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 **O**f **F**our


Erich Gamma


Ralph Johnson


John Vlíssides


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)

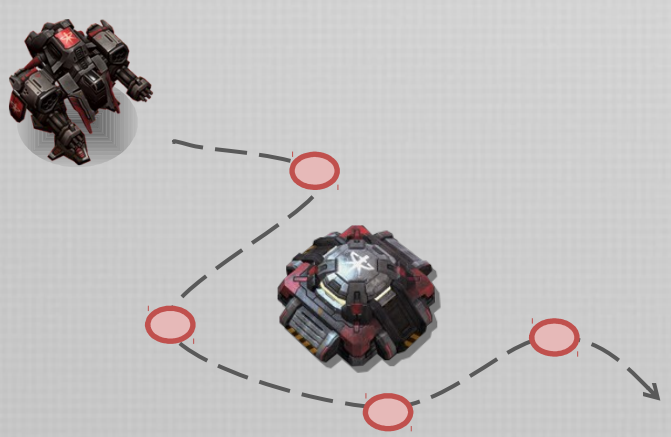
como de comunicación entre ellos.
Caracterizan flujo de control complejo.

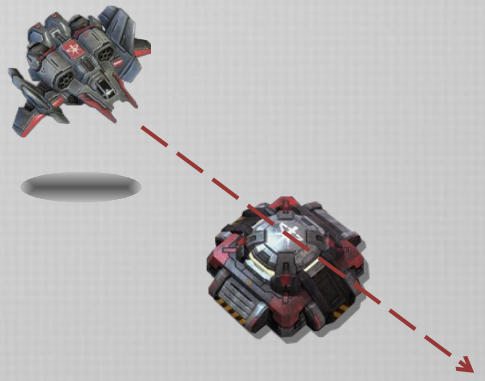
Los patrones de comportamiento de clases *(behavioral class patterns)*
utilizan herencia para distribuir el comportamiento entre las clases.

