

**Nombre:** Juan Ivan Velazquez Cabello

**Semana:** Tarea semana 3

Preguntas APX 02-Septiembre-2024.....	2
Preguntas Java.....	5
Java Cuestionario 3.....	21
Java Cuestionario 4.....	36

# Preguntas APX 02-Septiembre-2024

1. Las dependencias circulares son permitidas
  - Están prohibidas
2. Son dos de los protocolos físicos que APX admite en una solicitud.
  - JMS y REST
3. La invocación síncrona no está permitida. En caso de que una transacción debe invocar a otra, solo se puede realizar de forma Asincrónica.
  - Verdadero
4. ¿Cuántas veces es posible invocar Mainframe?
  - 2
5. Una Transacción o librería, dentro de su lógica, debe evitar invocar más de \_\_\_\_\_ librerías
  - 9
6. No Se debe manejar el objeto \_\_\_\_\_ directamente
  - Datasource
7. En un Batch, ¿Cuál es el step donde el desarrollador introducirá la lógica empresarial necesaria?
  - TASK
8. Implementación de PowerCurve en APX? Verdadero o falso?
  - Verdadero
9. Los componentes de seguridad lógica que dan servicio a la arquitectura no están acoplados a la Arquitectura APX.
  - Verdadero
10. DEBUG, sirve para información de muy bajo nivel solo útil para el debug de la aplicación.
  - Ciento
11. No es una característica de the blob
  - Manipulación directa en texto legible, Ejecución directa de lógica de negocio. Por lo tanto, una afirmación como "The Blob permite ejecución

**directa de lógica de negocio" sería falsa, ya que no es parte de sus funcionalidades.**

12. En el caso de Oracle el acceso a los datos es a través de:

- **JDBC**
- ORM
- JPA
- SQL

13. Componente que se encarga de encapsular toda la lógica empresarial y el acceso a datos.

- **Librería**

14. La creación de hilos o su gestión por las aplicaciones está permitida.

- **Falso**

15. El consumo de transacción asincronica pierde el control de la ejecución y solo se puede utilizar cuando la operación realizada de forma asíncrona \_\_\_\_\_ para el proceso de negocio, ya que el consumidor de esa transacción no sabrá si ha concluido con un commt OK o con rollback

- **No es crucial**

16. Son dos de las características ofrece la Arquitectura Batch para ampliar la funcionalidad que ofrece el framework.

- **Soporte Multi-BBDD, Reading/Writing de multiples formatos de archivo-acceso a conectores y utilidades**

17. ¿Quién es el responsable de iniciar un Job?

- A. JobExecution
- B. **JobLauncher**
- C. JobInstance
- D. JobBegginer

18. Los archivos objeto del procesamiento Batch de APX son de naturaleza estrictamente \_\_\_\_\_, y este archivo no es, en ningún caso, un mecanismo de persistencia de información a largo plazo.

- **Temporal**

19. ¿Cuál es la ubicación de archivos temporales utilizados en Batch?

- A. /fichetemporal/datent

- B. **/fichitemcomp/datent**
  - C. /temporalFiles/datent
  - D. /datentJobs/datent
- 20.** ¿Cómo solucionamos un Blob?
- A. Mover la funcionalidad a la transacción reduciendo la carga en la librería.
  - B. **Mover el comportamiento a sus librerías relacionadas, reduciendo el acoplamiento entre librerías y simplificando los mantenimientos.**
  - C. Moviendo la lógica a diferentes métodos y clases dentro de la misma librería
  - D. Mover el comportamiento a DTOs relacionados, reduciendo el acoplamiento entre los componentes y simplificando los mantenimientos
- 21.** Es una contradicción al usar el patrón CRUD
- A. Mantener todas las operaciones sobre la entidad encapsuladas en la librería
  - B. Tener varias librerías parciales para una única entidad**
  - C. Centralizar en una librería de todas las operaciones básicas sobre una entidad
  - D. Tener una implementación ligera, evitando extender la lógica más allá del objetivo de la operación
- 22.** Nivel de log que muestra información superior que permite monitorear la ejecución normal.
- **Info**
- 23.** En una librería, ¿Cuál es el archivo en donde se define la lógica del negocio?
- A. UUAAR000Impl
  - B. UUAAR001-app-osgi.xml y UUAAR001-app.xml
  - C. UUAAR001-arq-osgi.xml y UUAAR001-arc.xml**
  - D. UUAAR000Abstract
- 24.** En primer lugar, APX recibe la solicitud en uno de los protocolos físicos que admite (HTTP, JMS, REST,). Luego, crea un identificador único para la \_\_\_\_\_ y valide si los encabezados de solicitud están bien formateados de acuerdo con el protocolo lógico definido por Arquitectura.
- A. Pila de ejecución
  - B. Solicitud
  - C. Transacción**
  - D. Ejecución

# Preguntas Java

1. The BUILDER pattern is used to: **Simplify the creation of complex objects.**

2. In Java the difference between throws and throw is:

**throws indicates the type of exception that the method does not handle and throw an exception.**

3. A method is declared to take three arguments. A program calls this method and passes only two arguments. What is the result?

