Universidad de Londres

Programación orientada a objetos

HERENCIA Y POLIMORFISMO

DR. (C) NOÉ ALEJANDRO CASTRO SÁNCHEZ

Características de la Programación orientada a objetos

Características de la POO

- Abstracción
- Encapsulación
- **Herencia**
- **Polimorfismo**

Abstracción

- Visión general de un problema
  - Considera aspectos generales
  - Se enfoca a información necesaria

Lavadora
Marca Modelo Num_Serie Capacidad
agregarRopa() agregarDetergente() sacarRopa()

Lavadora
marca modelo num_Serie capacidad volumenTambor trampa Motor
velocidadMotor agregarRopa() agregarDetergente() sacarRopa() agregarBlanqueador() cronometrarRemojo()

Encapsulación

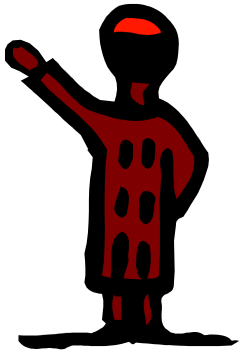
- Considera objetos como cajas negras
  - Los utilizamos sin importarnos la manera en que trabajan
  - Comportamiento y atributos conocidos, pero no su implementación (trabajo interno)
- Clase = Abstracción + encapsulación

Herencia

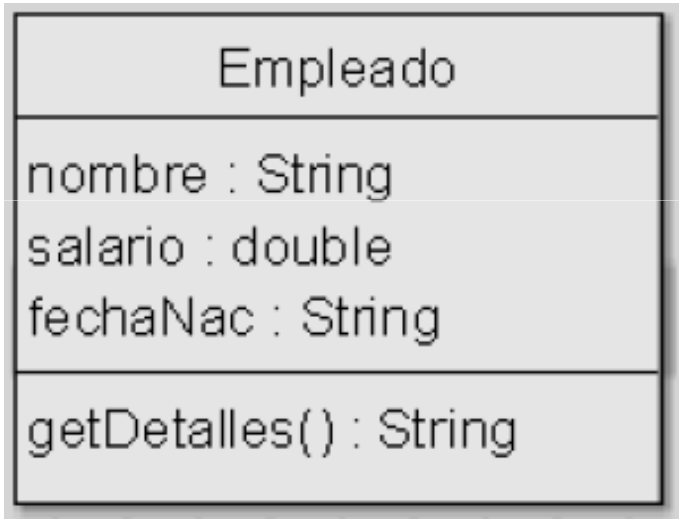
- Jerarquía
  - Los objetos del mundo se relacionan (agrupan) entre sí de manera jerárquica
  - Las agrupaciones se basan en los rasgos compartidos
- Las clases de un programa se organizan mediante una jerarquía
- **Reutilización de código**

