# GUIA ASO

1. ¿Cuál es el comando utilizado para deshacer el último commit en git?
   git reset HEAD-1
   Este comando deshace el ultimo commit realizado en la rama actual, git reset se utiliza para mover la punta de la rama actual a un commit especifico y el HEAD-1 especifica el commit anterior al HEAD actual y se utiliza para correcciones de error, reorganización de historial de commits
   git revert se utiliza para deshacer los efectos de un commit especifico creando un nuevo commit que revierte los cambios introducidos por ese commit, este crea un nuevo commit que deshace los cambios del commit especifico, manteniendo intacto el historial de commits.
2. ¿Cuál es la diferencia entre una clase abstracta y una interfaz en java 8?

   Una clase abstracta puede tener métodos abstractos sin implementación y métodos con implementación concretos, esta puede contener campos, constructores y métodos no abstractos, una clase puede extender solo una clase abstracta y actúa como base por medio de la palabra “extends”. Este tipo de clases puedes ser invocadas por otras clases. Sus atributos pueden ser de cualquier nivel (public, protected,default o private).

   Una interfaz sus métodos son abstractos por defecto, lo que significa que no tienen implementación y una clase puede implementar múltiples interfaces utilizando la palabra “implements” lo que significa que significa herencia múltiple. Las interfaces no tienen constructores porque no se pueden instanciar directamente y son contratos que las clases pueden implementar. Sus atributos son automáticamente públicos y abstractos o almenos que sean definidos como estáticos o por defecto.

3. De los siguientes ¿qué tipos de declaraciones se deben usar para contar la cantidad de monedas de 5 centavos en una matriz de cadenas de varias monedas? (Elije todas las correctas)
   La iteración es el proceso de repetir un conjunto de instrucciones un cierto número de veces o hasta que se cumpla una condición específica. En el caso de contar la cantidad de monedas de 5 centavos en una matriz de cadenas, necesitas iterar sobre cada elemento de la matriz para verificar si es igual a "5 centavos" y contar cuántas veces ocurre esta condición. Un bucle for o un bucle foreach son formas comunes de realizar esta iteración en Java.

4. ¿Qué es un archivo JAR en java?
   Es un archivo que se utiliza para empaquetar y distribuir uno o más archivos Java, como clases, recursos, metadatos y archivos auxiliares en una única unidad. Estos son similares a los archivos Zip pero estas diseñado específicamente para contener archivos y recursos relacionados con Java.

5. ¿Qué es la sobrecarga de métodos en Java?

Es un concepto utilizado que permite definir múltiples métodos en una clase con el mismo nombre, pero con diferentes listas de parámetros es decir que estos pueden ser diferentes entre ellos, lo que permite una funcionalidad similar, pero con diferentes invocaciones y adaptaciones a diferentes tipos de datos o situaciones.

***6.*** ¿Cuál es la diferencia entre un ArrayList y un LinkedList en Java?
ArrayList es un arreglo dinámico continuo de memoria que almacena elementos, y este se incrementa su tamaño automáticamente cuando alcanza su capacidad máxima, y su acceso a los elementos es más rápido, sus inserciones y eliminaciones son más lentas.
LinkedList es una lista doblemente enlazada para elementos, y estos se almacenan en un nodo que contiene una referencia al siguiente nodo y al nodo anterior, su acceso a sus elementos es más lento porque no hay un acceso aleatorio directo a sus elementos, se debe seguir un orden desde el nodo inicial o final a los nodos sucesivos para llegar al destino y sus inserciones y eliminaciones son más rápidas.

***7.*** ¿Cuándo se debe usar un bloque finally en una declaración try regular (no una prueba con recursos)?
Se utiliza cuando necesitas asegurar de que ciertas acciones se realicen, independientemente si ocurre una excepción o no dentro del bloque “try” para limpieza de estado donde las operaciones de limpieza o restauración del programa antes del salir del bloque de try o para manejo de excepciones en donde este se ejecutara incluso si no estar dentro del try o si no hay un catch.

***8.*** ¿Cuál es el propósito principal de los test unitarios?
Se utilizan para validar la funcionalidad, detectar errores, facilitar la refactorización, documentación dinámica, promover la calidad del código y facilitar la colaboración. Los test unitarios son una parte fundamental de las prácticas de desarrollo de software modernas y juegan un papel crucial en la creación de sistemas robustos, fiables y mantenibles.

***9.*** Bloque de código.

```java
public class Test3 {
    public static void main(String[] args) {
        String cad1 = "hola";
        String cad2 = new String( original: "hola");
        String cad3 = "hola";

        if (cad1 == cad2)
            System.out.println("ca1 es igual a cad2");
        else System.out.println("cad1 diferente a cad2");

        if (cad1 == cad3)
            System.out.println("cad1 es igual a cad3");
        else
            System.out.println("cad1 diferente a cad3");
    }
}
```

De acuerdo con como declaramos las variables de referencia y los objetos, podemos observar que cad1 y cad 3 son iguales, pero cad1 y cad2 son diferentes al declararlo como un new Strin(“Hola”) donde se esta creando un nuevo objeto y no hacemos que la variable de referencia apunte al mismo objeto que los otros string.

***10.*** ¿Cuál es la salida al ejecutar el siguiente código?

```java
class Mammal{
    public Mammal(int age){
        System.out.println("Mammal");
    }
}
public class Platypus extends Mammal{
    public Platypus(){
        System.out.println("Platypus");
    }

    public static void main(String[] args) {
        new Mammal(5);
    }
}
```

En el IDE imprime en pantalla Mammal

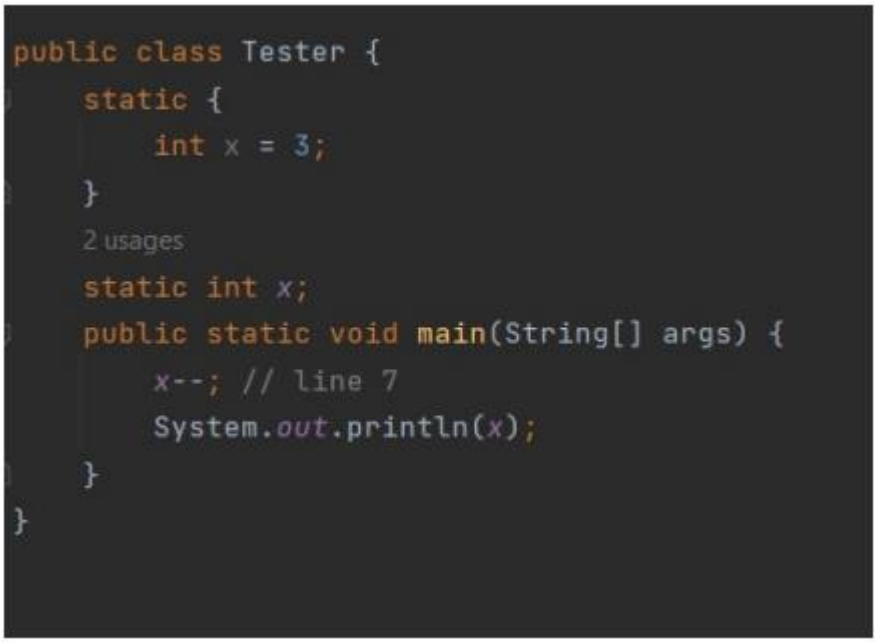
***11.*** ¿Cómo se manejan las excepciones en java?

Se utilizan los bloques **try** para contener el código propenso a excepciones, el bloque **catch** se utiliza para manejar excepciones específicas que pueden ocurrir en el bloque **try**, y el bloque **finally** se utiliza para contener código que se ejecuta siempre, independientemente de si se produce una excepción o no.

***12.*** ¿La anotación @Ignore es usada para omitir un test por lo que no se ejecuta?

La anotación @Ignore en JUnit se utiliza para omitir la ejecución de un método de prueba específico o de una clase de prueba completa. Cuando un método de prueba está anotado con @Ignore, JUnit lo tratará como si estuviera deshabilitado y no lo ejecutará durante la ejecución de la suite de pruebas.

***13.*** ¿Cuál es el resultado de compilar y ejecutar el siguiente código?



```
public class Tester {
    static {
        int x = 3;
    }
    2 usages
    static int x;
    public static void main(String[] args) {
        x--; // line 7
        System.out.println(x);
    }
}
```

Ya que la inicializacion del int x esta dentro de los corchetes al entrar a main esta al no estra inicializada tiene un valor de cero por default y cuando se decrementa su valor este cambia a -1 por eso se imprime -1 en pantalla.

***14.*** ¿Qué es un operador de short circuit?

Es un operador en programación que no evalúa todos sus operandos si no es necesario, y detiene la evaluación tanto como el resultado final puede ser determinado y estos pueden && y || .

***15.*** ¿Qué es el patrón de diseño DAO y cómo se implementa en Java?

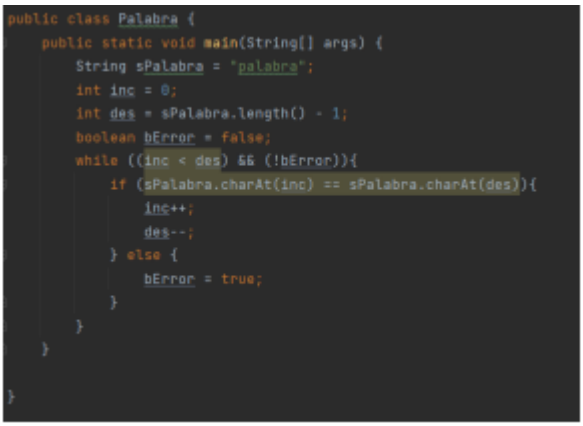
DAO (Data Access Object) es un patrón de diseño de software que se utiliza para separar la lógica de acceso a datos de la lógica de negocio en una aplicación. Su objetivo es proporcionar una capa de abstracción entre la aplicación y la fuente de datos (como una base de datos, un servicio web, etc.), lo que facilita el cambio y la evolución de la capa de acceso a datos sin afectar a la lógica de negocio.

Se proporcionan implementaciones concretas de la interfaz DAO para interactuar con la fuente de datos real. Estas implementaciones contienen la lógica específica para realizar operaciones CRUD en la fuente de datos. Por ejemplo, para una base de datos relacional, las implementaciones del DAO podrían utilizar JDBC para interactuar con la base de datos.

***16.*** ¿Qué es un endpoint en una API REST?

Se refiere a un punto final especifico (URL) en el servidor donde los clientes pueden interactuar con el sistema, cada uno de estos representa una operación o acción específica que el servidor puede realizar sobre los recursos que maneja. Tienen una url única utiliza los verbos de HTTP, devuelve datos en formatos como JSON, XML,HTML o dependiendo de la solicitud , y pueden requerir autenticación o autorización antes de ser accedidos o utilizados.

***17.*** ¿Qué hace el siguiente programa?



```
public class Palabra {
    public static void main(String[] args) {
        String sPalabra = "palabra";
        int inc = 0;
        int des = sPalabra.length() - 1;
        boolean bError = false;
        while ((inc < des) && (!bError)){
            if (sPalabra.charAt(inc) == sPalabra.charAt(des)){
                inc++;
                des--;
            } else {
                bError = true;
            }
        }
    }

}
```

Realiza un bucle que se hara hasta que este sea deacuerdo con el contador menor a la longitud de la cadena y al poner el condicional compara los valores por cada una de las letras de la palabra para verificar que es un palíndromo o no.

