

## 7.C. Herencia.

### 1. Herencia.

#### 1.6. Ampliación de métodos heredados.

Hasta ahora, has visto que para **redefinir** o **sustituir un método** de una **superclase** es suficiente con crear otro método en la **subclase** que **tenga el mismo nombre** que el método que se desea **sobrescribir**. Pero, en otras ocasiones, puede que lo que necesites no sea sustituir completamente el comportamiento del método de la superclase, sino simplemente **ampliarlo**.

Para poder hacer esto necesitas poder **preservar el comportamiento antiguo** (el de la **superclase**) y **añadir el nuevo** (el de la **subclase**). Para ello, puedes invocar desde el método **"ampliador"** de la **clase derivada** al método **"ampliado"** de la clase superior (teniendo ambos **métodos el mismo nombre**). ¿Cómo se puede conseguir eso? Puedes hacerlo mediante el uso de la referencia **super**.

La palabra reservada **super** es una referencia a la **clase padre** de la clase en la que te encuentres en cada momento (es algo similar a **this**, que representaba una referencia a la **clase actual**). De esta manera, **podrías invocar a cualquier método de tu superclase** (si es que se tiene acceso a él).

Por ejemplo, imagina que la clase **Persona** dispone de un método que permite mostrar el contenido de algunos datos personales de los objetos de este tipo (**nombre**, **apellidos**, etc.). Por otro lado, la clase **Alumno** también necesita un método similar, pero que muestre también su información especializada (**grupo**, **nota media**, etc.). ¿Cómo podrías aprovechar el método de la **superclase** para no tener que volver a **escribir su contenido en la subclase**?

Podría hacerse de una manera tan sencilla como la siguiente:

```
public void mostrar () {  
  
    super.mostrar (); // Llamada al método "mostrar" de la superclase  
  
    // A continuación mostramos la información "especializada" de esta subclase  
  
    System.out.printf ("Grupo: %s\n", this.grupo);  
  
    System.out.printf ("Nota media: %5.2f\n", this.notaMedia);  
  
}
```

Este tipo de **ampliaciones de métodos** resultan especialmente útiles por ejemplo en el caso de los **constructores**, donde se podría ir llamando a los **constructores** de cada **superclase** encadenadamente hasta el **constructor** de la clase en la **cúspide de la jerarquía** (el **constructor** de la clase **Object**).

#### Ejercicio resuelto

Dadas las clases **Persona**, **Alumno** y **Profesor**, define un método **mostrar** para la clase **Persona**, que muestre el contenido de los atributos (datos personales) de un objeto de la clase **Persona**. A continuación, define sendos métodos **mostrar** especializados para las clases **Alumno** y **Profesor** que **"amplíen"** la funcionalidad del método **mostrar** original de la clase **Persona**.

#### Solución:

1. Método **mostrar** de la clase **Persona**.

```
public void mostrar () {  
  
    SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");  
  
    String Stringfecha= formatoFecha.format(this.fechaNacim.getTime());  
  
  
    System.out.printf ("Nombre: %s\n", this.nombre);  
  
    System.out.printf ("Apellidos: %s\n", this.apellidos);  
  
    System.out.printf ("Fecha de nacimiento: %s\n", Stringfecha);  
  
}
```

```
}
```

## 2. Método **mostrar** de la clase **Profesor**.

Llamamos al método **mostrar** de su **clase padre (Persona)** y luego añadimos la **funcionalidad específica** para la **subclase Profesor**:

```
public void mostrar () {  
  
    super.mostrar (); // Llamada al método "mostrar" de la superclase  
  
    // A continuación mostramos la información "especializada" de esta subclase  
  
    System.out.printf ("Especialidad: %s\n", this.especialidad);  
  
    System.out.printf ("Salario: %7.2f euros\n", this.salario);  
  
}
```

## 3. Método **mostrar** de la clase **Alumno**.

Llamamos al método **mostrar** de su **clase padre (Persona)** y luego añadimos la **funcionalidad específica** para la **subclase Alumno**:

```
public void mostrar () {  
  
    super.mostrar ();  
  
    // A continuación mostramos la información "especializada" de esta subclase  
  
    System.out.printf ("Grupo: %s\n", this.grupo);  
  
    System.out.printf ("Nota media: %5.2f\n", this.notaMedia);  
  
}
```