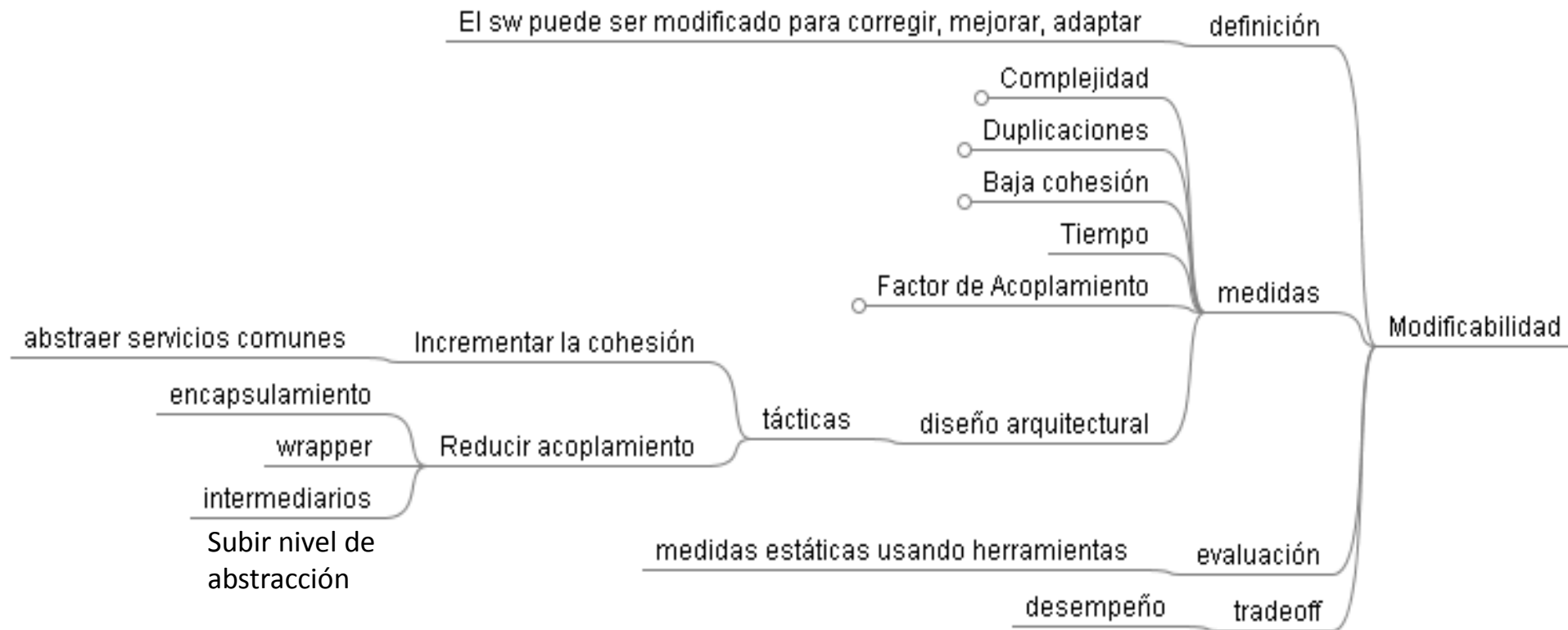


# Modificabilidad

El sw puede ser modificado para  
corregir, mejorar, adaptar

# Modificabilidad



# Propuestas para modificabilidad en la historia

Simula software



Dahl y Nygaard en la época de desarrollo de Simula.

C++ soporta la POO



1968

1972

1974

1980

1990

1994

OTAN Conference  
Crisis del Software.



Programación estructurada

NO MORE  
SPAGHETTI  
CODE



Programación modular



POO como una de las  
mejores maneras para  
resolver problemas.

Aparición de **CVS**

Concurrent Version System  
para el manejo de  
versiones

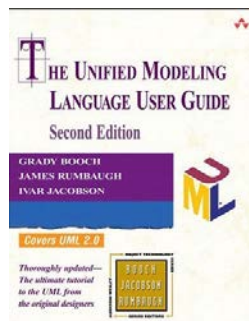
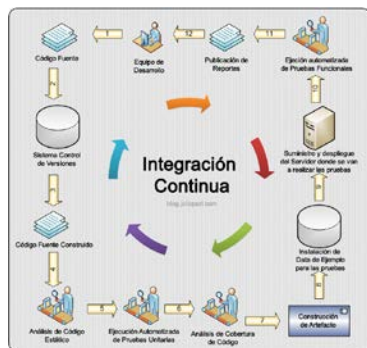


Primer release de Java  
con Javadoc

# Propuestas para modificabilidad en la historia



Extreme Programming



UML 2



1996

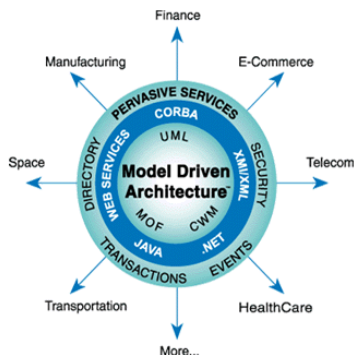
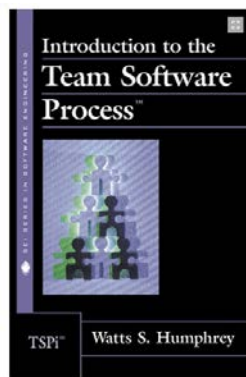
2000

2005

2010

2014

TSP



JUnit

Aparición de JUnit



Pruebas automatizadas



Aparición de GIT



# Conceptos básicos

- Responsabilidad
- Acoplamiento
- Cohesión

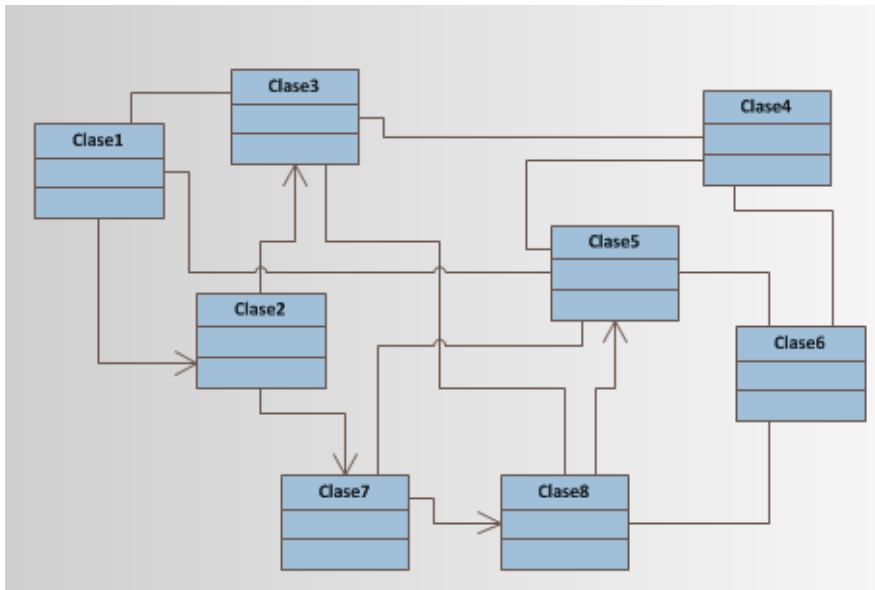
# Responsabilidad

- Obligación, acción o contrato de una clase
- Dos clases de **responsabilidades**
  - Conocer
    - Conocer la información privada del objeto
    - Conocer acerca de los objetos relacionados
    - Conocer acerca de lo que se puede calcular o derivar
  - Hacer
    - Realizar algo él mismo
    - Iniciar acciones en otros objetos
    - Controlar o coordinar actividades en otros objetos

