

Tecnología de Programación

Martín L. Larrea

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Patrones de diseño

Los patrones de diseño **nombran, explican y evalúan** un **diseño importante y recurrente** en los sistemas orientados a objetos.

Gang Of Four



Erich Gamma



Ralph Johnson



John Vlissides



Richard Helm

Patrones GoF

Los siguientes son los patrones de diseño conocidos como **GoF**

		PROPÓSITO		
		CREACIONAL	ESTRUCTURAL	COMPORTAMIENTO
SCOPE	CLASE	Factory Method	Adapter	Interpreter Template Method
	OBJETO	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Flyweight Observer State Strategy Visitor

Patrones de Comportamiento

Behavioral Patterns

Patrones de comportamiento

Los **patrones de comportamiento** se centran en los **algoritmos** y la asignación de **responsabilidades** entre los objetos.

Son patrones tanto de clases y objetos (similares a los anteriores)

*como de comunicación entre ellos.
Caracterizan flujo de control complejo.*

Los **patrones de comportamiento de clases** (*behavioral class patterns*)

utilizan herencia para distribuir el comportamiento entre las clases.

Los **patrones de comportamiento de objetos** (*behavioral object patterns*)

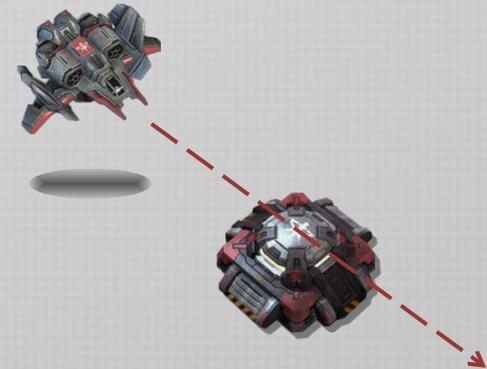
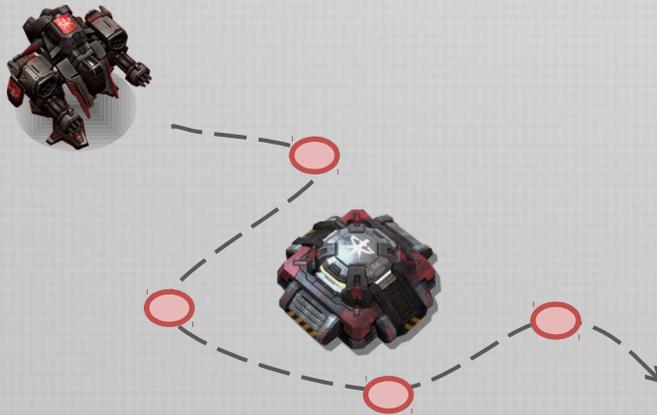
utilizan composición de objetos en lugar de herencia.

Algunos describen cómo los objetos cooperan entre sí para

Patrón *Strategy*



Viking
Starcraft 2



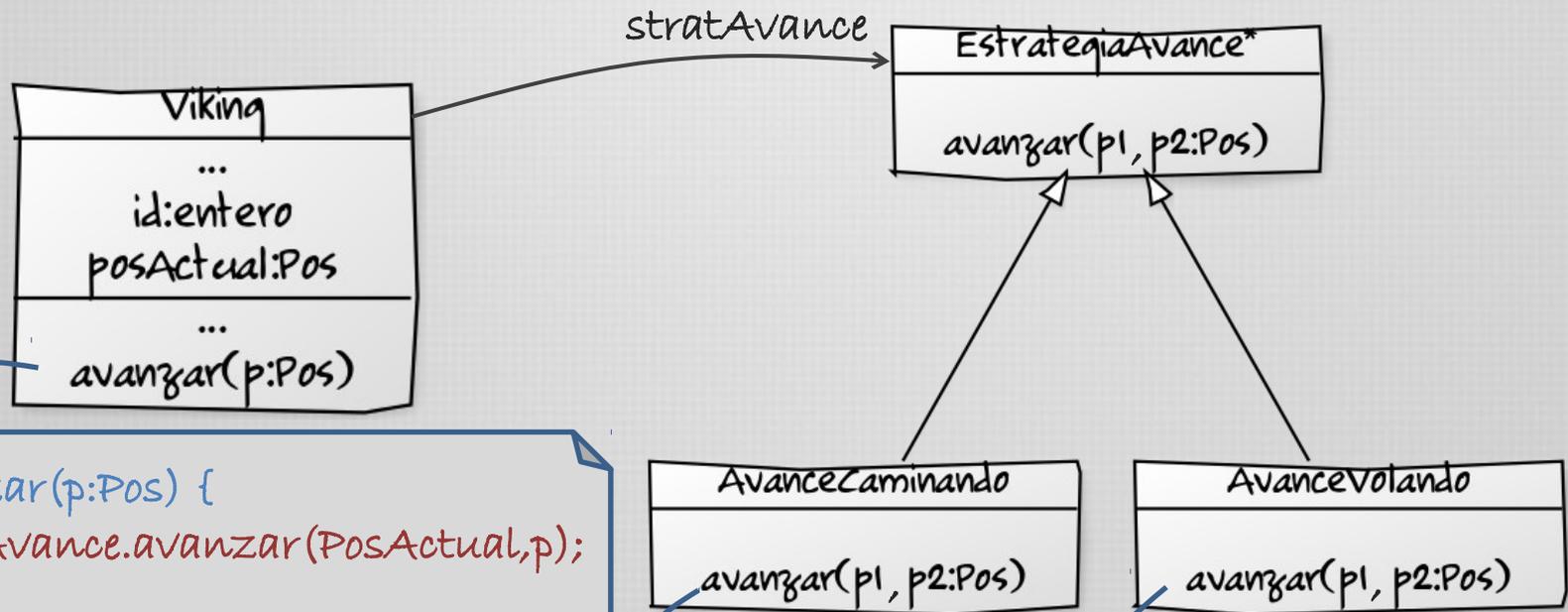
Patrón Strategy



```
avanzarVolando(p:Pos) {
    activarAnimacion("volando");
    moverA(p, velocidadVuelo);
}
```

```
avanzarCaminando(p:Pos) {
    Lista[Pos] ruta;
    activarAnimacion("caminando");
    ruta = buscarCamino(posActual, p);
    foreach (paso in ruta) {
        moverA(paso, velocidadPiso);
    }
}
```