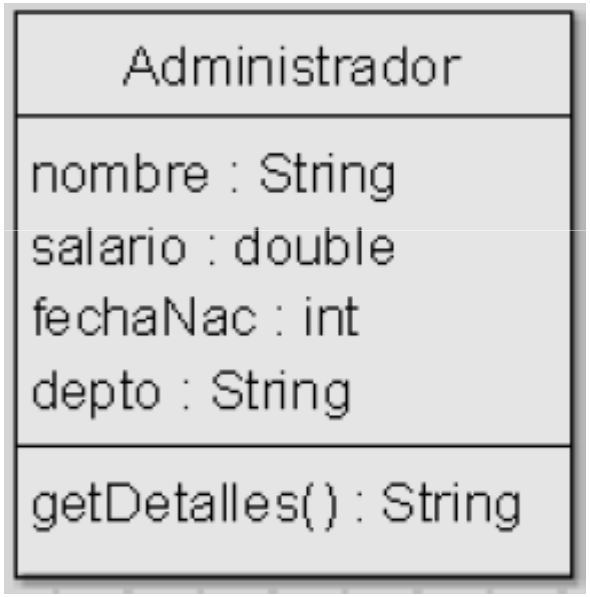
Herencia II

- Se puede modelar cualquier fenómeno



Herencia III





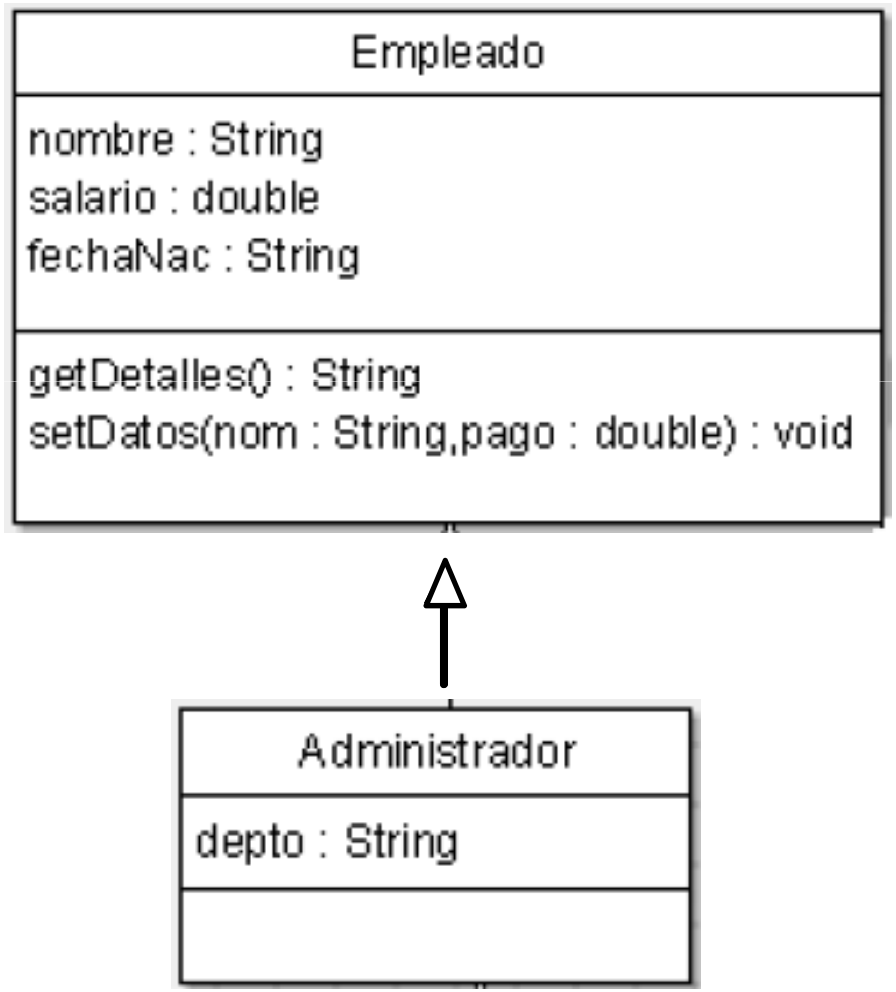
Herencia IV

```
public class Empleado {
  private String nombre;
  private double salario;
  private String fechaNac;

  public String getDetalles() {
     String det = nombre +   "sueldo:" +
                      salario;
     return det;
  }
}
```

```
public class Administrador {
  private String nombre;
  private double salario;
  private String fechaNac;
  private String depto;

  public String getDetalles() {
     String det = nombre + "sueldo:" + salario;
     return det;
  }
}
```

- Duplicidad de información
  - ¿Solución? **Herencia**

Herencia en UML



Herencia V

```
public class Empleado {
  private String nombre;
  private double salario;
  private String fechaNac;

  public String getDetalles() {
     String det = nombre + “sueldo:“  +  salario;
     return det;
  }
 public void setDatos(String nombre, double salario)
  {
      this.nombre = nombre;
      this.salario = salario;
  }
}
```

