FACULTAD DE INFORMÁTICA

Objetos y listas de objetos

UN REPASO

Programación orientada a objetos

Autor: Luis Hernández Yáñez

FdI UCM Las clases (1)

Una clase es la definición de un tipo de objetos.

Cada tipo de objetos (clase) se define...

... posiblemente, en base a otra clase existente (superclase).

... con un conjunto de atributos.

... con un conjunto de métodos.

Como módulo que es, la clase tiene interfaz e implementación.

La interfaz (lo público) se encuentra en el archivo de cabecera (**.h**) y consiste en el nombre de la clase y los prototipos de los métodos.

La implementación (lo privado) se encuentra repartida entre el archivo de cabecera y el archivo de implementación, aunque su mayor parte está en el segundo.

En el **.h** se encuentra la declaración de los atributos y de los métodos privados, y en el **.cpp** la implementación de los métodos y la inicialización de los atributos de clase.

FdI UCM Las clases (1)

.h

```
#ifndef listaper_h
#define listaper_h
#include "persona.h"
```

Interfaz

```
class Lista {
public:
  Lista();
  ~Lista();
  bool llena() const;
  bool vacia() const;
  bool insertar(Persona*);
  bool recuperar
    (int, Persona*&) const;
  void mostrar() const;
private:
  enum { MAX = 100 };
  Persona* _array[MAX];
  int _cont;
};

#endif
```

Implementación

.cpp

```
#include "listaper.h"
#include <iostream>
using namespace std;

Lista::Lista() : _cont(0) { }

Lista::~Lista() {
  for(int i = 0; i < _cont; i++)
    delete _array[i];
}

bool Lista::llena() const
{ return _cont == MAX; }

bool Lista::vacia() const
{ return _cont == 0; }

bool Lista::insertar(Persona* p) {
  if(_cont == MAX) return false;
  _array[_cont] = p;  _cont++;
  return true;
}
...
```


FdI UCM Las clases (2)

Existen clases estándar (predefinidas) que se pueden usar en otras clases (**iostream** para entrada/salida, **string** para cadenas).

Las clases son las definiciones de los tipos de objetos que se manejarán en los programas, pero lo que existe durante la ejecución son los objetos, los ejemplares de las clases.

Los atributos constituyen el contenido de los objetos y las relaciones que tienen establecidas con otros objetos. Son siempre privados.

Los métodos son los servicios que proporcionan los objetos de la clase. Se usan por medio de pasos de mensajes a objetos de esa clase. Aunque puede haber algunos privados (auxiliares), la mayoría son públicos.

Las clases pueden ser abstractas, estableciendo requisitos de interfaz para sus subclases. No se pueden crear ejemplares de clases abstractas.

La superclase (herencia)

Si una clase se define tomando como base otra clase (su *superclase*), automáticamente hereda todas las características de aquélla:

- ✓ Los atributos (propios o heredados).
- ✓ Los métodos (propios o heredados), a excepción de los constructores, el destructor y el operador de asignación.

C++ distingue distintas formas de aplicar la herencia (derivación pública, privada o protegida), aunque la pública es la más habitual (lo público de la superclase sigue siendo público en la subclase).

Lo privado de la superclase siempre es inaccesible en la subclase.
Cuando resulte imprescindible acceder a atributos de la superclase en los métodos de la subclase, se puede hacer que sean protegidos esos atributos.

Se pueden redefinir métodos heredados y se puede forzar la ejecución de los métodos de la superclase redefinidos.

Los constructores, destructores y operadores de asignación no se heredan (se puede forzar la ejecución de los de la superclase).

Los atributos (1)

Los atributos conforman el contenido de los objetos de la clase y las relaciones establecidas con otras clases de objetos.

Atendiendo a su cometido distinguimos dos tipos de atributos:

- ✓ Contenido de los objetos.
- ✓ Relaciones con otros objetos.

Las relaciones se implementan por medio de punteros.

El contenido de los objetos se puede implementar por medio de:

- ✓ Atributos estáticos.
- ✓ Atributos dinámicos.

Los atributos dinámicos se implementan por medio de punteros y se alojan en el montón *(heap*, la memoria dinámica*)*.

Las relaciones simplemente se establecen (copia de punteros), mientras que los atributos dinámicos se crean y se destruyen a medida que se crean y se destruyen los objetos que los contienen.

Los atributos (2)

En C++ un atributo puede ser un dato simple (de un tipo básico), un objeto de otra clase o un array de atributos.

La clase es cliente de todas aquellas clases que se usen en la definición de sus atributos.

- ✓ Se pueden definir atributos de clase, atributos que son compartidos por todos los objetos de la clase (`static`).
- ✓ Los atributos no se pueden inicializar al mismo tiempo que se definen.
- ✓ Los atributos de clase se inicializan en el archivo de implementación.
- ✓ Los atributos de ejemplar se inicializan en los constructores.

Los métodos (1)

Los métodos son funciones miembro que se pueden aplicar sobre los objetos de la clase. Si son públicos se pueden aplicar sobre los objetos de la clase por medio de pasos de mensaje. Si son privados se pueden aplicar sobre los objetos en las implementaciones de otros métodos.

Un método puede tener dos formas básicas:

- ✓ Función miembro normal.
  Para el paso de mensajes se usa la notación punto.
- ✓ Función miembro operadora.
  Para el paso de mensajes se usa la notación de operador.

