Algorítmica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

Programación Orientada a Objetos

Temas para esta sesión

- **Encapsulamiento**
- **Herencia**
- **Polimorfismo**
- **Abstracción**

ITAM

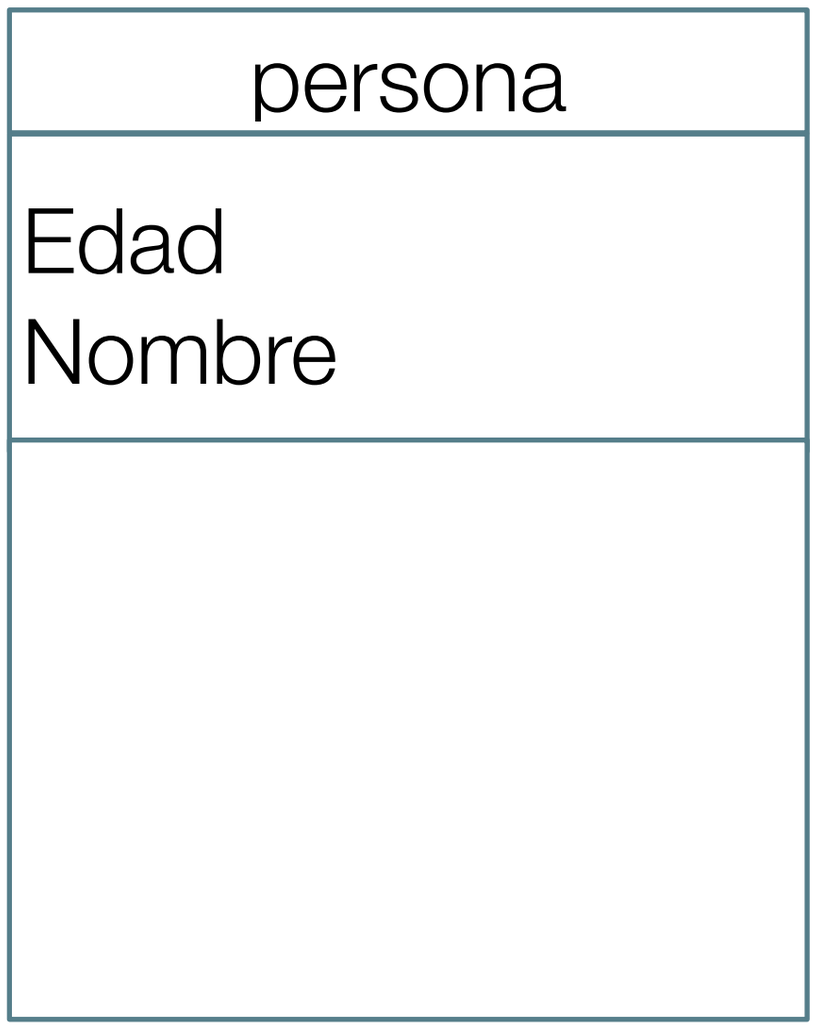
Encapsulamiento

Getters & Setters

- Controlar el acceso a atributos
- Uso de __ antes de definir atributos
- Obtener valor de atributos con métodos **get**
- Asignar valor a atributos con métodos **set**

