

HERENCIA

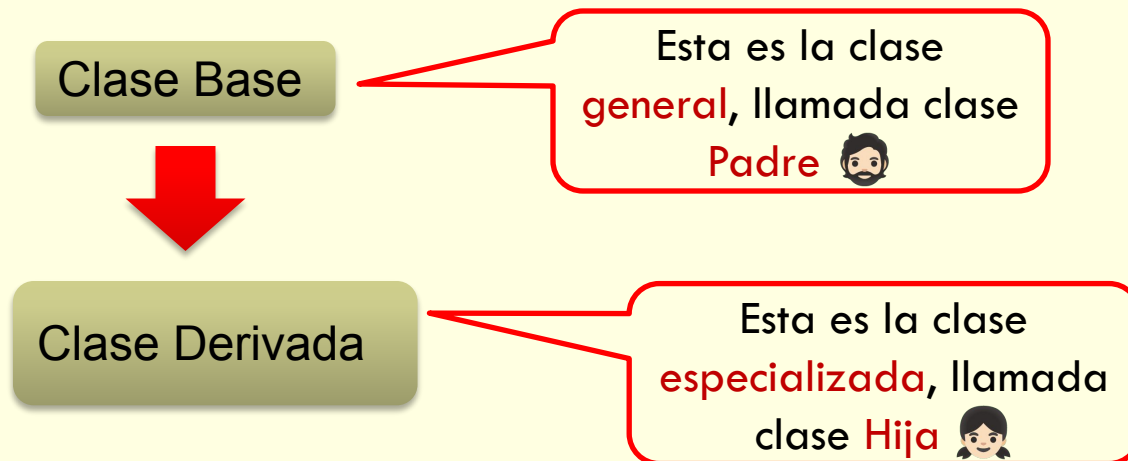


Por: Yolanda Martínez Treviño
Ma. Guadalupe Roque Díaz de León

Herencia

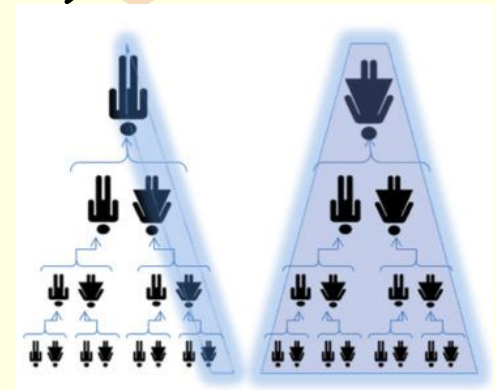


- La herencia es un concepto que permite definir una clase general y posteriormente definir otras clases más especializadas.
- Computacionalmente, la herencia es el proceso mediante el cual se crea una clase nueva, llamada **clase derivada** o **clase hija**, a partir de otra clase, llamada **clase base** 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 se van a codificar las diferencias - **especialización**.
- 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, edad
 - métodos de acceso (**getters**), modificadores (**setters**) y un método **str()**.
- La clase **Estudiante**:
 - **atributos** - nombre, edad, carrera.
- La clase **Maestro**:
 - **Atributos** - nombre, edad, departamento.
- 🧠 se puede utilizar la herencia!! heredando el nombre, la edad y métodos de la clase padre, en la clase derivada se agregan los elementos que la diferencian de la clase padre.



Ejemplo de Herencia:

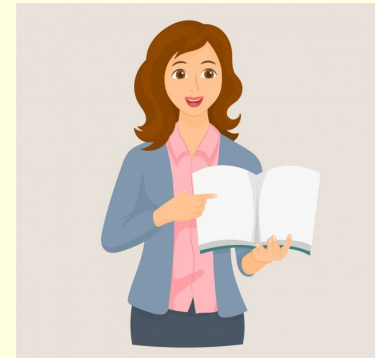
Clase Persona

Clase Base

Clase Estudiante

Clase Maestro

Clases Derivadas



Ejemplo de Herencia:

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

Clase Persona

clase base de la cual
heredarán las clases
derivadas.