Patrón Strategy

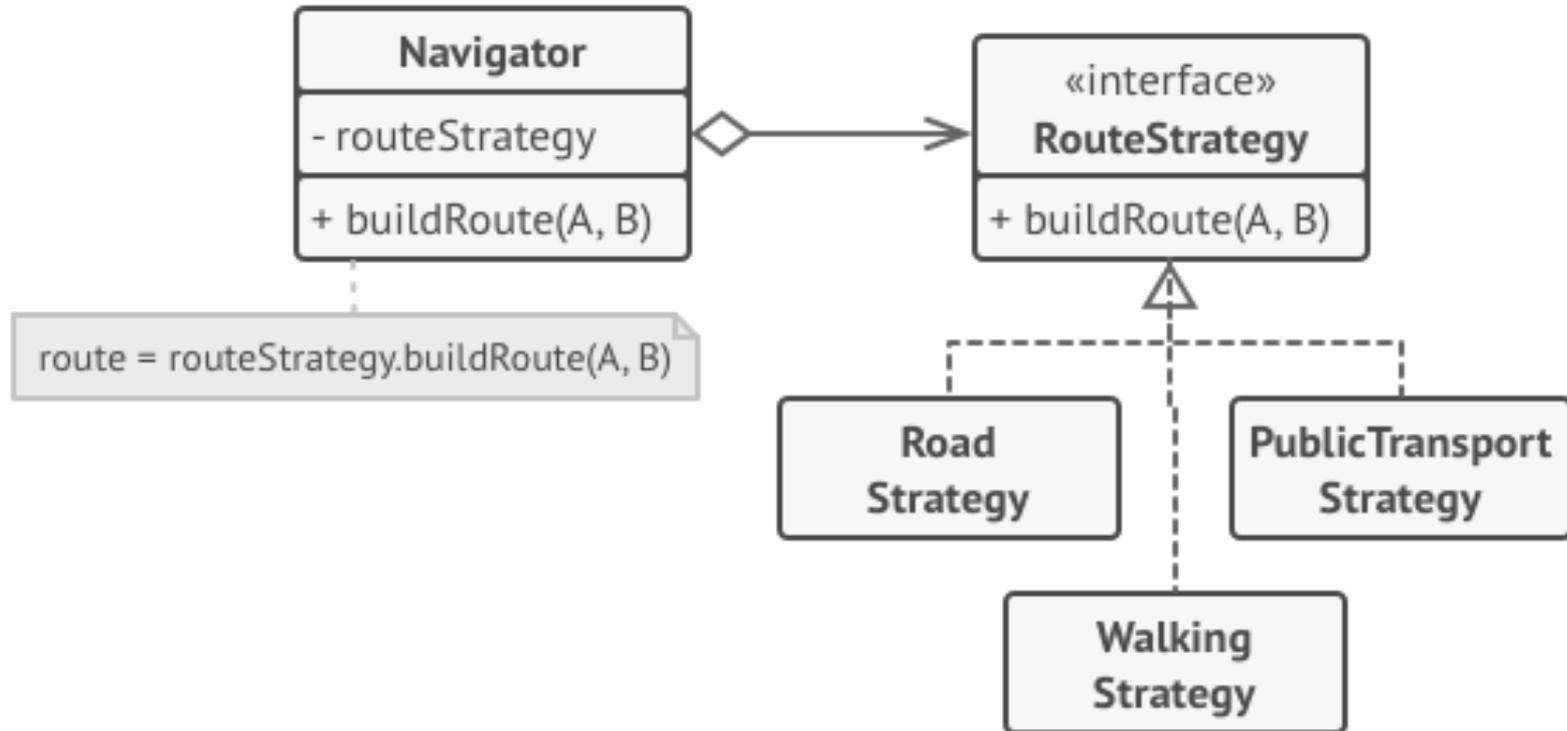


```
avanzar(p:Pos) {
    stratAvance.avanzar(posActual,p);
}
```

```
avanzar(p1,p2:Pos) {
    Lista[Pos] ruta;
    activarAnimacion("caminando");
    ruta = buscarCamino(posActual,p);
    foreach(paso in ruta) {
        moverA(paso,velocidadPiso);
    }
}
```

```
avanzar(p1,p2:Pos) {
    activarAnimacion("volando");
    moverA(p,velocidadVuelo);
}
```

Patrón *Strategy*



Route planning strategies.

Patrón State



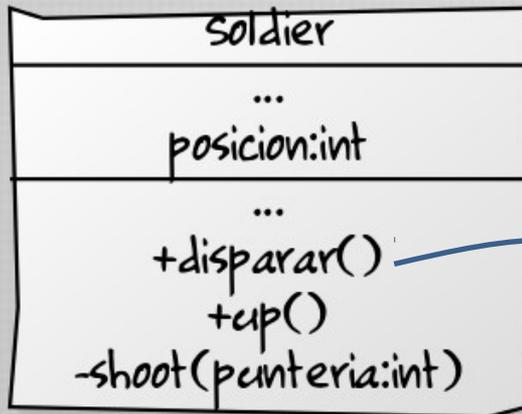
standing



kneeling



running



```
if (posición=1)//standing
then
  shoot(7);
else if (posición=2) //kneeling
then
  shoot(8)
else if (posición=3) //running
then
  shoot(2)
```

