

Orientacion a Objetos 2

Dra Alejandra Garrido

Dr. Alejandro Fernandez

Dr. Federico Balaguer

Dr. Gustavo Rossi



[garrido, alejandro.fernandez,federico.balaguer, gustavo]@lifia.info.unlp.edu.ar



Laboratorio de Investigación y Formación en Informática Avanzada
Facultad de Informática, Universidad Nacional de La Plata
Calle 50 esq. 115 - Primer piso (1900) La Plata, Argentina
TE: (0221) 422 8252 <http://www-lifia.info.unlp.edu.ar>

✓ Según Alexander

“A pattern describes a problem that occurs once and again in a context, and describe the core of the solution in such a way that it can be used millions of times without doing the same twice”

Definicion (GOF):

A design pattern names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design. The design pattern identifies the participating classes and instances, their roles and collaborations, and the distribution of responsibilities. Each design pattern focuses on a particular object-oriented design problem or issue. It describes when it applies, whether it can be applied in view of other design constraints, and the consequences and trade-offs of its use.

***Gamma, Helms, Johnson, Vlissides: Design Patterns.
Elements of Reusable Object-Oriented Software.***

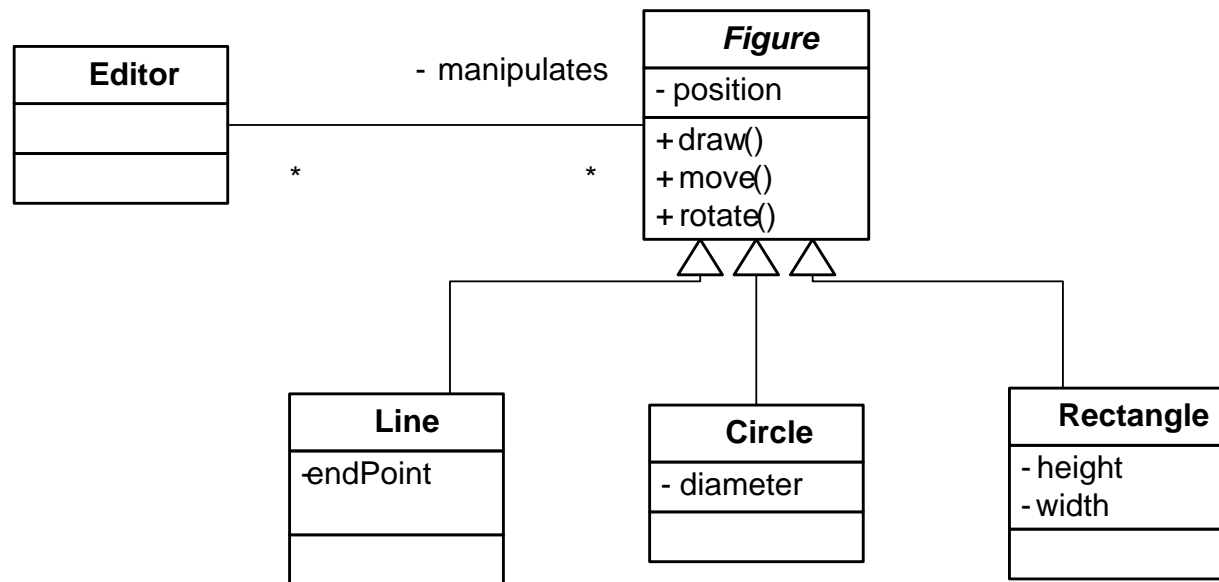
- ✓ Según GoF:
 - ✓ *Name*
 - ✓ *Intent*
 - ✓ *Motivation (Problem)*
 - ✓ *Aplicability*
 - ✓ *Structure*
 - ✓ *Participants*
 - ✓ *Collaborations*

Description....

- ✓ *Consequences*
- ✓ *Implementación*
- ✓ *Code*
- ✓ *Known Uses*
- ✓ *Related Patterns*

