Programación Orientada a Objetos POO

AED I

2020

UBP UNIVERSIDAD BLAS PASCAL

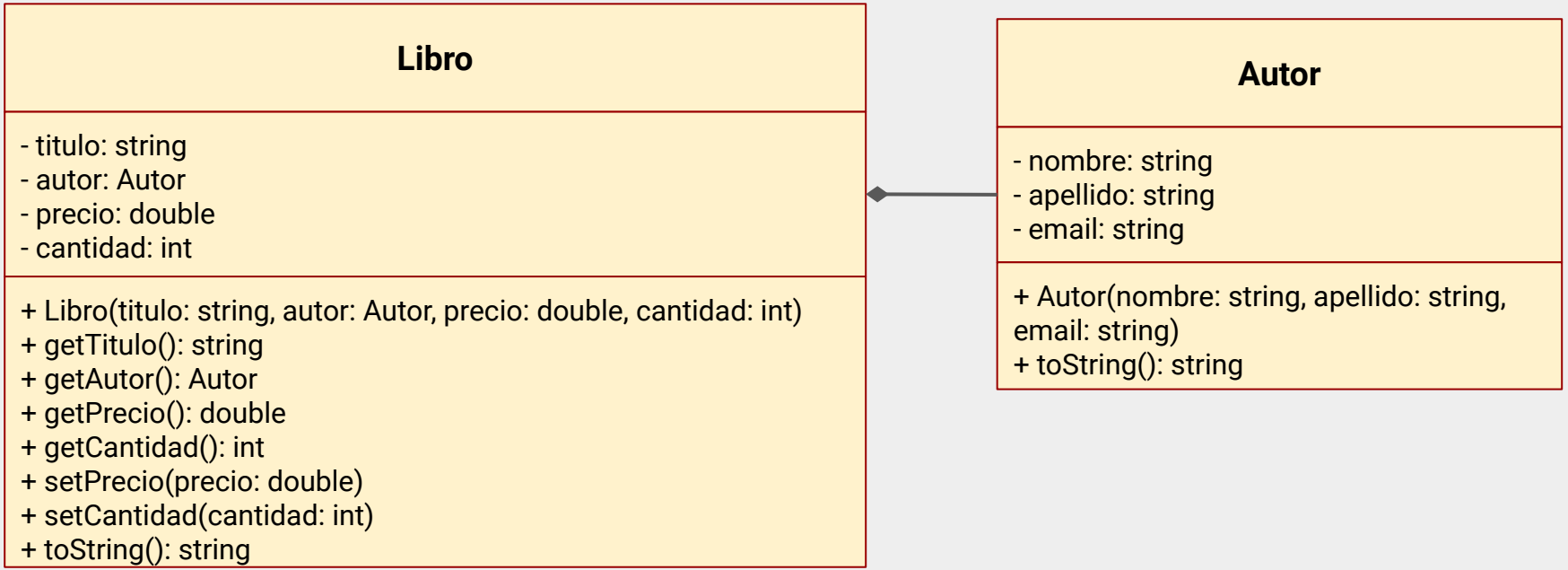
Agregación y Composición

Agregación

Composición

Ejercicio 1.05.

Agregación: Libro - Autor



agregación: clase

```
class Libro{
    private:
      string titulo;
      Autor* autor;
      double precio;
      int cantidad;
    public:
      Libro(string, Autor*, double, int);
      string getTitulo() const;
      Autor* getAutor() const;
      double getPrecio() const;
      int getCantidad() const;
      void setAutor(Autor*);
      string toString();
};
```

agregación: implementación del constructor

```
Libro::Libro(string titulo, Autor* autor, double precio, int cantidad){
      this->titulo = titulo;
      this->autor = autor;
      this->precio = precio;
      this->cantidad = cantidad;
}
```