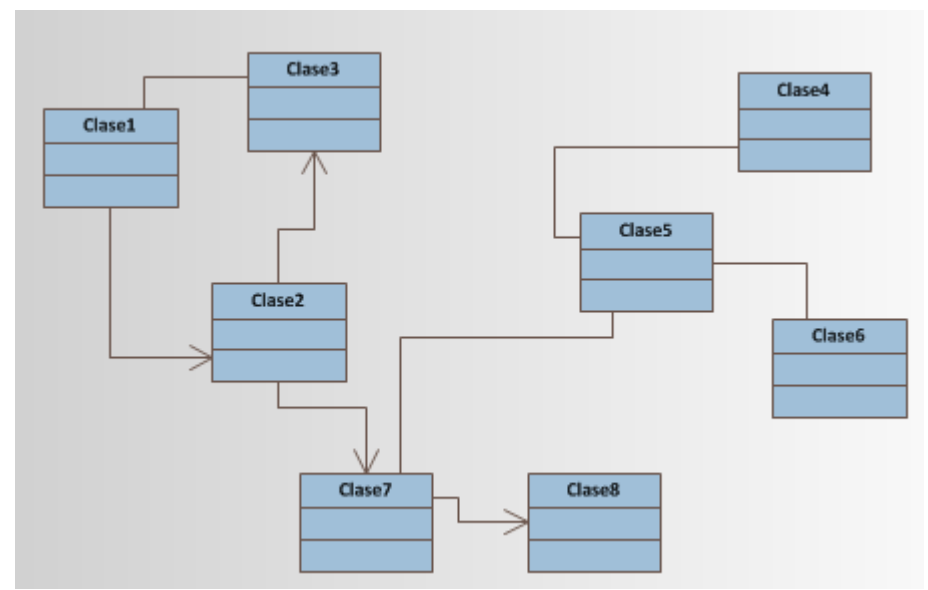
# Acoplamiento

Acoplamiento es la medida de cuánto una clase esta conectada (tiene conocimiento) de otras clases.

