UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

# Diseño en UML

## Modelando Estructura del Sistema

# Guías de Diseño OO

## Guías para Diagramas Flexibles

- Debe ser posible extender y/o modificar el software a través del tiempo.
- El modelo de diseño debe ser flexible para permitir extensión y reuso.
- Existen guías de diseño flexible:

✔ Cohesión y acoplamiento.
✔ Generalización y especialización.
✔ Especialización vs. Agregación.
✔ Agregación
✔ Uso de Patrones de Diseño.

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Cohesión

- Es una medida de la diversidad de características de una entidad.
- A menor diversidad de las características de una entidad, ésta es más cohesiva.
- Una entidad cohesiva representa un concepto simple.
- Para garantizar flexibilidad, cada entidad del diseño debe ser lo más cohesiva posible >>>

✔ Debe representar una abstracción única o
✔ Debe tener una responsabilidad general única.

# Guías de Diseño OO

## Problemas de Poca Cohesión

- La clase es más difícil de entender.
- Se está asumiendo que dos abstracciones representadas por la clase están siempre vinculadas en una relación uno-a-uno.
- Se puede tener que especializar la clase en diferentes dimensiones basadas en las diferentes abstracciones.

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

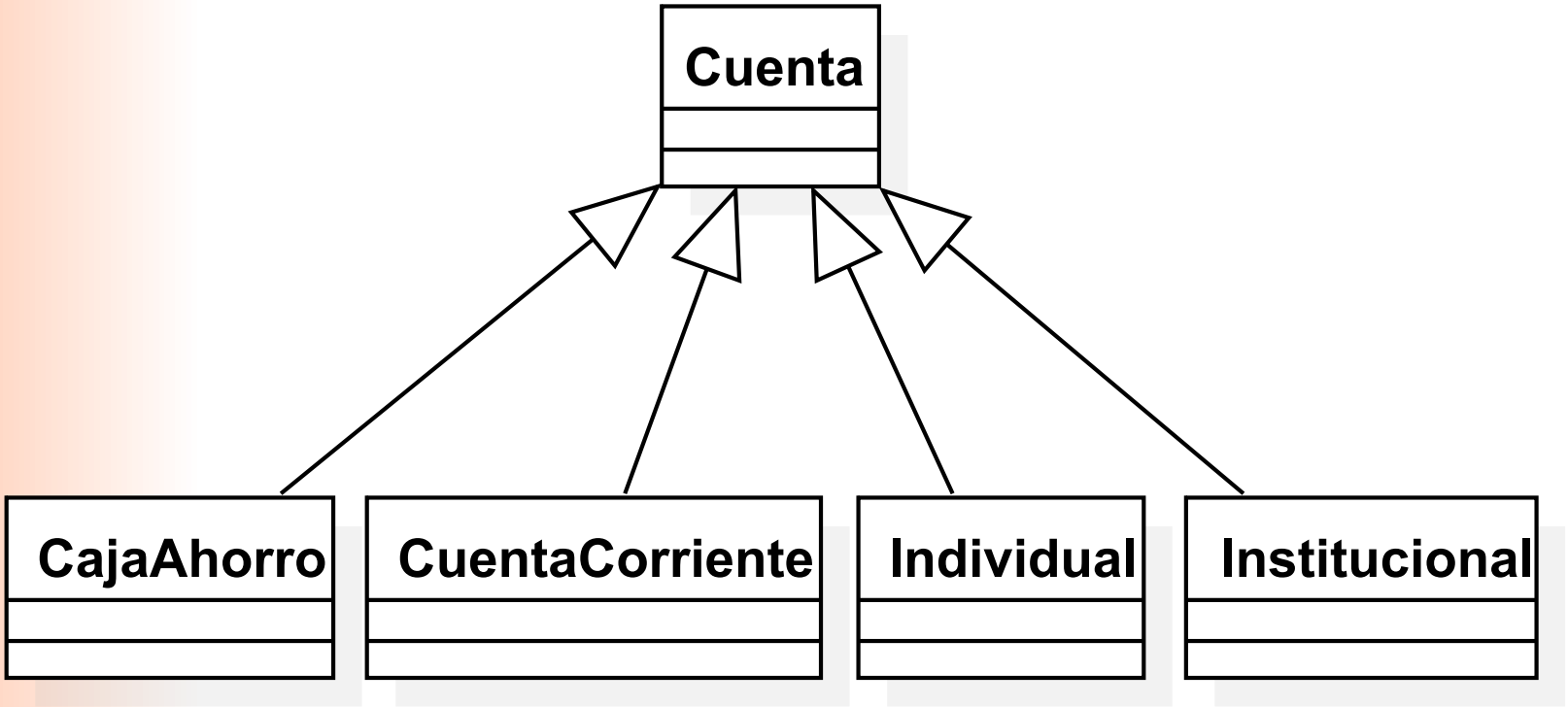
## Ejemplo

- Una clase Cuenta que incluye información del Cliente como atributo, tal como Nro Id y nombre.
- La clase no es cohesiva porque incluye las abstracciones Cuenta y Cliente.
- En ese caso se asume que la relación entre abstracciones es uno-a-uno.
- Si se permite al Cliente abrir más de una Cuenta, se duplicará el nombre de Cliente para cada cuenta.
- La cuenta no puede tener múltiples dueños, ya que en cada instancia de Cuenta se almacena información de un sólo cliente.
- La presencia de dos abstracciones en la clase Cuenta también puede llevar a la posibilidad de especializar la clase de dos formas diferentes

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Especialización de Cuenta

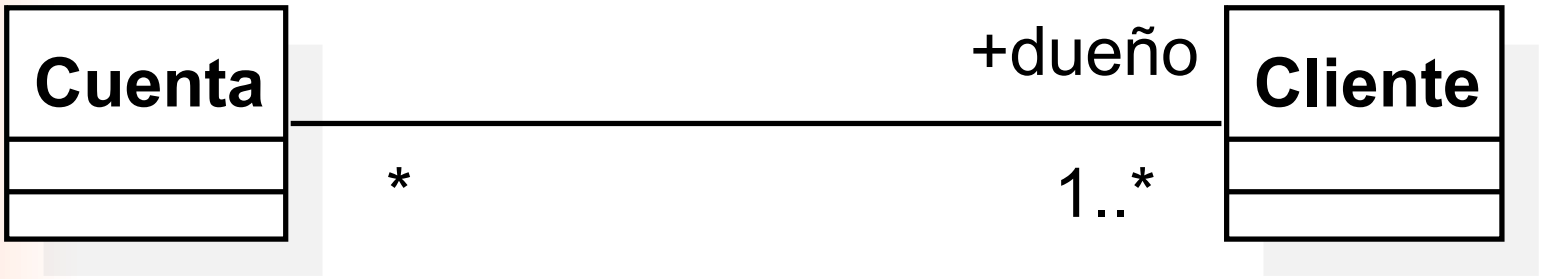



Este tipo de Jerarquías puede llevar a jerarquías complicadas de herencia Múltiple, no soportada por todos los lenguajes OO

# Guías de Diseño OO

## Diseño cohesivo del Ejemplo

La separación de las dos abstracciones lleva a un Diseño más cohesivo.

