1.- Which declaration initializes a boolean variable?

- ➢ a) boolean m = null
- ➢ b) Boolean j = (1<5)
- ➢ c) boolean k = 0
- ➢ d) boolean h = 1

2.- What is the DTO pattern used for?

- ➢ a) To Exchange data between processes
- ➢ b) To implement the data Access layer
- ➢ c) To implement the presentation layer

3.- Int a = 10; int b = 37; int z = 0; int w = 0;

If (a==b) {z=3; } else if (a>b) {z=6;}

w = 10 * z;

What is the result?

- ➢ a) 30
- ➢ b) 0
- ➢ c) 60

```
class Class1{String v1;}
class Class2{
        Class1 c1;
        String v2;
}
class Class3 {Class2 c1; String v3;}
```

4.- Which three options correctly describe the relationship between the classes?

- ➢ A. Class2 has-a v3.
- ➢ B. Class1 has-a v2.
- ➢ C. Class2 has-a v2.
- ➢ D. Class3 has a v1.
- ➢ E. Class2 has-a Class3.
- ➢ F. Class2 has-a Class1.

Pregunta 5

```
try {
   // assume "conn" is a valid Connection object
   // assume a valid Statement object is created
   // assume rollback invocations will be valid
   // use SQL to add 10 to a checking account
   Savepoint s1 = conn.setSavePoint();
   // use SQL to add 100 to the same checking account
   Savepoint s2 = conn.setSavePoint();
   // use SQL to add 1000 to the same checking account
   // insert valid rollback method invocation here
} catch (Exception e) {}
```

What is the result?

A) If conn.rollback(s1) is inserted, account will be incremented by 10.
B) If conn.rollback(s1) is inserted, account will be incremented by 1010.
C) If conn.rollback(s2) is inserted, account will be incremented by 100.
D) If conn.rollback(s2) is inserted, account will be incremented by 110.
E) If conn.rollback(s2) is inserted, account will be incremented by 1110.

Pregunta 6

Which two statements are true an Abstract ?

A) An abstract class can implement an interface.
B) An abstract class can be extended by an interface.
C) An interface CANNOT be extended by another interface.
D) An interface can be extended by an abstract class.
E) An abstract class can be extended by a concrete class.
F) An abstract class CANNOT be extended by an abstract class.

Pregunta 7

```
public class Main {
   public static void main(String[] args) throws Exception {
      doSomething();
   }

   private static void doSomething() throws Exception {
      System.out.println("Before if clause");
      if (Math.random() > 0.5) {
          throw new Exception();
      }
      System.out.println("After if clause");
   }
}
```

Which two are possible outputs?

A) Before if clause Exception in thread “main” java.lang.Exception at Main.doSomething (Main.java:21) at Main.main (Main.java:15).
B) Before if clause Exception in thread "main" java.lang.Exception at Main.doSomething (Main.java:21) at Main.main (Main.java:15) After if clause
C) Exception in thread "main" java.lang.Exception at Main.doSomething (Main.java:21) at Main.main (Main.java:15)
D) Before if clause After if clause

Explicación:
La opción a se produce cuando se evalúa en el if si el valor generado por Math.random() es mayor que 0.5. En ese caso, se lanza una excepción y no se ejecuta la línea System.out.println("After if clause");. Por lo tanto, solo se imprime "Before if clause" seguido del stack trace de la excepción.

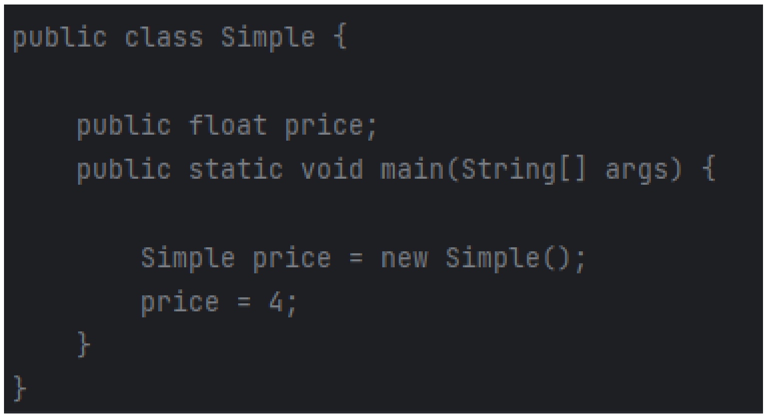
La opción d se produce cuando el valor generado por Math.random() es menor o igual a 0.5. Debido a que no entra en el if, no se lanza una excepción y ambas líneas System.out.println("Before if clause"); y System.out.println("After if clause"); se ejecutan.

