**Exámenes Completos**

1. ¿Cuál es el comando utilizado para deshacer el último commit en git?
2. git reset
3. git revert
4. git amend
5. git checkout

git revert : Esta orden deshace un commit anterior creando un nuevo commit que revierte los cambios introducidos por el commit anterior.

git reset : Esta orden se utiliza para deshacer cambios en el repositorio. Puede ser usada para eliminar cambios del área de preparación (git reset --soft), eliminar cambios del área de preparación y del directorio de trabajo (git reset --mixed), o eliminar cambios completamente (git reset --hard).

git amend :Esta orden se utiliza para realizar cambios en el commit más reciente.

git checkout: Esta orden se utiliza para cambiar entre ramas o para deshacer cambios en el directorio de trabajo. Puede ser utilizada para crear una nueva rama, cambiar a una rama existente, o para descartar cambios en archivos específicos en el directorio de trabajo.

1. ¿Cuál es la diferencia entre una clase abstracta y una interfaz en java 8?
2. Una clase abstracta puede contener implementaciones de métodos, mientras que una interfaz no puede.
3. Una clase abstracta puede contener variables de instancia, mientras que una interfaz no puede.
4. Una interfaz puede contener implementaciones de métodos, mientras que una clase abstracta no puede.
5. Una interfaz solo puede heredar de una clase, mientras que una clase abstracta puede heredar… (no se ve)
6. Una interfaz puede ser implementada por múltiples clases mientras que una clase abstracta solo puede ser subclasificada.

Diferencias entre una Clase Abstracta y una Interfaz

Implementación de métodos:

Clase abstracta: Puede contener métodos abstractos (sin implementación) y métodos concretos (con implementación). Las clases que heredan de una clase abstracta deben implementar todos los métodos abstractos o ser ellas mismas clases abstractas.

Interface: Solo puede contener métodos abstractos y no puede tener ningún método con implementación. Las clases que implementan una interfaz deben proporcionar implementaciones para todos los métodos definidos en la interfaz.

Herencia:

Clase abstracta: Las clases abstractas pueden extender otras clases (abstractas o concretas) y pueden implementar interfaces.

Interface: Las interfaces solo pueden extender otras interfaces, pero no pueden extender clases. Una clase puede implementar múltiples interfaces.

Constructores:

Clase abstracta: Puede tener un constructor, que puede ser llamado desde las clases hijas mediante la palabra clave super.

Interface: No puede tener un constructor. Las interfaces no pueden ser instanciadas, por lo que no tienen constructores.

Variables de instancia:

Clase abstracta: Puede tener variables de instancia (campos), métodos estáticos y métodos de instancia.

Interface: No puede tener variables de instancia, métodos estáticos o métodos de instancia. Solo puede contener constantes (variables públicas estáticas y finales) y métodos abstractos.

Uso y flexibilidad:

Clase abstracta: Es más útil cuando se desea proporcionar una implementación base común para varias clases relacionadas. Las clases abstractas pueden contener estado y comportamiento común que las clases hijas pueden heredar y modificar según sea necesario.

Interface: Es más adecuada para definir contratos de comportamiento. Las interfaces son útiles cuando se quiere garantizar que las clases que las implementan tengan un conjunto específico de métodos, sin preocuparse por la implementación real.

1. 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)
2. Conditional (Duda Preguntar)
3. Assertion
4. Assignment
5. Iteration

Conditional :

Assinment :

Iteration:

Assertion :

1. ¿Qué es un archivo JAR en java?
2. Un archivo que contiene un archivo de configuración Maven
3. Un archivo que contiene un archivo de configuración Git.
4. Un archivo que contiene una clase Java compilada
5. Un archivo que contiene un archivo de configuración de Spring

Archivo Jar:

Un archivo JAR (Java ARchive) es un archivo comprimido que se utiliza para distribuir y empaquetar uno o más archivos Java, como clases, recursos y metadatos relacionados. Estos archivos se utilizan principalmente para distribuir bibliotecas, aplicaciones o componentes escritos en Java.

Un archivo JAR puede contener:

Clases Java: Los archivos .class que contienen bytecode Java, que son ejecutados por la máquina virtual de Java (JVM).

Recursos: Archivos como imágenes, archivos de configuración, archivos de propiedades, etc., que son utilizados por la aplicación Java.

Metadatos: Archivos MANIFEST.MF que contienen información sobre el contenido del archivo JAR, como la versión de la aplicación, el autor, las dependencias, etc.

Además, los archivos JAR pueden tener una estructura interna que incluye directorios para organizar los archivos dentro del archivo JAR.

1. ¿Qué es la sobrecarga de métodos en Java?
2. Cuando un método tiene múltiples definiciones con el mismo nombre y tipo de parámetros.
3. Cuando un método tiene múltiples definiciones con diferentes nombres y cantidades de parámetros.
4. Cuando un método tiene múltiples definiciones con diferentes tipos de cantidades de parámetros.
5. Cuando un método tiene múltiples definiciones con diferentes nombres y tipos de parámetros.

Sobrecarga de Métodos en Java: La sobrecarga de métodos en Java es un concepto que permite definir múltiples métodos en una misma clase con el mismo nombre, pero con diferentes listas de parámetros. Esto significa que puedes tener varios métodos con el mismo nombre en una clase, siempre y cuando la lista de tipos o cantidad de parámetros sea diferente entre ellos.

La sobrecarga de métodos es una forma de polimorfismo en Java, donde se puede llamar al mismo método utilizando diferentes argumentos. Cuando se invoca un método sobrecargado, el compilador de Java determina cuál de los métodos sobrecargados debe ser llamado basándose en la lista de argumentos proporcionados.

1. ¿Cuál es la diferencia entre un ArrayList y un LinkedList en Java?
2. ArrayList es más rápido que un LinkedList para agregar y eliminar elementos.
3. ArrayList es más eficiente en el uso de memoria que LinkedList.
4. LinkedList es una clase abstracta mientras que ArrayList es una clase concreta.
5. LinkedList es más rápido que ArrayList para acceder a elementos aleatorios.

La diferencia principal entre un ArrayList y un LinkedList en Java radica en la forma en que se almacenan y acceden a los elementos dentro de la estructura de datos.

ArrayList:

Un ArrayList en Java implementa la interfaz List y utiliza un arreglo dinámico para almacenar los elementos. Esto significa que los elementos se almacenan en un arreglo de tamaño variable que se redimensiona automáticamente según sea necesario.

La ventaja principal de un ArrayList es que proporciona un acceso rápido a los elementos mediante índices, lo que significa que puedes acceder directamente a cualquier elemento de la lista utilizando su posición en el arreglo subyacente.

Sin embargo, la inserción y eliminación de elementos en un ArrayList pueden ser costosas si se realizan en posiciones intermedias de la lista, ya que puede requerir el desplazamiento de otros elementos en el arreglo.

LinkedList:

Un LinkedList en Java implementa la interfaz List pero utiliza una estructura de lista doblemente enlazada para almacenar los elementos. Cada elemento en la lista contiene una referencia al elemento anterior y al siguiente en la secuencia.

La principal ventaja de un LinkedList es que permite una inserción y eliminación eficientes de elementos en cualquier parte de la lista. Esto se debe a que solo se necesitan ajustes en los enlaces de los nodos, sin necesidad de desplazar grandes cantidades de datos como en el caso de un ArrayList.

Sin embargo, el acceso a elementos en un LinkedList puede ser más lento que en un ArrayList, especialmente cuando se accede a elementos de forma secuencial, ya que puede requerir recorrer la lista desde el principio hasta la posición deseada.

En resumen, un ArrayList es más adecuado cuando se requiere un acceso rápido a los elementos mediante índices y cuando la inserción y eliminación se realizan principalmente al final de la lista. Por otro lado, un LinkedList es más adecuado cuando se requieren inserciones y eliminaciones frecuentes en posiciones intermedias de la lista y cuando el acceso a los elementos no necesita ser tan rápido. La elección entre ambos depende de las necesidades específicas de la aplicación y del tipo de operaciones que se realizarán con la lista.

