**Question 1:** Which of the following statements is true about arrays in Java?

- a. An array has a fixed size.
- b. An array allows multiple dimensions.
- c. An array is mutable.
- d. An array is immutable.

**Question 2:** ¿Cuál es el comando en Bash para cambiar el directorio actual a uno especificado?

- a. cd
- b. sh
- c. move
- d. goto
- e. chdir

Por qué las otras opciones son incorrectas:

- sh: Este comando se utiliza para ejecutar scripts de shell. No sirve para cambiar de directorio.
- move: Este comando suele utilizarse en sistemas operativos Windows para mover archivos o directorios, no para cambiar el directorio actual.
- goto: No existe un comando llamado goto en Bash para cambiar de directorio.
- chdir: Si bien chdir es una función en algunos lenguajes de programación para cambiar el directorio de trabajo, no es un comando estándar en Bash.

**Question 3:** Which of the following statements is true about arrays in Java?

- a. An array has a fixed size.
- b. An array allows multiple dimensions.
- c. An array is mutable.
- d. An array is immutable.

**Question 4:** ¿Cuál de los siguientes componentes es parte de una solicitud HTTP?

- a. Headers, scripts, funciones.
- b. URL, headers, cuerpo de la solicitud.
- c. Encabezados de la respuesta, cuerpo de la respuesta, estado de la respuesta.
- d. URL, cookies, archivos adjuntos.

Por qué las otras opciones son incorrectas:

**Encabezados de la respuesta, cuerpo de la respuesta, estado de la respuesta:**
Estos son componentes de una **respuesta HTTP**, que es lo que el servidor envía al cliente en respuesta a una solicitud.

**URL, cookies, archivos adjuntos:**
Las cookies son pequeñas piezas de datos que se almacenan en el navegador y se envían con cada solicitud, pero no son la parte principal de una solicitud HTTP. Los archivos adjuntos suelen ser parte del cuerpo de la solicitud, pero no definen completamente la estructura de una solicitud.

**Headers, scripts, funciones:**
Los scripts y funciones son elementos del lado del cliente (generalmente JavaScript) y no forman parte de una solicitud HTTP.

**Question 5:** What is the output of the above Java code?

```
 public class LoopQuestion {
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);
}
}
```

- a. Count: 4
- b. Count: 6
- c. Count: 10
- d. Compilation fails
- e. Count: 5

**Question 6:** Which method is used to sort elements of a List in natural order in Java?

- a. Collections.sort()
- b. Collections.order()
- c. Arrays.sort()
- d. List.sort()

**Collections.sort()** is a more general method that can be used to sort various collections, including Lists. However, it's generally recommended to use List.sort() for sorting Lists directly.

**Collections.order()** is not a standard method in the Java Collections Framework.

**Arrays.sort()** is used to sort arrays, not Lists.

**Question 7:** ¿Qué significa que un cambio en el software sea retrocompatible?

- a. El cambio introduce nuevas funcionalidades que no afectan el comportamiento existente del software.
- b. El cambio corrige errores menores y realiza mejoras de rendimiento.
- c. El cambio puede romper la funcionalidad existente, requiriendo ajustes en el código que depende del software.
- d. El cambio garantiza que el software siga funcionando sin necesidad de modificaciones en el código que depende de él.

**El cambio introduce nuevas funcionalidades que no afectan el comportamiento existente del software**
Esto es posible, pero no garantiza la retrocompatibilidad. Si las nuevas funcionalidades interactúan con las antiguas, podrían causar conflictos.

**El cambio corrige errores menores y realiza mejoras de rendimiento**
Al igual que la opción anterior, esto puede ser parte de una actualización retrocompatible, pero no es suficiente por sí solo.

**El cambio puede romper la funcionalidad existente, requiriendo ajustes en el código que depende de él**
Si un cambio requiere ajustes en el código existente, entonces no es retrocompatible.

**Question 8:** Which statement about the following code is correct?

```
 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");
 }
}

class Test {
   public static void main(String[] args) {
        Derived derived = new Derived();
  }
}
```

- a. It will throw a compilation error.
- b. It will print “Base constructor” followed by “Derived constructor”
- c. It will print “Base constructor with message: Hello” followed by “Derived constructor”
- d. It will print “Base constructor” followed by “Base constructor with message: Hello” followed by “Derived constructor”

**Question 9:** ¿Cuál es la principal función de Ifrog Artifactory en un entorno de desarrollo de software?

- Gestionar y almacenar artefactos de software, como dependencias y bibliotecas, en un repositorio centralizado.
- Proporcionar un entorno de desarrollo integrado (IDE) para aplicaciones Java.
- Actuar como un servidor web para alojar sitios HTML y CSS.
- Ofrecer un sistema de control de versiones para proyectos de software.