# Acoplamiento



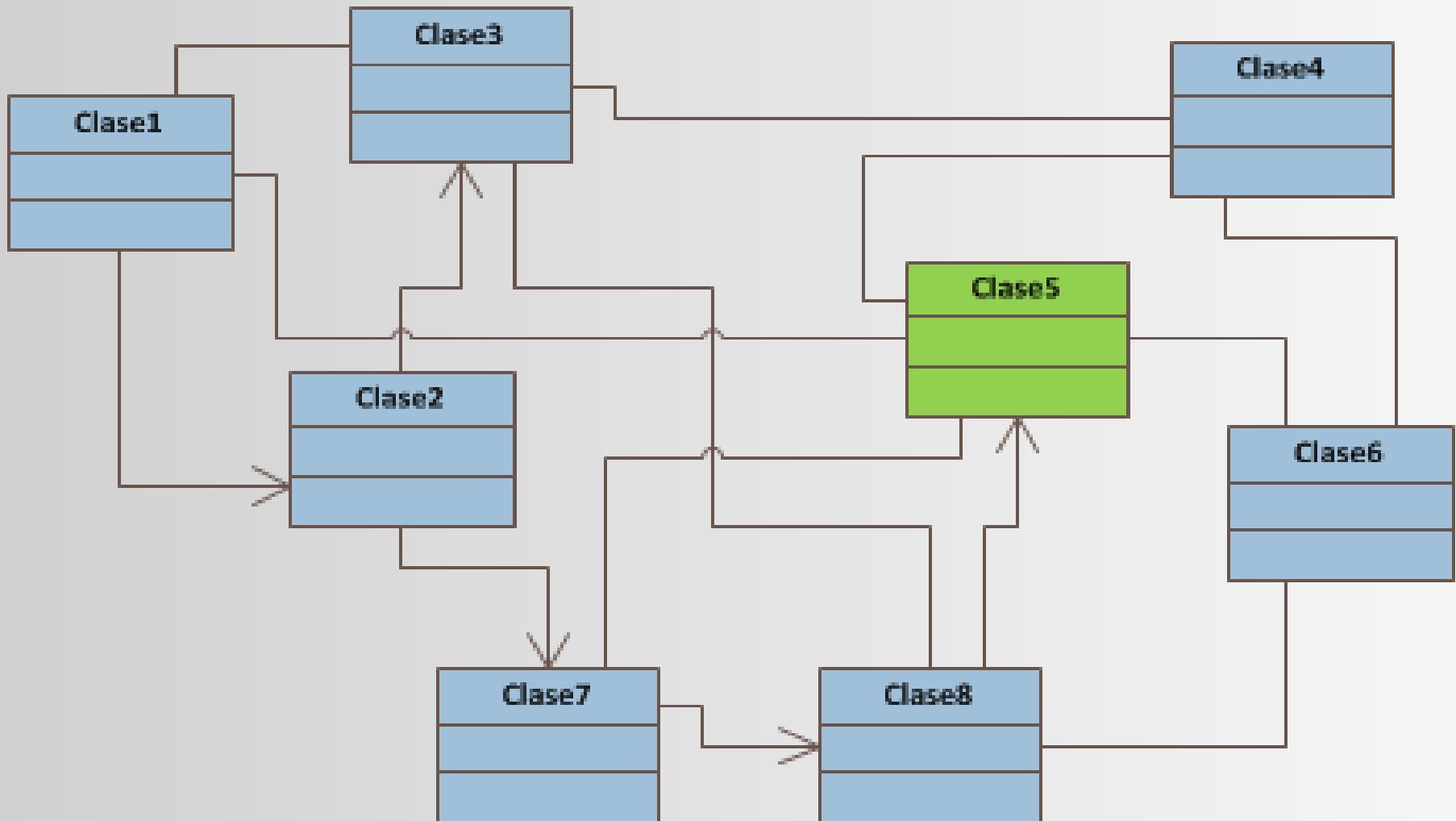
Alto Acoplamiento



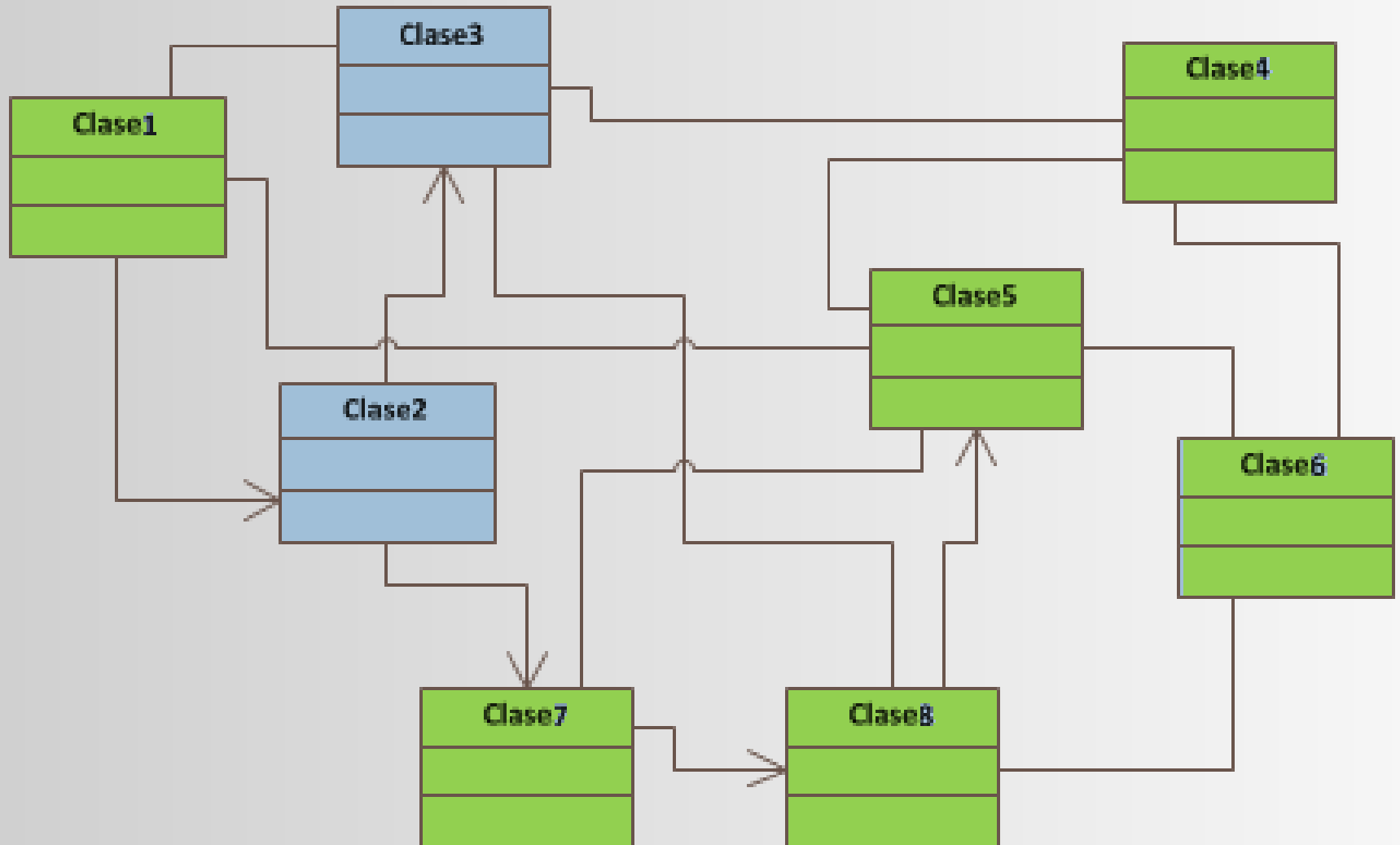
Bajo Acoplamiento



# Acoplamiento



# Acoplamiento



# Cohesión

- Es una medida que indica cuán relacionadas están las responsabilidades de una clase entre sí



# Patrón de Bajo Acoplamiento

- Este patrón nos dice que debemos tratar de mantener el nivel de acoplamiento bajo minimizando el conocimiento que unas clases deben tener de otras.

# Patrón de Bajo Acoplamiento

- Es un patrón evaluativo:
  - un bajo acoplamiento permite que el diseño de clases sea más independiente.
  - Reduce el impacto de los cambios y aumenta la reutilización.

# Patrón de Bajo Acoplamiento

- Quiere decir que cuando tomamos una decisión si esta decisión implica establecer una nueva relación entre las clases tenemos que evaluar si realmente eso es necesario porque al hacerlo vamos a aumentar el nivel de acoplamiento.
- No puede ser considerado aisladamente pero sí es una guía para tomar decisiones.

# Cohesión

- Vamos a explicar el concepto de Cohesión utilizando como ejemplo la navaja de la figura.
- Esta navaja tiene muchas funciones (responsabilidades en nuestros términos).
- Esta sirve como tijeras, cortador, lupa, lima, sierra, abridor de botellas, destornillador, pinzas, etc.

# Cohesión

- En este caso diríamos que esta navaja no es muy “cohesiva” ya que se ocupa de demasiadas funciones distintas en el mismo aparato.
- Esas funciones no están relacionadas entre sí.
- Fíjese que para cada una de ellas hay un elemento distinto que se ha agregado a la navaja.
- Si debo contestar qué hace la navaja?  
La respuesta no es simple, debo contestar utilizando muchos verbos.



# Alta Cohesión

- Este patrón nos dice que debemos tratar de mantener el nivel de cohesión al interior de una clase lo más alto posible.
- Es un patrón evaluativo:
  - entre más alta cohesión más fácil de entender, de cambiar, de reutilizar.

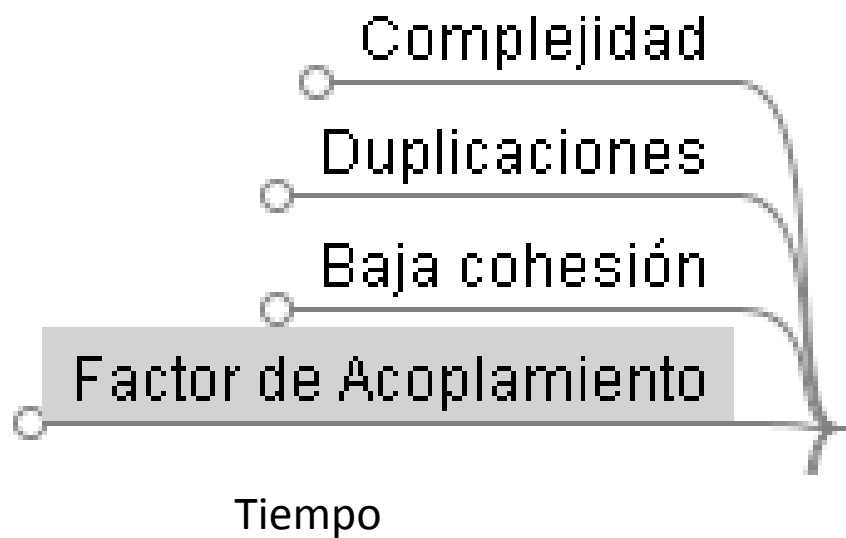


# Alta Cohesión

- Volviendo al ejemplo de la navaja, la de esta figura es mucho más cohesiva porque tiene una sola función: “cortar”.
- No puede ser considerado aisladamente pero sí es una guía para tomar decisiones.