1. ¿Cuándo se debe usar un bloque finally en una declaración try regular (no una prueba con recursos)?
2. Cuando no hay bloques catch en una declaración try.
3. Nunca.
4. Cuando hay dos o más bloques catch en una sentencia try.
5. Cuando hay exactamente un bloque catch en una sentencia try.
6. Cuando el código del programa no termina por sí solo.

El bloque finally en una declaración try regular se debe usar cuando es necesario ejecutar un código determinado sin importar si se produce o no una excepción en el bloque try. El bloque finally es opcional y proporciona una forma de garantizar que se ejecuten ciertas operaciones de limpieza, cierre de recursos, o cualquier otra tarea que deba realizarse independientemente del resultado de la operación en el bloque try.

Cuando hay un bloque finally pero no hay bloques catch, significa que estás utilizando un try sin capturar explícitamente ninguna excepción. En este caso, el bloque finally se ejecutará después de que el bloque try termine de ejecutarse, ya sea con éxito o generando una excepción.

1. ¿Cuál es el propósito principal de los test unitarios?
2. Comprobar la eficiencia del hardware.
3. Medir la velocidad de la aplicación.
4. Ahorrar tiempo en el desarrollo.
5. Asegurar la calidad del software.

El propósito principal de los test unitarios es verificar de manera aislada el correcto funcionamiento de las unidades individuales de código, típicamente funciones o métodos. Los test unitarios son una práctica fundamental en el desarrollo de software, ya que ayudan a asegurar que cada componente pequeño de una aplicación funciona según lo esperado.

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

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");

}

}

Texto

Descripción generada automáticamente

a) cad1 diferente a cad2

cad1 es igual a cad3

b) ca1 es igual a cad2

ca1 es igual a cad3

c) No compila

d) cad1 diferente a cad2

cad1 diferente a cad3

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

1: class Mammal (

2: public Mammal(int age) (

3:System.out.print("Mammal");

4: }

5:}

6: public class Platypus extends Mammal (

7: public Platypus() {

8:System.out.print("Platypus");

9:}

10: public static void main(String[] args) {

11: new Mammal (5);

12:}

13:}

Texto, Carta

Descripción generada automáticamente

1. Mammal.
2. MammalPlatypus.
3. El código no se compila en la línea 11.
4. El código no compila en la línea 8
5. ¿Cómo se manejan las excepciones en java?
6. Con la instrucción try-catch.
7. Con la instrucción if-else.
8. Con la instrucción for.
9. Con la instrucción while.

El bloque try-catch en Java se utiliza para manejar excepciones que pueden ocurrir durante la ejecución de un programa.

1. ¿La anotación @Ignore es usada para omitir un test por lo que no se ejecuta?
2. Verdadero
3. Falso

En Java, la anotación @Ignore se utiliza principalmente en el contexto de pruebas unitarias con bibliotecas como JUnit o TestNG. Cuando se usa con JUnit, por ejemplo, @Ignore se coloca encima de un método de prueba para indicar a JUnit que no debe ejecutar ese método durante la ejecución de la suite de pruebas.

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

public class Tester {

static {

int x = 3;

}

static int x;

public static void main(String[] args) {

x--; // line 7

System.out.println(x);

}

}

Texto

Descripción generada automáticamente

1. Error de compilación en la línea 7, x no se inicializa.
2. -1
3. -2
4. 0

Al compilar y ejecutar este código Java, la salida será 3.

Aquí está una explicación detallada de por qué:

La clase Tester tiene un campo estático x inicializado con el valor 3 dentro de un bloque de inicialización estática.

También hay otra declaración estática static int x;. Sin embargo, esta declaración no inicializa la variable x con un valor. En su lugar, se refiere a la misma variable x inicializada anteriormente en el bloque estático.

En el método main, la línea x--; decrementa el valor de x en 1. Como x inicialmente tiene el valor 3, después de esta operación, x tendrá el valor 2.

Finalmente, la línea System.out.println(x); imprime el valor actual de x, que es 2.

Sin embargo, en la imagen proporcionada, no se muestra la línea x--;. Por lo tanto, el valor de x permanece como 3, y la salida al ejecutar este código será simplemente 3.

1. ¿Qué es un operador de short circuit?
2. Sirve para realizar más eficientes las operaciones condicionales evitando ejecutar operaciones si estas ya no son necesarias.
3. Operador que nos sirve para crear una nueva clase anónima.
4. Sirve para lanzar una excepción personalizada.
5. Es un patrón de arquitectura de microservicios que nos permite evitar el consumo de servicios que están en mantenimiento .

Un operador de short-circuit (circuito corto) en programación se refiere a ciertos operadores lógicos que detienen la evaluación de una expresión tan pronto como el resultado está decidido.

1. ¿Qué es el patrón de diseño DAO y cómo se implementa en Java?
2. El patrón de diseño DAO es un patrón que se utiliza para abstraer la capa de acceso a datos en una aplicación. Se puede implementar en Java utilizando interfaces y clases concretas.
3. El patrón de diseño DAO es un patrón que se utiliza para abstraer la capa de negocios de una aplicación. Se puede implementar en Java utilizando clases abstractas y métodos estáticos.
4. El patrón de diseño DAO es un patrón que se utiliza para abstraer la capa de presentación en una aplicación. Se puede implementar en Java utilizando interfaces y clases concretas.
5. El patrón de diseño DAO es un patrón que se utiliza para abstraer la capa de infraestructura en una aplicación. Se puede implementar en Java utilizando excepciones y bloques try-catch.

El patrón de diseño DAO (Data Access Object) es un patrón estructural que proporciona una abstracción para las operaciones de acceso a la base de datos. Su principal objetivo es separar la lógica de acceso a datos de la lógica de negocio, lo que facilita el mantenimiento, la escalabilidad y la reutilización del código. Este patrón permite cambiar la fuente de datos (por ejemplo, de una base de datos SQL a una base de datos NoSQL) sin modificar la lógica de negocio.

1. ¿Qué es un endpoint en una API REST?
2. Un endpoint es la URL que se utiliza para acceder a una API REST.
3. Un endpoint es un método que se utiliza para procesar datos en una API REST.
4. Un endpoint es un controlador que se utiliza para administrar una API REST.
5. Un endpoint es un objeto que se utiliza para almacenar datos en una API REST.

Un endpoint en una API REST es una URL (Uniform Resource Locator) específica que permite a los clientes interactuar con un recurso particular de la API. Los endpoints son las rutas a través de las cuales se exponen los servicios y funcionalidades de la API, y cada endpoint corresponde a una dirección donde se puede realizar una operación particular utilizando métodos HTTP.

1. ¿Qué hace el siguiente programa?

Texto

Descripción generada automáticamente

1. El programa no compila.
2. Cuenta las letras que hay en una palabra.
3. Verifica si una palabra es un palíndromo.
4. No se ve

El propósito de este código es verificar si la cadena sPalabra es un palíndromo. Un palíndromo es una palabra que se lee igual de adelante hacia atrás y de atrás hacia adelante.

1. ¿Cuál de las siguientes opciones son verdaderas? (elija todas las correctas)
2. Java es un lenguaje orientado a objetos.
3. El código Java compilado en Windows puede ejecutarse en Linux.
4. Java permite la sobrecarga de operadores
5. Java es un lenguaje de programación funcional. 🡨 No sabemos
6. Java es un lenguaje procedimental. 🡨 No sabemos
7. Java tiene punteros a ubicaciones específicas en la memoria.

a) Java es un lenguaje orientado a objetos.

Verdadero. Java es conocido por ser un lenguaje de programación orientado a objetos (OOP), donde el enfoque principal es la creación y manipulación de objetos. Java utiliza conceptos como clases, objetos, herencia, polimorfismo, encapsulación y abstracción.

b) El código Java compilado en Windows puede ejecutarse en Linux.