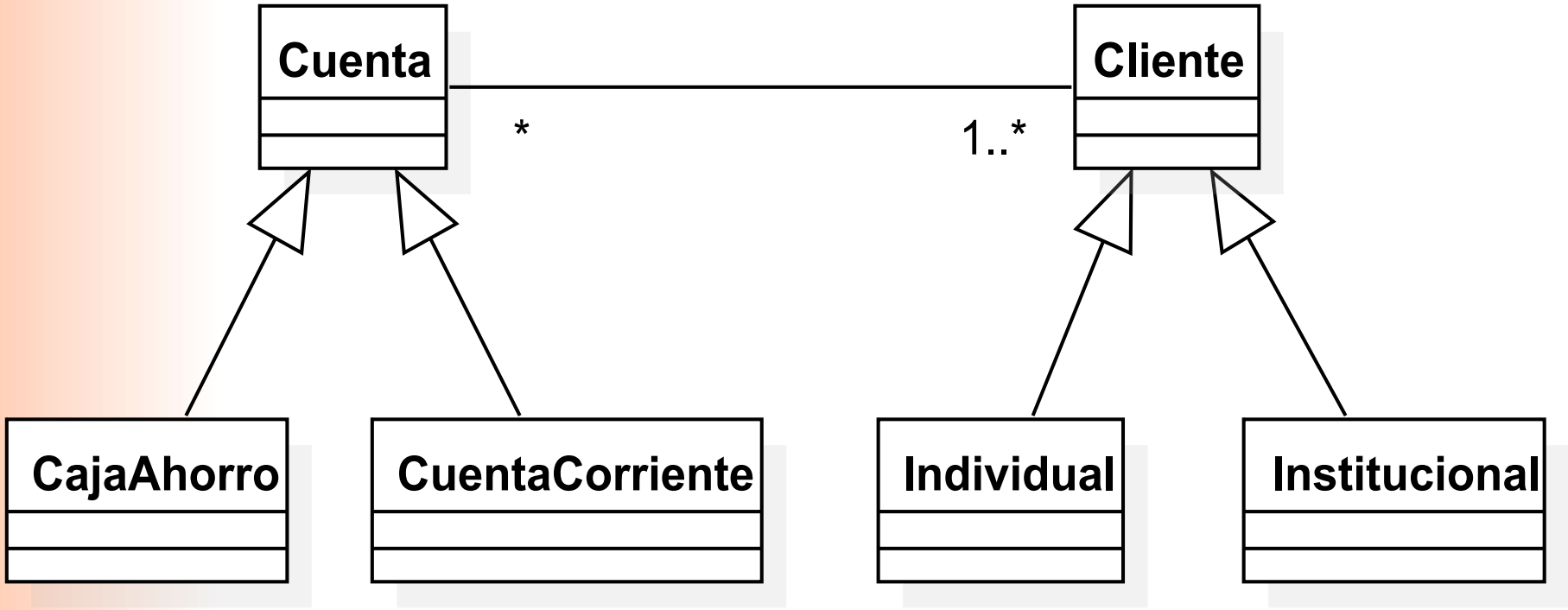



Mediante relaciones de cardinalidad se permite al cliente vincularse con más de una cuenta y a una cuenta tener más de un cliente

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

# Diseño cohesivo del Ejemplo



UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Acoplamiento

- Sucede cuando un elemento del diseño depende de otro en alguna forma. A mayor interdependencia, mayor es el nivel de acoplamiento.
- Cuando sea posible, debe evitarse para lograr un diseño más modificable y extensible.
- Si un objeto conoce a otro y la interfaz pública del segundo cambia, el primero puede verse afectado.
- Existen muchas formas de acoplamiento en Diseño OO, pero las más comunes son:

✔ Acoplamiento de Identidad.
✔ Acoplamiento de Subclase.
✔ Acoplamiento de Herencia

UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Acoplamiento de Identidad

- Es una medida del nivel de conectividad de un diseño.
- Si un objeto tiene una referencia (o puntero) a otro, dicho objeto “conoce” la identidad del otro y exhibe acoplamiento de identidad.
- Se puede reducir:

✔ Eliminando asociaciones innecesarias en el Diagrama de Clases.

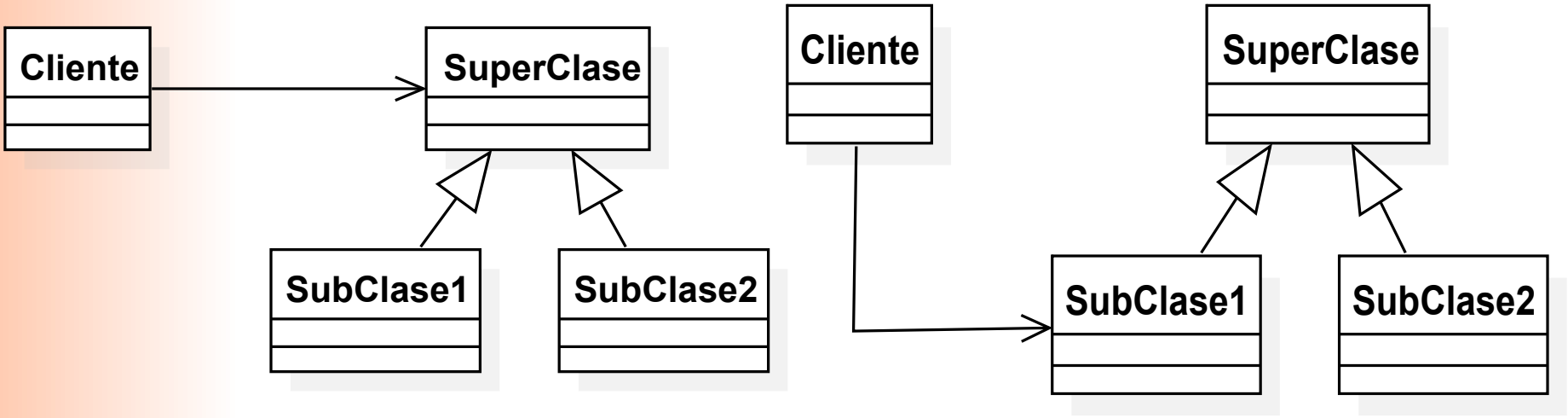
✔ Implementando asociaciones en una dirección.

