Ejemplo de patrón cadena de responsabilidad

DigitalHouse >
Coding School

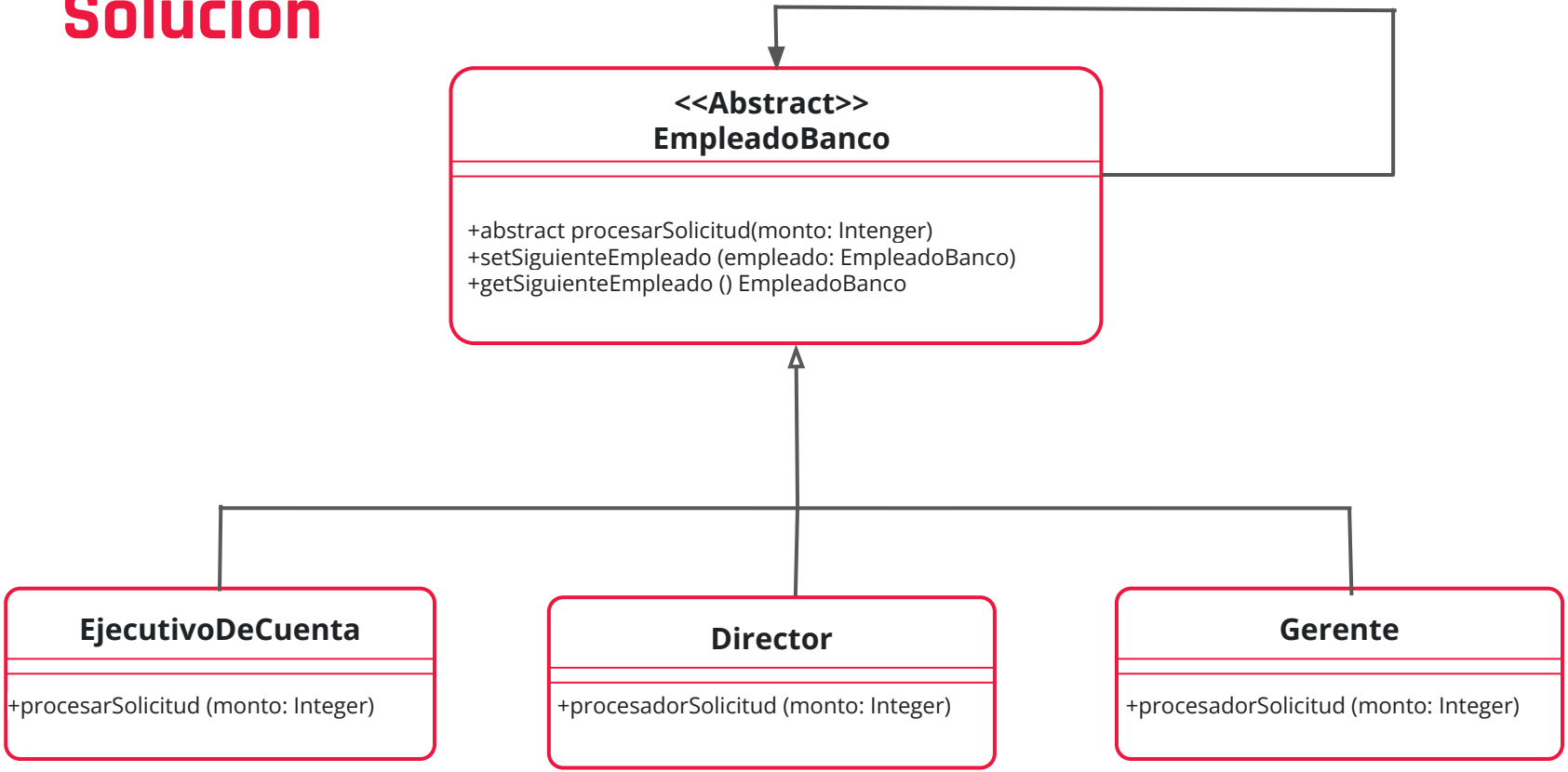
Certified Tech Developer
The Ultimate Degree

Ejemplo de patrón cadena de responsabilidad

Imaginemos que estamos desarrollando un sistema para el área de créditos en un banco y queremos que al momento de que un cliente solicite un crédito se envíe un pedido a los diferentes encargados de autorizarlo. El banco nos indicó que:

- Si el monto no supera los 60.000, entonces, el ejecutivo de cuenta puede aprobarlo.
- Si el monto está entre los 60.000 y 200.000, entonces, el gerente es quien lo aprueba.
- Si el monto se encuentra por encima de 200.000 lo aprueba el director.

Solución


<<Abstract>>
EmpleadoBanco
+abstract procesarSolicitud(monto: Intenger)
+setSiguienteEmpleado (empleado: EmpleadoBanco)
+getSiguienteEmpleado () EmpleadoBanco
EjecutivoDeCuenta
+procesarSolicitud (monto: Integer)
Director
+procesadorSolicitud (monto: Integer)
Gerente
+procesadorSolicitud (monto: Integer)

Implementación de las clases del diagrama UML

```
public abstract class EmpleadoBanco {

    private EmpleadoBanco sigEmpleadoBanco;
    public abstract void procesarSolicitud(Integer monto);

    public void setSigEmpleadoBanco(EmpleadoBanc emp) {
        sigEmpleadoBanco = emp;
    }
    public EmpleadoBanco getSigEmpleadoBanco() {
        return sigEmpleadoBanco;
    }

}
```

La **clase manejadora** es la responsable de comenzar la cadena con ella, en este **caso**, **EmpleadoBanco**.

```java
public class Director extends EmpleadoBanco {
    @Override
    public void procesarSolicitud(Integer monto) {
        if (monto > 200000)
            System.out.println("Yo me encargo de gestionarlo.
                                Director");
        else if (getSigEmpleadoBanco() != null)
            getSigEmpleadoBanco().procesarSolicitud(monto);
    }
}
```

Subclases en las que definiremos bajo qué criterio y cómo procesarán la solicitud.

```java
public class EjecutivoCuenta extends EmpleadoBanco {
    @Override
  public void procesarSolicitud(Integer monto) {
      if (monto < 60000)
          System.out.println("Yo me encargo de gestionarlo.
                              Ejecutivo de cuenta");
      else if (getSigEmpleadoBanco()  != null)
          getSigEmpleadoBanco().procesarSolicitud(monto);

  }
}
```

Subclases en las que definiremos bajo qué criterio y cómo procesarán la solicitud.

```
public class Gerente extends EmpleadoBanco {

    @Override
    public void procesarSolicitud(Integer monto) {
        if (monto >= 60000 && monto <= 200000)
            System.out.println("Yo me encargo de gestionarlo.
                                Gerente");
        else if (getSigEmpleadoBanco() != null)
            getSigEmpleadoBanco().procesarSolicitud(monto);


    }
}
```

Subclases en las que definiremos bajo qué criterio y cómo procesarán la solicitud.

```
public static void main(String[] args) {

      EmpleadoBanco empleado1 = new EjecutivoCuenta();
      EmpleadoBanco empleado2 = new Gerente();
      EmpleadoBanco empleado3 = new Director();

      empleado2.setSigEmpleadoBanco(empleado3);
      empleado1.setSigEmpleadoBanco(empleado2);

      empleado1.procesarSolicitud(78000);

}
```

Test


DigitalHouse>
Coding School