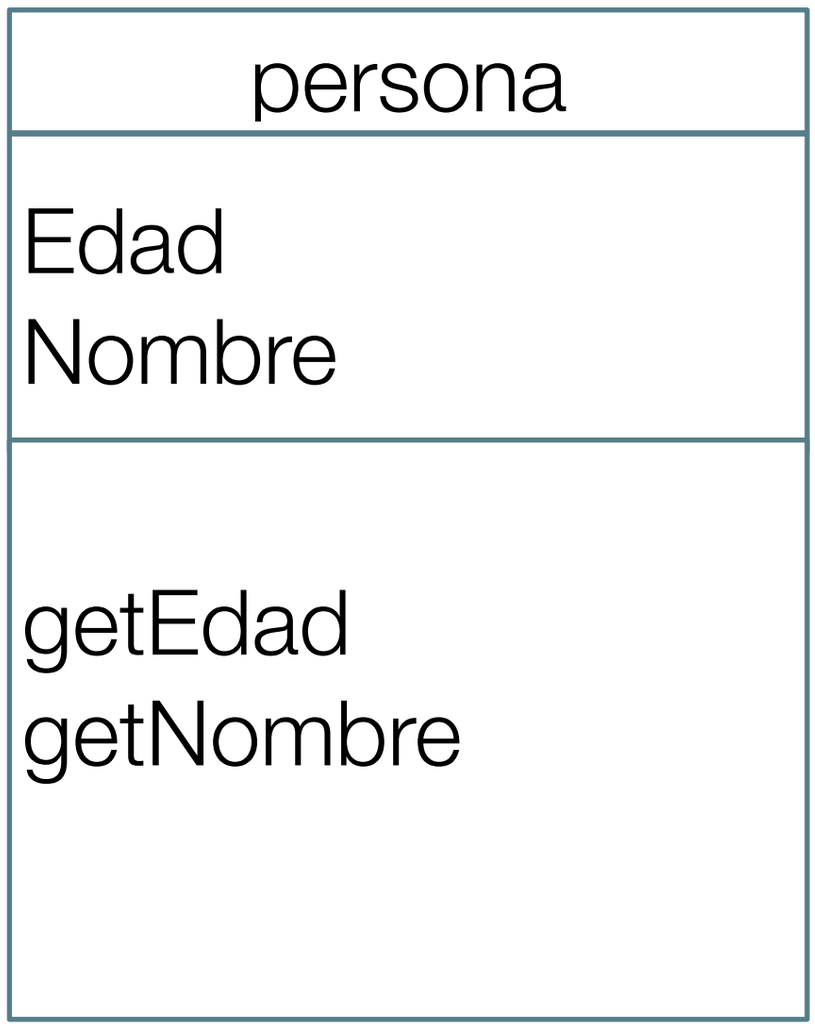
Getters & Setters

- Ejemplo: Agregando __ a atributos de clase



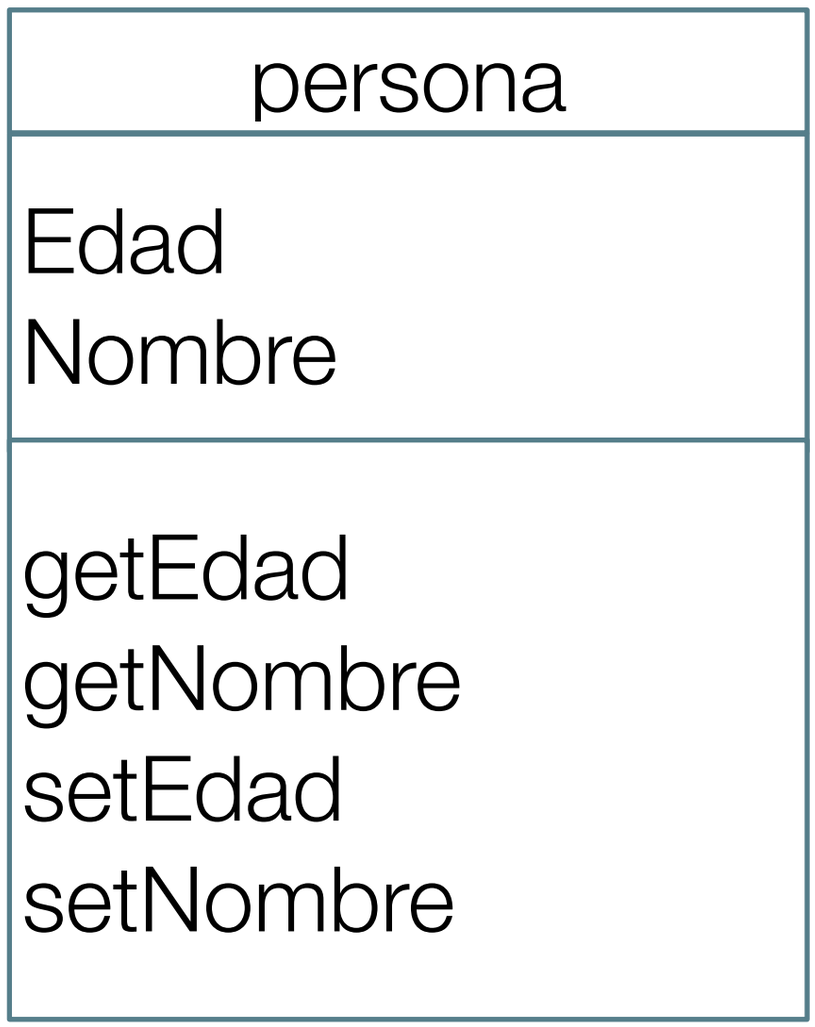
Getters & Setters

- Ejemplo: Definiendo getters



Getters & Setters

- Ejemplo: Definiendo setters



Limitar creación de nuevos atributos

- __slots__()