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Acoplamiento de Subclase

- Ocurre cuando un objeto cliente refiere a un objeto subclase a través de una referencia a la subclase en lugar de usar una referencia más general a la superclase.
- El Cliente debe referir la clase más general posible, desacoplando al cliente de la existencia de subclases específicas.




**Preferido**

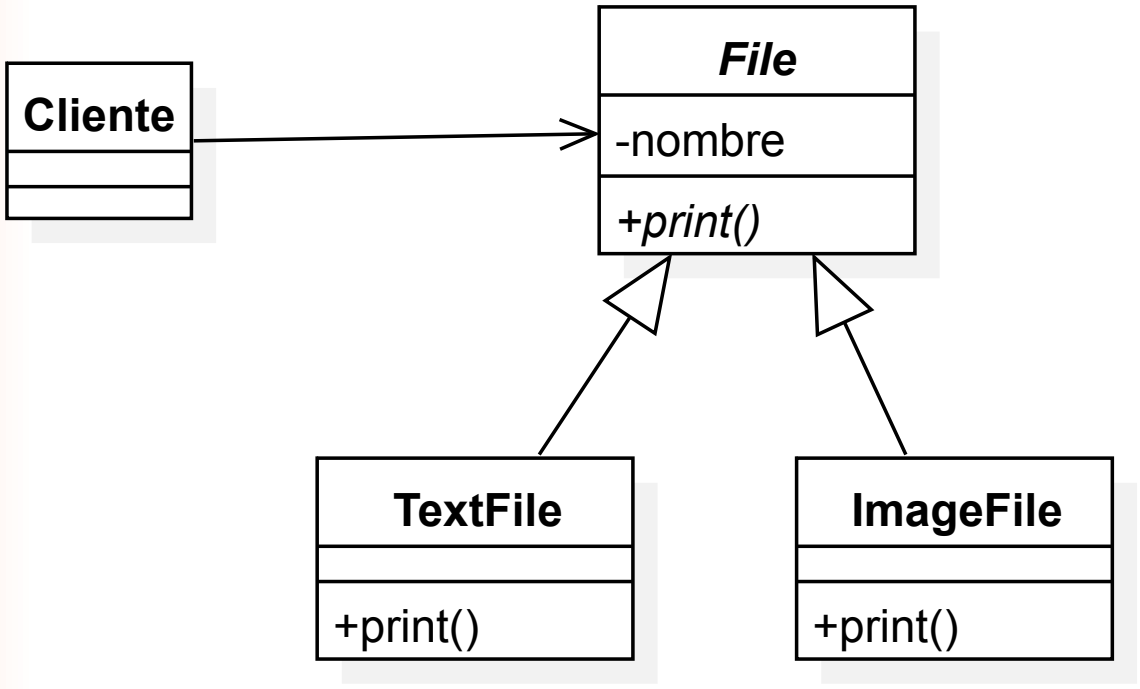
**Menos Preferido**

# Guías de Diseño OO

# Ejemplo:

- En el siguiente código Java, la clase Cliente sólo conoce a la clase File:

```
class Client {
   public void handleFile (File f) {
      f.print(); //llamada polimórfica a print()
   }
}
```



UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Ejemplo Clase File

- En el ejemplo, File en realidad es clase abstracta.
- El mensaje handleFile() es pasado a una referencia de instancia de la subclase (una referencia a TextFile o ImageFile)
- En el ejemplo se evita el acoplamiento de subclases de File, dado que el Cliente es ignorante de esas subclases.
- Como resultado, se pueden agregar nuevas clases de archivos a la jerarquía, sin alterar la clase cliente excepto en la porción que corresponde a la creación de instancias.

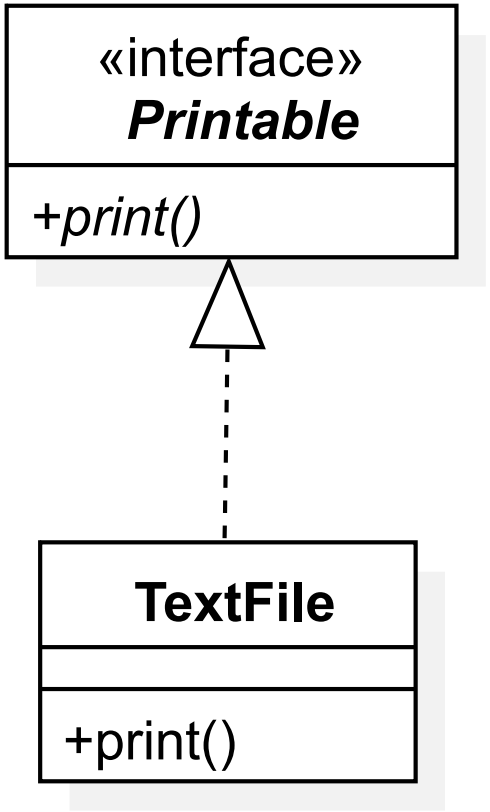
# Guías de Diseño OO

## Uso de Interfaces

- Una extensión de este concepto es el uso de *interfaces* (Java) >> No tiene estado o comportamiento, sólo métodos abstractos.
- UML provee un estereotipo <<interface>>.
- Otras clases implementan la interface. UML provee una *flecha realización* para mostrar esta relación.
- Esta flecha denota el concepto de realización de clase >> implementación de un tipo abstracto.

