Clases y objetos

Adolfo Sanz De Diego

Junio de 2011

# Conceptos

**Clase**

* Conjunto propiedades y métodos que modifican (o no) dichas propiedades.

**Objeto**

* Instancia de una clase.

# El modificador 'static'

Asocia una propiedad o un método a la clase, y no al objeto.

La propiedad o método marcado como **static** se 'comparte' entre todas las instancias de la clase.

# El modificador 'final'

**Clase**

* No permite que se herede de ella.

**Propiedad**

* No permite que se modifique.

**Método**

* No permite que se sobrescriba en clases hijas.

# El modificador 'abstract'

**Clase**

* La convierte en abstracta obligando a sus clases hijas a sobrescribirlos métodos abstractos.

**Método**

* Dentro de una clase abstracta los distintos métodos que las clases hijas deben de sobreescribir obligatoriamente.

# El modificador 'transient'

**Propiedad**

* Indica que no es parte persistente del objeto (serialización).

# Modificadores de visibilidad

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | La misma | Otra clase del | Subclase de | Otra clase de |
|  |  | clase | mismo paquete | otro paquete | otro paquete |
| \*\* | public\*\* | x | x | x | x |
| \*\* | protected\*\* | x | x | x |  |
| \*\* | default\*\* | x | x |  |  |
| \*\* | private\*\* | x |  |  |  |

# Constructores

Métodos que se llaman igual que la propia clase.

Si no hay ninguno se aplica el por defecto.

Puede haber uno o varios.

Pueden tener o no parámetros.

Para instanciar:

* Objeto o = new Objeto();

Las clases abstractas no se pueden instanciar.

# Destructores

No existen destructores.

El **recolector de basura**:

* Es quien se encarga de destruir objetos.
* Es un mecanismo complicado.
* De forma sencilla: destruye los objetos que ya no son referenciados en ningún sitio.
* Problemas: ej. referenciar en un Array objetos que ya no se usan.
* Soluciones: ej. Objeto o = null; (pero con esto sólo borras una referencia y puede haber más)

# Inversión de Dependencias

Los términos fueron acuñados por **Martin Fowler** en el 2004.

Intentan reducir el acoplamiento entre componentes.

**Inyección de dependencias**

* Es un **patrón de diseño** orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase quien cree el objeto.

**Inversión de control**

* Implementación del **Principio de Hollywood** cuyo nombre proviene de las típicas respuestas que se les dan a los actores amateurs: **no nos llames; nosotros te llamaremos**.
* En este caso es la libreria la que llama al código y no al revés.
* Consiste en delegar la gestión de las instancias en un **contenedor**.
* Ejemplos: **Spring** y **Google Guice**.