Preguntas

1. ¿Cuál es el comando utilizado para deshacer el último commit en git?

git reset

2. ¿Cuál es la diferencia entre una clase abstracta y una interfaz en java 8?

Una clase abstracta en Java es una clase que no puede ser instanciada directamente y puede contener tanto métodos abstractos como métodos concretos, permitiendo niveles de acceso variados y pudiendo tener constructores y campos de instancia.

Una interfaz en Java es una colección de métodos abstractos y, desde Java 8, puede incluir métodos default y static con implementación. Todos los métodos en una interfaz son implícitamente públicos y sus campos son public static final.

Una clase puede heredar de una sola clase abstracta, mientras que puede implementar múltiples interfaces, proporcionando una forma de herencia múltiple.

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)

   a) Conditional
   b) Assertion
   c) Assignment
   d) Iteration

Condicional: Dentro del bucle de iteración, se necesita una declaración condicional (if) para comprobar si el elemento actual es una moneda de 5 centavos.

Asignación: Cuando encuentre una moneda de 5 centavos, debe incrementar un contador. Esto se hace con una declaración de asignación, donde se actualiza el valor del contador (contador++)

Iteración: Necesita iterar a través de cada elemento en la matriz para verificar cada moneda. Esto se hace generalmente con un bucle for.

No necesita Assertion, ya que las afirmaciones (assert) se utilizan generalmente para pruebas y verificación de condiciones que se esperan que sean verdaderas, y no para la lógica de contar elementos.

4. ¿Qué es un archivo JAR en java?

Un archivo JAR (Java ARchive) es un archivo comprimido en formato ZIP que contiene archivos de clase Java, recursos y metadatos. Se utiliza para distribuir y desplegar aplicaciones y bibliotecas Java en un solo archivo. Incluye un archivo de manifiesto que puede especificar la clase principal para ejecutar el JAR. Los archivos JAR facilitan la portabilidad y organización de aplicaciones Java.

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

La sobrecarga de métodos en Java es una característica que permite definir múltiples métodos con el mismo nombre en una clase, pero con diferentes listas de parámetros. Los métodos sobrecargados deben diferir en el número, tipo, o el orden de sus parámetros. La sobrecarga permite a los métodos realizar operaciones similares con diferentes tipos de datos o cantidades de argumentos.

6. ¿Cuál es la diferencia entre un ArrayList y un LinkedList en Java?

En resumen, un ArrayList en Java utiliza una matriz dinámica para almacenar elementos, ofreciendo un acceso rápido por índice pero una inserción y eliminación más lenta en comparación con LinkedList, que utiliza una lista doblemente enlazada. Mientras ArrayList es más eficiente para acceso aleatorio y modificaciones frecuentes, LinkedList es más adecuado para inserciones y eliminaciones frecuentes, especialmente en el medio de la lista.

La elección entre ambos depende de los requisitos de la aplicación en términos de acceso, inserción, eliminación y el tamaño esperado de la lista.

7. ¿Cuándo se debe usar un bloque finally en una declaración try regular (no un try with resources)?

Se debe usar un bloque finally en una declaración try regular en Java cuando necesitas asegurarte de que ciertas operaciones se realicen, independientemente de si ocurre una excepción o no dentro del bloque try. El bloque finally se ejecuta siempre, ya sea que se lance una excepción dentro del bloque try o no.

8. ¿Cuál es el propósito principal de los test unitarios?

Los test unitarios son herramientas esenciales en el desarrollo de software, destinadas a verificar que unidades individuales de código, como métodos o funciones, funcionen correctamente de manera aislada. Su propósito principal es asegurar la calidad del código al detectar y corregir errores tempranos, facilitar la refactorización, documentar el comportamiento del código, promover la modularidad y la reutilización del mismo, y automatizar la verificación del código. Al proporcionar una red de seguridad durante el desarrollo, los test unitarios contribuyen significativamente a la calidad, la estabilidad y la mantenibilidad del software.

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

```java
public class pregunta9 {

    public static void main(String[] args) {
        String cad1 = "hola";
        String cad2 = new String("hola");
        String cad3 = "hola";
        if (cad1 == cad2) {
            System.out.println("cad1 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 es diferente a cad3");
        }
    }
}
```

cad1 diferente a cad2

cad1 es igual a cad3

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

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

public class pregunta10 extends Mammal {
    public pregunta10() { Implicit super constructor Mammal() is undefined. Must explicity
invoke another constructor
        System.out.println("Platypus");
    }

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

El código no compila en la línea 8

11. ¿Cómo se manejan las excepciones en java?