# Guías de Diseño OO

## Uso de Interfaces



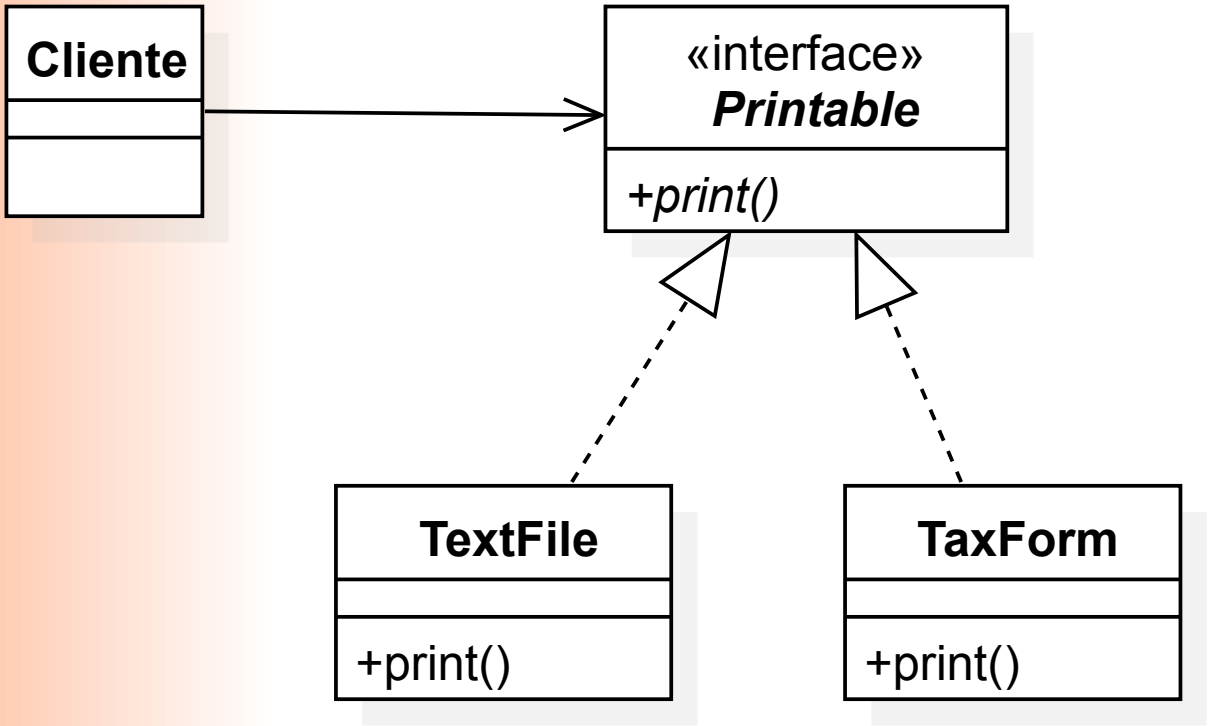

Usando esta interfaz, un cliente queda acoplado a un tipo aún más general que la clase File

UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

# Uso de Interfaces

Una instancia de cualquier clase que implemente esta interfaz puede pasarse al método handleFile() , tal como un objeto TaxForm




Las interfaces definen roles que juegan las clases

Las clases definen abstracciones

UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

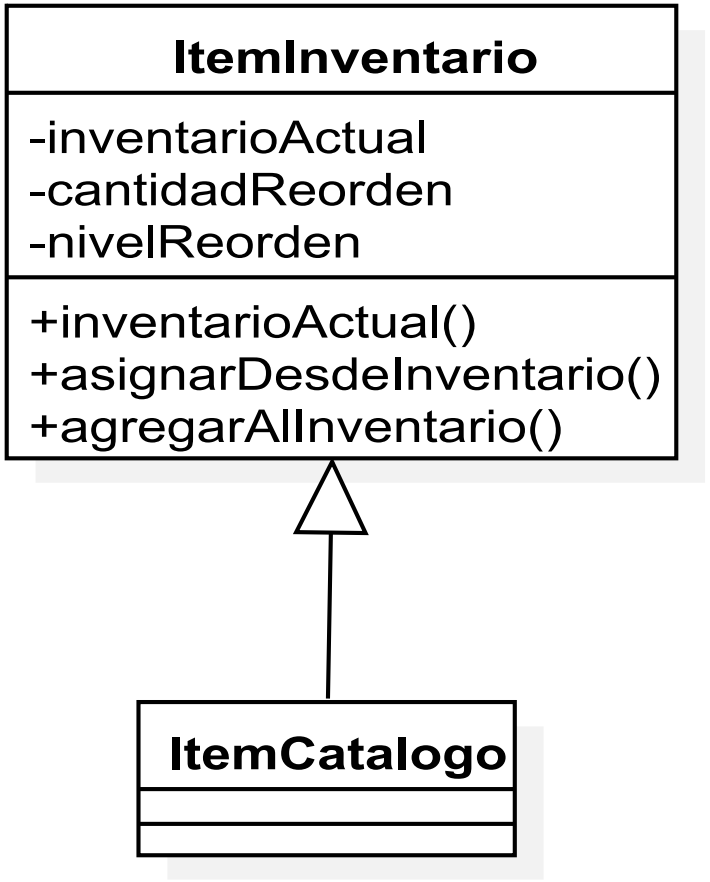
# Guías de Diseño OO

## Acoplamiento de Herencia

- Una subclase está acoplada a su superclase por acoplamiento de herencia.
- La especialización es una relación entre clases (no instancias) >> ocurre en tiempo de compilación.
- Lo que una subclase hereda en tiempo de compilación no puede descartarse en tiempo de ejecución.
- La diferencia con la agregación es que en esta última, un objeto puede agregar y disponer de sus partes durante la ejecución del programa.

# Guías de Diseño OO

## Ejemplo




ItemCatalogo hereda sus propiedades de ItemInventario
Una vez creado de esta forma, durante la ejecución siempre será un item de este tipo.

