PARADIGMAS DE PROGRAMACIÓN
Licenciatura en Informática
Programador Universitario

UNIDAD V
PARADIGMAS DE PROGRAMACIÓN
Polimorfismo, Funciones Virtuales y Enlace Dinámico

Polimorfismo

Capacidad de objetos de clases diferentes, relacionadas mediante herencia, de responder de distinta forma a una misma llamada a función miembro

Polimorfismo

- **Función Polimórfica**: Tiene el mismo nombre y la misma signatura para diferentes clases de la misma familia, pero tiene diferente implementación para las distintas clases
- **Objeto Polimórfico**: Tiene el mismo tipo que otros O de una jerarquía de clases, de manera tal que, cada O, aunque se relaciona a través de una clase común, puede tener distinto comportamiento
- **Comportamiento NO Polimórfico**: Cuando, a través de un apuntador de clase base se invoca a una función miembro **no virtual** que ha sido redefinida en las clases derivadas, se usará la versión de clase base, independientemente del contenido del puntero

2

Ver **FacturaTelefónicaConHerencia** en Eclipse

Polimorfismo

Funciones Virtuales → En C++ el uso de funciones virtuales y polimorfismo permite diseñar e implementar sistemas más fácilmente extensibles

- Se declaran en una clase **base** usando la palabra reservada **virtual**

  Se aplica SOLO a funciones miembro de clases
- Una función declarada como virtual **se conserva como virtual** a lo largo de toda la jerarquía de herencia
- Las subclases pueden redefinir funciones virtuales, éstas **deben tener el mismo tipo de retorno y la misma firma** que la función virtual base
- Cuando una clase derivada no redefine una función virtual, sólo heredará la función virtual de la clase base superior inmediata
- ***Polimorfismo* Estático** → Si se invoca a una **función polimórfica** haciendo referencia a un objeto por su nombre y usando el **operador de selección de miembro (.)**, se establece una **ligadura estática** y la función virtual llamada es aquella definida para la clase de dicho objeto (o heredada por ella)

3

Ver **Herencia_PEyPD** en Eclipse

Polimorfismo

Enlace

Dinámico ó Diferido	VS	**Estático o Ansioso**
Ocurre cuando se define una **función polimórfica** para diversas clases en una familia, pero no se une ni se anexa el código real para la función hasta el momento de ejecución		Ocurre cuando se define una **función polimórfica** para diversas clases en una familia y el código real para la función se anexa en el momento de la compilación
POLIMORFISMO DINÁMICO		**POLIMORFISMO ESTÁTICO**
Se implementa mediante → Herencia → Funciones virtuales → Enlace dinámico		El compilador determina a qué función llamará basándose en el tipo del objeto que invoca a dicha función polimórfica

Polimorfismo

Destructores virtuales

Ver **Herencia_PEyPD** en Eclipse

Problema

Si se aplica el operador **delete** a un puntero de clase base, se invocará la función destructor de clase base, independientemente del tipo del O que contenga el puntero

Solución

Se declara virtual al destructor de clase base. Esto hará, automáticamente, virtuales a todos los destructores de las clases derivadas, aunque no tengan el mismo nombre

Se aconseja que, si una clase contiene funciones virtuales, se incluya un destructor virtual, aún si la clase no lo requiere.