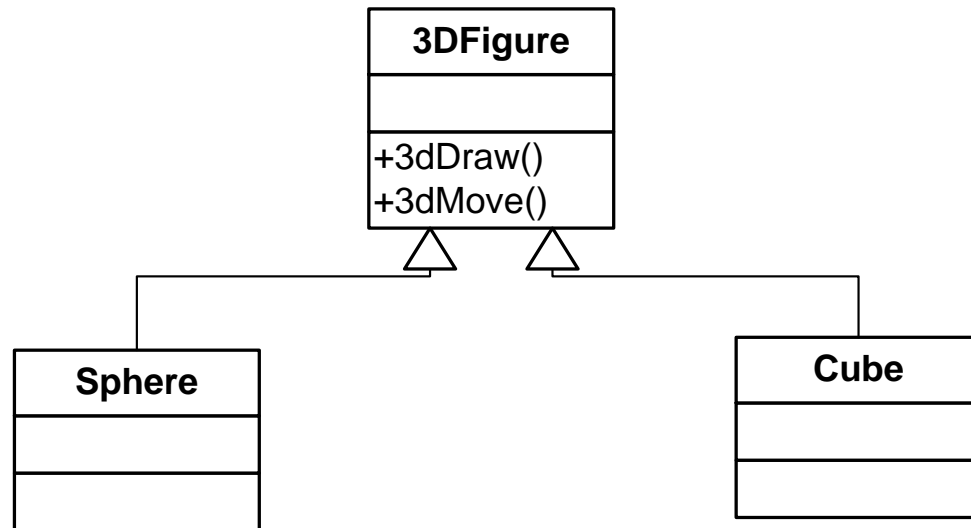
Ejemplo

- ✓ Editor grafico para manejar figuras geometricas:
- ✓ Observen polimorfismo en la relacion Editor-Figure



Ejemplo..

- ✓ Queremos extender el editor a figuras 3D



Problema

- ✓ Como integramos esta jerarquia?
- ✓ Que problemas tenemos?
- ✓ Sacrificamos polimorfismo?
- ✓ Editamos el codigo de la nueva jerarquia?

- ✓ Cambiamos todos los nombres de los métodos de la jerarquía 3d a:

draw

move

etc...

Que pasa si conseguimos una nueva version?

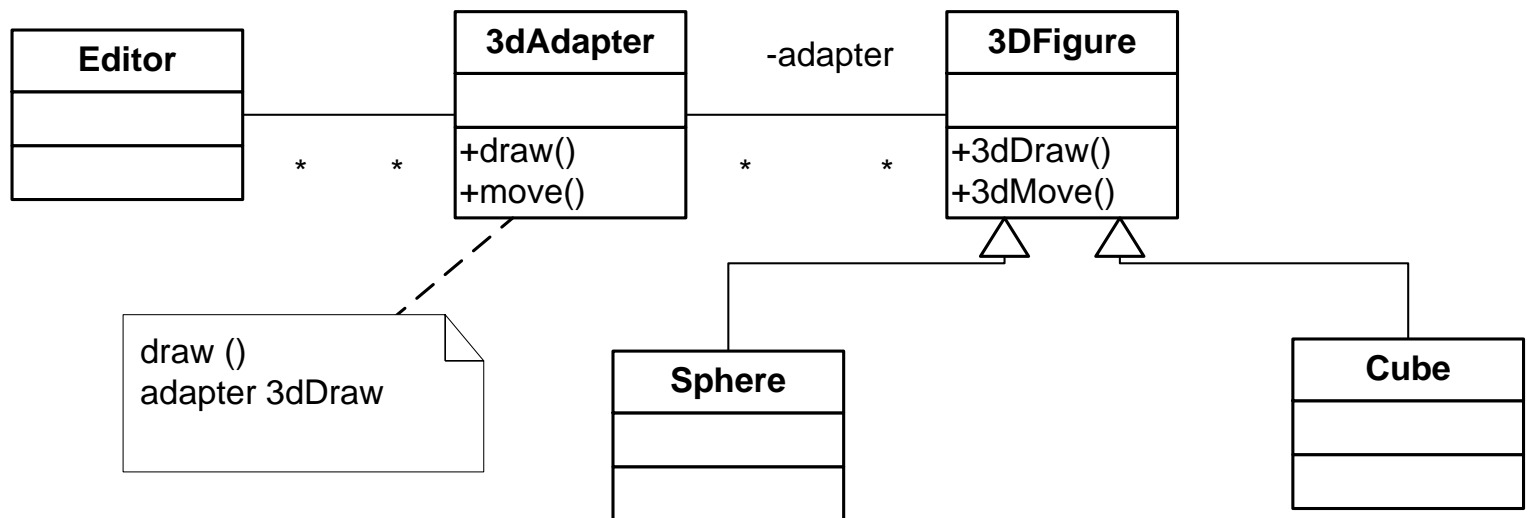
- ✓ En el código del editor metemos un if en todas las acciones que involucren un mensaje a figuras para decidir el mensaje que enviamos
- ✓ Que podría salir mal?

