1. Which of the following statements is true about arrays in Java?
R: An array has a fixed size

en Java, los arreglos (arrays) tienen un tamaño fijo una vez que se han declarado. Cuando se crea un array, es necesario especificar su tamaño, y este no puede cambiar durante la ejecución del programa. Si se necesita una estructura de datos con tamaño dinámico, se pueden utilizar colecciones como ArrayList

2. ¿Cuál de los siguientes componentes es parte de una solicitud HTTP?
R: URL, headers, cuerpo de la solicitud
Explicación técnica:

Una solicitud HTTP típica se estructura así:

Método URL Versión
Headers
[Espacio en blanco]
Cuerpo de la solicitud (opcional)

Ejemplo de una solicitud POST con todos los componentes:

```
POST /api/usuarios HTTP/1.1
Host: www.ejemplo.com
Content-Type: application/json
Authorization: Bearer token123

{
  "nombre": "Juan",
  "edad": 30
}
```

3.
```
public class LogicQuestion {
 public static void main(String[] args) {
 int count = 0;
 for (int i = 0; i < 10; i++) {
 if (i % 2 == 0) {
 count++;
 }
 }
 System.out.println("Count: " + count);
 }
}
```

R:Count 5

4. Which method is used to sort elements of a List in natural order in Java? (*) R:Collections.sort()
para ordenar los elementos de una List, se utiliza el método Collections.sort(), que organiza los elementos en orden natural si la lista contiene elementos que implementan la interfaz Comparable.
Collections.sort(List<T> list)

5. ¿Qué significa que un cambio en el software sea retro-compatible?
R: El cambio garantiza que el software será funcional sin necesidad de modificaciones en el código que depende de él

La retrocompatibilidad (o compatibilidad hacia atrás) en software significa que una nueva versión de un programa, sistema o biblioteca sigue siendo compatible con versiones anteriores. Es decir, los cambios introducidos no rompen la funcionalidad existente, permitiendo que aplicaciones o archivos creados en versiones previas sigan funcionando correctamente sin necesidad de modificaciones.

6.
```
class Base {
 public Base() {
 System.out.println("Base constructor");
 }
 public Base(String message) {
 System.out.println("Base constructor with message: " + message);
 }
}
class Derived extends Base {
 public Derived() {
 super("Hello");
 System.out.println("Derived constructor");
 }
}
public class Test {
 public static void main(String[] args) {
 Derived derived = new Derived();
 }
}
```

R: Base constructor with message: Hello
Derived constructor

Primeri se llama al constructor de Derived.
En la primera línea de este constructor, se invoca super("Hello");, lo que significa que el constructor de Base con un String es ejecutado.
El constructor Base(String message) imprime:

Base constructor with message: Hello

Después de que el constructor de Base termine su ejecución, el constructor de Derived continúa y ejecuta su propia línea de impresión:

Derived constructor

7. ¿Cuál es la principal función de Jfrog Artifactory en un entorno de desarrollo de software?
R: Gestionar y almacenar artefactos de software, como dependencias y bibliotecas, de manera centralizada
La principal función de JFrog Artifactory en un entorno de desarrollo de software es actuar como un repositorio de artefactos para gestionar, almacenar y distribuir dependencias y paquetes de software de manera eficiente y segura a lo largo del ciclo de desarrollo y entrega continua (CI/CD).

8. ¿Cuál de los siguientes métodos de Mockito se utiliza para verificar que un método de un mock ha sido llamado un número específico de veces? (*)
R: verify() / verify(mock, times(n))

el método utilizado para verificar cuántas veces se ha llamado a un método de un **mock** es `verify()`, junto con `times(n)`.

9.
```
abstract class Animal {
 public abstract void makeSound();
}
class Dog extends Animal {
 @Override
 public void makeSound() {
 System.out.println("Bark!");
 }
}
public class Test {
 public static void main(String[] args) {
 Animal myDog = new Dog();
 myDog.makeSound();
 }
}
```
R: Bark!

