UNIDAD CENTRA DEL VALLE DEL CAUCA

FACULTAD DE INGENIERÍAS

  INGENIERÍA DE SISTEMAS

TULUÁ- VALLE

2018

Manual del programador

**Contenido**

[1.Objetivo del manual 3](#_Toc529909577)

[2. Organización del código 4](#_Toc529909578)

[2.1 Sentencias Package e Import 4](#_Toc529909579)

[2.2 Declaraciones de Clases e Interfaces 5](#_Toc529909580)

[3. Estructura del código 6](#_Toc529909581)

[3.1 Encabezados de la Clase. 6](#_Toc529909582)

[3.2 Encabezados del Método. 6](#_Toc529909583)

[3.3. Indentación. 6](#_Toc529909584)

[3.4 Espacios en Blanco 7](#_Toc529909585)

[3.5 Llaves y saltos de línea 7](#_Toc529909586)

[3.6 Longitudes de Línea y Saltos de Línea 8](#_Toc529909587)

[4. Granularidad del código (tamaño del método) 10](#_Toc529909588)

[5. Variables 11](#_Toc529909589)

[5.1 Asignaciones de Variables. 11](#_Toc529909590)

[5.2 Inicialización. 11](#_Toc529909591)

[5.3 Uso de variables. 12](#_Toc529909592)

# 1.Objetivo del manual

El siguiente manual tiene como objetivo proveer la información necesaria para el desarrollador, así como la correcta utilización de algunas herramientas y la descripción de recomendaciones básicas de codificación para desarrollar un código más óptimo de cara al rendimiento, la homogeneidad y mantenibilidad del mismo

# 2. Organización del código

Una clase está dividida en varias secciones que deben estar separadas por líneas en blanco o por comentarios que son opcionales para documentar el código.

Deben evitarse las clases de más de 2000 líneas ya que son archivos incómodos de leer y entender. A continuación, se van a describir buenas prácticas de programación enfocadas a Java, cabe resaltar que estas prácticas generales se aplican independientemente del lenguaje de programación que se vaya a utilizar.

Las clases Java deben tener la siguiente ordenación:

• Sentencia package.

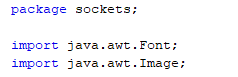
• Sentencias import.

• Declaraciones de clases e interfaces

## 2.1 Sentencias Package e Import

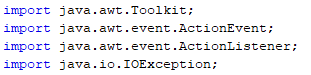
Enumere aquí a sus competidores e incluya los motivos por los que su compañía está bien posicionada frente a cualquier competidor.

La primera línea no-comentario de los archivos Java es la sentencia package. La convención es escribir todos los grupos en minúsculas. Después de ésta, pueden seguir varias sentencias import.



En la sección import, enumere cada módulo importado en forma explícita.

**Estándar**



**No estándar**



## 2.2 Declaraciones de Clases e Interfaces

A continuación, se describe las partes de la declaración de una clase o interfaz, en el orden en que deberían aparecer.

1. Comentario de documentación de la clase o Interfaz.
2. Sentencia class o interface.
3. Comentario de implementación de la clase o interfaz si fuera necesario (/\*...\*/): Este comentario debe contener cualquier información aplicable a toda la clase o interfaz que no fuera apropiada para estar en los comentarios de documentación de la clase o interfaz.
4. Variables de clase (static): Primero las variables de clase pública, después las protegidas, después las de nivel de paquete (sin modificador de acceso), y después las privadas.
5. Variables de instancia: Primero las públicas, después las protegidas, después las de nivel de paquete (sin modificador de acceso), y después las privadas.
6. Bloques estáticos: Si bloques estáticos deben utilizarse fuera de los métodos, los mismos debe utilizarse para la inicialización de variables estáticas y estos mismos deben ubicarse antes de la declaración de constructores.
7. Constructores.
8. Comentario de documentación del método (/\*\*...\*/).
9. Métodos: Estos métodos se deben agrupar por funcionalidad más que por visión o accesibilidad. Por ejemplo, un método de clase privado puede estar entre dos métodos públicos de instancia. El objetivo es hacer el código más legible y comprensible.[[1]](#footnote-1)

# 3. Estructura del código

## 3.1 Encabezados de la Clase.

• Escribir encabezados de clase en una única línea si hubiera espacio disponible.