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

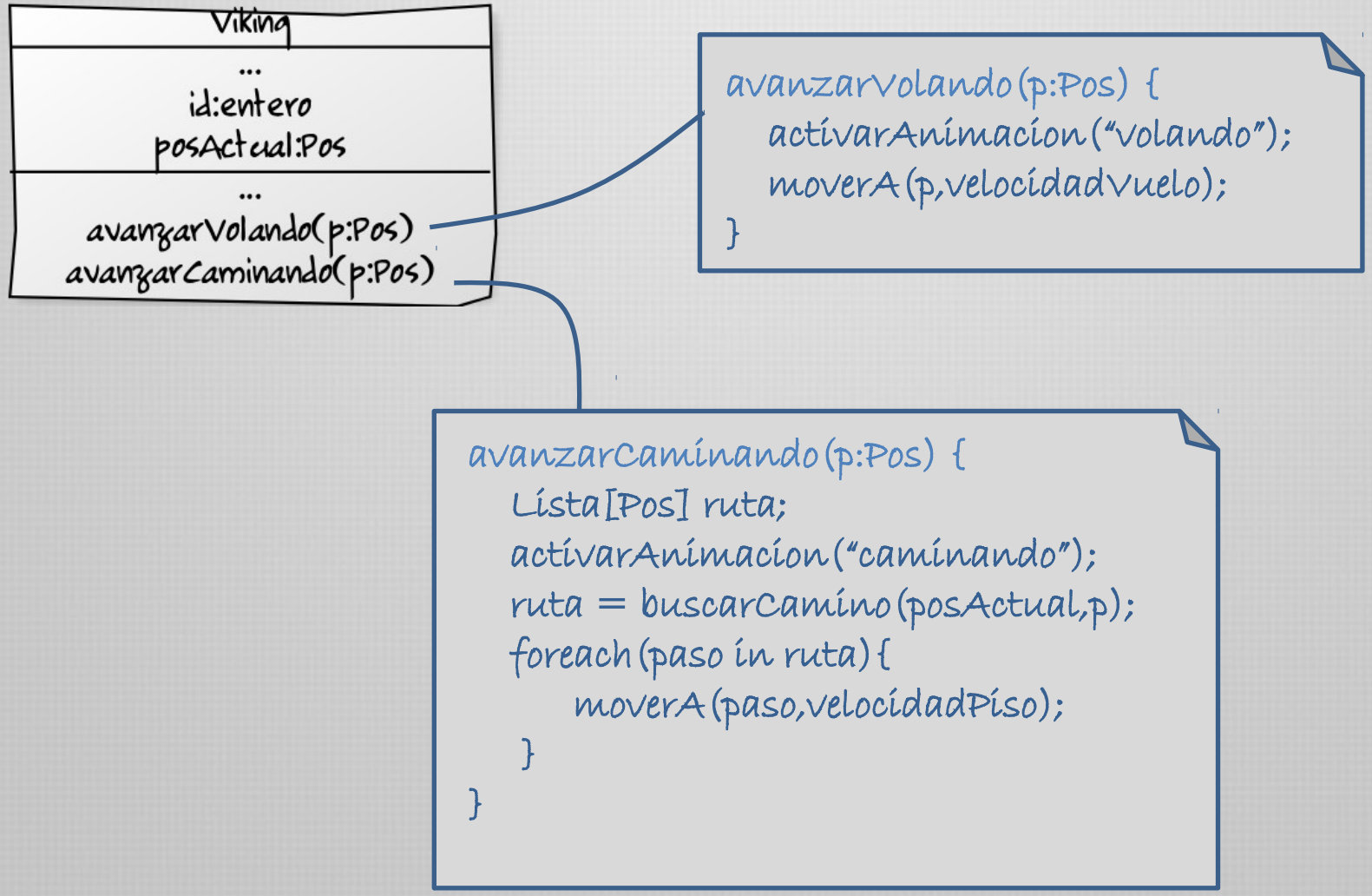
Patrón *Strategy*

Viking
Starcraft 2

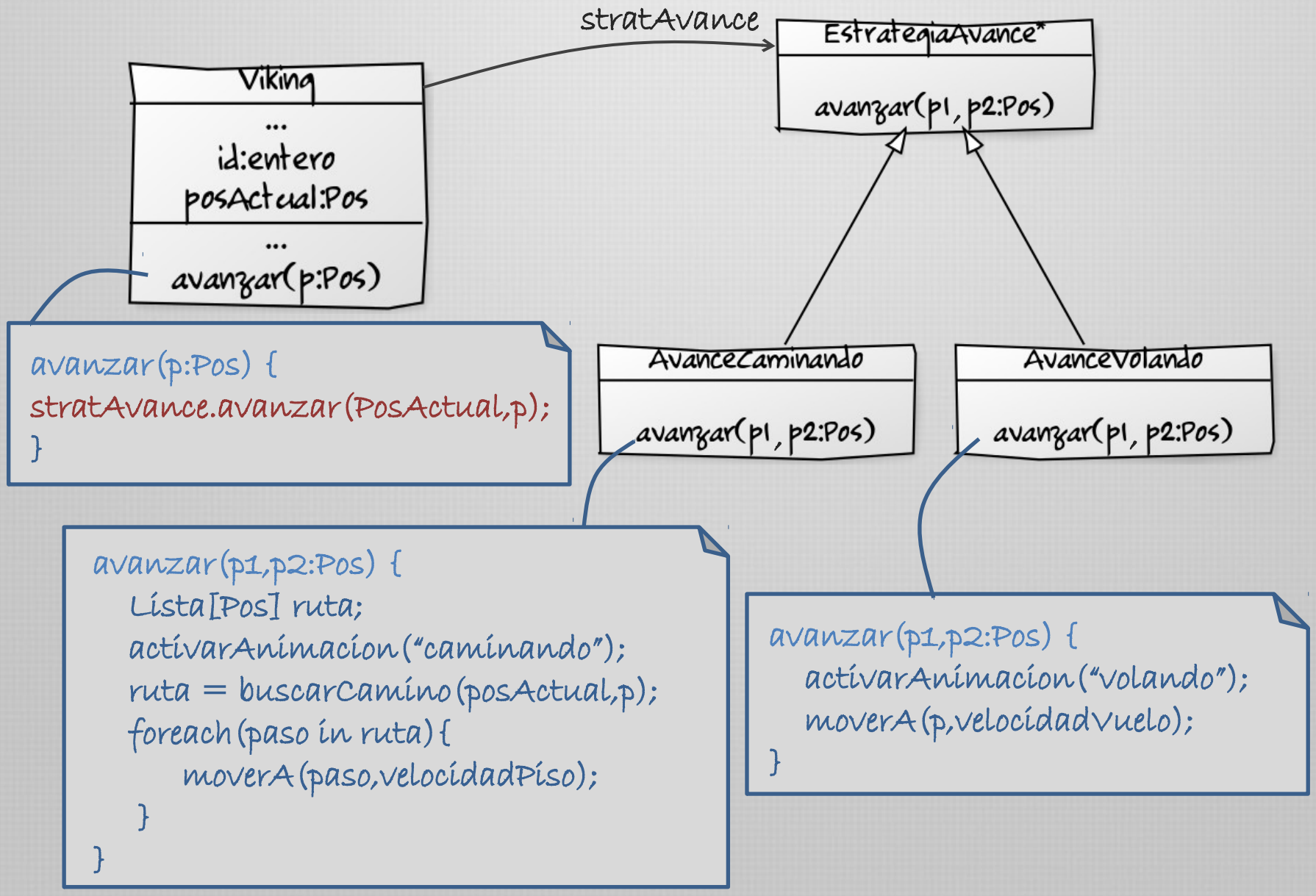




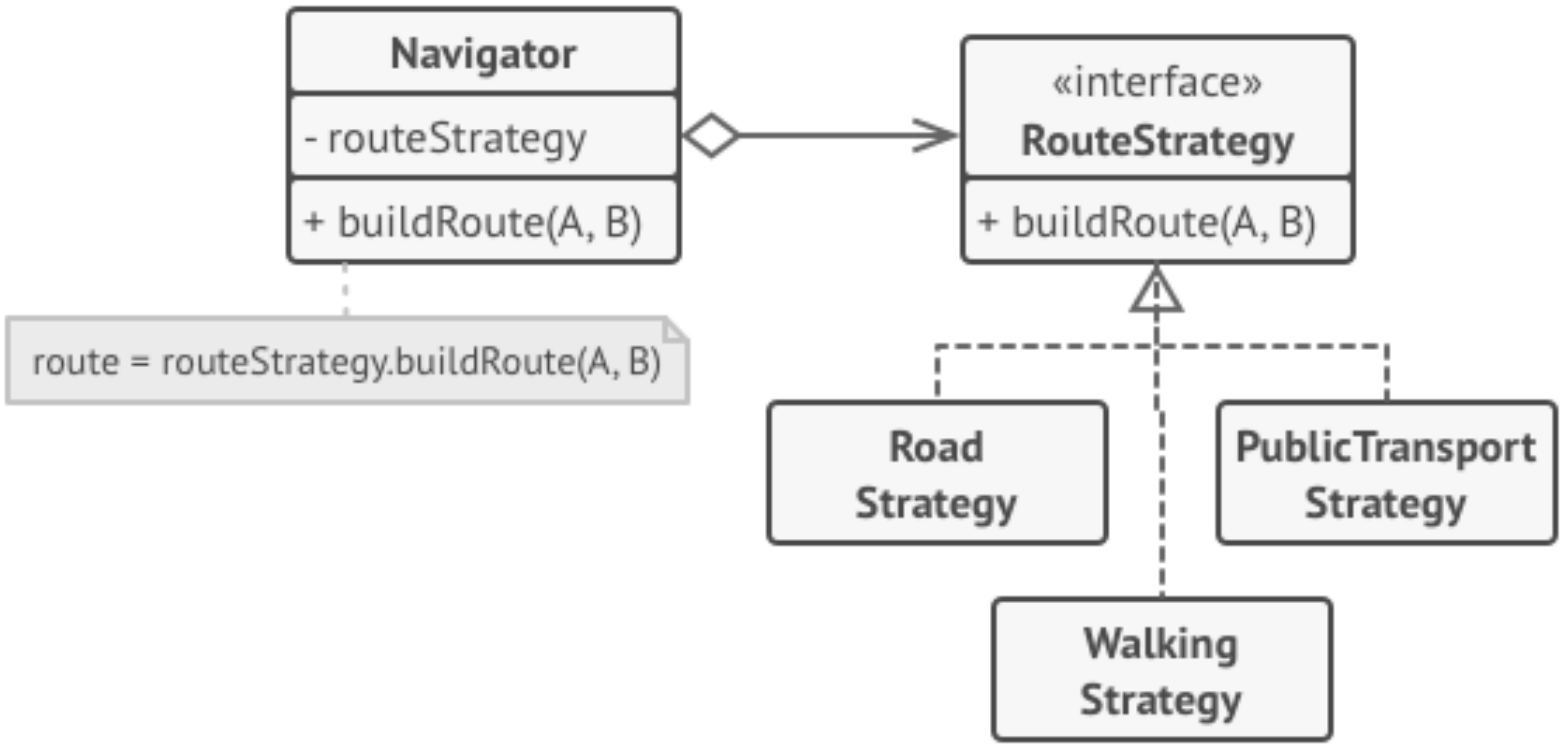
Patrón *Strategy*



Patrón *Strategy*



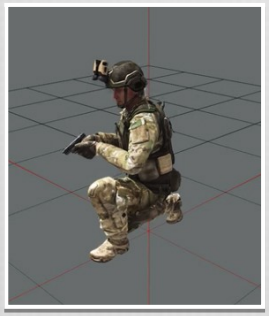
Patrón *Strategy*



Route planning strategies.