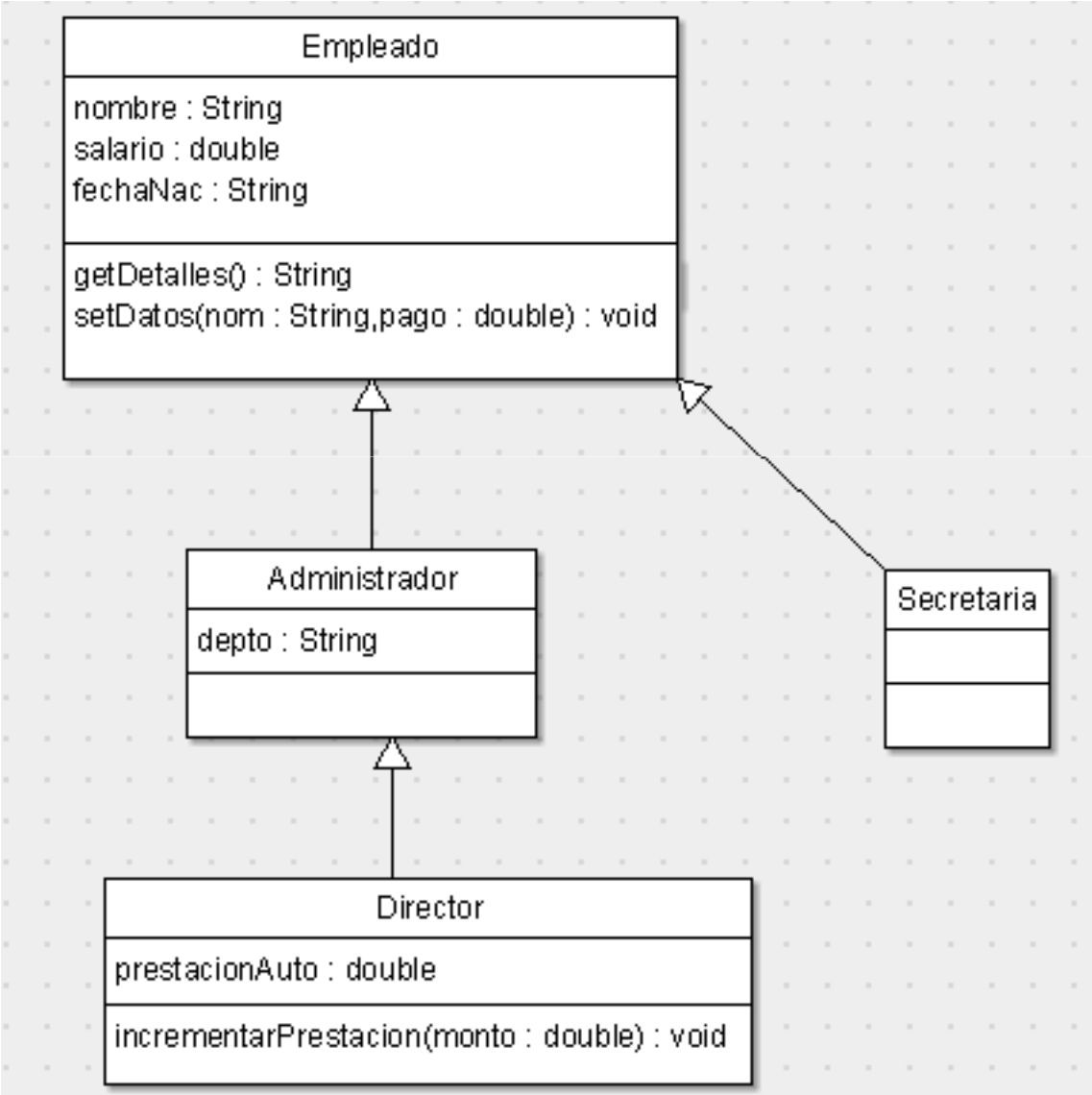
Herencia VI

```
public class Administrador extends Empleado
{
        private String depto;
}
```