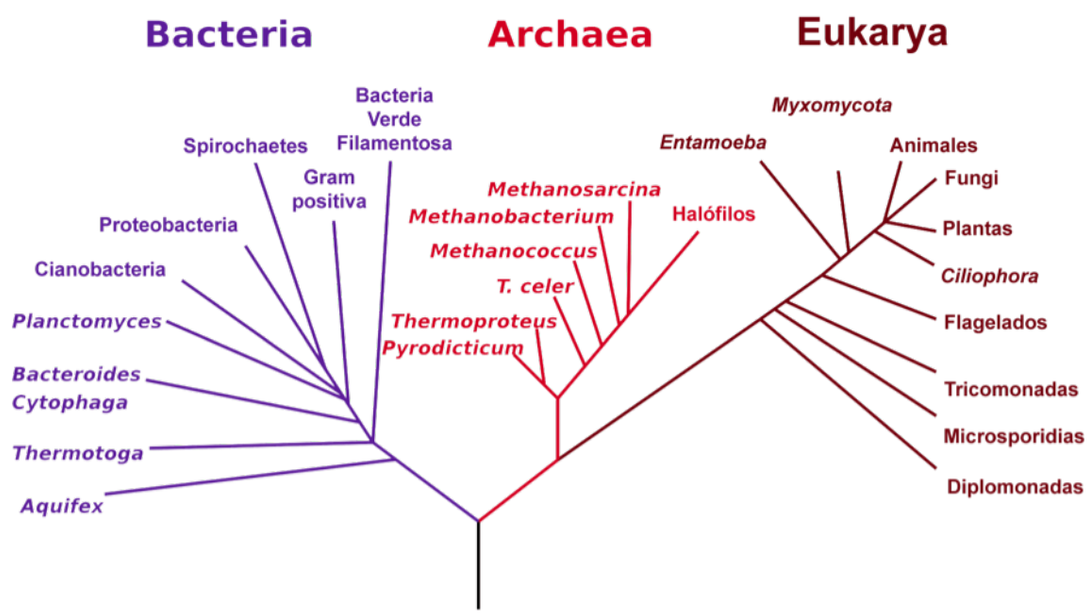
Diccionario de atributos

- tmpD = p1.__dict__

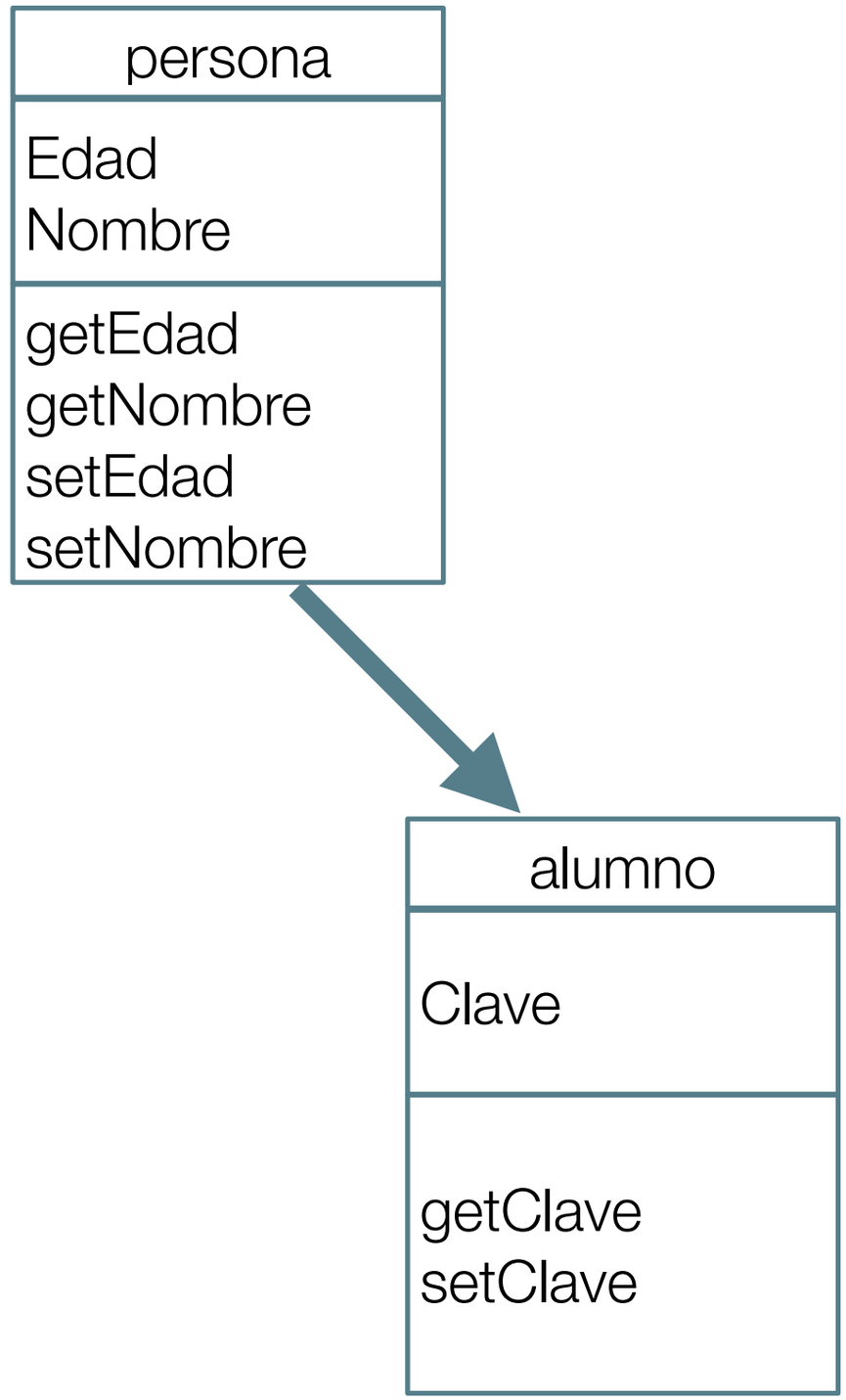
Herencia

Herencia

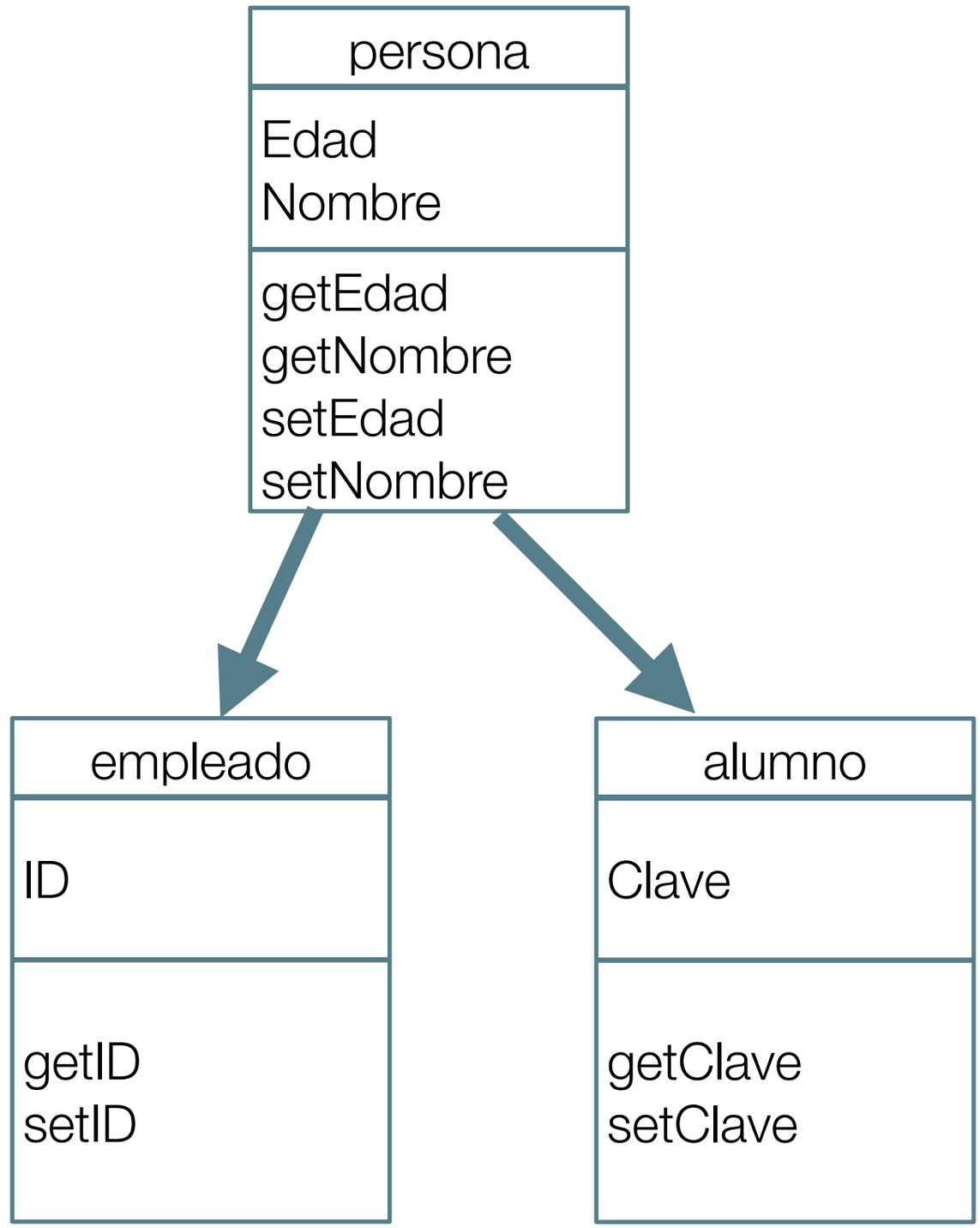
- Idea general
  - Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación
- Herencia sencilla
  - Una clase hereda atributos y métodos de una clase “superior”
- Herencia multiple
  - Una clase hereda atributos y métodos de más de una clase



