

UTN Facultad Regional
Avellaneda
Departamento Electrónica

Materia: Informática II
Tema: Nociones de C++
Clase 1
Ing. Gustavo Viard

Nociones básicas de Orientación a Objetos(OO).

“Un sistema orientado a objetos es un conjunto de objetos que colaboran entre sí enviándose mensajes”.

En el Mundo de los Objetos sólo existen dos cosas **objetos** y **mensajes**.

Definición de Objeto.

Un objeto es una entidad del mundo real.

Los Objetos son cosas:

Reales o imaginarias.

Simples o complejas.

Ejemplo de Objetos: Satélite, Empleado, pincel, Número Complejos, Cuenta bancaria, automóvil, etc.

Validación de los Objetos (I) .

1. Relevancia en el dominio del problema.

¿El objeto existe dentro del dominio del problema planteado?

¿Este objeto es necesario para que el sistema complete sus necesidades?

¿Es requerido como parte de la interacción entre el usuario y el sistema?

¿Puede ser una características de otros objetos?

Validación de los Objetos (II) .

2. Necesidad de existencia independiente.

Para ser un Objeto y no una característica de otros objetos, el objeto debe existir independientemente.

Validación de los Objetos (III) .

3. Que el objeto tenga atributos y operaciones.

Todo objeto debe tener:

Atributos: Son las características del objeto.

Operaciones: Son las cosas que un objeto puede hacer.

Ejemplo: El objeto nube tiene:

Atributos: Tamaño, cantidad de agua, forma.

Operaciones: Llover, tronar y nevar..

Objetos.

Nombre de atributos y operaciones:

Los nombres de atributos y operaciones comienzan con letra minúscula.

En le caso de ser varias palabras se concatenan, sin espacios intermedios.

Las siguientes palabras van en Mayúsculas.

Ejemplo: numOrden

Definición de Clase.

Una clase es una descripción de un **conjuntos de objetos** que comparten los mismos atributos, operaciones, relaciones y semántica.

Los nombres de las Clases van en letra mayúsculas por convención.

Un Objeto es una instancia específica de una clase dada(hojas, pelota auto)

Auto es una clase miAuto es un Objeto.

Definición de Clase.

Clase

Cliente
nroCliente
nombre
apellido
direccion
darAltaCliente()
darBajaCliente()
actualizarDatosClientes()

Objeto

JuanLopez:Cliente
43352
Juan
Lopez
San Martin6936 4 A
darAltaCliente()
darBajaCliente()
actualizarDatosClientes()

Pilares de Orientación a Objetos

1. Encapsulamiento
2. Herencia
3. Polimorfismo

Pilares de Orientación a Objetos (Encapsulamiento)

Un objeto **“contiene”** o **“conoce”** los datos necesarios para cumplir con sus tareas.

Estos **datos y los procedimientos** que el objeto utiliza para manejarlos son **inaccesibles desde “afuera”** del Objetos.

Un objeto es como una **“caja negra”** que recibe mensajes y actúa según su **esencia**. Desde afuera no podemos saber nada más.

Pilares de Orientación a Objetos (Encapsulamiento)

El encapsulamiento separa **los aspectos externos de un objeto** de los **detalles de implementación internos**.

Los **cambios internos** no afectan entonces la **interface externa**(que es un conjunto de operaciones que un objeto puede realizar).

Parte de un Objeto Encapsulado

Interfece Publica (externa)

Implementación (interna).

