Laboratorio

Herencia

1. Competencias

1.1. Competencias del curso

Conoce, comprende e implementa programas usando funciones del lenguaje de programación C++.

1.2. Competencia del laboratorio

Conoce, comprende e implementa programas usando funciones del lenguaje de programación C++.

2. Equipos y Materiales

- Un computador.
- IDE para C++.
- Compilador para C++.

3. Marco Teórico

3.1. Introducción

En algunos programas muchas veces necesitamos hacer pequeñas modificaciones a clases para generar una solución a un problema nuevo pero similar. Una forma de lograrlo es copiar y pegar el código de una clase existente en otra definición de clase y hacer las modificaciones necesarias.

Sin embargo, realizar esta actividad puede ser muy tediosa y desencadenar una serie de problemas:

- Un bug en la clase original que no fue solucionada a tiempo, podrá generar nuevos errores en la funcionalidad de nuestra nueva clase.
- Hacer este enfoque de copiar y pegar, no generara una relación entre las clases, simplemente serán consideradas como porciones de código completamente diferentes con una carga de trabajo bastante similar

- En caso de que se realice un cambio en la clase base que fue copiada, debido a que el autor mejoro su rendimiento, se tendrá que copiar o ubicar los elementos nuevos y copiarlos nuevamente.

Es por este motivo que C++ pone a disposición un mecanismo pensado en este tipo de situaciones, denominado herencia.

Cuando heredamos un contenido de una clase en una nueva definición, automáticamente el contenido será compartido en la nueva clase declarada. Se compartirán automáticamente todos los miembros de la clase, y los métodos de la clase a la cual se está heredando el contenido.

Por ejemplo, si disponemos de una clase superior:

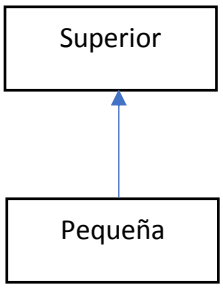
```
class Superior {
        public:
        Superior();
        void método1( );
        protected:
        void metodo2();

        private:
        int índice;
}
```

Si necesitamos una clase adicional que herede esta estructura, tendremos que indicar a C++ que necesitamos heredar parte del contenido de esta clase superior en otra clase.

```
class Pequena : public Superior {

        public:
        Superior (); //llamado al constructor superior
        void metodo3();
}
```

En la primera línea, se ha indicado mediante los dos puntos (:), que el contenido de la clase Superior es parte de la clase pequeña .Por lo tanto, toda modificación que se realizara en la implementación de la clase superior, esta se reflejaría de forma automática en la clase pequeña.

Los métodos y datos miembros, también serán heredados en la clase Pequeña, permitiendo acceder a estos elementos de la clase superior. El diagrama de clases del comportamiento de las clases se muestra a continuación:



Desde el código del cliente, se puede realizar una invocación a los métodos de la siguiente forma:

Pequena subclase;

subclase.metodo1();

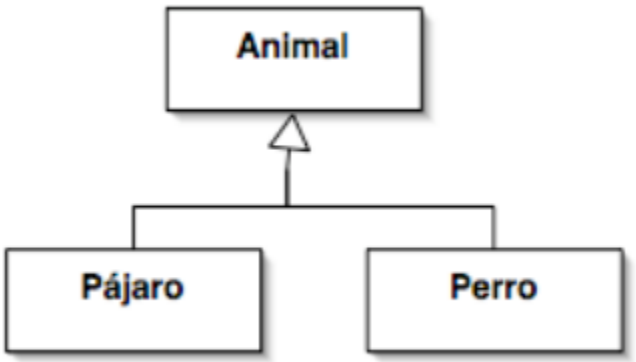
subclase.metodo3();

Ambas invocaciones son correctas, debido a que la clase que hereda contenido tiene las definiciones de la clase Superior en forma interna, cuando se hereda solo de una clase se llama herencia simple.

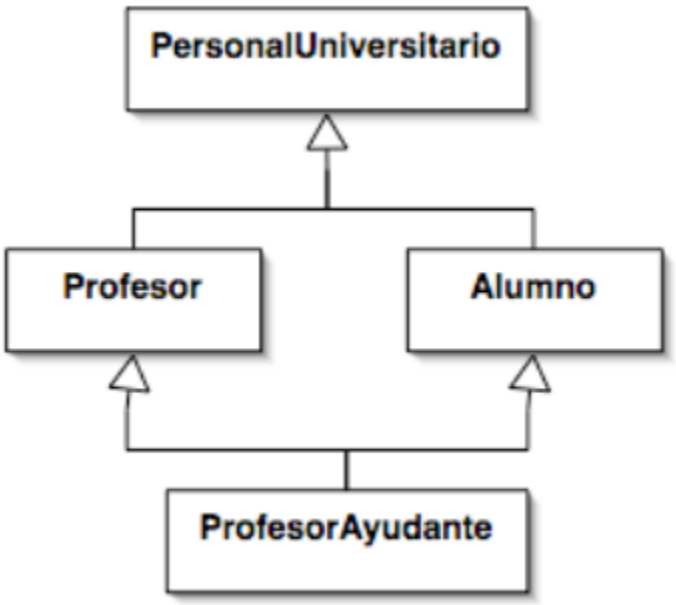
En el caso de C++ podemos heredar contenido de diferentes clases al mismo tiempo, haciendo que el contenido de una clase sea bastante amplio en unas pocas líneas de código, a esto se llama herencia múltiple:

class Subclase : public clase1, public clase2, …

Herencia simple:



Herencia múltiple:



Veamos un ejemplo:

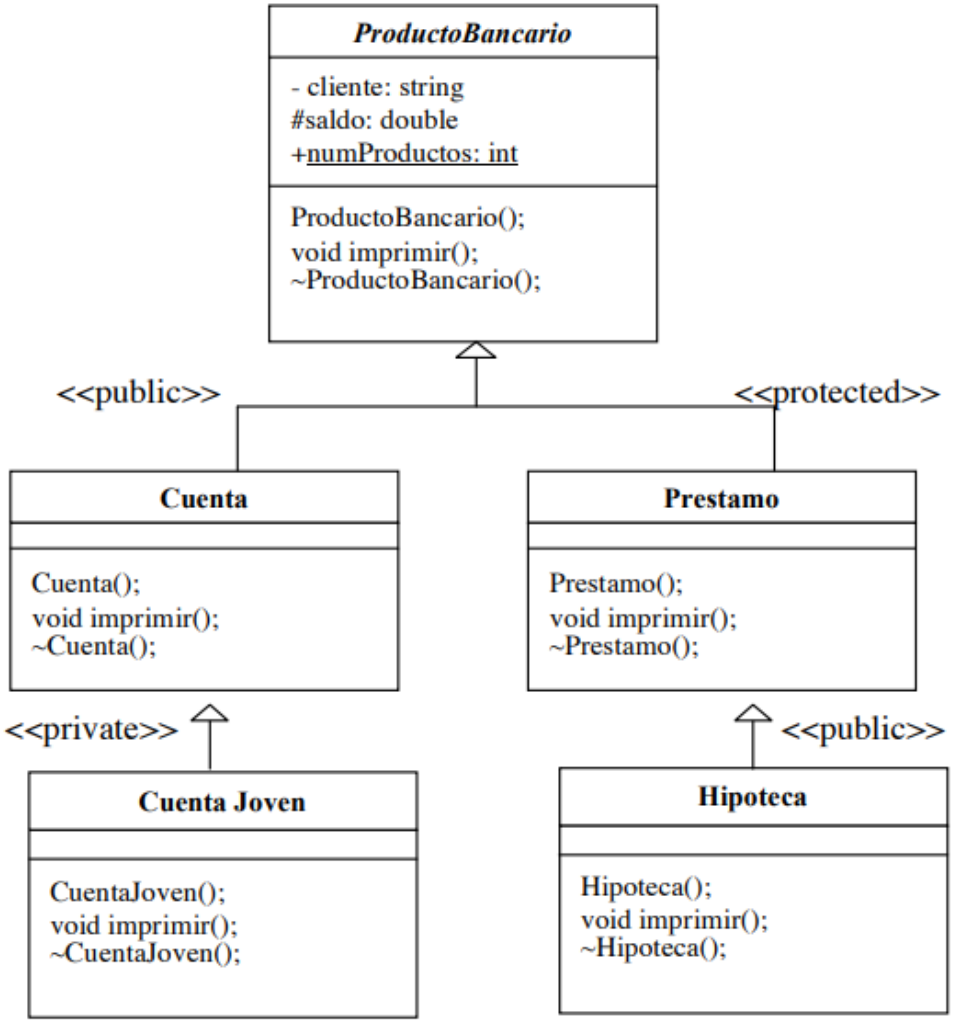
```
#pragma once
// ClaseBase.h
class ClaseBase
{
protected:
	int    valor1 = 10;
	int valor2 = 12;
public:
	ClaseBase();
	~ClaseBase();
	void UnMetodoClaseBase();
};


// ClaseBase.cp
#include "stdafx.h"
#include "ClaseBase.h"
#include <iostream>
using namespace std;
ClaseBase::ClaseBase()
{
}
ClaseBase::~ClaseBase()
{
}
```

```
void ClaseBase::UnMetodoClaseBase() {
	int rpta = valor1 + valor2;
	cout << "Ingreso a Metodo de Clase Base y realizó la suma; la respuesta
es: " <<rpta << endl;
}


// ClaseDerivada.h
#pragma once
#include "ClaseBase.h"
class ClaseDerivada : public ClaseBase
{
private:
	int valor3;
	int valor4;
public:
	ClaseDerivada(int,int);
	~ClaseDerivada();
	void UnMetodoClaseDerivada();
};


// ClaseDerivada.cpp
#include "stdafx.h"
#include "ClaseDerivada.h"
#include <iostream>
using namespace std;
ClaseDerivada::ClaseDerivada(int _valor3, int _valor4)
{
	valor3 = _valor3;
	valor4 = _valor4;
}

ClaseDerivada::~ClaseDerivada()
{
}
void ClaseDerivada::UnMetodoClaseDerivada()
{
	int rpta = valor3 + valor4 + valor1 + valor2;
	cout << "Ingreso a Metodo de Clase Derivada y la suma de 2 valores y la
respuesta es: " << rpta << endl;
}


// main

#include "stdafx.h"
#include "ClaseDerivada.h"
#include <iostream>
void main()
{
	ClaseDerivada obj1 = ClaseDerivada(3,4);
	obj1.UnMetodoClaseDerivada();
	obj1.UnMetodoClaseBase();
	system("pause");
}
```