Se esta llamando a myDog por lo que se imprime Bark

10: ¿Cuál es el propósito principal de la anotación @Test en JUnit?
R: Marcar un método como un método de prueba
Los métodos anotados con @Test son ejecutados automáticamente por JUnit, sin necesidad de llamarlos manualmente.
11. ¿Cuál de las siguientes características es fundamental en una base de datos relacional? R: Organización de datos en tablas con filas y columnas

12. Which line of code will compile successfully without any additional import statements?
```
public class Program{
 public static void main(String[] args) {
 // Line A
 String str = "Hello World!";
 // Line B
 ArrayList<String> list = new ArrayList<>();
 // Line C
 File file = new File("example.txt");
 // Line D
 URL url = new URL("http://example.com");
 }
}
```
R: Line A
En Java, la clase String no requiere un import explícito porque es parte de la librería estándar y se encuentra en el paquete java.lang
13:
```
abstract class Shape {
```

```
 public abstract void draw();
 public void printShape() {
 System.out.println("This is a shape");
 }
}
class Circle extends Shape {
 @Override
 public void draw() {
 System.out.println("Drawing a circle");
 }
}
public class Test {
 public static void main(String[] args) {
 Circle circle = new Circle();
 circle.draw();
 circle.printShape();
 }
}
```

R:Drawing a circle
This is a shape

La salida es la esperada, ya que el método draw() es implementado en la clase Circle, y el método printShape() es heredado de la clase abstracta Shape.

14: Cuál es el propósito principal del archivo pom.xml en un proyecto Maven?
R: Definir las dependencias, plugins y configuraciones del proyecto es gestionar la configuración y las dependencias del proyecto. Este archivo es fundamental para el proceso de construcción (build) y gestión de proyectos en Maven, y contiene una serie de configuraciones que controlan cómo se construye, prueba y despliega el proyecto.
15: Which section in the pom.xml file specifies the external libraries and dependencies required by the project?
R: <dependencies>
El archivo pom.xml especifica las dependencias necesarias para el proyecto, como bibliotecas y frameworks externos. Maven descargará automáticamente estas dependencias desde un repositorio central y las incluirá en el proyecto.
16. What will be the output of the following code snippet?

```
public class ScopeTest {
 private int value = 10;

 public void printValue() {
 int value = 20;
 System.out.println(this.value);
 }

 public static void main(String[] args) {
 ScopeTest test = new ScopeTest();
 test.printValue();
```

```
 }
}
```
R: 10

Campo value de la clase ScopeTest:

En la clase ScopeTest, hay un campo de instancia llamado value, que está inicializado en 10.

Método printValue():

Dentro del método printValue(), se declara una variable local llamada value, que está inicializada en 20. Esta variable local solo es accesible dentro del método y oculta al campo value de la clase.

Sin embargo, el uso de this.value en el println hace referencia al campo de instancia value, no a la variable local dentro del método. El prefijo this es utilizado para acceder a los miembros de instancia de la clase, en lugar de a las variables locales.

Llamada al método printValue() en main:

Cuando se ejecuta test.printValue(), dentro del método printValue() se imprime this.value, lo que se traduce en el valor del campo de instancia, que es 10

17: What is the primary purpose of a Data Transfer Object (DTO) in software design?
R: To transfer data between different layers or tiers of an application

El propósito principal de un Objeto de Transferencia de Datos (DTO) en el diseño de software es transferir datos entre diferentes capas de software o sistemas, generalmente a través de fronteras de red. Se utiliza para encapsular y transportar datos de manera eficiente entre distintas capas de una aplicación (por ejemplo, entre la capa de acceso a datos y la capa de presentación) o entre servicios en un sistema distribuido.

18: Which declaration correctly initializes a boolean variable in Java?
a. boolean f = "true";
b. boolean f = () => f;
c. boolean f = (1 + 0);
d. boolean d = (a < b);
e. boolean b = 0 < 1;
f. boolean a = (10 > 5 && 2 < 3);
R: e y f

opción c: incorrecta boolean f = (1 + 0);

Explicación:
La expresión 1 + 0 evalúa a un entero (1), no a un valor booleano, por lo que no se puede asignar directamente a una variable booleana.

