Laboratorio de Computación II

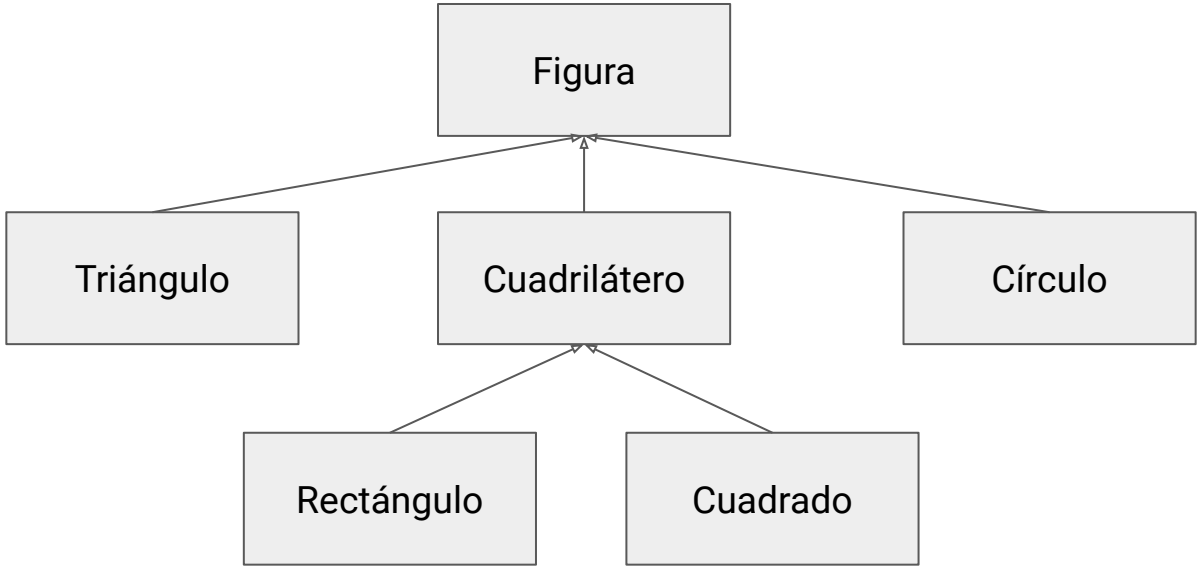
Programación orientada a objetos
Herencia simple

Características de la POO

- Construcción y destrucción: La capacidad de realizar acciones al crearse y al destruirse el objeto de manera automática.
- Encapsulamiento: Restricción del acceso a elementos (propiedades y métodos) que no deban accederse desde fuera de clase. Pueden accederse si se les desarrolla un método público para tal fin.
- Composición y agregación: Diferentes maneras de relacionar clases.
- Herencia: Poder crear una clase derivada a partir de una clase base de la cual obtendrá todos los rasgos.
- Polimorfismo: La posibilidad de que varios objetos de diferentes clases, pero con una base común, se comporten diferente ante una misma acción.

Herencia

Tipo de relación entre clases que permite que una clase Derivada herede atributos y métodos de una clase Base.



Herencia

Tipo de relación entre clases que permite que una clase Derivada herede atributos y métodos de una clase Base.

```
class Base{
  protected:
    int a, b;
  public:
     Base();
};
class Derivada:public Base{
  private:
    float c, d;
  public:
     Derivada();
};
```

En el ejemplo que se puede observar. La clase llamada Derivada hereda los atributos y métodos de la clase Base.
Para cumplir conceptualmente con el mecanismo de herencia la siguiente afirmación debe ser correcta: La clase Derivada *<<es un>>* o *<<es un tipo de>>* clase Base.
Alumno es una Persona → Heredar
Alumno es una Fecha → No heredar

Protected

```
class Persona{
  protected:
    int edad;
    char apenom[50];
  public:
     void cargar();
     void mostrar();
     int getEdad();
};
class Alumno:public Persona{
  private:
    int legajo;
  public:
     void cargar();
     void mostrar();
};
```

Es un especificador de acceso que permite a la clase y sus clases derivadas hacer uso de los métodos y atributos pero, a la vez, los encapsula si se quiere acceder desde fuera de la clase.

```
void Alumno::cargar(){
  cin >> edad >> apenom; // OK
  cin >> legajo;
}
```

```
int main(){
  Alumno obj;
  cout << obj.getEdad(); // OK
  cout << obj.edad; // Encapsulado
}
```

Métodos de la clase base

```
class Persona{
  protected:
    int edad;
    char apenom[50];
  public:
     void cargar();
     void mostrar();
     int getEdad();
};
class Alumno:public Persona{
  private:
    int legajo;
  public:
     void cargar();
     void mostrar();
};
```

A diferencia del encapsulamiento, no disponemos de una variable con los atributos y métodos de la clase base. Esto se debe a que nuestro objeto de clase derivada también es un objeto de la clase base.
Por lo tanto, si al cargar un objeto de Alumno quisieramos llamar al cargar de Persona. Tenemos que llamar al método indicando a qué clase nos referimos.

```
void Alumno::cargar(){
  Persona::cargar();
  cin >> legajo;
}
```

Herencia y constructores

```
class Persona{
  protected:
    int edad;
    char apenom[50];
  public:
    Persona(int, char *);
};
class Alumno:public Persona{
  private:
    int legajo;
  public:
     Alumno(int, int, char *);
};
```

Un constructor puede llamarse explícitamente dentro de otro constructor pero debe hacerse cumpliendo la siguiente sintaxis.

```
Alumno::Alumno(int l, int e, char *a){
  Persona::Persona(e, a);
  legajo = l;
}
```

```
Alumno::Alumno(int l, int e, char *a):Persona(e, a){
  legajo = l;
}
```

Ejemplos en C/C++