Ejemplo de Herencia:

- **Clase:** Estudiante
- **Atributos:** - carrera
- **Métodos:**
 - +void **setCarrera**(str _carrera)
 - +string **getCarrera**()
 - +string **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 setDepartamento(string)`
`string getDepartamento()`
`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, str).

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 sólo 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();
    int getEdad();
    void setNombre(string);
    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 _nombre, int _edad)
{
    nombre = _nombre;
    edad = _edad;
}

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

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

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

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

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

Herencia en C++: Clase Estudiante

```
Estudiante::Estudiante( ) : Persona( )  
{  
    carrera = "Chef";  
}
```

Para llamar al constructor default de la clase base.

```
Estudiante::Estudiante(string nombre, int edad, string ca) : Persona(nombre, edad)  
{  
    carrera = ca;  
}
```

Para llamar al constructor con parámetros de la clase base.

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

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

Observa 👁️: se usa directamente los atributos heredados de la clase base.

```
string Estudiante::str()  
{  
    return "Nombre: " + nombre + " edad: " + to_string(edad) + " Carrera: "+carrera;  
}
```



Herencia en C++: Aplicación



dudas ?

```
#include "Estudiante.h"
int main()
{
    Estudiante chabelo("Chabelo", 125, "Medico"), chilindrina;
    Persona chano("Chano", 80), chonita;
    cout << "Los datos del estudiante 1 son: " << chabelo.str() << endl;

    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: " << chano.str() << endl;

    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 pueda usar los atributos de la clase base, deben ser definidos como **protected**.
- Si son **private**, las clases derivadas pueden usar los atributos con los métodos de **acceso(get)** y los **modificadores(set)**.
- **Recuerda** 🖋️ - 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). 🧑
 - Pero, si la clase base(Padre) no incluye un constructor default, el compilador marcará un error 🤒 🏠

Refefinir un método

- Observa que el método `str()` está en la clase `Persona` y la clase `Estudiante`.
- En la clase `Estudiante`, se está `redefiniendo` el método `str()`.
- Para redefinir un método, se tiene que poner exactamente el mismo encabezado que éste tiene en la clase base;
- Si no se cumple ese requisito, se hereda el método de la clase base y la clase derivada tendría otro método con el mismo nombre.
- La clase derivada tendrá su versión propia de `str()`.

Refefinir un método

- En 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.
- Esto no es redefinición sino **sobrecarga** de nombres.

Ejercicio - Clase Medico

Crea una clase derivada de la clase Persona,

Nombre de la Clase **Medico**

Atributos:

- nombre, edad, especialidad, pacientesCovidAtendidos

Métodos:

- Constructor default y con parámetros
- Métodos de acceso(get)
- Métodos modificadores(set)
- **str()**

En la aplicación

- Añade 2 objetos de la clase **Medico** - uno con el constructor con parámetros y otro con el default.
- Despliega la información de los 2 objetos de la clase Medico usando el método **str()** y del otro médico usando los métodos (get)

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;
};
```

Medico.h

```
#include "Persona.h"
class Medico : public Persona
{
public:
    Medico();
    Medico(string, int, string, int);
    string getEspecialidad();
    int getPacientes();
    void setEspecialidad(string);
    void setPacientes(int);
    string str();
private:
    string especialidad;
    int pacientes;
};
```

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

Herencia en C++: Clase Medico

```
Medico::Medico( ) : Persona( )
{
    especialidad = "Infectologo";
    pacientes = 645;
}
// Constructor con parámetros
Medico::Medico(string _nombre, int _edad, string _especialidad, int _pacientes) : Persona(_nombre,
_edad)
{ // de pref. no usar - prefijo this-> para diferenciar el nombre del atributo al nombre de parámetro
    especialidad = _especialidad;
    pacientes = _pacientes;
}
string Medico::getEspecialidad()
{
    return especialidad;
}
int Medico::getPacientes()
{
    return pacientes;
}

string Medico::str( )
{
    return "Nombre: " + nombre + " edad: " + to_string(edad) + " Especialidad: "+especialidad + "
Pacientes: " + to_string(pacientes);
}
```

Llamar al constructor default de la clase base.

Llamar al constructor con parámetros de la clase base.

Observa 👁️: se usan directamente los atributos heredados de la clase base.