19: Which of the following statements about the 'throw' keyword is true?
R: It is used to manually throw an exception

la palabra clave throw se utiliza para lanzar manualmente una excepción. Esto permite que el programador controle cuándo y dónde debe ocurrir una excepción dentro del flujo de ejecución del programa.

20. Which of the following code snippets will throw a ClassCastException

a.
```
class A {}
class B extends A {}
public class Test {
 public static void main(String[] args) {
 B obj = new B();
 A a = (A) obj;
 }
}
```

b.
```
class A {}
class B extends A {}
public class Test {
 public static void main(String[] args) {
 A obj = new B();
 A a = (A) obj;
 }
}
```

c.
```
class A {}
class B extends A {}
public class Test {
 public static void main(String[] args) {
 A obj = new B();
 B b = (B) obj;
 }
}
```

d.
```
class A {}
class B extends A {}
public class Test {
 public static void main(String[] args) {
 A obj = new A();
 B b = (B) obj;
 }
}
```

R: d.

El error ClassCastException ocurre cuando intentas convertir (cast) un objeto de un tipo a otro tipo incompatible.

En este caso, obj es de tipo A, pero no es una instancia de B (es simplemente un objeto de tipo A). Intentar hacer un cast de A a B generará una ClassCastException en tiempo de ejecución porque A no es una subclase de B.

21. ¿Cuál de las siguientes afirmaciones es correcta sobre la aserción assert() en jUnit?

R: Se utiliza para verificar que una condición es verdadera

La aserción assert() en JUnit se utiliza para verificar que una condición especificada sea verdadera en el contexto de las pruebas unitarias. Si la condición es falsa, JUnit marcará la prueba como fallida.

22. ¿Cual es la funcion principal del JDK (Java Development Kit)?
R: Ofrecer herramientas necesarias para compilar, depurar y ejecutar aplicaciones Java.

El JDK es un conjunto de herramientas y utilidades que los desarrolladores de Java utilizan para crear y gestionar aplicaciones. Contiene todo lo necesario para desarrollar programas en Java, incluyendo:

Compilador (javac):
Convierte el código fuente Java (archivos .java) a bytecode (archivos .class), que es entendido por la Java Virtual Machine (JVM).

JVM (Java Virtual Machine):
Permite ejecutar el bytecode de Java de manera independiente del sistema operativo.

Herramientas de depuración:
Proporciona herramientas como el depurador (jdb) para encontrar y corregir errores en el código.

Bibliotecas estándar:
Incluye las bibliotecas de clases estándar de Java (por ejemplo, clases para manipulación de cadenas, colecciones, entrada/salida, etc.).

Documentación (javadoc):
Genera documentación en formato HTML para las clases y métodos definidos en el código fuente.

Herramientas de gestión de paquetes y entornos de ejecución:
Proporciona herramientas como jar (para empaquetar archivos .class en archivos .jar) y otras utilidades relacionadas con la administración de aplicaciones Java.

23. Which of the following statements accurately describe the differences between Comparator and Comparable interfaces in Java? R: All of the above.

Explicación de las diferencias clave:

Comparable:
Propósito: Permite que los objetos de una clase sean ordenados de manera natural (por ejemplo, en una lista o un conjunto ordenado).
Método: Implementa el método compareTo(T o).
Uso: Solo se puede usar cuando se quiere que los objetos de la clase tengan un orden predeterminado, es decir, la clase misma define cómo se comparan sus objetos.

Modificación de la clase: La clase debe implementar la interfaz Comparable, por lo que se necesita modificar la clase directamente.
Comparator:

Propósito: Permite definir un orden personalizado para los objetos de una clase, incluso si esa clase no implementa Comparable.
Método: Implementa el método compare(T o1, T o2).
Uso: Es útil cuando se necesitan múltiples formas de ordenar los objetos de una clase o cuando no se puede modificar la clase para implementar Comparable.
Modificación de la clase: No se necesita modificar la clase de los objetos, ya que se utiliza una clase externa que define el orden.

24. What is the purpose of the "throws" keyword in a method declaration in Java?
R: To indicate the exceptions that the method can throw to the caller.

