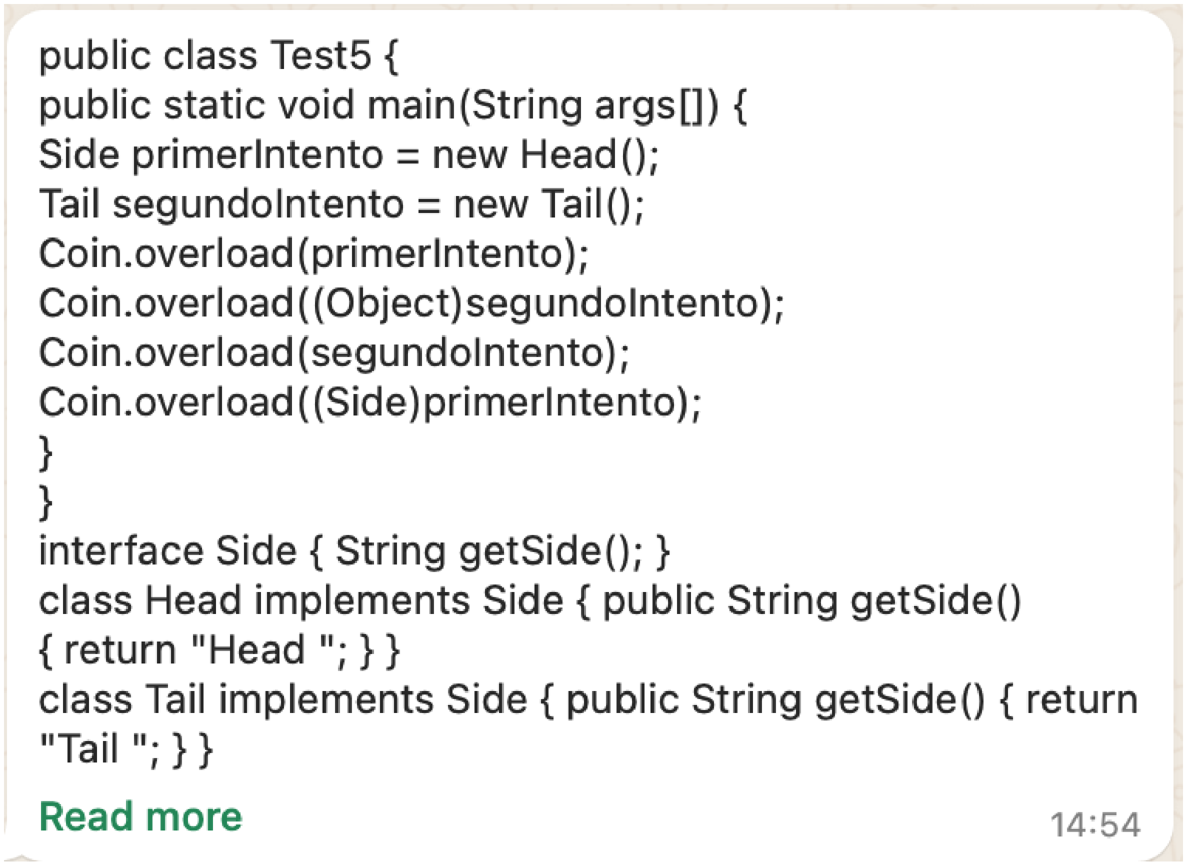
Since an SUV is a Car, you can assign an instance of an SUV to a variable of type Car.

```
ArrayList<Vehicle> al5 = new ArrayList<>();
Drivable d2 = al5.get(0);
```



```
public class Speak {
        public static void main(String[] args) {
                Speak speakIT = new Tell();
                Tell tellIt = new Tell();
                speakIT.tellItLikeItIs();
                (Truth) speakIT.tellItLikeItIs();
                ((Truth) speakIT).tellItLikeItIs();
                tellIt.tellItLikeItIs();
                (Truth) tellIt.tellItLikeItIs();
                ((Truth) tellIt).tellItLikeItIs();
        }
}

class Tell extends Speak implements Truth {
        @Override
        public void tellItLikeItIs() {
                System.out.println("Right on!");
        }
}

interface Truth {
        public void tellItLikeItIs();
}
```

El código proporcionado parece tener varios errores de sintaxis y de lógica



```
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 "; } }
```