***18.*** ¿Cuál de las siguientes opciones son verdaderas?

a) Java es un lenguaje orientado a objetos.
b) El código Java compilado en Windows puede ejecutarse en Linux.
c) Java permite la sobrecarga de operadores
d) Java es un lenguaje de programación funcional.
e) Java es un lenguaje procedimental.
f) Java tiene punteros a ubicaciones específicas en la memoria.

Java es un lenguaje de programación de propósito general que soporta múltiples paradigmas de programación, incluyendo el paradigma orientado a objetos, el paradigma imperativo (procedimental), y en cierta medida, el paradigma funcional. El código java en Windows puede ejecutarse en Linux siempre y cuando tenga independencia de plataforma, contenga una máquina virtual java y sea compatible con bibliotecas y dependencias

***19.*** ¿Qué es Maven y para qué se utiliza en el desarrollo de aplicaciones?

Maven es una herramienta de gestión de proyectos y construcción de software utilizada en el desarrollo de aplicaciones Java. Proporciona un conjunto de estándares y convenciones para la gestión de proyectos, la gestión de dependencias, la compilación, el empaquetado y la distribución de software.

Se utiliza en el desarrollo de app para la gestión de dependencias de un proyecto y manejar automáticamente la descarga e inclusión de las dependencias requeridas en el proyecto. Define un ciclo de vida de construcción estándar en fases predefinidas como compile, test, package, install, deploy. Integra repositorios de artefactos centralizados como el Repositorio Central de Maven, que permite búsqueda y descarga de dependencias.

***20.*** ¿Cuál de lo siguiente es cierto? (elija todas las correctas)

a) javac compila un archivo .java en un archivo .bytecode.
b) Java toma el nombre del archivo .bytecode como parámetro.
c) javac compila un archivo .java en un archivo .class

d) Java toma el nombre de la clase como parámetro.
e) Java toma el nombre del archivo .class como parámetro.
f) javac compila un archivo .class como archivo java.

javac es el compilador de java que toma un archivo fuente .java como entra y lo compila en un archivo bytecode .class . Java no puede tomar nombre del archivo bytecode como parámetro, la ejecución de un programa Java se realiza mediante el comando java. Java no toma el nombre de la clase como parámetro.

***21.*** ¿Qué es Git y cuáles son algunos de sus comandos básicos?
Es un sistema de control de versiones y sus comandos básicos son: init, commit, clone, add, status, log, branch, checkout, merge, pull y push.

***22.*** Dados los siguientes segmentos de código, ¿Qué respuesta no es una implementación de java válida?

a) int variableA = 10;

float variableB = 10.5f;

int variableC = variableA + variableB;

b) byte variableA = 10;

double variableB = 10.5f;

double variableC = variableA + variableB;

c) byte variableA = 10;

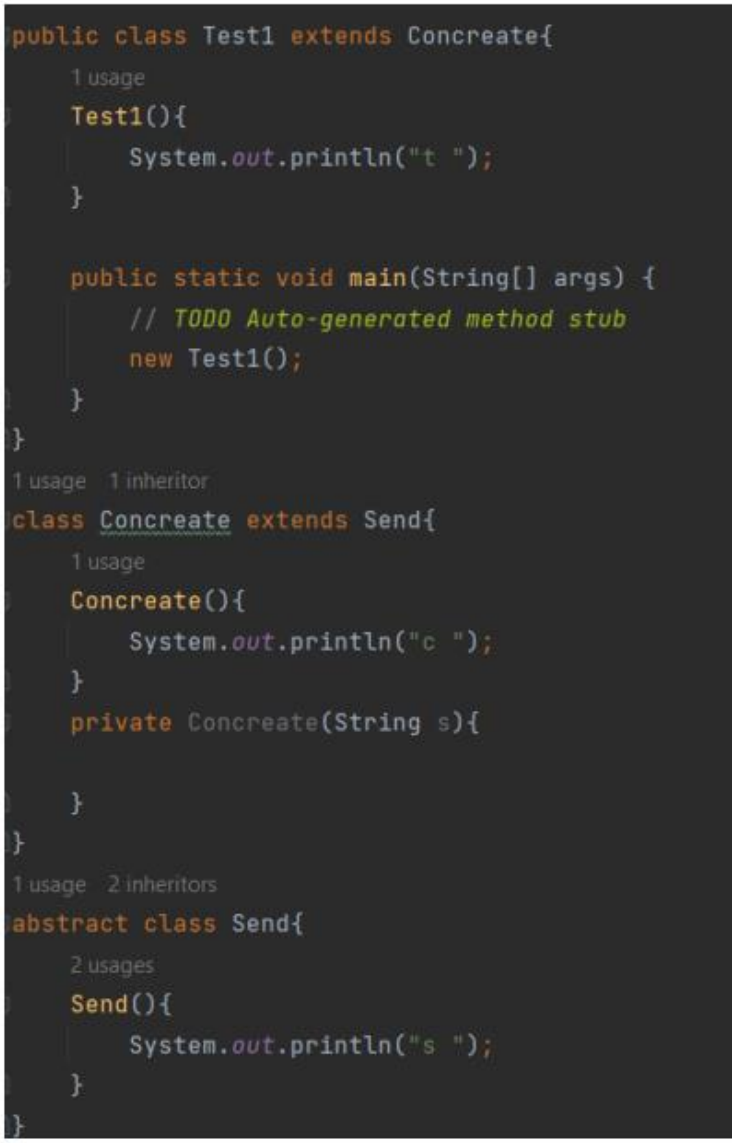
float variableB = 10.5f;

float variableC = variableA + variableB;

***23.*** ¿Qué escenario es el mejor uso de una excepción?
El uso de excepciones en Java es apropiado para manejar cualquier situación excepcional que pueda interrumpir el flujo normal de ejecución del programa y que no pueda ser manejada de manera adecuada en el contexto del código que la originó. Es importante diseñar y manejar las excepciones de manera adecuada para garantizar que el programa sea robusto, seguro y fácil de depurar.

***24.*** ¿Qué es un bean en Spring?
Es un objeto gestionado por el contenedor de Spring. Los beans son componentes fundamentales en la arquitectura de Spring, y representan los diferentes elementos de una aplicación que son gestionados por el contenedor de Spring

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



```
public class Test1 extends Concreate{
    1 usage
    Test1(){
        System.out.println("t ");
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new Test1();
    }
}
1 usage  1 inheritor
class Concreate extends Send{
    1 usage
    Concreate(){
        System.out.println("c ");
    }
    private Concreate(String s){

    }
}
1 usage  2 inheritors
abstract class Send{
    2 usages
    Send(){
        System.out.println("s ");
    }
}
```

De acuerdo con el código este podría imprimir s,c,t de acuerdo con la herencia,

***26.*** ¿Cuáles de las siguientes afirmaciones sobre el polimorfismo son verdaderas? (Elija todas las correctas)

a) Si un método toma una superclase de 3 objetos, cualquiera de esas clases puede pasarse como parámetro del método

b) Un método que toma un parámetro con tipo java.lang.object tomará cualquier referencia

c) Una referencia a un objeto se puede convertir a una subclase de objetos en una conversión explícita.

d) Todas las excepciones de conversión se pueden detectar en tiempo de compilación

e) Al definir un método de instancia pública en la súper clase, garantiza que el método específico se llamará al método en la clase principal en tiempo de ejecución

Un método toma una superclase como parámetro, puede pasar a cualquier superclase, así como un método de un parámetro de tipo Object puede tomar como referencia cualquier objeto ya que todas las clases en java son subclases de Object. Una referencia a un objeto se puede convertir en una subclase de objetos mediante una conversión explicita y es conocido como casting.

No todas las excepciones de conversión se pueden detectar en tiempo de ejecución y un método de instancia publica en una superclase no garantiza que le método especifico se llamara al método en la clase principal en tiempo de ejecución ya que estos se resuelven en tiempo de compilación del tipo estático de la referencia

***27.*** ¿Son patrones de diseño de software estructural?

- Adapter (Adaptador): Permite que interfaces incompatibles trabajen juntas.
- Bridge (Puente): Desacopla una abstracción de su implementación para que puedan variar independientemente.
- Composite (Composite): Compone objetos en estructuras de árbol para representar jerarquías de parte-todo.
- Decorator (Decorador): Añade funcionalidad a los objetos dinámicamente.
- Facade (Fachada): Proporciona una interfaz simplificada para un conjunto más grande de clases.
- Proxy (Proxy): Proporciona un sustituto o marcador de posición para controlar el acceso a un objeto.

***28.*** Seleccione la respuesta que considere correcta dado el siguiente bloque de código.

```
import java.util.Arrays;
import java.util.List;

public class Example {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
        double result = numbers.stream() Stream<Integer>
                .mapToInt(n -> n) IntStream
                .average() OptionalDouble
                .orElse( other: 0);
        System.out.println(result);
    }
}
```

Se tiene una lisy la cual contiene valores y ,se realiza el calculo de promedio de esta list y se imrpimira 3.0

***29.*** ¿Qué son las pruebas de integración?

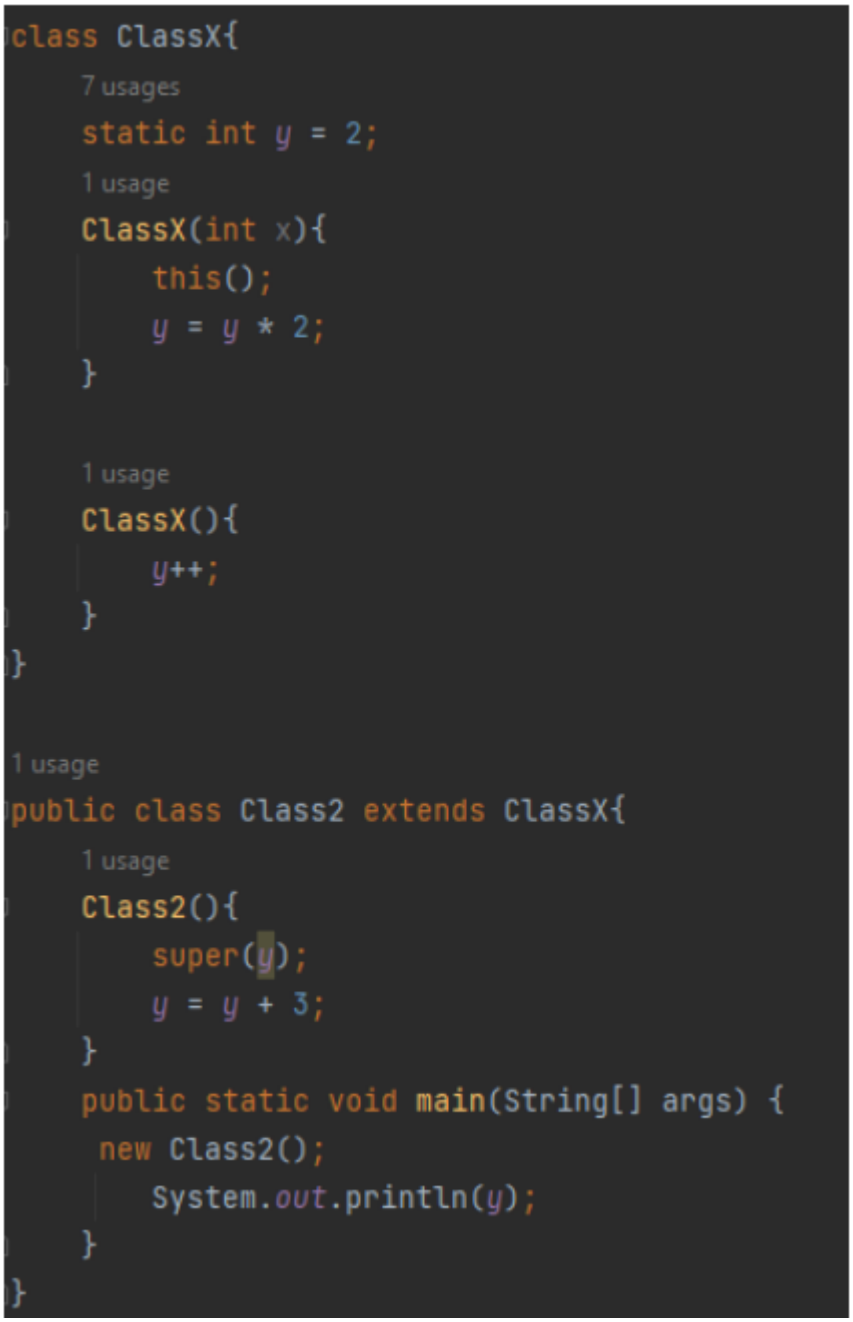
Son una fase de prueba en el desarrollo de software donde se prueban conjuntos de módulos o componentes de software ya integrados para asegurar que funcionen