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Problema de usar herencia

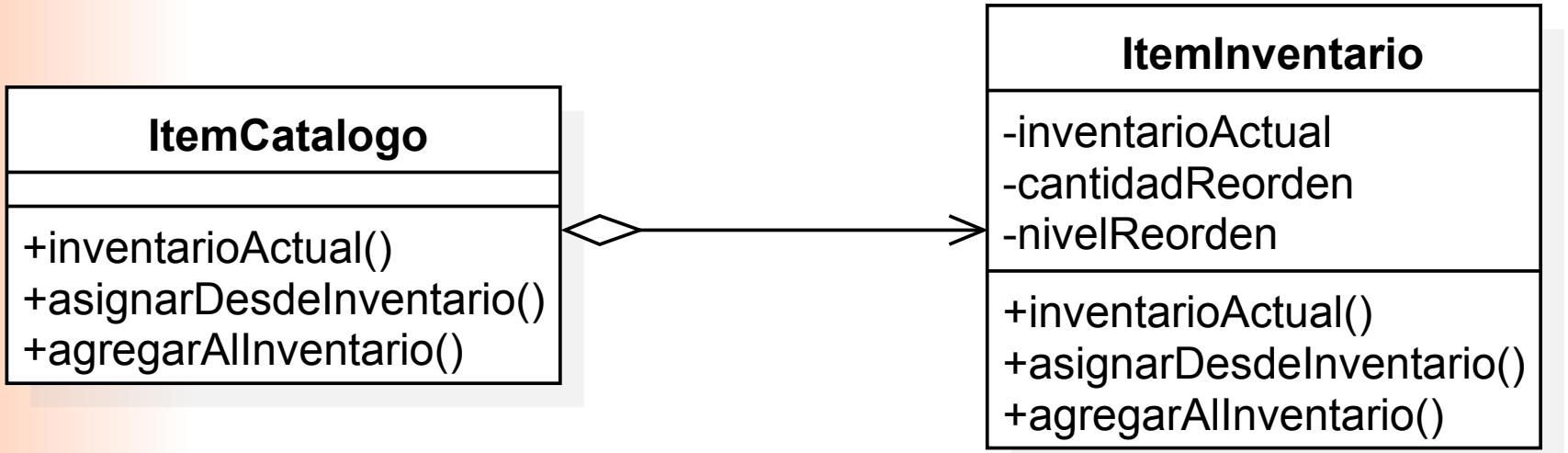
Si una Compañía determina que algunos items del Catálogo no deben retenerse en el Inventario sino obtenidos de proveedores sólo cuando sean ordenados, las instancias seguirán teniendo las propiedades heredadas de la clase Inventario

# Guías de Diseño OO

**Usando agregación**

ItemCatalogo tiene una instancia ItemInventario como una parte opcional. Algunos items tendrán esa parte y otros no.

Más aún un objeto ItemCatalogo puede agregar y remover esa parte en tiempo de ejecución




Una instancia de ItemCatalogo delega cualquier requerimiento de Inventario a su parte ItemInventario

UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Generalización y Especialización

- Una superclase define una generalización de dos o más subclases. Se introduce cuando:

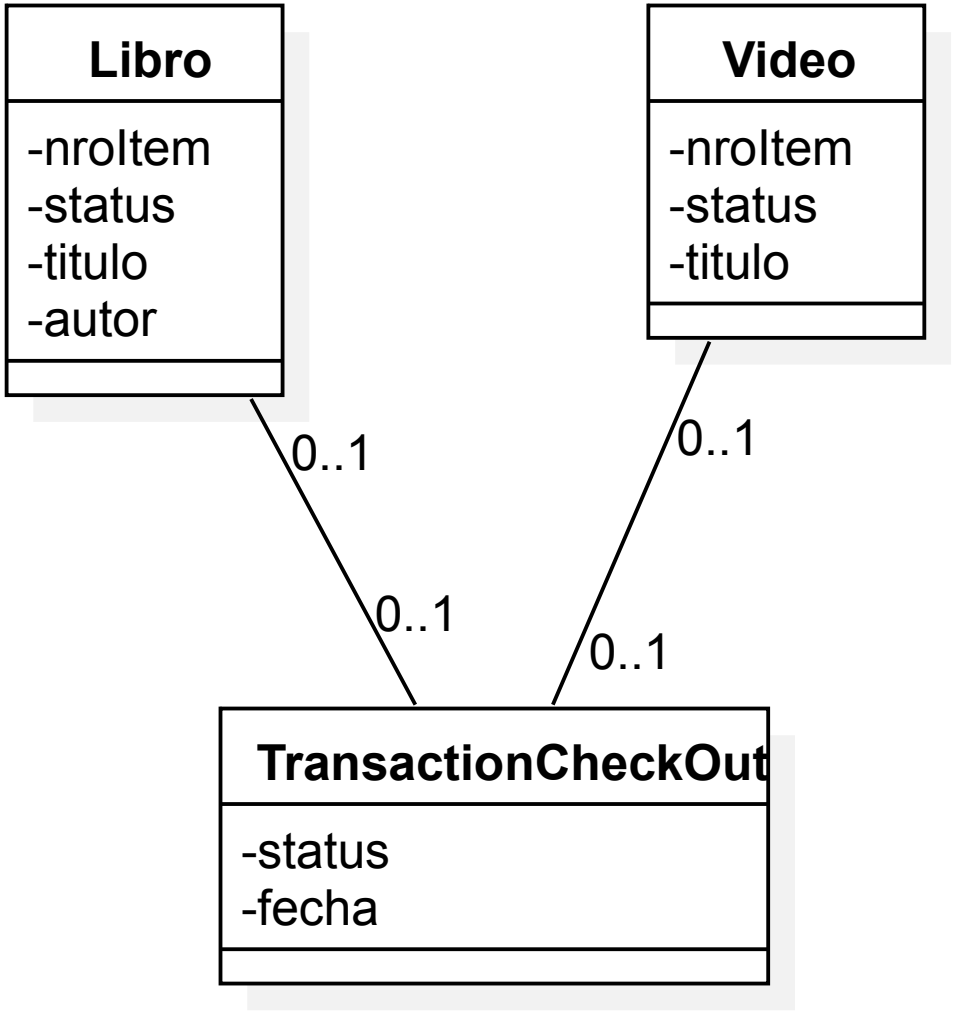
✔ Dos o más clases tienen una implementación común.

✔Dos o más clases comparten una misma interfaz

- La implementación o interfaz común se coloca en una superclase y se comparte a través de la herencia.
- En el ejemplo Libro y Video tienen atributos y asociaciones comunes. Obviamente, se evita implementar esas propiedades en ambas clases.
- Se introduce una superclase Prestable que provee una implementación única.

