Universidad Tecnológica del Perú

Docente:

Dilian Anabel HURTADO PONCE

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

Semana: 14

GRUPO 5

Integrantes:

Roberto Agustín Mejía Collazos

Miguel Ángel Velásquez Ysuiza

Manuel Ángel Pecho Santos

Daniel Wilfredo Sotomayor Beteta

07/11/24

Código GitHub

Cuenta.java

Bienvenido | Cuenta.java | App.java | DepositarImpl.java | Invoker.java | IOperacion.java | RetirarImpl.java

Command > src > main > java > com > mitocode > commands > Cuenta.java > ...

```java
mitocode, hace 6 años | 1 author (mitocode)
package com.mitocode.commands;

mitocode, hace 6 años | 1 author (mitocode)
//Receiver/Request
public class Cuenta {

    private int id;
    private double saldo;

    public Cuenta(int id, double saldo) {
        this.id = id;
        this.saldo = saldo;
    }

    public void retirar(double monto) {
        this.saldo = this.saldo - monto;
        System.out.println("[COMANDO RETIRAR] Cuenta: " + id + " Saldo: " + this.saldo);
    }

    public void depositar(double monto) {
        this.saldo = this.saldo + monto;
        System.out.println("[COMANDO DEPOSITAR] Cuenta: " + id + " Saldo: " + this.saldo);
    }

}

```

IOperacion.java

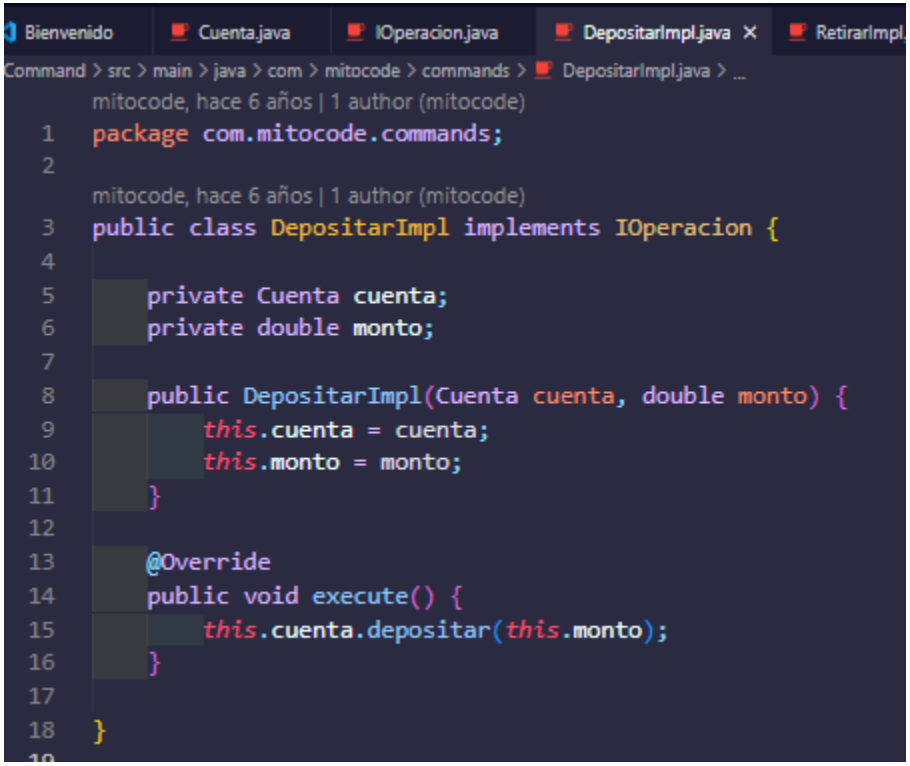
Bienvenido | Cuenta.java | IOperacion.java

