



Sesión 3. Herencia

MODELO EDUCATIVO
TEC21

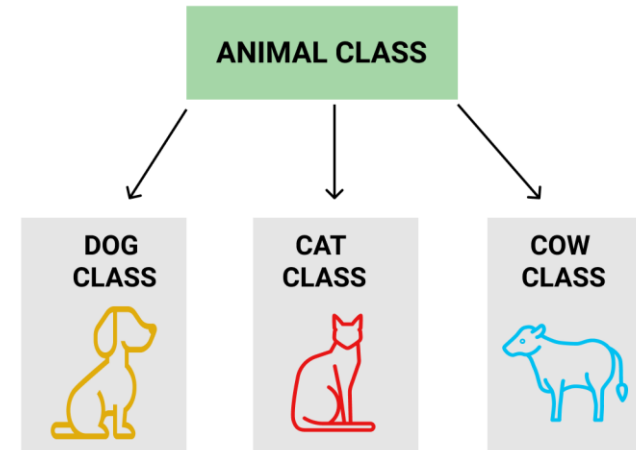
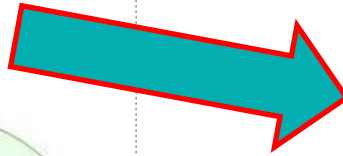
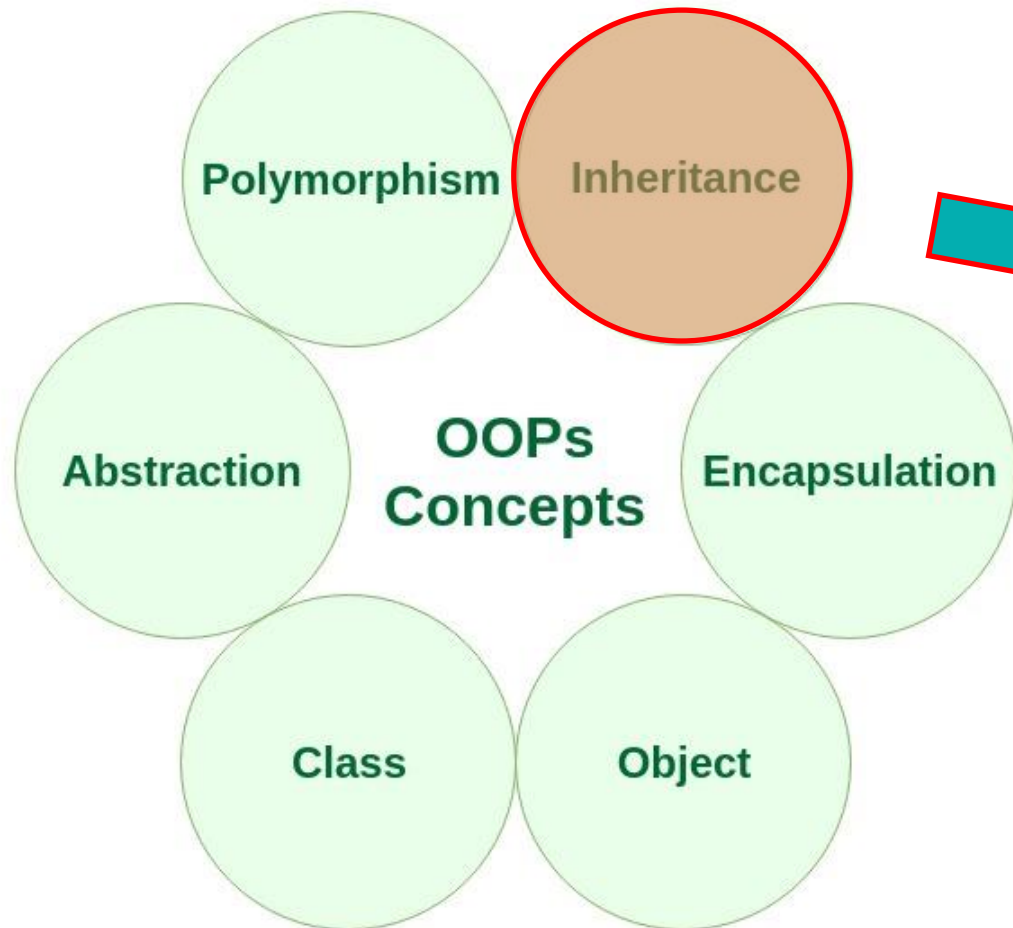
Concepto de Herencia y como programarla...



