Preguntas Java

1. ¿Cuántos errores tiene el siguiente código?

```java
public class MyClass {

    static final String s1 = "Hola";
    static final String s2 = "Mundo!";
    static final String s3 = "";

    static {
        s1 = "Hello";
    }

    static {
        s2 = "World";
    }

    public static void main(String args[]) {

      s3 = "Despues";

    }
}
```

3

Si una variable se declara final, ya no puede cambiar su valor

2.

```java
public class MyClass {
    public static void main(String args[]) {

      String Hola = "Hola mundo";

      System.out.println(Hola.charAt(11));
    }
}
```

StringIndexOutOfBoundsException

No llega al índice 11, llega solo al 9

3.

```java
public class MyClass {
    public static void main(String args[]) {

      int x = 10;
      int y = 5;
      int z = 111;
      //Aqui habia otras cosas de char´s

      switch(z){
          case 111:
              System.out.println("Todo bien");
          case x:
              System.out.println("valor de x" + x);
              break;
          case y:
              System.out.println("valor de y" + y);
              break;
      }
    }
}
```

Error de compilación

No puede haber variables en los case a menos que sean final

4.

The following are the complete contents of TestClass.java file. Which packages are automatically imported?

```java
class TestClass{
   public static void main(String[] args){
     System.out.println("hello");
   }
}
```

java. lang y el paquete sin nombre

5.

```java
public static void main(String[] args) {
      System.out.println("Hola Mundo");
      int [] arr = new int[]; //No se indicó la longitud
      int [][] d = new int[]; //No se indicó la longitude y se declaran diferentes tamaños en ambos lados
      int a [][] = new int[7][6];
      Object [][][] objects = new Object[4][0][5]; }
```

6.
```
public static void main(String[] args) {
     int a=0;
     System.out.println(a++ + 2);
     System.out.println(a); }
```

2 1

Al hacer la primera impresión no modifica a, se modifica al finalizar esa impresión

7. ¿Qué se va a imprimir al ejecutar el método main()?

```
public class Spider {
     void spider(int a){
          System.out.print("Spider"); } }
public class Genero {
     void genero(double b){
          System.out.print("Genero"); } }
public class SpiderMain {
     public static void main(String[] args) {
          Spider spider = new Spider();
          Genero genero = new Genero();
          spider.spider(4);
          genero.genero(9.0); } }
```

Spider Genero

Spider y Genero no tienen constructores declarados, solo funciones

8. ¿Con cuál de las siguientes líneas imprimirá true?

A) boolean r1 = a.equals(b);
B) boolean r2 = a==b;
C) boolean r3 = a.equalsIgnoreCase(b);

```
String a = "Hola";
String b = "hola";
```

```
boolean resultado = false;
if (.........) System.out.println("Equals");
else System.out.println("Not Equals");
```

Compara el contenido de la cadena pero sin diferenciar mayúsculas y minúsculas

9.

```
import java.util.*;


public class Main {

    public static void main(String[] args) throws Exception {

    short kk = 11;
    short ii;
    short jj = 0;
    for (ii = kk; ii > 6; ii -= 1)
    {jj++;}
    System.out.println("jj = " + jj);

    }
}
```

jj = 5: Va a entrar en el for mientras 11 > 6, es decir, 5 veces

10.

✓ Which three methods, inserted individually at line, will correctly complete class Two(Choose three)? *1/1

```
10.  class One {
11.      void foo() { }
12.  }
13.  class Two extends One {
14.      //insert method here
15.  }
```

A: int foo() { /*más código aquí*/ } //La firma es la misma así que es un overriding

B: void foo() { /*más código aquí*/ } //Es un duplicado de la función

C: public void foo() { /*más código aquí*/ } //Overriding

D: private void foo() { /*más código aquí*/ } //No se puede reducir la visivilidad

E: protected void foo() { /*más código aquí*/ } //Protected tiene más visibilidad que default

11. A: `public void suma(int... g, int []q){}` //Los varargs debe como último parámetro

B: `public void div(int[], ...int r){}` //Los puntos van después del tipo de dato

C: `public int suma(int.. d, float...o){return d+o;}` //Solo puede haber un varargs por función, y van con tres puntos, no dos

D: `public void t(int... h, double... t){}` //Solo puede haber un varargs por función