Las excepciones se manejan mediante el uso de bloques try, catch, finally y, en versiones más recientes, throw y throws.

- try: En este bloque se coloca el código que puede generar una excepción.
- catch: Dentro de este bloque se manejan las excepciones que son lanzadas en el bloque try. Puedes tener múltiples bloques catch para manejar diferentes tipos de excepciones.
- finally: Este bloque se ejecuta siempre, ya sea que ocurra una excepción o no. Se utiliza típicamente para realizar tareas de limpieza, como cerrar archivos o liberar recursos, independientemente de si ocurre una excepción.
- throw: Se utiliza para lanzar manualmente una excepción en un bloque de código.
- throws: Se utiliza en la firma de un método para indicar que el método puede lanzar una excepción y delegar su manejo al llamador.

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

Sí, la anotación @Ignore en JUnit se utiliza para omitir la ejecución de un test específico. Cuando se marca un método de prueba con @Ignore, se indica al framework de pruebas que ignore ese método y no lo ejecute durante la ejecución de la suite de pruebas. Esto puede ser útil temporalmente cuando un test no está listo para ser ejecutado o cuando se desea evitar la ejecución de un test específico por alguna razón, como problemas de dependencias externas o cambios en el entorno de pruebas

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

```java
public class pregunta13 {
    static {
        int x = 3;
    }
    static int x;

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

-1

14. ¿Qué es un operador de short circuit?

Los operadores de cortocircuito en Java, representados por && (AND lógico) y || (OR lógico), permiten evaluar solo una parte de una expresión lógica, dependiendo del resultado de la otra parte. Si el operando de la izquierda determina el resultado de la expresión, la evaluación se detiene sin evaluar el operando de la derecha. Esto mejora el rendimiento y evita errores, pero en ciertas situaciones puede ser necesario evaluar ambas partes de la expresión independientemente del valor de la primera.

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

El patrón de diseño DAO (Data Access Object) en Java se utiliza para abstraer el acceso a la capa de datos de una aplicación, separando la lógica de acceso a datos de la lógica de negocio. Se implementa mediante una interfaz DAO que define operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre objetos de dominio, una clase DAO concreta que proporciona la implementación real de estas operaciones, y una clase de dominio que representa los objetos de negocio. Esta separación mejora la modularidad y facilita el mantenimiento del código.

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

Un endpoint en una API REST es un punto de acceso específico a un recurso o conjunto de recursos en el servidor. Representa una URL que el cliente puede solicitar para interactuar con la API y realizar operaciones sobre los datos. Cada endpoint está asociado con una operación HTTP específica, como GET, POST, PUT, DELETE y otros métodos HTTP. Los endpoints en una API REST suelen seguir una convención de nomenclatura que refleja la estructura de los recursos y las operaciones que se pueden realizar sobre ellos, lo que facilita la comprensión y el uso de la API.

17. ¿Qué hace el siguiente programa?

