TP N°13 - Colaboracion y Herencia

A-1. La herencia de clases es cuando una clase deriva de otra recibiendo los metodos y propiedades de la clase original, la herencia es muy util debido a que facilita la creacion de clases similares.

2. El encapsulamiento es sumamente importante en el contexto de la herencia debido a que permite que los metodos y propiedades de una clase sean recibidos solo por una clase hija.

3. Las propiedades publicas pueden ser modificadas y accedidas por cualquiera, como clases hijas (las cuales las heredan), o una linea de codigo fuera de la clase.

Las propiedades protegidas solo pueden ser accedidas por una clase hija, que hereda estas mismas.

Las propiedades privadas solo se pueden acceder desde la propia clase base y no se heredan.

4. La especializacion es en el objetivo con el que fue creada una subclase, se especifica en una accion determinada en base a otra clase original.

5. El problema de la herencia multiple es que puede hacer un poco ambiguo el codigo, ademas de que si se heredan dos metodos de dos clases con la misma firma, pueden surgir errores.

B-1. El programa genera multiples clases, una clase "persona" que es la original y tiene "nombre" y "edad", las clases "empleado" y "estudiante", que se derivan de esta, agregando "sueldo" y "notafinal" respectivamente, a su vez, una subclase "tecnico", derivada de "estudiante", agrega la "especialidad". El codigo inicializa los distintos tipos de persona y muestra, dependiendo del tipo de persona, su nombre, edad, sueldo, nota final, o especialidad.

2. #include<iostream>

#include<stdlib.h>

using namespace std;

class Persona{

private: //atributos privados

string nombre;

int edad;

public: //metodos publicos

Persona(string,int); //constructor que inicializa el nombre y la edad

void mostrarPersona();

};

class Empleado : public Persona{

private: //atributo privado de la subclase

float sueldo;

public: //metodos publicos

Empleado(string,int,float); //constructor del nombre, edad y sueldo

void mostrarEmpleado();

};

class Estudiante : public Persona{

private: //atributo privado

float notaFinal;

public: //metodos publicos

Estudiante(string,int,float);

void mostrarEstudiante();

};

class Tecnico : public Estudiante{

private:// atributo privado

string especialidad;

public:

Tecnico(string,int,float,string); //constructor que inicializa nombre, edad, nota final y especialidad

void mostrarTecnico();

};

// estructura de los metodos de la clase persona

Persona::Persona(string \_nombre,int \_edad){

nombre = \_nombre;

edad = \_edad;

}

void Persona::mostrarPersona(){

cout<<"Nombre: "<<nombre<<endl;

cout<<"Edad: "<<edad<<endl;

}

// estructura de los metodos de la clase empleado

Empleado::Empleado(string \_nombre,int \_edad,float \_sueldo) : Persona(\_nombre,\_edad){

sueldo = \_sueldo;

}

void Empleado::mostrarEmpleado(){

mostrarPersona();

cout<<"Sueldo: "<<sueldo<<endl;

}

// estructura de los metodos de la clase estudiante

Estudiante::Estudiante(string \_nombre,int \_edad,float \_notaFinal) : Persona(\_nombre,\_edad){

notaFinal = \_notaFinal;

}

void Estudiante::mostrarEstudiante(){

mostrarPersona();

cout<<"Nota Final: "<<notaFinal<<endl;

}

// estructura de los metodos de la clase tecnico

Tecnico::Tecnico(string \_nombre,int \_edad,float \_notaFinal,string \_especialidad) : Estudiante(\_nombre,\_edad,\_notaFinal){

especialidad = \_especialidad;

}

void Tecnico::mostrarTecnico(){

mostrarEstudiante();

cout<<"Especialidad: "<<especialidad<<endl;

}

int main(){

Empleado empleado1("Juan",35,1300000);

cout<<"-Empleado-"<<endl;

empleado1.mostrarEmpleado();

cout<<"\n";

Estudiante estudiante1("Maria",21,16.7);

cout<<"-Estudiante-"<<endl;

estudiante1.mostrarEstudiante();

cout<<"\n";

Tecnico tecnico1("Alejandro",20,15.6,"Informatica");

cout<<"-Tecnico-"<<endl;

tecnico1.mostrarTecnico();

cout<<"\n";

return 0;

}

3. El nombre de la clase padre es Persona.

4. Las clases Empleado y Estudiante heredan de la clase Persona.

5. La relacion entre Estudiante y Tecnico es que Tecnico es una subclase de Estudiante.

6. El proposito del constructor de Empleado es inicializar el sueldo del empleado ademas de las anteriores propiedades.

7. El objetivo de mostrarPersona() es literalmente, mostrar los datos (las propiedades) de la persona; el nombre y la edad.

8. Para mostrar el sueldo del empleado se usa el metodo mostrarEmpleado().