Verdadero. Una de las características clave de Java es su portabilidad. El código Java se compila en bytecode, que es independiente de la plataforma. Este bytecode puede ejecutarse en cualquier sistema operativo que tenga una Java Virtual Machine (JVM) compatible, como Windows, Linux, macOS, etc.

c) Java permite la sobrecarga de operadores.

Falso. A diferencia de algunos otros lenguajes como C++, Java no permite la sobrecarga de operadores. Los operadores en Java tienen un comportamiento predefinido que no se puede cambiar.

d) Java es un lenguaje de programación funcional.

Falso. Aunque Java ha incorporado algunas características funcionales a partir de Java 8, como las expresiones lambda y la API de Streams, no es considerado un lenguaje de programación funcional puro. Java sigue siendo principalmente un lenguaje orientado a objetos.

e) Java es un lenguaje procedimental.

Falso. Java no es un lenguaje procedimental puro. Aunque se pueden escribir programas de estilo procedimental en Java, el enfoque principal es orientado a objetos. Los lenguajes procedimentales se centran en funciones o procedimientos y no en objetos.

f) Java tiene punteros a ubicaciones específicas en la memoria.

Falso. Java no utiliza punteros como C o C++. En su lugar, Java utiliza referencias para acceder a objetos. Esto es una medida de seguridad para evitar errores comunes y vulnerabilidades relacionadas con el manejo directo de la memoria.

1. ¿Qué es Maven y para qué se utiliza en el desarrollo de aplicaciones?
2. Maven es un lenguaje de programación. Se utiliza en el desarrollo de aplicaciones Java para escribir… (no se ve)
3. Maven es un servidor de base de datos. Se utiliza en el desarrollo de aplicaciones java para alojar… (no se ve)
4. Maven es un sistema de control de versiones. Se utiliza en el desarrollo de aplicaciones java para (no se ve)
5. Maven es una herramienta de gestión de dependencias. Se utiliza en el desarrollo de aplicaciones… (no se ve) en el proyecto.

Maven es una herramienta de gestión y comprensión de proyectos en Java que se utiliza principalmente para la gestión de dependencias, la construcción y la documentación de proyectos. Es un proyecto de la Apache Software Foundation y se basa en el concepto de un archivo de configuración central, llamado pom.xml (Project Object Model), que define las dependencias, las configuraciones de construcción y otra información sobre el proyecto.

1. ¿Cuál de lo siguiente es cierto? (elija todas las correctas)
2. javac compila un archivo .java en un archivo .bytecode. 🡨 Si no tuviera el punto sería correcta
3. Java toma el nombre del archivo .bytecode como parámetro.
4. javac compila un archivo .java en un archivo .class
5. Java toma el nombre de la clase como parámetro.
6. Java toma el nombre del archivo .class como parámetro.
7. javac compila un archivo .class como archivo java.
8. ¿Qué es Git y cuáles son algunos de sus comandos básicos?
9. Git es un lenguaje de programación. Algunos comandos básicos de Git incluyen “print” e “if-else”.
10. Git es una herramienta para el análisis de código. Algunos comandos básicos de Git incluyen “analyze”… (no se ve).
11. Git es un sistema de control de versiones. Algunos comandos básicos de Git incluyen “commit” y “push”.
12. Git es una herramienta para realizar pruebas de software. Algunos comandos básicos de Git incluyen… no se ve, pero no es correcta.

Git básicamente es un sistema de control de verisiones y dos de sus comando mas usados son “commit” y “push”.

1. Dados los siguientes segmentos de código, ¿Qué respuesta no es una implementación de java válida?
2. int variableA = 10;  
   float variableB = 10.5f;  
   int variableC = variableA + variableB;
3. byte variableA = 10;  
   double variableB = 10.5f;  
   double variableC = variableA + variableB;
4. byte variableA = 10;  
   float variableB = 10.5f;  
   float variableC = variableA + variableB;

Problema: En esta opción, variableA es un int y variableB es un float. Cuando sumas un int y un float, el resultado es un float. Intentar asignar un float a un int sin conversión explícita no es válido en Java.

1. ¿Qué escenario es el mejor uso de una excepción?
2. La computadora se incendió.
3. No sabe cómo codificar un método.
4. No se encuentra un elemento al buscar en una lista.
5. Se pasa un parámetro inesperado a un método.
6. Quiere recorrer una lista.

Las excepciones en Java son un mecanismo que permite manejar situaciones anómalas o errores que pueden ocurrir durante la ejecución de un programa. El propósito principal de las excepciones es proporcionar una forma estructurada y coherente de manejar los errores, separando la lógica de manejo de errores del código normal del programa. Esto hace que el código sea más legible, mantenible y robusto. Basándonos en este concepto la respuesta podría ser “C” o “D”.

1. ¿Qué es un bean en Spring?
2. Un archivo de configuración XML que se utiliza para definir la estructura de una tabla de base de datos.
3. Una instancia de una clase que se administra por el contendedor de Spring.
4. Una herramienta de inyección de dependencias que se utiliza para inyectar dependencias en una clase.
5. Una clase que se utiliza para configurar la conexión a una base de datos.

En el contexto de Spring Framework, un bean es un objeto que se instancia, se monta y se gestiona por el contenedor de Spring. En otras palabras, es un componente fundamental que Spring maneja y controla dentro de su entorno de ejecución.

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

public class Test1 extends Concreate{

1 usage

Test1(){

System.out.println(" ");

}

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

Jabstract class Send{

2 usages

Send() {

System.out.println("s");

a

}

}

Texto

Descripción generada automáticamente

1. c,s,t
2. t,s,c
3. Error en tiempo de ejecución
4. No compila

**NINGUNA ES CORRECTA**

1. ¿Cuáles de las siguientes afirmaciones sobre el polimorfismo son verdaderas? (Elija todas las correctas)
2. Si un método toma una superclase de 3 objetos, cualquiera de esas clases puede pasarse como parámetro del método
3. Un método que toma un parámetro con tipo java.lang.object tomará cualquier referencia
4. Una referencia a un objeto se puede convertir a una subclase de objetos en una conversión explícita 🡨 Cast
5. Todas las excepciones de conversión se pueden detectar en tiempo de compilación
6. 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
7. ¿Son patrones de diseño de software estructural?
8. Adapter, Proxy, Prototype y Bridge.
9. Adapter, Bridge, Proxy y Composite.
10. Agile, Builder, Singleton y Prototype.
11. Builder, Singleton y Prototype y Abstract Factory.

Adapter, Bridge, Proxy y Composite son patrones de diseño de software estructural. Estos patrones se centran en cómo componer objetos y clases para formar estructuras más grandes y complejas, facilitando la flexibilidad y reutilización del código.

1. 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()

.mapToInt(n -> n)

.average()

.orElse (0);

System.out.println(result);

}

}

Texto

Descripción generada automáticamente

1. 3.0
2. 1
3. 5
4. 2.5
5. ¿Qué son las pruebas de integración?
6. Pruebas que comprueben el rendimiento de la aplicación.
7. Pruebas que comprueben el funcionamiento de la interfaz de usuario.
8. Pruebas que comprueban el funcionamiento de varias unidades de código juntas.
9. Pruebas que comprueban el funcionamiento de una sola unidad de código.

Las pruebas de integración son un tipo de pruebas de software que se centran en verificar la interacción entre diferentes módulos, componentes o servicios de una aplicación para asegurar que funcionan correctamente juntos. En el contexto de desarrollo en Java, estas pruebas son fundamentales para garantizar que los distintos componentes desarrollados individualmente se integren de manera cohesiva.

1. ¿Qué comando se utiliza para enviar los cambios confirmados en un repositorio local al repositorio remoto?
2. git push
3. it pull
4. git commit
5. git add

El comando git push se utiliza para cargar los commits locales (cambios confirmados) en un repositorio remoto. Este comando actualiza el repositorio remoto con el contenido de la rama local especificada.