Command > src > main > java > com > mitocode > commands > IOpe

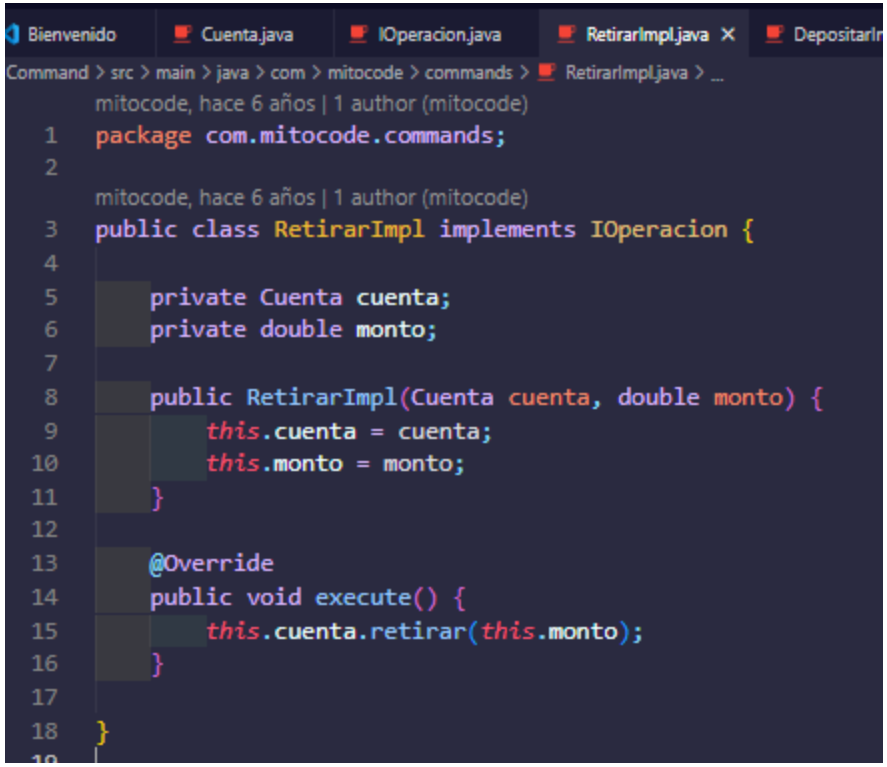
```java
mitocode, hace 6 años | 1 author (mitocode)
package com.mitocode.commands;

mitocode, hace 6 años | 1 author (mitocode)
//Command
@FunctionalInterface
public interface IOperacion {

    void execute();
}