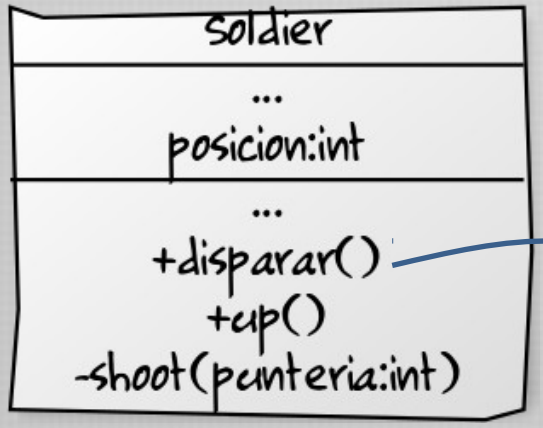
Patrón *State*

standing



kneeling

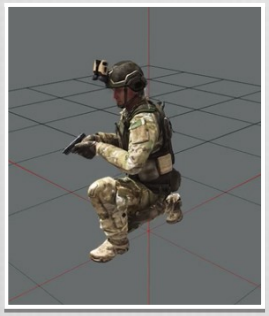
running




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

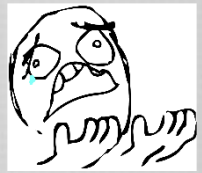
Patrón *State*

standing

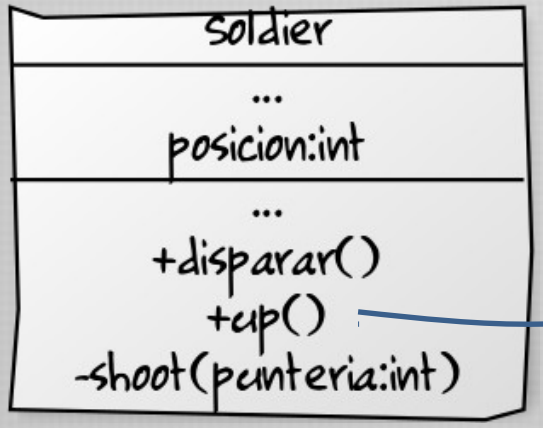


kneeling

running

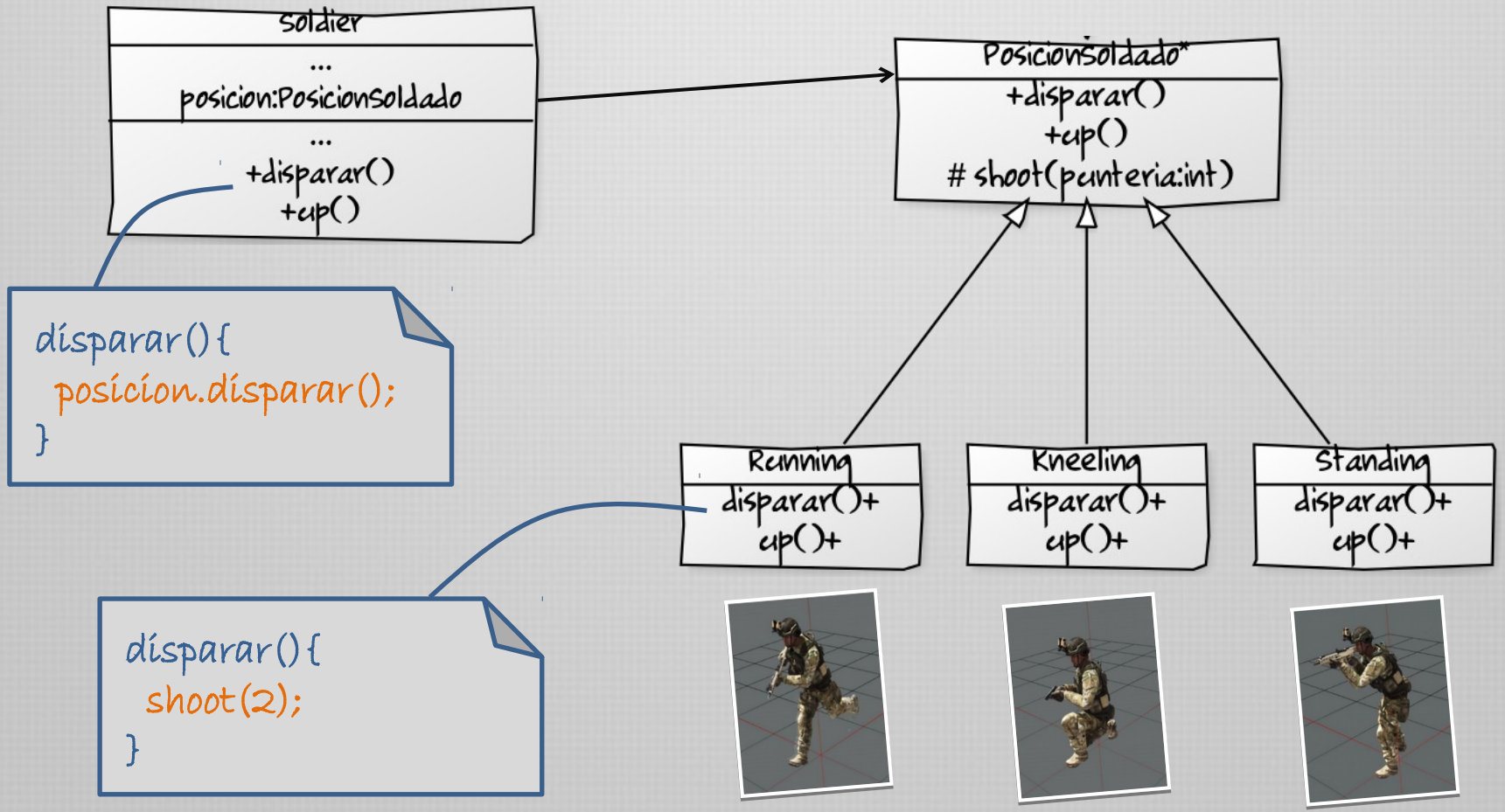


down

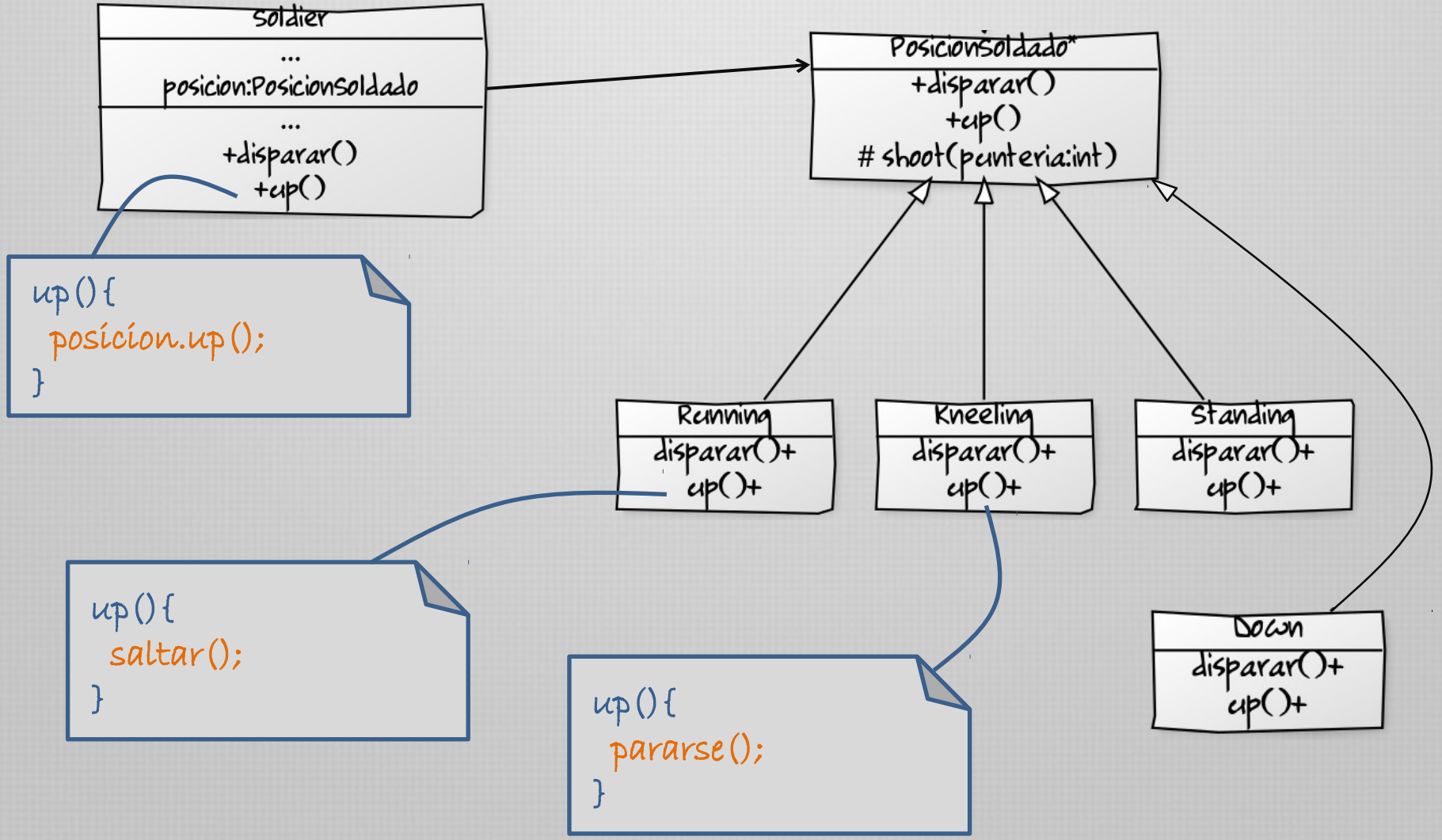



```
if (posicion=1)//standing
then
  saltar()
else if (posición=2) //kneeling
then
  pararse()
else if (posición=3) //running
then
  saltar()
```