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

Texto

Descripción generada automáticamente

1. Error de compilación
2. 9
3. 7
4. No se ve, pero la correcta es 9
5. ¿Cuál es el comando utilizado para crear una nueva rama en Git?
6. git branch
7. git merge
8. git commit
9. git push

El comando git branch se utiliza para listar, crear o eliminar ramas en Git. Cuando se utiliza con el nombre de una nueva rama, crea esa nueva rama.

1. ¿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;

}

}

Texto

Descripción generada automáticamente

1. Línea 15 no compila porque other.ISBN es un atributo con modificador de acceso private.
2. Línea 14 no compila porque no está declarada o manejada ClassCastException
3. La clase compila satisfactoriamente.
4. ¿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

}

Texto

Descripción generada automáticamente

1. r5
2. r4
3. r3
4. Ninguna de las anteriores.
5. ¿Qué es Git?
6. Una herramienta de automatización de compilación que se utiliza para compilar y construir un proyecto (no se ve lo demás).
7. Una herramienta de gestión de software de dependencias que se utiliza para descargar bibliotecas y paquetes en… no se ve
8. Una herramienta de generación de informes que se utiliza para generar informes sobre el rendimiento… no se ve
9. Una herramienta de control de versiones que se utiliza para almacenar y administrar el código… no se ve.

Git es un sistema de control de versiones distribuido ampliamente utilizado para el seguimiento de cambios en el código fuente durante el desarrollo de software. Fue creado por Linus Torvalds en 2005 con el propósito de gestionar el desarrollo del kernel de Linux, pero desde entonces se ha convertido en una herramienta esencial para el desarrollo de software en general.

1. ¿Cuál de las siguientes excepciones lanza la JVM? (Elija todas las correctas)
2. ArrayIndexOutOfBoundException
3. NumberFormatException
4. ExceptionInInitializerError
5. Java.io.IOException
6. NullPointerException

La excepción que lanza directamente la JVM (Java Virtual Machine) es la ExceptionInInitializerError. Esta excepción se lanza cuando ocurre un error durante la inicialización estática de una clase o interfaz. Puede ocurrir, por ejemplo, si se produce una excepción no controlada durante la ejecución de un bloque estático de inicialización.

* ArrayIndexOutOfBoundException: Esta excepción se lanza cuando se intenta acceder a un índice fuera del rango válido de un array.
* NumberFormatException: Esta excepción se lanza cuando se intenta convertir una cadena en un tipo numérico, pero el formato de la cadena no es válido para ese tipo numérico.
* Java.io.IOException: Esta excepción se utiliza para manejar errores relacionados con operaciones de entrada y salida (I/O) en Java, como lectura o escritura de archivos.
* NullPointerException: Esta excepción se lanza cuando se intenta acceder a un objeto o método a través de una referencia nula.

1. ¿Cuál es la diferencia entre una clase abstracta y una interfaz en Java 8?
2. Una interfaz solo puede heredar de una clase, mientras una clase abstracta puede heredar de múltiples interfaces
3. Una clase abstracta puede contener implementaciones de métodos, mientras que una interfaz no puede.
4. Una clase abstracta puede contener variables de instancia, mientras que una interfaz no puede.
5. Una interfaz puede contener implementaciones de métodos, mientras que una clase abstracta no puede.

mientras que las clases abstractas pueden contener tanto métodos abstractos como métodos con implementación, las interfaces en Java 8 solo pueden contener métodos abstractos y métodos con implementación por defecto. Además, las clases pueden extender solo una clase abstracta pero pueden implementar múltiples interfaces. La elección entre una clase abstracta y una interfaz depende de los requisitos de diseño y del comportamiento deseado en la aplicación.

1. ¿Cuál es el comando utilizado para fusionar una rama en Git?
2. git Branch
3. git merge
4. git push
5. git pull

En Git, el comando utilizado para fusionar una rama en otra rama es git merge. Este comando permite integrar los cambios de una rama (la rama de origen) en la rama actual (la rama de destino).

1. ¿Qué es REST y cuál es su relación con las API web?
2. REST es un protocolo de comunicación. Su relación con las API web es que las utiliza para definir los endpoints de una API
3. REST es in lenguaje de programación. Su relación con las API web es que se utiliza para crear aplicaciones web.
4. REST es un servicio en la nube. Su relación con las API web que se utiliza para alojar las aplicaciones web.

REST es una arquitectura para aplicaciones web. Su relación con las API web es que se utiliza para definir la estructura y funcionalidades de una API.

En el contexto de las API web, REST se refiere a un conjunto de principios y restricciones que definen cómo deben ser diseñadas y expuestas las interfaces de programación de aplicaciones (API) web. Estas API implementan los principios de REST para permitir la comunicación y la interoperabilidad entre sistemas distribuidos a través del protocolo HTTP.

1. ¿Cuál es el comando utilizado para actualizar la rama local con los cambios de la rama remota en Git?
2. git checkout
3. git clone
4. git push
5. git pull

El comando utilizado para actualizar la rama local con los cambios de la rama remota en Git es git pull.

git clone: Este comando se utiliza para clonar un repositorio remoto a tu máquina local. Este comando se utiliza principalmente para obtener una copia completa del repositorio remoto por primera vez.

git push: Este comando se utiliza para enviar los cambios de tu rama local al repositorio remoto.

git checkout: Este comando se utiliza para cambiar de una rama a otra, o para restaurar archivos de un commit específico.

1. ¿Qué es un microservicio?
2. Son componentes que se pueden desplegar de forma independiente, de función múltiple, es decir… no necesariamente están relacionados.
3. Es un componente que se pueden desplegar de forma independiente y suelen ser de función única, es decir, que están estrechamente relacionados.
4. Es el conjunto de endpoints contenidos en múltiples desarrollos que se despliegan en conjunto y que están estrechamente relacionados.
5. Ninguna de las anteriores

Un microservicio es un enfoque arquitectónico para el desarrollo de software donde una aplicación se compone de pequeños servicios independientes, cada uno enfocado en realizar una tarea específica o cumplir una funcionalidad concreta. Cada microservicio es una unidad de desarrollo, despliegue y mantenimiento por sí misma.

1. Dado el siguiente código:

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: www + suma);

}

}

Texto

Descripción generada automáticamente

¿Este código compila sin errores?

1. Si, compila sin errores.
2. No, hay un error en el ciclo for.
3. No, hay un error en la inicialización de la variable “suma”.
4. No, hay un error en la declaración del arreglo.
5. ¿Qué método se utiliza para obtener el mensaje de una excepción en Java?
6. getClass()
7. printStackTrace()
8. toString()
9. getMessage()
10. ¿Cuál de las siguientes afirmaciones son verdaderas? (Elije todas las correctas)
11. Puede declarar son lo excepciones no comprobadas (unchecked).
12. Las excepciones en tiempo de ejecución son lo mismo que las excepciones no comprobadas.
13. Las excepciones en tiempo de ejecución son lo mismo que las excepciones comprobadas.
14. Solo puede declarar excepciones comprobadas (checked)
15. Solo puede manejar subclases de Exception.

En esta te pide que selecciones varias, pero no se puede, la e) también sería correcta.

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

String s = "hello";

s.toUpperCase();

System.out.println(s);

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. NullPointerException
2. hello
3. HELLO
4. Hello

La razón es que en Java, los strings son inmutables, lo que significa que no se pueden modificar después de ser creados. Cuando llamas al método toUpperCase() en la variable s, este método devuelve un nuevo string que representa s en mayúsculas, pero no modifica el valor original de s. Por lo tanto, cuando imprimes s después de llamar a toUpperCase(), sigue manteniendo el mismo valor original en minúsculas, que es "hello".

1. ¿Qué son las pruebas de integración?
2. Pruebas que comprueben el funcionamiento de la interfaz de usuario.
3. Pruebas que comprueban el funcionamiento de varias unidades juntas.
4. Pruebas que comprueben el rendimiento de la aplicación.
5. Pruebas que comprueben el funcionamiento de una ola unidad de código.