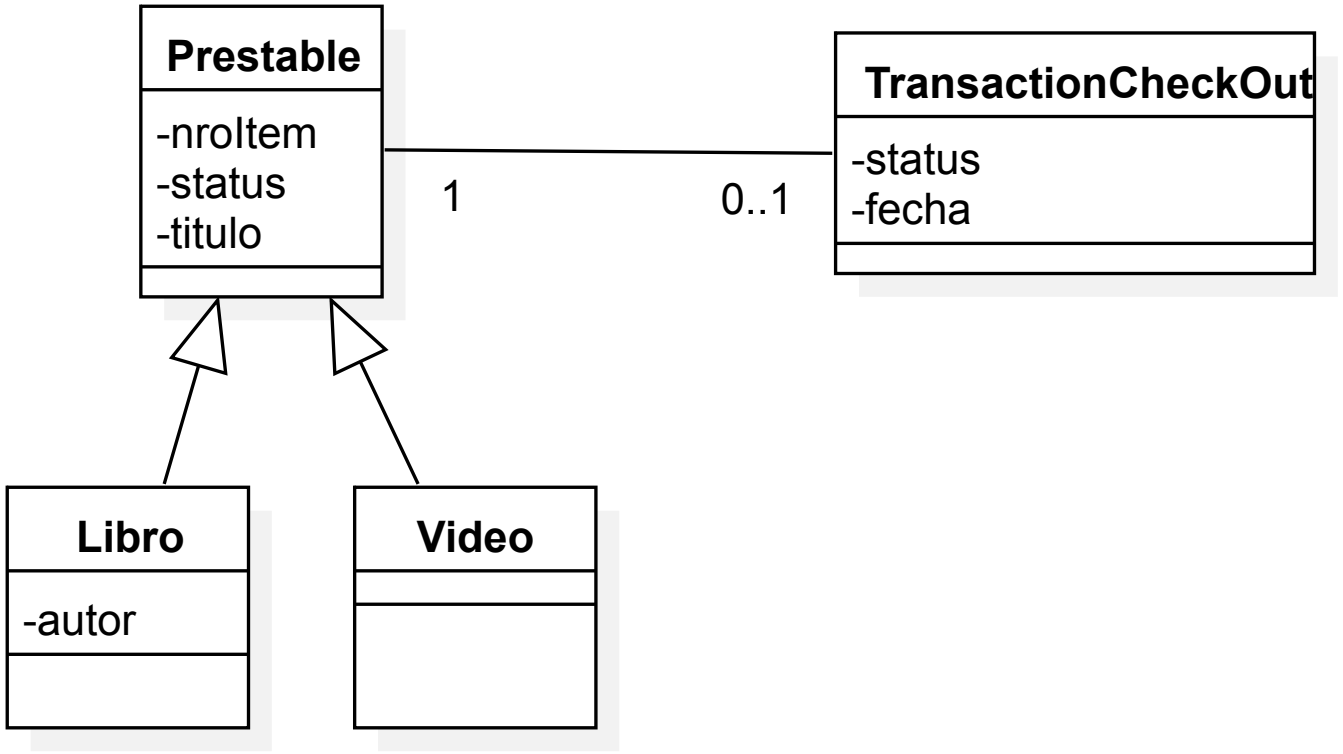
# Guías de Diseño OO

## Ejemplo



# Guías de Diseño OO

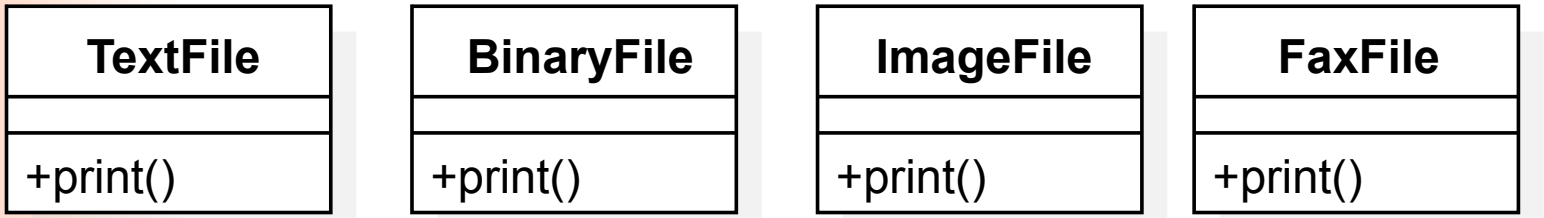
## Ejemplo



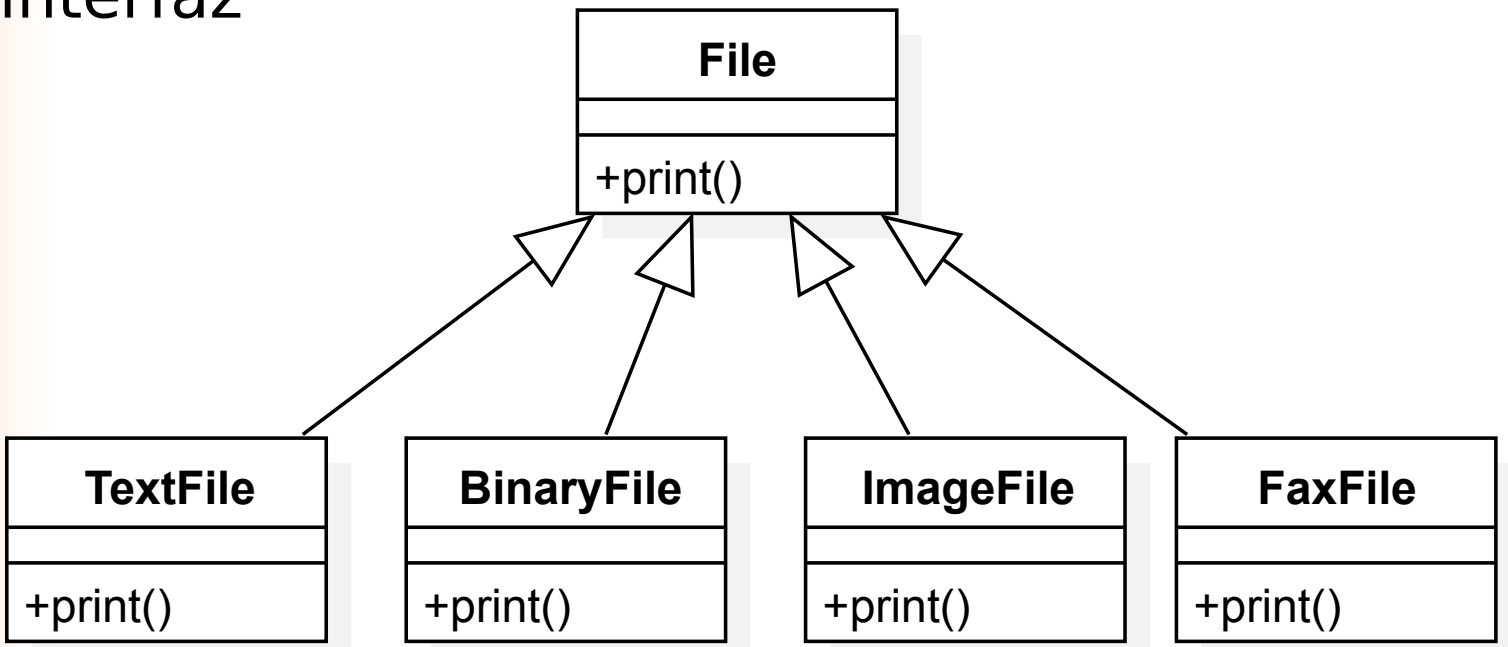

TransactionCheckOut tiene ahora una referencia a un Prestable genérico en lugar de a distintos tipos. Esto permite a las instancias de Transaccion manejar la adición futura de nuevos tipos de prestables.

# Guías de Diseño OO

## Generalización de interface




Se puede introducir una superclase para definir una interfaz común; un cliente puede usar polimórficamente esa interfaz



UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Especialización de Clases

- Formas en que una subclase especializa su superclase:

✔ Agrega información de estado en la forma de atributo.

✔ Agrega información de estado en la forma de asociación.

✔ Agrega comportamiento en forma de método (o implementando un método abstracto de la superclase)

✔ Reemplaza comportamiento reescribiendo un método de la superclase.