agregación: pasando la referencia al objeto

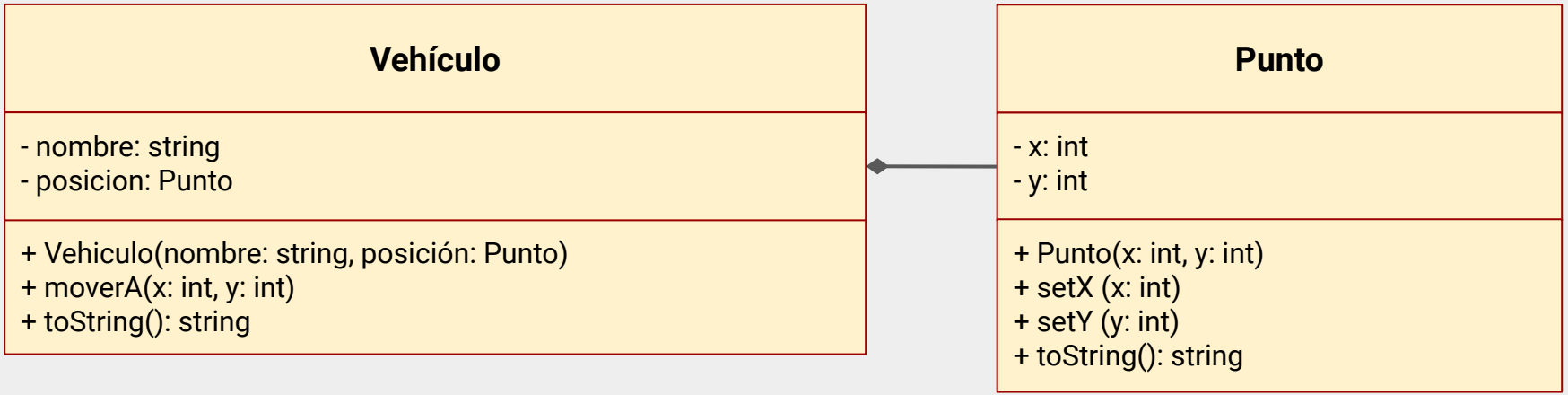
```
int main(){
    Autor a1("Gabriel", "Garcia Marquez", "ggarciamarquez@test.com");

    Libro l1("Cien años de soledad", &a1, 412.19, 45);
    Libro l2("El coronel no tiene quien le escriba", &a1, 224.23, 17);

    cout<<l1.toString();
    cout<<l2.toString();
    return 0;
}
```

Ejercicio 1.06.

Composición: Vehículo - Punto



composición: clase

```
class Vehiculo{
    private:
        string nombre;
        Punto posicion;
    public:
        Vehiculo(string);
        Vehiculo(string, int, int);
        void moverA(int, int);
        string toString();
};
```

composición: constructor

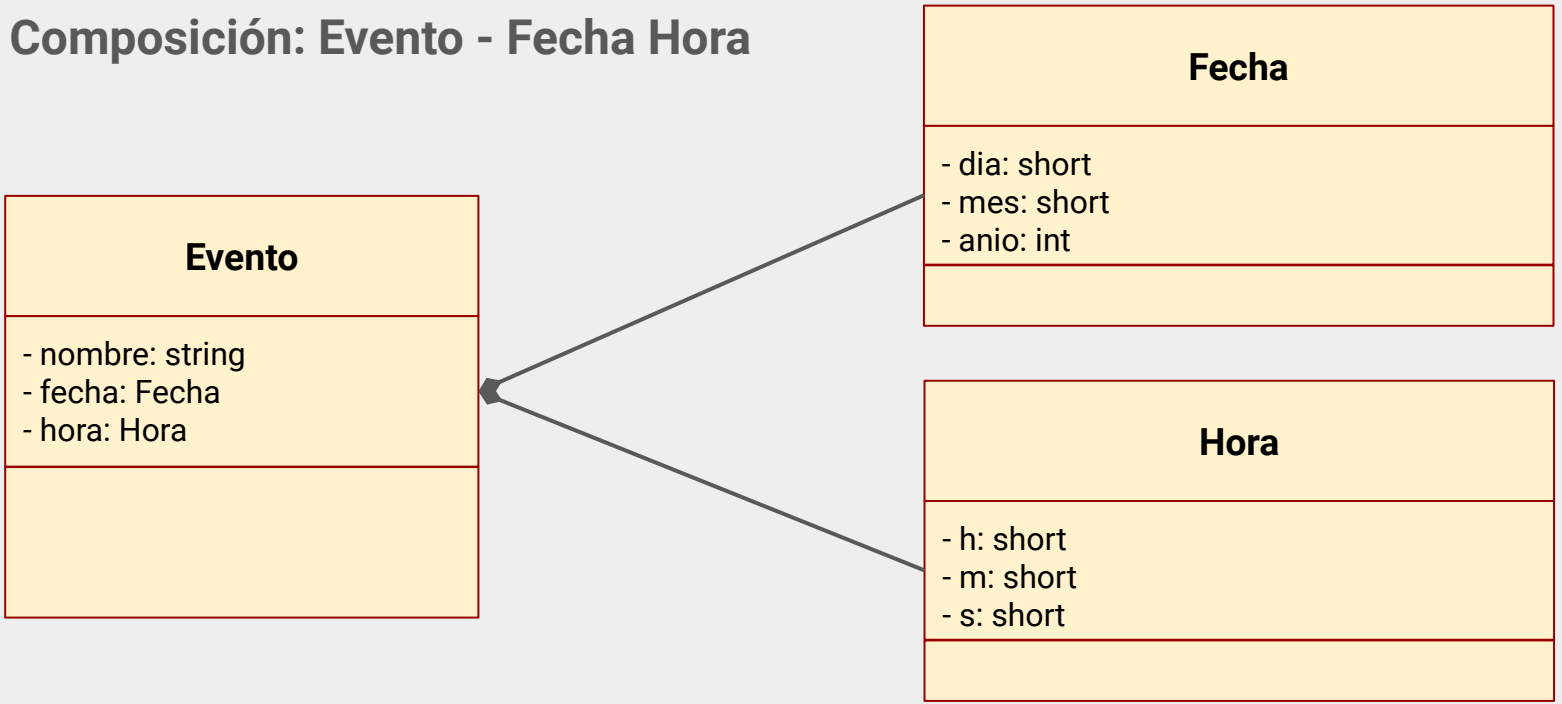
```
Vehiculo::Vehiculo(string nombre):posicion(0,0){
    this->nombre = nombre;
}
```