```

Código GitHub

DepositarImpl.java

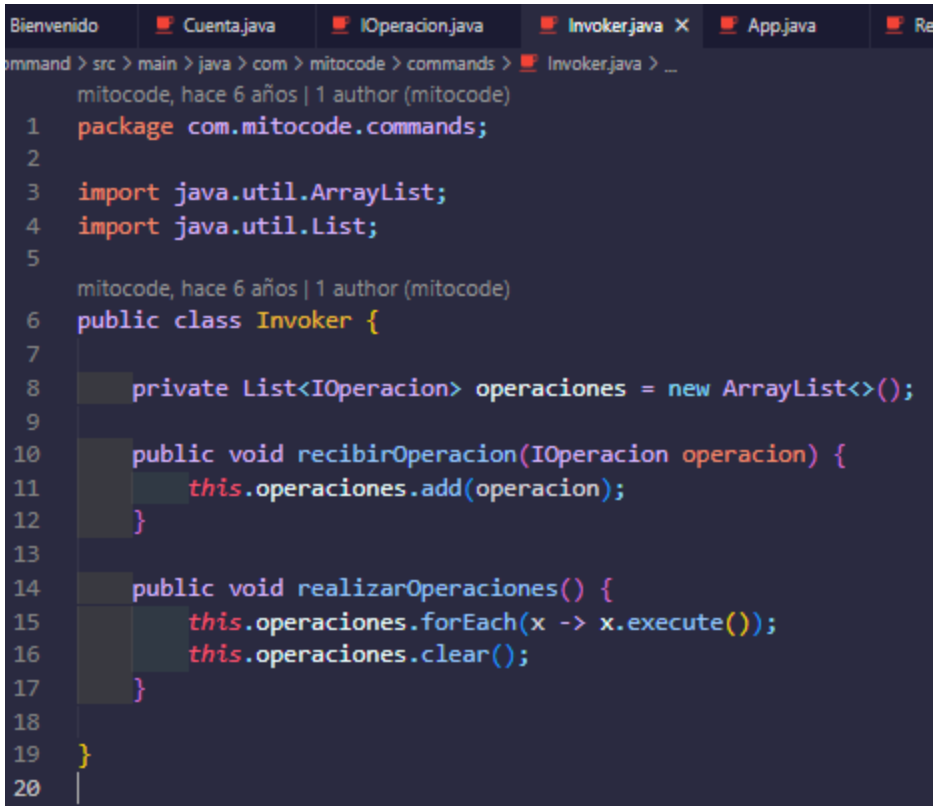


RetirarImpl.java



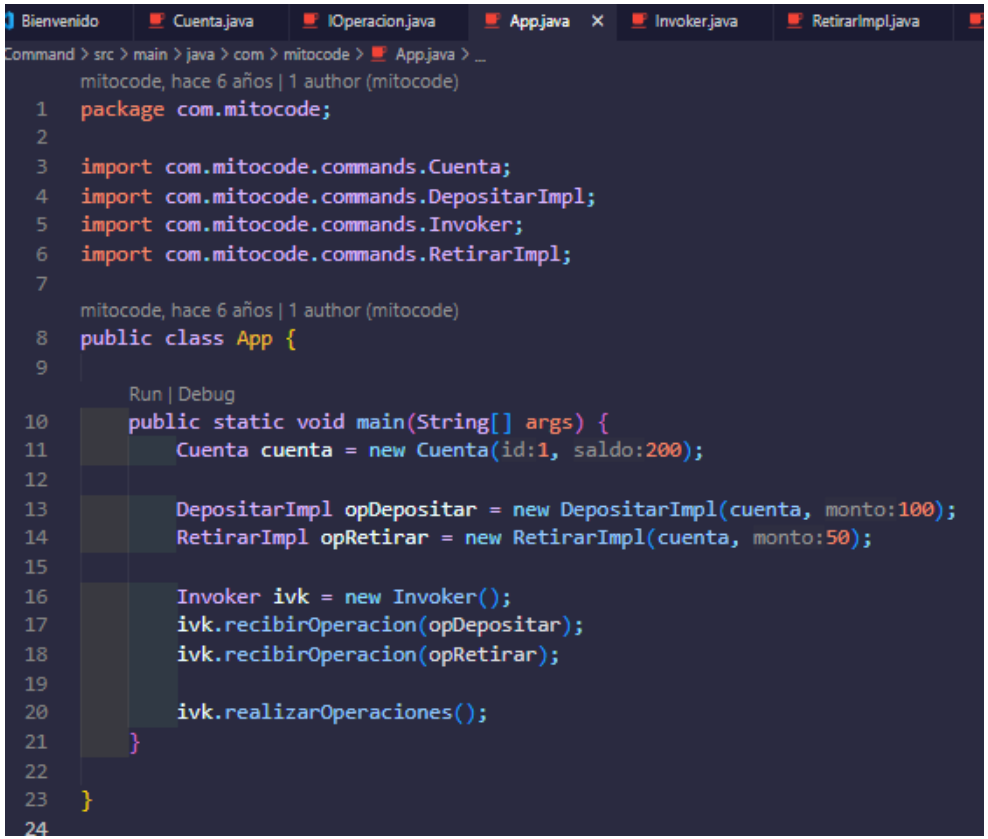
Código GitHub

Invoker.java



```
Bienvenido | Cuenta.java | IOperacion.java | Invoker.java | App.java | Re
ommand > src > main > java > com > mitocode > commands > Invoker.java > ...
    mitocode, hace 6 años | 1 author (mitocode)
1   package com.mitocode.commands;
2
3   import java.util.ArrayList;
4   import java.util.List;
5
    mitocode, hace 6 años | 1 author (mitocode)
6   public class Invoker {
7
8       private List<IOperacion> operaciones = new ArrayList<>();
9
10      public void recibirOperacion(IOperacion operacion) {
11          this.operaciones.add(operacion);
12      }
13
14      public void realizarOperaciones() {
15          this.operaciones.forEach(x -> x.execute());
16          this.operaciones.clear();
17      }
18
19  }
20
```

App.java



```
Bienvenido | Cuenta.java | IOperacion.java | App.java | Invoker.java | RetirarImpl.java
Command > src > main > java > com > mitocode > App.java > ...
    mitocode, hace 6 años | 1 author (mitocode)
1   package com.mitocode;
2
3   import com.mitocode.commands.Cuenta;
4   import com.mitocode.commands.DepositarImpl;
5   import com.mitocode.commands.Invoker;
6   import com.mitocode.commands.RetirarImpl;
7
    mitocode, hace 6 años | 1 author (mitocode)
8   public class App {
9
        Run | Debug
10      public static void main(String[] args) {
11          Cuenta cuenta = new Cuenta(id:1, saldo:200);
12
13          DepositarImpl opDepositar = new DepositarImpl(cuenta, monto:100);
14          RetirarImpl opRetirar = new RetirarImpl(cuenta, monto:50);
15
16          Invoker ivk = new Invoker();
17          ivk.recibirOperacion(opDepositar);
18          ivk.recibirOperacion(opRetirar);
19
20          ivk.realizarOperaciones();
21      }
22
23  }
24
```