Patrón State



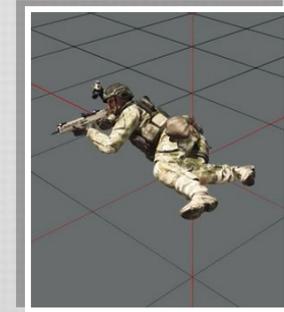
standing



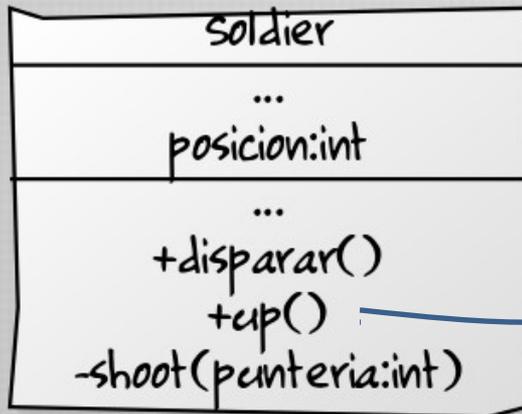
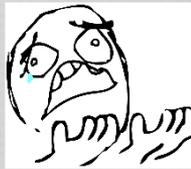
kneeling



running

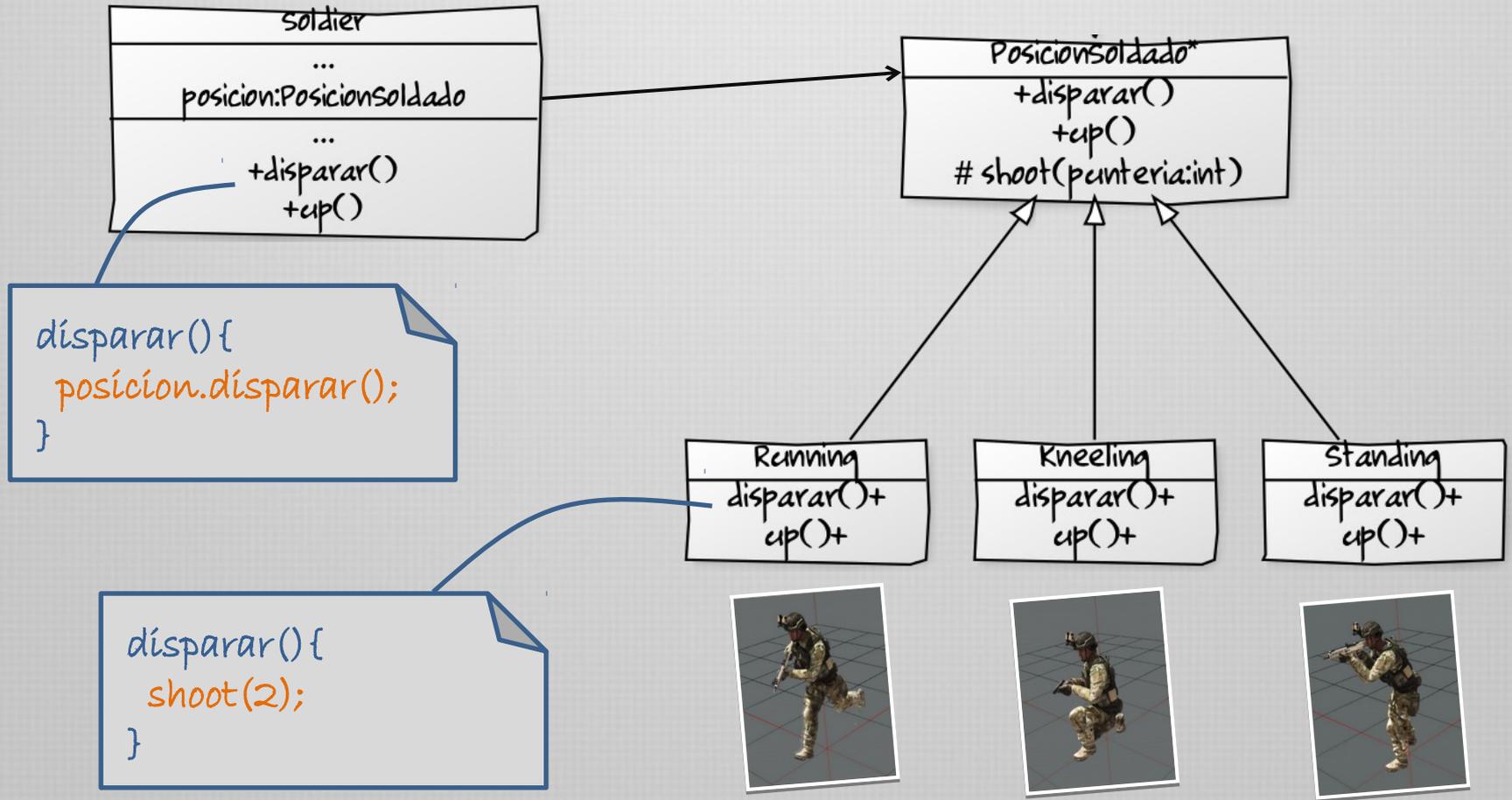


down

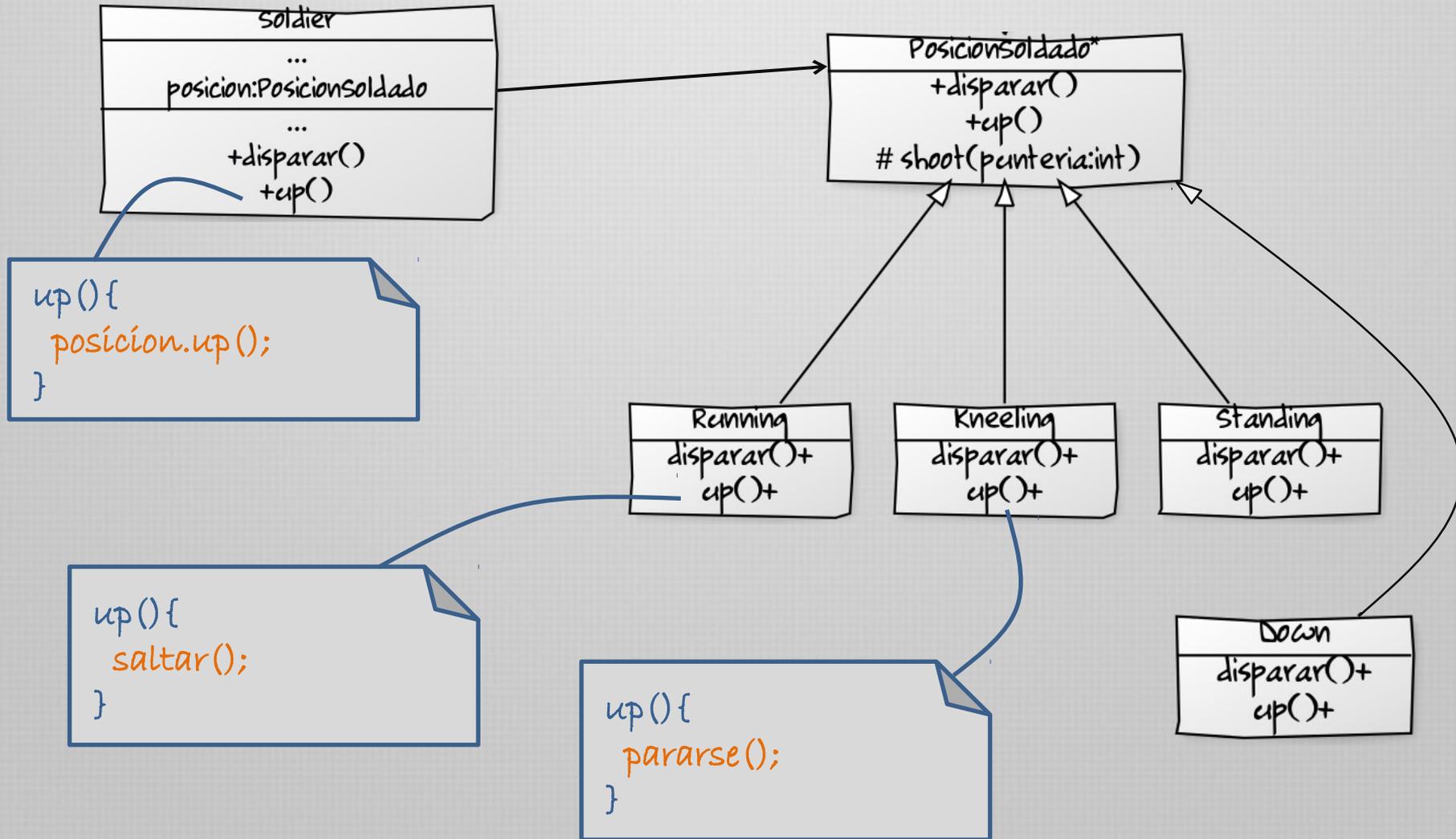


```
if (posición=1)//standing
then
  saltar()
else if (posición=2) //kneeling
then
  pararse()
else if (posición=3) //running
then
  saltar()
```