Solucion



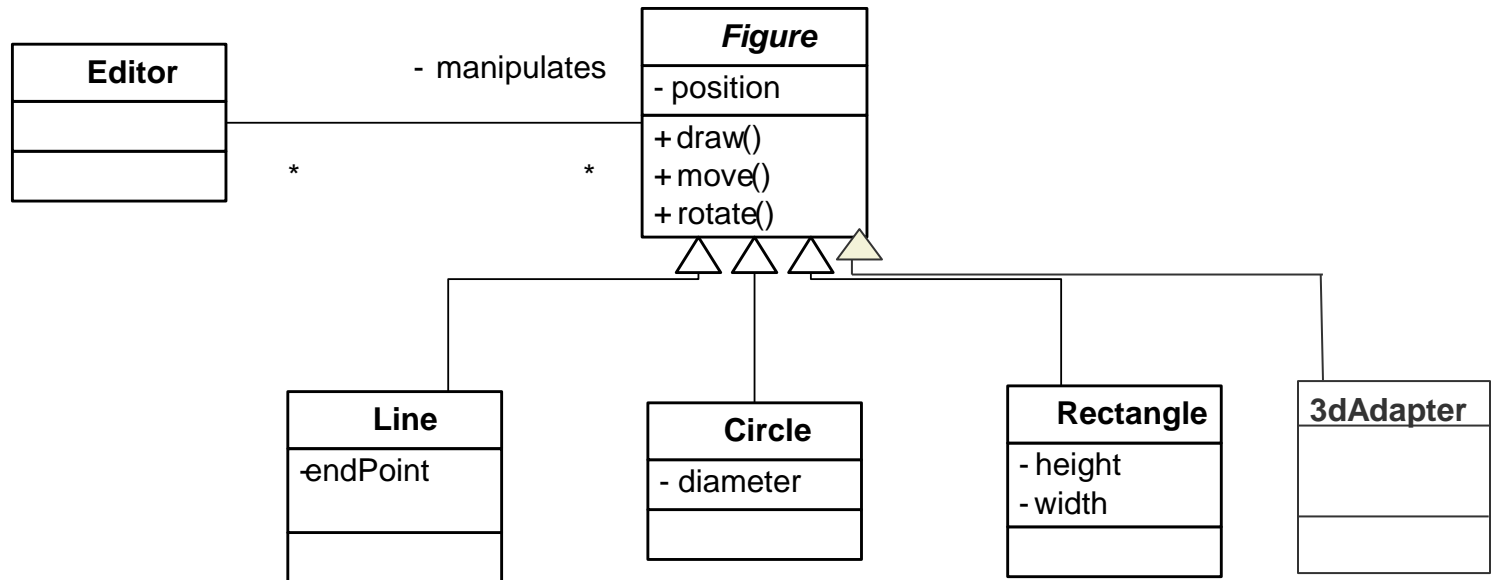
Solucion

- ✓ Cuando tratamos con interfaces incompatibles, intentar adaptarlas.



3dAdapter is sub-clase de?

3dAdapter



✓ **Intencion:**

“Convertir” la interfaz de una clase en otra que el cliente espera EL Adapter permite que ciertas clases trabajen en conjunto cuando no podrian por tener interfaces incompatibles