**Compilation fails**

4. Which three implementations are valid?

```
interface SampleCloseable {  
    public void close() throws java.io.IOException;  
}
```

```
class Test implements SampleCloseable { public void close() throws java.io.IOException { // do  
something } }
```

```
class Test implements SampleCloseable { public void close() throws Exception { // do  
something } }
```

```
class Test implements SampleCloseable { public void close() throws FileNotFoundException {  
// do something } }
```

```
class Test extends SampleCloseable { public void close() throws java.io.IOException { // do  
something } }
```

```
class Test implements SampleCloseable { public void close() { // do something } }
```

Respuesta correcta

```
class Test implements SampleCloseable { public void close() throws java.io.IOException { // do  
something } }
```

```
class Test implements SampleCloseable { public void close() throws FileNotFoundException {  
// do something } }
```

```
class Test implements SampleCloseable { public void close() { // do something } }
```

5. Which three lines will compile and output "Right on!"?

```
13. public class Speak {  
14.     public static void main(String[] args) {  
15.         Speak speakIT = new Tell();  
16.         Tell tellIT = new Tell();  
17.         speakIT.tellITLikeltls();  
18.         (Truth) speakIT.tellITLikeltls();  
19.         ((Truth) speakIT).tellITLikeltls();  
20.         tellIT.tellITLikeltls();  
21.         (Truth) tellIT.tellITLikeltls();  
22.         ((Truth) tellIT).tellITLikeltls();  
23.     }  
24. }
```

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


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


```

Línea 19 es correcta, línea 20 y línea 22

6. What changes will make this code compile? Selecciona 2

```
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 declaration of the one() method to protected**  
D. Removing the Y() constructor

**7. ¿Qué imprime el siguiente código?**

```

public class Main {
    public static void main(String[] args){
        int x = 2;
        for(;x<5;){
            x=x+1;
            System.out.println(x);
        }
    }
}

```

- A. 3,4,5**  
B. 3,4  
C. 2,4,5  
D. 2,3,4

**8. ¿Cuál es el resultado?**

```

11
12  public class Test1 {
13      public static void main(String[] args) {
14          int[][] array = {{0},{0,1},{0,2,4},{0,3,6,9},{0,4,8,12,16}};
15          System.out.println(array[4][1]);
16      }
17 }

```

El resultado a esta pregunta es que da un error de ejecución porque en la línea 15 se trata de acceder a la posición 1 del array bidimensional junta a su posición 4 y este array en la posición 1 solo tiene 2 elementos y no 4 por lo que causará el error de Exception in thread "main"

java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds  
for length 2

## 9. ¿Qué va a generar este código?

```
11  public class Main1 {  
12      [-]     public static void main(String[] args) throws Exception{  
13          doSomething();  
14      }  
15  
16      [-]     private static void doSomething() throws Exception{  
17          System.out.println("Before if clause");  
18          [-]         if(Math.random() > 5){  
19              throw new Exception();  
20          }  
21          System.out.println("After if clause");  
22      }  
23  }
```

Cuando el **Math.random** en la línea 18 sea menor a 5 se imprimirá en el programa

Before if clause

After if clause

Cuando el **Math.random** en la línea 18 sea mayor a 5 se imprimirá lo siguiente

Before if clause

Exception in thread "main" java.lang.Exception

at simuladores.Main1.doSomething(Main1.java:19)

at simuladores.Main1.main(Main1.java:13)

Esto porque al generarse la excepción y no ser controlada por el try catch si no que se la aventamos a la JVM está detiene el progreso solo imprimiendo una línea, pero si nosotros controlamos la excepción con un try catch si imprimiría todo el código como en este caso modificándolo de esta manera. La única diferencia es que aquí sale el nombre de la excepción que le enviamos y el tipo de excepción que se está manejando porque lo enviamos a imprimir.

```
11  public class Main1 {
12      public static void main(String[] args) {
13          try{
14              doSomething();
15          }catch(Exception e){
16              System.out.println("Exception de tipo :"+e);
17          }
18      }
19
20
21      private static void doSomething() throws Exception{
22          System.out.println("Before if clause");
23          if(6 > 5){
24              throw new Exception();
25          }
26          System.out.println("After if clause");
27      }
28  }
```

The code above is a Java program named Main1. It contains a main method that prints "Before if clause", checks if 6 is greater than 5, and if so, throws a new Exception. It then prints "After if clause". The output window shows the execution of the program, including the compilation message, the execution command, the printed messages ("Before if clause" and "Exception de tipo :java.lang.Exception"), the build success message, and the total execution time.

Si imprimimos con e.printStackTrace(); tampoco se detendrá lo único que pasa es que muestra la información completa.

10. ¿Cual es la salida del siguiente código?

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int a = 2;  
4         if (a == 2)  
5             System.out.println("A");  
6         else System.out.println("B");  
7         else System.out.println("C");  
8     }  
9 }  
10 }
```

La respuesta es que no compilaría porque en la línea 8 hay un error, tiene 2 else seguidos y esto no se puede tener un if sin else

11. How many times is 2 printed?

```
11 public class Menu {  
12     public static void main(String[] args) {  
13         String[] breakfast = {"beans", "egg", "ham", "juice"};  
14         for (String rs : breakfast) {  
15             int dish = 1;  
16             while (dish < breakfast.length) {  
17                 System.out.println(rs + ", " + dish);  
18                 ++dish;  
19             }  
20         }  
21     }  
22 }
```

Se imprime un total de 4 veces:

Salida: beans,1 beans,2 beans,3 egg,1 egg,2 egg,3 ham,1 ham,2 ham,3 juice,1 juice,2 juice,3

12. What is the result?

```
1 class Person{  
2     String name = "No name";  
3     public Person(String nm){  
4         name = nm;  
5     }  
6 }  
7  
8 class Employee extends Person{  
9     String empID = "0000";  
10    public Employee(String id) {  
11        }  
12    }  
13  
14 public class EmpleeTest {  
15     public static void main(String[] args){  
16         Employee e = new Employee("4321");  
17         System.out.println(e.empID);  
18     }  
19 }
```

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

**D. Compilation fails because of an error in line 10.**

(Falla porque se está tratando de mandar a llamar al constructor por default de la clase padre pero como se definió uno ya no se crea uno por defecto sin argumentos por lo que implícitamente en la línea 11 hay un super() mandando a llamar a ese constructor vacío pero en este caso no hay provocando el error)

**13. What is the result?**

```
11  class Atom {  
12  
13      [-]     Atom() {  
14          System.out.print("atom ");  
15      }  
16  }  
17  
18  class Rock extends Atom {  
19      [-]     Rock(String type) {  
20          System.out.print(type);  
21      }  
22  }  
23  
24  public class Mountain extends Rock {  
25      [-]     Mountain() {  
26          super("granite ");  
27          new Rock("granite ");  
28      }  
29  
30      [-]     public static void main(String[] a) {  
31          new Mountain();  
32      }  
33  } //atom granite atom granite  
34
```

- 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.**

**14.** What is the result?