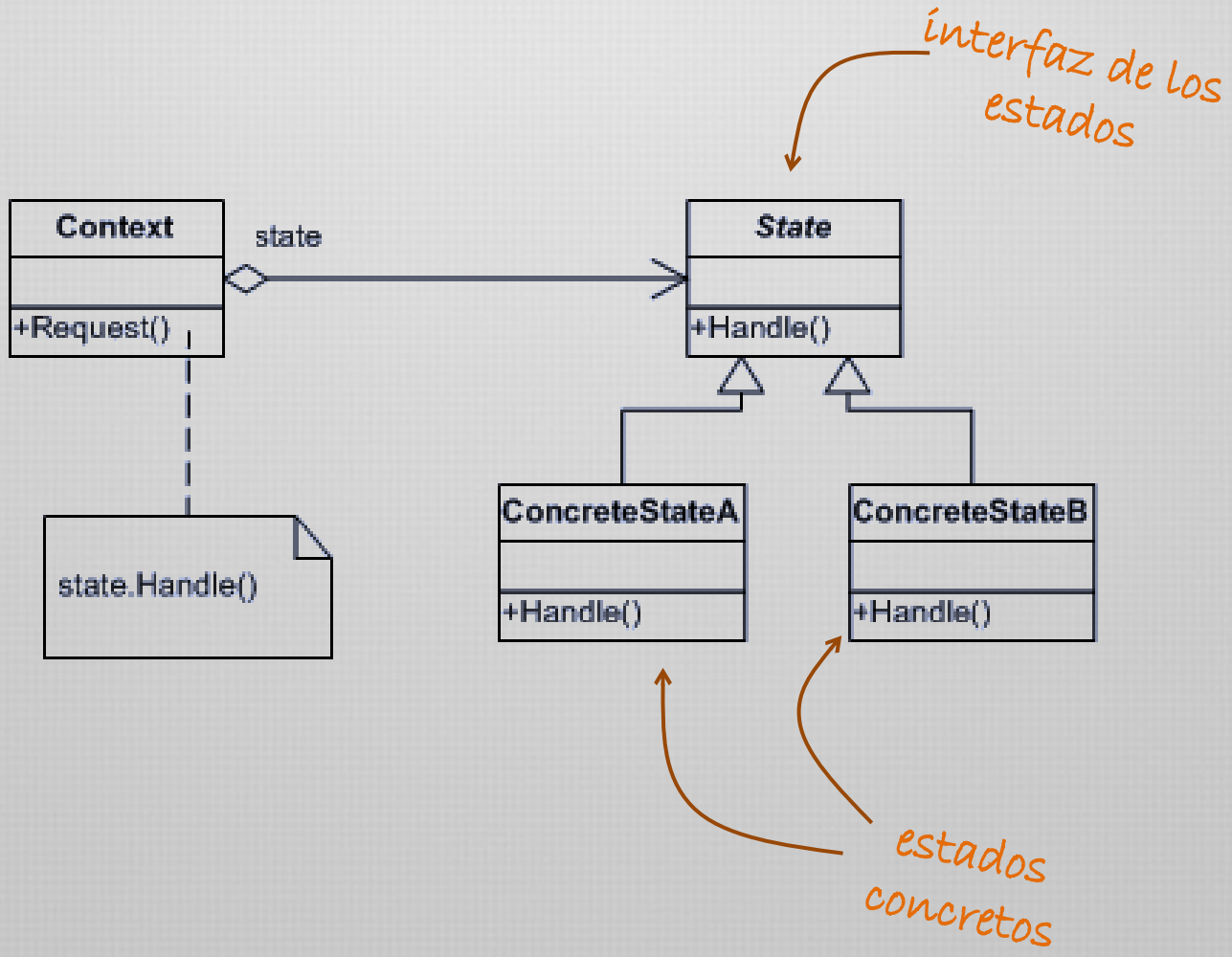
Patrón *State*



Patrón *State*



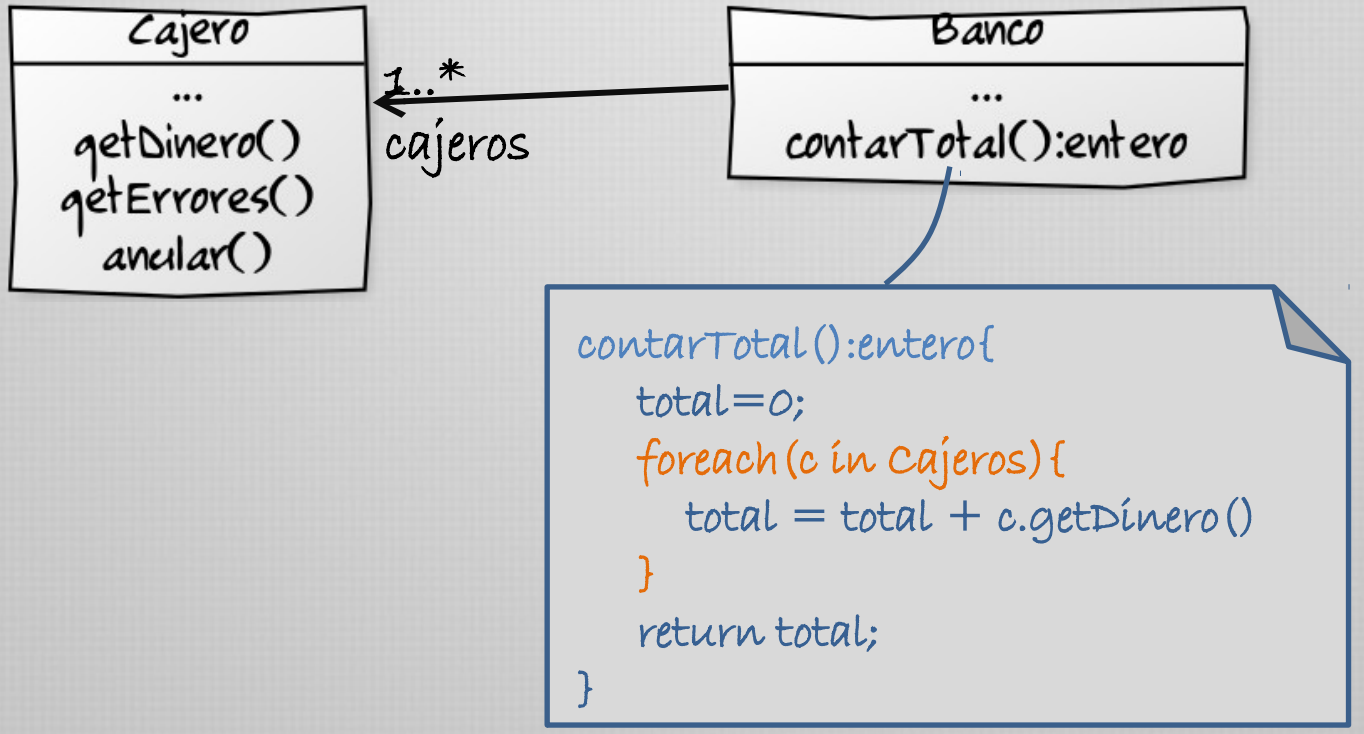
Patrón *State* - UML



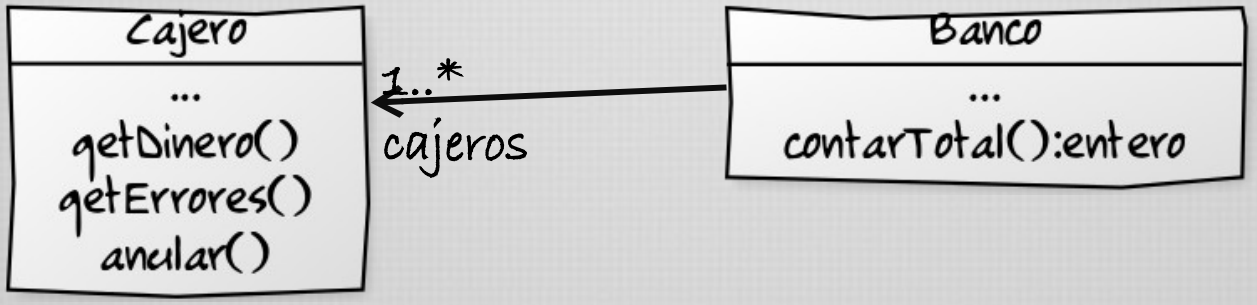
Patrón *Visitor*

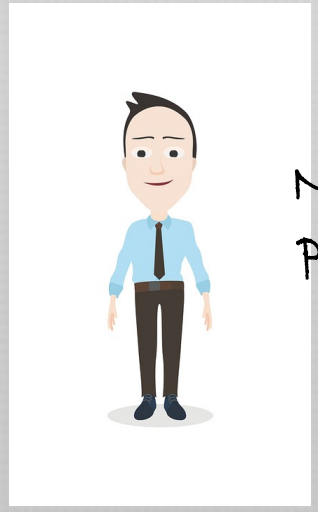
Necesito calcular cuánto dinero hay en total en todos nuestros cajeros

Patrón *Visitor*



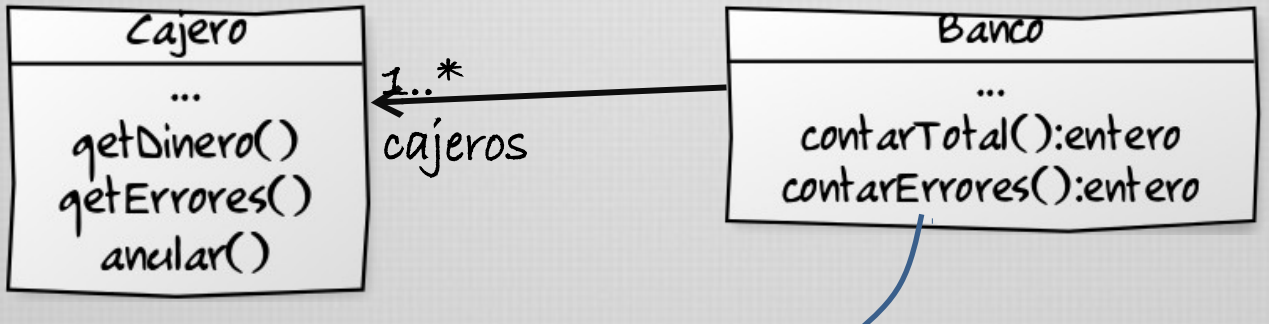
Patrón *Visitor*