Herencia VII

```
public class TestEmpleados {
    public static void main (String[] args)
    {
        Empleado e = new Empleado();

        e.setDatos("pedro", 1500);
        System.out.println(  e.getDetalles() );

        Administrador a = new Administrador();
        a.setDatos("Pilar", 40000);
         System.out.println (a.getDetalles() );
    }
}
```

Jerarquía de clases



Herencia VIII

```
public class Director extends Administrador
{
        private double prestacionGasolina = 0;

        public void incrementarPrestacion(double monto)
        {
                        prestacionGasolina + = monto;
            }
}
```

Ejercicio I

- Implementar y especializar las subclases
  - Secretaria
  - Ingeniero
  - Vendedor

Ejercicio II-a

```
class B {
        private void metodoPrivado( ) {
                System.out.println( "Hola desde Privado" );
        }
        public void metodoPublico( ) {
                metodoPrivado( );
        }
}
```

```
class A extends B {  }
```

```
public class TestA {
        public static void main( String[] args ) {
            A a = new A( );
            a.metodoPublico( );
        }
}
```

Ejercicio II-b

```
class B {
        private void metodoPrivado( ) {
                System.out.println( "Hola desde privado" );
        }
}
```

```
public class TestA {
        public static void main( String[] args ) {
            A a = new A( );
            a.metodoPublico( );
        }
}
```

```
class A extends B {
        public void metodoPublico( ) {
                metodoPrivado( );
        }
}
```

Herencia:
Sobreescritura de métodos

Sobreescritura de métodos

- Redefinir el comportamiento de una subclase
- El método sobrescrito debe cumplir
  - Lista de argumentos y tipo de retorno deben ser iguales al método sobrescrito
  - El nivel de acceso del nuevo método debe ser **igual o menos restrictivo** que el método sobrescrito
- Los métodos ***private*** no se heredan, por lo tanto no se pueden sobrescribir
- Los constructores no se heredan

Sobreescritura de métodos II

```
public class Animal {
    public void comer() {
        System.out.println("Animal comiendo...");
     }
    public void correr() {
        System.out.println("Animal corriendo...");
     }
    public void respirar(){
        System.out.println("Animal respirando…");
    }
}
```

Sobreescritura de métodos III

```
public class Caballo extends Animal {
    public void comer( ) {
        System.out.println("Caballo comiendo heno...");
    }

    public void trotar( ) {
        System.out.println("Caballo trotando...");
     }
}
```

Sobreescritura de métodos IV

```
public class TestAnimales {
    public static void main (String[] args){
        Animal  a = new Animal();
        Caballo b = new Caballo();
        a.comer();
        a.correr();
        b.comer();
        b.correr();
        b.respirar();
      }
}
```

Ejercicio III

1. Modificar la clase Animal, agregar los siguientes comportamientos genéricos:
   dormir()
   atacar()

2. Redefinir los comportamientos anteriores para las especializaciones de animales
   Tiburon
   Venado

3. Modificar la clase *TestAnimales* y demostrar el uso

Ejercicio IV

- Diseñar la clase genérica Figura con métodos:
  - getArea → Calcula el área de la figura
  - getPerimetro → Calcula el perímetro
  - Dibujar → Dibuja figura usando asteriscos
- Un atributo tendrá el nombre de la figura
- Cada subclases creada (*MCuadrado*, *Cuadrado*, *Rectangulo* y *Circulo*) especializa cada método según su forma particular
  - Área y perímetro se determina según la figura

Ejercicio IV-a

MCuadrado de 5x5

Nombre Figura:
“mitad de un cuadro”

*****

****

***

**

*

A= 12.5

P= 17.071

Circulo de radio 4

Nombre Figura: “círculo”

“No se puede graficar en modo textual”

A= 50.26

P= 25,13

Ejercicio IV-b

Cuadrado de lado = 3

Nombre Figura: “cuadro”

***

***

***

A= 9

P= 12

Rectangulo de 4 x 5

Nombre Figura: “rectangulo”

****

****

****

****

****

A= 20

P= 18