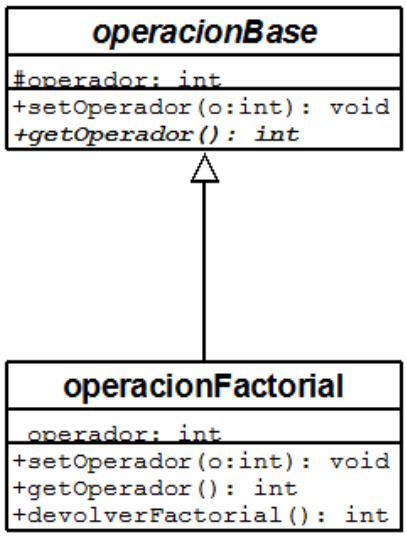
4. Ejercicios

Resolver los siguientes ejercicios planteados:

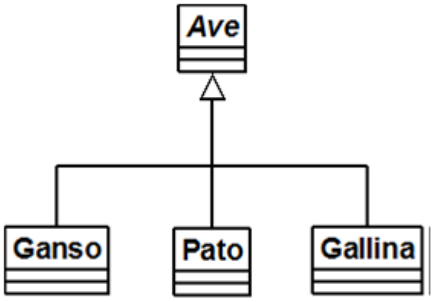
1. Crear una clase Persona del cual tendrá métodos asignar una edad y nombre. Una segunda clase, alumno, tendrá que heredar este contenido y a través de esta clase poder asignar los datos de edad y nombre de los estudiantes.
2. Crear una clase Color que mantenga 3 valores (RGB). Una segunda clase Material, tendrá como información una variable de texto que describa algún material (Ejemplo: madera, vidrio, platico, etc.) Una tercera clase, Objetos, deberá de heredar contenido de ambas clases con la finalidad de describir diferentes objetos en cuanto a color y el material. (Ejemplo: mesa de color café y material de plástico)
3. Dada la siguiente jerarquía de herencia, indica la visibilidad de los atributos de la clase ProductoBancario en las clases CuentaJoven e Hipoteca.



4. Escribe una clase de nombre ClaseDisco, que herede de la clase ClaseMultimedia los atributos y métodos definidos por usted. La clase ClaseDisco tiene, aparte de los elementos heredados, un atributo más también definido por usted. Al momento de imprimir la información debe mostrase por pantalla toda la información.
5. Escribe un programa que implemente la siguiente jerarquía de clases



6. Escribe un programa que implemente la siguiente jerarquía de clases, Debe implementar aquellos atributos y métodos necesarios para que se pueda ejecutar el siguiente programa:



5. Entregables

Al final estudiante deberá:

1. Compactar el código elaborado y subirlo al aula virtual de trabajo. Agregue sus datos personales como comentario en cada archivo de código elaborado.
2. Elaborar un documento que incluya tanto el código como capturas de pantalla de la ejecución del programa. Este documento debe de estar en formato PDF.
3. El nombre del archivo (comprimido como el documento PDF), será su LAB07_GRUPO_A/B/C_CUI_1erNOMBRE_1erAPELLIDO.

   (Ejemplo: LAB07_GRUPO_A _2022123_PEDRO_VASQUEZ).
4. Debe remitir el documento ejecutable con el siguiente formato:

   LAB07_GRUPO_A/B/C_CUI_ EJECUTABLE_1erNOMBRE_1erAPELLIDO

   (Ejemplo: LAB07_GRUPO_A_EJECUTABLE_2022123_PEDRO_VASQUEZ).

En caso de encontrarse trabajos similares, los alumnos involucrados no tendrán evaluación y serán sujetos a sanción.