



Programación Orientada a Objetos

Clase – C++

PARADIGMAS DE PROGRAMACION

UTN - La Plata



Características de C++.

- ◉ Es un lenguaje compilativo híbrido
- ◉ Todo programa en C++ consta de *objetos*
- ◉ Todo objeto tiene:
 - *estado*: dado por sus *datos miembros*
 - *comportamiento*: dado por sus *funciones miembro*
- ◉ Todo objeto es *instancia* de una *clase*
- ◉ La *clase* encapsula estado y comportamiento

Características de C++.

- Los objetos se comunican vía *mensajes*

Por ejemplo si p es un objeto de la clase Persona

`p.verEdad`

le envía el mensaje `verEdad` al objeto p

- El envío del mensaje desencadena la ejecución de la función miembro asociada

- C++ tiene 2 funciones miembro especiales:

- Constructor

- Se llama igual que la clase
- Se invocan al definir el objeto
- Admite parámetros
- No retorna valores
- Si no se define, el compilador lo crea automáticamente

Ejemplo: `class Persona → Persona p` (\approx new en small)

- Destructor

- El nombre es la negación del nombre de la clase
- No admite parámetros
- No retorna valores

Ejemplo: `~Persona p`

Definición de una clase.

- Se deben especificar sus dos componentes:
 - Una zona de declaración: contiene la lista de datos miembro (v.i. de Small)
 - Una zona de implementación: define e implementa las funciones miembro (métodos de Small)

Definición de una clase.

- ⦿ Una clase puede contener una parte pública y una privada. Por defecto la clase es *privada*.
- ⦿ Toda clase tiene una *visibilidad*: privada, pública o protegida. La *visibilidad* define qué se puede acceder y cuál es el tipo de acceso:
 - Lo público se accede directamente
 - Lo protegido y lo privado sólo a través de funciones miembros

Definición de una clase.

Class Persona

```
{ //datos miembro -- privados
    int edad;
    char nom[30];
```

```
    //funciones miembro
public:
    int verEdad ();
    char verNom ();
    void modEdad( int x);
    void modNom (char otroN [30];
    Persona ();
    ~Persona();
};
```


```
int Persona :: verEdad()
    {return edad;}
```

```
char Persona :: verNom()
    {return nom;}
```


```
void Persona :: modEdad( int x)
    {edad:= x;}
```

```
void Persona :: modNom( char otroN [30])
    {strcpy(nom,otroN);};
```

```
void Main()
{   Persona p1;
    p1.modEdad(23);
    p1.modNom('luis');
    .....
    .....
}
```



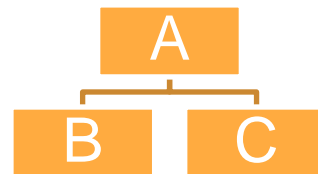
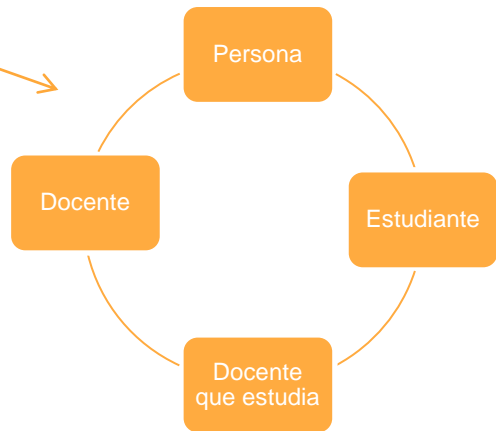
Se graba con extensión .hpp



Se graba con extensión .cpp

Herencia.

- C++ presenta herencia simple y múltiple



- La subclase:
 - Hereda datos miembro y funciones miembro de su/s superclase/s
 - No hereda el constructor ni el destructor
 - Agrega datos miembro y funciones miembro

Definición de una subclase.

- Se debe determinar de quién es subclase y el tipo de derivación: *pública, protegida y privada*
- El tipo de derivación define el acceso que tiene la subclase a los datos miembros de la superclase.
- Los datos miembros privados NO se heredan

Subclase:

- Pública* → hereda la parte pública y protegida de la superclase con igual visibilidad
- Protegida* → hereda la parte pública y protegida, ambas como protegidas
- Privada* → hereda la parte pública y protegida, ambas como privadas

Definición de una subclase con herencia simple.

