Conceptos avanzados de Programación Orientada a Objetos

Introducción a la Programación

Agenda

- Herencia
- Polimorfismo
- Práctica

Herencia

- Mecanismo mediante el cual se crean clases a partir de las existentes, en donde una nueva clase o subclase hereda características de clases padre o super clases

- Se pueden agregar nuevas características a las subclases

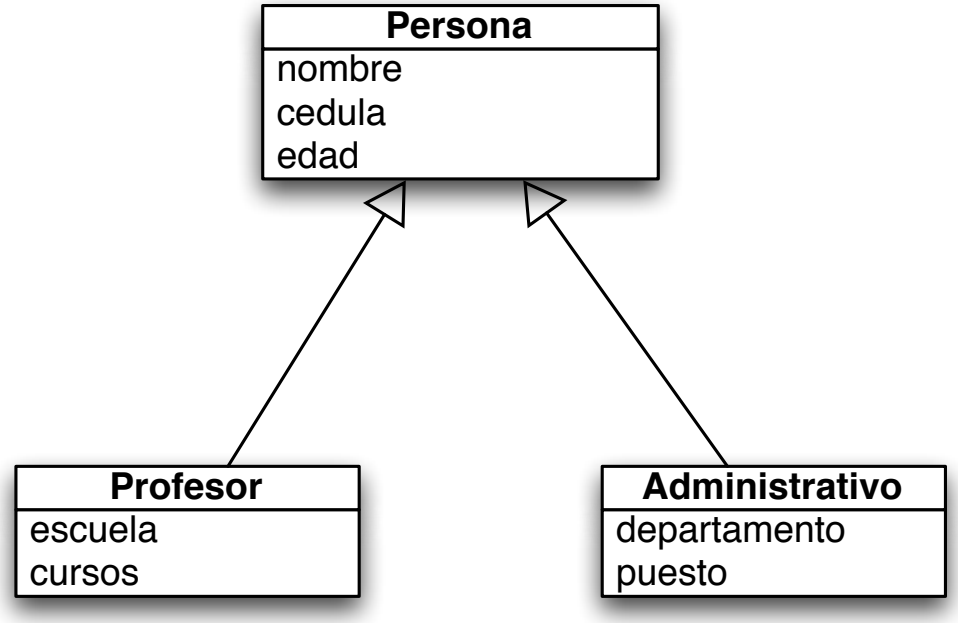
- Al acto de heredar una clase se le llama también extender una clase

```
class NombreClaseDerivada(NombreClaseBase)
```

- Si la clase base está en otro módulo, la sintaxis es la siguiente:

```
class NombreClaseDerivada(nombreModulo.NombreClaseBase)
```

Herencia


Persona
nombre
cedula
edad
Profesor
escuela
cursos
Administrativo
departamento
puesto

Herencia

```
class Persona:
  def __init__(self, cedula, nombre, edad):
     self.cedula = cedula
     self.nombre = nombre
     self.edad = edad

  def mostrar(self):
     print("Nombre: ", self.nombre)
     print("Cedula: ", self.cedula)
     print("Edad: ", self.edad)

  def __str__(self):
     return "Nombre: " + self.nombre + " Cedula:" +
str(self.cedula) + " Edad: " + str(self.edad)
```

Herencia

```
class Profesor(Empleado):
  def __init__(self, cedula, nombre, edad, escuela,
cursos):
     Empleado.__init__(self, cedula, nombre, edad)
     self.escuela = escuela
     self.cursos = []

  def asignaCursos(self, lista):
     self.cursos = lista

  def cantidadCursos(self):
     return len(self.cursos)

  def mostrar(self):
     Empleado.mostrar(self)
     print("Escuela: ", self.escuela)
     print("Cursos: ", self.cursos)
```

Herencia

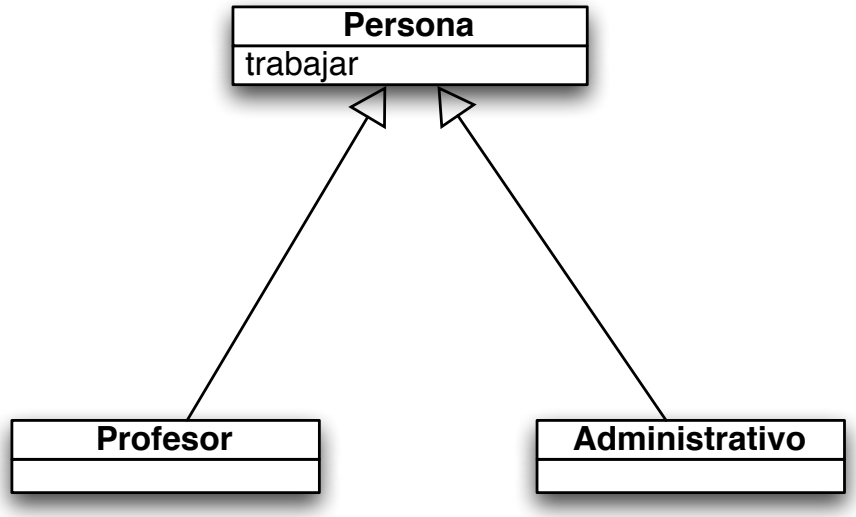
```
class Administrativo(Empleado):
  def __init__(self, cedula, nombre, edad, departamento,
puesto):
     Empleado.__init__(self, cedula, nombre, edad)
     self.departamento = departamento
     self.puesto = puesto

  def mostrar(self):
     Empleado.mostrar(self)
     print("Departamento: ", self.departamento)
     print("Puesto: ", self.puesto)
```

Herencia

```
>>> andrei = Profesor("1001", "Andrei", 25, "Computacion", [])
>>> maria = Administrativo("1002", "Maria", 35, "Recursos
Humanos","Psicóloga", [])
>>> andrei.asignaCursos(["Intro", "Taller"])
>>> andrei.cantidadCursos()
2
>>> andrei.mostrar()
Nombre:  Andrei
Cedula:  1001
Edad:    25
Escuela:  Computacion
Cursos:  ['Intro', 'Taller']
>>> maria.mostrar()
Nombre:  Maria
Cedula:  1002
Edad:    35
Departamento:  Recursos Humanos
Puesto:  Psicóloga
```

Polimorfismo

- Es un concepto que permite que dos métodos implementen acciones diferentes, dependiendo del objeto que lo ejecuta o los parámetros recibidos



Polimorfismo

```
class Persona:
  def trabajar(self):
     return "persona trabajando"

class Profesor(Persona):
  def trabajar(self):
     return "estoy dando clases"

class Administrativo(Persona):
  def trabajar(self):
     return "estoy llenando formularios"
```

Polimorfismo

```
>>> a=Persona()
>>> b=Profesor()
>>> c=Administrativo()

>>> a.trabajar()
'persona trabajando'

>>> b.trabajar()
'estoy dando clases'

>>> c.trabajar()
'estoy llenando formularios'
>>>
```

Práctica

- Escriba la clase Estudiante, que hereda de la clase Persona. La clase estudiante deberá tener los atributos carné, listaDeCursos, listaDeActividadesExtracurriculares

- La clase estudiante deberá tener métodos para agregar o eliminar un curso, y agregar o eliminar actividades extracurriculares