Explicación



Herencia



Herencia

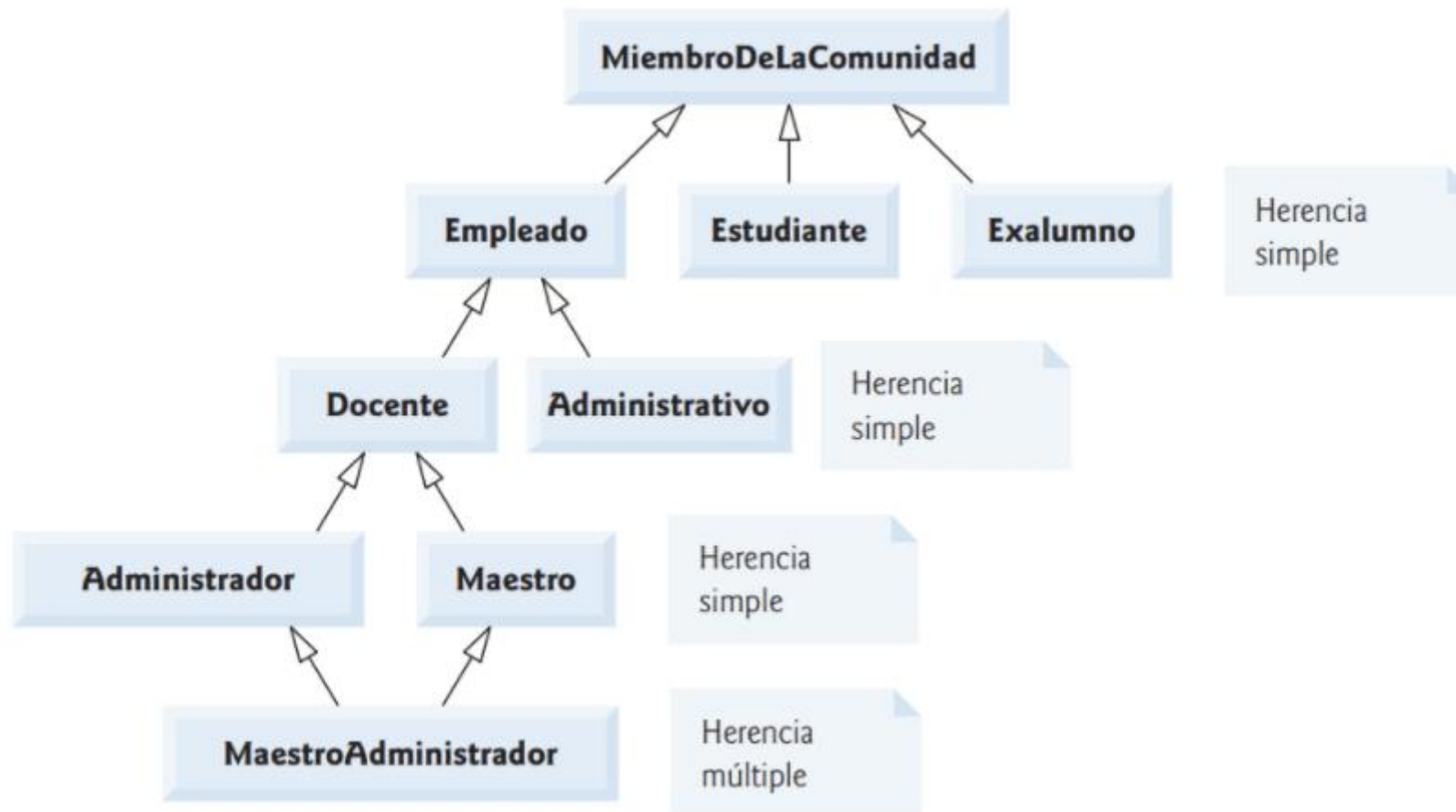
- Se da cuando una nueva clase deriva de otra, “absorbiendo” los atributos y comportamientos de la clase base.
- Una forma de **reutilización** de software.
- Clase **base** y clase **derivada**.
- La clase derivada representa una **especialización** de la clase base.
- Relación “es-un”
- En la herencia de tipo **public**, todo objeto de una clase derivada es también un objeto de la clase base de esa clase derivada