correctamente juntos como un sistema completo. En lugar de probar los componentes individualmente (como se hace en las pruebas unitarias), las pruebas de integración se centran en probar cómo interactúan los diferentes módulos cuando se combinan.

***30.*** ¿Qué comando se utiliza para enviar los cambios confirmados en un repositorio local al repositorio remoto?

Git Push envía los cambios confirmados en la rama local al repositorio remoto correspondiente. Por defecto, git push enviará los cambios de la rama local al mismo nombre de rama en el repositorio remoto.

***31.*** Seleccione la respuesta correcta, dado el siguiente bloque de código.



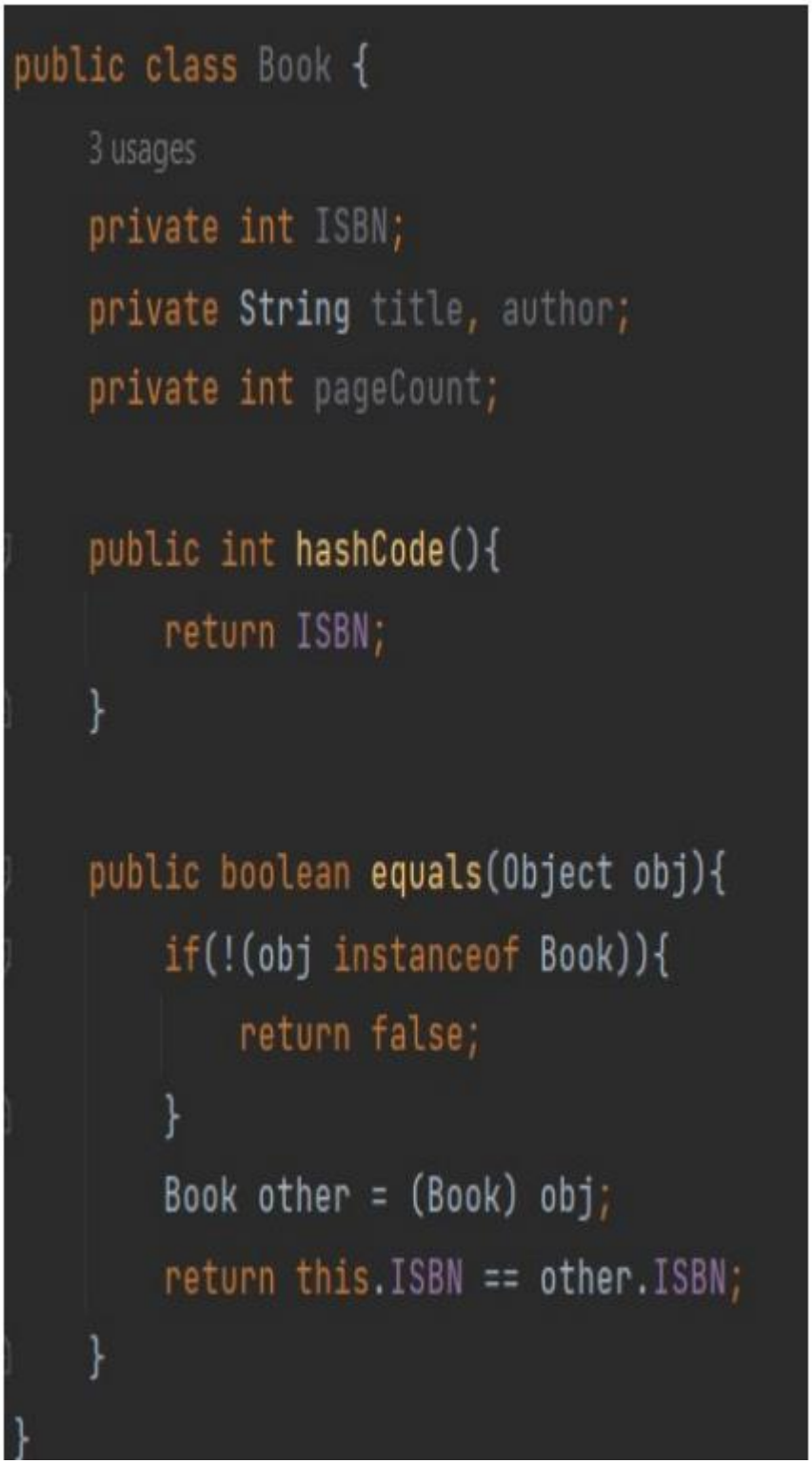
```
class ClassX{
    7 usages
    static int y = 2;
    1 usage
    ClassX(int x){
        this();
        y = y * 2;
    }

    1 usage
    ClassX(){
        y++;
    }
}

1 usage
public class Class2 extends ClassX{
    1 usage
    Class2(){
        super(y);
        y = y + 3;
    }
    public static void main(String[] args) {
     new Class2();
        System.out.println(y);
    }
}
```

Al no ponerle un valor en el constructor de la classX o de la class2 este inicializara el contructos que incrementa el valor de y y sera y=3 y despues utilizara el constructor donde se ingresa el valor de y donde multiplica y *2 y quedara como y=6 y entonces realizara la operación declarada en el constructor de la class2 lo que finalizara con el valor de y=9.

***32.*** ¿Cuál es el comando utilizado para crear una nueva rama en Git?

El comando es git branch <branch name>. Crea una nueva rama con el nombre especifico.

***33.*** ¿Cuál es el resultado de compilar la siguiente clase?



```
public class Book {
    3 usages
    private int ISBN;
    private String title, author;
    private int pageCount;

    public int hashCode(){
        return ISBN;
    }

    public boolean equals(Object obj){
        if(!(obj instanceof Book)){
            return false;
        }
        Book other = (Book) obj;
        return this.ISBN == other.ISBN;
    }
}
```

Ya que este no imprime nada en pantalla y las asignaciones y métodos son correctos este compilara correctamente.

***34.*** ¿Cuál es la primer línea en fallar al compilar?

```
1   class Tool {
2   private void repair() {} //r1
3   1 override
4   void use(){}
5   }
6   class Hammer extends Tool{
7   private int repair(){return 0; } //r3
8   private void use(){}//r4
9   public void bang(){}//r5
10  }
11
```

Marca error en la linea r4 ya que no se realiza el override en las clases .

***35.*** ¿Qué es Git?

Git es un sistema de control de versiones distribuido, diseñado para rastrear cambios en archivos y coordinar el trabajo en proyectos de desarrollo de software.

***36.*** ¿Cuáles son las excepciones para lanza la JVM?

IOException, FileNotFoundException, ParseException, SQLException, ClassNotFoundException, NullPointerException, ArrayIndexOutOfBoundsException, ArithmeticException, IllegalArgumentException, ClassCastException.

***37.*** ¿Cuál es el comando utilizado para fusionar una rama en Git?

El comando utilizado para fusionar una rama es git merge. Este comando permite combinar los cambios de una rama específica en la rama actual en la que te encuentras.

***38.*** ¿Qué es REST y cuál es su relación con las API web?

REST (Representational State Transfer) es un estilo de arquitectura para diseñar sistemas distribuidos, particularmente servicios web. Una API web (Application Programming Interface) es un conjunto de reglas y mecanismos que permite que diferentes programas se comuniquen entre sí a través de la web. Cuando se diseña una API web siguiendo los principios de REST, se le denomina API RESTful.

***39.*** ¿Cuál es el comando utilizado para actualizar la rama local con los cambios de la rama remota en Git?

En Git, el comando utilizado para actualizar la rama local con los cambios de la rama remota es git pull. Este comando realiza dos acciones: primero, recupera (fetch) los cambios desde el repositorio remoto y luego los fusiona (merge) con la rama local actual.

***40.*** ¿Qué es un microservicio?

Es una arquitectura de software que estructura una aplicación como una colección de servicios pequeños, independientes y autónomos. Cada microservicio es responsable de una funcionalidad específica y puede ser desarrollado, desplegado y escalado de manera independiente. Esta arquitectura es una evolución de la arquitectura monolítica, donde todos los componentes de una aplicación están integrados en un solo código base.

***41.*** Dado el siguiente código:

```java
public class Main {
    public static void main(String[] args) {
        int[] numeros = {1,2,3,4,5};
        int suma = 0;
        for (int i = 1; i <= numeros.length; i++){
            suma += numeros[i];
        }

        System.out.println("La suma de los números es: " + suma);
    }
}
```

Compila sin errores e imprimirá la suma del arreglo de numero.

***42.*** ¿Qué método se utiliza para obtener el mensaje de una excepción en Java?

Es el método getMessage() el cual pertenece a la clase Throwable que es la clase base para todas las excepciones y errores en java.

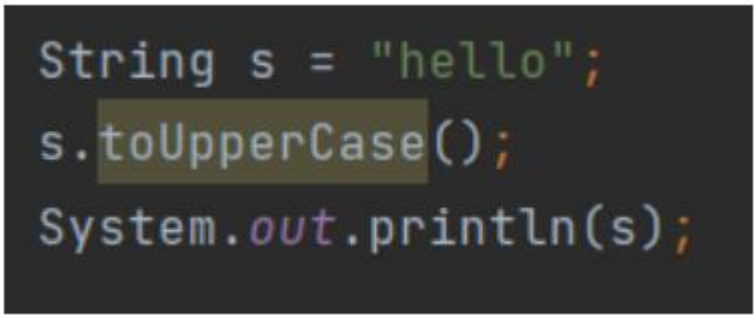
***43.*** ¿Cuál de las siguientes afirmaciones son verdaderas? (Elije todas las correctas)

a. Puede declarar solo excepciones no comprobadas (unchecked).
b. Las excepciones en tiempo de ejecución son lo mismo que las excepciones no comprobadas.
c. Las excepciones en tiempo de ejecución son lo mismo que las excepciones comprobadas.
d. Solo puede declarar excepciones comprobadas (checked).
e. Solo puede manejar subclases de Exception

En Java las excepciones no comprobadas son subclases de runtimeException y no necesitan ser declaradas en el método con throws. Las excepciones en tiempo de ejecución son lo mismo que las excepciones no comprobadas. Las excepciones en tiempo de ejecución no son lo mismo que las excepciones comprobadas, y estas deben ser declaradas en la firma del método y manejadas explícitamente. Pueden declararse excepciones comprobadas

como no comprobadas con el método usando throws. Se pueden usar subclases de Throwable, que incluye Error no es recomendable porque representa errores serios que una aplicación típica no debería intentar manejar.