Pregunta 8

```
public class MyFive {
    public static void main(String[] args) {
        //short kk = ?;
        short ii;
        short jj = 0;
        for (ii = kk; ii > 6; ii-=1) {
                    jj++;
        }
        System.out.println("jj = " + jj);
    }
}
```

What value should replace kk in line 18 to cause jj = 5 to be output?

A) -1
B) 1
C) 5
D) 8
E) 11



```
public class Simple {

    public float price;
    public static void main(String[] args) {

        Simple price = new Simple();
        price = 4;
    }
}
```

Pregunta 9

What will make this code compile and run?

A. Change line 3 to the following: public int price;
B. Change line 7 to the following: int price = new Simple();
C. Change line 7 to the following: float price = new Simple();
D. Change line 7 to the following: price = 4f;
E. Change line 7 to the following: price.price = 4;

Explicación:

Si quisiéramos cambiar la referencia de el price de la clase, tendríamos que ingresar al objeto para hacerlo, de otra forma estamos intentando cambiar la referencia de Simple price, y 4 (int) no es de la clase Simple.

Pregunta 10

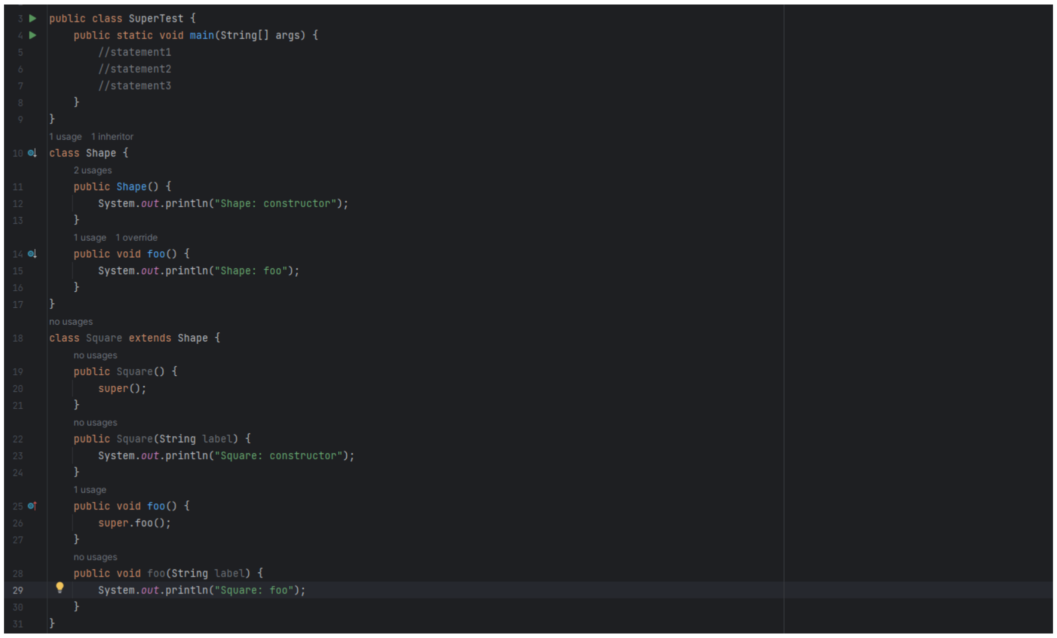
In the Java collections framework a Set Is:

A. A collection that cannot contain duplicate elements.
B. An ordered collection that can contain duplicate elements.
C. An object that maps value key sets and cannot contain values Duplicates.

Explicación:

Un Set pertenece a collection y no permite el tener elementos duplicados en su interior.

Los otros dos puntos hacen referencia a List y Map, respectivamente.



```java
public class SuperTest {
    public static void main(String[] args) {
        //statement1
        //statement2
        //statement3
    }
}
1 usage  1 inheritor
class Shape {
    2 usages
    public Shape() {
        System.out.println("Shape: constructor");
    }
    1 usage  1 override
    public void foo() {
        System.out.println("Shape: foo");
    }
}
no usages
class Square extends Shape {
    no usages
    public Square() {
        super();
    }
    no usages
    public Square(String label) {
        System.out.println("Square: constructor");
    }
    1 usage
    public void foo() {
        super.foo();
    }
    no usages
    public void foo(String label) {
        System.out.println("Square: foo");
    }
}
```

Pregunta 11

What should statement1, statement2, and statement3, be respectively, in order to produce the result?

