# Herencia en C++ (Parte 3) Polimorfismo

# Definición del concepto

Es la propiedad que tienen los objetos para comportarse de forma diferente ante el mismo mensaje, es decir, una función puede hacer que ocurran distintas acciones, dependiendo del tipo del objeto con el que se invoca la función.

Por ejemplo, para toda figura en 2D existe una función ***calcularArea***(), pero el área se calcula de forma distinta dependiendo del tipo de figura (objeto de clase) que llame a la función. En este sentido el Polimorfismo esta ligado al concepto de Herencia, ya que todas las clases de la jerarquía definirán la misma función ***calcularArea***() pero con distinta implementación.

# Funciones *virtual*

Para implementar el Polimorfismo es necesario declarar la función que es común para todas las clases de la jerarquía de Herencia como ***virtual***, y llamarla a través de un apuntador o referencia de la clase base. De esta forma C++ elije en forma dinámica la función correcta para la clase a partir de la cual se instanció el objeto.

En el ejemplo anterior, la función ***calcularArea***() sobrescrita en una clase derivada (Subclase) tiene la misma firma y el mismo tipo de valor de retorno que la función declarada en su clase base (Super Clase). Por lo tanto al declarar la función de la clase base como ***virtual***, podemos sobrescribir esa función para permitir el comportamiento polimórfico.

# Sintaxis de *virtual*

Para definir una función virtual, se antepone al prototipo de la función que estará en la clase base (Super Clase) la palabra reservada ***virtual***. Por ejemplo,

***virtual*** void ***calcularArea***();

Y para el llamado a esta función, el objeto de la clase derivada debe realizarlo por medio de un apuntador de la clase base a un objeto de la clase derivada. Por ejemplo,

figCuadrado->***calcularArea***(); o

figCirculo->***calcularArea***();

En este caso el programa elije la función ***calcularArea***() correcta de la clase derivada en forma dinámica (es decir, en tiempo de ejecución) con base en el tipo del objeto.

# Ejemplo (1)

Implementar una jerarquía de Herencia que permita representar las figuras en 2D: **Cuadrado**, **Rectángulo**, **Paralelogramo** y **Circulo**.

Para cada una de ellas es necesario calcular el área.

En este caso se declarara una Super Clase llamada **Figura2D** que definirá una función ***virtual calcularArea***() que será sobrescrita por las clases **Cuadrado**, **Rectangulo**, **Paralelogramo** y **Circulo**.

# Ejemplo (2)

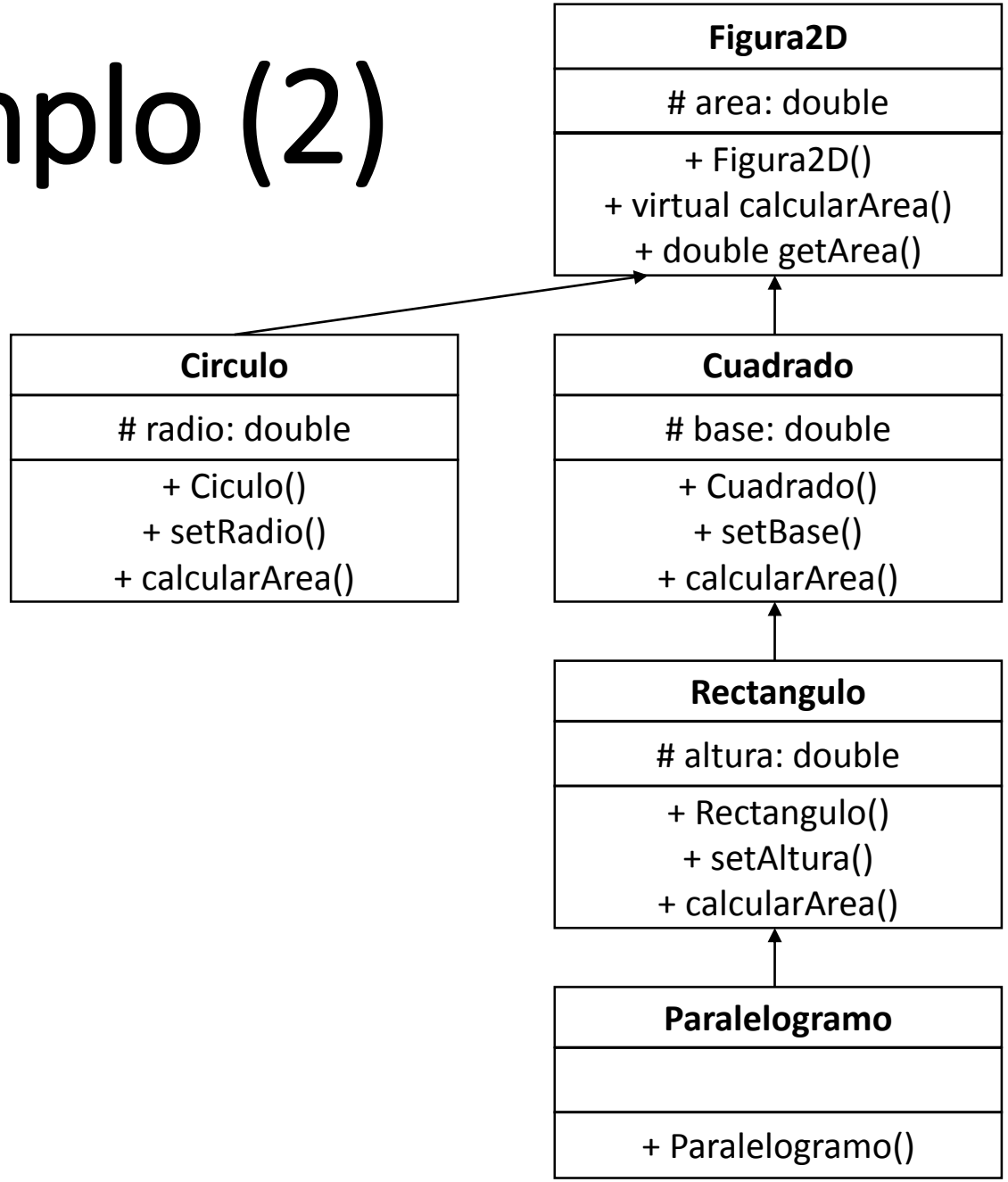


**Diagrama de Clases**

# Ejemplo (3)

## Declaración de las Clases (.h)

**SuperClase**

```
class Figuras2D
{
    public:
        Figuras2D();
        virtual ~Figuras2D();
        virtual void calcularArea();
        double getArea();

    protected:
        double area;

    private:
};
```

**SubClases**

```
class Circulo: public Figuras2D
{
    public:
        Circulo();
        virtual ~Circulo();
        void setRadio();
        void calcularArea();

    protected:
        int radio;

    private:
};

class Rectangulo: public Cuadrado
{
    public:
        Rectangulo();
        virtual ~Rectangulo();
        void setAltura();
        void calcularArea();

    protected:
        double altura;

    private:
};
```

```
class Cuadrado: public Figuras2D
{
    public:
        Cuadrado();
        virtual ~Cuadrado();
        void setBase();
        void calcularArea();

    protected:
        double base;

    private:
};

class Paralelogramo:
    public Rectangulo
{
    public:
        Paralelogramo();
        virtual ~Paralelogramo();

    protected:

    private:
};
```

**Ejemplo completo: PolimorfismoFiguras2D.zip**