• De lo contrario, aplique un salto de línea antes de extends e implements Indente las líneas sucesivas.

• Si el encabezado de la clase está en una única línea, coloque la llave de apertura al final de esa línea.

• Si el encabezado de la clase necesita líneas múltiples, coloque la llave de apertura alineada a la izquierda en la siguiente línea.

## 3.2 Encabezados del Método.

• Cada método debe tener como máximo 5 parámetros, preferentemente agrupar parámetros en objetos.

• Escribir encabezados de método en una única línea si hubiera espacio disponible.

• De lo contrario, aplique un salto de línea inmediatamente después de la apertura del paréntesis. Esto deja a todos los parámetros en la misma línea.

• Si aún no hubiera espacio suficiente, coloque cada parámetro en su propia línea.

• Si el encabezado del método está en una única línea, coloque la llave de apertura al final de esa línea.

• Si el encabezado del método necesita líneas múltiples, coloque la llave de apertura alineada a la izquierda en siguiente línea.

## 3.3. Indentación.

• La indentación estándar debe ser de 4 espacios. En realidad, la indentación es un tabulador, que debería estar definida para visualizar cuatro espacios.

• Utilice los tabuladores para la indentación solamente. Cualquier espacio en blanco luego del nivel de indentación deben ser espacios reales, de modo que el formateo será razonable independientemente de la cantidad de espacios a los que un tabulador sea equivalente.

• Si el uso de indentación de 4 espacios no ayuda en la lectura del código, utilice la regla de 8 espacios para indentación en la cabecera de bloque o en la declaración de métodos.

## 3.4 Espacios en Blanco

El espacio en blanco, habitualmente, mejora la legibilidad. Agregue un espacio en los siguientes lugares:

• Entre operadores.

• Luego de la coma en las declaraciones e invocaciones de método.

• Luego de punto y coma en bucles for.

• Antes y después del operador de asignación.

• No coloque espacios en los siguientes lugares:

o Entre el nombre de un método y el paréntesis de apertura

o Alrededor de los paréntesis de apertura y cierre en una declaración o invocación de función

o Alrededor de corchetes de apertura y cierre en la declaración o referencia de arrays.

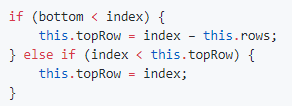
• Utilice las líneas en blanco para separar "párrafos" de líneas de código relacionados.

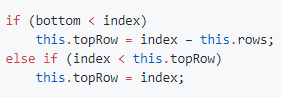
## 3.5 Llaves y saltos de línea

Siempre utilice llaves, incluso para los bloques que contienen sólo una sentencia. Esto elimina una fuente común de bugs y facilita el mantenimiento:

• Puede insertar o eliminar sentencias dentro de un bloque sin preocuparse por agregar o quitar llaves

• Nunca tendrá inconvenientes para relacionar las cláusulas else con las cláusulas if.

**Correcto**  **Incorrecto**



Esta regla se aplica a las siguientes construcciones:

• Bucles for, while y do-while.

• Sentencias if-else.

• Cláusulas try, catch y finally.

• bloques synchronized.

## 3.6 Longitudes de Línea y Saltos de Línea

• Una sentencia por línea.

• Trate de mantener la longitud de la línea en 80 caracteres máximo. Si debe aplicar un salto de línea, aplique la indentación en la(s) línea(s) de continuación.

• Si debe aplicar un salto de línea, hágalo de manera notoria finalizando la primera línea con algo que necesite una continuación:

o Asignaciones de salto de línea luego del operador de asignación.

o Aplique el salto en expresiones aritméticas y lógicas antes de un operador.