Las pruebas de integración en Java son un tipo de prueba de software que se centran en verificar que los distintos módulos o componentes de una aplicación funcionen correctamente juntos. Estas pruebas se realizan después de las pruebas unitarias y antes de las pruebas de sistema completas. El objetivo es identificar problemas que puedan surgir cuando los componentes individuales se combinan e interactúan entre sí.

1. ¿Cuál es el comando utilizado para crear una nueva rama en Git?
2. git commit
3. git branch
4. git merge
5. git push

El comando utilizado para crear una nueva rama en Git es git branch.

* git commit: Se usa para guardar los cambios en el repositorio local.
* git merge: Se usa para fusionar ramas.
* git push: Se usa para enviar los commits locales al repositorio remoto.

1. ¿Cuál es el paquete de importación necesario para usar la clase ArrayList?
2. import.java.net.\*;
3. import.java.awt.\*;
4. import.java.io. \*;
5. import.java.util.\*;

La clase ArrayList está ubicada en el paquete java.util, por lo que necesitas importar este paquete para utilizar ArrayList en tu código. La opción correcta para importar todo el contenido de java.util es import java.util.\*;.

1. ¿Cuál es el formato de los datos que se envían y reciben en una API REST?
2. YAML
3. XML
4. JSON
5. Todos los anteriores

En una API REST, el formato de los datos que se envían y reciben suele ser JSON (JavaScript Object Notation), aunque también se puede utilizar XML y otros formatos menos comunes como YAML o CSV. JSON es el formato más popular debido a su simplicidad, legibilidad y compatibilidad con la mayoría de los lenguajes de programación.

1. Cuál es la función del operador de doble dos puntos (::) en Java 8?
2. El operador doble dos puntos se utiliza para crear una nueva instancia de una clase en Java 8.
3. EL operador de doble dos puntos no se utiliza en java 8.
4. El operador de doble dos puntos se utiliza para acceder a métodos estáticos en Java 8.
5. El operador de doble dos puntos se utiliza para acceder a métodos no estáticos en Java 8.

En Java 8, el operador de doble dos puntos (::) se utiliza para referenciar métodos de forma abreviada, especialmente en contextos donde se espera una interfaz funcional compatible. Esto se conoce como referencias a métodos y es una característica de programación funcional introducida en Java 8 que permite pasar métodos como argumentos a métodos o construir implementaciones de interfaces funcionales sin necesidad de escribir expresiones lambda.

1. ¿Qué palabra clave se utiliza para definir una excepción personalizada en Java?
2. try
3. throw
4. finally
5. catch

En Java, throw es una palabra clave que se utiliza para lanzar una excepción explícitamente dentro de un bloque de código. Sirve para indicar que ha ocurrido una condición excepcional que no puede ser manejada dentro del método actual y necesita ser propagada hacia un contexto superior para su manejo.

Cuando utilizas throw, estás creando una instancia de una clase de excepción y la estás "lanzando" (throwing) para notificar al código que está llamando al método actual que ha ocurrido un problema.

1. ¿Qué es un operador de short circuit?
2. Sirve para realizar más eficientes las operaciones condicionales evitando ejecutar operaciones si estas ya no son necesarias.
3. Operador que nos sirve para crear una nueva clase anónima.
4. Sirve para lanzar una excepción personalizada.
5. Es un patrón de arquitectura de microservicios que nos permite evitar el consumo de servicios que están en mantenimiento.

En Java, un operador de short-circuit es un operador lógico que no evalúa su segundo operando si el primer operando es suficiente para determinar el resultado de la expresión. Esto se utiliza para optimizar el rendimiento y evitar errores innecesarios o costosos al evaluar condiciones.

1. ¿Cuál de los siguientes comandos elimina el directorio target antes de iniciar el proceso de construcción?
2. mvn site
3. mvn build
4. mvn answer
5. mvn clean

El comando que elimina el directorio target antes de iniciar el proceso de construcción es:

Mvn clean

El comando mvn clean ejecuta la fase clean del ciclo de vida de Maven, que elimina los archivos y directorios generados previamente durante la compilación y el proceso de construcción, incluido el directorio target. Esto garantiza que el entorno esté limpio antes de comenzar el proceso de construcción, lo que puede ser útil para evitar conflictos o problemas causados por archivos obsoletos.

1. ¿Cuál es el comando utilizado para ver el historial de cambios en Git?
2. git log
3. git status
4. git commit
5. git diff

El comando utilizado para ver el historial de cambios en Git es a) git log. Este comando muestra una lista de confirmaciones (commits) en orden cronológico inverso, lo que significa que la confirmación más reciente se muestra primero. Cada entrada en el registro de confirmaciones incluye detalles como el autor, la fecha y hora de la confirmación, y un mensaje descriptivo que proporciona información sobre los cambios realizados en esa confirmación.

1. ¿Qué es una expresión lambda en Java 8?
2. Una expresión lambda es una forma de escribir una clase anónima en Java 8.
3. Una expresión lambda es una forma de escribir una función anónima en Java 8.
4. Una expresión lambda es un método que se llama automáticamente cuando se crea un objeto.
5. Una expresión lambda es un método que se llama de forma explícita desde el código.

Una expresión lambda en Java 8 es una característica que permite representar de manera concisa y clara una implementación de una interfaz funcional. Proporciona una forma de expresar código de manera más compacta y legible, especialmente cuando se trata de operaciones que implican el uso de interfaces funcionales, como aquellas que requieren la implementación de un único método abstracto.

1. ¿Cuál es la diferencia entre una clase abstracta y una interfaz en Java 8?
2. Una interfaz solo puede heredar de una clase, mientras una clase abstracta puede heredar de múltiples interfaces
3. Una clase abstracta puede contener implementaciones de métodos, mientras que una interfaz no puede.
4. Una clase abstracta puede contener variables de instancia, mientras que una interfaz no puede.
5. Una interfaz puede contener implementaciones de métodos, mientras que una clase abstracta no puede.
6. ¿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");

}

Texto

Descripción generada automáticamente

1. Dos
2. Uno Dos Tres Otro número
3. Uno
4. Otro número
5. 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) {

ال

}

}

1 usage 2 inheritors

Jabstract class Send{

2 usages

Send(){

System.out.println("s");

}

Texto

Descripción generada automáticamente

1. c,s,t
2. t,s,c
3. Error en tiempo de ejecución
4. No compila
5. ¿Qué es REST y cuál es su relación con las API web?
6. REST es un protocolo de comunicación. Su relación con las API web es que las utiliza para definir los endpoints de una API
7. REST es in lenguaje de programación. Su relación con las API web es que se utiliza para crear aplicaciones web.
8. REST es un servicio en la nube. Su relación con las API web que se utiliza para alojar las aplicaciones web.
9. REST es una arquitectura para aplicaciones web. Su relación con las API web es que se utiliza para definir la estructura y funcionalidades de una API.
10. De los siguientes paquetes, ¿cuáles contienen clases para construir una interfaz gráfica? (Elije todas las que correspondan)
11. java.net
12. java.io
13. javax.swing
14. java.util
15. java.awt
16. ¿Cuál de las siguientes líneas deben ir en el espacio en blanco para que el código compile?

public class News < \_\_\_\_\_> { }

1. Solo N
2. Solo ?
3. Ninguna de las anteriores
4. News,y Object
5. ? y N
6. N, News y Object
7. ¿Qué es el patrón de diseño DAO y cómo se implementa en Java?
8. El patrón de diseño DAO es un patrón que se utiliza para abstraer la capa de acceso a datos en una aplicación. Se puede implementar en Java utilizando interfaces y clases concretas.
9. El patrón de diseño DAO es un patrón que se utiliza para abstraer la capa de negocios de una aplicación. Se puede implementar en Java utilizando clases abstractas y métodos estáticos.
10. El patrón de diseño DAO es un patrón que se utiliza para abstraer la capa de presentación en una aplicación. Se puede implementar en Java utilizando interfaces y clases concretas.
11. El patrón de diseño DAO es un patrón que se utiliza para abstraer la capa de infraestructura en una aplicación. Se puede implementar en Java utilizando excepciones y bloques try-catch.
12. ¿Cuál de las siguientes líneas deben ir en el espacio en blanco para que el código compile?