**Question 10:** ¿Cuál es la nomenclatura y la ruta de creación de un archivo que sigue el formato (UUID[CODE].VERSION[COUNTRY].xml en un proyecto?

- VERSION.xml y se crea en la ruta src/main/resources/
- Ninguna opción es correcta
- COUNTRY[CODE(UUID].xml y se crea en la ruta src/test/resources/
- [CODE]TUUAA[COUNTRY].xml y se crea en la ruta src/main/java/
- [UUIDCODE[COUNTRY].xml y se crea en la ruta src/main/resources/

**Answer 10:** a. VERSION.xml y se crea en la ruta src/main/resources/

**Question 11:** ¿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?

- when() & thenReturn()
- doReturn()
- doThrow()
- Ninguna opción es correcta.
- verify()

**when() & thenReturn():** Estos métodos se utilizan para configurar el comportamiento de un mock, es decir, para definir qué valor debe devolver un método cuando es llamado. No sirven para verificar las interacciones.

**doReturn():** Similar a when() & thenReturn(), se usa para configurar el comportamiento de un mock, pero en casos más complejos.

**doThrow():** Se utiliza para configurar un mock para que lance una excepción cuando un método es llamado.

**Question 12:** Given the following classes, what will be the output of the program?

```
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();
         }
}
```

- No output
- Bark
- Runtime error
- Compilation error

**Question 13:** ¿Cuál de las siguientes características es fundamental en una base de datos relacional?

- Almacenamiento de datos en un sistema de archivos distribuido.
- Ninguna opción es correcta.
- Utilización de nodos y relaciones para representar datos.
- Almacenamiento de datos en formato JSON.
- Organización de datos en tablas con filas y columnas.

**Almacenamiento de datos en un sistema de archivos distribuido:** Aunque algunas bases de datos relacionales pueden distribuirse en múltiples servidores, la característica fundamental sigue siendo la organización en tablas.

**Utilización de nodos y relaciones para representar datos:** Esta característica es más propia de las bases de datos gráficas o no relacionales (como Neo4j), donde los datos se representan como nodos conectados por relaciones.

**Almacenamiento de datos en formato JSON:** JSON es un formato de intercambio de datos, no una estructura de almacenamiento de datos relacional. Las bases de datos relacionales suelen utilizar formatos internos propios.

**Question 14:** Which line of code will compile successfully without any additional import statements?

```
public class PackageTest {
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");
}
}
```

- Line C
- Line D
- Line A
- Line B

String is a built-in class present in the java.lang package. All Java programs have implicit access to this package, so you don't need to import it explicitly.

ArrayList belongs to the java.util package. To use it, you need to import it using import java.util.ArrayList;

File is also in the java.util package, requiring an import statement.

URL resides in the java.net package, necessitating an import statement.

**Question 15:** Which of the following statements is true about the following code?

```
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();
}
}
```

- Compilation error because the Shape class is abstract
- The program will print "This is a shape" followed by "Drawing a circle"
- The program will print "Drawing a circle" followed by "This is a shape"
- Compilation error because the Circle class does not implement the draw method.

**Question 16**

Una transacción APX es la unidad aplicativo que se ejecutará en APX Batch.

- Verdadero
- Falso

**Question 17:** ¿Cuál es el propósito principal del archivo pom.xml en un proyecto Maven?

- Gestionar la interfaz gráfica de usuario del proyecto.
- Definir las dependencias, plugins y configuraciones del proyecto.
- Contener la documentación del código fuente del proyecto.
- Almacenar configuraciones de la base de datos del proyecto.

***Dependencias:** Especifica las bibliotecas externas que necesita el proyecto, como frameworks, utilidades, etc. Maven se encarga de descargar y gestionar estas dependencias automáticamente.

***Plugins**: Define los plugins que se utilizarán para realizar tareas específicas durante el ciclo de vida del proyecto, como compilar, empaquetar, ejecutar pruebas, etc.

***Configuraciones:** Permite personalizar la construcción del proyecto, como la versión del compilador, la codificación de los archivos, etc.

**Gestionar la interfaz gráfica de usuario del proyecto:** Esta tarea generalmente se realiza utilizando frameworks UI específicos y no es la función principal del pom.xml.