E: `public void ad(int b, double s, String... a){}`

12.
```
public class VariableFinal {
     public class Persona extends Carro{
          final String nombre;
          public Persona() {
               Carro carro = new Carro();
               nombre = "andres"; } }
     public class Carro{}
     public static void main(String[] args) {
          Persona persona = new Persona(); } }
```

Error de compilación

No se puede cambiar el valor de nombre

13.
```
boolean x = false;
boolean z = false;
int y = 20;
System.out.println(x);
x = (y!=10) ^ (z!=false);
System.out.println(x + y + z);
```

false true20false

x = (true) ^ (false) = true

14. Asume que este switch acepta datos de tipo entero. ¿Qué tipo de dato no se podría colocar en el espacio en blanco?

```
_____ dayOfWeek= 1;
switch (dayOfWeek) {
case 1: System.out.println("Monday");
case 2: System.out.println("Tuesday");
case 3: System.out.println("Other day"); break; }
```

A: int

B: long //Switch no acepta este tipo de dato

C: char

15. Elige todas las excepciones no verificadas

A: IllegalArgumentException

B: NumberFormatException

C: IOException

D: NullPointerException

E: Exception

F: ArrayIndexOutOfBoundsException

16. ¿Cuál es el resultado?

```
int[] ii = null;
for (int xx : ii) System.out.println(xx);
```

NullPointerException

No se puede iterar un elemento vacío

17. ¿Cuál es la salida?

```
String movie = "Thriller";
switch (movie) {
     case "Thriller":
          System.out.println("Movie Thriller");
     case "Comedy": System.out.println("Movie Comedy");
     case "Romance": System.out.println("Movie Romance");
          break;
     case "Action": System.out.println("Movie Action"); }
```

Movie Thriller Movie Comedy Movie Romance

Una vez que el switch encuentra un elemento que coincide, continua ejecutando código hasta que encuentra un break

18. ¿Cuál es el resultado?

```
String s = "ABCD";
s.trim();
s.toUpperCase();
s+= " 123";
System.out.println(s.length());
```

8

A ABCD le suma tres números y un espacio en blanco

19. Usando StringBuilder sb, elige las opciones que den el mismo resultado que la concatenación original

```
String name = "Joe";
int age = 31;
String result = "My name is " + name + ", my age is " + age;
System.out.println(result);
```

A: sb.append("My name is " + name + ", my age is " + age);

B: sb.insert("My name is " + name).append(", my age is "+ age); //insert require indicar el índice

C: sb.insert("My name is").insert(name).insert(", my age is ").insert(age); //insert require indicar el índice

D: sb.append("My name is ").append(name).append(", my age is ").append(age);

20. ¿Cuál es el resultado?

```
int[] at = { 1, 2 };
for (int x : at) {
     System.out.println(x + ", ");
     if (x < at.length) break; }
```

1,

El primer valor de x es 1. La longitud de at es 2. Por lo tanto, el bucle se rompe después de la primera impresióm

21. ¿Cuál es el resultado?

```
int i = 42;
String s = (i < 40) ? "Greater than" : false;
System.out.println(s);
```

Error de compilación

No se puede asignar un valor booleano a una variable int

22. ¿El siguiente código compila?

```
int num = 1;
switch (num) {
     case 1:
     case 2: System.out.println("Caso 2");
     case 3: System.out.println("Caso 3"); break; }
```

Sí

Imprime Caso 2 Caso 3

23. ¿Cuál de los siguientes códigos podría estar presente en un método que no retorna ningún valor? (Elige dos)

A: return null;

B: return void;

C: return;

D: Omitir el uso de la palabra clave return

24. _____ occurs when a subclass redefines a method from its superclass, while _____ happens when multiple methods in the same class share the same name but differ in their parameters.