Cuando un método en Java puede generar una excepción que no se maneja dentro de ese método (es decir, una excepción comprobada o checked exception), se debe usar la palabra clave throws en la firma del método para declarar las excepciones que podría lanzar. Esto permite que el llamador del método esté al tanto de las excepciones posibles y tome medidas para manejarlas.
public void myMethod() throws IOException, SQLException { // Código que puede generar excepciones }

25. ¿Cuáles de los siguientes comandos de Git se utilizan para gestionar ramas en un repositorio? (Seleccione todas las que correspondan).
R: git checkout, git branch, git merge

Para gestionar ramas en Git, los comandos más utilizados son git branch para crear o listar ramas, git checkout o git switch para cambiar entre ellas, git merge o git rebase para fusionarlas, y git branch -d o git push origin --delete para eliminar ramas locales o remotas, respectivamente.

26. ¿Cuál de los siguientes patrones de diseño es adecuado para crear una estructura de objetos en forma de arbol para representar jerarquias parte-todo, permitiendo a los clientes tratar objetos individuales y compuestos de manera uniforme?
R: Patrón Compuesto (Composite Pattern).

El Patrón Compuesto es un patrón estructural que permite tratar de manera uniforme tanto a objetos individuales como a composiciones de objetos (conjuntos de objetos). Este patrón es útil cuando se tiene una estructura jerárquica, como un árbol, donde cada objeto puede ser tanto un objeto "hoja" (sin hijos) como un objeto compuesto (que tiene subcomponentes).
Características clave del Patrón Compuesto:

Objeto Compuesto: El patrón define una clase base o interfaz que puede ser implementada tanto por los objetos individuales (hojas) como por las composiciones (nodos internos del árbol).

Uniformidad: El patrón permite que los clientes interactúen con objetos individuales y compuestos de la misma manera, sin tener que distinguir entre ellos.

Jerarquía parte-todo: Los objetos pueden ser organizados en una jerarquía, donde cada objeto puede ser un componente de un objeto más grande o un objeto independiente.

27. Which file is used to configure user specific settings in Maven?
R: settings.xml
Maven utiliza este archivo para almacenar configuraciones a nivel de usuario, tales como los repositorios remotos, credenciales de acceso, configuraciones de proxy y otras opciones específicas del entorno de desarrollo de un usuario. Este archivo se encuentra en el directorio {HOME}/.m2/ (por ejemplo, C:\Users\<Usuario>\.m2\ en Windows o /home/<usuario>/.m2/ en Linux).

28. What will be the output of the following code snippets?

```
import java.util.ArrayList;
import java.util.List;
public class GenericTest {
 public static <T> void addIfAbsent(List<T> list, T element) {
 if (!list.contains(element)) {
 list.add(element);
 }
 }
 public static void main(String[] args) {
 List<String> items = new ArrayList<>();
 items.add("apple");
 items.add("banana");
 addIfAbsent(items, "cherry");
 addIfAbsent(items, "apple");
 System.out.println(items);
 }
}
```

R: [apple, banana, cherry]

items.add("apple");: Añade el elemento "apple" a la lista items.
items.add("banana");: Añade el elemento "banana" a la lista items.
addIfAbsent(items, "cherry");: El método addIfAbsent agrega "cherry" a la lista solo si no está presente en la lista. Como "cherry" no está en la lista, se agrega.
addIfAbsent(items, "apple");: El método addIfAbsent verifica si "apple" ya está en la lista. Como "apple" ya está presente, no se agrega nuevamente.
System.out.println(items);: Finalmente, imprime la lista, que ahora contiene los elementos: [apple, banana, cherry].

29. What will be the output of the following code snippet?

```
public class StringConcatenationTest {
```

```
 public static void main(String[] args) {
 String str1 = "Hello";
 String str2 = "World";
 String str3 = str1 + " " + str2;
 String str4 = str1.concat(" ").concat(str2);
 String str5 = new StringBuilder().append(str1).append("
").append(str2).toString();
 System.out.println(str1.equals(str2) + " ");
 System.out.println(str3.equals(str4) + " ");
 System.out.println(str3 == str5 + " ");
 System.out.println(str4 == str5);
 }
}
```

R: false true false false

false: str1 y str2 no son iguales.
true: str3 y str4 tienen el mismo contenido.
false: str3 y str5 son objetos diferentes (aunque el contenido sea el mismo).
false: str4 y str5 también son objetos diferentes (aunque el contenido sea el mismo).