```
import java.text.*;
public class Align {
    public static void main(String[] args) throws ParseException {
        String[] sa = {"111.234", "222.5678"};
        NumberFormat nf = NumberFormat.getInstance();
        nf.setMaximumFractionDigits(3); //NO HACE NADA
        for (String s : sa) {
            System.out.println(nf.parse(s));
        }
    }
}
```

- A. 111.234 222.567
- B. 111.234 222.568
- C. **111.234 222.5678**
- D. An exception is thrown at runtime

15. What is the result?

```
1 interface Rideable {
2     String getTicket();
3 }
4
5 public class Camel implements Rideable {
6     int weight = 2;
7     String getGait() {
8         return mph + ", lope";
9     }
10
11    void go(int speed) {
12        ++speed;
13        weight++;
14        int walkrate = speed * weight;
15        System.out.print(walkrate + getGait());
16    }
17
18    public static void main(String[] args) {
19        new Camel().go(8);
20    }
21 }
```

- A. No compila porque mph no existe
- B. No compila porque mph no existe y no se implementa el método getTicket() de la interfaz**
- C. 27 mph , lope
- D. 27, lope

16. Which two actions, used independently, will permit this class to compile?

```
1 public class Y {  
2     public static void main(String[] args) {  
3         try {  
4             doSomething();  
5         } catch (RuntimeException e) {  
6             System.out.println(e);  
7         }  
8     }  
9  
10    static void doSomething() {  
11        if (Math.random() > 0.5) {  
12            throw new IOException();  
13        }  
14        throw new RuntimeException();  
15    }  
16}
```

- A. Adding throws IOException to the main() method signature and to the doSomething() method.
- B. Adding throws IOException to the main() method ... IOException.
- C. Adding throws IOException to the doSomething() method signature.**
- D. Adding throws IOException to the main() method signature.
- E. Adding throws IOException to the doSomething() method signature and changing the catch.**

**17. ¿Cuál es el resultado?**

What is the result? \*

1 punto

```
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) { }  
  
 If conn.rollback(s1) is inserted, account will be incremented by 10.  
 If conn.rollback(s1) is inserted, account will be incremented by 1010.  
 If conn.rollback(s2) is inserted, account will be incremented by 100  
 If conn.rollback(s2) is inserted, account will be incremented by 110.  
 If conn.rollback(s2) is inserted, account will be incremented by 1110
```

**by 10**

**by 110**

**18. El uso de LocalDateTime y Period es para manejar fechas y periodos.**

- A. **Cierto**
- B. Falso

**19. ¿Cuáles de las siguientes opciones son válidas?**

- A. El constructor predeterminado proporcionado por el compilador puede ser llamado utilizando this().
- B. Un constructor puede ser invocado desde un método de instancia.**
- C. Una variable de instancia puede ser accedida dentro de un método de clase (static).
- D. Un constructor puede ser llamado dentro de otro constructor utilizando la palabra clave this().**

**20.** ¿Cuál es el resultado?

```
class X {  
    static void m(int i) {  
        i += 7;  
    }  
    public static void main(String[] args) {  
        int i = 12;  
        m(i);  
        System.out.println(i);  
    }  
}
```

- A. 7
- B. 12**
- C. 19
- D. La compilación falla
- E. Una excepción ocurre en tiempo de ejecución

21. ¿Cuál es el resultado si value es 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("... ");
}
```

- The fox jump lazy ?
- The fox lazy ?
- Quick fox over lazy ?

**22.** ¿Cuál es el resultado?

```
public class Truthy{  
    public static void main(String[ ] args){  
        int x = 7;  
        assert (x == 6) ? "x == 6" : "x != 6";  
    }  
}
```

- A. Truthy.java compiles and the output is x != 6
- B. Truthy.java compiles and an AssertionError is thrown with x != 6 as additional output.
- C. Truthy.java does NOT compile.**
- D. Truthy.java compiles and an AssertionError is thrown with no additional output

**23.** Las utilidades APX

Las utilidades APX implementadas tanto para la arquitectura Online como \* 1 punto para la arquitectura Batch son:

- Communication Manager: Se crea una librería con la capacidad de invocar a G.U.C para el envío de notificaciones. Interbackend proxy: Se crea una librería con la capacidad de invocar a los conectores IMS (para acceso a Host).
- Compresión/Descompresión: Se crea una librería con la capacidad de comprimir y descomprimir archivos en zip. Generador de documentos: Se crea una librería con la capacidad de generar documentos.

#### 24. ¿Qué es APX online?

¿Qué es APX Online? \*

1 punt

- Basada en Java, diseñada y construida por BBVA con ámbito local para ser utilizada en el Banco.
- Esta plataforma contiene capacidades Front y está dirigida a aplicaciones que necesiten crear lógica de negocio sin acoplamiento con la presentación. Es un Sistema de alto rendimiento, con fácil escalabilidad y disponibilidad garantizada
- Ejecución en entornos normales tipo R2 es la misma que la arquitectura diseñada y planteada para entornos Cloud. Las versiones que se generan son no duales por lo que cambia el empaquetado de la Arquitectura y la estrategia de aprovisionamiento de componentes aplicativos.
- Habilita los servicios bancarios transaccionales fuera del entorno de mainframe, pero brinda las mismas capacidades en el mundo distribuido, reduce los costos generales del sistema, se basa en tecnologías de código abierto, permite a los desarrolladores aprovechar todos los servicios extendidos provistos en la plataforma para construir servicios lo más rápido posible

#### 25. Definición de Singleton

**The SINGLETON pattern allows: Having a single instance of a class, while allowing all classes have access to that instance**