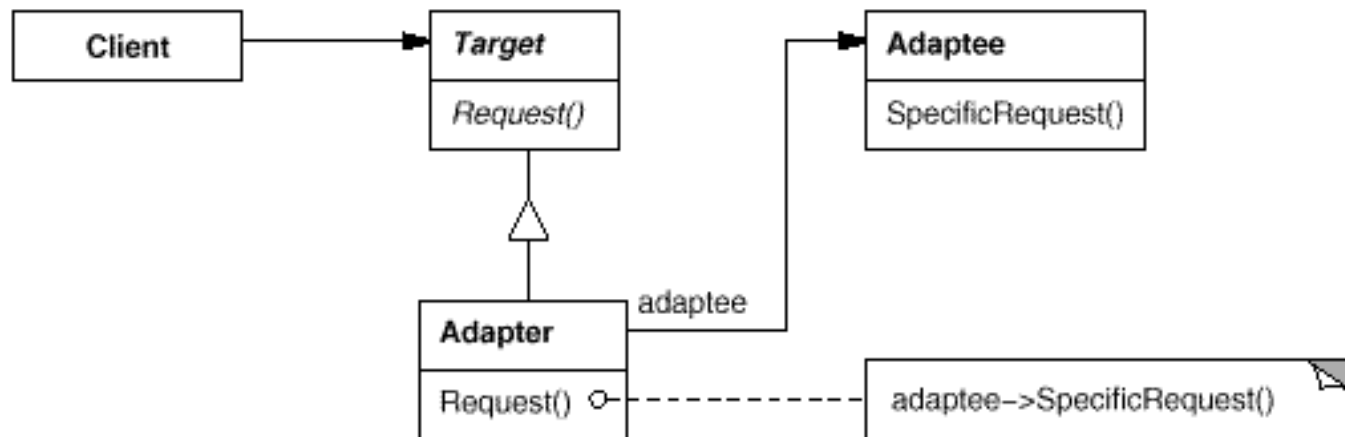
✓ **Aplicabilidad:**

Use el adapter cuando:

- ✓ Ud quiere usar una clase existente y su interfaz no es compatible con lo que precisa

Adapter

✓ Estructura



Como se lee/interpreta este diagrama?
Que representan las clases acá?

✓ **Participantes:**

✓ **Target** (Figura)

- ✓ Define la interfaz específica que usa el cliente

✓ **Client** (Editor)

- ✓ Colabora con objetos que satisfacen la interfaz de Target

✓ **Adaptee** (3DFigure)

- ✓ Define una interfaz que precisa ser adaptada

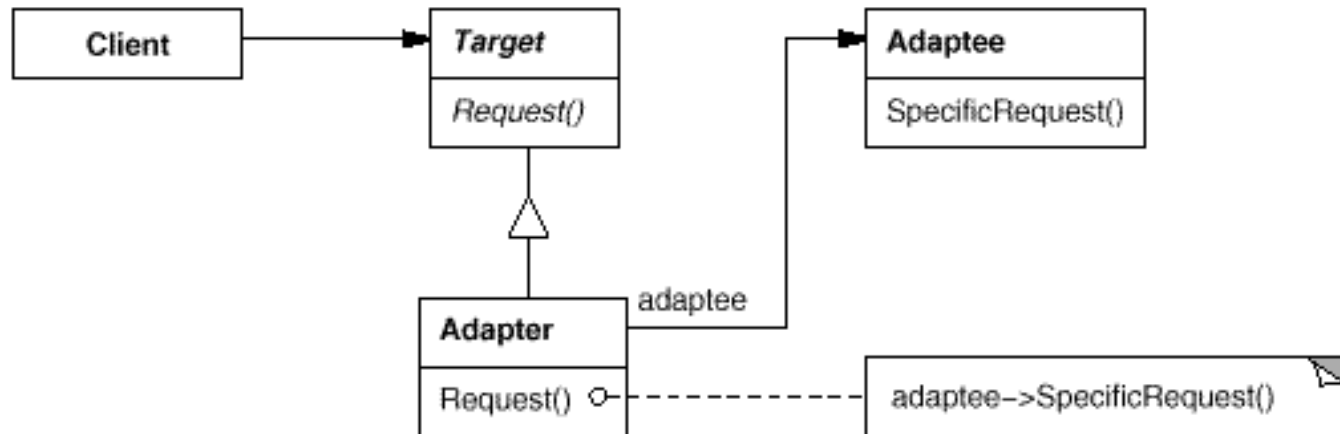
✓ **Adapter** (3DAdapter)

- ✓ Adapta la interfaz del Adaptee a la interfaz de Target

Usando Patrones

- ✓ Supongamos que conocemos patrones(e.g. Adapter).
- ✓ Como mapeamos un patron a un diseño específico?
- ✓ Como aplicamos el principio de Alexander (“use the patterns millions of times without doing the same thing twice”)?

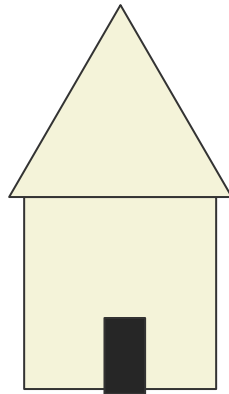
Adapter



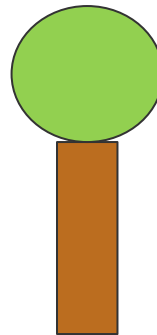
Como usamos esta informacion? Es suficiente?
Que mas necesitamos?

Problema

- ✓ Supongamos que queremos manejar figuras compuestas y tratarlas como figuras simples (moverlas, rotarlas, etc).