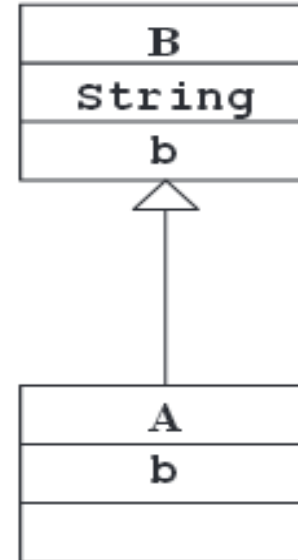
# Medidas



# Factor de acoplamiento (FoC)

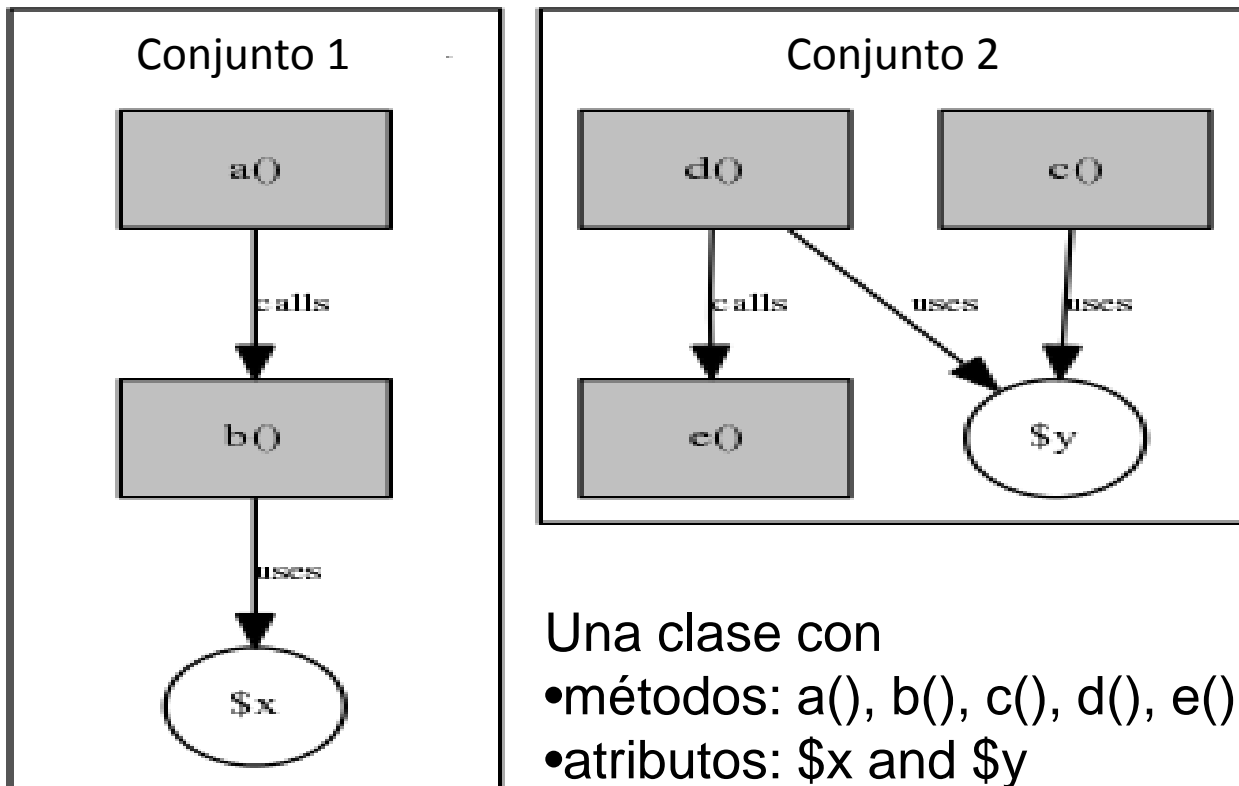
En una aplicación, es una fracción en donde el numerador representa el número de asociaciones que no son de herencia entre el número total de asociaciones

```
public class B  
{ ... }  
  
public class A  
extends B  
{ ... }
```



# Baja cohesión

Dentro de una clase, número de conjuntos de métodos locales que son disyuntos, es decir, que no acceden a los mismos atributos de la clase o llaman a los mismos métodos



# Duplicaciones

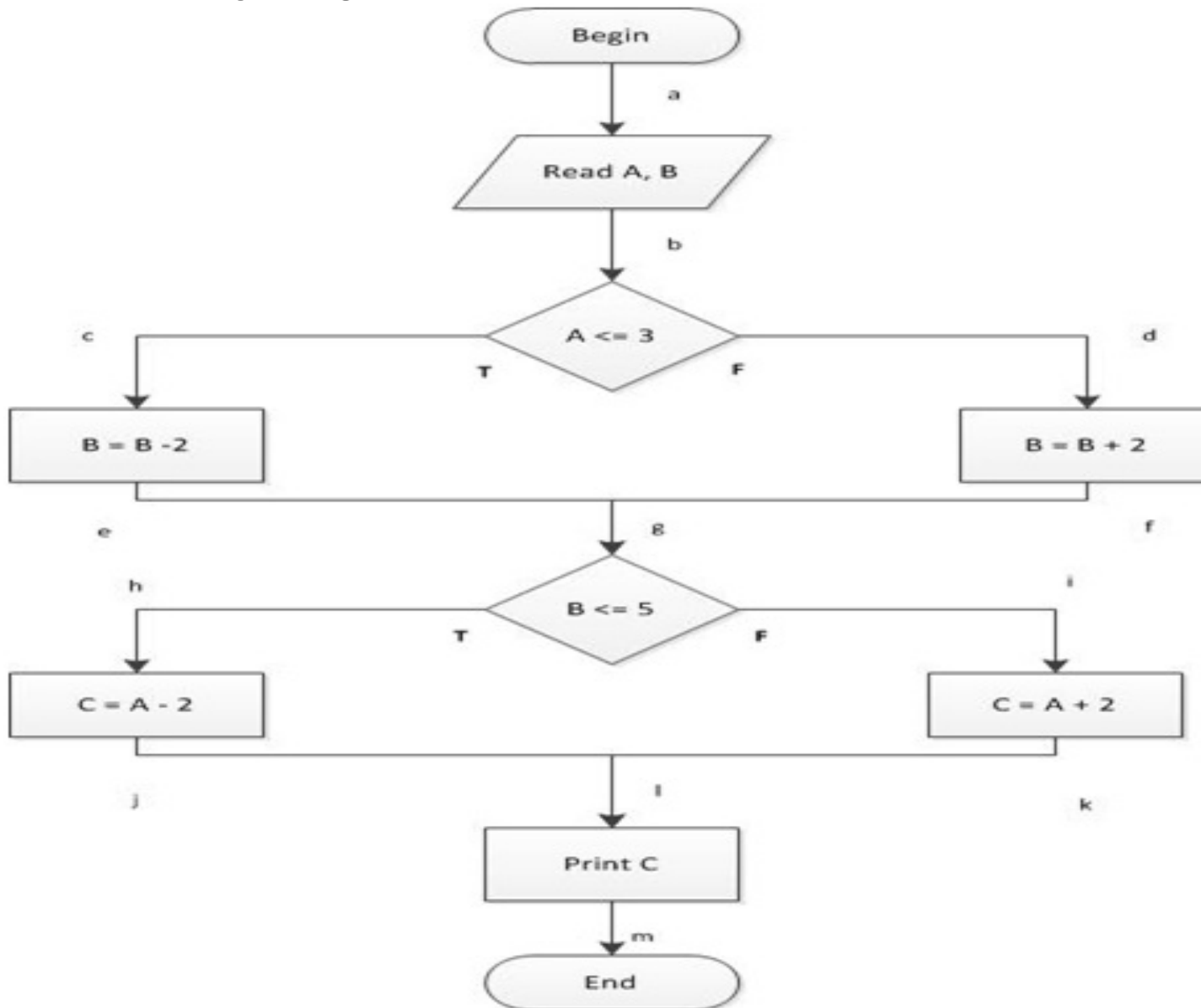
En una aplicación, número de bloques de líneas duplicados

```
0774         this.setButtonResetEnable(true);
0775         break;
0776     case STATUS_INPUT_VALID:
0777         this.setTreeViewerEnable(true);
0778         this.setInputParameterEnable(true);
0779         this.setOutputParameterEnable(true);
0780         this.setButtonExecuteEnable(true);
0781         this.setButtonResetEnable(true);
0782         break;
0783     case STATUS_EXECUTED_FAILURE:
0784     case STATUS_EXECUTED_SUCCESS:
0785         this.setTreeViewerEnable(true);
0786         this.setInputParameterEnable(true);
0787         this.setOutputParameterEnable(true);
0788         this.setButtonExecuteEnable(true);
0789         this.setButtonResetEnable(true);
0790         break;
0791     case STATUS_READY_FOR_USER_INPUT:
```