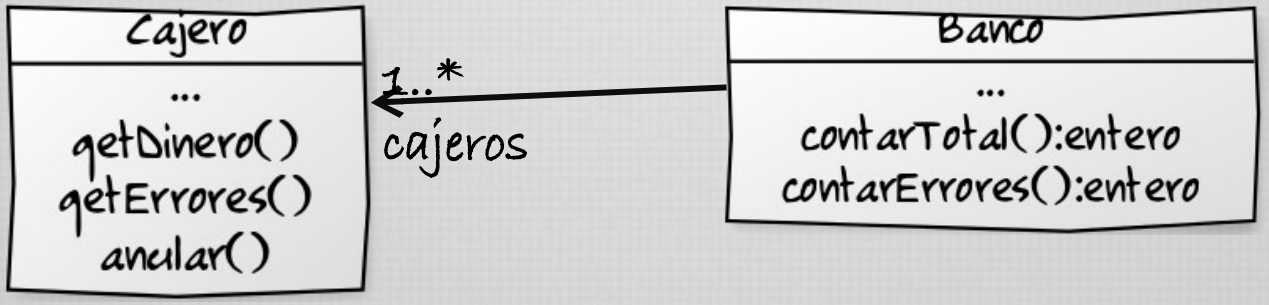
Necesito calcular el promedio de errores de todo nuestro sistema de cajeros

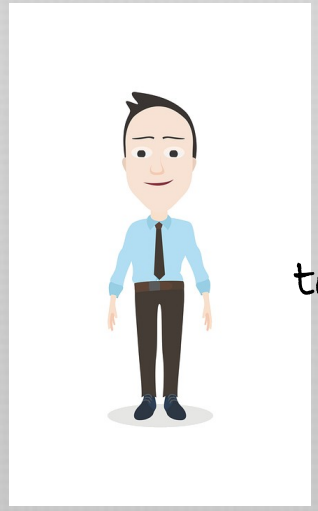
Patrón *Visitor*



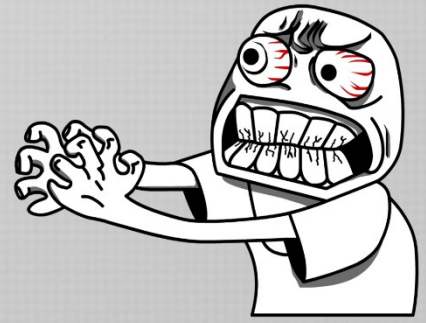

```
contarErrores():entero{
    total=0;
    cant=1;
    foreach(c in cajeros){
       total = total + c.getErrores();
       cant++;
    }
    return (total div cant);
}
```

