

HERENCIA

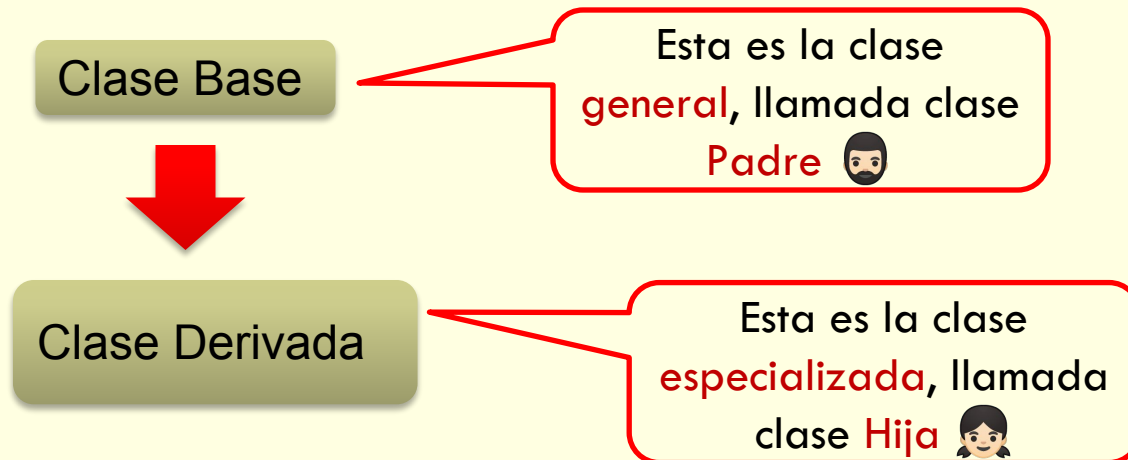


Por: Yolanda Martínez Treviño
Ma. Guadalupe Roque Díaz de León

Herencia

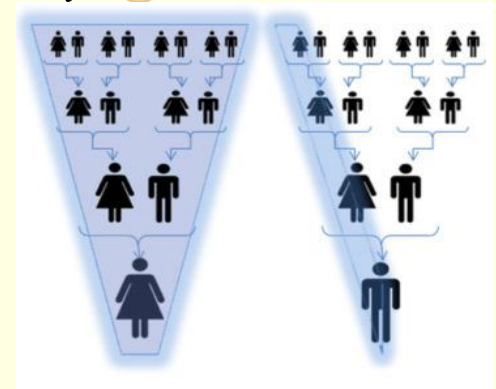


- La herencia es un concepto que permite definir una clase general para después definir otras clases más especializadas, agregando detalles a la declaración de cada una de estas clases especializadas.
- Computacionalmente, la herencia es el proceso mediante el cual se crea una clase nueva, llamada **clase derivada** o **clase hija**, a partir de otra clase, llamada **clase base** o **clase padre**.




Herencia

- Las clases derivadas heredan todas las propiedades de la clase base; es decir, heredan sus atributos (datos) y sus métodos (funciones).
- Al diseñar La clase derivada - hija 🧒, solo codificas las diferencias o especialización de código
- Se puede diseñar una jerarquía de clases, derivando clases de las clases hijas (es decir, las clases hijas pueden a su vez ser padres de otras clases) 👵



Ejemplo de Herencia

- Se tiene la clase general **Persona** :
 - **atributos** - nombre y edad;
 - métodos de acceso, modificadores y un método `str()`.
- La clase **Estudiante**:
 - **atributos** - nombre, edad, carrera.
- La clase **Maestro**:
 - **Atributos** - nombre, edad, departamento.
-  se puede utilizar la herencia!! donde se hereda el nombre, la edad y métodos de la clase padre, en la clase derivada se agregan los elementos que diferencian a los objetos de esta clase derivada.



Ejemplo de Herencia:

Clase Persona

Clase Base

Clase Estudiante

Clase Maestro

Clases Derivadas



Ejemplo de Herencia:

- **Clase:** Persona
- **Atributos:** -nombre
-edad
- **Métodos:** +setNombre(str)
+setEdad(int)
+getNombre() +getEdad()
+str()

Clase Persona

clase base de la cual
heredarán las clases
derivadas.

Ejemplo de Herencia:

- **Clase:** Estudiante
- **Atributos:** - carrera
- **Métodos:** +setCarrer(str)
 - +getCarrera()
+str()

Clase Persona

Clase Base

Clase Estudiante

Clase Derivada: Estudiante

clase **derivada**, heredará de la clase Persona sus atributos: nombre, edad y métodos :setNombre, setEdad, getNombre, getEdad y str.

Ejemplo de Herencia:

- **Clase:** Maestro
- **Atributos:** departamento
- **Métodos:** void setDep(string)
string getDepto()
string str()

Clase Persona

Clase Base

Clase Maestro





Clase Derivada: Maestro

Clase **derivada** que heredará de la clase Persona sus atributos (nombre y edad) y métodos (setNombre, setEdad, getNombre, getEdad, muestra).

Códificación - Herencia en C++

- Sintaxis de Herencia:

class ClaseDerivada : public ClaseBasePadre

-  La clase derivada hereda todos los atributos (o datos miembro) y todos los métodos (o funciones miembro) de la claseBasePadre .
- Pero  - la clase derivada solo puede usar los elementos públicos de la clase base;
- Pero  se pueden acceder los atributos privados, de la clase base por medio de los métodos de acceso(get) y modificadores(set).