public class News < \_\_\_\_\_> { }

1. Solo N
2. Solo ?
3. Ninguna de las anteriores
4. News,y Object
5. ? y N
6. N, News y Object -------------
7. ¿Qué es un stream en Java 8 y para qué se utiliza?
8. Un objeto que representa una conexión de entrada o salida de datos.
9. Un objeto que representa una secuencia de elementos y se utiliza para procesar colecciones de forma declarativa.
10. Un objeto que se utiliza para leer y escribir archivos de texto.
11. Un objeto que se utiliza para crear y manipular bases de datos.
12. Son patrones de diseño de microservicios
13. Circuit Breaker, Adaptative Lifo, MQ Strategy
14. System, Process y Client
15. Retry, Circuit Breaker, Adaptative Lifo y Bulkhead
16. Ninguna de las anteriores
17. ¿Cuándo se debe usar un bloque finally en una declaración try regular (no una prueba con recursos)?
18. Nunca.
19. Cuando el código del programa no termina por sí solo.
20. Cuando hay dos o más bloques catch en una sentencia try.
21. Cuando hay exactamente un bloque catch en una sentencia try
22. Cuando no hay bloques catch en una declaración try
23. 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?
24. Assertion
25. Iteration
26. Assignment
27. Conditional
28. ¿Qué afirmaciones son verdaderas tanto para las clases abstractas como para las interfaces? (Elije todas las correctas)
29. Ambos pueden contener métodos estáticos.
30. Ambos se pueden ampliar con la clave extend.
31. Ambos pueden contener métodos predeterminados.
32. Ambos heredan de java.lang.Object.
33. Ninguno de los dos puede ser instanciado directamente.
34. Ambos pueden contener variables finales estáticas públicas.
35. Supone que todos los métodos dentro de ellos son abstractos.
36. ¿Cuál no es un objetivo de Maven?
37. Clean
38. Package
39. Debug
40. Install
41. ¿Si deseas obtener una copia de un repositorio Git existente en un servidor qué comando se utiliza?
42. git commit
43. git log
44. git clone
45. git add
46. ¿Cuál es el comando utilizado para fusionar una rama en Git?
47. git push
48. git branch
49. git pull
50. git merge
51. ¿Qué es un repositorio remoto en Git?
52. Una herramienta que se utiliza para compartir y fusionar cambios entre diferentes ramas de un repositorio.
53. Una copia local de un repositorio que se utiliza para hacer cambios en el código fuente.
54. Un servidor Git que almacena una copia central del repositorio.
55. Un archivo que contiene una instantánea del código fuente en un momento determinado.
56. ¿Cuál es el comando utilizado para actualizar la rama local con los cambios de la rama remota en Git?
57. git pull
58. git push
59. git clone
60. git checkout
61. Dada la siguiente clase

public class Helper {

public static < U extends Exception > void

printEception(UU){

System.out.println(u.getMessage());

ב

}

public static void main(String[] args) {

//linea 9

}

}

Texto

Descripción generada automáticamente

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

1. Helper.printException(new Exception(“B”));
2. Helper. printException(new FileNotFoundException(“A”));
3. Helper.<Throwable>printException(new Exception(“C”));
4. Helper.<NullPointerException>printException(new NullPointerException (“D”));
5. Helper. printException(new Throwable(“E”));
6. ¿Cuál es la salida al ejecutar el siguiente código?

Texto

Descripción generada automáticamente

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); 11

}

}

1. 51tuna
2. 5tuna
3. 5
4. 41
5. 5 tuna
6. 4 1
7. El código no compila
8. ¿Qué es Git?
9. Una herramienta de automatización de compilación que se utiliza para compilar un proyecto.
10. Una herramienta de generación de informes que se utiliza para generar informes sobre el rendimiento de una aplicación.
11. Una herramienta de gestión de dependencias que se utiliza para descargar bibliotecas y paquetes en un proyecto de Java.
12. Una herramienta de control de versiones que se utiliza para almacenar y administrar el código fuente de un proyecto.
13. ¿Cuál de las siguientes opciones son correctas? (Elija todas las correctas)

Texto

Descripción generada automáticamente

1. s2 = s2
2. s3 = null
3. s1 = s1-------------
4. s3 = a
5. El código no compila
6. s2 = s2b
7. s1 = a
8. ¿A qué hace referencia el principio de Liskov?
9. Nos indica que una clase no debe tener solo una funcionalidad sino varias para reducir el uso de objetos.
10. Este principio nos indica que dentro del programa una clase puede ser sustituida por cualquier clase que se extienda de ella sin alterar el comportamiento del programa.
11. Nos indica que cualquier clase se puede extender para agregar funcionalidad, pero no se puede modificar.
12. Este principio nos indica que dentro del programa una clase puede ser sustituida por su clase padre sin alterar el comportamiento del programa.
13. ¿Qué es un “code smell”?
14. Un componente de la biblioteca estándar de Java
15. Un error en tiempo de compilación que se produce en Java
16. Un indicador de que puede haber un problema en el código que puede ser dificil de detectar o que podría ser una fuente potencial de errores o problemas de mantenimiento en el futuro.
17. Una práctica de programación recomendada en Java.
18. ¿Cuál de las siguientes afirmaciones son verdaderas? (Elije todas las correctas)
19. Las excepciones de tiempo de ejecución son lo mismo que las excepciones comprobadas.
20. Las excepciones en tiempo de ejecuciones son lo mismo que las excepciones no comprobadas.
21. Solo pueden manejar subclases de exception.
22. Solo puede declarar excepciones comprobadas (checked).
23. Puede declarar solo excepciones no comprobadas.
24. ¿Cuáles son las salidas del siguiente código? (Elije todas las correctas)

public class StrinBuilders {

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);

}

}

Texto

Descripción generada automáticamente

1. El código no compila.
2. s3 = a
3. s1 = a
4. s2 = s2
5. s1 = s1
6. s3 = null
7. s2 = s2b
8. Selecciona la respuesta correcta con respecto al resultado del bloque de código.

public class Test1 extends Concreate{

Test1(){

System.out.println("t");

}

public static void main(String[] args) {

// TODO Auto-generated method stub

new Test1();

}

}

class Concreate extends Send{

Concreate(){

System.out.println("c");

}

private Concreate (String s) {

}

}

abstract class Send{

Send(){

System.out.println("s");

}

}

Texto

Descripción generada automáticamente

1. Error en tiempo de ejecución.
2. No compila.
3. t,s,c
4. c,s,t
5. ¿Cuál es la función del operador de doble dos puntos (::) en Java 8?
6. El operador doble dos puntos se utiliza para crear una nueva instancia de una clase en Java 8.
7. EL operador de doble dos puntos no se utiliza en java 8.
8. El operador de doble dos puntos se utiliza para acceder a métodos estáticos en Java 8.
9. El operador de doble dos puntos se utiliza para acceder a métodos no estáticos en Java 8.
10. ¿Qué significa el acrónimo CRUD en una API REST?
11. Code, Register, Update, Debug
12. Create, Read, Update, Delete
13. Call, Receive, Use, Debug
14. Customize, Request, Use, Debug
15. ¿Qué es un bean en Spring?
16. Un archivo de configuración XML que se utiliza para definir la estructura de una tabla de base de datos.
17. Una instancia de una clase que se administra por el contendedor de Spring.
18. Una herramienta de inyección de dependencias que se utiliza para inyectar dependencias en una clase.
19. Una clase que se utiliza para configurar la conexión a una base de datos.
20. ¿Para qué nos sirve utilizar un profile dentro del archivo pom.xml?
21. Etiqueta por la cual podemos definir la versiones de nuestras dependencias.
22. Es la etiqueta por la cual podemos definir las características que tendrá nuestro proyecto al ser compiladas.
23. Etiqueta por la cual definimos los parámetros de conexión a un repositorio.
24. No existe esta etiqueta en Maven.
25. ¿Cuál es el comando utilizado para actualizar la rama local con los cambios de la rama remota en Git?
26. git checkout
27. git clone
28. git push
29. git pull
30. Dadas las siguientes clases Vehicle y Car

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

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)

1. mileage = 15285;
2. Ninguna de las anteriores.
3. make = “Honda”;
4. year = 2009;
5. model = “Pilot”;
6. Enumere cuatro interfaces de la API colecciones
7. List, Map, Set, Queue.
8. ArrayList, Map, Set, Queue.
9. List, HashMap, HashSet, PriorityQueue.
10. List, Map, HashSet, PriorityQueue.
11. Selecciona la respuesta correcta con respecto al resultado del bloque de código.