A: Overloading / Overriding

B: Overriding / Overloading

C: Inheriting / Overriding

D: Overloading / Inheriting

25. ¿Cuál sería la salida en consola?

```
class MyThread extends Thread {
    public void run() {
```

```
          System.out.println("Thread is running"); } }
public class Main {
    public static void main(String[] args) {
        Thread t1 = new MyThread();
        Thread t2 = new MyThread();
        t1.start();
        t2.start(); } }
```

Thread is running Thread is running

En orden aleatorio

26. ¿Cuál sería la salida en consola?

```
List<Integer> numbers = new ArrayList<>();
numbers.add(1);
numbers.add(2);
numbers.add(3);
try {
     for (int i = 0; i <= numbers.size(); i++) {
          System.out.println(numbers.get(i)); } }
catch (IndexOutOfBoundsException e) {
     System.out.println("Exception caught"); }
```

1 2 3 Exception caught

El for itera una vez fuera de los límites del arreglo

27. ¿Cuál sería la salida en consola?

```
import java.util.ArrayList;
import java.util.List;
interface Movable { void move(); }
abstract class Vehicle { abstract void fuel(); }
class Car extends Vehicle implements Movable {
    void fuel(){ System.out.println("Car is refueled"); }
    public void move(){
```

```
          System.out.println("Car is moving"); } }
public class Main {
    public static void main(String[] args) {
        Vehicle myCar = new Car();
        myCar.fuel();
        ((Movable) myCar).move(); } }
```

Car is refueled Car is moving

Aunque se haga el casteo, la interfaz y el objeto conocen el método

28. ¿Cuál sería la salida en consola al ejecutar este código?

```
class Box<T> {
    private T item;
    public void setItem(T item) { this.item = item; }
    public T getItem() throws ClassCastException {
        if (item instanceof String) return (T)item;
        throw new ClassCastException("Is not a String"); }}
public class Main {
    public static void main(String[] args) {
        Box<String> stringBox = new Box<>();
        stringBox.setItem("Hello");
        try {
            String item = stringBox.getItem();
            System.out.println(item);
        } catch (ClassCastException e) {
            System.out.println("Exception caught"); } } }
```

Hello

Solo se Lanza la excepción si T no es un String

29.

```
public class Main {
     public static void main(String[] args) {
          Padre objetoPadre = new Padre();
```

```
        Hija objetoHija = new Hija();
        Padre objetoHija2 = (Padre) new Hija();
        objetoPadre.llamarClase();
        objetoHija.llamarClase();
        objetoHija2.llamarClase();
        Hija objetoHija3 = (Hija) new Padre();
        objetoHija3.llamarClase(); } }
class Hija extends Padre {
    public Hija() {}
    @Override
    public void llamarClase() {
        System.out.println("Llame a la clase Hija"); } }
class Padre {
    public Padre() {}
    public void llamarClase() {
        System.out.println("Llame a la clase Padre"); } }
```

Llame a la clase Padre Llame a la clase Hija Llame a la clase Padre ClassCastException

objetoHija2 llama a la clase Padre porque se realiza un casteo. objetoHija 3 lanza una excepción porque no se puede castear a sub clases

30.

```
public class Main {
     public static void main(String[] args) {
          Animal uno = new Animal();
          Animal dos = new Dog();
          uno.makeSound();
          dos.makeSound();
          Dog tres = (Dog) new Animal();
          tres.makeSound(); } }
class Animal {
     void makeSound() {
          System.out.println("Animal sound"); } }
class Dog extends Animal {
```

```
void makeSound() { System.out.println("Wau Wau"); } }
```

Animal sound Wau wau ClassCastException

Imprime Wau wau porque cuando se trata de métodos, mira en la clase del objeto. Se lanza la excepción porque no se puede castear a sub clases

31. The feature which allows different methods to have the same name and arguments type, but the different implementation is called?

Overriding

32.
```
for(int i=10, j=1; i>j; i--, j++)
        System.out.println(j %i);
```

1 2 3 4 5

10, 1(true) -> 9, 2(true) -> 8, 3(true) -> 7, 4(true) -> 6, 5(true) -> 5, 6(false) -> entra 5 veces

33. We perform the following sequence of actions:

A: Insert the following elements into a set: 1,2,9,1,2,3,1,4,1,5,7.

B: Convert the set into a list and sort it in ascending order.

1 2 3 4 5 7 9

El set no acepta elementos duplicados

34. A public data member with the same name is provided in both base as well as derived classes

Hiding/ shaowing

35.
```
public class Constructor {
     public static void main(String... args){
          String [] cities = {"hola","dos"}; }
     static class MySort implements Comparator{
          public int compare(String a, String b){
               return b.compareTo(a); } } }
```

Error de compilación

Comparator tiene el método compare()