#### 26. Diferencias entre clases abstractas e interfaces

- A. Si una clase puede heredar de una interfaz (error)
- B. Si una clase puede heredar de multiples clases
- C. Si en una interfaz puede haber métodos con comportamiento y default**

# Java Cuestionario 3

1. ¿Cuál es el resultado?

```
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 head){
        System.out.println(head.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 ");
    }
}
```

```
//RESULTADO: side, object, tail, side
--- exec:3.1.0:exec (default-cli) @ Ejercicios ---
Side
Object
Tail
Side
```

2. ¿Cuál es el resultado para que imprima primero? Shape: constructor, shape: foo, Square: foo

```
public class SuperTest {
    public static void main(String[] args){
        //1 Square square = new Square(); // Solución
        //2 one.foo(); // Solución
        //3 one.foo("Hola"); // Solución
    }
}

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");
    }
}
```

- A. Square square = new Square(); square.foo(); square.foo("Hola");
- B. Square square = new Square(); square.foo(); square.foo();
- C. Square square = new Square(); square.foo(); square.foo(bar);
- D. Square square = new Square("bar"); square.foo("bar");
square.foo("bar");

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

A. int[1] array2D = ({0, 1, 2, 47(5, 6)}:

B. int[1] array2D = new int[1][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[][][] arrax3D = {{{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;

4. ¿Cuál es el resultado?

```
/*
public class Calculator {
    int num = 100;
    public void calc(int num){
        this.num = num * 20;
    }
    public void printNum(){
        System.out.println(num);
    }
    public static void main(String[] args){
        Calculator cal = new Calculator();
        Calculator cal2 = new Calculator();
        cal.calc(10);
        cal.printNum();
        cal2.printNum();
    }
} // RESULTADO 200 y 100
```

5. ¿Cuál es el resultado?

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

public class Cougar extends Feline {
    public Cougar(){
        System.out.println("cougar");
    }
    void go(){
        type = "c";
        System.out.println(this.type + super.type);
    }
    public static void main(String[] args){
        new Cougar().go();
    }
}//RESULTADO: "f","cougar","c","c"
```

6. ¿Cuál es el resultado?

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. X
- Compilation fails

Respuesta correcta

- Compilation fails

7. ¿Cuáles 3 opciones son válidas?

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; ✓

8. ¿Cuál es el resultado?

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

- A. Z only.
- B. X only.**
- C. X, Y and Z
- D. X and Y
- E. X and Z
- F. Y only
- G. Y and Z

**9. 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<>>();

**10. ¿Cuál es el resultado?**

```
int i = 1;
int j = i++;
if((i==++j) | (i++ ==j)) {
    i+=j;
}
System.out.println(i);
//SALIDA: 5
```

## 11. ¿Cuál es el resultado?

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

---

**X 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;  
}
```

12. ¿Cuál es el resultado?

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

13. ¿Cuál es el resultado?

```
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)) //No se cumple la condicion : 9
== 8
    c+=2;
System.out.println(c);
//SALIDA: a
```

14. ¿Cuál es el resultado?

```
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");
            }
        }
    }
}// RESULTADO: NADA ya que nunca entra al for por i>5 0>5
```

**15. ¿Cuál es el resultado?**

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

---

**X 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.
-

## 16. ¿Cuál es el resultado?

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{
```

**Han sido creado 4 objetos**

## 17. ¿Cuál es el resultado?

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

    public static void main(String[] args){
        System.out.println(f);
        System.out.println("");
        System.out.println(ch);
        System.out.println("");
        System.out.println(bool);
    }
}
//RESULTADO: 0.0  false
```

## 18. ¿Cuál es el resultado?

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(); }
}
```

**X You answered incorrectly**  
You had to select 3 options

c = c;

It will reassing '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;

19. ¿Cuál es el resultado?

```
public class TestClass1 {  
    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);  
    }  
}  
//RESULTADO: 0
```

20. Which of the following options will compile?

```
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> all = new ArrayList<>();  
SUV s = all.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);`

**21.**

```
1 public class Speak {  
2     public static void main(String[] args) {  
3         Speak speakIT = new Tell();  
4         Tell tellIT = new Tell();  
5         speakIT.tellItLikeItIs();  
6         (Truth)speakIT.tellItLikeItIs();  
7         ((Truth)speakIT).tellItLikeItIs();  
8         tellIT.tellItLikeItIs();  
9         (Truth)tellIT.tellItLikeItIs();  
10        ((Truth)tellIT).tellItLikeItIs();  
11    }  
12 }  
13  
14 class Tell extends Speak implements Truth{  
15     @Override  
16     public void tellItLikeItIs() {  
17         System.out.println("Right on");  
18     }  
19 }  
20  
21 interface Truth{  
22     public void tellItLikeItIs();  
23 }
```

//RESULTADO: No compila error en linea 6 y 9

# Java Cuestionario 4

## 1. ¿Cuál es el resultado?

```
public class Main {  
    public static void main(String[] args) {  
        String[] at = {"FINN", "JAKE"};  
        for (int x = 1; x < 4; x++) {  
            for (String s : at) {  
                System.out.println(x + " " + s);  
                if (x == 1) {  
                    break;  
                }  
            }  
        }  
    }  
}  
//1 FINN 2 FINN 2 JAKE 3 FINN 3 JAKE
```

## 2. ¿Qué 5 líneas son correctas?

```
class Saber extends Light {  
  
    private int lightsaber(int x) {  
        return 0;  
    } // Error el modificador de acceso en la clase derivada no puede  
    ser más restrictivo que el modificador de acceso en la clase base  
  
    protected int lightsaber(long x) {  
        return 0;  
    } // Correcto Sobreescritura de metodo adecuada, por cambio de  
    parametro  
  
    private int lightsaber(long x) {  
        return 0;  
    } // Correcto No se esta sobreescribiendo el metodo, al tener  
    otro parametro se trata de un metodo independiente  
  
    protected long lightsaber(int x) {
```

```

        return 0;
    } // Error Para que la sobrescritura sea válida, los métodos
    deben tener la misma firma, incluyendo el tipo de retorno

protected long lightsaber(int x, int y) {
    return 0;
} //Correcto

public int lightsaber(int x) {
    return 0;
} // Correcto

protected long lightsaber(long x) {
    return 0;
} // Valido por ser sobrecarga de metodo
}

