Programación Orientada a Objetos POO

AED I

2020

UBP UNIVERSIDAD BLAS PASCAL

paradigmas de programación

- **Imperativo**: expresa el **cómo**
- **Declarativo**: expresa el **qué**

paradigmas de programación

- **Imperativos**:
  - **Estructurado:** implementa estructuras básicas: secuencia, condición, repetición
  - **Modular:** división de programa en subprogramas o módulos
  - **Orientado a Objetos:** el problema se describe en forma de objetos que interactúan
- **Declarativos**:
  - **Funcional**: basado en el cálculo lambda y lógica combinatoria
  - **Lógico**: basado en la lógica de predicados
  - **Reactivo**: basado en la teoría de grafos

programación orientada a objetos - poo

La **Programación Orientada a Objetos** es una metodología de programación que asocia **estructuras de datos** con un conjunto de **operaciones** que actúan sobre ellas

fortalezas de la poo

- Modela problemas del mundo real con una **perspectiva** más cercana a la del usuario
- Favorece la **reutilización** de software
- Se puede **modificar** y **extender** las implementaciones de los componentes sin afectar al resto de los componentes

clase

Una **clase** es una **abstracción** de una entidad del mundo real.
Es la definición de los objetos del mismo tipo.

Especifica:

- las **propiedades** o **atributos**
- y los **métodos** o el **comportamiento**

objeto

Un **objeto** es instancia de una **clase.** Su representación real.

Un objeto tiene:

- **Identidad**
- **Estado**
- **Comportamiento**

mensajes

Los **objetos** interactúan a través de **mensajes**

encapsulamiento

El objeto ofrece una interfaz con la cual puede interactuar el resto de los objetos ocultando su estado interno que no puede ser modificado o accedido sino a través de esa interfaz.

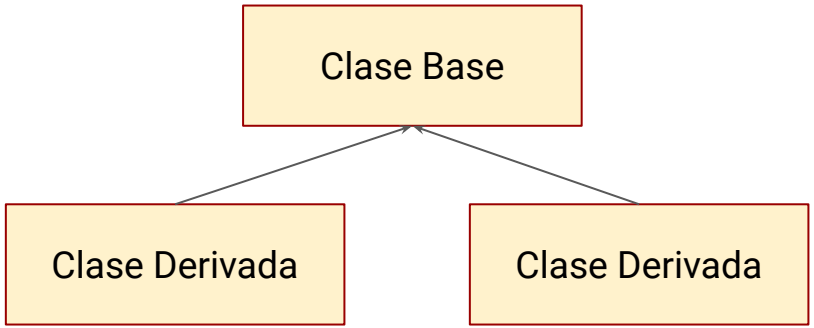
- Actúa como una caja negra
- Permite modificar su implementación sin afectar al resto de los objetos

herencia

Es el mecanismo que nos permite definir clases a partir de otras clases existentes.

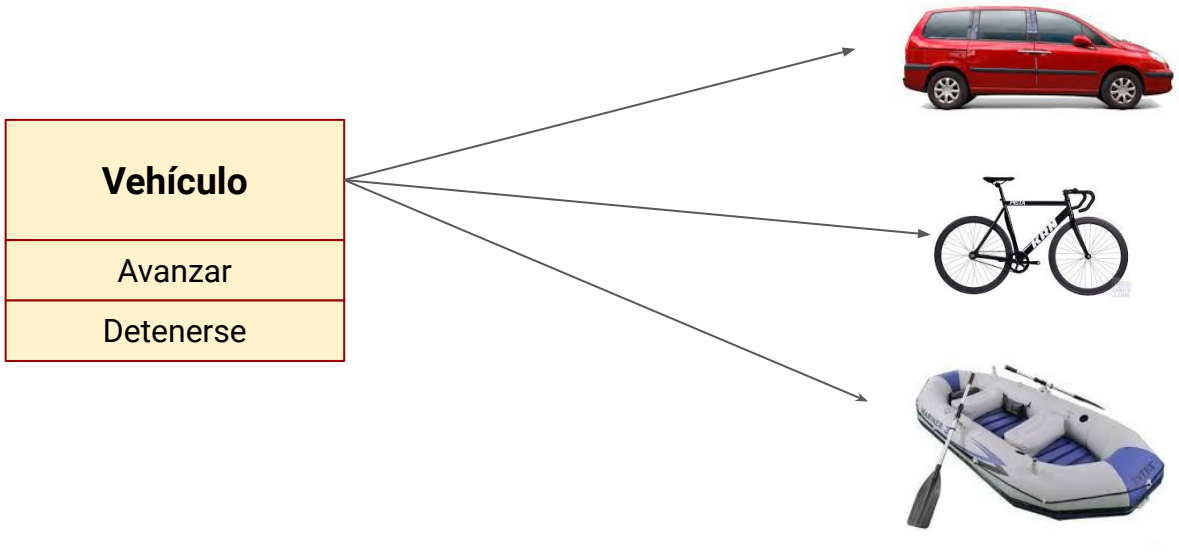
Las **clases derivadas** o **subclases** **heredan** atributos y comportamiento de la **clase base** o **superclase**