***44.*** ¿Cuál es el resultado de ejecutar el siguiente código?




Imprimira en pantalla hello ya que aunque comvertimos las cadena en mayusculas no hacemos la asigancion para que apunte a ese objeto donde hello esta en mayusculas.

***45.*** ¿Cuál es el paquete de importación necesario para usar la clase ArrayList?
Para usar la clase ArrayList en Java debemos de importar el paquete java.util .

***46.*** ¿Cuál es el formato de los datos que se envían y reciben en una API REST?
Usan típicamente JSON pero también puede usar formatos como XML ,YAML o incluso plain text dependiendo de la especificación de la API y las necesidades del cliente y servidor.

***47.*** ¿Cuál es la función del operador de doble dos puntos (::) en Java 8?
Es un operador de referencia de método, y de una forma concisa de referencia de métodos y constructores sin tener que invocarlos explícitamente.

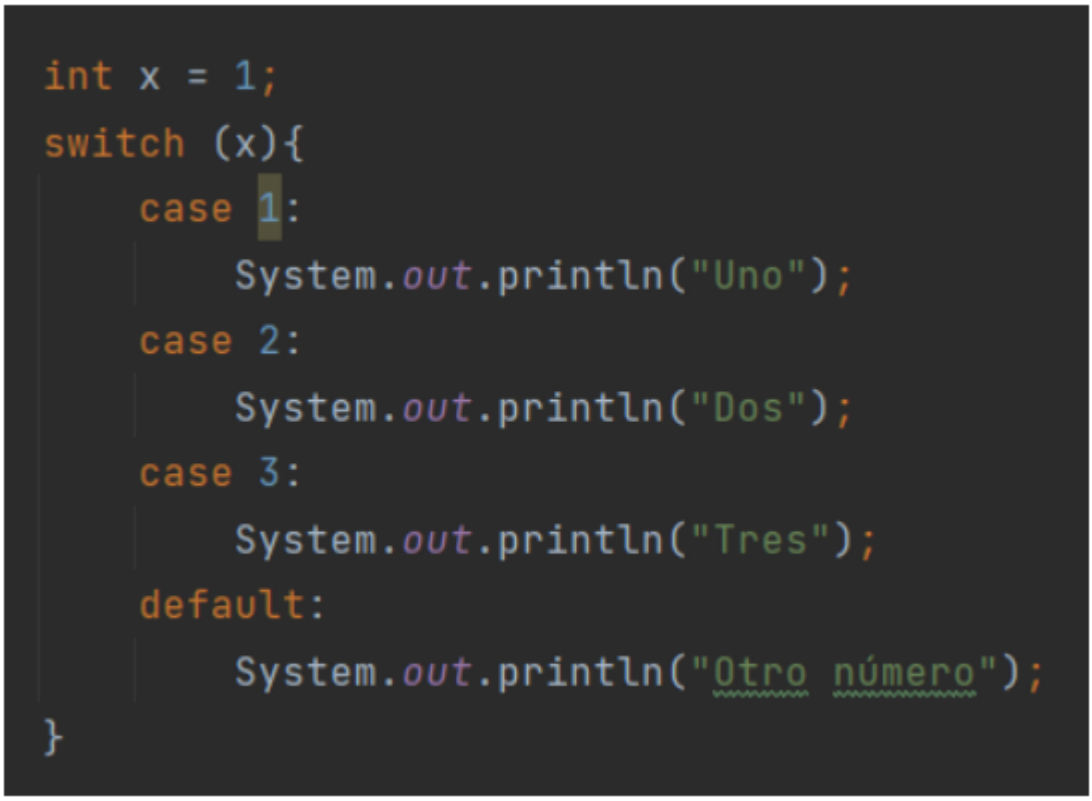
***48.*** ¿Qué palabra clave se utiliza para definir una excepción personalizada en Java?
Para definir una excepción personalizada en Java, se utiliza la palabra clave class. Una vez definida la excepción personalizada, puedes lanzarla utilizando la palabra clave throw y manejarla con un bloque try-catch.

***49.*** ¿Cuál de los siguientes comandos elimina el directorio target antes de iniciar el proceso de construcción?
En el uso en un archivo de construcción de Maven se utiliza el comando mvn clean para eliminar el directorio target

***50.*** ¿Cuál es el comando utilizado para ver el historial de cambios en Git?
Se utiliza el comando git log , que muestra una lista de commits realizados en el repositorio junto con los detalles como el hash del commit, autor, fecha y mensaje del commit.

***51.*** ¿Qué es una expresión lambda en Java 8?
Una expresión lambda en Java 8 es una función anónima que se puede utilizar para proporcionar una implementación concisa de una interfaz funcional, es decir, una interfaz con un único método abstracto. Las expresiones lambda permiten tratar la funcionalidad como un argumento del método, o sea, pueden pasar bloques de código como si fueran datos.

***52.*** ¿Qué muestra el siguiente código fuente por pantalla?



```
int x = 1;
switch (x){
    case 1:
        System.out.println("Uno");
    case 2:
        System.out.println("Dos");
    case 3:
        System.out.println("Tres");
    default:
        System.out.println("Otro número");
}
```

Imprimira todos los mensajes de los case , ya que ninguno contiene un break en cada uno de los case.

***53.*** De los siguientes paquetes, ¿cuáles contienen clases para construir una interfaz gráfica? (Elije todas las que correspondan)

Las principales clases y paquetes que se utilizan para construir interfaces gráficas en Java se encuentran en el paquete java.awt y en el paquete javax.swing.

***54.*** ¿Cuál de las siguientes líneas deben ir en el espacio en blanco para que el código compile? public class News < _____> { }

Indica un parámetro de tipo genérico, donde puede definir clases genéricas que toman uno o más parámetros de tipo. Puede ser T o como identificar la variable,

Se puede utilizar News pero estamos limitando la clase News a que trabaje solo son instancias News o subclases de este y para utilizar Object estamos permitiendo que la clase News tarbaje como cualquier tipo de objeto, ya que es de Object es de la clase base de todas las clases en Java.

***55.*** ¿Qué es un stream en Java 8 y para qué se utiliza?

Stream es una secuencia de elementos que permite realizar operaciones de forma secuencial o paralela, proporcionando una forma más declarativa y funcional de trabajar con colecciones de datos. Realiza operaciones de transformación, filtrado, mapeo, ordenamiento entre otras de una manera eficiente y concisa. Trabaja con volúmenes de datos de manera eficiente, facilita la programación paralela al permitir el procesamiento en paralelo en sistemas multinucleo.

***56.*** ¿Son patrones de diseño de microservicios?

Estos patrones ayudan a diseñar y gestionar sistemas basados en microservicios de manera eficiente, abordando desafíos comunes como comunicación, gestión de datos, despliegue, y resiliencia. Aquí hay algunos de los patrones de diseño más conocidos para microservicios:

1. API Gateway: Actúa como un punto de entrada único para las solicitudes externas, manejando la redirección de solicitudes a los servicios apropiados y proporcionando funcionalidades como autenticación, enrutamiento, y agregación de resultados.
2. Service Discovery: Permite que los microservicios encuentren y se comuniquen entre sí dinámicamente. Puede ser implementado con registros de servicios como Consul, Eureka, o utilizando soluciones basadas en DNS.
3. Circuit Breaker: Previene que los fallos en un servicio se propaguen en cascada a otros servicios, cerrando el circuito cuando un servicio no responde correctamente y proporcionando respuestas de reserva o fallbacks.
4. Saga: Maneja la consistencia de datos y la gestión de transacciones distribuidas mediante la coordinación de una serie de pasos compensatorios para asegurar que el sistema regrese a un estado coherente en caso de fallos.
5. Event Sourcing: Captura todos los cambios de estado como una secuencia de eventos, permitiendo la reconstrucción del estado actual del sistema y facilitando auditorías y análisis históricos.
6. CQRS (Command Query Responsibility Segregation): Separa las operaciones de lectura y escritura del sistema en diferentes modelos, optimizando cada uno para su propósito específico y mejorando el rendimiento y la escalabilidad.
7.Sidecar: Despliega funcionalidades auxiliares, como proxies o componentes de monitoreo, junto a los microservicios principales en el mismo entorno de ejecución, facilitando la implementación de características transversales.
8. Bulkhead: Aísla los recursos utilizados por diferentes servicios o partes de una aplicación, para que un fallo en una parte no afecte a las demás, mejorando la resiliencia del sistema.
9. Strangler Fig: Facilita la migración de un monolito a microservicios reemplazando gradualmente las funcionalidades del monolito con microservicios, mientras se mantienen ambas arquitecturas coexistiendo durante la transición.
10. Backends for Frontends (BFF): Crea backends específicos para diferentes tipos de clientes (web, móvil, etc.), optimizando las interacciones y las experiencias de usuario.

Estos patrones son esenciales para abordar los desafíos inherentes a la construcción y operación de sistemas de microservicios, proporcionando soluciones probadas y prácticas para mejorar la eficiencia, escalabilidad y resiliencia.

***57.*** ¿Qué afirmaciones son verdaderas tanto para las clases abstractas como para las interfaces? (Elije todas las correctas)

a) Ambos pueden contener métodos estáticos.
b) Ambos se pueden ampliar con la clave extend.
c) Ambos pueden contener métodos predeterminados.
d) Ambos heredan de java.lang.Object.

e) Ninguno de los dos puede ser instanciado directamente.
f) Ambos pueden contener variables finales estáticas públicas.
g) Supone que todos los métodos dentro de ellos son abstractos.

Desde Java 8, las interfaces pueden contener métodos estáticos. Las clases abstractas siempre han podido contener métodos estáticos. Las interfaces no se amplían con la clave extends, sino que se implementan usando la clave implements. Sin embargo, las interfaces pueden extender otras interfaces usando extends. Desde Java 8, las interfaces pueden contener métodos predeterminados (default methods). Las clases abstractas pueden contener métodos concretos que actúan como métodos predeterminados. Solo las clases (incluidas las clases abstractas) heredan directamente de java.lang.Object. Las interfaces no heredan de java.lang.Object, aunque los objetos que implementan las interfaces tendrán los métodos de java.lang.Object. No se pueden crear instancias de clases abstractas ni de interfaces directamente. Las interfaces pueden contener variables que son implícitamente public static final. Las clases abstractas también pueden contener variables public static final. No todos los métodos en una clase abstracta o una interfaz son necesariamente abstractos. Las clases abstractas pueden contener métodos concretos, y desde Java 8, las interfaces pueden contener métodos predeterminados y estáticos.