Casa= Cuerpo + Techo
Cuerpo= Puerta + Pared



Arbol= Tronco + Copa

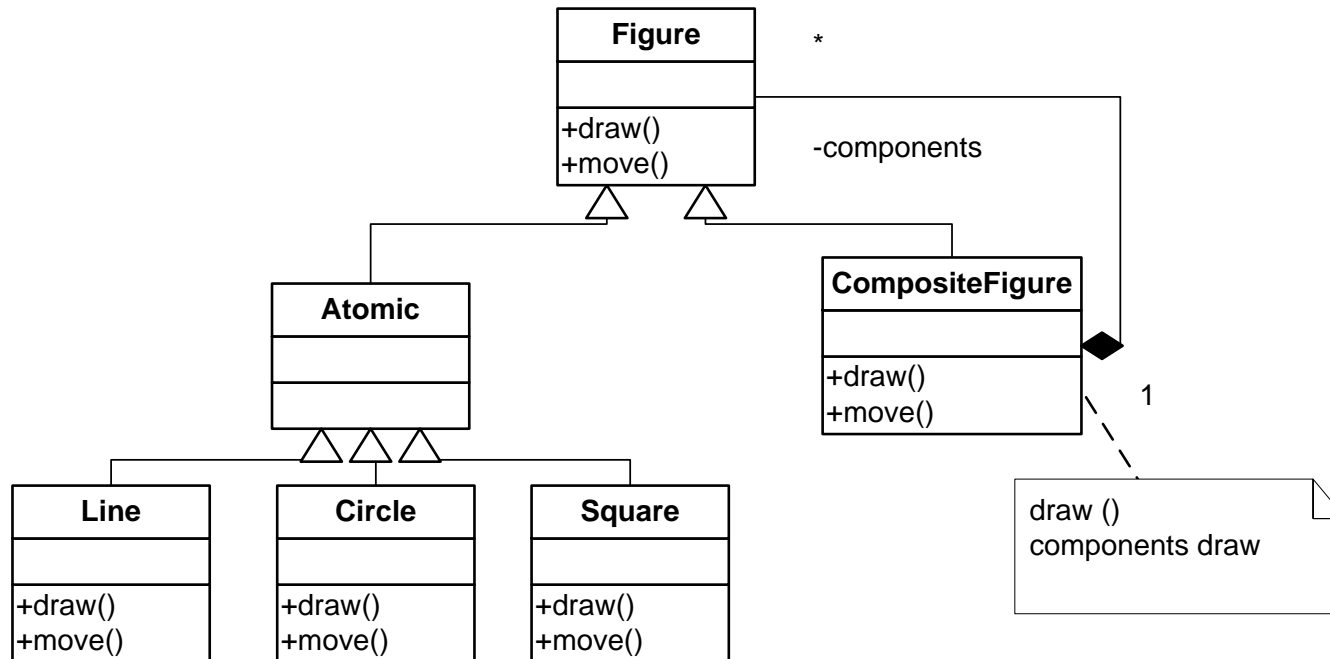
Propiedad= Casa + Arbol

Soluciones?

- ✓ Podemos tener un arreglo de figuras y marcarlas como partes de otras....
- ✓ Como seria tal arreglo cuando hay composiciones muy “profundas?”
- ✓ Por ejemplo “Barrio”

Una solución mas modular

✓ Tratarlas uniformemente



Obsérvese la composición recursiva de **CompositeFigure** con **Figure**, usando la colección “components”, cada uno de los cuales es una **Figure**

Como funciona?

- ✓ El editor solo maneja figuras
- ✓ Mantenemos la interfaz polimorfica
- ✓ Problemas? Como creamos figuras compuestas?

Como sería una
instancia de la clase
CompositeFigure?

Pattern Composite

✓ Intent

Componer objetos en estructuras de arbol para representar jerarquias parte-todo. El Composite permite que los clientes traten a los objetos atomicos y a sus composiciones uniformemente