DB: Method com.avaloq.adt.actioncall.presentationmodel.GenericActionCallModel.setStatus(int) uses the same code for two  
This method uses the same code to implement two clauses of a switch statement. This could be a case of duplicate code, but

```
0796         Assert.fail("Status not defined.");
0797     }
0798
0799     // provide the status to the view
```

# Complejidad ciclomática (CC, McCabe)



En una  
aplicación (o  
Clase)  
complejidad  
para cambiar,  
entender un  
programa

$$CC = e - n + 2$$

n: no. de  
nodos  
e: no. de  
caminos

Valor mínimo:  
1  
Valor  
máximo: 10

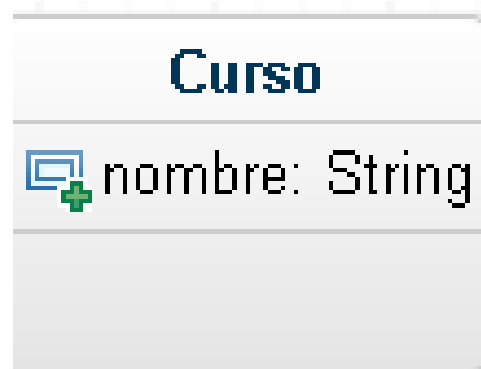
# Tácticas para alta cohesión y bajo acoplamiento

- Aumentar cohesión
  - Abstraer servicios comunes
- Reducir acoplamiento
  - Encapsulamiento
  - Wrappers
  - Usar intermediarios
  - Subir el nivel de abstracción



