**1. Principios SOLID:**

* **S**: **Single Responsibility Principle** (Principio de responsabilidad única).
* **O**: **Open/Closed Principle** (Principio de abierto/cerrado).
* **L**: **Liskov Substitution Principle** (Principio de sustitución de Liskov).
* **I**: **Interface Segregation Principle** (Principio de segregación de interfaces).
* **D**: **Dependency Inversion Principle** (Principio de inversión de dependencias).

**2. Herencia y Polimorfismo:**

* **Herencia**:
  + La herencia en Java se realiza mediante extends.
  + **Herencia simple**: Una clase hija hereda de una única clase padre.
  + **Herencia múltiple** en Java: **Interfaces** pueden heredar de otras interfaces.
* **Polimorfismo**:
  + **Sobreescritura de métodos**: Una clase hija puede sobrescribir un método de la clase padre.
  + **Vinculación dinámica**: En tiempo de ejecución, se llama al método más cercano en la jerarquía de clases (usando el tipo real del objeto, no el tipo de referencia).
  + **Sobrecarga de métodos**: Permite tener varios métodos con el mismo nombre, pero con diferentes parámetros (número o tipo).

**3. Interfaces:**

* Todos los **métodos** en una interfaz son **public** y **abstract** por defecto (excepto los métodos default y static).
* Los **atributos** en una interfaz son **public**, **static** y **final** (constantes).
* Una interfaz puede **heredar** de otras interfaces usando extends. **Múltiple herencia** en interfaces.
* **Métodos default** en interfaces (desde Java 8): permiten tener implementación dentro de la interfaz.

**4. Clases Abstractas:**

* No se puede **instanciar** una clase abstracta.
* Se puede tener métodos **abstractos** (sin implementación) y **no abstractos** (con implementación) en la misma clase.
* Las clases hijas **deben implementar** los métodos abstractos a menos que ellas mismas sean abstractas.

**5. ArrayList y HashMap:**

* **ArrayList**: Colección dinámica que permite almacenar elementos de manera ordenada y acceder a ellos por índice.
  + Operaciones: **Añadir**, **modificar**, **eliminar**, **recorrer** (con bucles for o forEach).
* **HashMap**: Estructura de datos que permite almacenar elementos como **pares clave-valor**.
  + Métodos: **put()**, **get()**, **remove()**, **containsKey()**, etc.

**6. Métodos de una clase:**

* **Sobrescritura de métodos**: Una clase hija puede sobrescribir un método de la clase padre para dar una nueva implementación.
* **Sobrecarga de métodos**: Se pueden tener múltiples métodos con el mismo nombre pero con diferentes parámetros.

**7. Composición:**

* **Composición**: Una clase puede tener como **atributo** a otra clase, formando una relación de "tiene un" (es decir, la clase A tiene un objeto de la clase B).
  + **Relación "tiene un"**: Ejemplo de un objeto **Biblioteca** que tiene un objeto **Libro** como atributo.

**8. Constructores:**

* En Java, se pueden tener **n constructores** en una clase (sobrecarga de constructores).
* Un **constructor** puede inicializar los atributos de una clase al momento de crear una instancia de la misma.

**9. Métodos Estáticos:**

* Los **métodos estáticos** se pueden llamar sin necesidad de crear una instancia de la clase.
* **Métodos estáticos en interfaces**: Desde Java 8, las interfaces pueden tener métodos estáticos.

**10. Singleton Pattern:**

* El patrón **Singleton** asegura que una clase tenga **una única instancia** y proporciona un punto de acceso global a esa instancia.
* Se implementa usando un **constructor privado** y un **método estático** para obtener la instancia.

**11. Array Estático vs Dinámico:**

* **Arrays estáticos**: Tienen un tamaño fijo determinado en el momento de la declaración.
* **Arrays dinámicos (como ArrayList)**: Pueden crecer y decrecer dinámicamente, lo que les permite ser más flexibles.

**12. Herencia Múltiple de Interfaces:**

* **Una interfaz puede extender** de varias interfaces.
* Java permite la **herencia múltiple solo en interfaces**, no en clases.

**13. Método default en Interfaces:**

* A partir de Java 8, las interfaces pueden tener **métodos con implementación** utilizando la palabra clave default.

**14. Ejercicio de Biblioteca:**

* Creación de una clase **Biblioteca** que utilice un **ArrayList** para manejar **libros** (añadir, modificar, eliminar, imprimir).
* Los métodos de **Biblioteca** pueden implementar interfaces para garantizar una **implementación común**.