**Contener la documentación del código fuente del proyecto:** Aunque el pom.xml puede contener información sobre la documentación, no es su función principal. La documentación del código se suele generar a partir de comentarios en el código o utilizando herramientas de generación de documentación.

**Almacenar configuraciones de la base de datos del proyecto:** Las configuraciones de la base de datos suelen almacenarse en archivos de propiedades o en variables de entorno, no en el pom.xml.

**Question 18:** Which section in the pom.xml file specifies the external libraries and dependencies required by the project?

- <build>
- <dependencies>
- <repositories>
- <plugins>

**<build>:** This section defines the build process, including plugins for compilation, testing, packaging, etc.

**<repositories>:** This section defines the repositories where Maven should look for dependencies, such as Maven Central or other private repositories.

**<plugins>:** This section defines the plugins that will be used during the build process, such as the Maven Compiler Plugin, the Surefire Plugin for testing, and the Jar Plugin for packaging.

**Question 19:** 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();
}
}
```

- 20
- Compilation Error
- 10
- Runtime error.

**Question 20:** Which of the following is NOT part of the Agile Software development lifecycle?

- Testing
- Coding
- Planning
- Documenting

**Planning**: Abarca la **Definición del producto, Priorización** y **Planificación del sprint.**

Coding: Parte del **Desarrollo**

Testing: Parte de la **Revisión**

**Retrospectiva:** Es un Análisis en el equipo reflexiona sobre lo que fue bien, lo que podría mejorar y qué acciones se pueden tomar para mejorar el proceso en el siguiente sprint.

**Question 21:** What is the primary purpose of a Data Transfer Object (DTO) in Software Design?

- To provide a user interface to the data model
- To encapsulate the data flow within the application.
- To handle addition operations directly
- To handle data flow between different layers of the application

**Question 22:** Which declaration correctly initializes a boolean variable in Java?

- boolean f = "true";
- boolean c = (7 > 7);
- boolean a = (3 < 6);
- boolean d = (4 != 4);
- boolean b = (5 == 2);
- boolean e = (10 > 5 && 2 < 3);

**Question 23:** Which of the following statements about the 'throw' keyword is true?

- It is used to define the cleanup code that must be executed.
- It is used to declare that a method can throw an exception.
- It is used to catch exceptions thrown by other methods.
- It is used to manually throw an exception.

**Question 24**

Which of the following code snippets will throw a ClassCastException at runtime?

```
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 A();
B b = (B) obj;
}
}

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

**Question 25:** ¿Cuál de las siguientes afirmaciones es correcta sobre la aserción assertEquals en JUnit?

- Se utiliza para verificar que una colección contiene un elemento específico.
- Se utiliza para verificar que dos valores son iguales.
- Se utiliza para verificar que una condición es verdadera.
- Se utiliza para verificar que dos objetos referencian la misma instancia.

**Por qué las otras opciones son incorrectas:**

- **Se utiliza para verificar que una colección contiene un elemento específico:** Para esto, se utilizan métodos como assertTrue(collection.contains(elemento)).
- **Se utiliza para verificar que una condición es verdadera:** Para esto, se utiliza assertTrue.

- **Se utiliza para verificar que dos objetos referencian la misma instancia:** Esto describe la función **assertSame**, que verifica que dos referencias apuntan al mismo objeto.

**Question 26:** ¿Cuál es la función principal del JDK (Java Development Kit)?

- Servir como un servidor web para aplicaciones Java.
- Proporcionar un entorno de ejecución para aplicaciones Java.
- Ofrecer herramientas necesarias para compilar, depurar y ejecutar aplicaciones Java.
- Ninguna opción es correcta.
- Permitir la edición de archivos HTML y CSS.

**Por qué las otras opciones son incorrectas:**

- **"Servir como un servidor web para aplicaciones Java":**
  - Esto describe el propósito de un servidor de aplicaciones, como Apache Tomcat o JBoss, no el JDK.
- **"Proporcionar un entorno de ejecución para aplicaciones Java":**
  - Esto es tarea de la **JVM (Java Virtual Machine)**, que forma parte del **JRE (Java Runtime Environment)**, no del JDK completo.
- **"Ninguna opción es correcta":**
  - Incorrecto, porque la opción seleccionada describe adecuadamente el propósito del JDK.
- **"Permitir la edición de archivos HTML y CSS":**
  - Esto es tarea de editores o IDEs, como VS Code o IntelliJ IDEA, no del JDK.