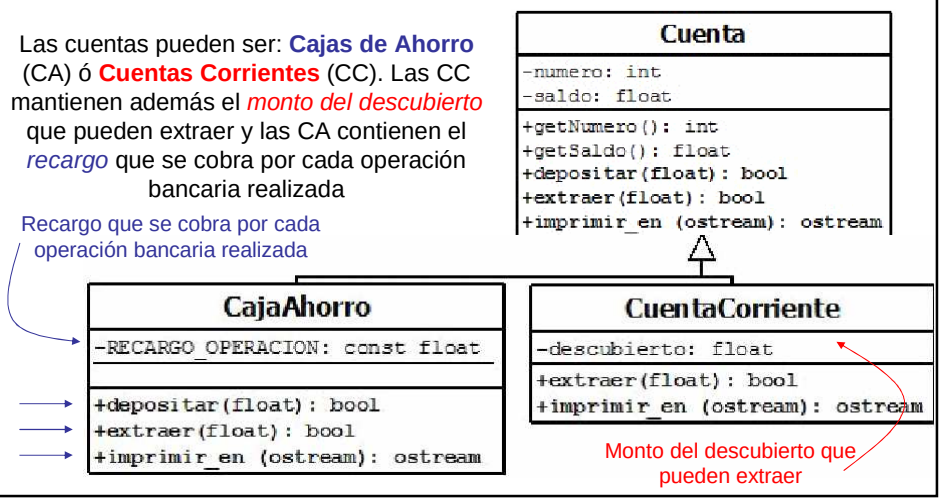
5

Ver **Polimorfismo** en Eclipse

Polimorfismo

Un banco maneja Cuentas que contienen **número de referencia** y un **saldo**.
Las operaciones bancarias que se pueden realizar sobre las Cuentas son ***extraer*** y ***depositar***, y una operación para ***imprimir*** el estado de la cuenta

Las cuentas pueden ser: **Cajas de Ahorro** (CA) ó **Cuentas Corrientes** (CC). Las CC mantienen además el *monto del descubierto* que pueden extraer y las CA contienen el *recargo* que se cobra por cada operación bancaria realizada



Polimorfismo

VENTAJAS

EXTENSIBILIDAD

El software se escribe independientemente del tipo de los objetos a los cuales se les envía mensajes

DESARROLLO DE CÓDIGO MÁS COMPACTO

Evita constructores condicionales de ramificación (enunciados switch) que comprueban el tipo de objeto e invocan a la función correspondiente

CLARIDAD

El código generado es más legible y comprensible

7

Clases Abstractas

CLASES

Ver **Polimorfismo** en Eclipse

Concretas

Clases a partir de las cuales **SI se pueden** generar objetos

Abstractas

Clases a partir de las cuales **NO se pueden** producir objetos

Objetivo

Proporcionar una clase base apropiada, a partir de las cuales las clases puedan heredar interfaz y/o implementación

Clase que contiene una o más **funciones virtuales puras**

```
virtual float ganancias( ) = 0;
```

Función virtual que en su declaración contiene un inicializador = 0

Una **función pura** DEBE ser declarada como virtual

8

Funciones Virtuales y Enlace Dinámico

PROBLEMA
La ligadura dinámica requiere que, en tiempo de ejecución, se encamine la llamada de una función miembro virtual hacia la versión de función virtual apropiada para la clase

vtable

Cada clase que contiene funciones virtuales contiene una **vtable** y para cada función virtual dentro de la clase, la vtable tiene una entrada que contiene un apuntador a la versión de función virtual para uso de un objeto de dicha clase

Cada objeto de una clase con funciones virtuales contiene un **apuntador a la vtable** de dicha clase. Se obtiene el puntero apropiado de función en la vtable y se desreferencia para completar la llamada en tiempo de ejecución

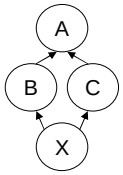
DESVENTAJA → Esta búsqueda y desreferenciación requiere una sobrecarga nominal en tiempo de ejecución

9

Herencia Virtual

Herencia Virtual

Evita la duplicidad de los miembros que se heredan indirectamente más de una vez

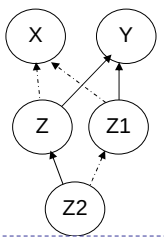


Se añade la palabra **virtual** al especificador de la clase base en el momento de la herencia

Ver **HerenciaVirtual2** en Eclipse

Características

- Cuando una clase se hereda virtualmente todos sus miembros, incluidos los destructores son virtuales
- Los constructores de las clases base heredados virtualmente son invocados antes que los de las clases base heredados no virtualmente. Y dentro de los virtuales, se los invoca en el orden en que fueron declarados
- Cabe recordar que los constructores de las clases base heredados no virtualmente tienen más prioridad que cualquiera de sus derivados

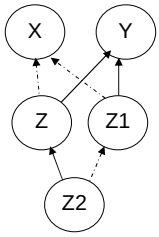


Ver **HerenciaVirtual** en Eclipse

Herencia Virtual

Orden de llamada a los constructores

- Los constructores de las clases base heredados virtualmente son invocados antes que los de las clases base heredados no virtualmente. Y dentro de los virtuales, se los invoca en el orden en que fueron declarados
- Cabe recordar que los constructores de las clases base heredados no virtualmente tienen más prioridad que cualquiera de sus derivados




```
Orden de llamada a los constructores de a
Constructor X
Constructor Y
Constructor Z1
Constructor Y
Constructor Z
Constructor Z2
```

11