Shape: constructor

Shape: foo

Square: 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();

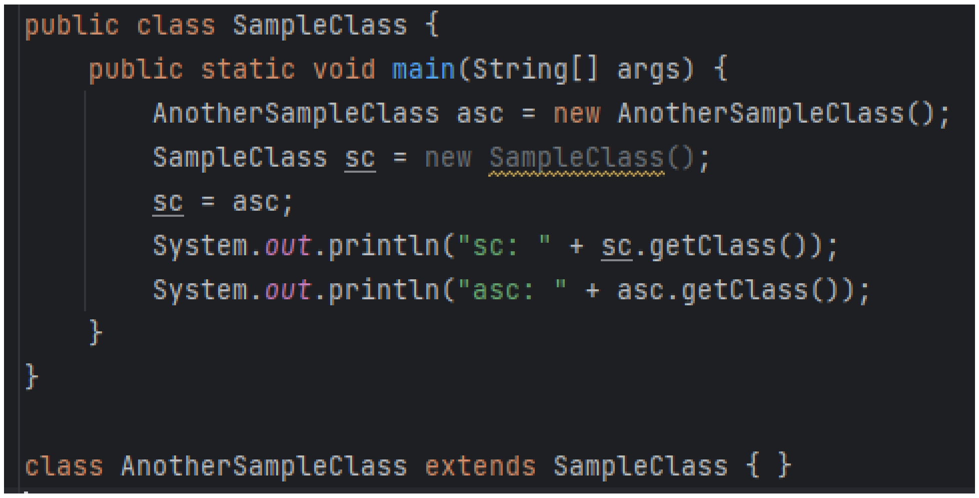
Explicación:

Solo en la opción D obtenemos el resultado que buscamos, ya que:

Square square = new Square();: Imprime "Shape: constructor".

square.foo();: Imprime "Shape: foo".

square.foo("bar");: Imprime "Square: foo".



```java
public class SampleClass {
    public static void main(String[] args) {
        AnotherSampleClass asc = new AnotherSampleClass();
        SampleClass sc = new SampleClass();
        sc = asc;
        System.out.println("sc: " + sc.getClass());
        System.out.println("asc: " + asc.getClass());
    }
}

class AnotherSampleClass extends SampleClass { }
```

Pregunta 12
What is the result?

A. sc: class.Object asc: class.AnotherSampleClass

B. sc: class.SampleClass asc: class.AnotherSampleClass

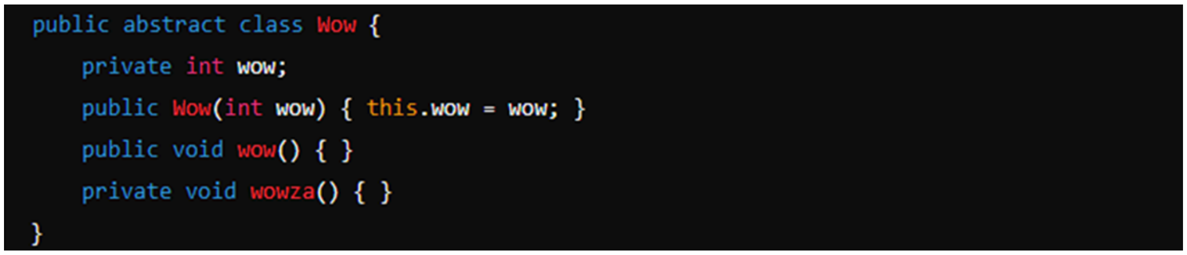
C. sc: class.AnotherSampleClass asc: class.SampleClass

D. sc: class.AnotherSampleClass asc: class.AnotherSampleClass

Explicación:

getClass evalúa el la clase de el objeto, y no de la variable de referencia, por lo que al pasarle a sc el objeto AnotherSampleClass() , nos da este resultado.

13. What is true about the class Wow?



```java
public abstract class Wow {
    private int wow;
    public Wow(int wow) { this.wow = wow; }
    public void wow() { }
    private void wowza() { }
}
```

A. It compiles without error.

B. It does not compile because an abstract class cannot have private methods.

C. It does not compile because an abstract class cannot have instance variables.

D. It does not compile because an abstract class must have at least one abstract method.

E. It does not compile because an abstract class must have a constructor with no arguments.

Explicación:

Respuesta: *A. It compiles without error.*

Una clase abtracta puede tener constructores, variables de instancia, métodos privados y no abstractos.