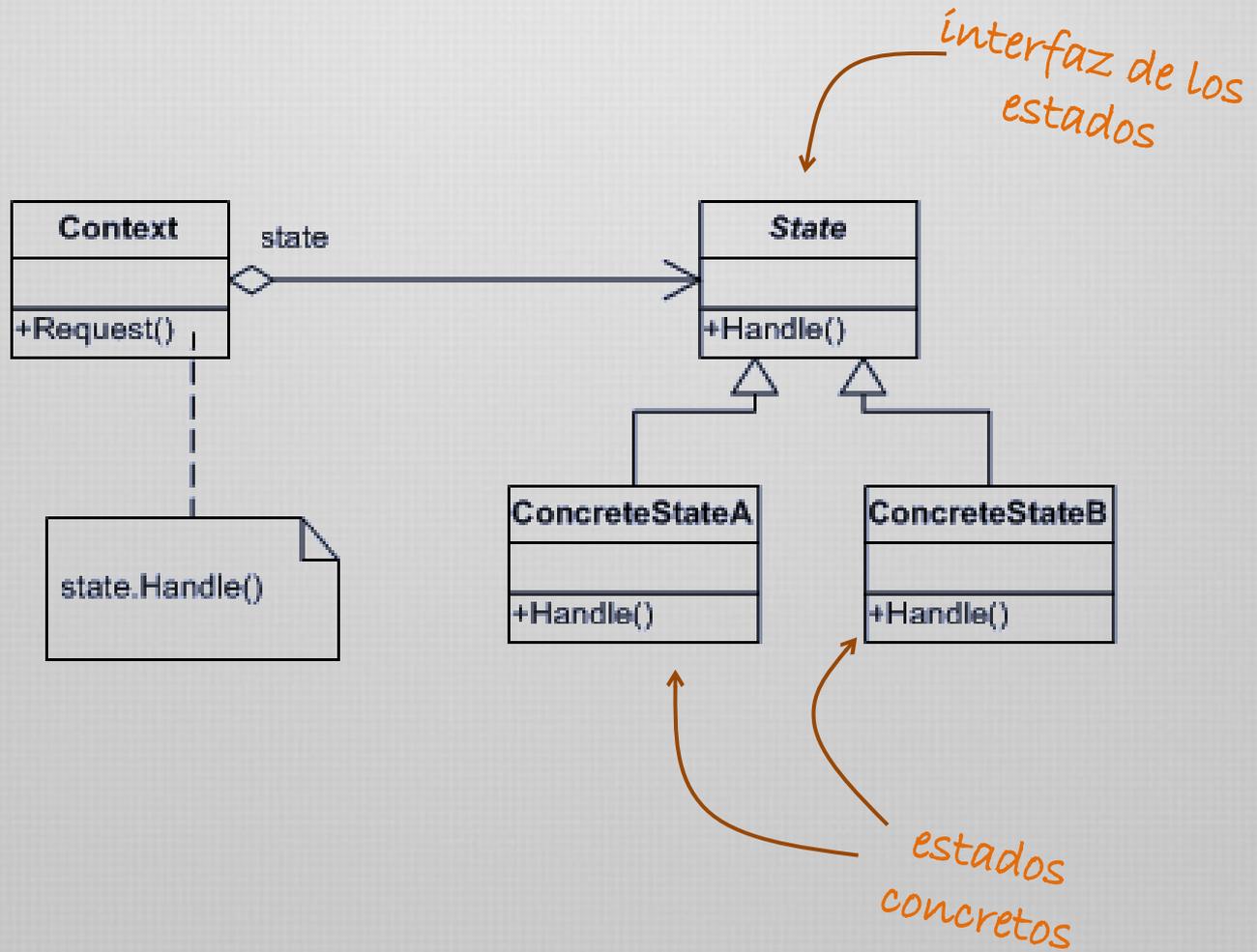
Patrón State



Patrón State



Patrón State - UML



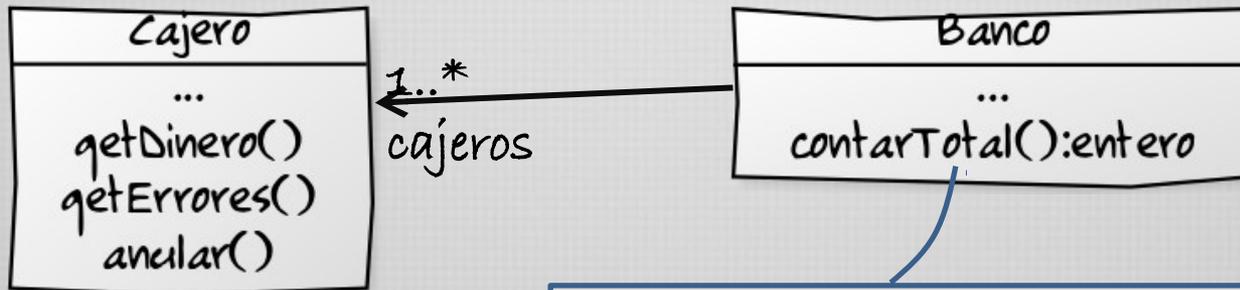
Patrón Visitor



Necesito calcular
cuánto dinero
hay en total en
todos nuestros
cajeros

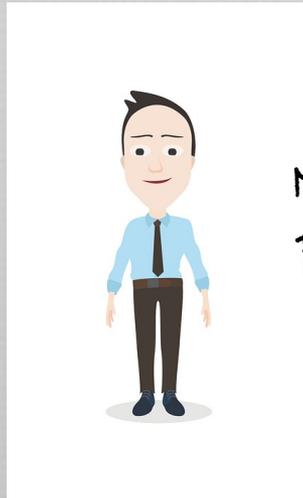
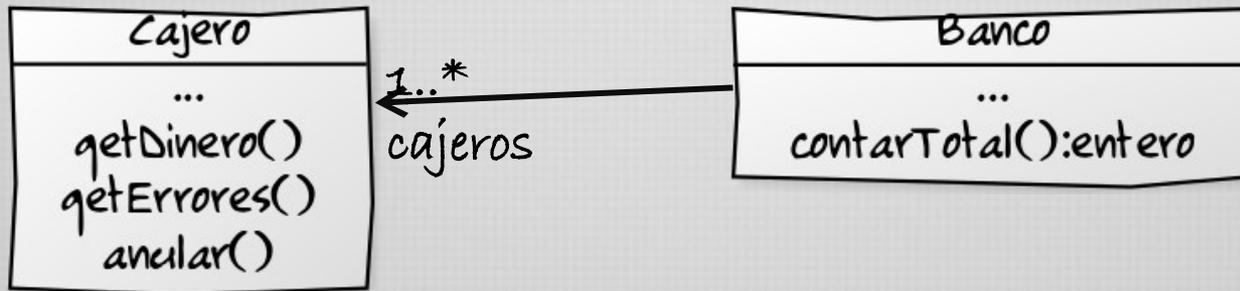


Patrón Visitor



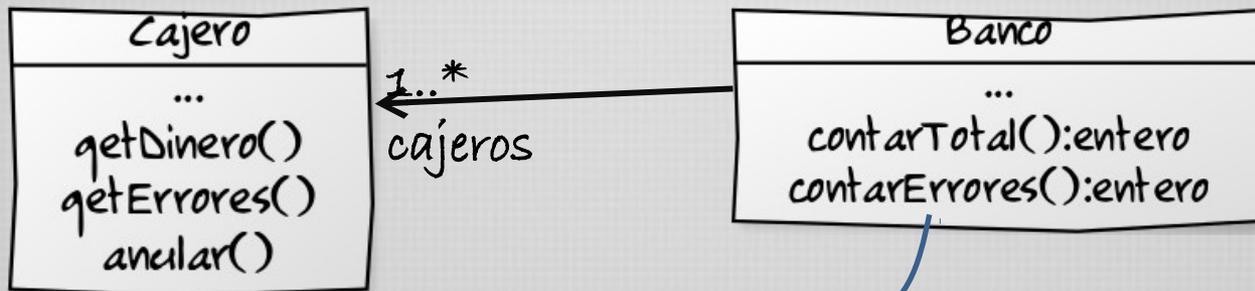
```
contarTotal(): entero {
    total = 0;
