

Semana: 14

**GRUPO 5**

**Integrantes:**

Roberto Agustín Mejía Collazos

Miguel Ángel Velásquez Ysuiza

Manuel Ángel Pecho Santos

Daniel Wilfredo Sotomayor Beteta

**Tarea 14: PATRONES COMPORTAMIENTO: PATRÓN COMNAND Y PATRÓN MEMENTO.**

**Dilian Anabel HURTADO PONCE**

07/11/24

**Docente:**

***Texto

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamente Cuenta.java IOperacion.java***

***Texto

Descripción generada automáticamente Depositarlmpl.java Retirarlmpl.java***

***Texto

Descripción generada automáticamente***

***Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente Invoker.java App.java***

### 

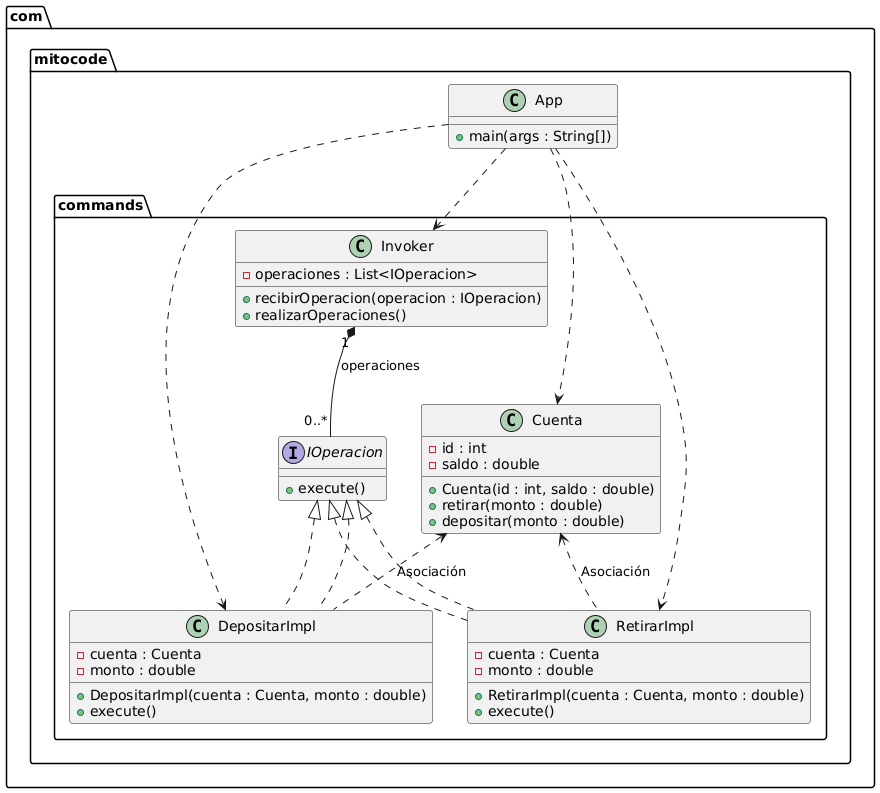
### Análisis del código y Uso del patrón Command

Este código implementa el **Patrón Command**, el cual permite encapsular operaciones como objetos para ejecutarlas, encolarlas y extenderlas de forma independiente del receptor.

1. **Cuenta**: Es el **Receiver** (receptor) que realiza las acciones (retirar y depositar) sobre su saldo.
2. **IOperacion**: Define la interfaz Command con el método execute(), usado para ejecutar cualquier comando de forma uniforme.
3. **DepositarImpl y RetirarImpl**: Son los **comandos concretos** que implementan IOperacion y encapsulan las operaciones de depósito y retiro, llamando a los métodos correspondientes en Cuenta.
4. **Invoker**: Es el encargado de ejecutar los comandos. Almacena una lista de operaciones y las ejecuta en secuencia mediante el método realizarOperaciones().
5. **App**: Actúa como **Cliente**. Esta clase inicializa la Cuenta, los comandos DepositarImpl y RetirarImpl, y los pasa al Invoker. App organiza el flujo de las operaciones sin interactuar directamente con Cuenta, facilitando la extensión de comandos sin modificar el cliente.

**Diferencia con un Enfoque Tradicional de POO**

En un enfoque tradicional, las operaciones de Cuenta (depositar y retirar) se llamarían directamente desde el cliente. Sin embargo, con el patrón Command, estas operaciones se encapsulan, permitiendo la flexibilidad de organizar, encolar o agregar nuevas operaciones sin modificar la estructura del cliente ni el receptor.



**Explicación del Diagrama UML**

* **Cuenta: Es el receptor que realiza las operaciones sobre el saldo.**
* **IOperacion: Define el método execute común para todos los comandos.**
* **DepositarImpl y RetirarImpl: Son las implementaciones concretas de IOperacion que encapsulan el comportamiento de las operaciones de Cuenta.**
* **Invoker: Almacena y ejecuta los comandos, permitiendo la ejecución en secuencia y el manejo flexible de múltiples operaciones.**
* **App: El cliente que configura y coordina la ejecución de comandos, creando instancias de Cuenta, DepositarImpl, RetirarImpl y Invoker para ejecutar las operaciones en Cuenta sin interactuar directamente con ella.**