14. The SINGLETON pattern allows:

A. Have a single instance of a class and this instance cannot be used by other classes.

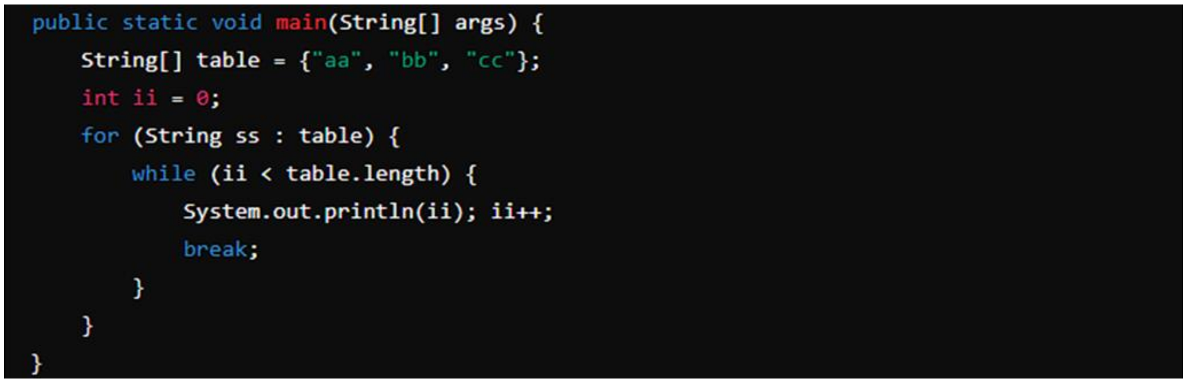
B. Having a single instance of a class, while allowing all classes have access to that instance.

C. Having a single instance of a class that can only be accessed by the first methods that calls it.

Explicación:

Respuesta: *B. Having a single instance of a class, while allowing all classes have access to that instance.*

15. How many times is 2 printed?



```
public static void main(String[] args) {
    String[] table = {"aa", "bb", "cc"};
    int ii = 0;
    for (String ss : table) {
        while (ii < table.length) {
            System.out.println(ii); ii++;
            break;
        }
    }
}
```

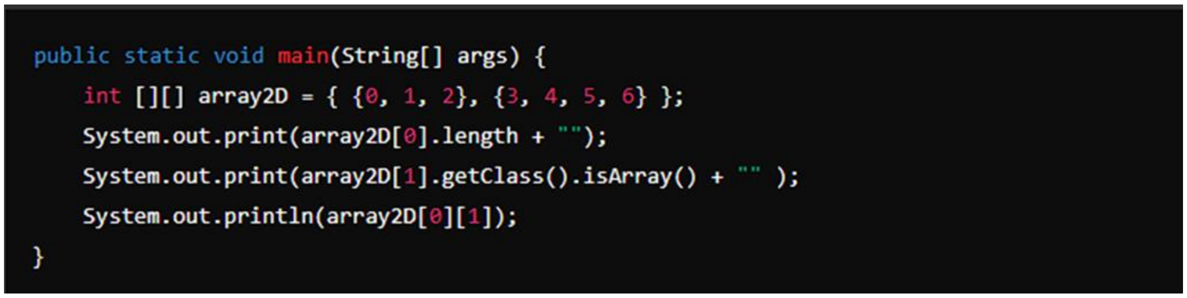
A. Zero.
[✓ B. Once.]
C. Twice.
D. Thrice.
E. It is not printed because compilation fails.

Explicación:

Respuesta: *B. Once.*

Dentro del foreach tenemos un ciclo que implementa un *break* sin condición por lo que el ciclo while no continuará después de imprimir e incrementar el valor de “ii”, el valor de “ii” se incrementará hasta 2, debido a la expresión booleana dada en el ciclo. Así que sólo una vez tendrá el valor de 2 y solo se podrá imprimir una vez.

16.What is the result?



```
public static void main(String[] args) {
    int [][] array2D = { {0, 1, 2}, {3, 4, 5, 6} };
    System.out.print(array2D[0].length + "");
    System.out.print(array2D[1].getClass().isArray() + "" );
    System.out.println(array2D[0][1]);
}
```

A. 3false1
B. 2true3
C. 2false3
[✓ D. 3true1]
E. 3false3
F. 2true1
G. 2false1

Explicación:

Respuesta: *D. 3true1*

En la primera impresión se obtiene el tamaño del primer arreglo en array2D que es de 3, la segunda impresión obtiene la clase a la que pertenece el segundo arreglo almacenado en array2D, y verifica si es de tipo Array, por lo que se obtiene true, y la última impresión imprime el entero almacenado en el primer arreglo que se encuentra en el índice 1, y corresponde a 1.

17.- In Java the difference between throws and throw is:

A. Throws throws an exception and throw indicates the type of exception that the method.
B. Throws is used in methods and throw in constructors.
C. Throws indicates the type of exception that the method does not handle and throw an exception.