También se puede escribir:

```
Vehiculo::Vehiculo(string nombre):nombre(nombre),posicion(0,0){}
```

Ejercicio 1.07.

Composición: Evento - Fecha Hora



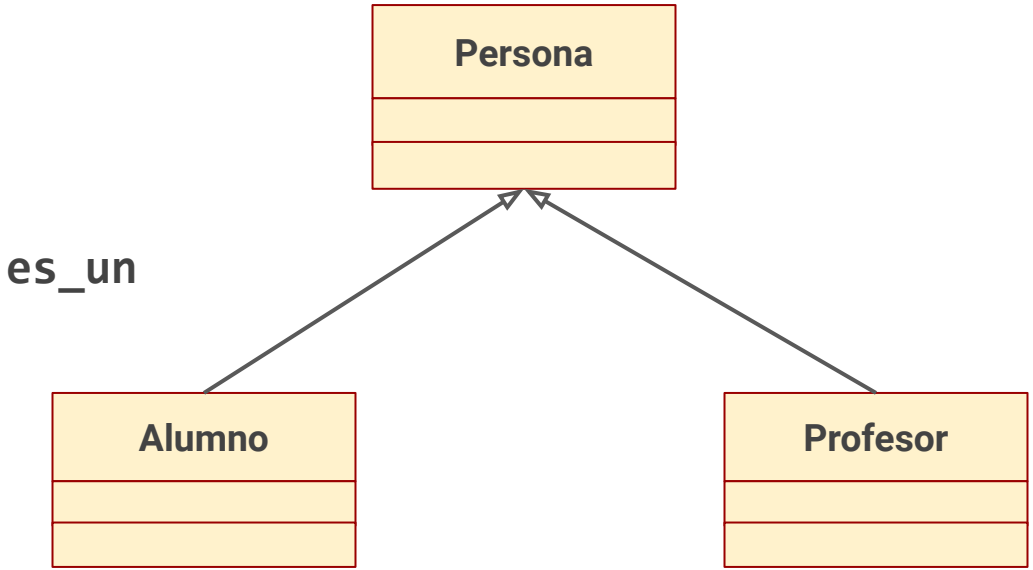
Ejercicio 1.08.

Abstracción:

En este punto vos serás quien plantee el problema y quien haga el ejercicio de abstracción.

Pensá en un problema que se pueda modelar con un abordaje orientado a objetos, realizá la abstracción del problema e implementalo, simplificandolo si es necesario

Herencia



¿qué se hereda?

¿Qué hereda la clase derivada de la clase base?