***58.*** ¿Cuál no es un objetivo de Maven?

No es un objetivo de Maven proporcionar un entorno de desarrollo integrado (IDE). Maven se centra en la gestión de dependencias, automatización de la construcción y el ciclo de vida del proyecto, así como en la generación de informes y documentación, pero no es un IDE. Los IDEs como IntelliJ IDEA, Eclipse, y NetBeans pueden integrarse con Maven para utilizar sus capacidades, pero Maven en sí no proporciona un entorno de desarrollo.

***59.*** ¿Si deseas obtener una copia de un repositorio Git existente en un servidor qué comando se utiliza?

El comando git clone no solo descarga el contenido del repositorio, sino que también obtiene toda la información del historial del repositorio, lo que te permite trabajar con todas las ramas y commits del repositorio original.

***60.*** ¿Qué es un repositorio remoto en Git?

Es una versión del proyecto que se almacena en un servidor remoto, fuera de tu entorno de desarrollo local. Este repositorio remoto puede ser accesible a través de la red, permitiendo a múltiples desarrolladores colaborar en el mismo proyecto. Los repositorios remotos son fundamentales para el trabajo en equipo y la colaboración, ya que permiten compartir y sincronizar cambios entre diferentes copias del repositorio.

Un repositorio remoto tiene las características de ubicación en un servidor, acceso a través de redes, colaboración, centralización de datos.

```
public class Helper {
    public static < U extends Exception > void
        printEception(U u){
        System.out.println(u.getMessage());
    }

    public static void main(String[] args) {
        //linea 9
    }
}
```

***61.*** ¿ Cuál de las siguientes instrucciones puede colocarse en la línea 9 para que la clase Helper compile?

a) Helper.printException(new Exception(“B”));
b) Helper. printException(new FileNotFoundException(“A”));
c) Helper.printException(new Exception(“C”));
d) Helper.printException(new NullPointerException (“D”));
e) Helper. printException(new Throwable(“E”));

***62.*** ¿Cuál es la salida al ejecutar el siguiente código?

```
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);
    }
}
```

a) 51tuna
b) 5tuna
c) 5
d) 41
e) 5 tuna
f) 4 1
g) El código no compila

La asignación de la variable String anotherFish = numFish + 1 ; no es correcta ya que esta operación daría como resultado un valor int y la variables que estamos declarando es tipo String por lo tanto causaría un error para que fuera correcto este debería de ser declarado como *String anotherFish= numFish + 1+ “”, o usar String.valueOf o Integer.toString*. Por lo tanto el código no compila.

```
public class StringBuilders {
    1 usage
    public static StringBuilder work(StringBuilder a, StringBuilder b){
        a = new StringBuilder("a");
        b.append("b");
        return a;
    }

    public static void main(String[] args) {
        StringBuilder s1 = new StringBuilder("s1");
        StringBuilder s2 = new StringBuilder("s2");
        StringBuilder s3 = work(s1,s2);
        System.out.println("s1 = " + s1);
        System.out.println("s2 = " + s2);
        System.out.println("s3 = " + s3);
    }
}
```

***63.***

a) s2 = s2
b) s3 = null
c) s1 = s1
d) s3 = a
e) El código no compila
f) s2 = s2b
g) s1 = a

s2 = s2 es correcta aunque seguirán apuntando al mimso objeto y no habría cambio en el valor de referencia, sucedería lo mismo con el caso s1 = s1 , para el caso de s3 = null, no es correcta ya que de acuerdo por la función de work s3 tiene como objeto a y por lo tanto no es igual a null, para s3 = a como observamos anteriormente la función work s3 tiene un objeto de a y por lo tanto a es igual a, para s2 = s2b no tenemos declara una variable de referencia como sb2 por lo que existirá un error de compilación, y para s1=a sabemos que s1 es igual al objeto s1 asi que es incorrecta

***64.*** ¿A qué hace referencia el principio de Liskov?

Este principio se refiere a la relación entre clases base y clases derivadas, y se puede enunciar de la siguiente manera

Si $S$ es una subclase de $T$, entonces los objetos de tipo $T$ en un programa pueden ser reemplazados con objetos de tipo $S$ sin alterar ninguna de las propiedades deseables del programa (corrección, tarea realizada, etc.). Una clase derivada (subclase) debe poder

sustituir a su clase base (superclase) sin que el comportamiento del programa cambie. Esto implica que la subclase debe cumplir con el contrato establecido por la superclase. En otras palabras, las funciones que utilizan punteros o referencias a la clase base deben ser capaces de usar objetos de la clase derivada sin saberlo, y sin que el programa falle.

***65.*** ¿Qué es un “code smell”?

Un "code smell" (olor a código) es un término utilizado en desarrollo de software para describir ciertos síntomas en el código fuente que podrían indicar un problema más profundo. Aunque un code smell no es necesariamente un error o un bug, puede sugerir que el código necesita ser refactorizado para mejorar su legibilidad, mantenibilidad o eficiencia. Los code smells suelen ser señales de diseño pobre o de prácticas de codificación subóptimas. Las principales características de un code smell son el indicador de problemas, mantenibilidad, complejidad y refactorización,

***66.*** ¿Qué significa el acrónimo CRUD en una API REST?

- Create (Crear): Añadir un nuevo recurso. Método HTTP ”POST”
- Read (Leer): Recuperar información sobre un recurso o recursos. Método HTTP “GET”
- Update (Actualizar): Modificar un recurso existente. Método HTTP “PUT/PATCH”
- Delete (Eliminar): Borrar un recurso existente. Método HTTP “DELETE”

***67.*** . ¿Para qué nos sirve utilizar un profile dentro del archivo pom.xml?

Los perfiles en el archivo pom.xml de Maven se utilizan para definir diferentes configuraciones de construcción (build) y entornos de ejecución (runtime). Estos perfiles permiten ajustar el comportamiento de Maven para satisfacer diversas necesidades sin tener que modificar el archivo pom.xml principal cada vez. Sirve para configuraciones de entorno, dependencias específicas, propiedades y configuraciones, activación de perfiles.

***68.*** Con el siguiente código :

```
package my.vehicles;

public class Vehicle {
    public String make;
    protected String model;
    private int year;
    int mileage;
}
```

```
package my.vehicles.cars;

import my.vehicles.*;

public class Car extends Vehicle {
    public Car() {
        //linea 7
    }
}
```

¿Cuál de las siguientes instrucciones pueden colocarse en la línea 7 para que la clase Car compile correctamente? (Seleccione las que apliquen)

a) mileage = 15285;
b) Ninguna de las anteriores.
c) make = “Honda”;
d) year = 2009;
e) model = “Pilot”

mileage no se puede asiganar un obejto fuera de la clase almenos que tengamos un get o set para darle valor ya que esta está en default(package-private), y es menos permisiva, para el caso de make ya que es un atributo público se puede acceder fuera de la clase para asignarle un objeto, y para year es una variable privada y solo puede asigarse valor dentro de la clase, y model al ser protected puede ser utilizada dentro o fuera de la clase.

***69.*** Enumere cuatro interfaces de la API colecciones

a) List, Map, Set, Queue.

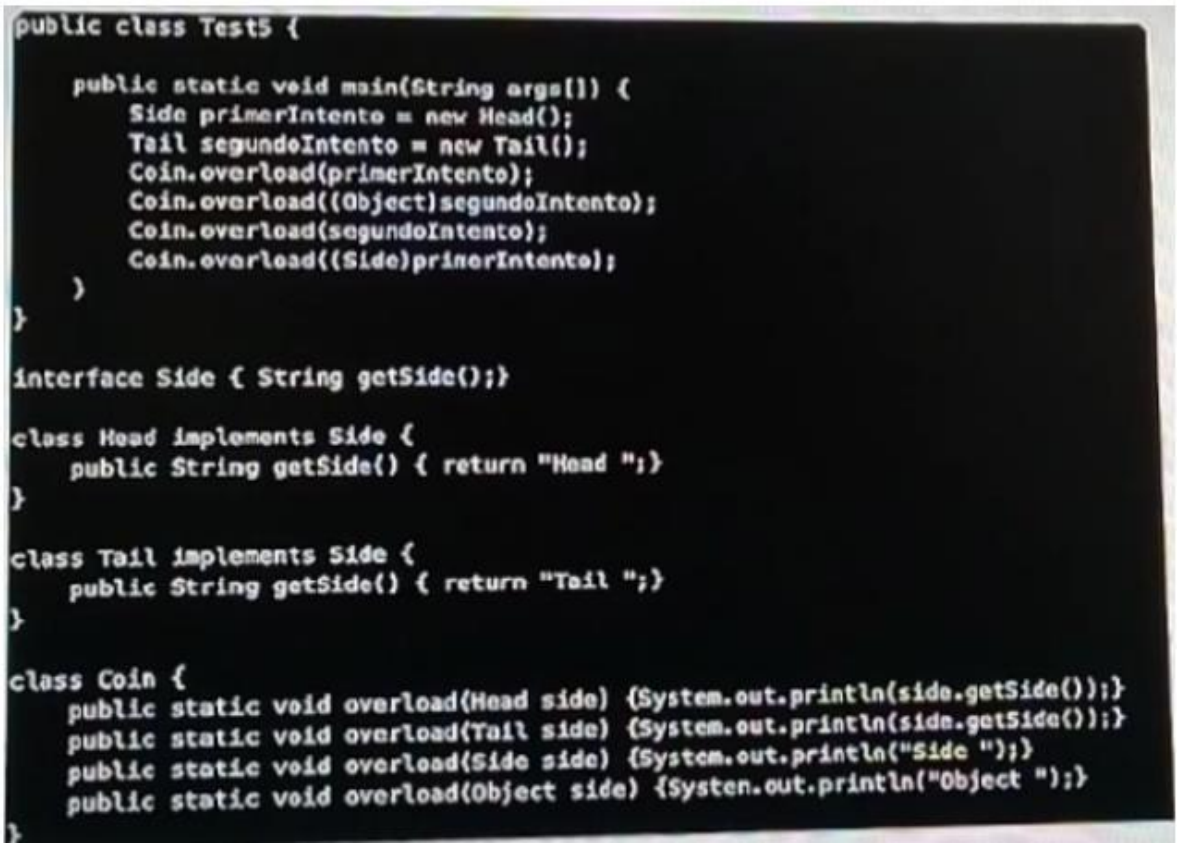
b) ArrayList, Map, Set, Queue.

c) List, HashMap, HashSet, PriorityQueue.

d) List, Map, HashSet, PriorityQueue

la API de colecciones es una parte fundamental de la biblioteca estándar y proporciona varias interfaces que definen diferentes tipos de colecciones y operaciones sobre ellas. Las interfaces principales de la API de colecciones de Java son: Collection <E>, List <E>, Set<E>, SortedSet <E>, NavigableSet<E>, Queue<E>, Deque <E>, Map <K,V>, SortedMap<K,V>, NavigableMap<K,V>.