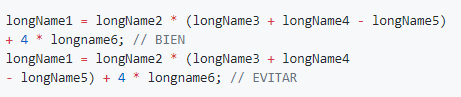
o Aplique el salto de línea para enfatizar sub-expresiones importantes.

o Aplique los saltos en las invocaciones de método luego del paréntesis de apertura. Si la lista del parámetro aún no encaja, aplique un salto entre cada parámetro.

o Aplique los saltos a las declaraciones de método de la misma manera, y coloque la llave de apertura en la línea siguiente, sin identar.

o Si necesita aplicar un salto a las expresiones condicionales (por ej., en sentencias if o while), siga las reglas mencionadas anteriormente, y coloque la llave de apertura en la línea siguiente, sin identar.

**Salto de línea en expresiones aritméticas**



# 4. Granularidad del código (tamaño del método)

• Una cantidad de líneas razonable para un método depende de su complejidad. Un módulo que consista de sentencias secuenciales puede ser más largo que un método que contenga bucles y condiciones complejas. Si el código secuencial es repetitivo, como una inicialización de estructura índice por índice, el método puede ser tan largo como requiera. (Debería, sin embargo, pensar dos veces en el diseño. Tal vez exista una mejor manera de hacerlo)

• Como regla general, evitar métodos con contenido superior a 35 líneas de código “reales” (es decir, sin contar las líneas en blanco que se agregan para mayor legibilidad y los comentarios).

• Preferentemente, un método debería realizar una sola cosa (cohesión), y el nombre del método debería reflejar esto en forma precisa. Si realiza más de una cosa, asegúrese que esto esté reflejado en el nombre del método. Si esto resulta en un nombre feo para el método, vuelva a considerar la estructura de su código. Si tuviera una función llamada initPanelManagerAndReadAccountList, el código probablemente se beneficiaría si se separa en métodos llamados initPanelManager y readAccountList.

# 5. Variables

Utilice solamente una declaración de variable por línea. Esto mejora la lectura y facilita el mantenimiento:

**Correcto Incorrecto**

Todos los campos y variables de clase deben ser privadas.

## 5.1 Asignaciones de Variables.

No asigne el mismo valor a varias variables en la misma sentencia. Es difícil de leer.



No usar asignación embebida como un intento de mejorar el rendimiento en tiempo de ejecución. Ese es el trabajo del compilador.

**Correcto Incorrecto**

## 5.2 Inicialización.

• Todas las variables, incluidas las variables de campos y clases, deben inicializarse, si fuera posible, en su misma declaración. A pesar de que todas las declaraciones de Java tienen valores de inicialización determinados (0, nulo, falso), descríbalo detallada y explícitamente.

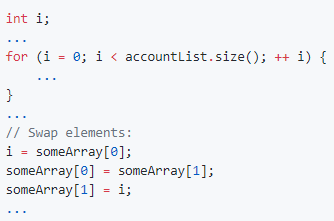
• Java permite la inicialización de arrays usando la misma sintaxis que C y C++, al incluir un conjunto de valores delimitados por una coma y entre llaves. Se permite una coma luego del valor final. Utilice esta alternativa, ya que facilita el mantenimiento: es más fácil agregar valores adicionales o quitarlos del final de la lista.

• Desde Java 1.1, se permiten bloques inicializadores entre las declaraciones. Un bloque de inicialización es una sección de un código entre llaves. Hay dos tipos de bloques de inicialización: estática y de instancia.

## 5.3 Uso de variables.

Utilice siempre una variable para un único propósito. A veces es tentador volver a usar una variable existente; evite esta tentación:

**Incorrecto**



Los dos usos de i arriba no tienen nada que ver entre ellos. La creación de variables únicas para cada propósito hace que su código sea más legible.

1. Tomado de: https://github.com/kosme10/Buenas-Practicas-de-codificacion [↑](#footnote-ref-1)