**Question 27:** Which of the following statements accurately describe the differences between Comparator and Comparable interfaces in Java?

- Comparable must be implemented by the class whose objects are being compared, whereas Comparator can be implemented by any class
- Comparable defines the compareTo method, whereas Comparator defines the compare method.
- Comparator allows for multiple ways of comparing objects, while Comparable allows only one way of comparing objects.
- All of the above.
- Comparable is used to compare the natural ordering of objects, whereas Comparator is used for custom ordering.

**Question 28:** What is the purpose of the “throws” keyword in a method declaration in Java?

- To catch exceptions thrown by other methods.
- To indicate the exceptions that the method can throw to the caller.
- To create a new exception instance.
- To throw an exception within the method.

**Question 29:** ¿Cuáles de los siguientes comandos de Git se utilizan para gestionar ramas en un repositorio? (Seleccione todas las que correspondan).

- git commit
- git merge
- git checkout
- git branch
- git init

**Los comandos de Git que se utilizan para gestionar ramas en un repositorio son:**

- **git branch:** Este es el comando principal para trabajar con ramas. Permite crear nuevas ramas, listar las ramas existentes, eliminar ramas y cambiar entre ellas.
- **git checkout:** Se utiliza para cambiar de una rama a otra. También se puede usar para crear una nueva rama a partir de una existente.
- **git merge:** Este comando se emplea para combinar los cambios de una rama en otra. Es decir, se utiliza para fusionar dos ramas.

**Los comandos que no están directamente relacionados con la gestión de ramas son:**

- **git commit:** Este comando se usa para guardar los cambios realizados en los archivos y crear un nuevo commit.
- **git init:** Este comando se utiliza para inicializar un nuevo repositorio de Git en un directorio.

**Question 30:** ¿Cuál de los siguientes patrones de diseño es adecuado para crear una estructura de objetos en forma de árbol para representar jerarquías parte-todo, permitiendo a los clientes tratar objetos individuales y compuestos de manera uniforme?

- Ninguna opción es correcta.
- Patrón Adaptador (Adapter Pattern).
- Patrón Fachada (Facade Pattern).
- Patrón Compuesto (Composite Pattern).
- Patrón Estrategia (Strategy Pattern).

**Por qué las otras opciones son incorrectas:**

1. **Patrón Adaptador (Adapter Pattern):**
    - Se utiliza para convertir la interfaz de una clase en otra interfaz que el cliente espera, facilitando la compatibilidad entre clases con interfaces diferentes.
2. **Patrón Fachada (Facade Pattern):**
    - Proporciona una interfaz simplificada para un conjunto de subsistemas complejos, pero no trata con estructuras jerárquicas.
3. **Patrón Estrategia (Strategy Pattern):**
    - Define un conjunto de algoritmos intercambiables, permitiendo cambiar dinámicamente el comportamiento de un objeto, pero no se relaciona con jerarquías parte-todo.

**Question 31:** Which file is used to configure user specific settings in Maven?

- settings.xml
- build.xml
- pom.xml
- user.xml

- **settings.xml**:
  Este archivo se utiliza para configurar ajustes específicos del usuario en Maven, como las ubicaciones de los repositorios, configuraciones de proxy y otros parámetros específicos del entorno. Normalmente se encuentra en el directorio personal de Maven del usuario, que suele ser ~/.m2/settings.xml en sistemas Unix o %USER_HOME%\.m2\settings.xml en Windows.

- **build.xml**:
  Este archivo se utiliza en Apache Ant, no en Maven. Define configuraciones de construcción para proyectos gestionados con Ant.

- **pom.xml**:
  Este es el archivo de modelo de objeto de proyecto (Project Object Model) en Maven. Contiene configuraciones específicas del proyecto, como dependencias, plugins y perfiles de construcción.

- **user.xml**:
  Este archivo no se utiliza en Maven y no forma parte de la estructura estándar de configuración de Maven.