# Encapsulamiento



- Atributos simples



# Encapsulamiento

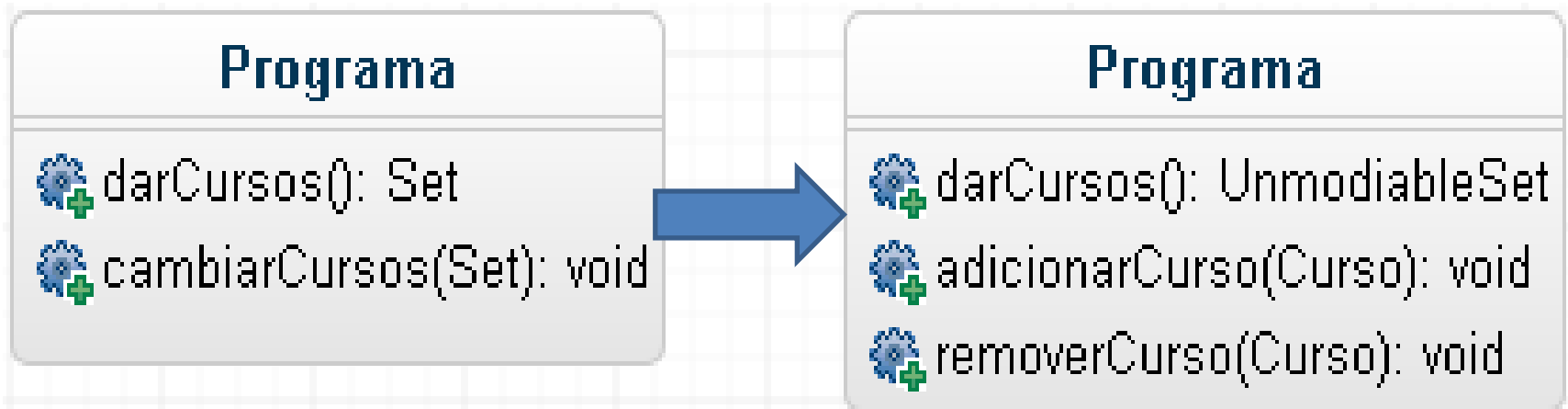
- Atributos multivalor

## Programa

 darCursos(): Set  
 cambiarCursos(Set): void

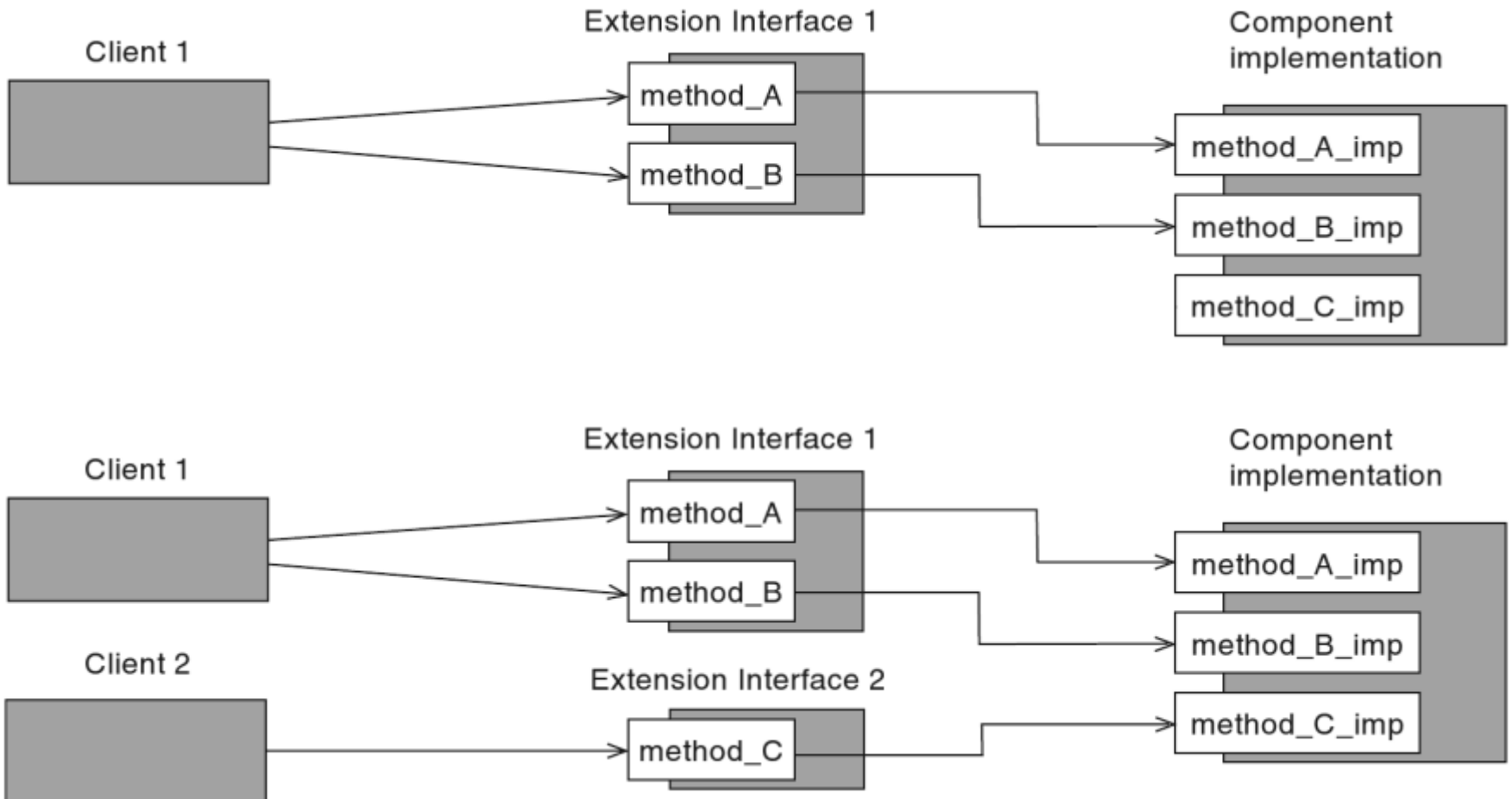
# Encapsulamiento

- Atributos multivalor



