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

---

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

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);
 }
}
```
Los valores que cumplen la condición son 0, 2, 4, 6, 8, por lo tanto la suma llegaría a 5
R:Count 5

---

4. Which method is used to sort elements of a List in natural order in Java? (*)
R:Collections.sort()
Para ordenar elementos de una Lista en orden natural, se utiliza el método Collections.sort() de la clase Collections o el método sort() de la interfaz List.

Ejemplo:

```
List<String> names = new ArrayList<>();
Collections.sort(names);
```

O bien,

```
List<String> names = new ArrayList<>();
names.sort(null); //Cuando pasas null como argumento al método sort(), estás indicando que quieres usar el ordenamiento natural de los elementos (el comparador por defecto).
```

---

5. ¿Qué significa que un cambio en el software sea retro-compatible?

La retro-compatibilidad (o compatibilidad hacia atrás/backwards compatibility) significa que una nueva versión de software mantiene la capacidad de:

1. Funcionar con versiones anteriores del mismo software.
2. Mantener la compatibilidad con datos, archivos y configuraciones creados en versiones previas.
3. No romper las funcionalidades que los usuarios ya estaban utilizando.

R: El cambio garantiza que el software será funcional sin necesidad de modificaciones en el código que depende de él.

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

Primero nos vamos hasta el constructor de la superclase, después recibe como parámetro el “Hello” y por último, volvemos a Derived.

---

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.

JFrog Artifactory es un repositorio universal para almacenar y gestionar todos los artefactos de software y dependencias del proyecto.

Sus funciones principales son:

1. Gestión de Dependencias.
2. Control de Versiones Binario.
3. Integración con CI/CD.
4. Seguridad.
5. Cache y Proxying.

---

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))

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!

Dog sobrescribe el método makeSound()

10: ¿Cuál es el propósito principal de la anotación @Test en JUnit?
R: Marcar un método como un método de prueba

---

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

Todas las demás se deben importar.

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

Como no hay nada en sus constructores, nos vamos a los métodos, primero draw() que fue sobreescrito y después printShape() que sólo lo tiene su padre.

14: Cuál es el propósito principal del archivo pom.xml en un proyecto Maven?
R: Definir las dependencias, plugins y configuraciones del proyecto

---

15: Which section in the pom.xml file specifies the external libraries and dependencies required by the project?
R: <dependencies>

---

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

Debido al this en print, se imprimirá la variable de referencia establecida en la clase y no el value que sólo vive en el bloque del método.

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

---

18: Which declaration correctly initializes a boolean variable in Java?
a. boolean f = "true"; //no, porque es String
b. boolean f = () => f; // no, porque es lambda
c. boolean f = (1 + 0); //El resultado es un número (1), no un booleano
d. boolean d = (a < b); // sólo si a y b estuviesen declaradas previamente y fuesen primitivos y no objetos
e. boolean b = 0 < 1; // Es una comparación que resulta en un booleano
f. boolean a = (10 > 5 && 2 < 3); // Es una expresión booleana compuesta
R: e y f

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

1. `throw`:

- Se usa para lanzar explícitamente una excepción
- Va dentro del código del método

2. `throws`:

- Se usa en la declaración del método
- Indica que el método puede lanzar ciertas excepciones
- Delega el manejo de la excepción al código que llama al método
- Puede listar múltiples excepciones

---

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; //Es un upcasting seguro - B es una subclase de A, el cast es innecesario aquí.
 }
}
b. class A {}
class B extends A {}

```
public class Test {
 public static void main(String[] args) {
 A obj = new B();
 A a = (A) obj; //Es un cast del mismo tipo (A a A). El objeto ya es de tipo A, por lo que el cast es redundante

 }
}
```

c.
```
class A {}
class B extends A {}
public class Test {
 public static void main(String[] args) {
 A obj = new B();
 B b = (B) obj; //Es un downcasting seguro. El objeto realmente es una instancia de B. Aunque está referenciado como A, contiene un B
 }
}
```

d.
```
class A {}
class B extends A {}
public class Test {
 public static void main(String[] args) {
 A obj = new A();
 B b = (B) obj; //Es un downcasting inseguro. Intenta convertir un objeto que es solo A en B. No se puede convertir un objeto padre en hijo si no fue creado como hijo