```java
public class pregunta17 {
    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;
            }
        }
    }
}
```

Verifica si una palabra es un palíndromo.

18. ¿Cuál de las siguientes opciones son verdaderas? (elija todas las correctas)

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.

a) Verdadero. Java se diseñó principalmente como un lenguaje orientado a objetos.

b) Verdadero. Java utiliza la máquina virtual Java (JVM), lo que permite que el código compilado sea ejecutable en diferentes plataformas que tengan una JVM instalada, incluidas Windows y Linux.

c) Falso. Java no permite la sobrecarga de operadores como lo hacen algunos otros lenguajes como C++.

d) Falso. Java no es un lenguaje de programación funcional en el sentido puro. Aunque introduce elementos de programación funcional con las expresiones lambda en versiones más recientes, no se considera un lenguaje puramente funcional como Haskell o Lisp.

e) Falso. Java se centra en el paradigma de programación orientada a objetos, aunque admite procedimientos y funciones a través de métodos.

f) Falso. Java no tiene punteros explícitos a ubicaciones específicas en la memoria como lo hacen algunos otros lenguajes de programación de más bajo nivel como C o C++. Java gestiona la memoria de manera automática a través del recolector de basura y utiliza referencias para acceder a los objetos en memoria.

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

Maven es una herramienta de gestión de proyectos de software utilizada principalmente en el desarrollo de aplicaciones Java. Se utiliza para gestionar dependencias, automatizar la construcción del proyecto, definir ciclos de vida y configuraciones del proyecto.

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.

a) Verdadero. javac es el compilador de Java que toma un archivo fuente en formato .java y lo compila en un archivo de bytecode en formato .class.

b) Falso. Java no toma el nombre del archivo .bytecode como parámetro al ejecutar un programa.

c) Verdadero. javac compila un archivo fuente .java en un archivo de clase .class.

d) Verdadero. Cuando se ejecuta un programa Java, se especifica el nombre de la clase principal como parámetro, no el nombre del archivo .java.

e) Verdadero. Al ejecutar un programa Java, se especifica el nombre de la clase principal, que generalmente corresponde al nombre del archivo .class que contiene esa clase.

f) Falso. javac compila archivos .java en archivos .class, no al revés. No convierte archivos .class en archivos .java.

21. ¿Qué es Git y cuáles son algunos de sus comandos básicos?

Git es un sistema de control de versiones que rastrea cambios en el código. Algunos comandos básicos incluyen:

- git init: Inicia un nuevo repositorio.
- git clone: Clona un repositorio existente.
- git add: Añade cambios para ser incluidos en el próximo commit.
- git commit: Guarda los cambios en el repositorio.
- git push: Sube cambios locales al repositorio remoto.
- git pull: Obtiene cambios del repositorio remoto y los fusiona con el local.
- git status: Muestra el estado actual del repositorio.
- git branch: Maneja ramas.
- git checkout: Cambia entre ramas o restaura archivos.
- git merge: Fusiona cambios de una rama a otra.

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;
```

En esta opción, se está tratando de asignar una suma de un entero (variableA) y un flotante (variableB) a una variable de tipo entero (variableC), lo cual no es permitido en Java sin una conversión explícita.

23. ¿Qué escenario es el mejor uso de una excepción?
    a) La computadora se incendió.
    b) No sabe cómo codificar un método.
    c) No se encuentra un elemento al buscar en una lista.
    d) Se pasa un parámetro inesperado a un método.
    e) Quiere recorrer una lista.

El mejor uso de una excepción es cuando ocurre un escenario excepcional que interrumpe el flujo normal de ejecución del programa y no puede ser manejado directamente en el código.

a) Este escenario está más allá del ámbito del programa en sí.

b) Esto es más un problema de desarrollo o conocimiento del programador que una situación excepcional durante la ejecución del programa.

c) Cuando intentas acceder a un elemento que no existe en una lista, esto puede considerarse un escenario excepcional y, por lo tanto, lanzar una excepción como `IndexOutOfBoundsException` o `NoSuchElementException`.

d) Si un método recibe un parámetro inesperado, esto es más un error de lógica o de diseño del programa que una situación excepcional que justifique el uso de una excepción. Este tipo de errores generalmente se manejan con validación de datos y pruebas adecuadas.

e) Recorrer una lista es una operación común en la programación y no representa una situación excepcional. Es una parte normal del flujo de ejecución del programa y no justifica el uso de una excepción.

24. ¿Qué es un bean en Spring?

En el contexto de Spring, un "bean" es un objeto gestionado por el contenedor de Spring, configurado y administrado por Spring para su uso en la aplicación. Los beans son las piezas fundamentales de la aplicación en Spring, y pueden ser configurados de varias formas, como mediante XML de Spring, anotaciones o mediante configuración programática.