Patrón *Visitor*

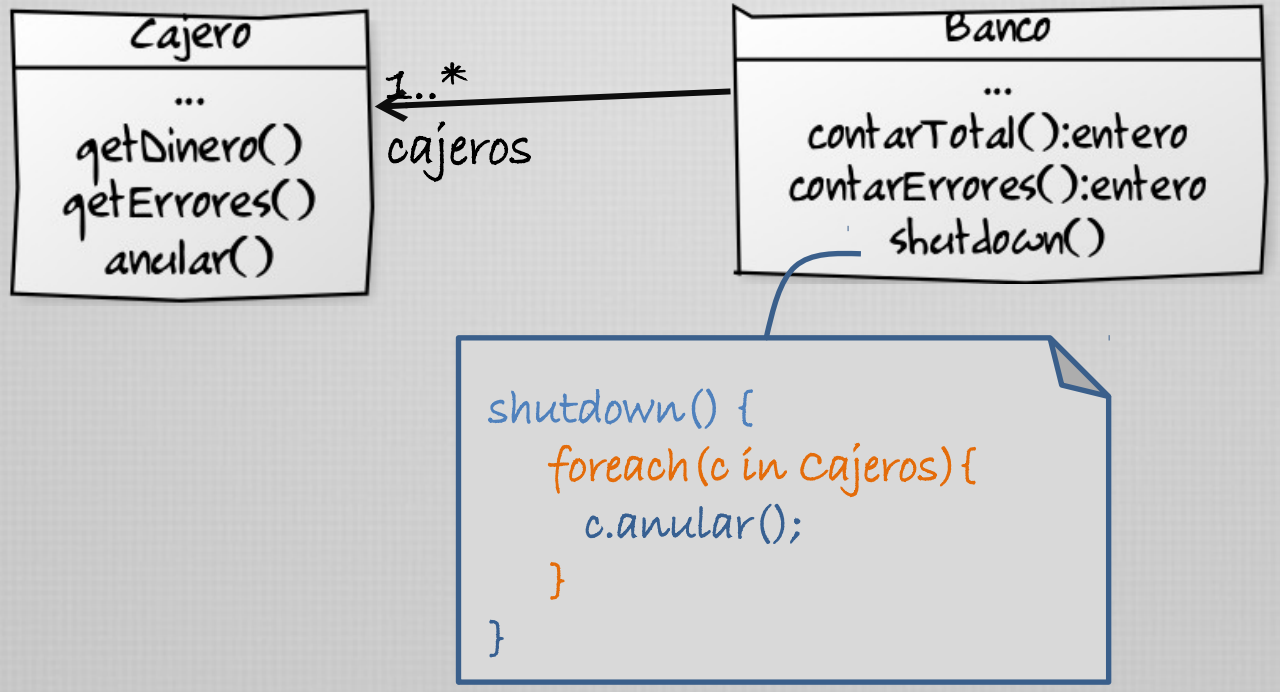




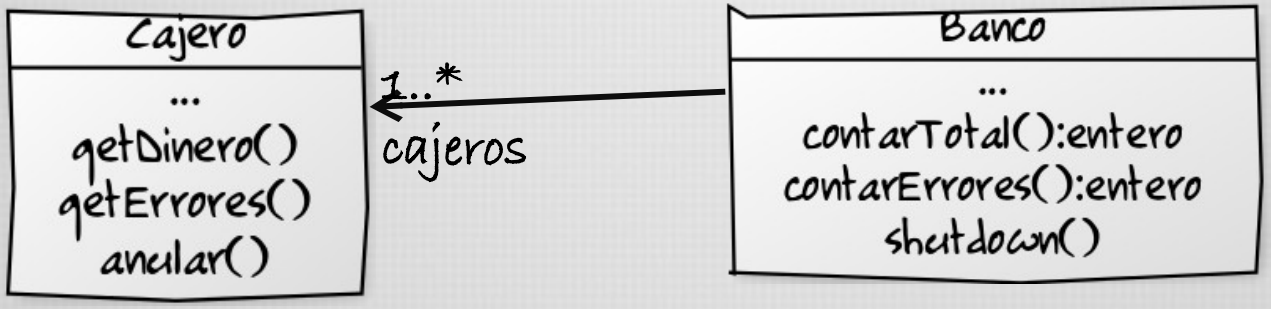
Necesito apagar todos los cajeros del sistema



Patrón *Visitor*

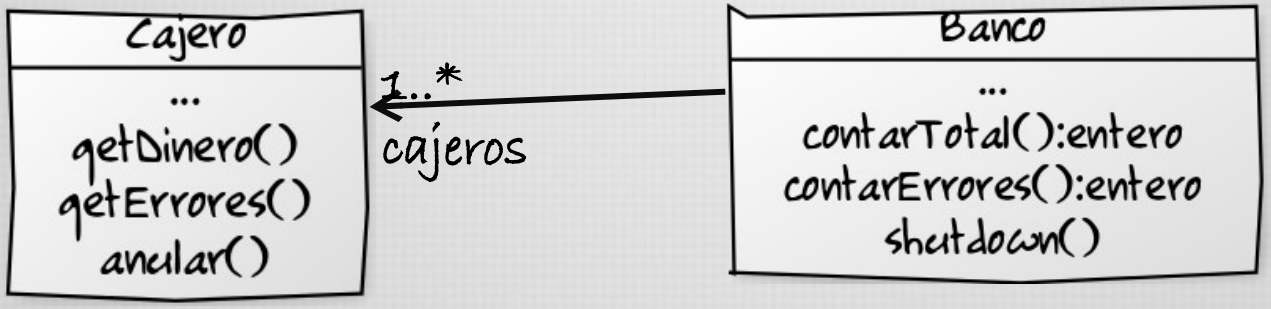


Patrón *Visitor*



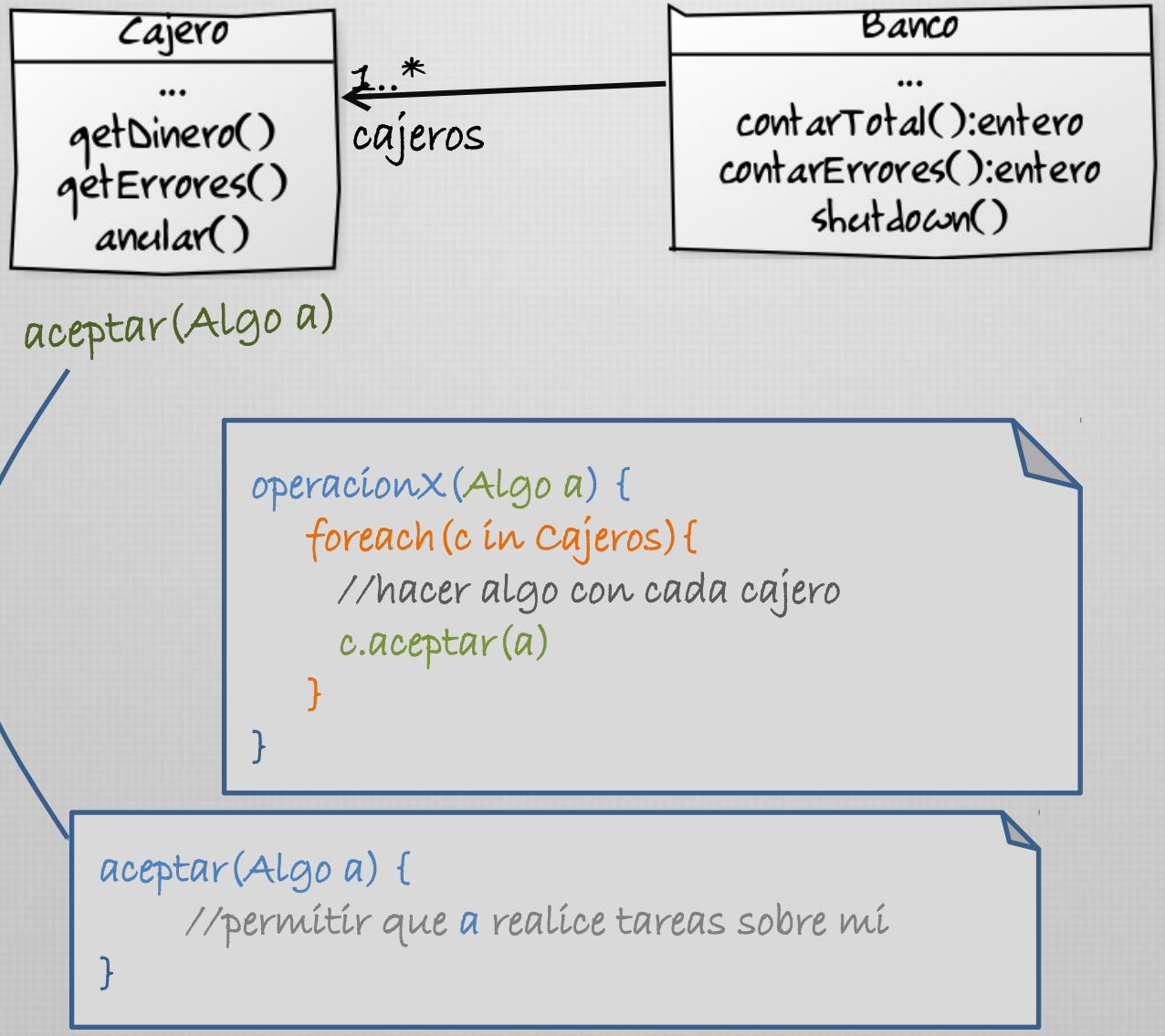

```
operacionX() {
    foreach(c in cajeros){
      //hacer algo con cada cajero
    }
}
```