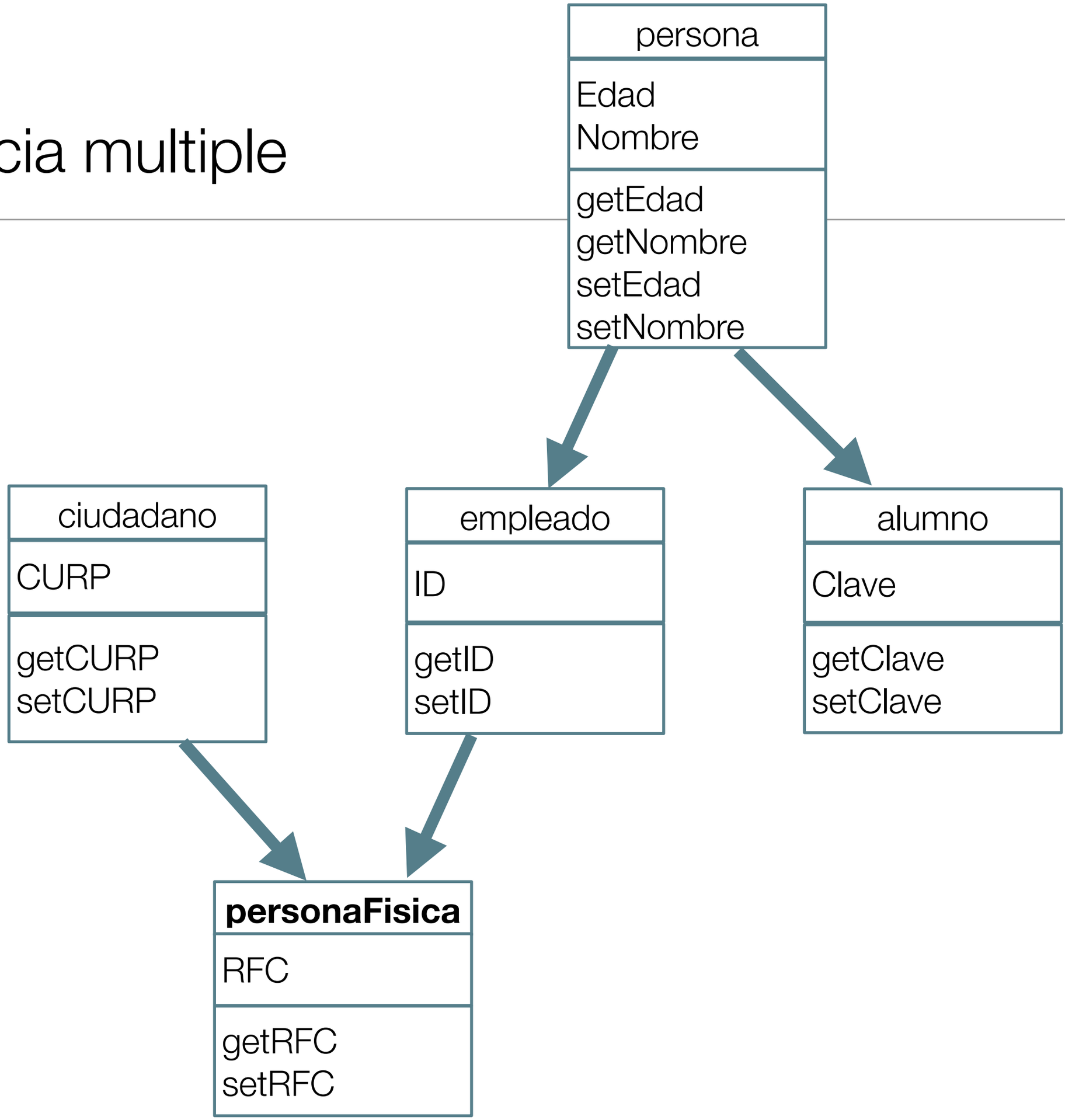
Herencia simple



Herencia simple



Herencia multiple



Polimorfismo

Polimorfismo

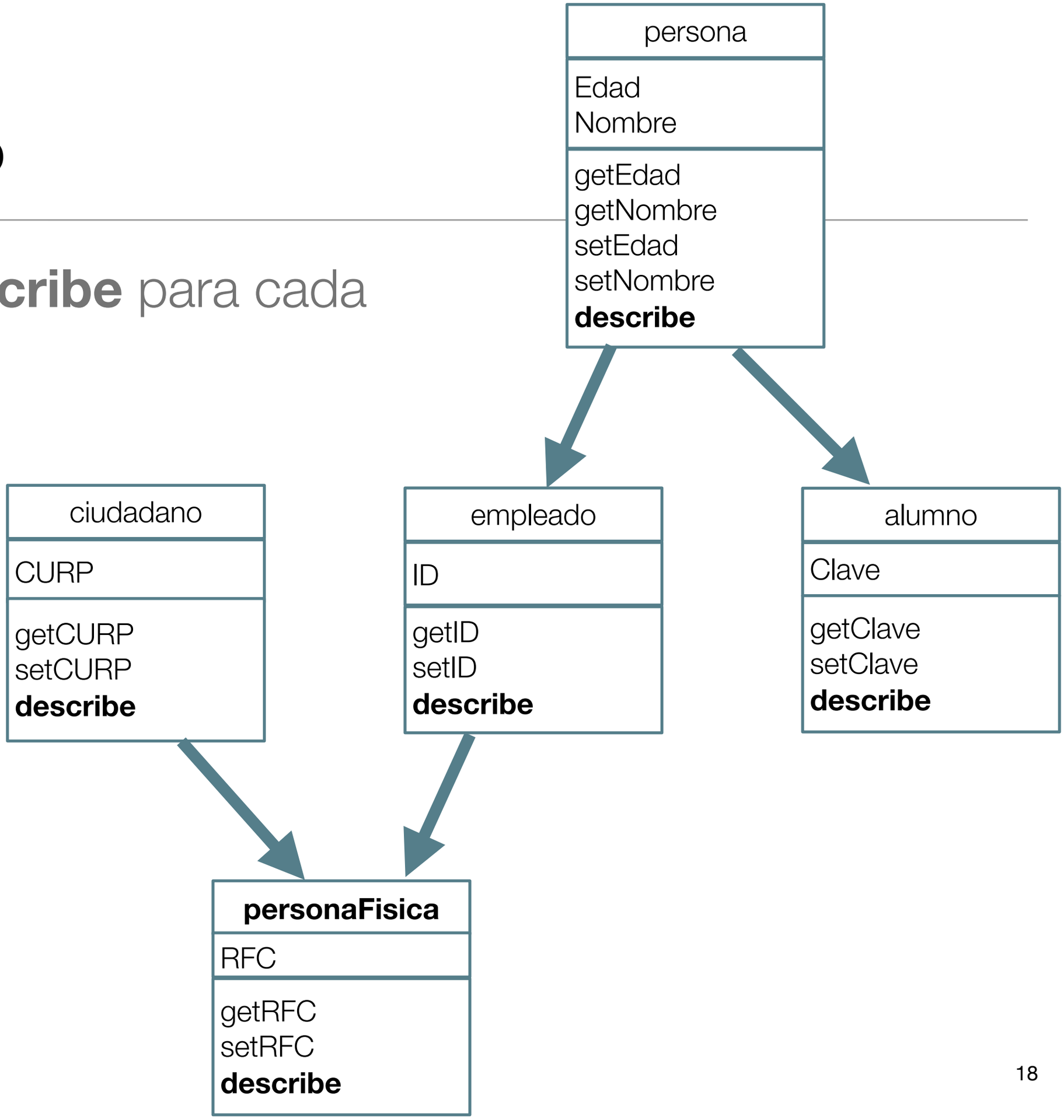
- La palabra polimorfismo, del griego poly morphos (varias formas), se refiere a la habilidad de objetos de distintas clases de responder al mismo mensaje.