18.- What is the result?

```
class Person {
String name = "No name";
public Person (String nm) {name=nm}
}
class Employee extends Person {
        String empID = "0000";
        public Employee(String id) { empID " //18
}
}
public class EmployeeTest {
        public static void main(String[] args) {
                Employee e = new Employee("4321");
                System.out.println(e.empID);
        }
}
```

A. 4321.
B. 0000.
C. An exception is thrown at runtime.

D. Compilation fails because of an error in line 18.

19.- Which is true?

```
class Building {}
    public class Barn extends Building{
        public static void main(String[] args){
            Building build1 = new Building();
            Barn barn1 = new Barn();
            Barn barn2 = (Barn) build1; //10
            Object obj1 = (Object) build1; //11
            String str1 = (String) build1; //12
            Building build2 = (Building) barn1; //13
    }
}
```

A. If line 10 is removed, the compilation succeeds.
B. If line 11 is removed, the compilation succeeds.
C. If line 12 is removed, the compilation succeeds.
D. If line 13 is removed, the compilation succeeds.
E. More than one line must be removed for compilation to succeed.

20.- What is the result?

```
class Atom {
    Atom() {System.out.print("atom ");}
}
class Rock extends Atom {
    Rock(String type) {System.out.print(type);}
}
public class Mountain extends Rock {
    Mountain(){
        super("granite ");
        new Rock("granite ");
    }
    public static void main(String[] a) {new Mountain();}
}
```

A. Compilation fails.
B. Atom granite.

C. Granite granite.
D. Atom granite granite.
E. An exception is thrown at runtime.
F. Atom granite atom granite.

correcta: F. *atom granite atom granite.

Pregunta 21

Which statement is true?

A. 420 is the output.

B. An exception is thrown at runtime.

C. All constructors must be declared public.

D. Constructors CANNOT use the private modifier.

E. Constructors CANNOT use the protected modifier.

```
class ClassA {
        public int numberOfInstances;
        protected ClassA(int numberOfInstances) {
                this.numberOfInstances = numberOfInstances;
        }
}
public class ExtendedA extends ClassA {
        private ExtendedA(int numberOfInstances) {
                super(numberOfInstances);
        }
        public static void main(String[] args) {
                ExtendedA ext = new ExtendedA(420);
                System.out.print(ext.numberOfInstances);
        }
}
```

Pregunta 22

What is the result?

A. 4 Null.

B. Null 4.

C. An IllegalArgumentException is thrown at run time.

D. 4 An ArrayIndexOutOfBoundsException is thrown at run time.

```
public class Test {
        public static void main(String[] args) {
                int[][] array = { {0}, {0,1}, {0,2,4}, {0,3,6,9}, {0,4,8,12,16} };
        System.out.println(array[4][1]);
        System.out.println(array[1][4]);
        }
}
```

Pregunta 23

What is the result?

A. There is no output.

B. d is output.

C. A StringIndexOutOfBoundsException is thrown at runtime.

D. An ArrayIndexOutOfBoundsException is thrown at runtime.

E. A NullPointException is thrown at runtime.

F. A StringArrayIndexOutOfBoundsException is thrown at runtime.

```
public class X {
        public static void main(String[] args) {
                String theString = "Hello World";
                System.out.println(theString.charAt(11));
        }
}
```

Pregunta 24

What is the result?

A. The program prints 1 then 2 after 5 seconds.

B. The program prints: 1 thrown to main.

C. The program prints: 1 2 thrown to main.

D. The program prints: 1 then t1 waits for its notification.

```
public class Bees {
        public static void main(String[] args) {
                try {
                new Bees().go();
                }catch (Exception e) {
        System.out.println("thrown to main");
                }
        }

        synchronized void go() throws InterruptedException {
                Thread t1 = new Thread();
                t1.start();
                System.out.print("1 ");
                t1.wait(5000);
                System.out.print("2 ");
        }
}
```

Pregunta 25
¿Cuál será el resultado?

```
public class SampleClass {
        public static void main(String[] args) {
                SampleClass sc, scA, scB;
                sc = new SampleClass();
                scA = new SampleClassA();
                scB = new SampleClassB();
                System.out.println("Hash is: " + sc.getHash() +
                ", " + scA.getHash() + ", " + scB.getHash());
        }
        public int getHash() {
                return 111111;
        }
}
class SampleClassA extends SampleClass {
        public int getHash() {
                return 44444444;
        }
}
class SampleClassB extends SampleClass {
        public int getHash() {
                return 999999999;
        }
}
```

a) Compilation fails
b) An exception is thrown at runtime
c) There is no result because this is not correct way to determine the hash code
d) Hash is: 111111, 44444444, 999999999.

sc es una instancia de SampleClass, scA es una instancia de SampleClassA y scB es una instancia de SampleClassB. Tanto scA y scB están haciendo un Override al método getHash();. El método main imprimirá los valores de las instancias, dando como resultado sc 111111, scA 44444444 y scB 999999999.