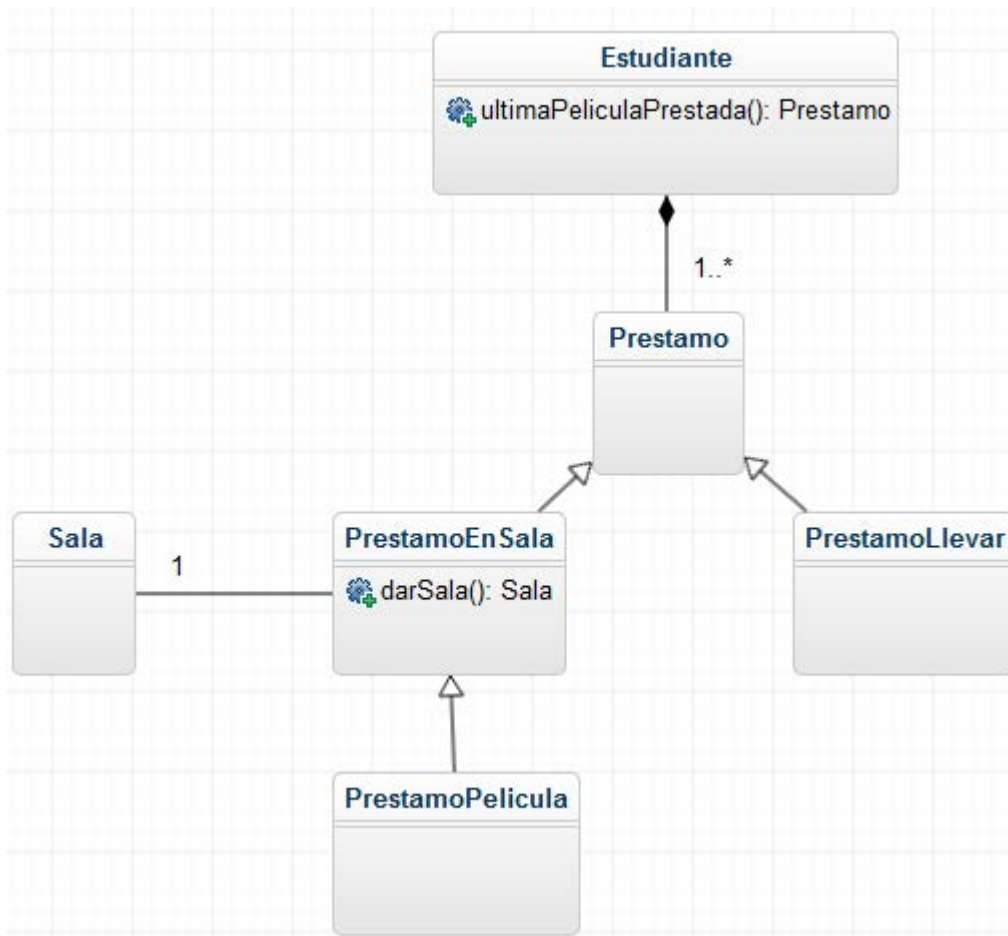
# Encapsulamiento

- Métodos



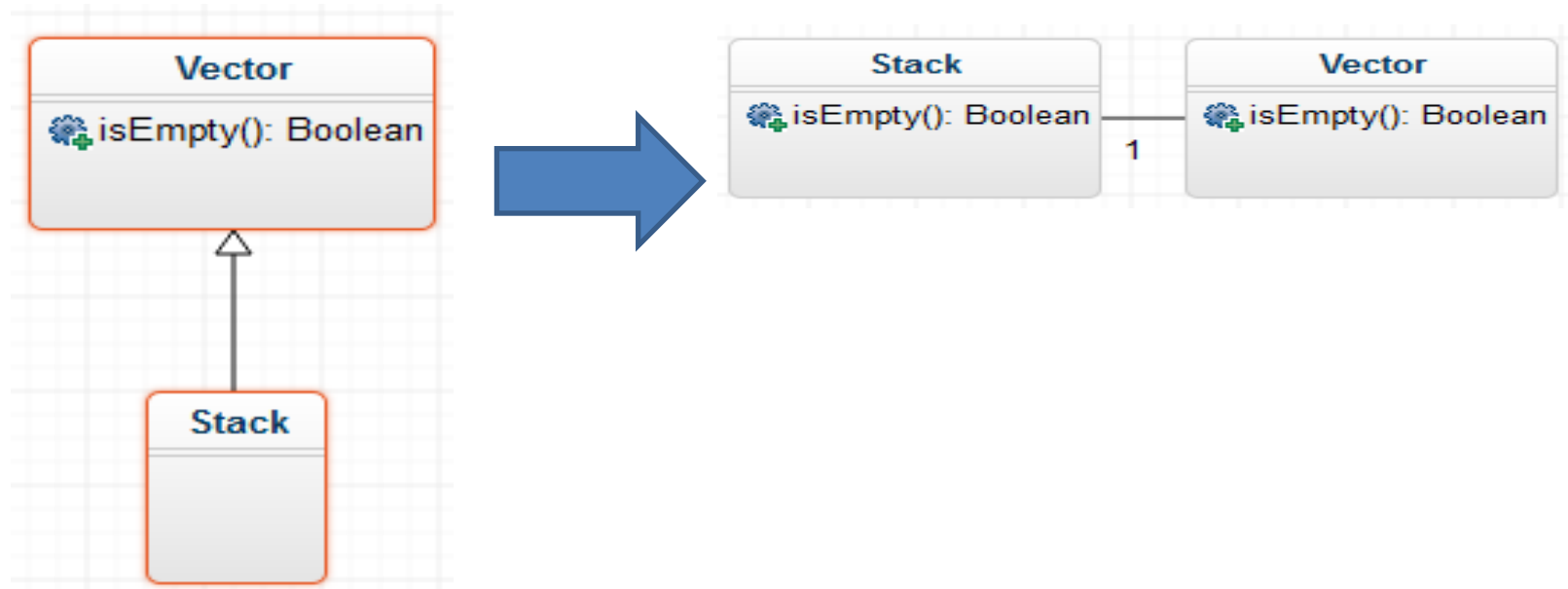
# Encapsulamiento

- Downcasting

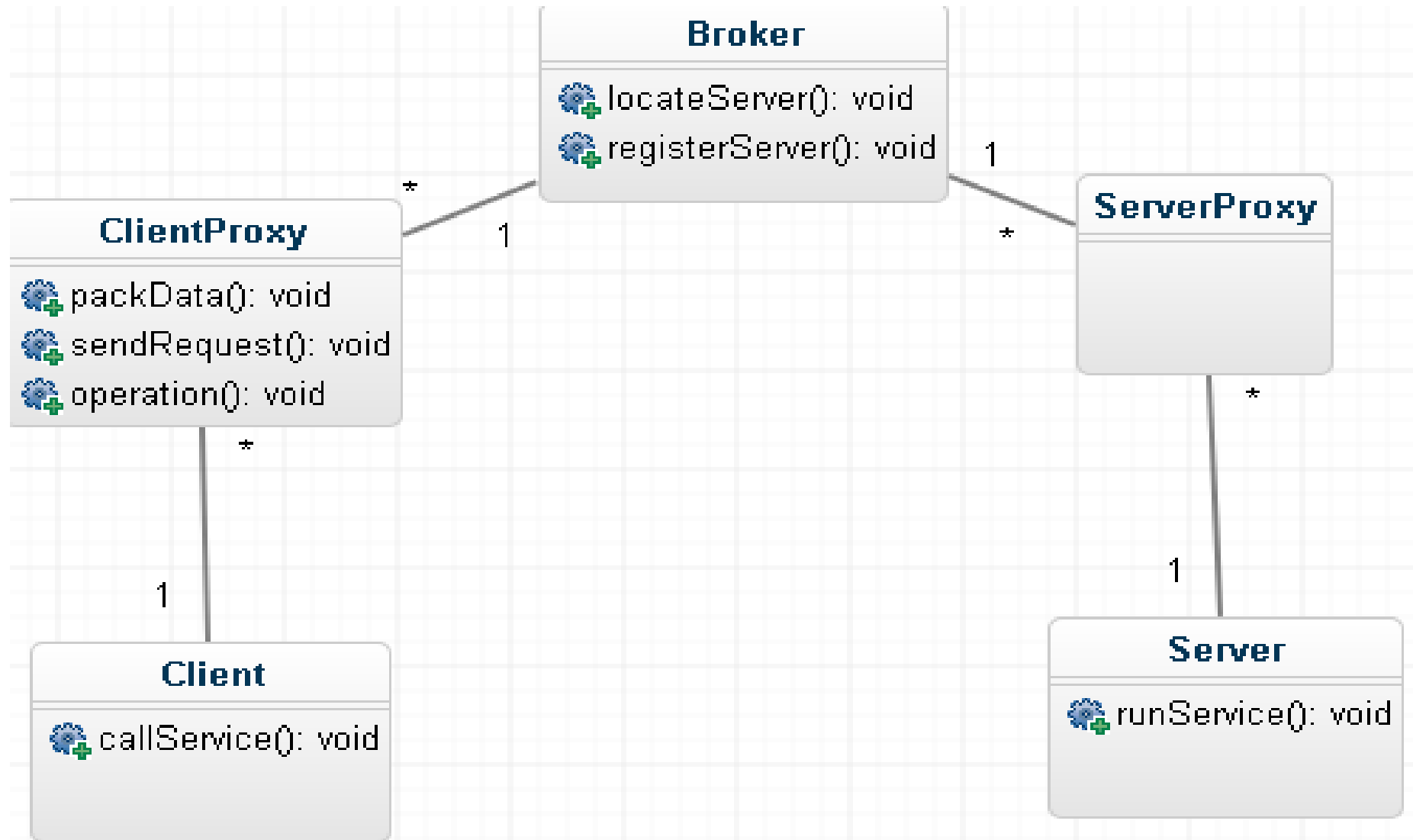


```
Prestamo p =  
e.ultimaPeliculaPrestada();  
  
((PrestamoEnSala)p).darSala();
```