**Question 32:** 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); } }
```

- [apple, banana, cherry]
- [banana, cherry]
- [apple, banana]
- [apple, banana, cherry, apple]

**Question 33:** 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);

           }

}
```

- Compilation fails
- false true false false
- false false false false
- true true false false
- false false true true
- true true true true

**Question 34:** Which of the following code snippets will result in a compilation error when implementing the vehicle interface?

```
interface Vehicle{

  void start();

  void stop();

}
```

a.
```
public class Bike implements Vehicle {
 public void start(){
 System.out.println("Bike starts");
 }
 void stop(){
 System.out.println("Bike stops");
 }
 }
```

b.
```
public class Truck implements Vehicle {
 public void start(){
 System.out.println("Truck starts");
 }
 public void stop(){
 System.out.println("Truck stops");
 }
 public void load(){
 System.out.println("Truck loads");
 }
 }
```

c.
```
public class Scooter implements Vehicle {
 public void start(){
 System.out.println("Scooter starts");
 }
 public void stop(){
 System.out.println("Scooter stops");
 }
 }
```

d. public class Car implements Vehicle {
public void start(){
System.out.println("Car starts");
}
public void stop(){
System.out.println("Car stops");
}
}

**Question 35:** 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();

      }

}
```

a. Static block
Instance block
Constructor
Static method

b. Static block
Static method
Instance block
Constructor

c. Compilation error

d. Static method
Static block
Instance block
Constructor

**Question 36:** Una transacción APL representa un proceso funcional ACID donde se ejecuta un conjunto de operaciones lógicas.

Seleccione una:

- Verdadero
- Falso

La afirmación "Una transacción APL representa un proceso funcional ACID donde se ejecuta un conjunto de operaciones lógicas" es incorrecta.

Explicación:

- APL (A Programming Language): Es un lenguaje de programación de alto nivel conocido por su sintaxis concisa y su enfoque en operaciones matriciales. No está directamente ligado al concepto de transacciones ACID, que se relaciona más con bases de datos y sistemas de gestión de bases de datos.
- Transacciones ACID: Las transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) son un conjunto de propiedades que garantizan la integridad y consistencia de los datos en una base de datos relacional. Se utilizan para asegurar que las operaciones en una base de datos se ejecutan de manera confiable y segura.

Por qué la afirmación es falsa:

- APL no es una base de datos: APL es un lenguaje de programación, no un sistema de gestión de bases de datos. Por lo tanto, no maneja transacciones ACID de forma nativa.

- Enfoque diferente: APL está diseñado para realizar cálculos matemáticos y manipulación de datos de manera eficiente, mientras que las transacciones ACID se centran en garantizar la integridad de los datos en un contexto de base de datos.

En resumen:

Si bien APL puede utilizarse para realizar operaciones sobre datos que eventualmente podrían ser almacenados en una base de datos que soporta transacciones ACID, el lenguaje en sí mismo no implementa directamente este concepto.

**Question 37:** Which of the following statements accurately describe the relationships that can exist between classes in Java?

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

**Question 38:** Todo el acceso a los datos debe estar encapsulado en una biblioteca para facilitar la reutilización y control de acceso.

Seleccione una:

- Verdadero
- Falso

**Question 39:** ¿Cuál es el marco de trabajo en el que se basa APIX Batch?

- Spring Cloud
- Spring Microservices
- Spring Job
- Spring Batch
- Spring Web

**Spring Cloud y Spring Microservices:** Se centran en la construcción de arquitecturas distribuidas y microservicios, mientras que Spring Batch está diseñado específicamente para el procesamiento por lotes.

**Spring Job:** No es un marco de trabajo independiente, sino más bien un concepto dentro de Spring Batch.

**Spring Web:** Se utiliza para desarrollar aplicaciones web, no para procesamiento por lotes.

**Question 40:** En el contexto de Maven, ¿Cuál es la función principal del archivo settings.xml?

- Ninguna opción es correcta.
- Generar informes de construcción y documentación.
- Especificar las dependencias del proyecto.
- Configurar la información del repositorio local y remoto, así como las credenciales y perfiles de usuario.
- Definir la estructura de directorios del proyecto.

**Generar informes de construcción y documentación:** Esta tarea se realiza principalmente utilizando plugins como Maven Site Plugin, que se configuran dentro del archivo pom.xml de cada proyecto.

**Especificar las dependencias del proyecto:** Las dependencias específicas de un proyecto se declaran en el archivo pom.xml de ese proyecto. El archivo settings.xml proporciona una configuración global que se aplica a todos los proyectos.

**Definir la estructura de directorios del proyecto:** La estructura de directorios de un proyecto Maven está definida por convenciones estándar y puede ser ligeramente personalizada en el pom.xml. El settings.xml no se utiliza para esta configuración.

**Question 41:** Where do you configure the plugins used for various build tasks in Maven's pom.xml?

- <repositories>
- <plugins>
- <dependencies>
- <build>

- <repositories>:
    - Esta sección se utiliza para configurar repositorios remotos donde Maven debe buscar dependencias y plugins.
    - No se utiliza directamente para configurar los plugins en sí mismos.
    - Aquí se declaran las URLs de los repositorios, como Maven Central, repositorios privados, etc.

- <plugins>:
    - Esta sección se encuentra dentro de la sección `<build>` del archivo `pom.xml`.
    - No puede existir como un elemento de nivel superior por sí solo.
    - Aquí se definen los plugins que se utilizarán durante el proceso de construcción del proyecto, como el plugin de compilación (`maven-compiler-plugin`), el plugin de pruebas unitarias (`maven-surefire-plugin`), etc.

- <dependencies>:
    - Esta sección se utiliza para definir las dependencias del proyecto, es decir, las bibliotecas externas que necesita el proyecto para funcionar correctamente.
    - No se utiliza para configurar los plugins del proceso de construcción.

En resumen:

- <repositories>: Define dónde buscar dependencias y plugins.
- <plugins>: Define los plugins que se utilizarán durante la construcción (dentro de `<build>`).
- <dependencies>: Define las bibliotecas externas que necesita el proyecto.

**Question 42:** 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);

            }

}
```