Los métodos pueden tener definidos parámetros, pueden usar variables/objetos locales y pueden devolver resultados.

El código de los métodos es una combinación de instrucciones de control, expresiones y pasos de mensajes a objetos.

Los métodos (2)

Distinguimos distintas categorías de métodos:

- ✓ Constructores y destructores (FCO).
- ✓ Operador de asignación (FCO).
- ✓ Accedentes y mutadores (adecuados para clases de información, pero posiblemente no para clases que implementan *máquinas de datos*).
- ✓ Lectores y visualizadores.
- ✓ Clonadores.
- ✓ Otros.

Los constructores pueden ser *normales* o de copia.
Los constructores *normales* pueden tener definidos parámetros y argumentos implícitos para ellos, pero siempre debe existir un constructor predeterminado (que se pueda invocar sin argumentos).

Los constructores deben inicializar, por medio de inicializadores si es posible, los atributos que no se inicialicen automáticamente (objetos de otras clases) o que se quieran inicializar con valores distintos de los predeterminados. Pueden traspasar al constructor de la superclase argumentos para que los utilice cuando le toque ejecutarse.

Los métodos (3)

Los constructores deben crear los atributos dinámicos e inicializar por defecto a **`NULL`** los atributos que son relaciones.

El constructor de copia inicializa los atributos con los valores de los atributos del objeto usado como punto de partida.

Los destructores deben destruir los atributos dinámicos.

El operador de asignación debe copiar todos los atributos del objeto origen sobre los atributos del objeto destino, pero teniendo en cuenta que los atributos dinámicos ya existen en el objeto destino.

Los accedentes deben devolver copias de los atributos. Si el atributo es dinámico lo que devuelve es una copia del objeto apuntado.

Los mutadores copian los valores proporcionados en los atributos.

Los métodos clonadores deben devolver copias dinámicas de los objetos.

Los métodos (4)

Un método público se invoca por medio de un paso de mensaje a un objeto de esa clase. Ese objeto se identifica como el objeto receptor del mensaje y se puede manejar en el código del método por medio de su representación **`this`** (puntero). Para pasar un mensaje dentro de un método al objeto receptor no es necesario colocar delante del mensaje objeto receptor alguno.

Un método será constante (**`const`**) si no modifica de ninguna forma el objeto receptor del mensaje.

En los métodos se usan los objetos parámetros y los objetos locales de la misma forma que cualquier otro objeto: pasándoles mensajes.

Los métodos pueden devolver objetos como resultado de su ejecución y también el propio objeto receptor del mensaje, con el fin de poder encadenar varios mensajes seguidos:

```
objeto.mensaje1().mensaje2();          (objeto++)++;
```


Los métodos (5)

Los métodos de las superclases que se vean afectados por problemas de vinculación deben ser métodos virtuales.

Los destructores deben ser siempre virtuales.

Un método virtual puro es un método virtual que no se implementa en la clase. Eso convierte a la clase automáticamente en abstracta.

Los objetos

Un objeto es un ejemplar de una clase. Se crea declarándolo como de esa clase (nombre de la clase como tipo).

Un objeto se puede crear en el programa principal (objeto global), dentro de un método (objeto local) o ser un parámetro que representa al objeto argumento que se pasará al método.

Los objetos siempre se manejan por medio de pasos de mensajes.

Un objeto constante es un objeto cuyo estado no se puede modificar (no se pueden modificar sus atributos).

Un objeto puede ser estático o dinámico. Los objetos dinámicos se crean en memoria dinámica por medio del operador **new** y se destruyen con el operador **delete**.

La regla de compatibilidad de objetos permite asignar a un objeto (o un puntero a objeto) otro objeto (o puntero a otro objeto), siempre que el que se copie sea un objeto de la misma clase o de una subclase de la del que recibe la copia.

Polimorfismo

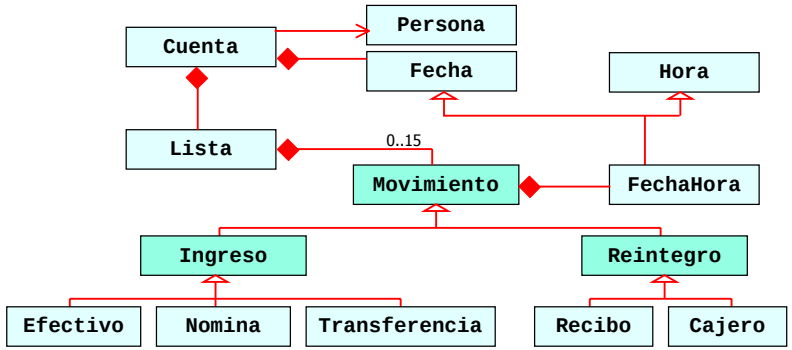
Con la herencia se consiguen jerarquías de clases que, junto con la regla de compatibilidad de objetos, nos permiten crear listas polimórficas, listas que contienen distintas clases de objetos.

Para que las listas polimórficas funcionen bien es necesario que se implementen como colecciones de punteros y que se resuelvan los problemas de vinculación por medio de los métodos virtuales.

La copia de listas polimórficas requiere que se disponga de métodos de clonación en las clases de la jerarquía correspondiente.

Un caso de estudio

Analicemos detenidamente el siguiente sistema:



(No es exactamente el mismo caso que el del taller, ya que la clase **Cuenta** tiene establecida una relación con la clase **Persona**.)

Un caso de estudio