- Herencia

- Sobrecarga de métodos / funciones

Polimorfismo

- Método **describe** para cada clase

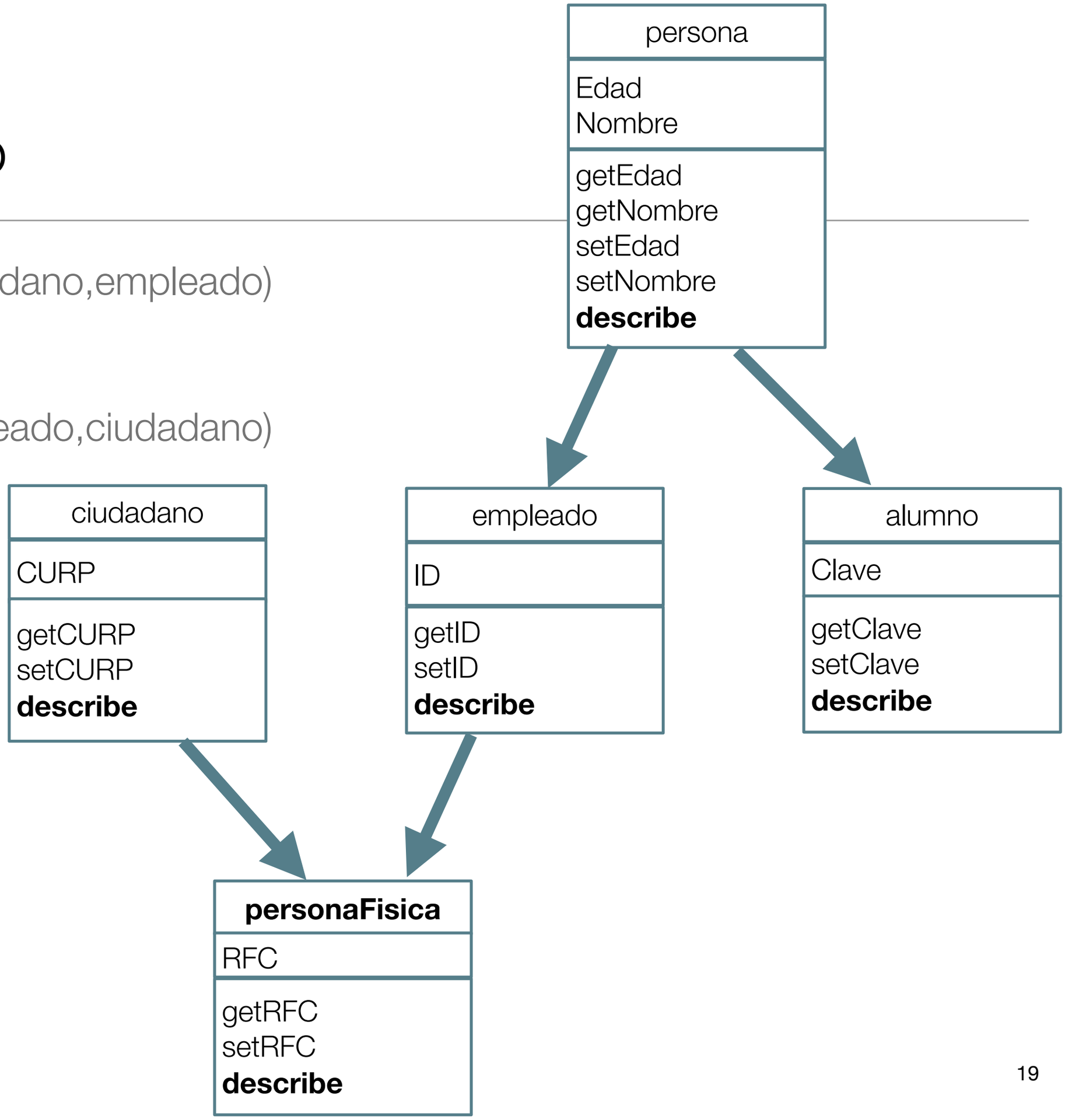


Polimorfismo

personaFisica(ciudadano,empleado)

VS

personaFisica(empleado,ciudadano)



Otros métodos especiales

Métodos especiales

- **__str__(self)**
  - Método llamado para crear una cadena de texto que represente a nuestro objeto. Se utiliza cuando usamos **print** para mostrar nuestro objeto o cuando usamos la función **str**(obj) para crear una cadena a partir de nuestro objeto
- **__del__(self)**
  - Método llamado cuando el objeto va a ser borrado. También llamado destructor, se utiliza para realizar tareas de limpieza

Persona

```
class persona:
    __Edad = 0
    __Nombre = ""
    __VarOculto = 0

    #__slots__ = ('__Edad','__Nombre')

    def __init__(self,val1,val2):
        self.setNombre(val1)
        self.setEdad(val2)

    def getNombre(self):
        return self.__Nombre

    def getEdad(self):
        return self.__Edad

    def setNombre(self,val1):
        self.__Nombre = val1
        print('Cambio de nombre realizado:', val1)

    def setEdad(self,val1):
        # Calculo y/o validaxiones
        print('Validando edad ...')
        self.__Edad = val1
        self.__VarOculto = val1 * 3
        print('Edad valida: ', val1)

    def describe(self):
        print("Objeto tipo persona")
        print('Nombre', self.getNombre())
```

Alumno

Empleado

```
class alumno(persona):
    __Clave = ""
    def __init__(self,val1):
        self.__Clave = val1

    def getClave(self):
        return self.__Clave