**Información interna para la
implementación.**

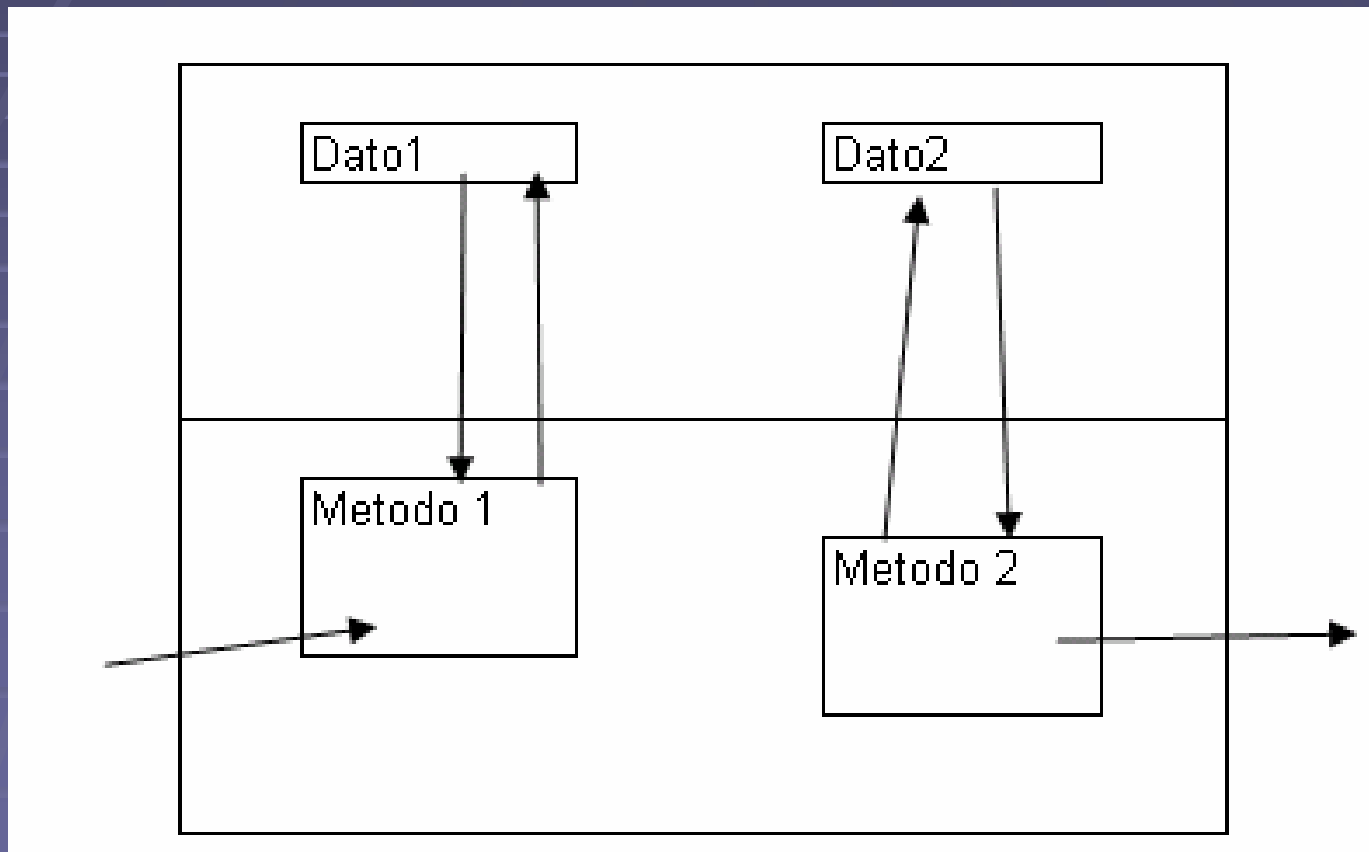
Cómo Implementar Encapsulamiento

Los atributos y operaciones de un objeto son sus miembros.

Los miembros de un objeto pueden ser privados o públicos.

En los sistemas OO puros, los atributos son privados y pueden ser cambiados o accedidos a través de operaciones o métodos públicos.

Cómo Implementar Encapsulamiento



Herencia

Permite agrupar clases relacionadas de modo que puedan ser manejadas colectivamente.