Nos permite formar jerarquías de clases relacionadas por este mecanismo



polimorfismo

Un solo mensaje puede producir respuestas diferentes, dependiendo del objeto que lo reciba



clase

```
class Persona {

}
```

atributos (estado)

```
class Persona {
   private:
         int edad;
         string nombre;


}
```

constructor por defecto

```
class Persona {
   private:
         int edad;
         string nombre;
   public:
         Persona();
}
```

constructor

```
class Persona {
   private:
         int edad;
         string nombre;
   public:
         Persona(int,string);
}
```

constructor: sobrecarga

```
class Persona {
   private:
         int edad;
         string nombre;
public:
         Persona(int,string);
         Persona(int);
}
```

constructor: implementación

```
Persona::Persona(int _edad, string _nombre){
    edad = _edad;
    nombre = _nombre;
}

Persona::Persona(string _nombre){
    nombre = _nombre;
    edad = -1;
}
```

instancias (objetos)

```
int main(){
      Persona p1 = Persona(21, "Maria");
      Persona p2(23, "Pedro");
      Persona p3("Juan");
}
```

this

```
Persona::Persona(int edad, string nombre){
   this->edad = edad;
   this->nombre = nombre;
}

Persona::Persona(string nombre){
   this->nombre = nombre;
   this->edad = -1;
}
```

destructor

```
class Persona {
   private:
         int edad;
         string nombre;
   public:
         ~Persona();
}
```

getters y setters

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

void Persona::setNombre(string nombre){
    this->nombre = nombre;
}
```

getters y setters

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

void Persona::setNombre(string nombre){
    this->nombre = nombre;
}
```

constructor de copia

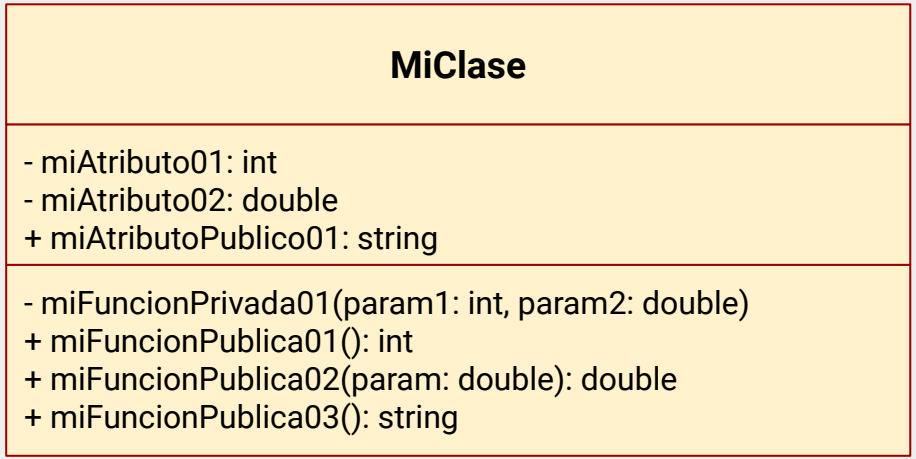
```
Persona(const Persona &persona){
        nombre = persona.nombre;
        apellido = persona.apellido;
        edad = persona.edad;
}
```