Todos los miembros de la clase base excepto:

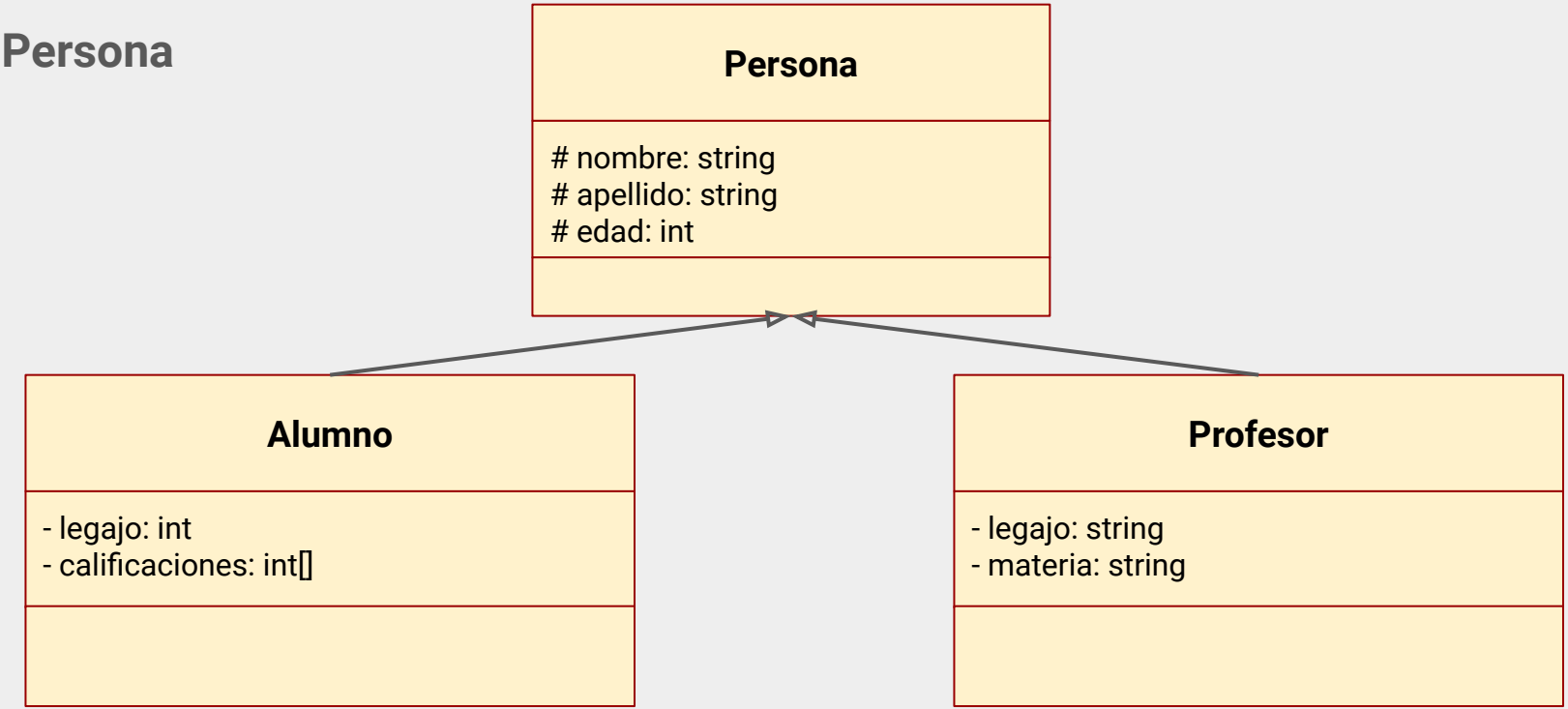
- Constructor
- Destructor
- Operador de asignación (=)
- Friends
- Miembros privados

modificadores

Modificador de acceso en la clase base	Modificador de acceso cuando se hereda como public	Modificador de acceso cuando se hereda como private	Modificador de acceso cuando se hereda como protected
Public	Public	Private	Protected
Private	Inaccessible	Inaccessible	Inaccessible
Protected	Protected	Private	Protected