```

### 3. ¿Qué resultado arroja?

```

class Light {
    protected int lightsaber(int x) {
        return 0;
    }
}

class Mouse{
    public int numTeeth;
    public int numWhiskers;
    public int weight;
    public Mouse (int weight){
        this(weight,16);
    }
    public Mouse (int weight, int numTeeth){
        this(weight, numTeeth, 6);
    }
    public Mouse (int weight, int numTeeth, int numWhiskers){
        this.weight = weight;
        this.numTeeth= numTeeth;
        this.numWhiskers = numWhiskers;
    }
    public void print (){
        System.out.println(weight + ""+ numTeeth+ ""+ numWhiskers);
    }
    public static void main (String [] args){
        Mouse mouse = new Mouse (15);
        mouse.print();
    }
}

```

```
        }
} // Salida: 15 , 16 , 6
```

**4. ¿Cuál es la salida?**

```
class Arachnid {
    public String type = "a";
    public Arachnid() {
        System.out.println("arachnid");
    }
}
class Spider extends Arachnid{
    public Spider(){
        System.out.println("spider");
    }
    void run(){
        type = "s";
        System.out.println(this.type + " " + super.type);
    }
    public static void main(String[] args) {
        new Spider().run();
    }
}
//Salida: arachnid spider s s
```

**5. ¿Cuál es la salida?**

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

    }
}

class Sheep {
    public static void main(String[] args) {
        int ov = 999;
        ov--;
        System.out.println(--ov);
        System.out.println(ov);
    }
}
// Respuesta correcta: 997, 997
```

**6. ¿Cuál es la salida?**

```

class Overloading {
    public static void main(String[] args) {
        System.out.println(overload("a"));
        System.out.println(overload("a", "b"));
        System.out.println(overload("a", "b", "c"));
    }
    public static String overload(String s){
        return "1";
    }
    public static String overload(String... s){
        return "2";
    }
    public static String overload(Object o){
        return "3";
    }
    public static String overload(String s, String t){
        return "4";
    }
}
// Salida: 1, 4, 2

```

7. ¿Cuál es la salida?

```

class Basel extends Base{
    public void test(){
        System.out.println("Basel");
    }
}
class Base2 extends Base{
    public void test(){
        System.out.println("Base2");
    }
}
class Test {
    public static void main(String[] args) {
        Base obj = new Basel();
        ((Base2) obj).test();
    }
}
// ClassCastException: se produce cuando se intenta realizar una
conversión de tipos entre clases no relacionadas en una jerarquía
de herencia

```

8. ¿Cuál es la salida?

```

public class Fish {
    public static void main(String[] args) {
        int numFish = 4;
    }
}

```

```

        String fishType= "Tuna";
        String anotherFish = numFish +1;
        System.out.println(anotherFish + " " + fishType);
        System.out.println(numFish + " " + 1);
    }
}

// El codigo no compila

```

**9.** ¿Cuál es la salida?

```

class MathFun {
    public static void main(String[] args) {
        int number1 = 0b0111;
        int number2 = 0111_000;
        System.out.println("Number1: "+number1);
        System.out.println("Number2: "+number1);
    }
}

//Salida: 7 7 ojo que imprime dos veces number 1

```

**10.** ¿Cuál es la salida?

```

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();
    }
}

// Salida: 20

```

**11.** ¿Qué Aseveraciones son correctas?

```

class ImportExample {
    public static void main (String [] args){
        Random r = new Random();
    }
}

```

```

        System.out.println(r.nextInt(10));
    }

}

```

- If you omit java.util import statements java compiles gives you an error
- java.lang and util.random are redundant
- you dont need to import java.lang

**12.** ¿Cuál es la salida?

```

public class Main {
    public static void main(String[] args) {
        int var = 10;
        System.out.println(var++);
        System.out.println(++var);
    }
} //salida: 10, 12

```

**13.** ¿Cuál es la salida?

```

class MyTime {
    public static void main (String [] args){
        short mn =11;
        short hr;
        short sg =0;
        for (hr=mn;hr>6;hr-=1){
            sg++;
        }
        System.out.println("sg="+sg);
    }
}
// Salida sg=5; Respuesta correcta mn = 11

```

**14.** ¿Cuáles son verdad?

- An **ArrayList** is **mutable**:
- An **Array** has a **fixed size**
- An **array** is **mutable**
- An **array** allows **multiple dimensions**
- An **arrayList** is **ordered**
- An **array** is **ordered**

**15.** ¿Cuál es la salida?

```

Public class MultiverseLoop {
    public static void main (String [] args){
        int negotiate = 9;

```

```

        do{
            System.out.println(negotiate);
        }while (--negotiate);
    }
} //Errores de compilacion, necesita un bool el while

```

**16.** ¿Cuál es la salida?

```

class App {
    public static void main(String[] args) {
        Stream<Integer> nums = Stream.of(1,2,3,4,5);
        nums.filter(n -> n % 2 == 1);
        nums.forEach(p -> System.out.println(p));
    }
} //Exception at runtime, se debe encadernar el stream por que se
consume

```

**17.** suppose the declared type of x is a class, and the declared type of y is an interface.  
When is the assignment x = y; legal?

- When the type of X is Object

**18.** when a byte is added to a char, what is the type of the result?

- int

**19.** the standart application programmming interface for accesing databases in java?

- JDBC segun CHATGPT

**20.** Which one of the following statements is true about using packages to organize your code in Java ?

- Packages allow you to limit access to classes, methods, or data from classes outside the package.

**21.** Forma correcta de inicializar un booleano

- boolean a = (3>6);

**22.** ¿Cuál es la salida?