Patrón *Visitor*

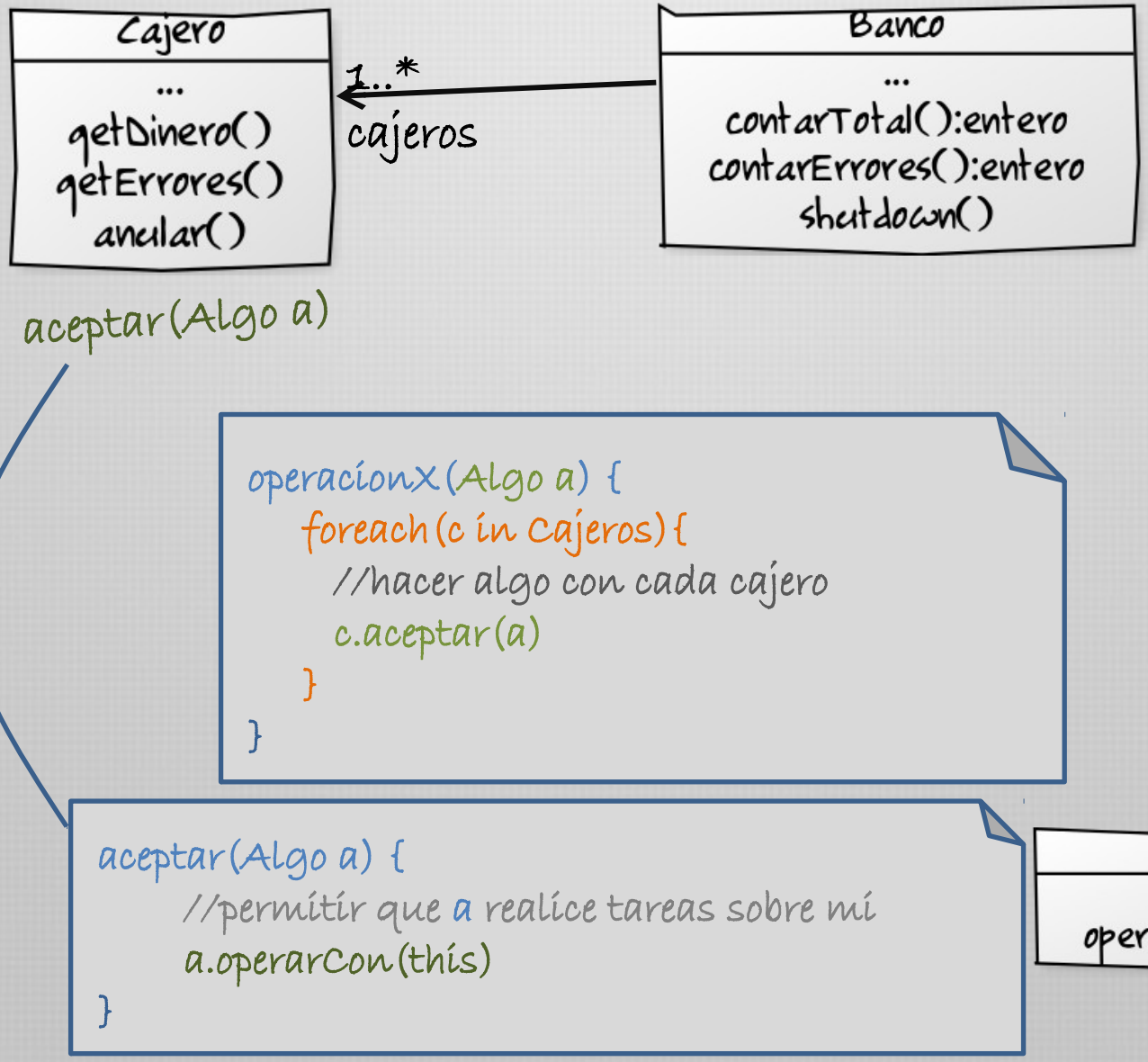



```
operacionX(Algo a) {
    foreach(c in Cajeros){
      //hacer algo con cada cajero
    }
}
```