 }
}
```

R: d.

---

21. ¿Cuál de las siguientes afirmaciones es correcta sobre la aserción assert() en jUnit?
R: Se utiliza para verificar que una condición es verdadera

---

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

---

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

R: All of the above.

Tanto Comparator como Comparable son interfaces utilizadas para ordenar objetos.

Característica	Comparable	Comparator
Paquete	`java.lang`	`java.util`
Método	`compareTo(T o)`	`compare(T o1, T o2)`
Dónde se define el orden	Dentro de la clase	Fuera de la clase
Número de criterios de ordenación	Uno solo (natural)	Múltiples (personalizados)
Uso recomendado	Cuando hay un único criterio de orden	Cuando se requieren diferentes criterios

---

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.

---

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

---

26. ¿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?
R: Patrón Compuesto (Composite Pattern).

---

27. Which file is used to configure user specific settings in Maven?
R: settings.xml

---

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]

Apple ya no se mete nuevamente porque aunque sea List y permita elementos duplicados, en el método se especifica que sólo si no lo contiene
if (!list.contains(element))

---

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

Cuando son Strings, éste sobrescribe equals y revisa que el contenido sea igual, pero en el caso de StringBuilder revisa que apunten al mismo objeto, por lo que resulta falso.

---

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() { ... }

//siempre que usemos una interface es necesario ponerle el public al sobrescribir, ya que la accesibilidad no se puede reducir, y si no ponemos public, éste sería default, por lo que estaríamos reduciendo.
}

---

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

Primero van los static (sólo una vez)
Después van los de instancia y al final el constructor.

Si se tienen dos instancias, ya no se repiten los static pero sí los de instancia y el 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.

El principio **ACID** en bases de datos relacionales define las propiedades esenciales que debe cumplir una transacción para garantizar la **integridad, confiabilidad y consistencia de los datos**.
A - Atomicidad (**Atomicity**)
C - Consistencia (**Consistency**)
I - Aislamiento (**Isolation**)
D - Durabilidad (**Durability**)

(Atomicity) Una transacción es atómica, lo que significa que se ejecuta totalmente o no se ejecuta en absoluto.

(Consistency) La base de datos debe pasar de un estado válido a otro estado válido, asegurando que las reglas de integridad se mantengan.

(Isolation) Asegura que las transacciones se ejecuten de manera independiente, sin interferencias de otras transacciones concurrentes.

(Durability) Una vez que una transacción se confirma (COMMIT), sus cambios quedan almacenados permanentemente, incluso si ocurre un fallo del sistema.

---

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
//INCORRECTO
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

Esta es una **buena práctica en el desarrollo de software**. Encapsular permite la reutilización, el mantenimiento, seguridad y escalabilidad

---

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

---

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

<build>: Defines build-related configurations for the project.
<plugins>: Contains all the plugins used in the build lifecycle.
<plugin>: Defines an individual plugin.
<groupId>: Identifies the organization that maintains the plugin.
<artifactId>: The specific name of the plugin.
<version>: Specifies the version of the plugin.
<configuration>: Allows customization of the plugin's behavior.

---

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]

Removemos el índice 1 con valor 2.

---

38. Which of the following methods can be used to remove all elements from an ArrayList?
R: clear()

clear() It removes all elements while keeping the list instance.

removeAll() Also removes all elements, but requires passing another collection.

What does NOT work?

list = null; → This removes the reference but doesn't clear the list.
list.forEach(e -> list.remove(e)); → Causes ConcurrentModificationException.

---

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.

---

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.

---

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

---

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

Stages of Agile SDLC

1. Concept / Planning (Product Backlog Creation)
2. Iteration / Sprint Planning
3. Design & Development (Incremental Build)
4. Testing & Quality Assurance (Continuous Testing)
5. Deployment & Release
6. Review & Feedback (Sprint Review & Retrospective)