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

```
public class pregunta25 extends Concreate {
    pregunta25() {
        System.out.println("t ");
    }

    public static void main(String[] args) {
        new pregunta25();
    }
}

class Concreate extends Send {
    Concreate() {
        System.out.println("c ");
    }

    private Concreate(String s) {

    }
}
```

```
abstract class Send {
    Send() {
        System.out.println("s ");
    }
}
```
s
c
t

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

a) Es verdadera porque es una característica clave del polimorfismo en Java. Cuando un método toma como parámetro una superclase, puede recibir instancias de esa superclase o de cualquier subclase de ella. Esto se debe a que las subclases heredan los métodos y comportamientos de su superclase, por lo que pueden ser tratadas de manera polimórfica como su clase base. Por lo tanto, si un método toma una superclase como parámetro, cualquier instancia de esa superclase o de sus subclases puede ser pasada como argumento al método.

b) Esta afirmación es falsa porque aunque es cierto que un parámetro de tipo Object puede tomar referencias a cualquier objeto en Java debido a la jerarquía de clases de Java, esto no garantiza que cualquier objeto pueda ser pasado como argumento en tiempo de compilación sin necesidad de conversión explícita. Si bien cualquier objeto puede ser convertido implícitamente a Object, no todos los objetos pueden ser convertidos implícitamente de Object a su tipo específico en tiempo de compilación.

c) Esta afirmación es verdadera porque en Java puedes convertir una referencia a un objeto a una subclase de ese objeto utilizando una conversión explícita, también conocida como "casting". Por ejemplo, si tienes un objeto de una superclase y deseas tratarlo como un objeto de una subclase específica, puedes hacerlo utilizando un casting explícito. Sin embargo, debes tener cuidado al hacer esto para evitar ClassCastException si el objeto real no es una instancia de la subclase.

d) Esta afirmación es falsa porque las excepciones de conversión, como las excepciones de clase (ClassCastException), ocurren en tiempo de ejecución, no en tiempo de compilación. Si intentas realizar una conversión de un objeto a un tipo incompatible, como convertir un objeto de una clase a una subclase incompatible, la excepción se lanzará en tiempo de ejecución cuando se produzca el intento de conversión, no en tiempo de compilación. Por lo tanto, no todas las excepciones de conversión se pueden detectar en tiempo de compilación.

e) Esta afirmación es verdadera debido al concepto de polimorfismo en Java. Cuando una clase hereda de otra clase, puede anular (override) los métodos de la clase base. Esto significa que si una subclase tiene un método con la misma firma (nombre y parámetros) que un método en la superclase, se llamará al método de la subclase cuando se trabaje con una instancia de la subclase, incluso si el objeto se declara como una instancia de la superclase. Esto se conoce como enlace dinámico, y asegura que se ejecute el método adecuado en tiempo de ejecución, basándose en el tipo real del objeto, no en el tipo declarado del objeto.

27. ¿Son patrones de diseño de software estructural?

Los patrones de diseño de software estructural son un tipo de patrones de diseño que se centran en la composición de clases y objetos para formar estructuras más grandes y complejas. Estos patrones se utilizan para encontrar formas eficientes de organizar y manejar las relaciones entre los objetos y las clases en una aplicación. Algunos ejemplos de patrones de diseño estructurales incluyen

- Adapter (Adaptador): Permite que interfaces incompatibles trabajen juntas.
- Bridge (Puente): Separa una abstracción de su implementación para que puedan variar independientemente.
- Composite (Compuesto): Permite tratar objetos individuales y composiciones de objetos de manera uniforme.
- Decorator (Decorador): Agrega funcionalidades a objetos dinámicamente.
- Facade (Fachada): Proporciona una interfaz unificada para un conjunto de interfaces en un subsistema.
- Proxy (Proxy): Proporciona un representante o sustituto de otro objeto para controlar el acceso a él.

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

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

public class pregunta28 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
        double result = numbers.stream()
                .mapToInt(n -> n)
                .average()
                .orElse(0);
        System.out.println(result);
    }
}
```

3.0

29. ¿Qué son las pruebas de integración?

Las pruebas de integración son pruebas en el desarrollo de software que aseguran que los diferentes componentes o módulos de un sistema funcionen correctamente juntos. Evalúan la interacción entre los elementos para detectar problemas de compatibilidad y flujo de datos.

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

git push

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

```
class ClassX {
    static int y = 2;

    ClassX(int x) {
        this();
        y = y * 2;
    }

    ClassX() {
        y++;
    }
}

public class pregunta31 extends ClassX {
    pregunta31() {
        super(y);
        y = y + 3;
    }

    public static void main(String[] args) {
        new pregunta31();
        System.out.println(y);
    }
}
```

9

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

git branch <nombre-de-la-rama>

Este comando crea una nueva rama con el nombre especificado, pero no cambia a esa rama. Para eso:

git checkout -b <nombre-de-la-rama>

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