Pregunta 26

¿Cuál sería el resultado?

```
public class Test {
                public static void main(String[] args) {
                                int b = 4;
                                b–;
                                System.out.println(--b);
                                System.out.println(b);
                }
}
```

a) 2 2 b se inicializa en 4, posterior se convierte el 3 por el decremento, posteriormente se le aplica un predecremento lo que le da un valor de 2 e imprime el valor de b con el predecremento. En la última línea se pide imprimir el valor de b, el cual ahora es un 2.

b) 1 2
c) 3 2
d) 3 3

Pregunta 27. ¿Cuál sería el resultado?

```
import java.util.*;
public class App {
	public static void main(String[] args) {
		List p = new ArrayList();
		p.add(7);
		p.add(1);
		p.add(5);
		p.add(1);
		p.remove(1);
		System.out.println(p);
	}
}
```

a) [7, 1, 5, 1]
b) [7, 5, 1]
Dentro del código se añaden 7, 1, 5 y 1 con el p.add. Posterior a esto, con p.remove se quita el elemento del índice 1, por lo que el primer 1 se elimina, dejando así 7, 5, 1.
c) [7, 5]
d) [7, 1]

Pregunta 28. ¿Cuál sería el resultado?

```
public classDoCompare4 {
	public static void main(String[] args) {
		String[] table = {“aa”, “bb”, “cc”};
		int ii = 0;
		do {
			while (ii < table.length) {
				System.out.println(ii++);
			}
		} while (ii < table.length);
	}
}
```

a) 0
b) 0 1 2
   inicia en 0, se incrementa en 1 por lo que ya es 1 que sigue siendo menos que la longitud, se incrementa en 1 la i y ahora 2, por lo que sigue siendo menor a la longitud, por lo que imprime 012

c) 0 1 2 0 1 2 0 1 2
d) Compilation fails

Pregunta 29. ¿Cuál sería el resultado?

Parecido al ejercicio 15. con respuestas iguales, resultados diferentes.

```
public class DoCompare1 {

    public static void main(String[] args) {

        String[] table = {"aa", "bb", "cc"};

        for (String ss : table) {

            int ii = 0;9

            while (ii < table.length) {

                System.out.println(ss + ", " + ii);

                ii++;

            }

        }

    }

}
```

A) Zero.
B) Once.
C) Twince
D) Thrice
E) Compilation fails

El resultado final de las impresiones será:aa, 2 -bb, 2 - cc, 2 = 3 veces sale el 2.

aa, 0

aa, 1

aa, 2

bb, 0

bb, 1

bb, 2

cc, 0

cc, 1

cc, 2

Es ver cada tarjeta una por una y escribir tres líneas para cada tarjeta, cada línea con la palabra en la tarjeta y un número del 0 al 2.

30. What is the result?

```
public class Boxer1 {
    Integer i = 0; // Inicializar i con 0
    int x;

    public Boxer1(int y) {
        x = i + y;
        System.out.println(x);
    }
    public static void main(String[] args) {
        new Boxer1(new Integer(4));
    }
}
```

A. The value “4” is printed at the command line.

B. Compilation fails because of an error in line 5.

C. Compilation fails because of an error in line 9.

D. A NullPointerException occurs at runtime.

E. A NumberFormatException occurs at runtime.

F. An IllegalStateException occurs at runtime.

Solución

Para evitar el NullPointerException, debemos asegurarnos de que i esté inicializada antes de usarla en la operación. Aquí hay una versión corregida del código:

Ahora, cuando ejecutemos el código, y estará inicializada a 0, por lo que la operación x = i + y será x = 0 + 4, y se imprimirá 4.

31. What is the result?

A. Class Base1 { abstract class Abs1 { } }

- Esto es legal. Es posible tener una clase abstracta dentro de otra clase.

B. Abstract class Abs2 { void doit() { } }

- Esto es legal. Una clase abstracta puede tener métodos concretos.

C. class Base2 { abstract class Abs3 extends Base2 { } }

- Esto es legal. Es posible tener una clase abstracta que extiende otra clase dentro de una clase.

D. class Base3 { abstract int var1 = 89; }

- Esto es ilegal. Las variables no pueden ser abstractas. La palabra clave abstract solo se aplica a métodos y clases.

Por lo tanto, el fragmento de código ilegal es el D.

Pregunta 32. ¿Cuál sería el resultado?

```
public class ScopeTest {
   int z;
```

```
    public static void main(String[] args) {
        ScopeTest myScope = new ScopeTest();
        int z = 6;
        System.out.print(z);
        myScope.doStuff();
        System.out.print(z);
        System.out.print(myScope.z);
    }

    void doStuff() {
        int z = 5;
        doStuff2();
        System.out.print(z);
    }

    void doStuff2() {
        z = 4;
    }
}
```

A. 6564
B. 6554
C. 6566
D. 6565

Para confirmar nuestra comprensión:

1. System.out.print(z); en main imprime 6.
2. myScope.doStuff(); llama a doStuff:
    - doStuff declara int z = 5;.
    - doStuff2 establece myScope.z = 4;.