Ejemplos de uso de herencia

Clase base	Clases derivadas
Estudiante	EstudianteGraduado, EstudianteNoGraduado
Figura	Circulo, Triangulo, Rectangulo, Esfera, Cubo
Prestamo	PrestamoAuto, PrestamoMejoraHogar, PrestamoHipotecario
Empleado	Docente, Administrativo
CuentaBanco	CuentaCheques, CuentaAhorros

Fig. 11.1 | Ejemplos de herencia.

Herencia simple y múltiple



Tipos de acceso en herencia

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

Tipos de herencia

Base class member access specifier	Type of Inheritance		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

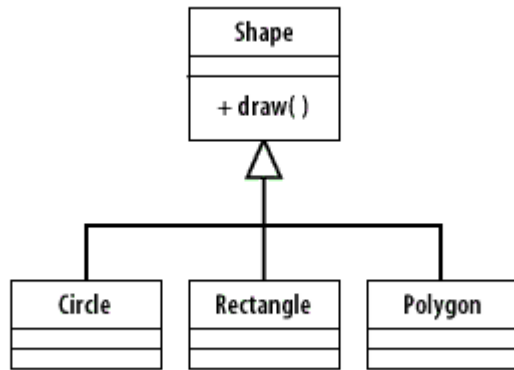
¿Qué no se hereda en C++?

- Constructores/Destructores.
- Operadores sobrecargados.
- Funciones amigas.

Herencia - UML

La notación **UML** permite representar este concepto.

Para este caso, lo que nos dice este diagrama es:



La figura **padre** se llama Shape y contiene un método público (símbolo +) llamado draw().

Las figuras hijas se llaman Circle, Rectangle y Polygon.

Notar que, por definición, las clases Hijas ya tienen el método draw (ya que fue definido en la clase padre) por lo que no es necesario agregarlo.

Herencia – Representación en código

Para dejar claro el concepto de herencia el código que se presenta asume lo siguiente:

- La clase **Shape** contiene los atributos “**X** , **Y**” que representan el origen de una figura (para un círculo es el centro, para cualquier otra figura es su esquina superior derecha).
- La clase **Circle** tiene el atributo “radio”.
- La clase **Rectangle** tiene los atributos “Alto y Ancho”.
- La clase **Polygon** tiene el atributo “NumeroDeLados”.

Para efectos demostrativos, el método draw devuelve un string “soy una figura” sin importar si es Rectangle, Circle o Shape.

