UD. 11- Subclases.

Programación Nuria Fuentes Pérez

CENTRO FP SUPERIOR
CAMPUS CÁMARA COMERCIO SEVILLA

Índice

1. Clases y sus miembros.
2. Qué se hereda de una clase.
3. Ejemplo de herencia.
4. Miembros con los mismos nombres.
5. Redefinir o sobrescribir métodos.
6. Constructores en una jerarquía.

1. Definición de una clase

```
class nombreClase {
   // cuerpo de la clase
}
```

La sintaxis completa sería:

```
[acceso] [tipoDeClase] class nombreClase [superclase] [interface] {
   [lista de atributos]
   [lista de métodos]
}
```

1. Definición de una clase

- [acceso]
  - ✓ ***public***: puede ser usada por cualquier clase en cualquier paquete.
  - ✓Nada (paquete): sólo puede ser usada por las clases del mismo paquete.
- [tipoDeClase]
  - ✓***final***: es aquella que no puede tener clases que la herede.
  - ✓***abstract***: es una clase que puede tener herederos pero no puede ser instanciada. Sirven para modelar conceptos.

1. Definición de una clase

- [superclase]
  - ✓ **extends** *ClaseMadre*
- [Interface]
  - ✓ **implements** *Interface1[,Interface2]…*

1. Miembros de una clase

- [acceso]
  - ✓ ***public***
  - ✓ ***private***
  - ✓ ***protected***
  - ✓ Nada

ACCESO DESDE:	private	protected	public	nada
La propia clase				
Subclase en el mismo paquete				
Otra clase en el mismo paquete				
Subclase en otro paquete				
Otra clase en otro paquete				

S	Puede acceder
N	NO puede acceder

2. ¿Qué se hereda en una clase?

• Una subclase hereda todos los miembros de su superclase, excepto los constructores, lo que significa que tiene acceso a todos los miembros.
• Una subclase no tiene acceso directo a los miembros privados (private) de su superclase.
• Una subclase sí puede acceder directamente a los miembros públicos (public) y protegidos (protected) de su superclase, y en el caso de que pertenezca al mismo paquete de su superclase, también puede acceder a los miembros predeterminados.
• Una subclase puede añadir sus propios atributos y métodos. Si el nombre de alguno de estos miembros coincide con el de un miembro heredado, este último queda oculto para la subclase (ya no puede acceder a él). Evidentemente siempre que fuera un miembro al que podía acceder.
• Los miembros heredados por una subclase pueden, a su vez, ser heredados por más subclases de ella. (propagación de herencia).

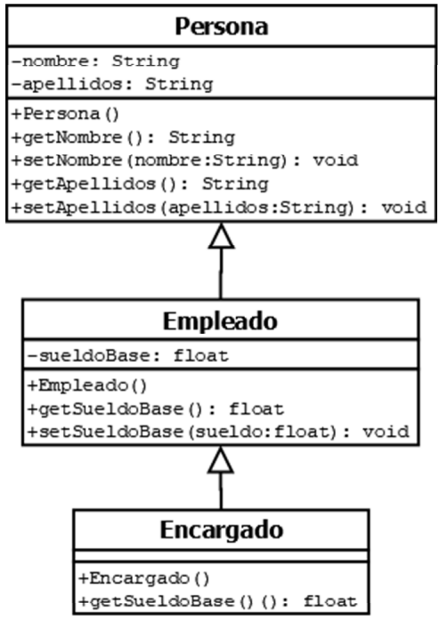
3. Ejemplo

Persona:
Objeto que se caracteriza por tener un nombre y apellidos.

Empleado:
Es una Persona que tiene un sueldo base

Encargado:
Es un Empleado que cobra el 10% más de su sueldo como Empleado

3. Ejemplo


Persona
-nombre: String
-apellidos: String
+Persona()
+getNombre(): String
+setNombre(nombre:String): void
+getApellidos(): String
+setApellidos(apellidos:String): void
Empleado
-sueldoBase: float
+Empleado()
+getSueldoBase(): float
+setSueldoBase(sueldo:float): void
Encargado
+Encargado()
+getSueldoBase(): float

4. ¿Qué ocurrirá si definimos…?

… algún atributo con el mismo nombre?

```
class ClaseA{

  public int atributoX = 1;

  public int metodoX(){
    return atributoX * 10;
  }
  public int metodoY(){
    return atributoX + 100;
  }
}
```

```
class ClaseB extends ClaseA{

  protected int atributoX = 2;

  public int metodoX(){
    return atributoX * -10;
  }
}
```

4. ¿Qué ocurrirá si definimos…?

Aplicación

```
public class AmismoAtributo{
  public static void main(String[] args){
    ClaseB objClaseB = new ClaseB();
    System.out.println(objClaseB.atributoX);
    System.out.println(objClaseB.metodoY());
    System.out.println(objClaseB.metodoX());
  }//Fin Programa
}
```

4. ¿Qué ocurrirá si definimos…?

… algún atributo con el mismo nombre?

¿Y si quisiéramos acceder al atributo de la superclase desde la subclase?

- Cambiamos el nombre:
- Usamos **super**.

4. ¿Qué ocurrirá si definimos…?

… algún atributo con el mismo nombre?

```
class ClaseA{

  public int atributoX = 1;

  public int metodoX(){
    return atributoX * 10;
  }
  public int metodoY(){
    return atributoX + 100;
  }
}
```

```
class ClaseB extends ClaseA{

  protected int atributoX = 2;

  public int metodoX(){
    return super.atributoX * -10;
  }
}
```

5. Redefinir métodos en la subclase

Redefinir o sobrescribir…

Definido en Superclase como:	Se podría redefinir en Subclase como:
protected	protected o public
public	public
private	No tiene sentido pues no puede acceder a él.

5. Redefinir métodos en la subclase

Redefinir o sobrescribir…

```
class ClaseA{
  public int atributoX = 1;

  public int metodoX(){
    return atributoX * 10;
  }
  public int metodoY(){
    return atributoX + 100;
  }
}
```

```
class ClaseB extends ClaseA{
  protected int atributoX = 2;

  public int metodoX(){
    return atributoX * -10;
  }
  public int metodoZ(){
    atributoX = super.atributoX + 3;
    return super.metodoX() + atributoX;
  }
}
```

5. Redefinir métodos en la subclase

```
public class AmismoMétodo{
  public static void main(String[] args){
    ClaseB objClaseB = new ClaseB();
    System.out.println(objClaseB.atributoX);
    System.out.println(objClaseB.metodoY());
    System.out.println(objClaseB.metodoX());
    System.out.println(objClaseB.metodoZ());
  }
}
```

6. Constructores en una jerarquía

Encargado objEncargado=new Encargado();

public Encargado(){

}

public Empleado(){

}

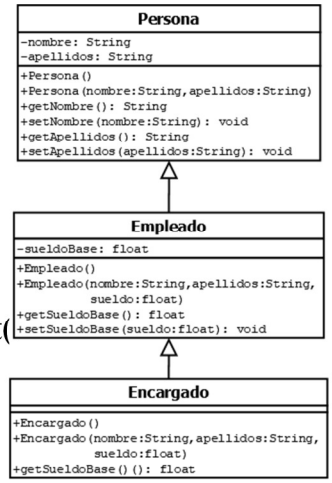
public Persona(){

}

public Object(

}

Añadir constructores con parámetros a la jerarquía Persona



6. Constructores en una jerarquía

```
public nombreSubclase(lista de parámetros){
    super(lista de parámetros);
    //cuerpo del constructor de la subclase
}
```