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



```
public class Test5 {

    public static void main(String args[]) {
        Side primerIntento = new Head();
        Tail segundoIntento = new Tail();
        Coin.overload(primerIntento);
        Coin.overload((Object)segundoIntento);
        Coin.overload(segundoIntento);
        Coin.overload((Side)primerIntento);
    }
}

interface Side { String getSide();}

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

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

class Coin {
    public static void overload(Head side) {System.out.println(side.getSide());}
    public static void overload(Tail side) {System.out.println(side.getSide());}
    public static void overload(Side side) {System.out.println("Side ");}
    public static void overload(Object side) {Systen.out.println("Object ");}
}
```

a) Head Object Tail Side

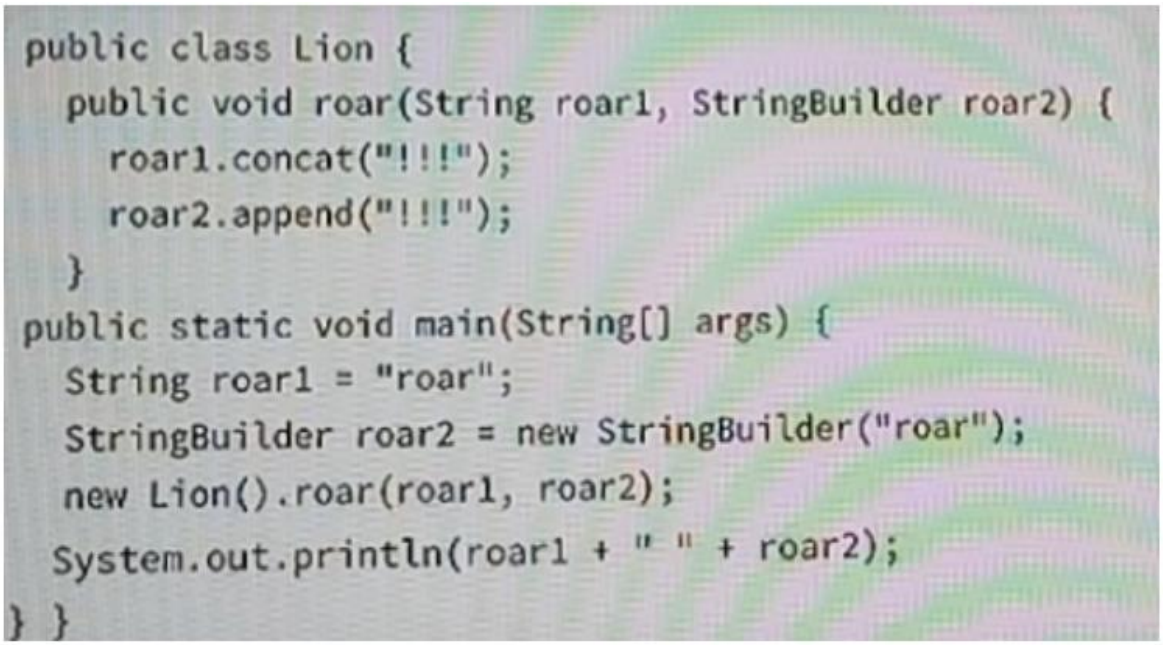
b) No compila

c) Side Object Tail Side

d) Head Head Tail Tail

e) Side Head Tail Side

***71.*** ¿Cuál es la salida al ejecutar el siguiente código?



```
public class Lion {
  public void roar(String roar1, StringBuilder roar2) {
    roar1.concat("!!!");
    roar2.append("!!!");
  }
public static void main(String[] args) {
  String roar1 = "roar";
  StringBuilder roar2 = new StringBuilder("roar");
  new Lion().roar(roar1, roar2);
  System.out.println(roar1 + " " + roar2);
} }
```

a) roar roar!!!
b) roar!!! Roar
c) Se lanza una excepción
d) roar!!! roar!!!
e) roar roar
f) El código no compila

Se imprimirá roar roar!!! Ya que al realizar el constructor de la class lion este para el string roar1 solo concat !!! pero no estamos haciendo que el valor roar1 apunte al objeto con la concatenación asi que mantiene su mismo objeto y para roar2 este esta agregando a la cadena estos valores !!!.

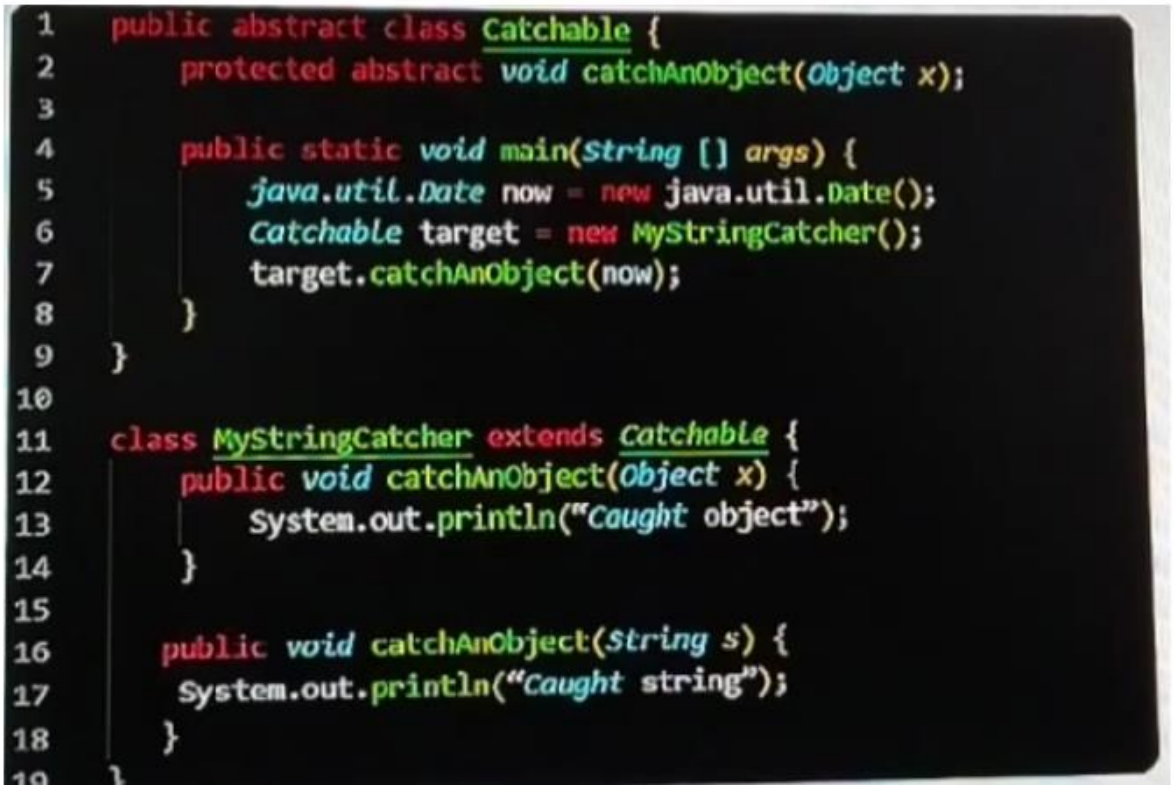
***72.*** ¿Cuál de los siguientes es cierto acerca de una subclase concreta?

a) Una subclase concreta no se puede marcar como final.
b) Una subclase concreta debe implementar todos los métodos definidos en una interfaz heredada.
c) Una subclase concreta debe implementar todos los métodos abstractos heredados.
d) Una subclase concreta puede declararse como abstracta.
e) Los métodos abstractos no pueden ser anulados por una subclase concreta.

Una subclase concreta puede ser marcada como final, lo que significa que no puede ser subclasificada más adelante. Sin embargo, una subclase final no puede ser abstracta. Una subclase concreta no necesita implementar los métodos de una interfaz heredada si ya están implementados en una superclase. Sin embargo, si la clase base no los implementa, la subclase debe hacerlo. Una subclase concreta no puede ser abstracta. Una subclase concreta debe proporcionar implementaciones concretas para todos los métodos heredados, por lo que no puede ser abstracta. Una subclase concreta puede anular (o

sobrescribir) métodos abstractos heredados de su superclase. De hecho, es una de las principales razones por las que una clase se hace concreta: para proporcionar implementaciones concretas de métodos abstractos. Una subclase concreta (también conocida como clase no abstracta o clase final) debe proporcionar implementaciones concretas para todos los métodos abstractos heredados de sus clases base (superclases o interfaces). Si una subclase no abstracta no implementa todos los métodos abstractos heredados, debe ser declarada como abstracta.

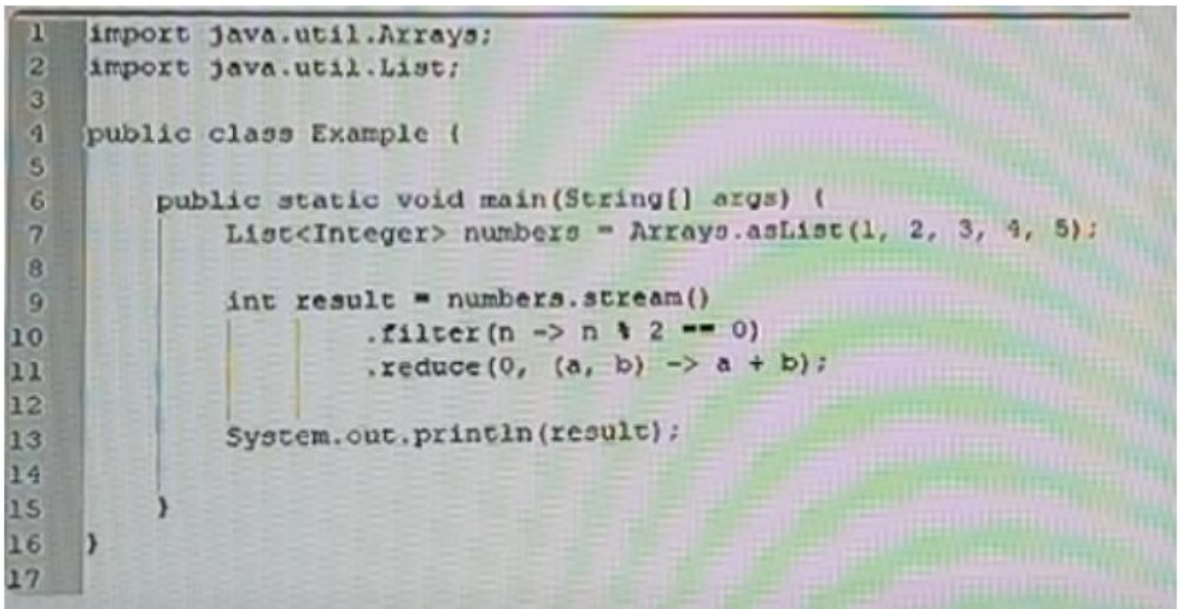
***73.*** ¿Cuál es la salida del siguiente código?



```
1  public abstract class Catchable {
2      protected abstract void catchAnObject(Object x);
3
4      public static void main(String [] args) {
5          java.util.Date now = new java.util.Date();
6          Catchable target = new MyStringCatcher();
7          target.catchAnObject(now);
8      }
9  }
10
11 class MyStringCatcher extends Catchable {
12     public void catchAnObject(Object x) {
13         System.out.println("Caught object");
14     }
15
16     public void catchAnObject(String s) {
17      System.out.println("Caught string");
18     }
19 }
```

a) Error de compilación línea 12
b) Error compilación línea 16
c) Caught string
d) Error compilación línea 2
e) Caught Object

***74.*** Seleccione la respuesta que considere correcta, dado el siguiente bloque de código



```
1  import java.util.Arrays;
2  import java.util.List;
3
4  public class Example {
5
6      public static void main(String[] args) {
7          List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
8
9          int result = numbers.stream()
10                 .filter(n -> n % 2 == 0)
11                 .reduce(0, (a, b) -> a + b);
12
13         System.out.println(result);
14
15     }
16 }
17
```

a) 3
b) 9
c) 14
d) 6

Con la función filter que se realiza al arreglo solo se tomaran encuenta los numero que sean pares , y la función reduce toma los valore para irlos sumando y realiza la suma de 0+2+4 por lo tanto el resultado no da como 6,

***75.*** ¿Qué declaración representa una declaración válida que permitirá la inclusión de clases del paquete java.util?

a) #include java.util.*;
b) #include java.util;
c) import java.util.*;
d) import java.util;

import java.util.*;

Esta declaración importará todas las clases del paquete java.util, lo que te permitirá utilizarlas en tu código sin tener que importar cada clase de forma individual. La expresión * es un comodín que indica que se deben importar todas las clases del paquete.