Modificadores de acceso

- Los atributos y métodos de una clase pueden ser:
 - **public:** Pueden ser usados afuera de la clase.
 - **private:** Sólo se pueden usar por sus funciones miembro y friend, no se heredan.
 - **protected:** pueden ser usados por sus funciones miembro y friend y además por las funciones miembro y friend de sus clases derivadas.

Codificación - Herencia en C++

■ Para nuestro ejemplo tenemos las siguientes declaraciones:

Persona.h

```
class Persona
{public:
    Persona();
    Persona(string, int);
    string getNombre();
    void setNombre(string);
    int getEdad();
    void setEdad(int);
    string str();
    protected:
    string nombre;
    int edad;
};
```

Estudiante.h

```
#include "Persona.h"

class Estudiante : public Persona
{
    public:
        Estudiante();
        Estudiante(string, int, string);
        string getCarrera();
        void setCarrera(string);
        string str();
    private:
        string carrera;
};
```

Indica que es una
clase derivada de la
clase **Persona**

Herencia en C++: clase **Persona**

```
Persona::Persona()
{
    nombre = "Chilindrina";
    edad = 100;
}

Persona::Persona(string nom, int ed)
{
    nombre = nom;
    edad = ed;
}

string Persona::getNombre()
{
    return nombre;
}

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

```
void Persona::setNombre(string nom)
{
    nombre = nom;
}

void Persona::setEdad(int ed)
{
    edad = ed;
}

string Persona::str()
{
    return Nombre: "+" nombre + " edad: " +
    tostrign(edad);
}
```

Herencia en C++: Clase Estudiante

```
Estudiante::Estudiante() : Persona()
```

```
{  
    carrera = "N/A";  
}
```

Para llamar al constructor default de la clase base.

```
Estudiante::Estudiante(string nom, int ed, string ca) : Persona(nom, ed)
```

```
{  
    carrera = ca;  
}
```

Para llamar al constructor con parámetros de la clase base.

```
string Estudiante::getCarrera()  
{  
    return carrera;  
}
```

```
void Estudiante::setCarrera(string ca)  
{  
    carrera = ca;  
}
```

```
string Estudiante::str()  
{  
    cout<<"Nombre: "<<nombre<<" edad: "<<edad<<" Carrera:  
"<<carrera;  
}
```

Nota que uso directamente los atributos heredados de la clase base.

Herencia en C++: Aplicación








En el programa se eliminaron los `cout<<endl;` para que cupiera en el filmina.

```
#include "Estudiante.h"
int main()
{
    Estudiante chabelo("Chabelo", 125, "Dr."), chilindrina,
    Persona chano("Chano", 80), chonita;
    cout<<"Los datos del estudiante 1 son: "<<endl;
    chabelo.str();
    cout<<"Los datos del estudiante 2 son: "<<endl;
    cout<<"Nombre "<<chilindrina.getNombre()<<" Edad: "
    << chilindrina.getEdad()<<" Carrera "<< chilindrina.getCarrera();
    cout<<"Los datos de la persona 1 son: "<<endl;
    chano.muestraDatos();
    cout<<"Los datos de la persona 2 son: "<<endl;
    cout<<"Nombre "<<chonita.getNombre()<<" Edad: "
    "<<chonita.getEdad();
    return 0;
}
```

Algunos detalles importantes: `protected`

- Para que la clase derivada utilice directamente los datos miembros de la clase base, estos se deben declarar como `protected`.
- También es posible utilizar los datos en la clase base de tipo privado y usarlos en las clases derivadas a través de sus métodos de `acceso(get)` y `modificadores(set)`.
- En el ejemplo se uso el modificador `protected`, pero se podría usar `private`.

Algunos detalles importantes: constructor de la clase derivada

- El constructor de la clase derivada- hija  debe llamar al constructor de su clase base .
- Si el constructor no incluye dicha llamada  , C++ hace una llamada implícita al constructor default de la clase base(Padre). 
 - Si la clase base(Padre) no incluye un constructor default, el compilador marcará un error  

Refefinir un método

- Nota que en la clase **Persona** existe el método **str** y en la clase **Estudiante** también existe el método **str**.
- En la implementación de la clase **Estudiante**, se está redefiniendo el método **str**.

Refefinir un método

- Para redefinir un método, una clase derivada define de nuevo el método con exactamente el mismo encabezado que éste tiene en la clase base; con esto se consigue que un objeto de la clase derivada tenga su propia versión de dicho método.
- En el ejemplo un objeto de la clase **Estudiante** no puede utilizar el método **str** de la clase **Persona** porque solamente mostraría nombre y edad y no mostraría la carrera; entonces es necesario redefinir el método para adaptarlo a los cambios que tiene la clase derivada con respecto a la clase base.

Refefinir un método

- Para redefinir un método, éste debe tener el **mismo nombre y lista de parámetros** que el método al que sustituye.
- Si no se cumple este requisito, se heredaría el método de la clase base y la clase derivada tendría otro método con el mismo nombre.
- Esto no es redefinición sino **sobrecarga** de nombres.