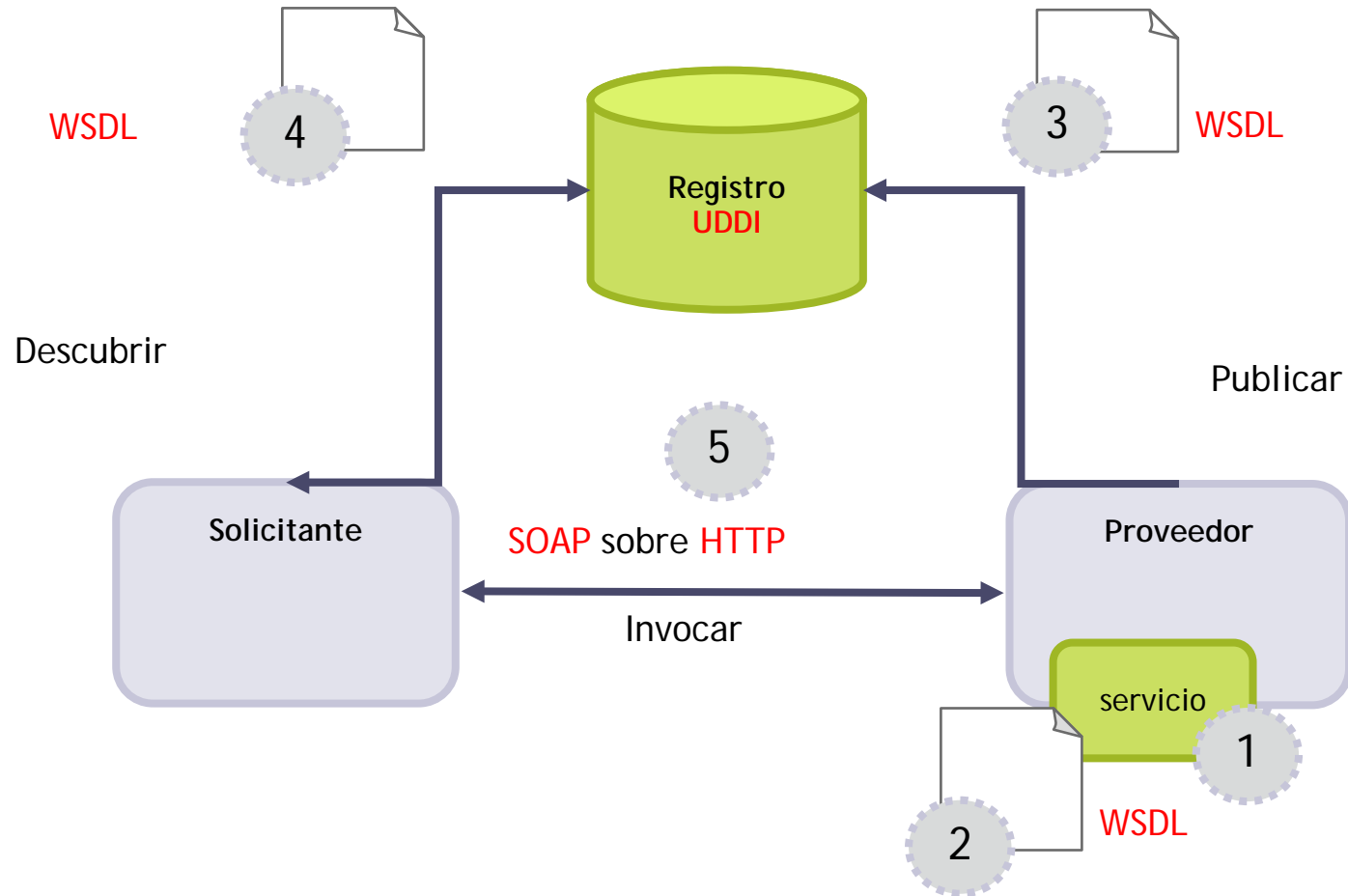
# Wrapper (Delegate)



# Intermediarios

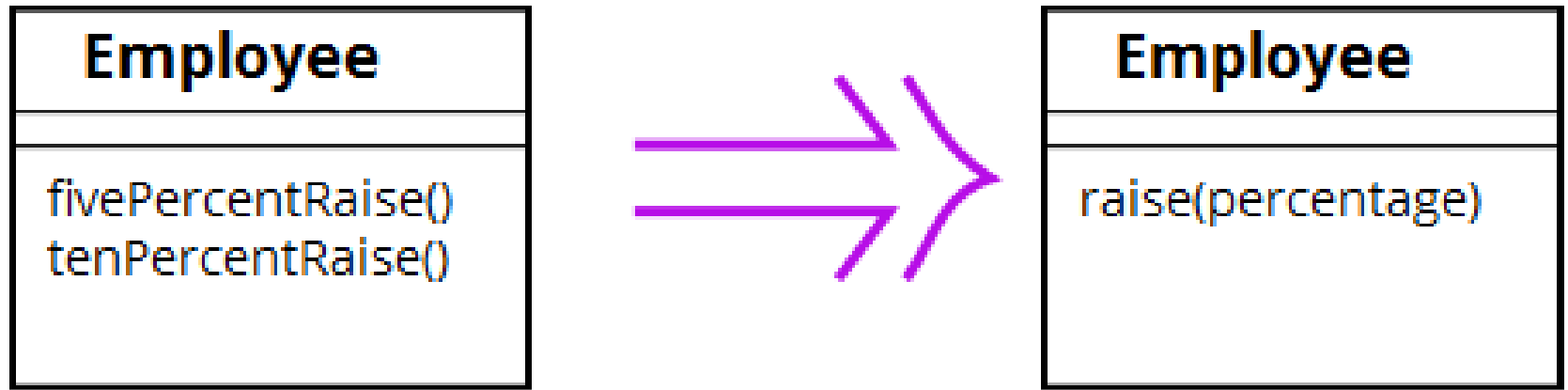


# Intermediarios





# Subir nivel de abstracción



# Ejemplo escenario de calidad de modificabilidad

- *Se necesita una interfaz de usuario más usable que la que existe, de forma que pueda realizar el pago de una forma intuitiva y en poco tiempo*
- **Escenario de Calidad**

Elemento del Escenario	Valores
Fuente de Estimulo	Desarrollador
Estímulo	Reemplazar la interfaz de usuario para la funcionalidad de pagos
Artefacto	Componente de Interfaz de usuario para pagos
Respuesta	El componente es reemplazado sin afectar otros componentes
Medida de la respuesta	El tiempo de cambio debe ser menor a <b>un día.</b>

# Ejemplo escenario de calidad de modificabilidad

- *Un dispositivo adicional se agrega al sistema. Por ejemplo, un nuevo dispositivo que envía información muy precisa a un servidor sobre la ubicación de carros de transporte. Se necesita que el sistema existente pueda recibir la información del nuevo dispositivo. Ya existen otros dispositivos que envían la posición pero usando otros protocolos de comunicación*
- **Escenario de Calidad**

Elemento del Escenario	Valores
Fuente de Estimulo	El desarrollador
Estímulo	Se adiciona un nuevo componente de servicios al sistema
Artefacto	Componente de servicios, Base de datos e interfaz de usuario
Respuesta	El componente es agregado sin alterar el comportamiento y la calidad del código del sistema en general
Medida de la respuesta	El porcentaje de duplicaciones del sistema general debe ser igual a <b>0%</b>

# Ejemplo de escenario de calidad de modificabilidad

- *Una nueva funcionalidad es agregada al componente de lógica de negocio*
- **Escenario de Calidad**

Elemento del Escenario	Valores
Fuente de Estimulo	El desarrollador
Estímulo	Se adiciona un nuevo método
Artefacto	Componente de lógica de negocio
Respuesta	El método es agregado sin alterar el comportamiento y la calidad del código del sistema en general
Medida de la respuesta	La complejidad ciclo somática del método es inferior a <b>10</b> puntos de complejidad.