***76.*** ¿Qué es la cobertura de código?

La cobertura de código es una métrica utilizada en el desarrollo de software para medir la cantidad de código fuente que ha sido ejecutada durante la ejecución de un conjunto de pruebas. Se utiliza para evaluar la efectividad de las pruebas automatizadas al identificar qué porcentaje del código fuente ha sido ejecutado por las pruebas.

***77.*** ¿Cuál es el formato correcto para hacer un commit en Git?

El formato correcto para hacer un commit en Git sigue una convención comúnmente conocida como "mensaje de commit con formato" o "formato de mensaje de commit". Este formato generalmente consta de tres secciones: título, cuerpo y pie de página.

***78.*** ¿Qué es el patrón de diseño Singleton y cómo se implementa en Java 8?

El patrón de diseño Singleton es un patrón de creación que garantiza que una clase tenga una única instancia y proporciona un punto de acceso global a esa instancia. Esto es útil cuando se necesita exactamente una instancia de una clase para coordinar acciones en todo el sistema. Sus características principales son una única instancia, acceso global, constructor privado.

***79.*** En los verbos REST ¿Cuál es la diferencia en el uso de PATCH y PUT?

La diferencia principal entre PUT y PATCH radica en la cantidad de datos que se envían y se procesan durante la actualización del recurso. PUT se utiliza para reemplazar completamente un recurso, mientras que PATCH se utiliza para realizar cambios parciales en un recurso existente.

***80.*** ¿Cuál es la diferencia entre las anotaciones: @RestController, @Component, @Service y @Repository?

La anotación @RestController se utiliza para marcar clases que implementan controladores en una aplicación Spring MVC.

La anotación @Component es una anotación genérica utilizada para marcar cualquier clase como un componente de Spring.

La anotación @Service se utiliza para marcar clases de servicio en una aplicación Spring.

La anotación @Repository se utiliza para marcar clases que acceden y gestionan datos en una aplicación Spring.

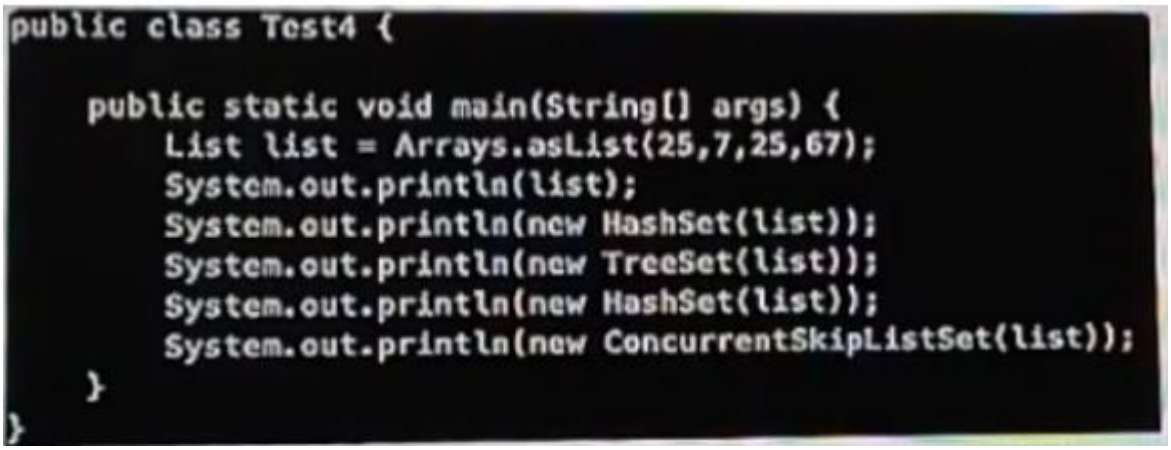
***81.*** ¿Cuál es una buena práctica al escribir pruebas unitarias?

Al escribir pruebas unitarias, es importante seguir varias buenas prácticas para garantizar que las pruebas sean efectivas, mantenibles y escalables. Algunas de estas prácticas incluyen escribir pruebas independientes y aisladas, utilizar nombre descriptivos para las pruebas, seguir el principio de FIRST(Fast-Rápidas, Independientes, Repetibles,Self-Validating-Autovalidantes y Timely-Oportuna), probar casos limite y casos de error, utilizar aserciones claras y específicas, mantener las pruebas actualizadas, automatizar las pruebas y ejecutarlas con frecuencia y utilizar herramientas de prueba adecuadas.

***82.*** ¿Cuál es la ventaja de usar APIs REST sobre otros tipos de servicios web?

Las APIs REST ofrecen una serie de ventajas, incluida su simplicidad, flexibilidad, escalabilidad y compatibilidad con múltiples plataformas, visibilidad y descubrimiento, cacheabilidad así como soporte para CRUD, lo que las convierte en una opción popular para el desarrollo de servicios web en entornos modernos y distribuidos.

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



```
public class Test4 {

    public static void main(String[] args) {
        List list = Arrays.asList(25,7,25,67);
        System.out.println(list);
        System.out.println(new HashSet(list));
        System.out.println(new TreeSet(list));
        System.out.println(new HashSet(list));
        System.out.println(new ConcurrentSkipListSet(list));
    }
}
```

a) No compila

b) [25, 7, 25, 67] [67, 7, 25] [7, 25, 67] [67, 7, 25] [7, 25, 67]

c) [25, 7, 67] [67, 7, 25] [7, 25, 67] [67,7, 25] [7, 25, 67]

d) [67, 7, 25] [67, 7, 25] [67, 7, 25] [67, 7, 25] [67, 7, 25]

e) [25, 7, 25, 67] [7, 25, 67] [67, 7, 25] [7, 25, 67] [67, 7, 25]

La primera impresión en pantalla nos motrara la List de acuerdo a como esta almacaenada, depues la función HashSet nos imprimirá en pantalla valores del list desordenados y no contendrá valores repetidos, y con la función TreeSet imprimirá los valores ordenados de forma ascendente y tampoco permite valores repetidos, después volverá a imprimir el resultado con HashSet y finalmente imprimira los valores ordenado de forma ascendente sin repetir valores.

***84.*** ¿Cuáles son los 4 pilares de la programación orientada a objetos?

a) Polimorfismo, Coerción, Herencia y Encapsulamiento.

b) Encapsulamiento, Coerción, Polimorfismo y Abstracción.

c) Polimorfismo, Herencia, Encapsulamiento y Sincronía.

d) Polimorfismo, Abstracción, Herencia y Encapsulamiento

***85.*** ¿Qué utilidad de línea de comandos basada en MS Windows le permitirá ejecutar el intérprete de Java sin abrir la ventana de la consola?

a) jconsole

b) javaw

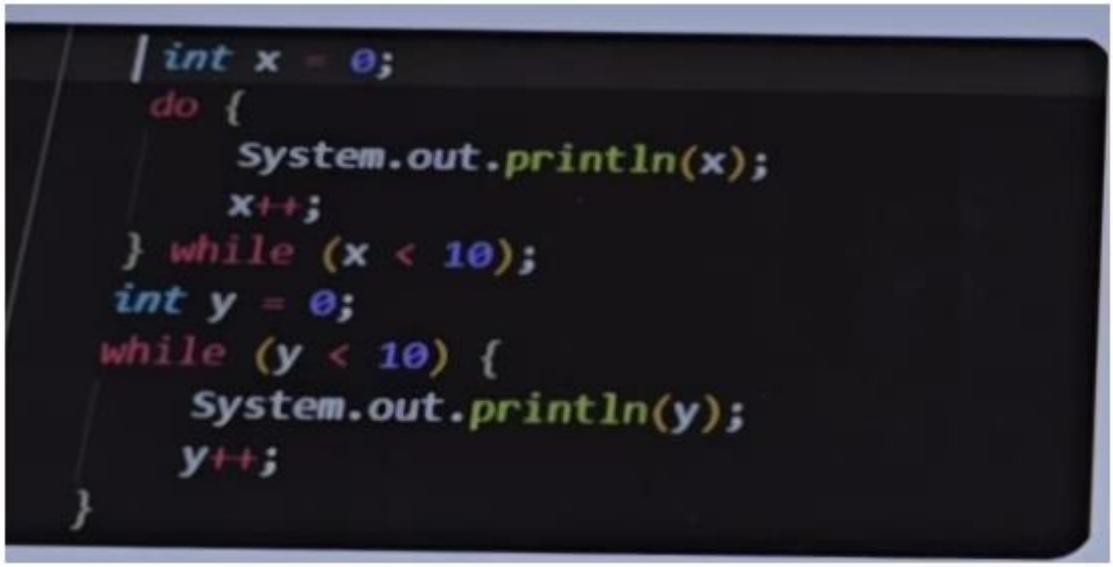
c) interpw

d) java –wo

***86.*** ¿Qué es un endpoint en una API REST?

Un "endpoint" es un término que se refiere a un punto final específico de una API que representa un recurso o una operación. En otras palabras, un endpoint es una URL (Uniform Resource Locator) específica que se puede utilizar para acceder a un recurso o realizar una operación específica en el servidor.