30. Which of the following code snippets will result in a compilation error when implementing the vehicle interface?

```
interface Vehicle{
void start();
void stop(); //public abstract
}
```

R:

```
public class Bike implements Vehicle {
 public void start() { ... }
 void stop() { ... }
}
```

Porque void stop no es public.

31. What will be the output of the following code snippet?

```
public class StaticNonStaticBlockTest {
 static {
 System.out.println("Static block");
 }
 {
 System.out.println("Instance block");
 }
 public StaticNonStaticBlockTest() {
 System.out.println("Constructor");
 }
 public static void staticMethod() {
 System.out.println("Static method");
 }
 public static void main(String[] args) {
 StaticNonStaticBlockTest test = new StaticNonStaticBlockTest();
```

```
 new StaticNonStaticBlockTest();
 }
}
```

R:

```
Static block
Instance block
Constructor
Instance block
Constructor
```

el bloque estático se ejecuta una sola vez cuando se carga la clase, antes de cualquier instancia de la clase. En este caso, el mensaje "Static block" se imprime al cargar la clase StaticNonStaticBlockTest.

El bloque de instancia se ejecuta cada vez que se crea una nueva instancia de la clase, antes de la ejecución del constructor. En este caso, el primer mensaje "Instance block" se imprime cuando se crea la primera instancia de StaticNonStaticBlockTest.
Después de eso, el constructor se ejecuta y se imprime "Constructor"

Se crea una segunda instancia de StaticNonStaticBlockTest. Nuevamente, se imprime "Instance block" seguido por "Constructor".

32. En el contexto de bases de datos relacionales, si una transacción cumple con la propiedad de Aislamiento (Isolation) del principio ACID, esto significa que:
R: Las transacciones se ejecutan como si fueran la única operación en el sistema, sin interferencia de otras transacciones concurrentes.

En el contexto de bases de datos relacionales, si una transacción cumple con la propiedad de Aislamiento (Isolation) del principio ACID, esto significa que la ejecución de la transacción es independiente y no se ve afectada por otras transacciones que se estén ejecutando al mismo tiempo.

Es decir, aunque múltiples transacciones se puedan estar ejecutando simultáneamente, cada transacción debe ejecutarse como si fuera la única en el sistema, sin interferencia de otras. Las modificaciones realizadas por una transacción no deben ser visibles para otras transacciones hasta que la transacción esté completamente confirmada (committed).

33. Which of the following statements accurately describe the relationships that can exist between classes in Java?

a. Both inheritance and composition can be used together to model complex relationships.
b. Inheritance represents an "is-a" relationship where one class derives from another class.
c. Composition represents a "has-a" relationship where one class contains an instance of another class.
d. Composition should be preferred over inheritance to promote code reuse and flexibility.
e. Usage (or association) represents a "uses-a" relationship where one class uses methods or instances of another class.

f. Inheritance should be preferred over composition to promote code reuse and flexibility
R: a, b, c ,d, e

34. Todo el acceso a los datos debe estar encapsulado en una biblioteca para facilitar la reutilización y control de acceso.
R: Verdadero

35. En el contexto de Maven, ¿Cuál es la función principal del archivo settings.xml?
Configurar la información del repositorio local y remoto, así como las credenciales y perfiles de usuario.

En el contexto de Maven, la función principal del archivo settings.xml es configurar las propiedades globales y específicas del usuario para el funcionamiento de Maven. Este archivo permite personalizar la configuración de Maven, como los repositorios, las credenciales, las configuraciones de proxy, y las opciones de construcción.

36. Where do you configure the plugins used for various build tasks in Maven's pom.xml?
R: <build>

37. What will be the result of the following code execution?

```
import java.util.ArrayList;
import java.util.List;

public class ArrayListTest {
public static void main(String[] args) {
List<Integer> list = new ArrayList<>();
list.add(1);
list.add(2);
list.add(3);
list.remove(1);
System.out.println(list);
}
}
```

R: [1, 3]

Se crea una lista de enteros (list) y se añaden los valores 1, 2, 3.
Se elimina el elemento en el índice 1, que es el valor 2.
Al imprimir la lista, los elementos restantes son 1 y 3, por lo que se imprime [1, 3].