Código de Shape (.h)

```
*Shape.h  X  Shape.cpp  X  *main.cpp  X  Cirde.h  X  Cirde.cpp  X

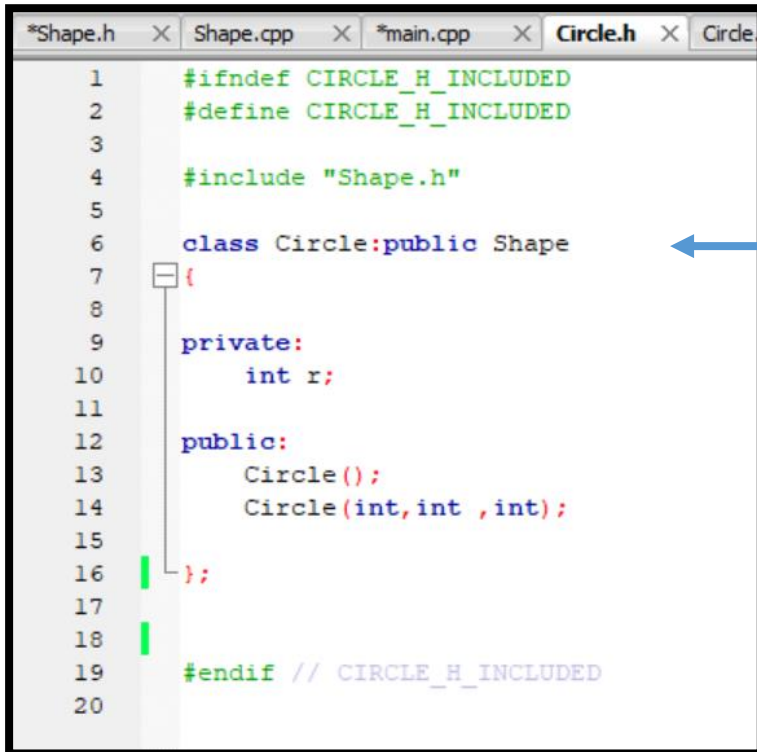
1  #ifndef SHAPE_H_INCLUDED
2  #define SHAPE_H_INCLUDED
3
4  #include <iostream>
5
6  using namespace std;
7
8  class Shape
9  {
10 private:
11     int x;
12     int y;
13
14 public:
15     Shape();
16     Shape(int, int);
17     string draw();
18
19
20     // los métodos debajo se usarán para imprimir los valores
21     // de x y de y , NO ES LA MEJOR SOLUCION
22     // pero se mejorara más adelante
23     int getValueX();
24     int getValueY();
25 };
26
27 #endif // SHAPE_H_INCLUDED
28
```


Código de Shape (.cpp)

```
*Shape.h  X  Shape.cpp  X  *main.cpp  X  Cirde.h  X  Cirde.cpp  X

1      #include "Shape.h"
2
3      Shape::Shape ()
4      {
5          x = 0;
6          y = 0;
7      }
8
9      Shape::Shape (int valX, int valY)
10     {
11         x = valX;
12         y = valY;
13     }
14
15     string Shape::draw()
16     {
17         return "soy una figura" |;
18     }
19
20     int Shape::getValueX()
21     {
22         return x;
23     }
24
25     int Shape::getValueY()
26     {
27         return y;
28     }
29
```

Código de Circle (.h)



```
1  #ifndef CIRCLE_H_INCLUDED
2  #define CIRCLE_H_INCLUDED
3
4  #include "Shape.h"
5
6  class Circle:public Shape
7  {
8
9  private:
10     int r;
11
12  public:
13     Circle();
14     Circle(int,int ,int);
15
16 };
17
18
19 #endif // CIRCLE_H_INCLUDED
20
```

La notación para heredar es poner “:” después de la declaración de la clase, el modificador de acceso (private, protected, public) y la clase de la que hereda.