- Para utilizar subclases, se debe asegurar que cada una de ellas agrega comportamiento diferente

UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Especialización de Clases

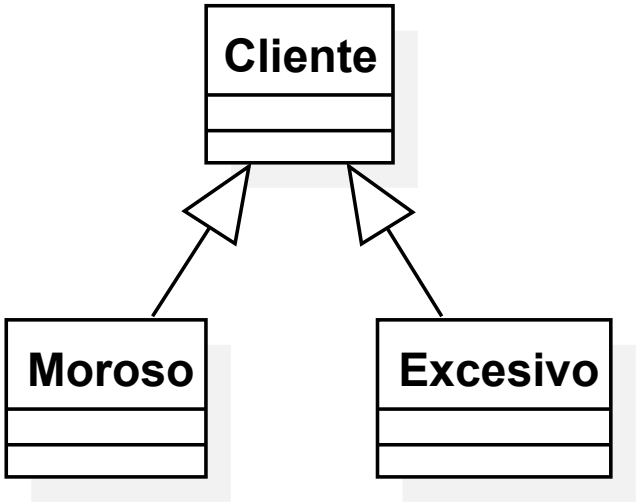
- Suponer una clase Cuenta en la cual se representen todas las formas en que un Cliente puede pagar una orden.
- Esta clase debe considerar características de la tarjeta de crédito, balance de cuenta (para asegurar fondos en el pago en efectivo), métodos diferentes para cada forma de pago, etc.
- Resulta ser una clase sin cohesión.
- Como alternativa, pueden introducirse tres subclases de Cuenta, donde cada una especializa a la clase Cuenta:

✔ CuentaTarjetaCredito
✔ CuentaCash
✔ CuentaFactura

# Guías de Diseño OO

## Distinción basada en estado

- Se debe evitar la especialización de clases basándose en los diferentes estados de una instancia




Cada subclase representa un cliente en un estado particular

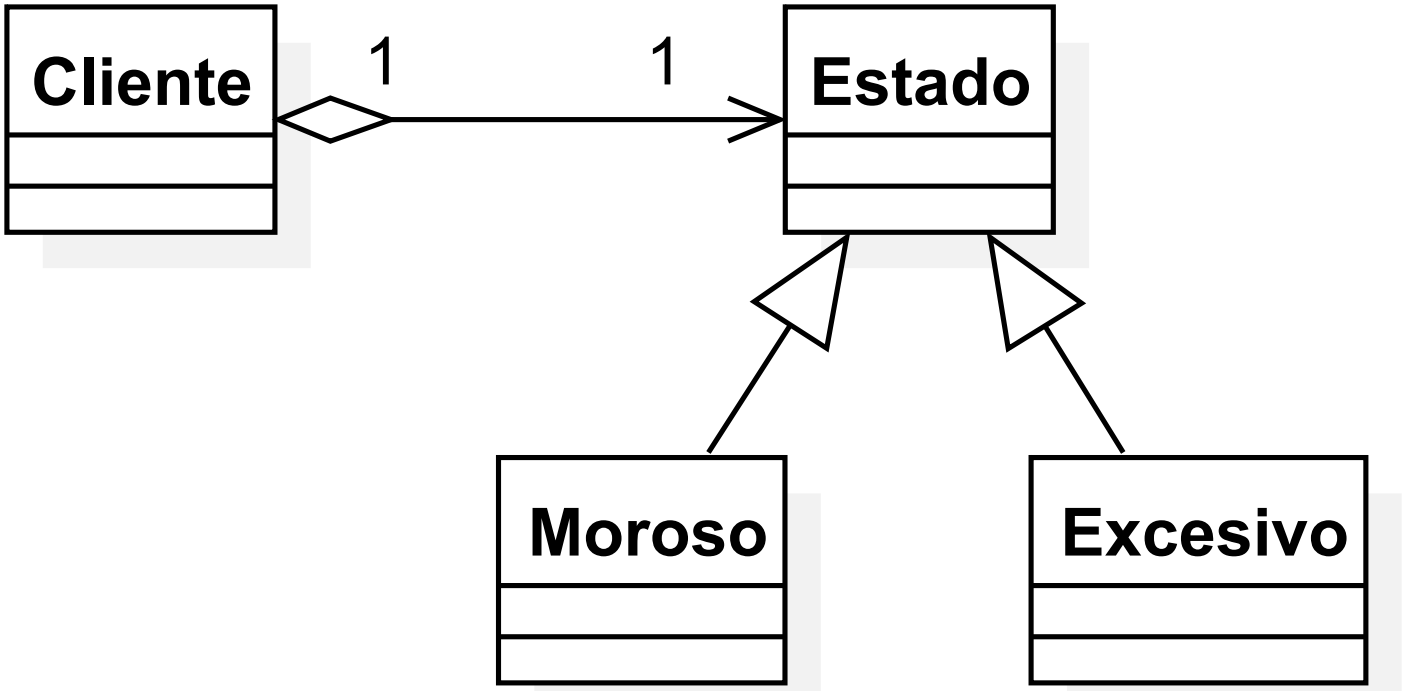
Si un Cliente cambia de un estado a otro, se debe reemplazar una instancia por otra