    foreach (c in cajeros) {
        total = total + c.getDinero()
    }
    return total;
}
```

Patrón Visitor



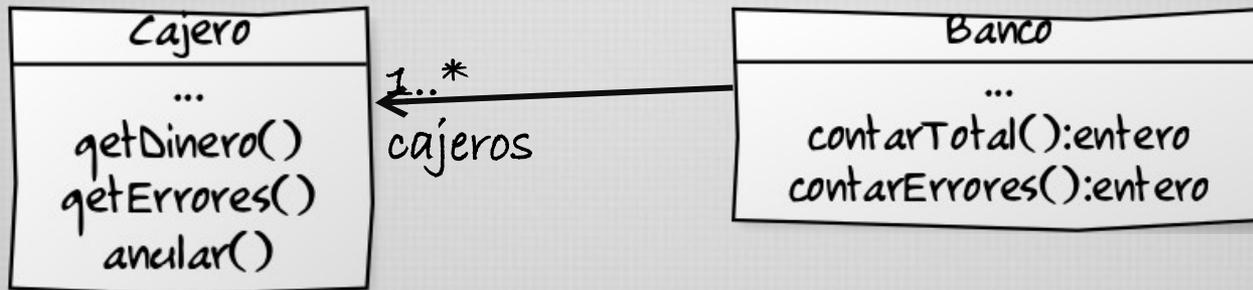
Necesito calcular el promedio de errores de todo nuestro sistema de cajeros

Patrón Visitor



```
contarErrores(): entero {
    total = 0;
    cant = 1;
    foreach (c in cajeros) {
        total = total + c.getErrores();
        cant++;
    }
    return (total div cant);
}
```