    - System.out.print(z); en doStuff imprime 5.
3. System.out.print(z); en main después de doStuff imprime 6.
4. System.out.print(myScope.z); imprime 4.

Entonces, el orden de las impresiones y sus valores será 6564.

33.- What is the result?

class Foo {

public int a = 3;

public void addFive() { a += 5; System.out.print("f"); }

}

class Bar extends Foo {

public int a = 8;

public void addFive() { this.a += 5; System.out.print("b"); })

Invoked with:

Foo f = new Bar(): f.addFive(): System.out.println(f.a);

A. b 3.

B. b 8.

C. b 13.

D. f3.

E. f 8.

F. f 13.

G. Compilation fails.

H. An exception is thrown at runtime.

}

34.- Which one will compile, and can be run successfully using the following

command? java Fred1 hello walls

A. class Fred 1{ public static void main(String args) { System.out.println(args[1]); } }

B. abstract class Fred1 { public static void main(String[] args) {
System.out.println(args[2]); } }

C. class Fred1 { public static void main(String[] args) { System.out.println(args); } }

D. class Fred1 { public static void main(String[] args) { System.out.println(args[1]); }
}

35.- What is printed out when the program is excuted?

public class MainMethod (void main() {

System.out.println("one");

System.out.println("two");

static void main(String args) {

}

}

public static final void main(String[] args) {

System.out.println("three");

void mina(Object[] args) {

System.out.println("four");

}

}

A. - one.

B. - two.

C. - three.

D. - four.

E. - There is no output.

37.- Which five methods, inserted independently at line 5, will compile? (Choose five.)

```
1. public class Blip {
2.  protected int blipvert(int x) { return 0; }
3. }
4. class Vert extends Blip {
5.// insert code here
6. }
```

A. Public int blipvert(int x) { return 0; }.

B. Private int blipvert(int x) { return 0; }.

C. Private int blipvert(long x) { return 0; }.

D. Protected long blipvert(int x) { return 0; }.

E. Protected int blipvert(long x) { return 0; }.

F. Protected long blipvert(long x) { return 0; }.

G. Protected long blipvert(int x, int y) { return 0; }.

38.- What values of x, y, z will produce the following result?

```
1234

1234

1234

--------

1234

-------
```

```
public static void main(String[] args) { // insert code here int j = 0, k = 0;
for (int i = 0; i < x; i++){ do {
k = 0;
while (k <z) {
}
k++;
System.out.print(k + " ");
System.out.println(" ");
} while (j < y):
System.out.println("--");
}
}
```

A. - int x=4, y=3, z=2;

B. int x=3, y=2, z=3;

C. - int x=2, y = 3, z=3;

D. - int x = 4, y = 2, 2=3;

E. - int x = 2, y = 3, z = 4;

39.- What is the result if the integer value is 33?

```
public static void main(String[] args) { if (value >= 0) {
if (value != 0) {
System.out.print("the");
} else {
System.out.print("quick");
}
if (value < 10) {
System.out.print("brown");
}
if (value>30) {
System.out.print("fox");
} else if (value <50) {
System.out.print("jumps");
} else if (value < 10){
System.out.print("over");
} else {
System.out.print("the");
}
if (value > 10) {
System.out.print("lazy");
} else {
System.out.print("dog");
}
System.out.print("... ");
}
}
```

A. The fox jump lazy?

B. The fox lazy?

C. Quick fox over lazy?

D. Quick fox the ?

40.- Which two declarations will compile?

```
14. public static void main(String[] args) {
15. int a, b, c = 0;
16. int a, b, c;
17. int g, int h, int i = 0;
18. int d, e, f;
19. int k, l, m, = 0;
20.}
```

A. Line 15.

B. Line 16.

C. Line 17.

D. Line 18.

E. Line 19.

F. Line 20.

41. What code should be inserted?

```
4. public class Bark {
5. // Insert code here - Line 5
6. public abstract void bark();
7.}
8.
9. // Insert code here – Line 9
10.  public abstract void bark(){
11. System.out.println("woof");
12.}
13.}
14. }
```

A. 5. class Dog {9. public class Poodle extends Dog {

B. 5. abstract Dog {9. public class poodle extends Dog {

C. 5. abstract class Dog {9. public class Poodle extends Dog {

D. 5. abstract Dog {9. public class Poodle implements Dog {

E. 5. abstract Dog {9. public class Poodle implements Dog {

F. 5. abstract class Dog {9. public class Poodle implements Dog {

42.- Which statement initializes a stringBuilder to a capacity of 128?