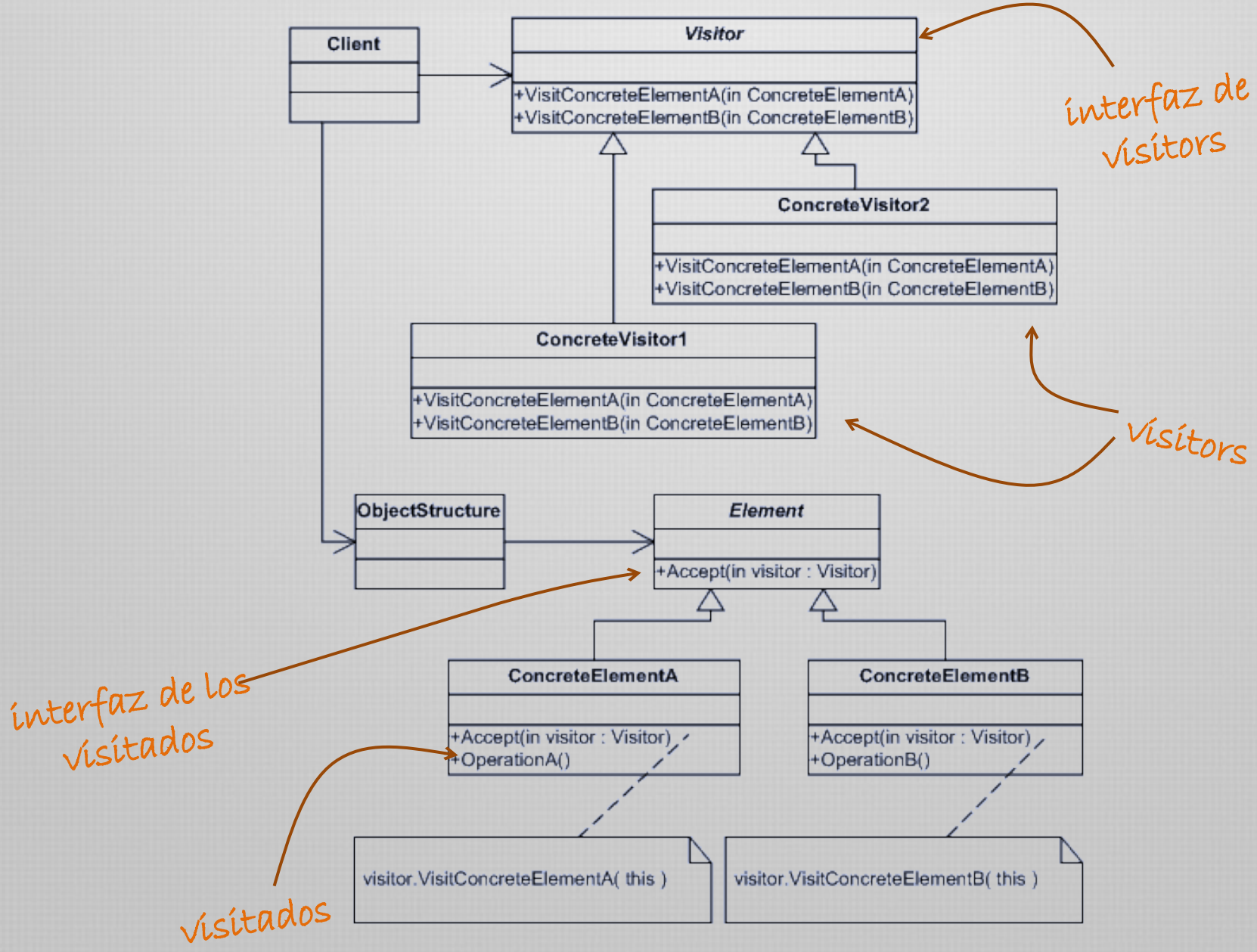
Patrón *Visitor*



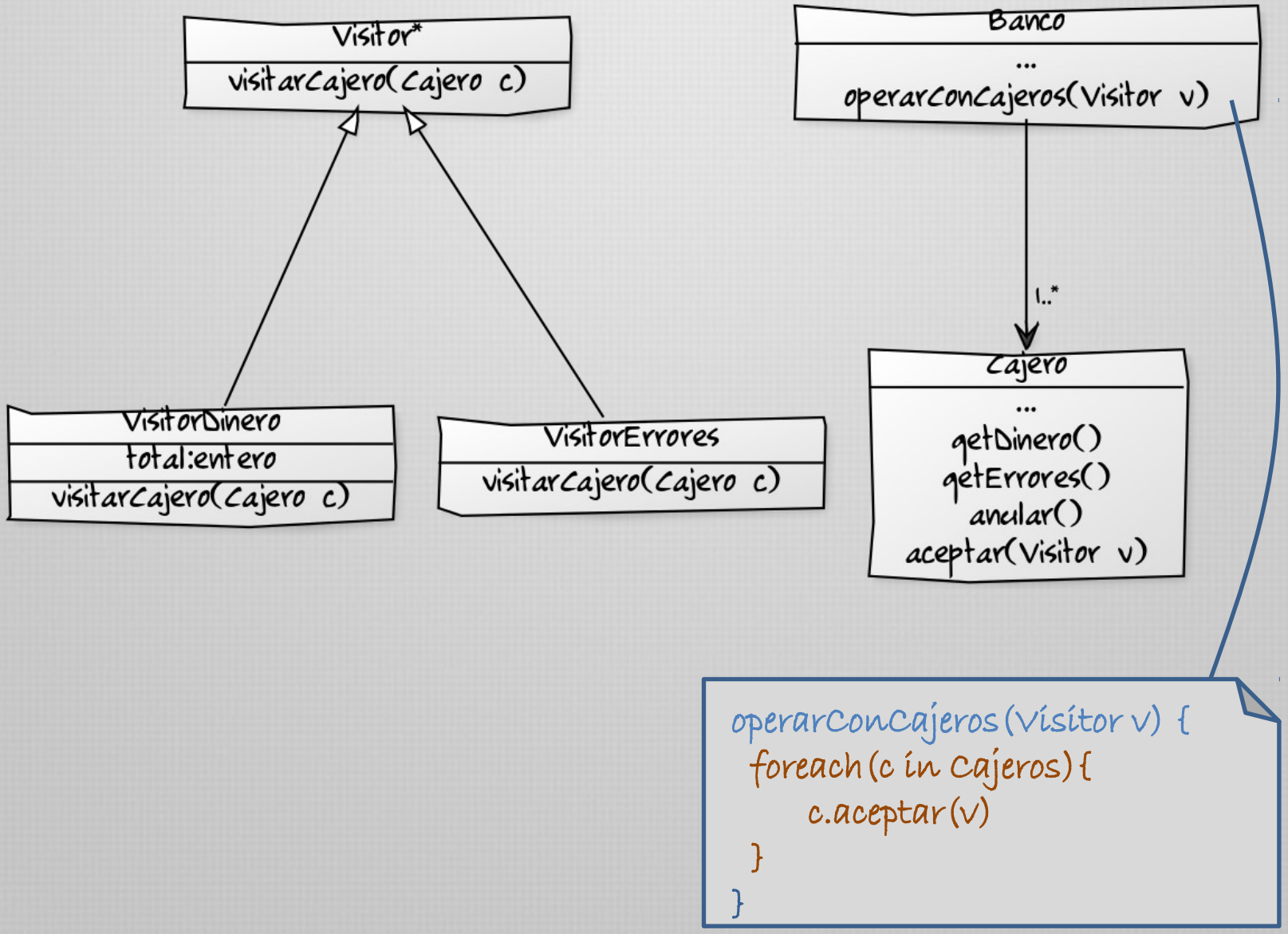
Patrón *Visitor*



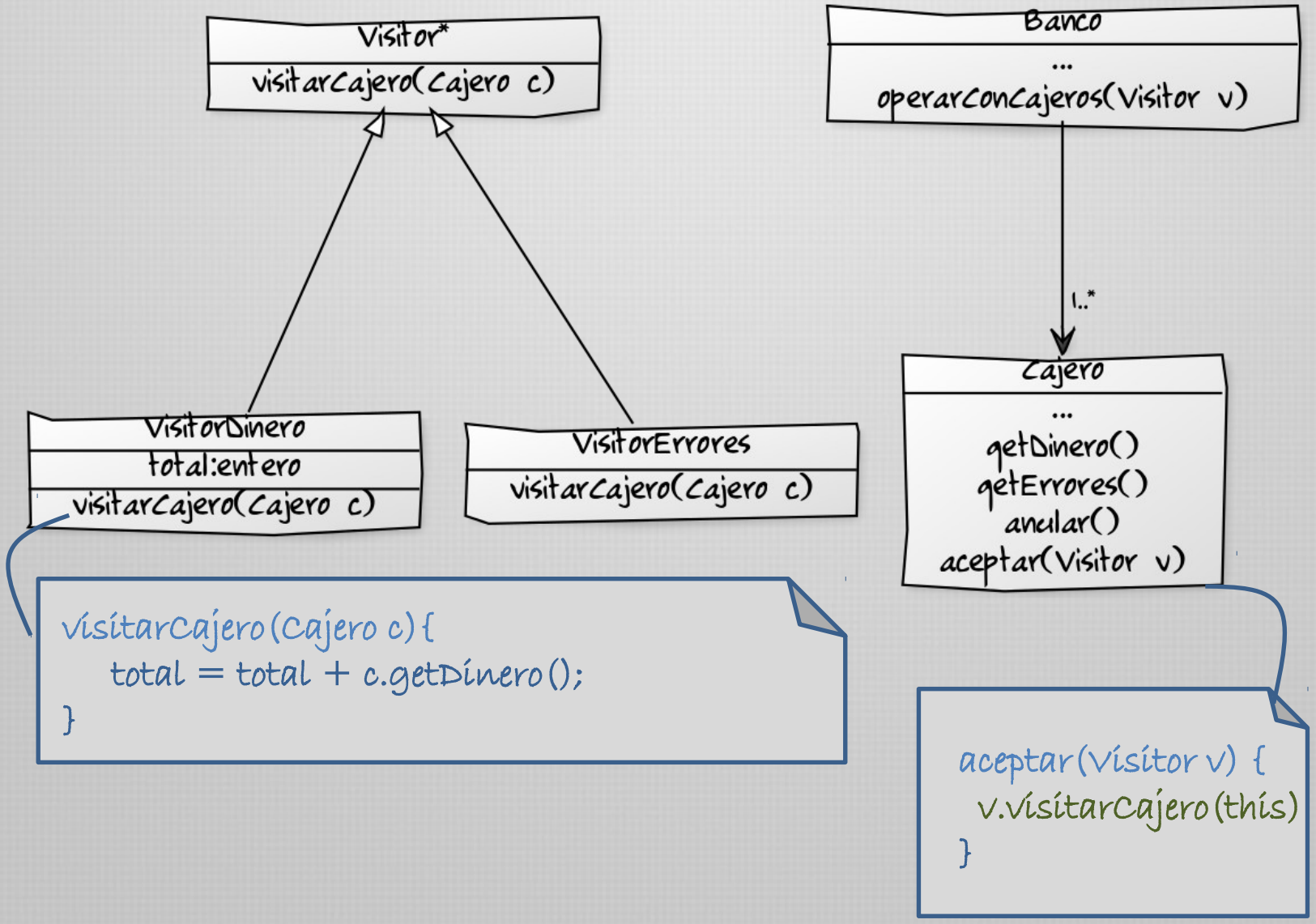
Patrón *Visitor*



Patrón *Visitor*



Patrón *Visitor*



Patrón *Visitor*

PatternCraft
Visitor Pattern

https://www.youtube.com/watch?v=KSEyIXnknoY