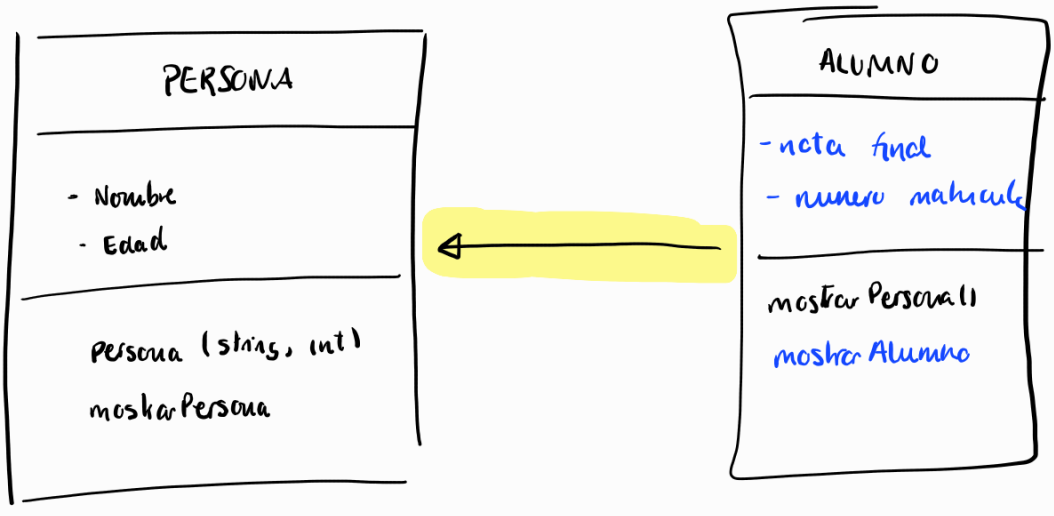
1: HERENCIA, CLASES PADRE E HIJA

Lo veré directamente sobre el siguiente EJEMPLO: PERSONA Y ALUMNO



Cuando represento gráficamente la relación entre clases, las hijas se unen a la padre con una flecha de punta vacía.

Viendo el código:

1: CABECERA.H

```
1  #pragma once
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <iostream>
6
7  using namespace std;
```

(Listado de librerías y formatos usados. Este archivo es reciclado y reciclable)

2: PERSONA.H (definición de la clase Persona)

```
#pragma once
#include "cabecera.h"

class Persona{

        private:
                string nombre;
                int edad;

        public:
                Persona(string, int); // Constructor.
                ~Persona(); // Destructor.
                void mostrarPersona(); //Método para mostrar nombre y edad.
};
```

Hasta aquí, es una clase normal y corriente.

voy a hacerlo así, aunque debería hacerlo con getters y setters.

3: PERSONA.CPP → Métodos y constructor de la clase Persona:

```
#include "Persona.h"

Persona::Persona(string n, int e) {
        nombre = n;
        edad = e;
}

Persona::~Persona() {
}

void Persona::mostrarPersona() {
        cout << "la persona se llama: " << nombre << endl;
        cout << "y tiene " << edad << "anos\n" << endl;
}
```

constructor

Destructor

El método mostrarPersona enseña el nombre y la edad de la Persona.

4: ALUMNO.H

```
#pragma once
#include "Persona.h"

class Alumno : public Persona {
/*La clase alumno es una clase "hija" de persona. El "public" significa
que puedo acceder a todos los atributos y métodos que sean públicos de la
clase Padre (Persona)*/

        private:
                float notafinal;
                int numero_matricula;

        public:
                Alumno(string, int, float, int);
                /* No solo tiene los métodos propios, sino que tiene también los pertenecientes
                a la clase Padre*/
                /*Pero, ¿En qué orden?:
                  Hay que respetar los rangos. Primero van los de la clase padre. Estos eran
                el nombre (string) y la edad (int). Después vienen los propios al alumno,
                  notafinal(float) y el número de matrícula (int)*/

                ~Alumno();
                void mostrarAlumno();

};
```

Esto significa que la Clase Alumno hereda los atributos y métodos públicos (y protegidos) de la clase Persona.

Es decir, desde un objeto Alumno, puedo usar / llamar el metodo mostrarPersona().

de clase Persona: nombre, edad — de clase Alumno: notafinal, numero_matricula

5. Alumno.CPP → Métodos de la clase Alumno

```
#include "Alumno.h"

Alumno::Alumno(string nomb, int _edad, float _nota, int _nummat):Persona (nomb,_edad){
        nummat = _nummat;
        notafinal = _nota;
}
/*Declaro todos los atributos que necesita el constructor, pero el nombre son
pertenecientes a la clase persona. Lo indico al final.*/

Alumno::~Alumno() {
} //El destructor sigue estando vacío.

void Alumno::mostrarAlumno() {
        mostrarPersona(); // Desde un método de Alumno, puedo usar métodos Públicos de Persona.
        /*Tengo acceso al metodo mostrarPersona, perteneciente a la clase
        Persona, porque la clase alumno es de tipo public Persona (tiene
         acceso a los métodos Puúblicos de la clase Persona.*/

        /* La función mostrarPersona() muestra el nombre y edad. Aún me falta
        mostrar los atributos "nota final" y "numero de matrícula" */
        cout << "El numero de matricula del alumno es: " << nummat << endl;
        cout << "Y su nota final es: " << notafinal << endl;

}
```

El constructor necesita todos estos, Pero estos son de Persona.

numero_matricula.

6 : MAIN.CPP

```cpp
#include "Alumno.h"

int main(){

        Alumno alumno1("Jimmy", 21, 8.75, 20259);

        /*Como ya vi en el constructor de alumno, tengo que pasarle :
        nombre, edad, nota final y número de matrícula.
        El nombre y la edad son los atributos de la clase persona. */

        alumno1.mostrarAlumno();

        system("pause");
        return 0;
}
```

de clase Persona

Propios de Alumno.