38. Which of the following methods can be used to remove all elements from an ArrayList?
R: clear()
Para eliminar todos los elementos de un ArrayList en Java, se pueden usar los siguientes métodos:

clear(): Este método elimina todos los elementos de la lista. La lista queda vacía después de su ejecución.

removeAll(Collection<?> c): Este método elimina todos los elementos que estén presentes en la colección especificada. Si se pasa una colección que contiene todos los elementos de la lista, la lista se vaciará.

39. ¿Cuál es la principal desventaja del antipatrón "contenedor mágico" en el desarrollo de software?
R: Oculta demasiada lógica de negocio en un contenedor genérico, lo que hace que el código sea difícil de entender y depurar.

La principal desventaja del antipatrón "contenedor mágico" (también conocido como "God Object" o "Objeto Dios") en el desarrollo de software es que crea una clase u objeto que tiene demasiadas responsabilidades y que maneja un gran número de tareas y funcionalidades diferentes. Esto puede resultar en varios problemas:

Dificultad para mantener y extender el código: Cuando un objeto o clase tiene demasiadas responsabilidades, se vuelve difícil de comprender, mantener y modificar sin afectar otras partes del sistema.

Violación del principio de responsabilidad única (SRP): Un contenedor mágico suele violar el principio de responsabilidad única, que establece que cada clase debe tener una única razón para cambiar. Al tener muchas responsabilidades, se hace más probable que los cambios en una parte del código afecten a otras áreas, lo que puede generar errores inesperados.

Acoplamiento elevado: Este tipo de objetos tiende a tener un alto acoplamiento con muchas otras clases del sistema, lo que dificulta la reutilización, las pruebas y la modificación de cualquier parte del sistema sin afectar otras partes.

Poca flexibilidad y escalabilidad: A medida que el sistema crece, el contenedor mágico se convierte en un punto de fricción, limitando la capacidad de escalabilidad del sistema y la flexibilidad para adaptarse a nuevos requerimientos.

40. ¿Cuál de las siguientes afirmaciones describe mejor un Step en el contexto de Spring Batch?

R: Un objeto de dominio que encapsula una fase independiente y secuencial de un trabajo por lotes.

un Step representa una unidad de trabajo independiente dentro de un Job. Es decir, cada Step define una parte específica del procesamiento de datos dentro de un flujo de trabajo más amplio.

Un Step en Spring Batch generalmente sigue la estructura de lectura, procesamiento y escritura de datos (Read-Process-Write). Puede incluir operaciones como la validación de datos, transformación, filtrado o carga en una base de datos.

41. Which method override is valid given the following classes?

```
class Parent {
void display() {
System.out.println("Parent");
}
}
class Child extends Parent {
// Override here
}
```

R:
```
public void display() { System.out.println("Child");
}
```

42. ¿Cuál es la rama principal de Github en la que se integran las nuevas funcionalidades antes de lanzarlas a producción?
R: develop

La rama principal en la que se integran las nuevas funcionalidades antes de lanzarlas a producción en GitHub suele ser:
develop (en flujos como Git Flow)

Sin embargo, en muchos proyectos que siguen una estructura más simple:
main o master también pueden ser utilizadas para integrar cambios antes de un lanzamiento.

43. ¿Cuál es el comando en Bash para cambiar el directorio actual a uno especificado?
R: cd

44. Which of the following is NOT part of the Agile Software development lifecycle?
R: Documenting
Fases del Ciclo Ágil:

- Concepto e Iniciación – Identificación de necesidades y alcance.
- Planificación – Creación y priorización del backlog.
- Diseño – Arquitectura y prototipos.
- Desarrollo – Programación iterativa en sprints.
- Pruebas – Validación y corrección de errores.
- Entrega – Despliegue y feedback del usuario.
- Mantenimiento – Optimización y mejoras continuas.