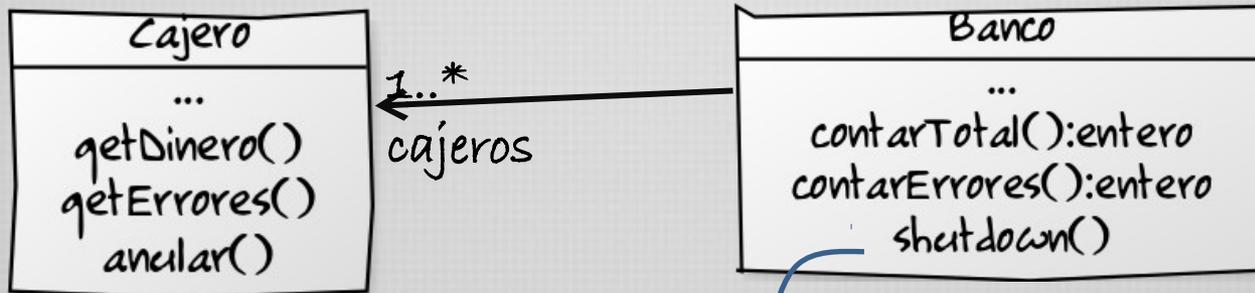
Patrón Visitor



Necesito apagar todos los cajeros del sistema

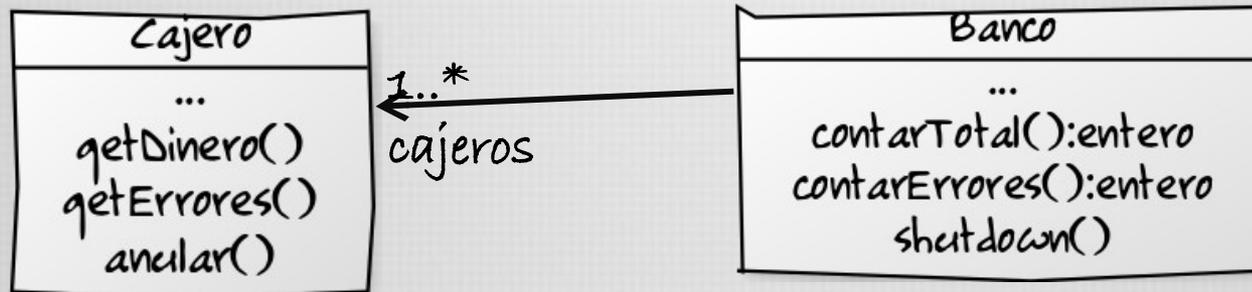


Patrón Visitor



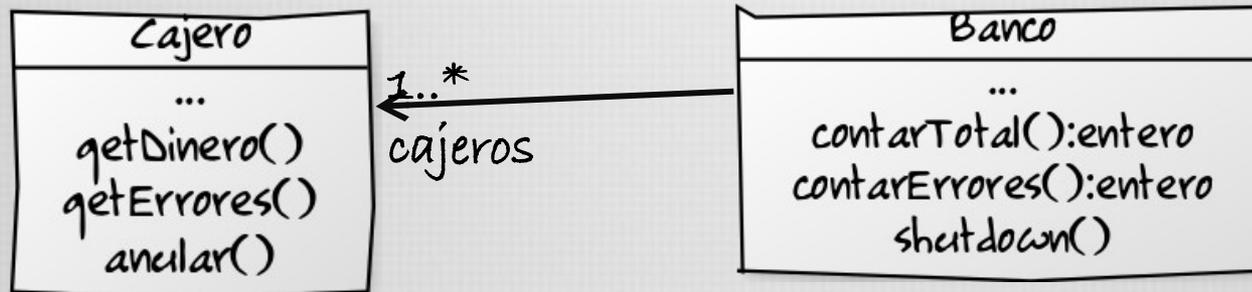
```
shutdown() {
    foreach(c in cajeros){
        c.anular();
    }
}
```

Patrón Visitor



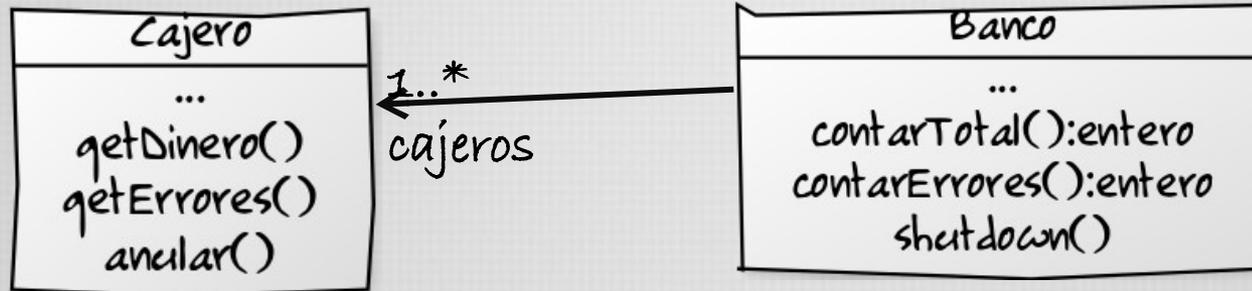
```
operacionX() {
    foreach(c in cajeros) {
        //hacer algo con cada cajero
    }
}
```

Patrón Visitor



```
operacionX(Algo a) {
    foreach(c in cajeros) {
        //hacer algo con cada cajero
    }
}
```

Patrón Visitor

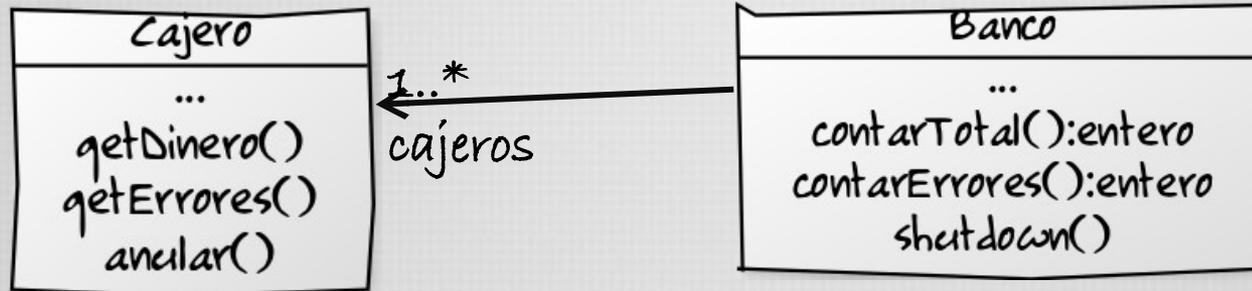


`aceptar(Algo a)`

```
operacionX(Algo a) {
    foreach(c in Cajeros) {
        //hacer algo con cada cajero
        c.aceptar(a)
    }
}
```

```
aceptar(Algo a) {
    //permitir que a realice tareas sobre mí
}
```