A. StringBuilder sb = new String("128");

B. StringBuilder sb = StringBuilder.setCapacity(128); C.

C. StringBuilder sb = StringBuilder.getInstance(128); D.

D. StringBuilder rsb = new StringBuilder(128);

43.- What is the result?

```
class MyKeys {
Integer key.
MyKeys(Integer k) { key = k; } public boolean equals(Object o) {
return ((MyKeys) o).key == this.key.
}
}
```

And this code snippet:

```
Map m = new HashMap(); MyKeys m1 = new MyKeys(1); MyKeys m2 = new MyKeys(2);
MyKeys m3 = new MyKeys(1);
MyKeys m4 = new MyKeys(new Integer(2));
m.put(m1, "car");
m.put(m2, "boat");
m.put(m3, "plane");
m.put(m4, "bus"); System.out.print(m.size());
```

A. 2

B. 3

C. 4

D. Compilation fails.

44.- What changes will make this code compile?

```
class X {
X(){}
private void one(){}
}
public class Y extends X{
Y(){}
private void two(){
one():
}
public static void main(String[] args) {
new Y().two():
}
}
```

A. Adding the public modifier to the declaration of class X.

B. Adding the protected modifier to the X() constructor.

C. Changing the private modifier on the declarationof the one() method to protected.

D. Removing the Y() constructor.

E. Removing the private modifier from the two() method.

Respuesta correcta la C.

45.- What is the best way to test that the values of h1 and h2 are the same?

public static void main(String[] args) {

String h1 = "Bob";

String h2= new String("Bob");

}

A. if (h1 == h2).

B. if (h1.equals(h2)).

C. if (h1.toString() == h2.toString()).

D. if (h1.same(h2)).

46.- Which three are valid? (Choose three.)

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();

A. p0 = p1;

B. p1 = p2;

C. p2 = p4;

D. p2 = (ClassC)p1;

E. p1 = (ClassB)p3;

F. p2 = (ClassC)p4;

47.- What is the result?

```
public static void main(String[] args) {
String color = "Red";
switch (color) {
case "Red":
System.out.println("Found Red");
case "Blue":
System.out.println("Found Blue");
case "White":
System.out.println("Found White");
break;
Default:
System.out.println("Found Default");
}
```

A. Found Red.

B. Found Red Found Blue.

C. Found Red Found Blue Found White.

D. Found Red Found Blue Found White Found Default.

48.- What is the result?

```
class MySort implements Comparator<integer> {

public int compare(Integer x, Integer y) {

return y.compareTo(x);

}

}
```

And the code fragment:

```
Integer[] primes = {2, 7, 5, 3);

MySort ms = new MySort();

Arrays.sort(primes, ms):

for (Integer p2: primes) {

System.out.print(p2 + " ");

}
```

A. 2357

B. 2753

[✓ C. 7532]

D. Compilation fails.

49-What is the result?

```
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();  // Esto imprimirá 20
    }
}
```

a-20
b-100
c-1000
d-2

50-¿Qué tres modificaciones, hechas de manera independiente, permitirán que la clase Greet compile y se ejecute?

```
package handy.dandy;
public class KeyStroke {
    public void typeExclamation() {
        System.out.println("!");
    }
}

package handy;
public class Greet {
    public static void main(String[] args) {
        String greeting = "Hello";
        System.out.print(greeting);
        KeyStroke stroke = new KeyStroke();
        stroke.typeExclamation();
    }
}
```

A. Line 8 replaced with handy.dandy.KeyStroke stroke = new KeyStroke();
B. Line 8 replaced with handy.*.KeyStroke stroke = new KeyStroke();

C. Line 8 replaced with handy.dandy.KeyStroke stroke = new handy.dandy.KeyStroke();
D. import handy.*; added before line 1.
E. import handy.dandy.*; added after line 1.
F. import handy.dandy.KeyStroke; added after line 1.
G. import handy.dandy.KeyStroke.typeExclamation(); added after line 1.

52-What is the result?

```
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();
    }
}
```

A. Cougar c f.
B. Feline cougar c c.
C. Feline cougar c f.
D. Compilation fails.

53-What is the result?

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

public class Camel implements Rideable {
    int weight = 2;
    String getGait() {
        return mph + ", lope"; // Error de compilación: mph no está
definido
    }

    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);
    }
}
```

A. 16 mph, lope.
B. 24 mph, lope.
C. 27 mph, lope.
D. Compilation fails.

54-¿Cuáles de las siguientes opciones son instanciaciones e inicializaciones válidas de un arreglo multidimensional?

```
a-int[][] array2D = {{0, 1, 2, 4}{5, 6}};
  int[][] array2D = new int[2][2];

b- 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;
```