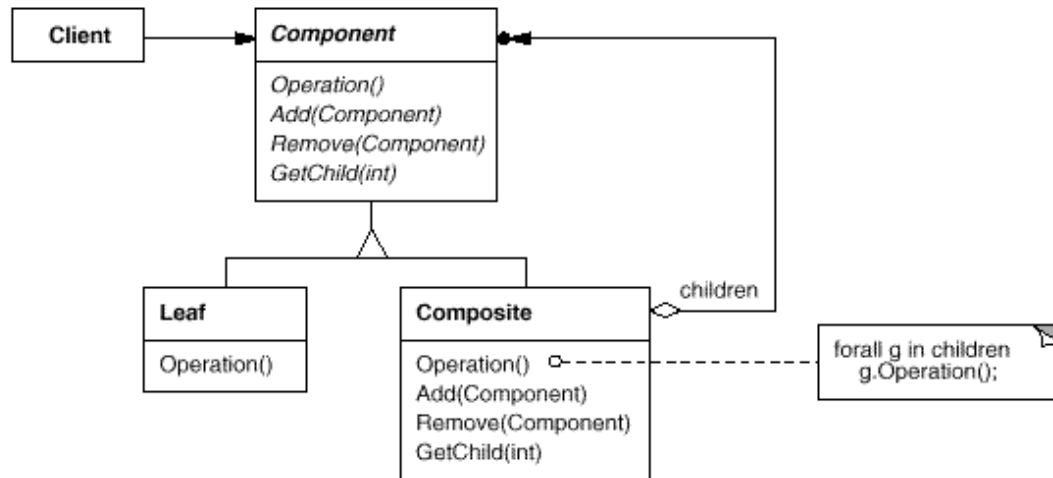
✓ Applicability

Use el patron Composite cuando

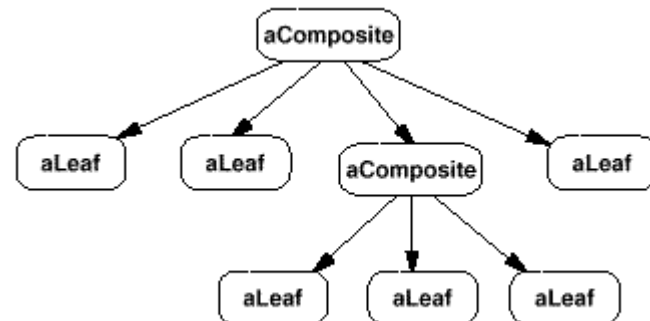
- ✓ quiere representar jerarquias parte-todo de objetos.
- ✓ quiere que los objetos “clientes” puedan ignorar las diferencias entre composiciones y objetos individuales. Los clientes trataran a los objetos atomicos y compuestos uniformemente.

Pattern Composite

✓ Structure



Una estructura compuesta
tipica se vera asi



✓ Participants

✓ Component (Figure)

- ✓ Declara la interfaz para los objetos de la composicion.
- ✓ Implementa comportamientos default para la interfaz comun a todas las clases
- ✓ Declara la interfaz para definir y acceder “hijos”.
- ✓ (opcional) define una interfaz para para acceder el “padre” de un componente en la estructura recursiva y la implementa si es apropiado.

Composite

- ✓ **Leaf** (Rectangle, Line, Text, etc.)
 - ✓ Representa arboles “hojas” in la composicion. Las hojas no tienen “hijos”.
 - ✓ Define el comportamiento de objetos primitivos en la composicion.
- ✓ **Composite** (CompositeFigure)
 - ✓ Define el comportamiento para componentes con “hijos”.
 - ✓ Contiene las referencias a los “hijos”.
 - ✓ Implementa operaciones para manejar “hijos”.

✓ Consecuencias

El patron composite

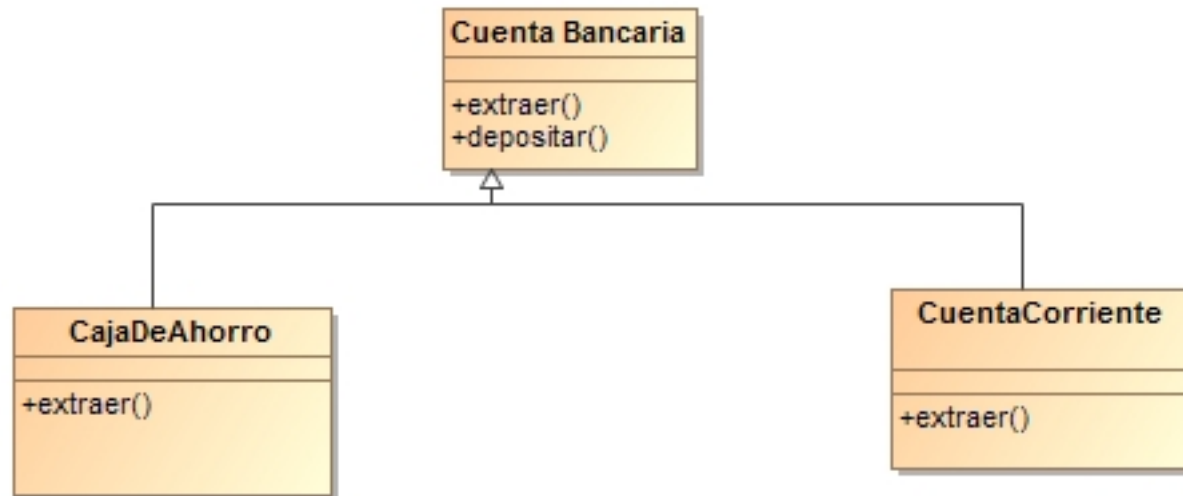
- ✓ Define jerarquías de clases consistentes de objetos primitivos y compuestos. Los objetos primitivos pueden componerse en objetos complejos, los que a su vez pueden componerse y así recursivamente. En cualquier lugar donde un cliente espera un objeto simple, puede aparecer un compuesto.
- ✓ Simplifica los objetos cliente. Los clientes pueden tratar estructuras compuestas y objetos individuales uniformemente. Los clientes usualmente no saben (y no deberían preocuparse) acerca de si están manejando un compuesto o un simple. Esto simplifica el código del cliente, porque evita tener que escribir código taggeado con estructura de decision sobre las clases que definen la composición