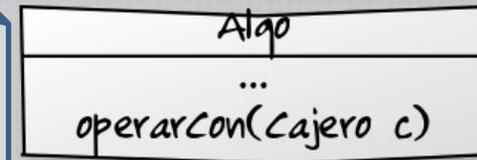
Patrón Visitor



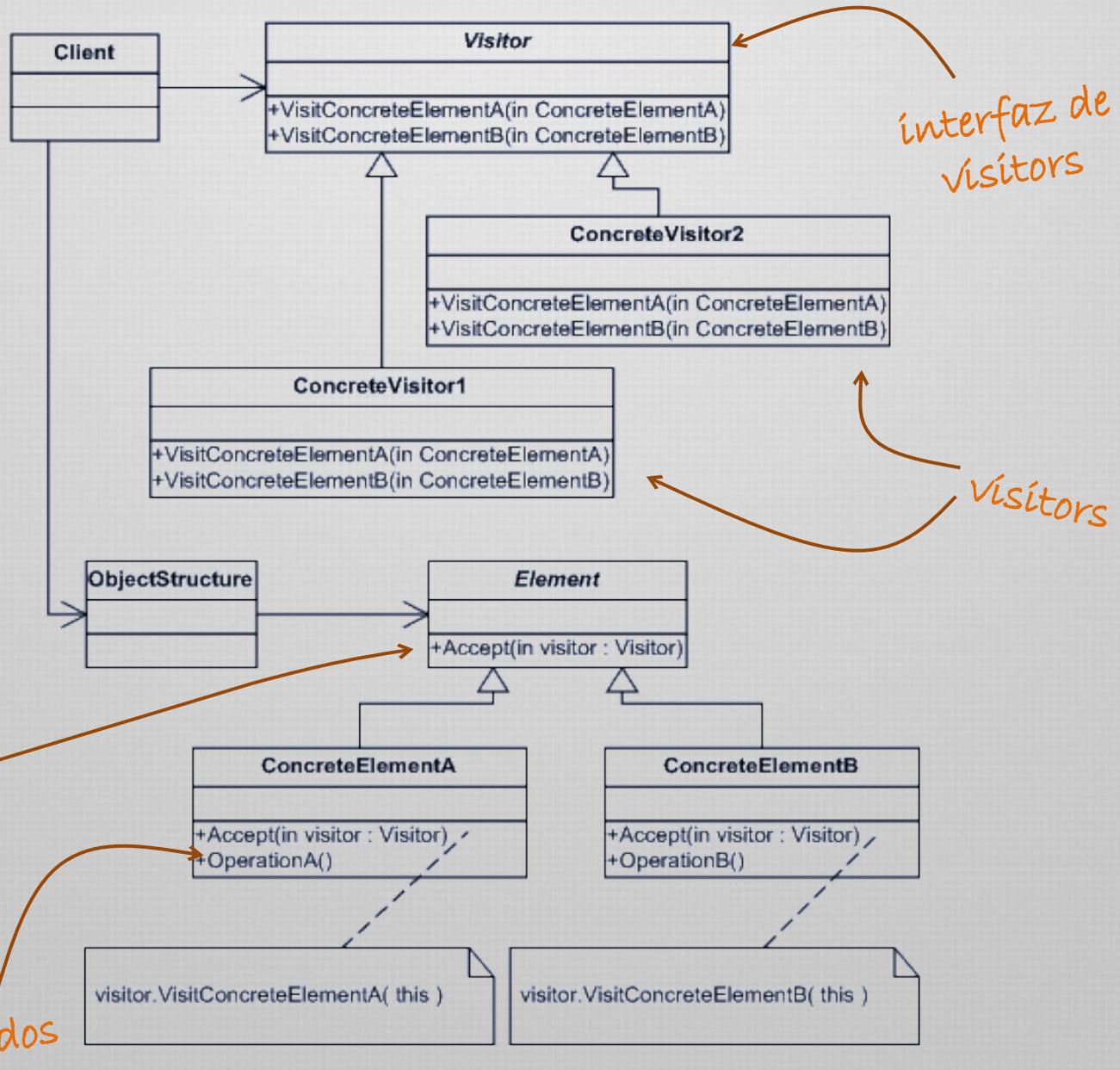
aceptar(Algo a)

```
operacionX(Algo a) {
    foreach(c in Cajeros) {
        //hacer algo con cada cajero
        c.aceptar(a)
    }
}
```

