

# Java OCA Capítulo 1

## Entendiendo la estructura de una clase Java

### comentarios

- 1 línea:

```
//esto es un comentario de una sola línea
```

- múltiples líneas

```
/* un comentario
 * con múltiples
 * líneas
 */
```

- javadoc:

```
/** un comentario javadoc
 * @author Miguel Angel Torres
 */
```

### clases VS ficheros

- cada clase java se define en su propio fichero .java
- el nombre de la clase ha de coincidir con el nombre del fichero
- puedes poner 2 o más clases, pero solo 1 puede ser pública y esta es la que ha de coincidir en nombre con el fichero

## escribiendo un método main()

- el método main() es el puente entre el proceso de arranque de Java, gestionado por la JVM<sup>1)</sup> y el código Java del programador
- compilación y ejecución:

```
$ javac MiClase.java
$ java MiClase
```

- para compilar es necesario:
  - que el fichero tenga extensión .java
  - que el nombre de la clase coincida con el nombre del fichero (case sensitive)
  - esto genera un código compilado bytecode, interpretable por cualquier JVM (multiplataforma)
- para ejecutar el código compilado, se omite la extensión (.class)
- public static void main(String[] args)
  - String[] args -> String args[] -> String... args son expresiones equivalentes

## Declaraciones de paquetes e importaciones

- Java pone las clases en paquetes (package), agrupaciones lógicas.
- `import` le dice a Java en que paquetes ha de buscar clases
- los paquetes siguen un orden jerárquico de carpetas
- si un paquete empieza por `java` o `javax`, son clases que pertenecen al JDK del propio Java
- los nombres de los paquetes pueden contener números y letras
  - *en el examen OCA no se usarán nombres de paquetes inválidos (aunque si nombres de variables inválidos)*
- las clases del paquete `java.lang` se incluyen automáticamente sin tener que realizar un `import` de dicho paquete

### comodines (wildcards)

- se puede usar un *shortcut* de `import`, usando comodines (\*) para importar todas las clases de un paquete

```
import java.util.*;
```

### importaciones redundantes

```
import java.lang.System; // redundante, java.lang se carga automáticamente
import java.util.Random
import java.util.* // Esta o la anterior son redundantes... con esta última
incluimos Random y todas las de java.util (no a la inversa)
```

### conflictos de nombres

#### creando un nuevo paquete

#### Formateo de código en el examen

## Creando objetos

### constructores

### leyendo y escribiendo variables de objeto

### bloques inicializadores de instancia (Instance Initializer Blocks)

## Diferencia entre referencias a objetos y primitivas

### tipos primitivos

**tipos de referencia**

**principales diferencias**

## **Declarando e inicializando variables**

**declaraciones múltiple de variables**

**identificadores**

## **Entendiendo alcance de las variables**

## **Ordenando los elementos de una clase**

## **Destruyendo objetos**

**garbage collection**

**finalize()**

<sup>1)</sup>

Java Virtual Machine

From:

<https://miguelangel.torresegea.es/wiki/> - miguel angel torres egea



Permanent link:

<https://miguelangel.torresegea.es/wiki/info:libros:javaoca:cap1?rev=1524915689>

Last update: **28/04/2018 04:41**