Ejemplo:

Class A

```
{ int a;  
  public:  
    int b;  
  protected  
    float c;  
};
```

Class B: public A //tipo de derivación

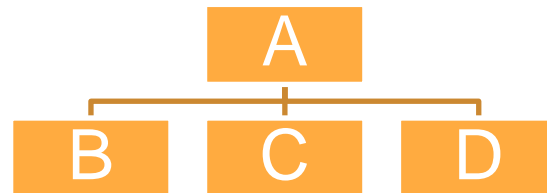
```
{ .....};
```

Class C: protected A

```
{ .....};
```

Class D: private A

```
{ .....};
```



- La variable **a** NO se hereda
- Las variables **b** y **c**:
 - Se heredan en B como están en A
 - Se heredan en C ambas como protegidas
 - Se heredan en D ambas como privadas

Definición de una subclase con herencia múltiple.

Class A

{};

Class B

{};

Class C: public A, public B

{.....};

➤ La clase C hereda de A y de B, las variables no privadas, de acuerdo a como están en dichas clases.



Programación Orientada a Objetos

Python

PARADIGMAS DE PROGRAMACION

UTN - La Plata



Python es un lenguaje de programación interpretado.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Es un lenguaje dinámico y multiplataforma.

Python al igual que C++ presenta herencia simple y múltiple.

Definición de una clase.

En Python para definir una nueva clase se coloca la palabra `class` seguida de su nombre:

- `class Auto:`
- `class Mascota:`
- Por convención los nombres de clases deben comenzar en mayúscula .

Definición de una clase.

Dentro de la clase se especifican los métodos y los atributos comunes a todos los objetos de dicha clase.

Todo lo que va dentro de una clase se escribe con sangría.

La primer sentencia que no esté con sangría, no pertenece a la clase.

Definición de una clase.

- Los constructores se ejecutan automáticamente justo después de crear o instanciar un objeto.
- Los constructores son métodos especiales mediante los cuales los programadores pueden inicializar una instancia.
- Los constructores en Python se definen codificando un método especial llamado **`__init__`**.
- En el **`__init__`** se inicializa las variables de instancia del objeto, dejándolas disponibles para comenzar a operar con ellas a través de los métodos.

Definición de una clase: uso de self

- La palabra reservada **self** sirve para referirse al objeto actual, que es el que recibe el mensaje.
- Es utilizada dentro del método para señalarse a sí mismo y sirve para poder acceder a los atributos y métodos del objeto actual.

En todos los métodos de una clase el primer parámetro es siempre **self**.

Definición de una clase.

El guión bajo indica que los atributos son privados.

```
class Mascota:  
    def __init__(self,nom,esp,due,ed):  
        self.__nombre=nom  
        self.__especie=esp  
        self.__duenio=due  
        self.__edad=ed
```

} Constructor

```
    def verNom(self):  
        return self.__nombre
```

```
    def verEdad(self):  
        return self.__edad
```

```
    def verEsp(self):  
        return self.__especie
```

```
    def verDuenio(self):  
        return self.__duenio
```

Se guarda con el nombre clamascota.py

Definición de una clase.

```
def modNom(self,otro):  
    self.__nombre=otro
```

```
def modEd(self,otra):  
    self.__edad=otra
```

```
def modEsp(self,otra):  
    self.__especie=otra
```

```
def modDuenio(self,otro):  
    self.__duenio=otro
```

Creación de objetos.

- Para crear un objeto hay que instanciarlo a partir de una clase.
- `from clamascota import *`
- `m=Mascota('beto','perro','Jorge',5)`
- La forma en que los métodos son invocados difiere de la forma de invocar a las funciones. Se debe utilizar la notación de punto para invocar los métodos.
- `m.modNom('lolo')`
- `print(m.verEdad())`

Creación de una subclase con herencia simple.

En Python para definir una subclase se coloca la palabra `class` seguida de su nombre y entre paréntesis el nombre de la superclase:

- `class Auto(Vehiculo):`
- `.....`
- `class Mascota (Animal):`
- `.....`
- Hereda todos los atributos y métodos.

Creación de una subclase con herencia múltiple.

- `class Base1: pass`
- `class Base2: pass`
- `class Multiderivada (Base1,Base2): pass`

En este caso, cuando busca un método o atributo lo hace en la clase Multiderivada primero, luego va a Base1 y finalmente a Base2 (de izq a derecha según definición de derivacion)

