Junior-Advanced QA Manual Exercises and Interview Questions

# Topics

Generics

Java API Collections

Composition

Polymorphism

Encapsulation

Override

Overload

JDBC

Joins

Unions

XML

JSON

# Exercises

1. Generics: Crea una clase genérica `Triple` que pueda contener tres elementos de cualquier tipo. Implementa métodos para obtener y establecer los elementos. Además, implementa un método para intercambiar dos elementos dentro del `Triple`.

2. Java API Collections: Utiliza la interfaz `Map` para contar la frecuencia de palabras en un texto, pero esta vez considera también las palabras que empiezan con una mayúscula como diferentes de las que empiezan con minúscula. Implementa una opción para ignorar esta diferencia.

3. Composition: Crea una clase `Address` con atributos como calle, ciudad y código postal. Luego, crea una clase `Company` que contenga un objeto `Address` y agregue atributos adicionales como nombre de la empresa y número de empleados. Implementa métodos para cambiar la dirección de la empresa.

4. Polymorphism: Crea una interfaz `Movable` con métodos para `move` y `stop`. Luego, crea clases `Car` y `Bicycle` que implementen esta interfaz. Implementa una prueba donde creas una lista de objetos `Movable` y llames a los métodos de forma polimórfica.

5. Encapsulation: Refactoriza la clase `Person` para que todos sus atributos sean privados y accedidos únicamente a través de getters y setters. Implementa validaciones en los setters (por ejemplo, la edad no puede ser negativa).

6. Override / Overload: Crea una clase `Calculator` con métodos sobrecargados para `add` que acepten diferentes tipos de parámetros (por ejemplo, enteros, dobles). Luego, crea una clase `ScientificCalculator` que extienda `Calculator` y sobrescriba el método `add` para incluir funcionalidades adicionales.

7. JDBC: Crea una base de datos con tres tablas: `Clientes`, `Pedidos` y `Productos`. Utiliza JDBC para realizar una consulta que muestre los clientes que han realizado pedidos de un producto específico y la cantidad total de ese producto pedido por cada cliente.

8. Joins / Unions: Utiliza JDBC para realizar una consulta que combine datos de las tablas `Clientes` y `Productos` utilizando JOIN y UNION. Por ejemplo, muestra información de clientes que han realizado pedidos y los productos más pedidos en una sola consulta.

9. XML: Lee un archivo XML utilizando la biblioteca `javax.xml.parsers` y transforma los datos en un objeto Java. Luego, realiza operaciones sobre el objeto como agregar nuevos elementos y guardar los cambios en un nuevo archivo XML.

10. JSON: Lee un archivo JSON utilizando la biblioteca `org.json` y transforma los datos en un objeto Java. Implementa métodos para modificar los datos del objeto y guardarlos nuevamente en un archivo JSON.

# Technical Interview Questions

1. Generics: ¿Qué son los genéricos en Java y cuáles son sus beneficios? ¿Cómo se puede restringir un tipo genérico a ciertas clases?

2. Java API Collections: Explica las diferencias entre `ArrayList` y `LinkedList`. ¿Cómo se puede evitar las colisiones en un `HashMap`?

3. Composition: ¿Cuál es la diferencia entre composición y herencia? Da un ejemplo de cuándo usarías la composición en lugar de la herencia.

4. Polymorphism: ¿Qué es el polimorfismo y cómo se implementa en Java? ¿Cuáles son las ventajas del polimorfismo?

5. Encapsulation: ¿Qué es la encapsulación y por qué es importante? ¿Cómo se puede aplicar la encapsulación en una clase Java?

6. Override / Overload: ¿Cuál es la diferencia entre sobrecarga y sobrescritura de métodos? Da un ejemplo de cada uno.

7. JDBC: ¿Qué es JDBC y para qué se utiliza? ¿Cuáles son los pasos principales para conectar una aplicación Java a una base de datos utilizando JDBC?

8. Joins / Unions: Explica las diferencias entre un JOIN y un UNION en SQL. ¿Cuándo usarías un JOIN en lugar de un UNION?

9. XML: ¿Qué es XML y para qué se utiliza? ¿Cuáles son las ventajas de utilizar XML en aplicaciones Java?

10. JSON: ¿Qué es JSON y por qué es tan popular en el intercambio de datos? ¿Cómo se puede manipular JSON en Java?