```
public class pregunta33 {
    private int ISBN;
    private String title, author;
    private int pageCount;

    public int hashCode() {
        return ISBN;
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof pregunta33)) {
            return false;
        }
        pregunta33 other = (pregunta33) obj;
        return this.ISBN == other.ISBN;
    }
}
```

Compila sin problemas

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

```
class Tool {
    private void repair() {
    }// r1

    void use() {
    }
}

class Hammer extends Tool {
    private int repair() {
        return 0;
    }// r3

    private void use() {
    }// r4

    public void bang() {
    }// r5
}

public class pregunta34 {

}
```

R4

35. ¿Qué es Git?

Git es un sistema de control de versiones distribuido de código abierto que se utiliza para rastrear cambios en archivos y coordinar el trabajo en proyectos de desarrollo de software.

36. ¿Cuál de las siguientes excepciones lanza la JVM? (Elija todas las correctas)

a) ArrayIndexOutOfBoundException
b) NumberFormatException
c) ExceptionInInitializerError
d) Java.io.IOException
e) NullPointerException

Estas excepciones son excepciones estándar de Java que son lanzadas automáticamente por la JVM en ciertas circunstancias, como errores de índice de array fuera de rango, errores en inicializadores estáticos de clases, y acceso a referencias de objetos nulos.

Las otras opciones son excepciones estándar de Java, pero no son lanzadas directamente por la JVM.

b) Esta excepción se lanza cuando se intenta convertir una cadena a un tipo numérico, pero la cadena no tiene el formato adecuado para ser convertida.

d) Esta excepción se lanza cuando ocurre un error durante la operación de entrada/salida, como leer o escribir en archivos.

Estas excepciones son lanzadas por el código Java en lugar de ser lanzadas directamente por la JVM. Aunque la JVM puede lanzar excepciones relacionadas con operaciones de E/S (como IOException), generalmente son lanzadas por el código Java que está ejecutando la aplicación.

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

git merge <nombre-de-la-rama>

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

REST (Representational State Transfer) es un estilo arquitectónico para diseñar sistemas distribuidos, como las API web. Se basa en principios como la interfaz uniforme, la ausencia de estado y el uso de HTTP. Las API RESTful utilizan estos principios para proporcionar una interfaz de programación de aplicaciones simple y escalable para acceder y manipular recursos a través de la web.

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

git pull

40. ¿Qué es un microservicio?

Un microservicio es una pequeña unidad de software independiente que realiza una función específica dentro de una aplicación más grande. Se comunica con otros microservicios a través de interfaces bien definidas y puede ser desarrollado, implementado y escalado de forma independiente.

41. Dado el siguiente código:

```java
public class pregunta41 {
    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 numeros es: " + suma);
    }
}
```

¿Este código compila sin errores?

Si

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

En Java, se utiliza el método getMessage() para obtener el mensaje asociado a una excepción. Este método está definido en la clase Throwable, que es la superclase de todas las excepciones en Java. Permite recuperar el mensaje de texto que se pasó como argumento al constructor de la excepción cuando se creó.

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.

a) Falsa: En Java, se pueden declarar tanto excepciones comprobadas como no comprobadas en la firma de un método. Las excepciones no comprobadas son aquellas que extienden de RuntimeException o Error, mientras que las excepciones comprobadas son todas las demás subclases de Exception, excluyendo RuntimeException y sus subclases.

b) Verdadero: Las excepciones en tiempo de ejecución y las excepciones no comprobadas se refieren al mismo tipo de excepciones en Java, aquellas que no necesitan ser declaradas en la firma del método o capturadas explícitamente por el código.

c) Falsa: Las excepciones en tiempo de ejecución son un subconjunto de las excepciones no comprobadas. Se diferencian en que las excepciones en tiempo de ejecución son lanzadas durante la ejecución del programa y no necesitan ser declaradas o capturadas explícitamente, mientras que las excepciones comprobadas deben ser declaradas o manejadas.

d) Falsa. Se pueden declarar tanto excepciones comprobadas como no comprobadas en la firma de un método, aunque las comprobadas deben ser manejadas o declaradas explícitamente.

e) Verdadero: En Java, los bloques try-catch solo pueden manejar excepciones que sean subclases de la clase Exception. Esto excluye a las clases de Error y RuntimeException.

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