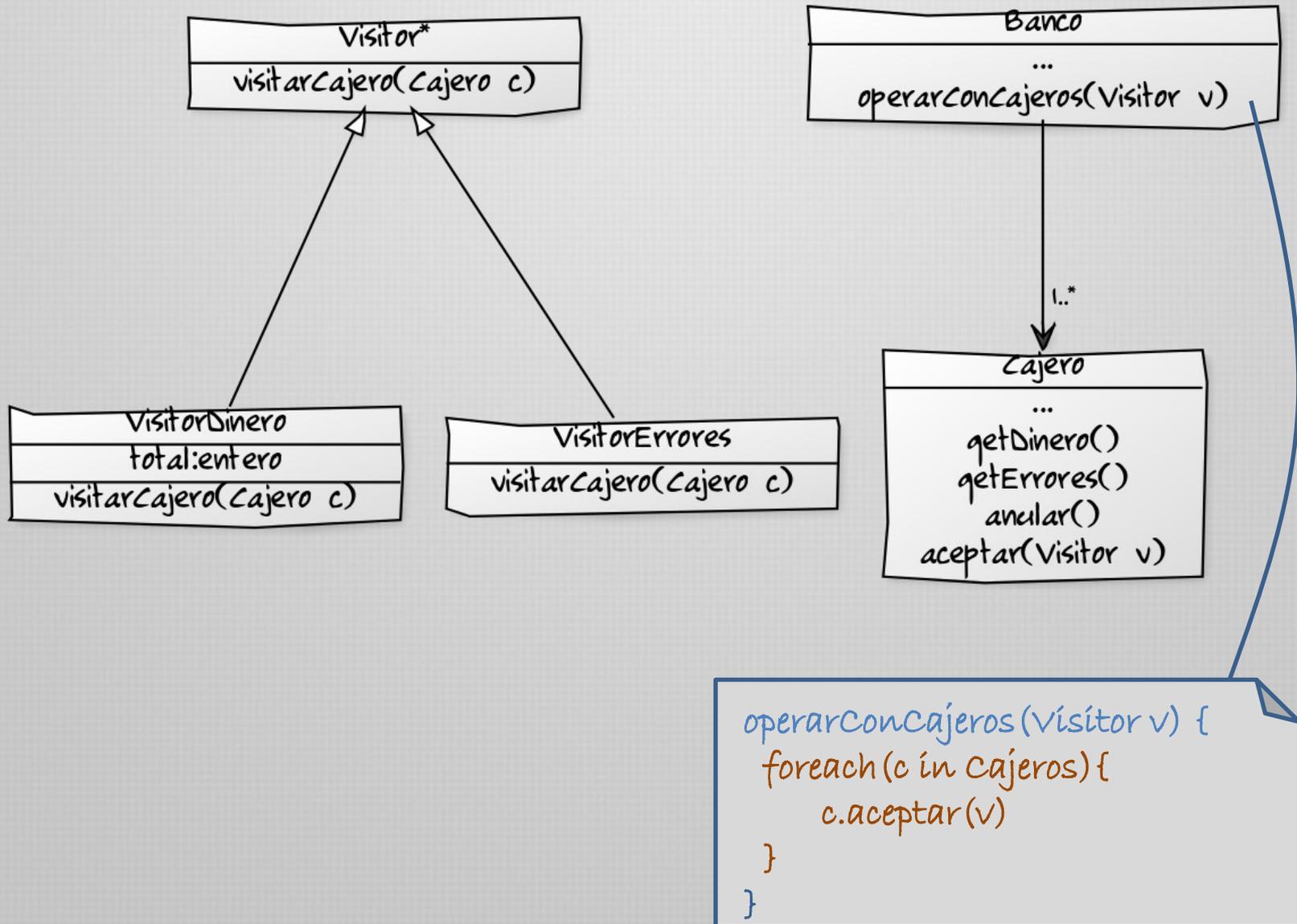
```
aceptar(Algo a) {
    //permitir que a realice tareas sobre mí
    a.operarCon(this)
}
```



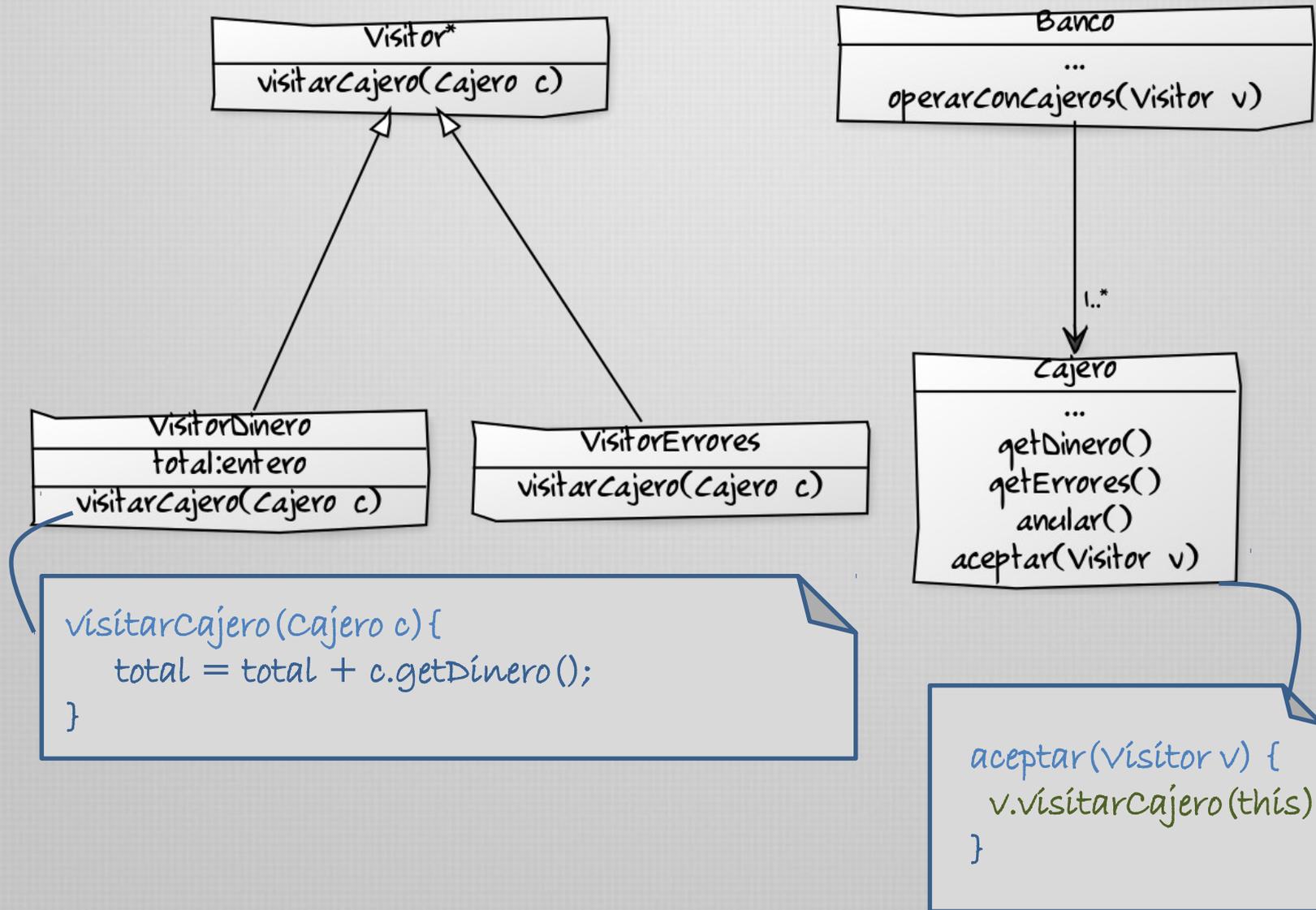
Patrón Visitor



Patrón Visitor



Patrón Visitor



Patrón *Visitor*

PatternCraft
Visitor Pattern



[https://www.youtube.com/watch?
v=KSEyIXnknoY](https://www.youtube.com/watch?v=KSEyIXnknoY)