Pattern Composite

- ✓ Hace mas fácil el agregado de nuevos tipos de componentes porque los clientes no tienen que cambiar cuando aparecen nuevas clases componentes.
- ✓ Puede hacer difícil restringir las estructuras de composición cuando hay algún tipo de conflicto (por ejemplo ciertos compuestos pueden armarse solo con cierto tipo de atómicos)

Problema

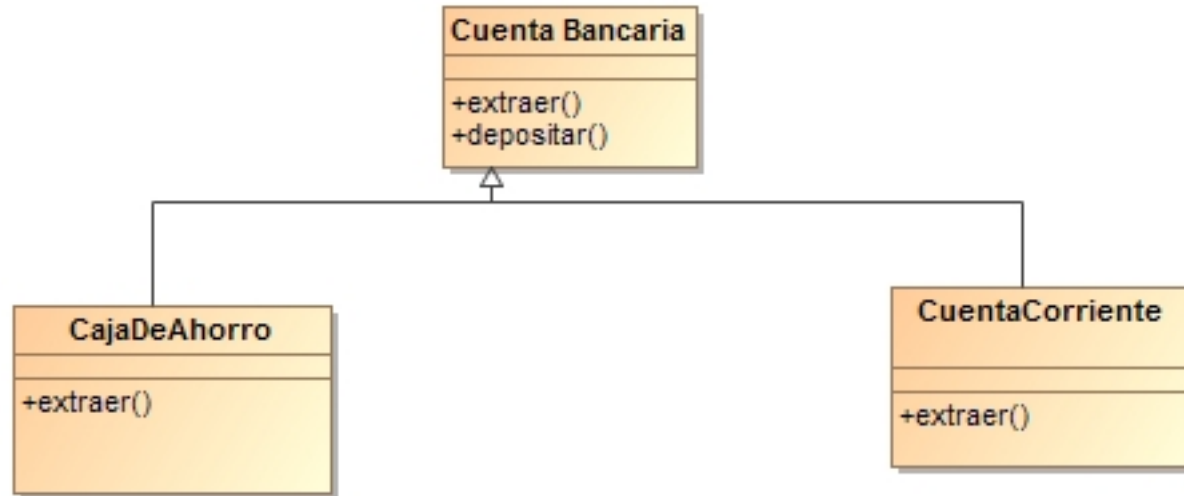
- ✓ Supongamos una jerarquía de cuentas bancarias y una operación con “variantes” de acuerdo a la cuenta



El Metodo Extraer

- ✓ En la Clase Caja de Ahorro tiene que controlar el saldo contra 0 y la cantidad de extracciones
- ✓ En la Clase Cuenta Corriente tiene que controlar el saldo contra un “rojo” permitido y la situacion impositiva del cliente

