



IES Sant Vicent Ferrer
Algemés



Programación

UD 2: Introducción a JAVA

- Estilo de Codificación -

Estilo de Codificación

1.- Introducción

2.- Normas y estilo de codificación

2.1.- Nombres de ficheros

2.2.- Organización de ficheros

2.3.- Identación

2.4.- Comentarios

2.5.- Declaraciones

2.6.- Sentencias

2.7.- Separaciones

2.8.- Nombres

1.- Introducción

1.- Introducción

Una vez realizado el diseño se realiza el proceso de codificación.

En esta etapa, el programador recibe las especificaciones del diseño y las transforma en un conjunto de instrucciones escritas en un lenguaje de programación.

A este conjunto de instrucciones se le llama **código fuente**.

En cualquier proyecto en el que trabaja un grupo de personas debe haber unas **normas de codificación y estilo**, claras y homogéneas.

Estas normas facilitan las tareas de **corrección y mantenimiento** de los programas, sobre todo cuando se realizan por personas que no los han desarrollado.

1.- Introducción

```
import java.util.Scanner;

public class Suma {

    public static void main (String[] args){
        int suma = 0;
        int contador = 0;

        while (contador < 10) {
            contador++;
            suma = suma + contador;
        }
        System.out.println("Suma => " + suma);
    }
}
```

good code
is like a
good joke
- it needs no explanation

```
import java.util.Scanner;
public class
Suma {
    public static

void main (String[] args){
    int suma = 0; int contador = 0;
    while (contador < 10)
    {    contador++;

    suma = suma + contador; }
    System.out.println("Suma => " + suma);
    }
}
```

2.- Normas y estilo de codificación

2.- Normas y estilo de codificación

2.1.- Nombres de ficheros:

La extensión para los ficheros de código fuente es *.java*

La extensión para los ficheros compilados es *.class*

2.- Normas y estilo de codificación

2.2.- Organización de ficheros:

Cada fichero debe contener una sola clase pública y debe ser la primera.

Las clases privadas e interfaces asociados con esa clase pública se pueden poner en el mismo fichero después de la clase pública.

Las secciones en las que se divide el fichero son:

- Comentarios
- Sentencias del tipo *package* e *import*
- Declaraciones de clases e interfaces

2.- Normas y estilo de codificación

2.2.- Organización de ficheros:

Comentarios:

Todos los ficheros fuente deben comenzar con un comentario que muestre:

- El nombre de la clase
- Información de la versión
- La fecha de creación y/o modificación
- Aviso de derechos de autor

```
/**  
 * Nombre de la clase  
 *  
 * @version  
 *  
 * @since  
 *  
 * @author  
 *  
 */
```

2.- Normas y estilo de codificación

2.2.- Organización de ficheros:

Package e import:

Van después de los comentarios, la sentencia package va delante de import.

Ejemplo:

```
package paquete.ejemplo;  
  
import java.util.ArrayList;
```

2.- Normas y estilo de codificación

2.2.- Organización de ficheros:

Clases e interfaces:

Comentario de documentación (`/** ... */`)

Sentencia *class* o *interface*

Variables estáticas, en este orden: públicas, protegidas y luego privadas.

Variables de instancia, en este orden: públicas, protegidas y luego privadas.

Constructores

Métodos. Se agrupan por su funcionalidad, no por su alcance.

2.- Normas y estilo de codificación

2.3.- Identación

Como norma general se usarán cuatro espacios.

La longitud de las líneas de código no debe superar 80 caracteres.

La longitud de las líneas de comentarios no debe superar 70 caracteres.

Cuando una expresión no cabe en una sola línea: romper después de una coma, romper antes de un operador, alinear la nueva línea al principio de la anterior.

IMPORTANTE:

ctrl + i (en Eclipse)

2.- Normas y estilo de codificación

2.4.- Comentarios

Los comentarios deben contener solo la información que es relevante para la lectura y la comprensión del programa.

Existen 2 tipos de comentarios:

De documentación: Están destinados a describir la especificación del código. Se utilizan para describir las clases Java, las interfaces, los constructores, los métodos, ...

De implementación: Son para comentar algo acerca de la aplicación particular, de qué está realizando el algoritmo, ...

2.- Normas y estilo de codificación

2.5.- Declaraciones

Se recomienda declarar una variable por línea.

Inicializar las variables locales donde están declaradas y colocarlas al comienzo del bloque.

En las clases e interfaces:

- No se ponen espacios en blanco entre el nombre del método y el paréntesis “(”.
- La llave de apertura “{” se coloca en la misma línea que el nombre del método o clase.
- La llave de cierre “}” aparece en una línea aparte.

2.- Normas y estilo de codificación

2.6.- Sentencias

Cada línea debe contener una sentencia.

Si hay un bloque de sentencias, este debe ser sangrado con respecto a la sentencia que lo genera y debe estar entre llaves aunque solo tenga una sentencia.

2.- Normas y estilo de codificación

2.7.- Separaciones

Mejoran la legibilidad del código. Se utilizan:

Dos líneas en blanco: entre las definiciones de clases e interfaces.

Una línea en blanco: entre los métodos, la definición de las variables locales de un método y la primera instrucción, antes de un comentario, entre secciones lógicas de un método para mejorar la legibilidad.

Un carácter en blanco: entre una palabra y un paréntesis, después de una coma, los operadores binarios menos el punto, las expresiones del for, y entre un cast y la variable.

2.- Normas y estilo de codificación

2.8.- Nombres

Los nombres de las variables, métodos, clases, etc., hacen que los programas sean más fáciles de leer ya que pueden darnos información acerca de su función.

Las normas para asignar nombres son las siguientes:

2.- Normas y estilo de codificación

2.8.- Nombres

Paquetes: El nombre se escribe en **minúscula**, se pueden utilizar puntos para reflejar algún tipo de jerarquía. Ejemplo: java.io

Clases e interfaces: Los nombres deben ser **sustantivos**. Se deben utilizar nombres descriptivos. La primera letra siempre será en **mayúscula**.
Ejemplo: Clientes

Métodos*: Se deben usar **verbos en infinitivo**. Ejemplo: asignarDestino()

Variables*: Deben ser **cortas y significativas**. Ejemplo: sumaTotal.

Constantes: El nombre debe ser **descriptivo**. Se escriben en **mayúsculas** y si son varias palabras unidas por subrayado. Ejemplo: MAX_VALOR

**ver siguiente diapositiva*

2.- Normas y estilo de codificación

2.8.- Nombres

Si las variables o métodos tienen varias palabras para hacerlas mas entendibles, se puede utilizar una de las siguientes 3 notaciones:

CamelCase

camelCase

snake_case

kebab-case: no se puede utilizar en Java

RESUMEN

	Regla	Ejemplo
paquetes	todo en minúscula	dominio
clases	empiezan con Mayúscula en singular	Cliente
variables	en minúsculas (camelCase)	cantidadTotal
constantes	en mayúsculas (snake_case)	VALOR_PTAS
métodos	en minúsculas verbos en infinitivo	asignarDestino()

En instrucciones

Una instrucción por línea.

Líneas de separación antes condicionales, bucles, comentarios y métodos.

En expresiones

Poner espacios entre las variables y los operadores;

En Eclipse:

ctrl + i -> indentar el código

ctrl + mayus + f -> separación entre expresiones (espacios en blanco e intros)

Bibliografía

Bibliografía

Buenas Prácticas de Codificación

<https://github.com/kosme10/standards/wiki/Buenas-Practicas-de-codificacion>

Java - Estándares de programación

<http://javafoundations.blogspot.com.es/2010/07/java-estandares-de-programacion.html>