Texto

Descripción generada automáticamente

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)prinarIntento);

interface Side ( String getSide();}

class Head implements Side (

public String getSide() { return "Head "1

}

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 ");}

}

1. Head  
   Object  
   Tail  
   Side
2. No compila
3. Side  
   Object  
   Tail  
   Side
4. Head  
   Head  
   Tail  
   Tail
5. Side  
   Head  
   Tail  
   Side
6. ¿Cuál es la salida al ejecutar el siguiente código?

Texto, Carta

Descripción generada automáticamente

public class Lion {

public void roar (String roarl, 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);

}

1. roar roar!!!
2. roar!!! Roar
3. Se lanza una excepción
4. roar!!! roar!!!
5. roar roar
6. El código no compila
7. ¿Cuál de los siguientes es cierto acerca de una subclase completa?
8. Una subclase concreta no se puede marcar como final.
9. Una subclase concreta debe implementar todos los métodos definidos en una interfaz heredada.
10. Una subclase concreta debe implementar todos los métodos abstractos heredados.
11. Una subclase concreta puede declararse como abstracta.
12. Los métodos abstractos no pueden ser anulados por una subclase concreta.
13. ¿Cuál es la salida del siguiente código?

protected abstract void catchAnObject(Object x);

public static void main(String[] args) {

java.util.Date now = new java.util.Date();

Catchable target = new MyStringCatcher();

target.catchAnObject(now);

}

}

class MyStringCatcher extends Catchable {

public void catchAnObject(Object x) {

System.out.println("Caught object");

}

public abstract class Catchable {

public void catchAnObject(String s) {

System.out.println("Caught string");

}

}

Una captura de pantalla de una computadora

Descripción generada automáticamente con confianza media

1. Error de compilación línea 12
2. Error compilación línea 16
3. Caught string
4. Error compilación línea 2
5. Caught Object
6. Seleccione la respuesta que considere correcta, dado el siguiente bloque de código.

Texto

Descripción generada automáticamente

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);

int result = numbers.stream()

.filter (n ->20)

.reduce(0, (a, b) -> a + b);

System.out.println(result);

}

)

1. 3
2. 9
3. 14
4. 6
5. Qué declaración representa una declaración válida que permitirá la inclusión de clases del paquete java.util?
6. #include java.util.\*;
7. #include java.util;
8. import java.util.\*;
9. import java.util;
10. ¿Qué es la cobertura de código?
11. La cantidad de veces que se ejecuta una línea de código.
12. La cantidad de errores detectados por una prueba.
13. La cantidad de código que se ejecuta durante una prueba.
14. La cantidad de tiempo que tarda una prueba en ejecutarse.
15. ¿Qué es Git?
16. Una herramienta de automatización de compilación que se utiliza para compilar un proyecto.
17. Una herramienta de generación de informes que se utiliza para generar informes sobre el rendimiento de una aplicación.
18. Una herramienta de gestión de dependencias que se utiliza para descargar bibliotecas y paquetes en un proyecto de Java.
19. Una herramienta de control de versiones que se utiliza para almacenar y administrar el código fuente de un proyecto.
20. ¿Cuál es el formato correcto para hacer un commit en Git?
21. Descripción breve del cambio y nombre del autor.
22. Tipo de cambio, descripción breve, cuerpo opcional y notas de pie de página.
23. Solo se necesita una breve descripción del cambio.
24. Nombre de la rama, descripción detallada del cambio y fecha.
25. ¿Qué es el patrón de diseño Singleton y cómo se implementa en Java 8?
26. El patrón de diseño Singleton es un patrón que se utiliza para garantizar que una clase tenga una única instancia en todo el sistema. Se implementa utilizando una variable estática y un constructor privado.
27. El patrón de diseño Singleton es un patrón que se utiliza para abstraer la capa de infraestructura en una aplicación. Se implementa utilizando excepciones y bloques try-catch.
28. El patrón de diseño Singleton es un patrón que se utiliza para abstraer la capa de presentación en una aplicación. Se implementa con interfaces y clases concretas.
29. El patrón de diseño Singleton es un patrón que se utiliza para abstraer la capa de negocios en una aplicación. Se implementa utilizando clases abstractas y métodos estáticos.
30. ¿Qué es un Microservicio?
31. Ninguna de las anteriores
32. Es un componentes que se pueden desplegar de forma independiente en función múltiple, es decir, englobando endpoint que no necesariamente están relacionados.
33. Es el conjunto de endpoints contenidos en múltiples desarrollos que se despliegan en conjunto y que están estrechamente relacionados.
34. Es un componentes que se pueden desplegar de forma independiente y que suelen ser de función única, es decir, englobando endpoint y están estrechamente relacionandos.
35. ¿Cuál no es un objetivo de Maven?
36. Debug
37. Clean
38. Package
39. Install
40. En los verbos REST ¿Cuál es la diferencia en el uso de PATCH y PUT?
41. Son exactamente iguales, no hay diferencia de uso.
42. PATCH requiere se le envíe la entidad completa mientras que PUT solo los atributos a modificar.
43. PUT requiere se le envíe la entidad completa mientras que PATCH solo los atributos a modificar.
44. PATCH es un verbo deprecado sustituido por PUT.
45. ¿Qué es una expresión lambda en Java 8?
46. Una expresión lambda es una forma de escribir una clase anónima en Java 8.
47. Una expresión lambda es una forma de escribir una función anónima en Java 8.
48. Una expresión lambda es un método que se llama automáticamente cuando se crea un objeto.
49. Una expresión lambda es un método que se llama de forma explícita desde el código.
50. ¿Cuál es la diferencia entre una clase abstracta y una interfaz en Java 8?
    1. Una interfaz solo puede heredar de una clase, mientras una clase abstracta puede heredar de múltiples interfaces
    2. Una clase abstracta puede contener implementaciones de métodos, mientras que una interfaz no puede.
    3. Una clase abstracta puede contener variables de instancia, mientras que una interfaz no puede.
    4. Una interfaz puede contener implementaciones de métodos, mientras que una clase abstracta no puede.
51. ¿Cuál es la diferencia entre las anotaciones: @RestController, @Component, @Service y @Repository?
52. @Controller es una anotación que nos ayuda a construir una API REST mientras que @Service, @Component y @Repository solo marcan las clases que se deben de inicializar.
53. @Controles, @Component son anotaciones que crean bean y exponen la serialización de las clases mientras que @Service y @Repository requieren de una inicialización manual.
54. No existe diferencia funcional entre ellas sino semántica, las 4 son anotaciones de Spring que crean un bean y lo agregan al contexto de Spring.
55. @Service y @Repository son anotaciones que crear bean y exponen la serialización de las clases mientras que @Controller. @Component requiere de una inicialización manual.
56. ¿Cuál es una buena práctica al escribir pruebas unitarias?
57. Ejecutar pruebas con poca frecuencia.
58. Asegurarse de que las pruebas sean claras y concisas.
59. Probar solo una pequeña parte de una función.
60. Hacer que las pruebas dependan de otras pruebas.
61. ¿Cuál es la ventaja de usar APIs REST sobre otros tipos de servicios web?
62. Mayor seguridad.
63. Mayor facilidad de implementación.1
64. Mayor velocidad de transferencia de datos.
65. Mayor compatibilidad con diferentes plataformas.
66. Selecciona la respuesta correcta con respecto al resultado del bloque de código.