```java
public class pregunta44 {
    public static void main(String[] args) {
        String s = "hello";
        s.toUpperCase();
        System.out.println(s);
    }
}
```

hello

45. ¿Cuál es el paquete de importación necesario para usar la clase ArrayList?

import java.util.ArrayList;

46. ¿Cuál es el formato de los datos que se envían y reciben en una API REST?

En una API REST, los datos se envían y reciben típicamente en formato JSON (JavaScript Object Notation) o XML (eXtensible Markup Language). Además de JSON y XML, otros formatos posibles en una API REST incluyen texto plano, HTML y binario.

47. ¿Cuál es la función del operador de doble dos puntos (::) en Java 8?

En Java 8, el operador de doble dos puntos (::) se utiliza en la programación funcional como una referencia a un método o constructor. Esto permite hacer referencia a un método existente o a un constructor de una manera más concisa, sin tener que definir una expresión lambda para ello.

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 extends seguida de la clase de excepción de la cual se desea heredar. Por lo tanto, para crear una excepción personalizada, se crea una nueva clase que extienda de alguna de las clases de excepción predefinidas en Java, como Exception o RuntimeException.

49. ¿Cuál de los siguientes comandos elimina el directorio target antes de iniciar el proceso de construcción?
    a) mvn site
    b) mvn build
    c) mvn answer
    d) mvn clean

La opción correcta es d) mvn clean. Este comando elimina el directorio target, que es donde Maven coloca los archivos generados durante el proceso de construcción, antes de iniciar el proceso de construcción nuevamente.

a) mvn site: Este comando genera un sitio web basado en los informes de Maven en el directorio target/site.

b) mvn build: Este comando no es válido. El comando correcto para construir un proyecto Maven es mvn package o simplemente mvn, que ejecuta el objetivo package por defecto.

c) mvn answer: No es un comando válido en Maven. No existe ningún comando llamado mvn answer.

50. ¿Cuál es el comando utilizado para ver el historial de cambios en Git?

git log

51. ¿Qué es una expresión lambda en Java 8?

En Java 8, una expresión lambda es una forma concisa de representar una función anónima que puede ser tratada como un valor y pasada como argumento a métodos o almacenada en variables.

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

```java
public class pregunta52 {
    public static void main(String[] args) {
        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 numero");
        }
    }
}
```

Uno
Dos
Tres
Otro numero

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

a) java.net
b) java.io
c) javax.swing
d) java.util
e) java.awt

Estos paquetes contienen clases y herramientas para la creación y manipulación de elementos de la interfaz gráfica de usuario (GUI) en aplicaciones Java.

Las otras opciones no contienen clases para construir una interfaz gráfica en Java por las siguientes razones:

a) java.net: Este paquete proporciona clases para la comunicación de red, como sockets y URLConnections, pero no está relacionado con la creación de interfaces gráficas.

b) java.io: Este paquete proporciona clases para entrada y salida de datos, como streams y archivos, pero no está relacionado con la creación de interfaces gráficas.

d) java.util: Este paquete proporciona clases y utilidades para estructuras de datos y otros propósitos generales, pero no está relacionado específicamente con la creación de interfaces gráficas.

54. ¿Cuál de las siguientes líneas deben ir en el espacio en blanco para que el código compile?

public class News < _____> { }

a) Solo N
b) Solo ?
c) Ninguna de las anteriores
d) News,y Object
e) ? y N
f) N, News y Object

La declaración de una clase genérica en Java requiere que se especifique un identificador que represente el tipo genérico. En este caso, el espacio en blanco debe ser reemplazado con un identificador que cumpla con las reglas de nomenclatura de Java y que represente el tipo genérico. La opción ? es un identificador válido y comúnmente utilizado para representar un tipo genérico desconocido o no específico. La letra N también es un identificador válido y podría ser utilizado en este contexto. Las otras opciones no son válidas ya que News es el nombre de la clase y Object es el nombre de una clase existente en Java, pero no es un identificador que represente el tipo genérico en esta declaración.

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

En Java 8, un stream es una secuencia de elementos que se puede procesar de manera funcional y declarativa. Se utiliza para realizar operaciones de procesamiento de datos en colecciones de manera eficiente y concisa.