La relación conocida como HERENCIA es la relación:

A es un B

Ejemplo, un Gato es un animal

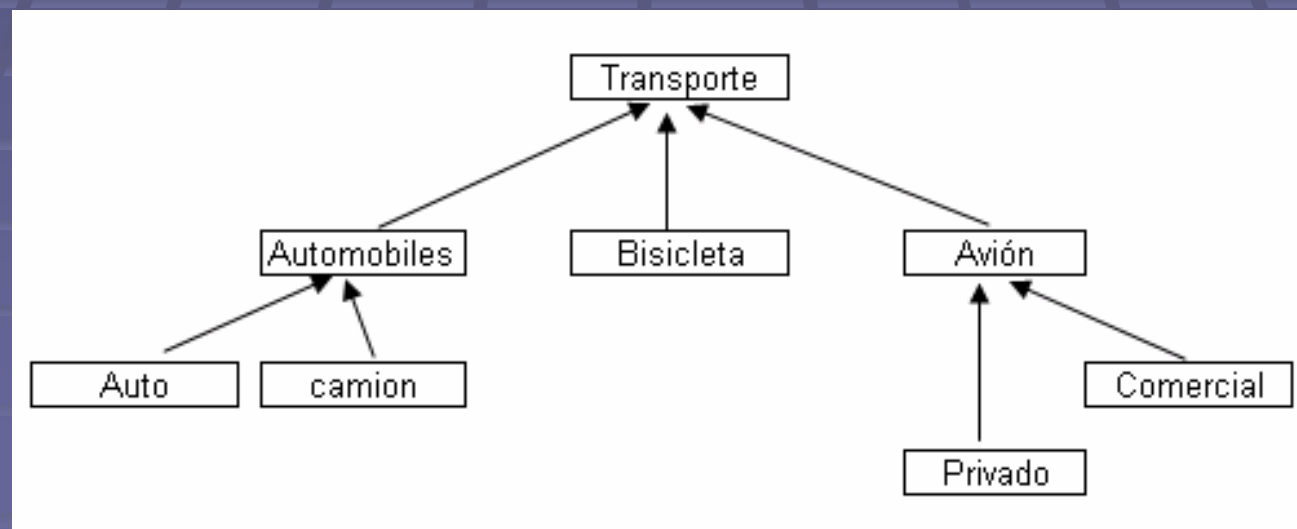
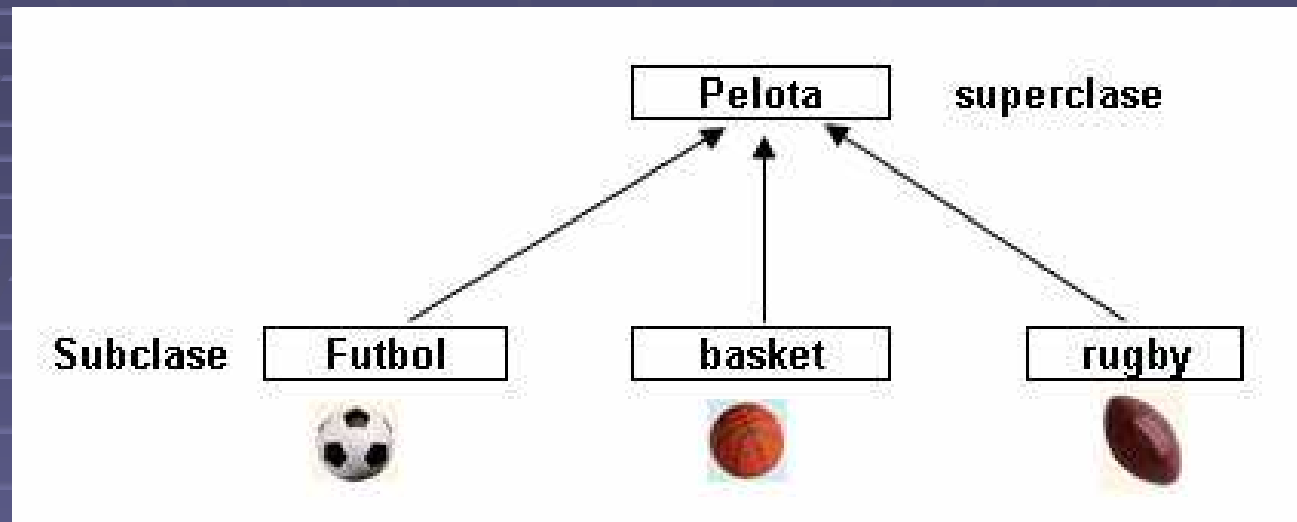
Herencia

Una Clase hereda de su clase padre (superclase) su interfaz (que es un conjunto de operaciones) y su Comportamiento.