```

class Y{
    public static void main(String[] args) throws IOException {
        try {

```

```

        doSomething();
    }catch (RuntimeException exception){
        System.out.println(exception);
    }
}
static void doSomething() throws IOException {
    if (Math.random() > 0.5){
    }
    throw new RuntimeException();
}
}
• Adding throws IOException to the main() method signature

```

**23.** ¿Cuál es la salida?

```

interface Interviewer {
    abstract int interviewConducted();
}

public class Manager implements Interviewer{
    int interviewConducted() {
        return 0;
    }
} //Wont compile

```

**24.** ¿Cuál es la salida?

```

class Arthropod {
    public void printName(double Input) {
        System.out.println("Arth");
    }
}

class Spider extends Arthropod {
    public void printName(int input) {
        System.out.println("Spider");
    }
    public static void main(String[] args) {
        Spider spider = new Spider();
        spider.printName(4);
        spider.printName(9.0);
    }
} // Spider, Arth

```

**25.** ¿Cuál es la salida?

```

public class Main {
    public enum Days{Mon,Tue, Wed}
    public static void main(String[] args) {

```

```

        for (Days d:Days.values())
            )
        Days[] d2 = Days.values();
        System.out.println(d2[2]);
    }
}
} // wed

```

**26.** ¿Cuál es la salida?

```

public class Main{
    public enum Days {MON, TUE, WED};
    public static void main(String[] args) {
        boolean x= true, z = true;
        int y = 20;
        x = (y!=10)^ (z=false);
        System.out.println(x + " " + y + " "+ z);

    }
} // true 20 false

```

**27.** ¿Cuál es la salida?

```

class InitializacionOrder {
    static {add(2);}
    static void add(int num){
        System.out.println(num+"");
    }
    InitializacionOrder(){add(5);}
    static {add(4);}
    {add(6);}
    static {new InitializacionOrder();}
    {add(8);}
    public static void main(String[] args) {}
} //2 4 6 8 5

```

**28.** ¿Cuál es la salida?

```

public class Main {
    public static void main(String[] args) {
        String message1 = "Wham bam";

```

```

        String message2 = new String("Wham bam");
        if (message1!=message2){
            System.out.println("They dont match");
        }else {
            System.out.println("They match");
        }
    }
}

// They dont match

```

29. ¿Cuál es la salida?

```

class Mouse{
    public String name;
    public void run(){
        System.out.println("1");
        try{
            System.out.println("2");
            name.toString();
            System.out.println("3");
        }catch(NullPointerException e){
            System.out.println("4");
            throw e;
        }
        System.out.println("5");
    }
    public static void main(String[] args) {
        Mouse jerry = new Mouse();
        jerry.run();
        System.out.println("6");
    }
} // Salida 1 2 4 NullPointerException

```

31. ¿Cuál es la salida?

```

public class Main {
    public static void main(String[] args) {
        try (Connection con = DriverManager.getConnection(url, uname,

```

```

        pwd) ) {
            Statement stmt =con.createStatement();
            System.out.print(stmt.executeUpdate("INSERT INTO User
VALUES (500, 'Ramesh')"));
        }
    }
// Salida: arroja 1

```

32. ¿Cuál es la salida?

```

class MarvelClass{
    public static void main (String [] args){
        MarvelClass ab1, ab2, ab3;
        ab1 =new MarvelClass();
        ab2 = new MarvelMovieA();
        ab3 = new MarvelMovieB();
        System.out.println ("the profits are " + ab1.getHash()+ "," +
ab2.getHash()+","+ab3.getHash());
    }
    public int getHash(){
        return 676000;
    }
}
class MarvelMovieA extends MarvelClass{
    public int getHash (){
        return 18330000;
    }
}
class MarvelMovieB extends MarvelClass {
    public int getHash(){
        return 27980000;
    }
}
// the profits are 676000, 18330000, 27980000

```

33. ¿Cuál es la salida?

```

class Song{
    public static void main (String [] args){
        String[] arr = {"DUHAST", "FEEL", "YELLOW", "FIX YOU"};

```

```

        for (int i =0; i <= arr.length; i++){
            System.out.println(arr[i]);
        }
    }
}

//4 An arrayindexoutofbondsexception

```

**34.** ¿Cuál es la salida?

```

class Menu {
    public static void main(String[] args) {
        String[] breakfast = {"beans", "egg", "ham", "juice"};
        for (String rs : breakfast) {
            int dish = 2;
            while (dish < breakfast.length) {
                System.out.println(rs + "," + dish);
                dish++;
            }
        }
    }
/*
beans,2
beans,3
egg,2
egg,3
ham,2
ham,3
juice,2
juice,3
* Respuesta correcta: ONCE */

```

**35.** Which of the following statement are true:

- string builder es generalmente más rápido que string buffer
- string buffer is threadsafe; stringbuilder is not

**36.** ¿Cuál es la salida?

```

class CustomKeys{
    Integer key;
    CustomKeys(Integer k){
        key = k;
    }
}

```

```

        }
    public boolean equals(Object o) {
        return ((CustomKeys)o).key==this.key;
    }
}
// Salida: compilation fail

```

37. The catch clause is of the type:

- Throwable
- **Exception but NOT including RuntimeException**
- CheckedException
- RunTimeException
- Error

38. an enhanced for loop

- also called for each, offers simple syntax to iterate through a collection but it can't be used to delete elements of a collection

39. which of the following methods may appear in class Y, which extends X ?

- public void doSomething(int a, int b){...}

40. ¿Cuál es la salida?

```

public class Main {
    public static void main(String[] args) {
        String s1= "Java";
        String s2 = "java";
        if (s1.equalsIgnoreCase(s2)) {
            System.out.println ("Equal");
        } else {
            System.out.println ("Not equal");
        }
    }
}
// Salida: Equal; respuesta: s1.equalsIgnoreCase(s2)

```

41. ¿Cuál es la salida?