invocar al constructor de copia

```
Persona p2 = p1;
Persona p3(p1);
```

diagrama de clases

Referencia rápida



repaso

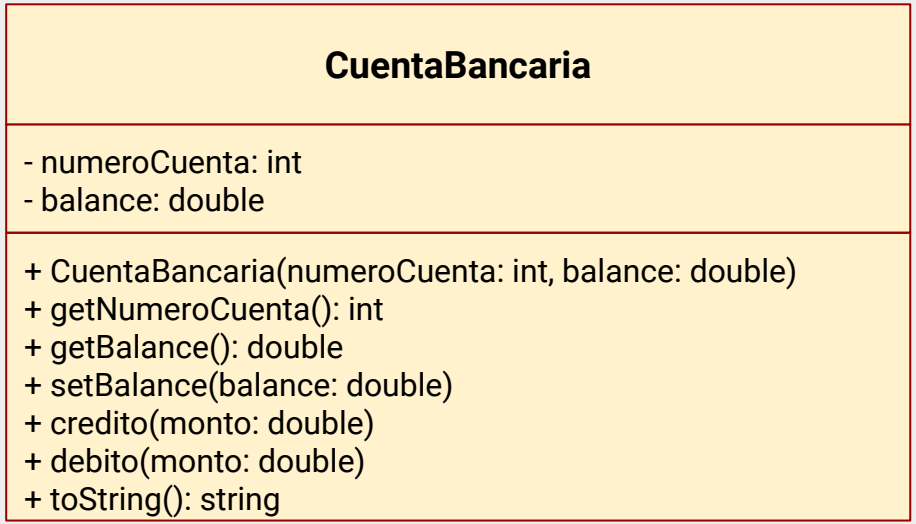
- Paradigma POO
- Características de un objeto
- Clase vs Objeto
- Miembros de una clase
- Constructor
- Constructor por defecto
- Destructor
- this
- Encapsulamiento: getters y setters

tareas

Realizar los ejercicios 1.01, 1.02, 1.03, 1.04 de la guía

Ejercicio 1.01.

Cuenta bancaria

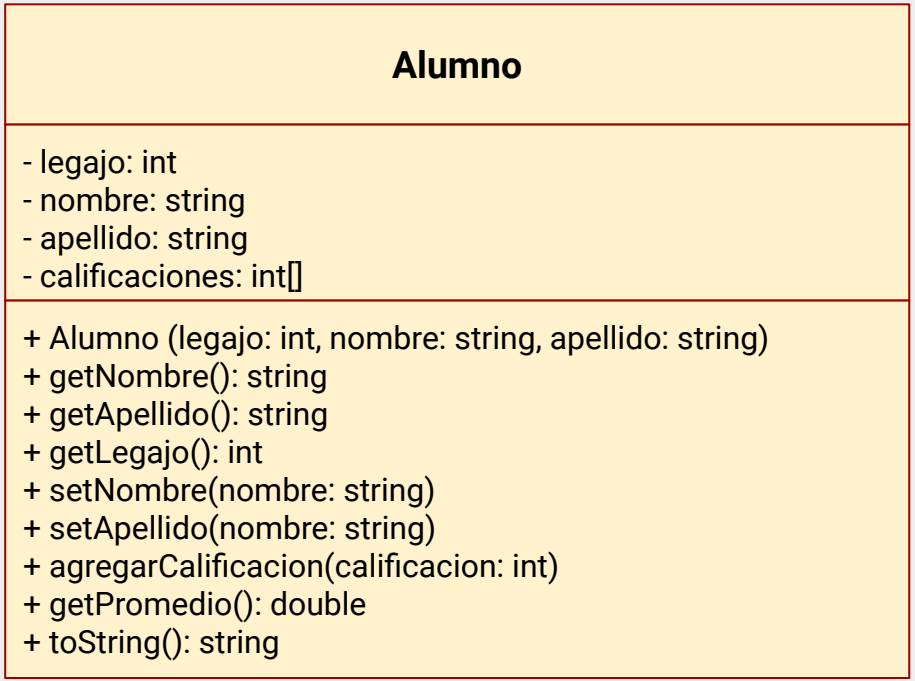


Ejercicio 1.02.

Fecha

Ejercicio 1.03.

Alumno



Ejercicio 1.04.

Punto