La clase que hereda puede modificar o ampliar dicho comportamiento. O sea que las clases hijos pueden sobrescribir los métodos heredados.

El proceso de diseñar una clase heredando de otra se llama subclasificación.

Herencia



Consola del C++

Salida por consola estilo C++

`cin >>`

`cout <<`

`cin` y `cout` son corrientes asociadas
(streams) standar de consola

Consola del C++

c-in --> input ingreso : esta representando el teclado .

cin >> variable;

c-out --> Output : esta representando a la pantalla.

cout << expresión; (expresión: variables, literales, caracteres de control...)

Consola del C++

```
#include <iostream.h>
void main()
{
    int entero=1;
    float flotante=2.36;
    double doble=2.36969966;
    char caracter='L';
    char cadena[]="Cadena de caracteres";
    cout << entero;
    cout << flotante;
    cout << doble;
    cout << caracter;
    cout << cadena;
}
```

Consola del C++

En C++ los comentarios son // y abarcan toda la línea

Igualmente es compatible con los comentarios estilo c /* */

```
// comentario estilo c++  
/*  
comentario estilo c  
*/
```

Noción de clases

Unas de las características más importantes el C++ es el concepto de clase.

Las clases son el tipo de dato a partir de la cual se puede crear un objeto.

Las “funciones” y “variables” se los denomina miembros de la clase.

Noción de clases

Puede haber distintos tipos de miembros, por defecto los miembros son Privados (**private**), esto quiere decir que solo puede ser accedidos por funciones (se suelen llamar métodos) miembros .

Noción de clases

Por otra parte el tipo publico (**public**) significa que el miembro puede ser accedido como un campo de la struct.

```
class <nombre>
{
    lista privados
    public:
    lista publicos
};
```

<nombre> ← Pasa a ser un nuevo tipo de dato.

Noción de clases

```
class primera {
    int x;
    public:
    int y;
    void f(int u);
};

// Los :: se llama resolución de ámbito: es para
// saber a que clase pertenece la función.

void primera :: f(int u)
{
    x=u; <==Puede acceder a x ya que es miembro
}
.....
primera ob;

ob.x=1; <==MAL El dato es private
ob.y=1; <==Bien El dato es public
ob.f(5); <==Bien El dato es public
```

Principales diferencias entre es c y el c++

- A. Prototipos:
 - * C es opcional
 - * C++ Obligatorios
- B. Los valores de retornos en C++ son Obligatorios.
- C. Declaración de Variables
 - * En C solo al principio del Bloque
 - * En C++ En cualquier lugar
- D. El tipo **void** como parámetro en una función
 - * C es necesario
 - * C++ es redundante

Sobrecarga de funciones

Dos funciones pueden compartir el nombre siempre y cuando varíen en el **nro de parámetro y/o el tipo de dato**. El compilador va a decidir cual se invoca de acuerdo a los datos suministrado en la llamada.

Sobrecarga de funciones

```
#include <iostream.h>
int suma(int x,int y);
float suma(float x,float y);
void main()
{
float fnro1,fnro2;
int inro1,inro2;
cout << "\nIngrese dos nros Flotantes: ";
cin >> fnro1 >> fnro2;
cout << "\nIngrese dos nros Enteros: ";
cin >> inro1 >> inro2;
cout << "\nLa suma de los flotantes es :" << suma(fnro1,fnro2);
cout << "\nLa suma de los enteros es :" << suma(inro1,inro2);
}
int suma(int x,int y)
{
return x+y;
}

float suma(float x, float y)
{
return x+y;
}
```