```

class App {
    public static void main(String[] args) {
        String[] fruits = {"banana", "apple", "pears", "grapes"};
        // Ordenar el arreglo de frutas utilizando compareTo
    }
}

```

```

        Arrays.sort(fruits, (a, b) -> a.compareTo(b));
        // Imprimir el arreglo de frutas ordenado
        for (String s : fruits) {
            System.out.println(""+s);
        }
    }
}
/* apple
banana
grapes
pears */

```

42. ¿Cuál es la salida?

```

public class Main {
    public static void main(String[] args) {
        int[] countsofMoose = new int [3];
        System.out.println(countsofMoose[-1]);
    }
}
//this code will throw an arrayindexoutofboundsexpression

```

43. ¿Cuál es la salida?

```

class Salmon{
    int count;
    public void Salmon (){
        count =4;
    }
    public static void main(String[] args) {
        Salmon s = new Salmon();
        System.out.println(s.count);
    }
}
// Salida: 0 -> cero

```

44. ¿Cuál es la salida?

```

44 pregunta
class Circuit {
    public static void main(String[] args) {
        runlap();
    }
}

```

```

        int c1=c2;
        int c2 = v;
    }
    static void runlap(){
        System.out.println(v);
    }
    static int v;
}
// corregir linea 6; c1 se le asigna c2 pero c2 aun no se declara

```

**45.** ¿Cuál es la salida?

```

class Foo {
    public static void main(String[] args) {
        int a=10;
        long b=20;
        short c=30;
        System.out.println(++a + b++ *c);
    }
} // salida: 611 (11+20*30)

```

**46.** ¿Cuál es la salida?

```

public class Shop{
    public static void main(String[] args) {
        new Shop().go("welcome",1);
        new Shop().go("welcome", "to", 2);

    }
    public void go (String... y, int x){
        System.out.print(y[y.length-1]+"" );
    }
}
// Compilation fails

```

**47.** ¿Cuál es la salida?

```

class Plant {
    Plant() {
        System.out.println("plant");
    }
}

```

```

}

class Tree extends Plant {
    Tree(String type) {
        System.out.println(type);
    }
}

class Forest extends Tree {
    Forest() {
        super("leaves");
        new Tree("leaves");
    }

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

/*plant
leaves
plant
leaves*/

```

**48.** ¿Cuál es la salida?

```

class Test {
    public static void main(String[] args) {
        String s1 = "hello";
        String s2 = new String ("hello");
        s2=s2.intern(); // el intern() asigna el mismo hash conforme a
la cadena
        System.out.println(s1==s2);
    }
} // Salida: true

```

**49.** Cuál de las siguientes construcciones es un ciclo infinito while:

- while(true);
- while(1==1){}

**50.** ¿Cuál es la salida?

```

class SampleClass{
    public static void main(String[] args) {
        AnotherSampleClass asc =new AnotherSampleClass ();
        SampleClass sc = new SampleClass();
    }
}

```

```

    //sc = asc;
    //TODO CODE

}

}

class AnotherSampleClass extends SampleClass {}

// Respuesta: sc = asc;

```

**51.** ¿Cuál es la salida?

```

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

// Salida: 0 -> cero

```

**52.** ¿Cuál es la salida?

```

public class Main{
    public static void main(String[] args) {
        course c = new course();
        c.name="java";
        System.out.println(c.name);
    }
}

class course {
    String name;
    course(){
        course c = new course();
        c.name="Oracle";
    }
} // Exception StackOverflowError

```

**53.** ¿Cuál es la salida?

```

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

```

```
        System.out.println(a.toString());
    }
} // builder fails
```

54. ¿Cuál es la salida?

```
public class Main{
    public static void main(String[] args) {
        System.out.println(2+3+5);
        System.out.println("+"+2+3+5);
    }
} // salida 10 + 235
```

55. ¿Cuál es la salida?

```
public class Main {
    public static void main(String[] args) {
        int a = 2;
        int b = 2;
        if (a==b)
            System.out.println("Here1");
        if (a!=b)
            System.out.println("here2");

        if (a>=b)
            System.out.println("Here3");
    }
} // salida: Here1 , here 3
```

56. ¿Cuál es la salida?

```
public class Main extends count {
    public static void main(String[] args) {
        int a = 7;
        System.out.println(count(a,6));
    }
}
class count {
    int count(int x, int y){return x+y;}
}// builder fails
```

57. ¿Cuál es la salida?

```
class trips{
    void main(){
        System.out.println("Mountain");
    }
}
```

```
static void main (String args){  
    System.out.println("BEACH");  
}  
  
public static void main (String [] args){  
    System.out.println("magic town");  
}  
void mina(Object[] args){  
    System.out.println("city");  
}  
} // Salida: magic town
```

58. ¿Cuál es la salida?

```
public static void main(String[] args) {  
    int a=0;  
    System.out.println(a++ +2);  
    System.out.println(a);  
}  
} // salida: 2,1
```

59. ¿Cuál es la salida?

```
public class Main{  
    public static void main(String[] args) {  
        List<E> p =new ArrayList<>();  
        p.add(2);  
        p.add(1);  
        p.add(7);  
        p.add(4);  
    }  
} // builder fails
```

60. ¿Cuál es la salida?

```
public class Car{  
    private void accelerate(){  
        System.out.println("car acelerating");  
    }  
    private void break(){  
}
```

```

        System.out.println("car breaking");
    }
    public void control (boolean faster){
        if(faster==true)
            accelerate();
        else
            break();
    }
    public static void main (String [] args){
        Car car = new Car();
        car.control(false);
    }
} break es una palabra reservada

```

61. ¿Cuál es la salida?