    def setClave(self,parmA):
        self.__Clave = parmA

class empleado(persona):
    __ID = ''
    def __init__(self,val1):
        self.__ID = val1

    def getID(self):
        return self.__ID

    def setID(self,val1):
        self.__ID = val1

    def describe(self):
        print("Objeto tipo empleado ... ")
```

Ciudadano

```
class ciudadano:
    __CURP = ''
    def __init__(self,val1):
        self.__CURP = val1

    def getCURP(self):
        return self.__CURP

    def setCURP(self,val1):
        print("Validando CURP ... ")
        # Calcular aquí o llamar otro metodo
        self.__CURP = val1

def describe(self):
    print("Objeto tipo ciudadano")
```

Ciudadano

```
class ciudadano:
    __CURP = ''
    def __init__(self,val1):
        self.__CURP = val1

    def getCURP(self):
        return self.__CURP

    def setCURP(self,val1):
        print("Validando CURP ... ")
        # Calcular aquí o llamar otro metodo
        self.__CURP = val1

def describe(self):
    print("Objeto tipo ciudadano")
```

Persona física

```
class personaFisica(empleado,ciudadano):
    __RFC = ''
    def __init__(self,val1):
        self.__RFC = val1
        self.setNombre("Bob")

    def getRFC(self):
        return self.__RFC

    def setRFC(self,val1):
        self.__RFC = val1

    def __str__(self):
        return 'Persona : ' + self.getNombre() + ' RFC: ' + self.getRFC()

    def __del__(self):
        print("Eliminando ", self.getNombre())


#p1 = persona("Alice",23)
#p2 = persona("Bob",25)
#p3 = alumno("ABC123")
#p4 = empleado("XYZ456")

p5 = personaFisica("RFC123ABC")
```