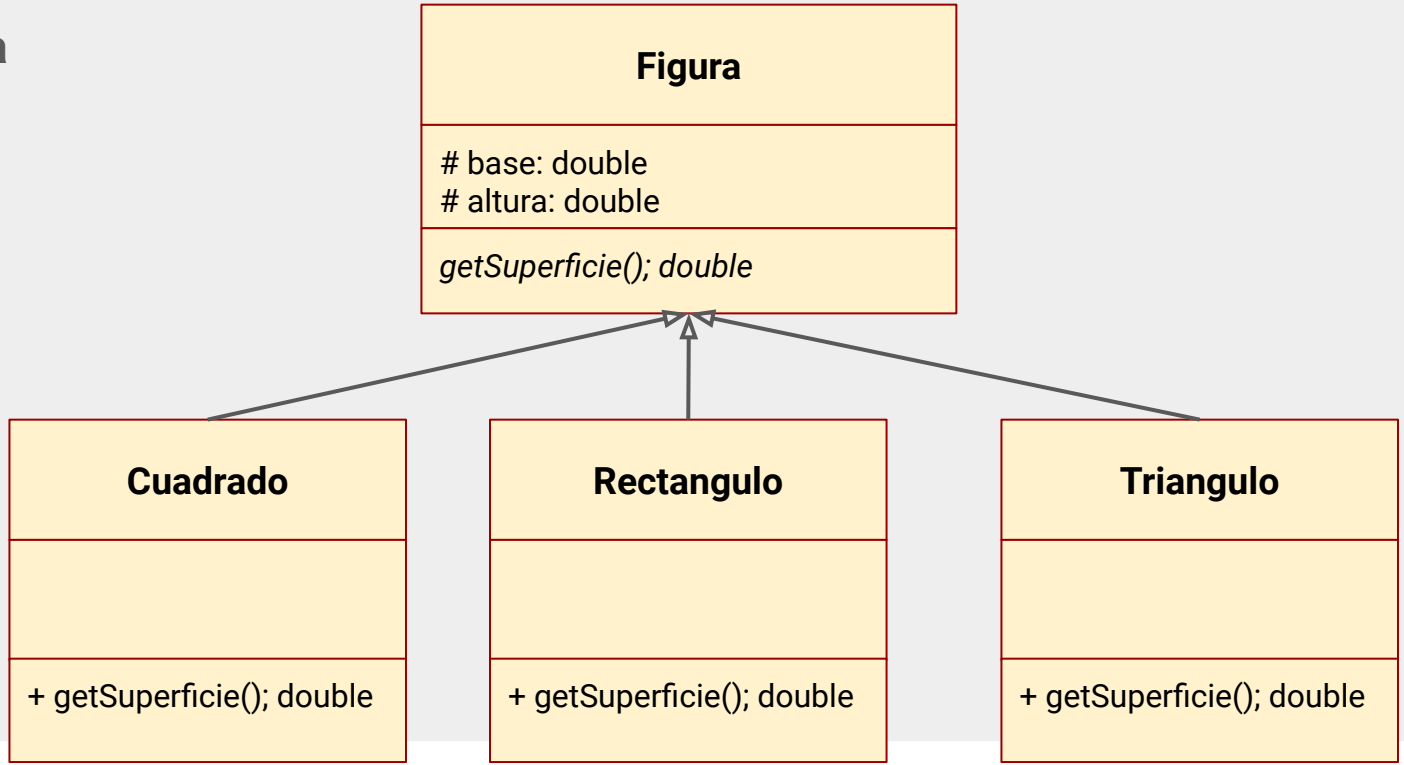
Ejercicio 1.09.

Persona



Ejercicio 1.10.

Figura



polimorfismo: ejemplo

```
class Base{
      public:
      virtual void imprimir(){ cout << "Imprimir() desde la clase base" << endl;}
      void mostrar(){ cout << "mostrar() desde la clase base" <<  endl; }
};
class Derivada : public Base{
      public:
      virtual void imprimir(){ cout << "Imprimir() desde la clase derivada" << endl; }
      void mostrar(){ cout << "mostrar() desde la clase derivada" << endl; }
};
int main(){
      Base* bptr;
      Derivada d;
      bptr = &d;
      bptr->imprimir();
      bptr->mostrar();
}
```

Ejercicio 1.11.

A la clase `Vehiculo` del ejercicio 1.06 agregale el método

```
double Distancia(Vehiculo);
```

que retorna la distancia a otro vehículo, este método puede (y debería) usar el método

```
double Punto::Distancia(Punto);
```

anteriormente implementado en la clase `Punto`

Trabajo Práctico Grupal

Diseñar e implementar una clase o biblioteca de clases que sirva para manejar la salida por consola de nuestros programas.

Consideraciones:

- Crear un diseño simple que evolucione de acuerdo a las necesidades que surjan a posteriori
- Tener en cuenta distintas estructuras de datos que conocemos hasta ahora (vector, matriz, etc), la biblioteca irá evolucionando en la medida que incorporemos nuevas estructuras de datos

Lecturas

- **SOLID, cinco principios básicos de la programación orientada a objetos:**
  http://bemobile.es/blog/2016/09/solid-5-principios-basicos-de-la-programacion-orientada-a-objetos/
- **10 OOP Design Principles Every Programmer Should Know:**
  https://hackernoon.com/10-oop-design-principles-every-programmer-should-know-f187436caf65