Una forma de resolverlo es reemplazar especialización por agregación

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Distinción basada en estado




Cuando el Cliente se convierte en moroso se agrega la parte opcional de estado (Ver State pattern)

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Evitar superclases concretas

- Se logra un diseño más flexible utilizando clases abstractas o interfaces que superclases concretas.
- De esta forma existe un lugar donde residen las propiedades únicas de cada subclase y una interfaz para las propiedades comunes.

UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Resumen Guías de Diseño

- Mantener clases cohesivas. Una clase debe representar una abstracción única.
- Reducir acoplamiento de identidad, reduciendo las asociaciones en el Diagrama de clases e implementando asociaciones en una sola dirección cuando sea posible.
- Reducir acoplamiento de subclases poniendo el conocimiento de tipos del cliente lo más alto en la jerarquía de clases que sea posible. Cuando sea posible, el Cliente debe referir a una superclase más que a una subclase específica.
- Evitar subclases que no especialicen nada. Evitar aquellas que difieren solo en el valor de un tipo de atributo.
- Mantener sólo las propiedades en las superclases que sean significativas para las subclases. No es conveniente heredar lo que no se necesita

UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Resumen Guías de Diseño

- Evitar especializar una clase a través de varias dimensiones.
- Usar especialización sólo cuando se tiene una relación “es-tipo-de”.
- Evitar clases donde un objeto debe cambiar su clase dinámicamente, o sea cuando migra de una clase a otra. En particular, evitar especializar una clase basado en los estados de sus instancias o en los roles que juegan esas instancias.
- Evitar composición a través de herencia múltiple cuando se tiene agregación.
- Evitar superclases concretas.
- No olvidar que la agregación es una alternativa a la Herencia cuando se comparte implementación. Usar delegación a través de la asociación cuando sea apropiado.

# Guías de Diseño OO

## Resumen Guías de Diseño

- Cuando sea posible, no permitir al mundo exterior ver las partes de un objeto agregado. Esto permite a las partes cambiar sin alterar el mundo exterior.
- Cuando sea posible, no permitir a las partes de un agregado ver a otro. Esto permite reemplazar una parte sin afectar las otras.
- Para aumentar reuso de las partes, desacoplar las partes de un agregado de su todo.

UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Guías de Diseño Dinámico Flexible

- Una dificultad del Diseño OO es asignar correctamente comportamiento a las clases, base del diseño flexible.
- Las cuestiones a tener en cuenta son >>>
  - Cohesión de Clases y Métodos: Los métodos de una clase deben encuadrar en un único tipo de responsabilidad general.
  - Distribución del Comportamiento: Si un objeto tiene un estado para ejecutar una actividad, debe tener asimismo el comportamiento
  - Extensión vs. Reuso: Se debe lograr un equilibrio en el diseño que favorezca alcanzar ambos requerimientos.

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Cohesión de Clases y Métodos

- Cohesión es una medida de cuán diversas son las características de una entidad >>> Mientras más diversas son las características, menos cohesiva es la entidad.
- Una clase cohesiva representa una abstracción simple.
- Desde el punto de vista del comportamiento, una clase debe encajar en un solo tipo de responsabilidad general.
- Cada método de una clase debe ser cohesivo >>> Debe llevar adelante una función simple específica.

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Distribución de Comportamiento

- Un principio básico de OO es que se debe asignar el estado y el comportamiento referido a ese estado en una misma clase. Formas >>>
- Distribuir comportamiento lo más uniformemente posible. Evitar *objetos mudos* solo almacenan datos.
- Evitar el uso de un controlador >> Objeto que pide información de estado de otro objeto y luego usa esa información para tomar una decisión o ejecutar un cálculo. Ese comportamiento debe estar en el objeto consultado , que es quien tiene el estado. El objeto Cliente debe decir al Servidor que ejecute dicha operación.

UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Distribución de Comportamiento

- Formas >>>

- Reducir acoplamiento Representacional, que mide cuán general es la interfaz usada, cuánto expone del objeto servidor. Si el Cliente obtiene el valor de las variables de estado del servidor, implica que está fuertemente acoplado. Conviene elevar el nivel de la interfaz.
- Utilizar un estilo de programación asertivo en lugar de inquisitivo. El objeto Cliente le dice al servidor qué hacer, en lugar de solicitarle información.

Estas guías son diferentes manifestaciones del mismo principio >> *Un objeto con el estado requerido para alguna actividad, debe llevar adelante dicha actividad*

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Extensión vs. Reuso

- Existe una relación inversa entre la extensibilidad del diseño y la reusabilidad de las clases.
- El hecho de agregar métodos a las clases para extender comportamiento, a veces las vuelve menos reusables.
- Si la meta es desarrollar clases reusables, más que diseños extensibles, se deben aplicar diferentes guías de diseño para asignar comportamiento.

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

# Guías de Diseño OO

## Ejemplo

- Cuál es la forma de conseguir extensibilidad y reuso en una clase Cuenta?
- Para conseguirlo, pueden definirse un par de clases:
- La clase Cuenta es genérica y reusable y puede incluirse en una librería de clases.
- Define los métodos primitivos requeridos por una Cuenta. La clase Cuenta Stock Trading define la interfaz que debe tener una clase Cuenta en una aplicación específica de transacciones de Stock.
- Esta clase usa la clase Cuenta para mantener su cash.
- Las otras clases del sistema refieren sólo a Cuenta Stock Trading. La clase Cuenta y sus instancias son invisibles al resto de la aplicación

# Guías de Diseño OO

## Ejemplo