Análisis del código

Análisis del código y Uso del patrón Command

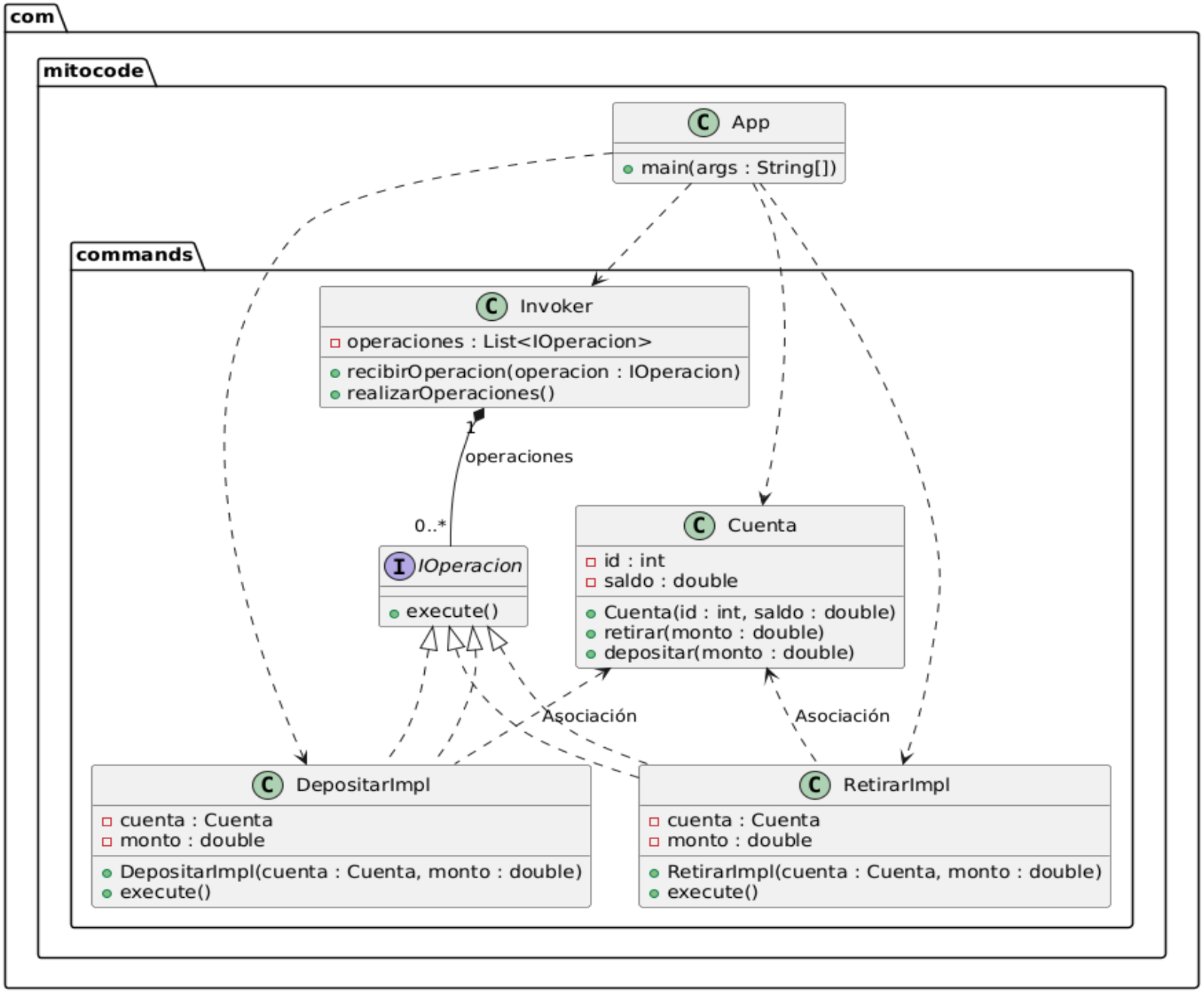
Este código implementa el **Patrón Command**, el cual permite encapsular operaciones como objetos para ejecutarlas, encolarlas y extenderlas de forma independiente del receptor.

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.

DIAGRAMA UML



Explicación UML

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.**