# Clase 1: Comentarios e Identificadores

## 1.1 Comentarios

### ¿Qué son los comentarios?

Los comentarios son anotaciones de texto que se escriben entre las líneas de código de los código fuente de los programas.

Tienen la finalidad de ofrecer aclaraciones e indicaciones sobre ciertas líneas de código y el proyecto en general.

Es importante saber que los comentarios son escritos específicamente para los desarrolladores que leen el código fuente de los programas ya que el compilador y la máquina virtual de Java no leen ni compilan los comentarios, es decir, los pasan por alto.

### ¿Por qué son importante los comentarios?

Un software siempre estará compuesto por cientos sino miles líneas de código. Muchas veces esas líneas de código pueden ser expresiones complejas por lo que los comentarios nos ayudan a entender ese código con mayor facilidad.

La lógica de programación puede ser diferente entre desarrolladores. Para lo que un desarrollador es completamente lógico y sumamente claro de entender, no necesariamente lo será para otro desarrollador. Por lo que los comentarios son convenientes para entender el código escrito por otro desarrollador. Además, en el campo laboral todos los proyectos se hacen en equipos de trabajo por lo que constantemente tendremos que leer código escrito por otro desarrollador.

### Tipos de comentarios en Java

1. **Comentarios de línea**
   - También llamado comentario de fin de línea.
   - Se utiliza escribiendo los caracteres (//) en nuestra línea de código.

```
1  //Este es un comentario de línea
```

   - Son breves anotaciones para dar indicaciones sencillas.
   - Ocupan solamente una (1) línea de código.
   - No necesita empezar una línea; también puede estar en medio de ella y continuar hasta el final.

*Leído por el compilador* *Ignorado por el compilador*

```
1  int num = 10; //este es un comentario de línea
```

2. **Comentarios de bloque**
   - También llamados comentarios tradicionales (También utilizado en otros lenguajes de programación).
   - Se utiliza escribiendo los caracteres (/*) como delimitador de inicio y los caracteres (*/) como delimitador final.
   - Pueden ocupar una (1) línea de código como muchas líneas de código.

```
1  /*Comentario de bloque de (1) línea*/
```

```
2
3  /*Comentario de bloque que ocupa
4  tres (3) líneas
5  de código*/
```

- Todo lo que se encuentre entre el delimitador de inicio (/*) y el delimitador de final (*/) el compilador lo tomará como comentario y lo ignorará, incluso si hay código entre los delimitadores

```
1  /*Inicio del comentario de bloque
2
3  int num = 10;
4
5  Fin del comentario de bloque*/
```

En la línea 3 estamos declarando una variable de tipo entero llamada num y la estamos inicializando con un valor de 10. Si bien esta línea de código contiene instrucciones java válidas, como se encuentra dentro de los limitadores del comentario de bloque: /* (al inicio de la línea 1) y */ (al final de la línea 5), el compilador de java lo ignora ya que lo toma como un comentario.

**3. Comentarios JavaDoc**

- Se utiliza escribiendo los caracteres (/**) como delimitador de inicio y los caracteres (*/) como delimitador final.
- Estos comentarios nos permiten incrustar la documentación de manera directa en nuestros programas dándole instrucciones al propio sistema.
- Al utilizarlos se genera automáticamente una documentación técnica y legible en HTML a partir del código fuente.
- Se utiliza como estándar para documentar clases en Java.
- Además de los delimitadores de inicio y final, se utilizan etiquetas que se abren utilizando el carácter (@).

```
1  /**
2   * Ejemplo: Nombre de la Clase
3   * @author: Mario R. Rancel
4   * @version: 22/09/2016/A
5   */
```

- Si bien los JavaDoc son los comentarios mas importantes a la hora de presentar un proyecto profesional, es un concepto avanzado que podría enredar a los que están apenas comenzando a estudiar Java por lo que se podrá explicar en un futuro cuando los conceptos básicos de Java y POO estén más claros. Por ahora solo debes saber que existen y son importantes.

## 1.2 Identificadores

### ¿Qué son los identificadores?

Los identificadores son los nombres que los desarrolladores asignan a variables, constantes, clases, métodos, etc. de un programa.

Toda la información dentro de nuestro programa Java debe tener un nombre para poder identificarlo.

Para utilizar identificadores se debe seguir las siguientes reglas:

1. Pueden estar formados por letras, números, guiones bajos (_) y signos de moneda ($).
   `Bienvenido1`, `$valor`, `_valor`, `m_campoEntrada1`, `boton7`.
2. No pueden empezar por un número.
   `7boton`, `1num`, `8valor`.
3. No pueden tener espacios en blanco entre los caracteres del identificador (el compilador los consideraría dos identificadores distintos).
   `nombre completo`, `num uno`, `valor nuevo`.
4. No pueden tener caracteres especiales distintos a (_) y ($).
   `var#`, `Andy´s`, `año-nacimiento`, `precio:final`.
5. No pueden ser una palabra reservada de Java.
6. El carácter (ñ) y las tildes (´) sí son permitidas sin embargo **NO SE RECOMIENDA SU USO**.
   **año**, **canción**, **árbol**, **mañana**.
7. Java es sensible a mayúsculas y minúsculas, este punto es sumamente clave ya que dos identificadores pueden llamarse igual, pero si uno comienza con mayúscula y otro con minúscula no son iguales.
   `nombre ≠ Nombre`, `num1 ≠ Num1`, `status ≠ Status`.

### Palabras Reservadas en Java

Son identificadores predefinidos que tienen un significado específico para el compilador de java y por lo tanto no pueden usarse como identificadores creados por los desarrolladores.

Estas son todas las palabras reservadas en Java:

| | | | | |
|---|---|---|---|---|
| abstract | continue | for | new | switch |
| assert | default | goto | package | synchronized |
| boolean | do | if | private | this |
| break | double | implements | protected | throw |
| byte | else | import | public | throws |
| case | enum | instanceof | return | transient |
| catch | extends | int | short | try |
| char | final | interface | static | void |
| class | finally | long | strictfp | volatile |
| const | float | native | super | while |

Las palabras `true`, `false` y `null` no son palabras reservadas en Java sin embargo son valores literales y tampoco se pueden utilizar como identificadores.

## Técnica de naming: Camel Case

Los lenguajes de programación emplean distintas convenciones para la implementación de identificadores.

En Java utilizamos dos convenciones referentes al nombramiento de identificadores: Upper Camel Case y Lower Camel Case.

Llevan su nombre debido a su similitud con las de un camello y sus jorobas

camelCase

## Upper Camel Case

- Se implementa cuando la primera letra de la primera palabra va en mayúscula y la primera letra de las siguientes palabras van en mayúscula también.
- Debemos usar Upper Camel Case en el nombramiento de clases e interfaces.

```
1  // Upper Camel Case:
2  class SoyUnaClase {}
```

## Lower Camel Case

- Se implementa cuando la primera letra de la primera palabra va en minúscula y la primera letra de las siguientes palabras va en minúscula.
- Debemos usar Lower Camel Case en el nombramiento de variables, métodos, objetos.

```
1  // Lower Camel Case
2  int soyUnNumeroInt = 10;
```

A pesar de que el uso del camel case suene como algo muy sencillo y sin importancia, es sumamente importante implementar estas buenas prácticas desde un principio y siempre mantenerlas en mente.

Verás que cuando no se usan correctamente se puede diferenciar fácilmente de un desarrollador sin fundamentos básicos a uno que si los tiene.