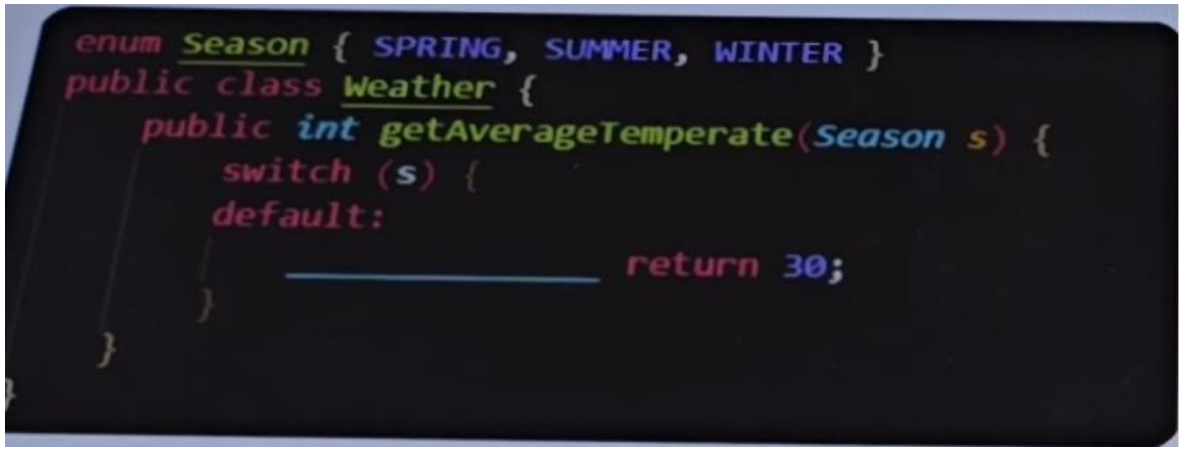
**87.** ¿Cuál es el valor de x e y al final el programa?



```
int x = 0;
do {
    System.out.println(x);
    x++;
} while (x < 10);
int y = 0;
while (y < 10) {
    System.out.println(y);
    y++;
}
```

a) X=9 y=10 b) X=10 y=9 c) X=10 y=10 d) X=9 y=9

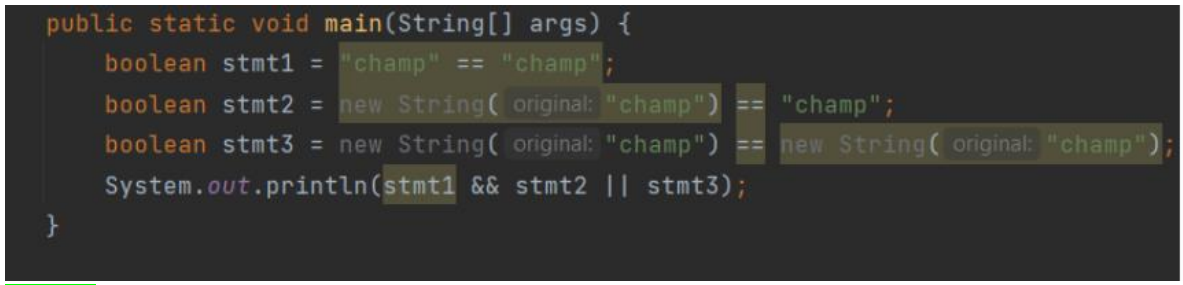
**88.** ¿Dado el siguiente enum y clase cuál es la opción que puede ir en el espacio en blanco para que el código compile?



```
enum Season { SPRING, SUMMER, WINTER }
public class Weather {
    public int getAverageTemperate(Season s) {
        switch (s) {
        default:
            _____ return 30;
        }
    }
}
```

a) Ninguno de los anteriores b) case SUMMER -> c) case Season.Winter: d) case FALL: e) case Winter, Spring: f) case SUMMER | WINTER:

**89.** ¿Cuál es el resultado de compilar y ejecutar el siguiente programa?



```
public static void main(String[] args) {
    boolean stmt1 = "champ" == "champ";
    boolean stmt2 = new String( original: "champ") == "champ";
    boolean stmt3 = new String( original: "champ") == new String( original: "champ");
    System.out.println(stmt1 && stmt2 || stmt3);
}
```

a) False b) no se produce salida c) true d) error de compilación

**90.** ¿Cómo se manejan las excepciones en Java?

En Java, las excepciones se manejan mediante el uso de bloques try-catch, try-catch-finally o mediante la propagación de excepciones a través de métodos.

***91.*** ¿Qué clase del paquete java.io permite leer y escribir archivos en ubicaciones específicas dentro de un archivo?

a) File
b) filename filter
c) file descriptor
d) RandomAccessFile

***92.*** Todas las siguientes definiciones de clases my School classroom y my City School ¿qué números de línea en el método main generan un error de compilación? (Elija todas las opciones correctas)

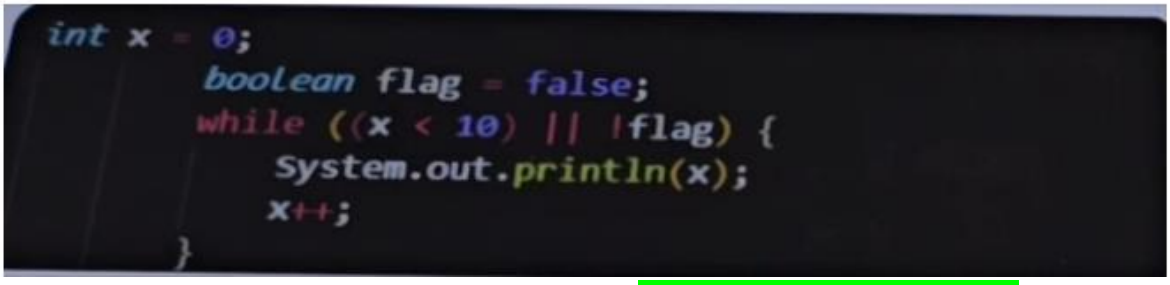
```
1: package my.school;
2: public class Classroom {
3:   private int roomNumber;
4:   protected String teacherName;
5:   static int globalKey = 54321;
6:   public int floor = 3;
7:   Classroom(int r, String t) {
8:     roomNumber = r;
9:     teacherName = t; } }

1: package my.city;
2: import my.school.*;
3: public class School {
4:   public static void main(String[] args) {
5:     System.out.println(Classroom.globalKey);
6:     Classroom room = new Classroom(101, ""Mrs. Anderson");
7:     System.out.println(room.roomNumber);
8:     System.out.println(room.floor);
9:     System.out.println(room.teacherName); } }
```

a) Ninguna, el código compila bien b) línea 6 c) línea 9 d) línea 7 e) línea 8 f) línea 5

***93.*** ¿Qué es una expresión lambda en Java?

Una expresión lambda en Java es una forma concisa de representar una función anónima, es decir, una función sin nombre que se puede pasar como argumento a métodos o asignar a variables. Las expresiones lambda se introdujeron en Java 8 como parte de las nuevas características del lenguaje para admitir la programación funcional.

***94.*** ¿Qué hace el siguiente código fuente?



```
int x = 0;
boolean flag = false;
while ((x < 10) || !flag) {
    System.out.println(x);
    x++;
}
```

a) Muestra los números del 1 al 10 b) muestra un 10 c) se queda en un bucle infinito ! Es correcto d) muestra los números del 0 al 9

Ya que por la compuerta || siempre será true aunque uno sea false ,esto ocasionara un ciclo infinito , y para while (true) siempre serán bucles infinitos.

***95.*** ¿Qué son las anotaciones en java?

Las anotaciones son metadatos que se pueden agregar al código fuente para proporcionar información adicional sobre el programa. Las anotaciones no afectan directamente al comportamiento del programa en tiempo de ejecución, pero pueden ser procesadas por herramientas y marcos de trabajo durante la compilación o la ejecución para realizar acciones específicas.

Las anotaciones se definen mediante el símbolo @ seguido del nombre de la anotación, opcionalmente seguido de parámetros. Pueden aplicarse a clases, métodos, variables, parámetros y otros elementos del código Java.

***96.*** ¿Qué son las expresiones regulares en Java?

En Java, las expresiones regulares (también conocidas como regex) son secuencias de caracteres que definen un patrón de búsqueda. Estos patrones se utilizan para buscar y manipular cadenas de texto de manera eficiente. Las expresiones regulares permiten realizar tareas como búsqueda, validación, extracción y reemplazo de texto basado en un patrón predefinido.

***97.*** Las expresiones regulares en Java se implementan mediante la clase java.util.regex.Pattern y java.util.regex.Matcher.¿Qué es una anotación en Spring?

En el contexto de Spring Framework, una anotación es una etiqueta que se coloca sobre una clase, método o campo para proporcionar metadatos adicionales y configuración al contenedor de Spring. Las anotaciones en Spring son una forma de configurar y administrar componentes y funcionalidades específicas del framework de manera declarativa, lo que facilita el desarrollo de aplicaciones basadas en Spring.

Las anotaciones en Spring se utilizan para diversas tareas, como la configuración de beans, la inyección de dependencias, la definición de controladores web, la administración de transacciones, la seguridad, entre otras.

***98.*** ¿Cuál es la salida?

```java
import java.util.ArrayList;

public class OtherExample {
    public static void main(String[] args) {
        var list = new ArrayList<String>();
        list.add("Austin");
        list.add("Boston");
        list.add("San Francisco");
        var c :long = list.stream()
                .filter(a -> a.length() > 10) //linea x
                .count();
        System.out.println(c + " " + list.size());
    }
}
```

a) Ninguna de las anteriores b) 1 3 c) 1 1 d) El código no compila en la línea x e) 2 3

Realiza y verifica que la longitud del objeto sea menor a 10 y si es el caso cuenta cuantas veces aparece en la lista por lo tanto imprimirá 1 que son las veces que aparece ese valor en la lista y 3 es la longitud de tamaño de elementos que tiene la lista.

***99.*** ¿Cuál es la salida de la siguiente aplicación?

```java
interface Speak { default int talk(){ return 7;} }
interface Sing { default int talk(){ return 5; }}

public class Performance implements Speak, Sing {
    public int talk(String... x) {
        return x.length;
    }

    public static void main(String[] notes) {
        System.out.println(new Performance().talk());
    }
}
```

a) 5 b) 7 c) El código compila sin problemas la salida no se puede determinar hasta el tiempo de ejecución. d) Ninguna de las anteriores. e) El código no compila.

***100.*** ¿Qué conjunto de modificadores, cuando son agregados a un método default dentro de una interfaz, evitan que sea sobrescrito por la clase que lo implementa?

Agregar el modificador final a un método default de una interfaz evita que sea sobrescrito por las clases que implementan dicha interfaz. Esto asegura que el comportamiento del método default sea consistente en todas las implementaciones de la interfaz y no pueda ser modificado por las clases implementadoras.

***101.*** ¿Qué tipo de excepción se produce cuando se intenta realizar una operación incompatible con el tipo de datos en Java?

a) ArrayIndexOutOfBoundsException
b) ArithmeticException
c) IllegalArgumntException
d) ClassCastException

***102.*** ¿Qué es un starter? un "starter" (iniciador) es una dependencia que proporciona una configuración predefinida y un conjunto de características relacionadas para un tipo específico de aplicación. Los starters en Spring Boot están diseñados para simplificar el proceso de configuración y desarrollo de aplicaciones al proporcionar un conjunto coherente de dependencias y configuraciones comunes para escenarios de desarrollo comunes.

Los starters se crean como módulos independientes que incluyen un conjunto de dependencias relacionadas y configuración predeterminada para un propósito específico, como el desarrollo de aplicaciones web, acceso a bases de datos, seguridad, pruebas, etc. Cada starter está diseñado para proporcionar todo lo necesario para comenzar a desarrollar un tipo específico de aplicación con Spring Boot.

***103.*** Selecciona la respuesta correcta con respecto al resultado del siguiente bloque de código.

```
public class Test2 extends Thread{
    public static void main(String[] args) {
        protected Thread t = new Thread(new Test2());
        Thread t2 = new Thread(new Test2());

        t.start();
        t2.start();
    }

    public void main(){
        for (int i = 0;i<2;i++)
            System.out.println(Thread.currentThread().getName()+" ");
    }
}
```