Texto

Descripción generada automáticamente

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));

}

}

1. No compila
2. [25, 7, 25, 67]

[67, 7, 25]

[7, 25, 67]

[67, 7, 25]

[7, 25, 67]

1. [25, 7, 67]  
   [67, 7, 25]  
   [7, 25, 67]  
   [67,7, 25]  
   [7, 25, 67]
2. [67, 7, 25]  
   [67, 7, 25]  
   [67, 7, 25]  
   [67, 7, 25]  
   [67, 7, 25]
3. [25, 7, 25, 67]  
   [7, 25, 67]  
   [67, 7, 25]  
   [7, 25, 67]  
   [67, 7, 25]
4. ¿Cuál es la salida al ejecutar el siguiente código?

Texto

Descripción generada automáticamente

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);

}

}

1. 51tuna
2. 5tuna
3. 5
4. 41
5. 5 tuna
6. 4 1
7. El código no compila
8. ¿Cuáles son los 4 pilares de la programación orientada a objetos?
9. Polimorfismo, Coerción, Herencia y Encapsulamiento.
10. Encapsulamiento, Coerción, Polimorfismo y Abstracción.
11. Polimorfismo, Herencia, Encapsulamiento y Sincronía.
12. Polimorfismo, Abstracción, Herencia y Encapsulamiento.
13. ¿Cuál es el comando utilizado para ver el historial de cambios en git?
14. git diff
15. git status
16. git log
17. git commit
18. ¿Qué utilidad de línea de comandos basada en MS Windows le permitirá ejecutar el intérprete de Java sin abrir la ventana de la consola?
19. jconsole
20. javaw
21. interpw
22. java -wo
23. ¿Qué es un endpoint en unaAPI REST?
24. Un endpoint es un objeto que se utiliza para almacenar datos en una API REST.
25. Un endpoint es un método que se utiliza para procesar datos en una API REST.
26. Un endpoint es un controlador que se utiliza para administrar una API REST.
27. Un endpoint es la URL que se utiliza para acceder a una API REST.
28. ¿Cuál de lo siguiente es cierto?
29. java toma el nombre del archivo .bytecode como parámetro.
30. javac compila un archivo .java en un archivo .class
31. Java toma el nombre de la clase como parámetro.
32. javac compila un archivo .class en un archivo .java
33. javac compila un archivo .java en un archivo. bytecode
34. Java toma el nombre del archivo .class como parámetro.
35. ¿Cuál es el valor de x e y al final el programa?

Pantalla de computadora con letras

Descripción generada automáticamente

1. X=9 y=10
2. X=10 y=9
3. X=10 y=10
4. X=9 y=9
5. ¿Dado el siguiente enum y clase cuál es la opción que puede ir en el espacio en blanco para que el código compile?

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

1. Ninguno de los anteriores
2. case SUMMER ->
3. case Season.Winter:
4. case FALL:
5. case Winter, Spring:
6. case SUMMER | WINTER:
7. ¿Cuál es una buena práctica al escribir pruebas unitarias?
8. Hacer que las pruebas dependan de otras pruebas
9. probar solo una pequeña parte de una función
10. ejecutar pruebas con poca frecuencia
11. asegurarse de que las pruebas sean claras y concisas
12. ¿Cuál es el resultado de compilar y ejecutar el siguiente programa’

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

1. False
2. no se produce salida
3. true
4. error de compilación
5. ¿Cómo se manejan las excepciones en Java?
6. Las excepciones el manejan con bloques switch case en Java. La excepción trae with Resources es una forma de lanzar una excepción en un método.
7. Las excepciones se manejan con bloques while en Java. La excepción trae with Resources es una forma de manejar excepciones de compilación
8. las excepciones se manejan con bloques if else en Java. La excepción trae with resource es una forma de manejar las excepciones en tiempo de ejecución.
9. Las excepciones se manejan con bloques try catch finally en Java. La excepción trae with Resources es una forma de cerrar automáticamente los recursos abiertos en un bloque try.
10. ¿Qué clase del paquete java.io permite leer y escribir archivos en ubicaciones específicas dentro de un archivo?
11. File
12. filename filter
13. file descriptor
14. RandomAccessFile
15. 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)

Texto, Carta

Descripción generada automáticamente

1. Ninguna, el código compila bien
2. línea 6
3. línea 9
4. línea 7
5. línea 8
6. línea 5
7. ¿Qué es una expresión lambda en Java?
8. Una instancia de una clase que implementa una interfaz funcional
9. Una instancia de una clase abstracta que se utiliza para implementar métodos anónimos
10. Una forma concisa de representar una función anónima que se puede pasar como argumento
11. Un método que no tiene cuerpo
12. ¿Qué hace el siguiente código fuente?

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

1. Muestra los números del 1 al 10
2. muestra un 10
3. se queda en un bucle infinito
4. muestra los números del 0 al 9

------------------------------------------------------------------------------------------------------------------------

1. Comando Docker que se utiliza para construir la imagen a partir del Dockerfile

docker build

1. ¿Qué hace el siguiente Código fuente?

int x = 0;

boolean flag = false;

while (( x < 10) II !flag) {

System.out.println (x);

x++;

}

* 1. Muestra los números del 0 al 9
  2. Muestra un 10
  3. Se queda en un bucle infinito
  4. Muestra los números del 1 al 10

1. ¿Cuál es la diferencia entre una interfaz y una clase abstracta en java?
2. Una clase abstracta puede contener variables y métodos concretos mientras que una interfaz sólo puede contener métodos abstractos
3. Una clase abstracta puede ser instanciada mientras que una interfaz no puede ser instanciada
4. Una interfaz puede ser implementada por múltiples clases
5. Una interfaz puede contener variables y métodos concretos
6. ¿Cuál de los siguientes es cierto acerca de una subclase concreta?
   1. Una subclase concreta debe implementar
   2. Una subclase concreta puede declararse como abstracta
   3. Los métodos abstractos no pueden ser // por una
   4. Una subclase concreta debe implementar todos los métodos abstractos heredados
   5. Una subclase concreta no se puede marcar como final
7. ¿Cuándo se debe usar un bloque finally en una declaración de try regular no una prueba con recursos?
8. Cuando hay exactamente un bloque catch en una sentencia try
9. Cuando hay dos o más bloques catch en una sentencia try
10. Nunca
11. Cuando no hay bloques catch en una declaración try
12. cuando el código del programa no termina por sí solo
13. ¿En los verbos REST cuál es la diferencia entre el verbo PATCH y PUT?

PUT requiere se le envíe entidad concreta mientras que PATCH solo los atributos a modificar

1. De los siguientes paquetes ¿Cuáles contienen clases para construir una interfaz

* java.awt
* javax.swing

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

Probar sobre una pequeña parte de una función

1. ¿Seleccione la respuesta que considere correcta, dado el siguiente bloque de código?

import java.util.Arrays;

public class Example {

public static void main( String [] args ) {

int [] [] [] matrix= {{{ 1, 2}, {3, 4}}, {{5, 6}, {7, 8}}};

int [] [] flattened= Arrays.stream(matrix)

.flatMapToIn(layer)Arrays.string(layer)

.flatMapToIn(Arrays::stream))

.boxed().map(n -> new int[](n)).toArray(int[] []::new);

System.out.println(Arrays.deepToString(flattened));

}

}

[ [1], [2], [3], [4], [5], [7], [8] ]

[ [1, 2 ]], [ [ 3, 4]], [[ 5, 6]], [ [ 7, 8]]

[ [ 1, 2], [ 3,4], [ 5,6], [ 7,8]]

1. Seleccione 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));

}

Output:

[67, 7, 25]

[7, 25, 67]

[67, 7, 25]

[7, 25, 67]

1. Cuáles son las salidas del siguiente código. Elige todas las correctas

public class StringBuilders {

public static StringBuilder work (StringBuilder a,

StringBuilder 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);

}

}

Output

s3 = null