- [1, 2, 3]
- [1, 3]
- [1, 2]
- [2, 3]

**Question 43:** Which of the following methods can be used to remove all elements from an ArrayList?

- removeAll()
- clear()
- eraseAll()
- deleteAll()

**Question 44:** ¿Cuál de las siguientes opciones son capacidades ofrecidas en APIX?

- Todas las opciones son correctas.
- Sin integración con servicios de seguridad.
- Cold Deployment.
- Procesamiento transaccional.
- Procesamiento batch.

**Question 45:** ¿Cuáles son los tipos de componentes en APIX?

- Librerias
- Todas las opciones son correctas
- DTOS
- Transacciones
- Jobs

**Question 46:** ¿Cuál es la principal desventaja del antipatrón "contenedor mágico" en el desarrollo de software?

- Utiliza un número excesivo de patrones de diseño, complicando la estructura del código.
- Introduce dependencias circulares que son difíciles de resolver.
- Oculta demasiada lógica de negocio en un contenedor genérico, lo que hace que el código sea difícil de entender y depurar.
- Depende en gran medida de servicios externos, lo que reduce la portabilidad del software.

**Question 47:** ¿Cuál de los siguientes componentes de APIX representa una entidad comercial en forma de un bean?

- Transacciones
- Librerías
- Jobs
- DTOs
- Todas las opciones son correctas.

**Question 48:** ¿Cuál de las siguientes afirmaciones describe mejor un Step en el contexto de Spring Batch?

- Un objeto de dominio que encapsula una fase independiente y secuencial de un trabajo por lotes.
- Una interfaz que define los métodos para realizar operaciones CRUD en los datos del trabajo por lotes.
- Una clase que gestiona la configuración de la base de datos utilizada por un trabajo por lotes.
- Un componente que se encarga exclusivamente de la validación de datos en un trabajo por lotes.

En Spring Batch, un Step representa una unidad de trabajo independiente y secuencial dentro de un proceso por lotes. Un Step encapsula la lógica necesaria para ejecutar una fase específica de un Job.

**Question 49:** Which method override is valid given the following classes?

```
    class Parent {

        void display() {

      System.out.println("Parent");

        }

}

class Child extends Parent {

  // Override here

}
```

- public void display() { System.out.println("Child"); }
- private void display() { System.out.println("Child"); }
- void display() { System.out.println("Child"); }
- static void display() { System.out.println("Child"); }

**Question 50:** ¿Cuál es la rama principal de Gitflow en la que se integran las nuevas funcionalidades antes de lanzarlas a producción?

- feature
- develop
- release
- hotfix
- master

**develop:** Esta rama es el tronco principal donde se integra todo el desarrollo activo. Las nuevas funcionalidades se crean en ramas de feature y luego se fusionan en develop. Una vez que se considera que develop está lo suficientemente estable, se crea una rama de release para preparar el lanzamiento.

**feature:** Estas ramas se crean para desarrollar funcionalidades específicas y se fusionan en develop cuando están completadas.

**release:** Se crea a partir de develop cuando se está preparando una nueva versión para su lanzamiento. Se utilizan para realizar pequeñas correcciones o ajustes antes de la liberación.

**hotfix:** Se crean directamente a partir de master para solucionar errores críticos en producción.

**master:** Contiene el código de la última versión estable y lista para producción.