7.C. Herencia.

1. Herencia.

1.7. Constructores y herencia.

Recuerda que cuando estudiaste los **constructores** viste que un **constructor** de una clase puede llamar a otro **constructor** de la misma clase, previamente definido, a través de la referencia **`this`**. En estos casos, la utilización de **`this`** sólo podía hacerse en la primera línea de código del **constructor**.

Como ya has visto, un **constructor** de una **clase derivada** puede hacer algo parecido para llamar al **constructor** de su **clase base** mediante el uso de la palabra **`super`**. De esta manera, el **constructor** de una **clase derivada** puede llamar primero al **constructor** de su **superclase** para que inicialice los **atributos heredados** y posteriormente se inicializarán los **atributos específicos** de la clase: los no heredados. Nuevamente, esta llamada también **debe ser la primera sentencia de un constructor** (con la única excepción de que exista una llamada a otro constructor de la clase mediante **`this`**).

Si no se incluye una llamada a **`super()`** dentro del **constructor**, el compilador incluye automáticamente una llamada al constructor por defecto de **clase base** (llamada a `super()`). Esto da lugar a una **llamada en cadena de constructores de superclase** hasta llegar a la clase más alta de la jerarquía (que en Java es la clase **`Object`**).

En el caso del **constructor por defecto** (el que crea el compilador si el programador no ha escrito ninguno), el compilador añade lo primero de todo, antes de la inicialización de los atributos a sus valores por defecto, una llamada al constructor de la **clase base** mediante la referencia **`super`**.

A la hora de destruir un objeto (método **`finalize`**) es importante llamar a los finalizadores en el **orden inverso** a como fueron llamados los constructores (p**rimero se liberan los recursos de la clase derivada y después los de la clase base** mediante la llamada **`super.finalize()`**).

Si la clase **`Persona`** tuviera un constructor de este tipo:

```
public Persona (String nombre, String apellidos, GregorianCalendar fechaNacim) {
    this.nombe= nombre;
    this.apellidos= apellidos;
    this.fechaNacim= new GregorianCalendar (fechaNacim);
}
```

Podrías llamarlo desde un constructor de una clase derivada (por ejemplo **`Alumno`**) de la siguiente forma:

```
public Alumno (String nombre, String apellidos, GregorianCalendar fechaNacim, String grupo, double notaMedia) {
    super (nombre, apellidos, fechaNacim);
    this.grupo= grupo;
    this.notaMedia= notaMedia;
}
```

En realidad se trata de otro recurso más para optimizar la **reutilización de código**, en este caso el del **constructor**, que aunque no es heredado, sí puedes invocarlo para no tener que rescribirlo.

Autoevaluación

Puede invocarse al constructor de una superclase mediante el uso de la referencia `this`. ¿Verdadero o Falso?

- ○ Verdadero
- ○ Falso

Ejercicio resuelto

Escribe un constructor para la clase `Profesor` que realice una llamada al constructor de su clase base para inicializar sus atributos heredados. Los atributos específicos (no heredados) sí deberán ser inicializados en el propio constructor de la clase `Profesor`.

Solución:

```
public Profesor (String nombre, String apellidos, GregorianCalendar fechaNacim, String especialidad, double salario) {
    super (nombre, apellidos, fechaNacim);
    this.especialidad= especialidad;
    this.salario= salario;
}
```