Solucion



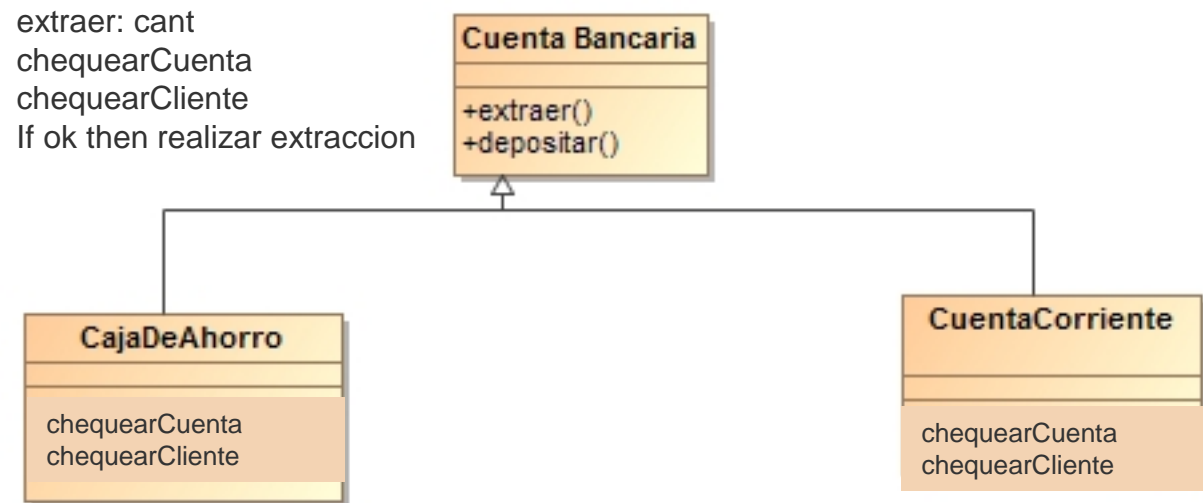
extraer (cant)
If `cantExtracciones < 5` and `saldo > cant` then `saldo=saldo-cant`

extraer (cant)
If `saldo-cant > rojoPermitido` and `cliente pagoImpuestos`
then `saldo=saldo -cant`

Problemas con esta Solucion

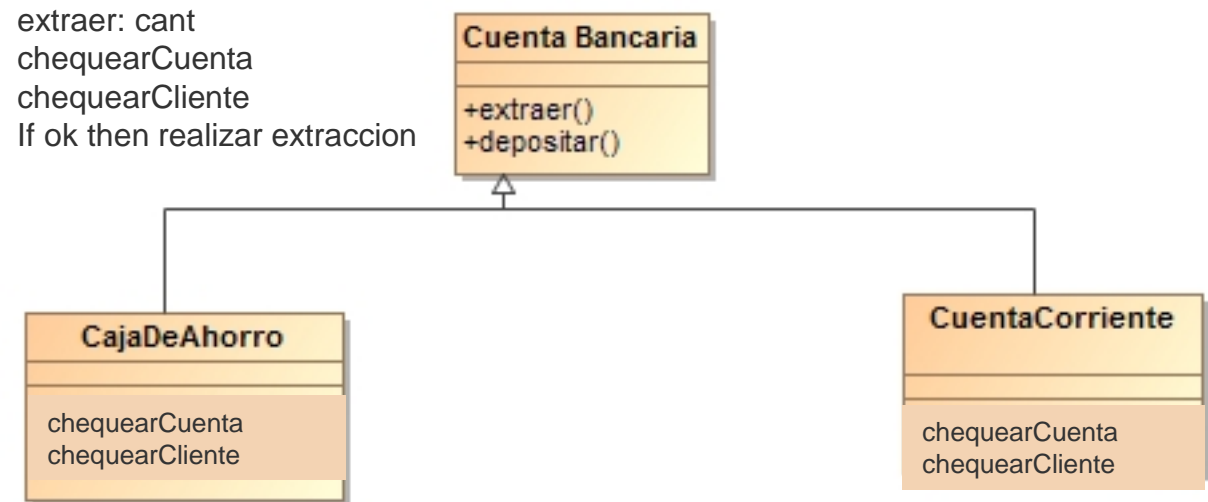
Template Method

- Abstraer cada “parte” del código de las sub-clases que difiere
- Expresarlo como un método más primitivo
- Definir el algoritmo en la clase abstracta y expresar la variabilidad en los métodos primitivos en cada sub-clase



Observaciones

- ✓ En el código de “extraer” pareciera que los mensajes fluyen “hacia abajo” (de la clase abstracta a las sub-clases)
- ✓ Recordar que “extraer” se ejecuta como si estuviera en las sub-clases con lo cual el “this” es el mismo objeto receptor



Template Method

✓ Intent:

- ✓ Definir el esqueleto de un algoritmo en un metodo, difiriendo algunos pasos a las subclases. El template method permite que las subclases re definas ciertos aspectos de un algoritmo sin cambiar su estructura

✓ Aplicabilidad

- ✓ Para implementar las partes invariantes de un algoritmo una vez y dejas que las sub-clases implementen los aspectos que varian

Template Method

✓ Structure