```

App() {
    System.out.println("1");
}
App(Integer num) {
    System.out.println("3");
}
App(Object num) {
    System.out.println("4");
}
App(int num1, int num2, int num3) {
    System.out.println("5");
}
public static void main(String[] args) {
    new App(100);
    new App(100L);
}
} // Salida: 3, 4 ...

```

62. ¿Cuál es la salida?

```

class App {
    public static void main(String[] args) {
        int i=42;
        String s = (i<40)?"life":(i>50)?"universe":"everething";
        System.out.println(s);
    }
}

```

```

        }
    } // Salida: everething

63. ¿Cuál es la salida?
class App {
    App() {
        System.out.println("1");
    }
    App(int num) {
        System.out.println("2");
    }
    App(Integer num) {
        System.out.println("3");
    }
    App(Object num) {
        System.out.println("4");
    }
    public static void main(String[] args) {
        String[]sa = {"333.6789","234.111"};
        NumberFormat inf= NumberFormat.getInstance();
        inf.setMaximumFractionDigits(2);
        for(String s:sa){
            System.out.println(inf.parse(s));
        }
    }
} // java: unreported exception java.text.ParseException; must be
caught or declared to be thrown

```

64. ¿Cuál es la salida?

```

class Y{
    public static void main(String[] args) {
        String s1 = "OCAJP";
        String s2 = "OCAJP" + "";
        System.out.println(s1 == s2);
    }
} // salida: true

```

65. ¿Cuál es la salida?

```

class Y{
    public static void main(String[] args) {
        int score = 60;
        switch (score) {
            default:

```

```
        System.out.println("Not a valid score");
        case score < 70:
            System.out.println("Failed");
            break;
        case score >= 70:
            System.out.println("Passed");
            break;
    }
} // salida: Error de compilacion - java: reached end of file while
parsing
```

**66.** ¿Cuál es la salida?

```
class Y{
    public static void main(String[] args)  {
        int a = 100;
        System.out.println(-a++);
    }
} // salida -100
```

**67.** ¿Cuál es la salida?

**Pregunta:**

Which of the following is not a valid array declaration?

**Respuesta correcta:**

```
int arr4[][] = new int[][][8];
```

**Explicación:**

1st array dimension must be specified at the time of declaration. new int[][][8]; gives compilation error as 1st dimension is not specified.

VALE

**68. ¿Cuál es la salida?**

```
class Y{
    public static void main(String[] args)  {
        byte var = 100;
        switch(var) {
            case 100:
                System.out.println("var is 100");
                break;
            case 200:
                System.out.println("var is 200");
                break;
            default:
                System.out.println("In default");
        }
    }
} // salida: Error de compilacion - java: incompatible types: possible
lossy conversion from int to byte
```

**69. ¿Cuál es la salida?**

Which of the following array declarations and initializations is NOT legal?

- |   |  |
|---|--|
| <input type="radio"/> A int [] arr3 = new int[3]{10, 20, 30}; | <input type="radio"/> B byte [] val = new byte[10];      |
| <input type="radio"/> C int [] arr2 = {1, 2, 3, 4, 5};        | <input type="radio"/> D char [] arr1 [] = new char[5][]; |

**ENVIAR RESPUESTA**

**Pregunta:**

Which of the following array declarations and initializations is NOT legal?

**Respuesta correcta:**

int [] arr3 = new int[3]{10, 20, 30};

**Explicación:**

You can't specify size at the time of initializing with data, hence new int[3]{10, 20, 30}; gives compilation error.

**VALE**

**70. ¿Cuál es la salida?**

```
class Y{
    public static void main(String[] args)  {
        A obj1 = new A();
        B obj2 = (B)obj1;
        obj2.print();
    }
}
class A {
    public void print(){
        System.out.println("A");
    }
}
class B extends A {
    public void print(){
        System.out.println("B");
    }
}
// ClassCastException
```

71. ¿Cuál es la salida?

```
class Y{
    public static void main(String[] args)  {
        String fruit = "mango";
        switch (fruit) {
    default:
        System.out.println("ANY FRUIT WILL DO");
    case "Apple":
        System.out.println("APPLE");
    case "Mango":
        System.out.println("MANGO");
    case "Banana":
        System.out.println("BANANA");
        break;
    }
}
}
```

72. ¿Cuál es la salida?

```
private String name;

Animal(String name) {
    this.name = name;
}

public String getName() {
    return name;
}
}

class Dog extends Animal {
    private String breed;
    Dog(String breed) {
        this.breed = breed;
    }

    Dog(String name, String breed) {
        super(name);
        this.breed = breed;
    }

    public String getBreed() {
        return breed;
    }
}
```

```

}

class Test {
    public static void main(String[] args) {
        Dog dog1 = new Dog("Beagle");
        Dog dog2 = new Dog("Bubbly", "Poodle");
        System.out.println(dog1.getName() + ":" + dog1.getBreed() +
":"
+ dog2.getName() + ":" + dog2.getBreed());
    }
} // compilation fails

```

**73.** ¿Cuál es la salida?

```

public class Main {
    public static void main(String[] args) throws ParseException {
        String[] sa = {"333.6789", "234.111"};
        NumberFormat nf = NumberFormat.getInstance();
        nf.setMaximumFractionDigits(2);
        for (String s: sa
            ) {
            System.out.println(nf.parse(s));
        }
    }
}/*Salida
333.6789
234.111
*/

```

**74.** ¿Cuál es la salida?

```

public class Main {
    public static void main(String[] args) throws ParseException {
        Queue<String> products = new ArrayDeque<String>();
        products.add("p1");
        products.add("p2");
        products.add("p3");
        System.out.println(products.peek());
        System.out.println(products.poll());
        System.out.println("");
        products.forEach(s -> {
            System.out.println(s);
        });
    }
} /**
 *p1
 * p1

```

\*  
\* p2  
\* p3

75. ¿Cuál es la salida?

```
public static void main(String[] args) throws ParseException {
    System.out.println(2+3+5);
    System.out.println("+"+2+3*5);
}
}// Salida: 10 + 215
```