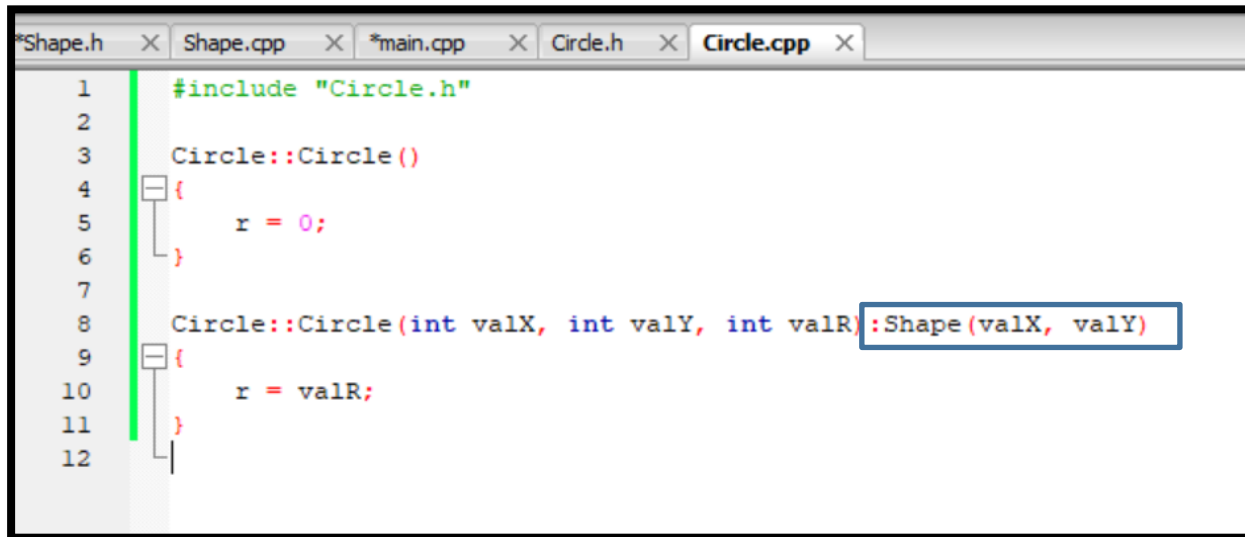
En este caso **Circle** hereda **todo** lo que se definió en **Shape**, esto significa que, tiene “X, Y”, los constructores y el método draw.

Esto incluye lo que se definió en el archivo **Shape.cpp**

Quieres saber más?

<https://www.learncpp.com/cpp-tutorial/115-inheritance-and-access-specifiers/>

Código de Circle (.cpp)



```
1  #include "Circle.h"
2
3  Circle::Circle()
4  {
5      r = 0;
6  }
7
8  Circle::Circle(int valX, int valY, int valR):Shape(valX, valY)
9  {
10     r = valR;
11 }
12
```

Para el constructor con parámetros, hay que llamar al constructor del padre para inicializar los valores.

Notar que hay menos métodos, ya que en la clase Shape se definieron.

Código de main (.cpp)

```
Shape.h x Shape.cpp x main.cpp x Circle.h x Circle.cpp x
1  #include <iostream>
2  #include "Shape.h"
3  #include "Circle.h"
4
5  using namespace std;
6
7  int main()
8  {
9
10     Shape figural(1,2);
11
12     cout << figural.draw() << " con valor x en : " << figural.getValueX() << " y valor y en : " << figural.getValueY() << endl;
13
14     Circle circulo1(2,3,5);
15
16     cout << circulo1.draw() << " con valor x en : " << circulo1.getValueX() << " y valor y en : " << circulo1.getValueY() << endl;
17
18     return 0;
19
20 }
21
```

En este ejemplo se crean dos figuras, un **Shape** y un **Circle**, notar como circulo1 que es de tipo Circle, puede usar **draw** sin problemas.

Práctica

- Definir las clases **Rectangle** y **Polygon**, ambas heredan de Shape.



Breakout rooms



Actividad en Vocareum

Actividad supervisada de aprendizaje

Clase CtaBanc y CtaComision

Clase Empleado Vendedor

Actividad en Vocareum

Tarea

Comida, Hamburguesa y Pizza



Quiz



Quiz: Herencia

- Realizar Quiz rápido del tema de Herencia para la siguiente sesión.

Cierre y avisos

- Leer la situación problema, ¿puedes de manera preliminar identificar algunas clases que puedan estar involucradas en la resolución de la situación problema? Ahora que ya conoces herencia, ¿dónde se podría aplicar?