Carlos Aurelio Alcantara Perez 406

ESTÁNDAR DE PROGRAMACIÓN EN JAVA

Organización de ficheros

Las clases de JAVA se devén de agrupar en paquetes cada paquete se debe de organizar de manera jerárquica. Un fichero consta de secciones que deben de estar separadas por lineas en blanco y comentarios opcionales que identifiquen cada sección. Deben evitarse paquetes de gran tamaño sin sobre pasar las 100 lineas dado que esto provoca que la clase no encapsule un comportamiento definido.

Fichero fuente JAVA (.java)

Los ficheros de JAVA solo deben de contener una única clase o interfaz pública. Los nombres deben de coincidir. Cuando existan varias clases privadas asociadas a una clase publica se deben de agrupar en el mismo fichero fuente. Las secciones de los ficheros son:

- COMENTARIO DE INICIO.
- SENTENCIA DEL PAQUETE.
- SENTENCIAS DE IMPORTACIÓN.
- DECLARACIONES DE CLASES E INTERFACES.

COMENTARIOS DE INICIO

Son los comentarios que van al inicio del fichero deben de contener la información de la versión del mismo la fecha y el copyright.

SENTENCIAS DE PAQUETE

La primera linea no comentada de un fichero java debe de ser la sentencia de paquetes que indica el paquete al que pertenecen las clases incluidas en el fichero fuente.

SENTENCIAS DE IMPORTACIÓN

Después de la declaración de paquetes se incluirán las sentencias de importación de los paquetes necesarios

DECLARACIONES DE CLASES E INTERFACES

Elementos de declaración de una clase / interfaz	Descripción
Comentario de documentación de la clase/interfaz /** ... */	Permite describir la clase/interfaz desarrollada. Necesario para generar la documentación de la api mediante javadoc.
Sentencia class / interface	

Comentario de implementación de la clase/interfaz, si es necesario /* ... */	Este comentario incluye cualquier información que no pueda incluirse en el comentario de documentación de la clase/interfaz.
Variables de clase (estáticas)	En primer lugar las variables de clase públicas (public), después las protegidas (protected), posteriormente las de nivel de paquete (sin modificador), y por último las privadas (private).
Variables de instancia	Primero las públicas (public), después las protegidas (protected), luego las de nivel de paquete (sin modificador), y finalmente las privadas (private).
Constructores	
Métodos	Deben agruparse por funcionalidad en lugar de agruparse por ámbito o accesibilidad. Por ejemplo, un método privado puede estar situado entre dos métodos públicos. El objetivo es desarrollar código fácil de leer y comprender.

SANGRÍA

Como norma general se establecen 4 caracteres como unidad de sangría los entornos de IDE tales como eclipse o NetBeans incluyen facilidades parar formatear códigos en JAVA.

LONGITUD DE LÍNEA

La longitud de línea no debe de pasar los 80 caracteres

DIVISIÓN DE LÍNEAS

Cuando una expresión ocupe más de una línea, esta se podrá romper o dividir en función de los siguientes criterios,

- Tras una coma.
- Antes de un operador.
- Se recomienda las rupturas de nivel superior a las de nivel inferior.
- Alinear la nueva línea con el inicio de la expresión al mismo nivel que la línea anterior.
- Si las reglas anteriores generan código poco comprensible, entonces estableceremos tabulaciones de 8 espacios.

COMENTARIOS

Distinguimos dos tipos de comentarios: los comentarios de implementación y los de documentación

De implementación:

Son los comentarios que nos ayudan a describir la funcionalidad del código

Comentarios de documentación:

Estos se utilizan para describir una funcionalidad del código en especifico la cual les servirá a los próximos desarrolladores a tener una idea clara de los que se hace.

Declaraciones

Declaración por línea

Este tipo de declaración se ocupa para el uso adecuado de los comentarios.

Inicialización

Toda variable local debe de estar iniciada en el momento de su declaración.

Localización

Las declaraciones deben de situare al principio de cada bloque.

Declaracion de clases e interfaces

Durante la codificacion se debe de seguir el fotmateo

- No incluir ningún espacio entre el nombre del método y el paréntesis inicial del listado de parámetros.
- El carácter inicio de bloque ("{") debe aparecer al final de la línea que contiene la sentencia de declaración.
- El carácter fin de bloque ("}") se sitúa en una nueva línea tabulada al mismo nivel que su correspondiente sentencia de inicio de bloque, excepto cuando la sentencia sea nula, en tal caso se situará detrás de "{".
- Los métodos se separarán entre sí mediante una línea en blanco.

SENTENCIAS

Cada línea debe contener como máximo una sentencia. Ejemplo,

int contador++;

int variable--;

Las sentencias pertenecientes a un bloque de código estarán tabuladas un nivel más a la derecha con respecto a la sentencia que las contiene. El carácter inicio de bloque "{" debe situarse al final de la línea que inicia el bloque. El carácter final de bloque "}" debe situarse en una nueva línea tras la última línea del bloque y alineada con respecto al primer carácter de dicho bloque.

Todas la sentencias de un bloque deben encerrarse entre llaves "{ ... }", aunque el bloque conste de una única sentencia. Esta práctica permite añadir código sin cometer errores accidentalmente al olvidar añadir las llaves. Ejemplo,

```
if (condicion) {
 variable++;
}
```

La sentencia "try/catch" siempre debe tener el formato siguiente,

```
try {
   sentencias;
} catch (ClaseException e) {
   sentencias;
}
```

En el bloque "catch" siempre se imprimirá una traza de error indicando el tipo de excepción generada y posteriormente se elevará dicha excepción al código invocante, salvo que la lógica de ejecución de la aplicación no lo requiera. Siempre se utilizará el bloque "finally" para liberar recursos y para imprimir trazas de monitorización de fin de ejecución.

```
try {
   sentencias;
} catch (ClaseException e) {
   sentencias;
} finally {
   sentencias;
}
```

Nomenclatura de identificadores

Las convenciones de nombres de identificadores permiten que los programas sean más fáciles de leer y por tanto más comprensibles. También proporcionan información sobre la función que desempeña el identificador dentro del código, es decir, si es una constante, una variable, una clase o un paquete, entre otros.

Paquetes

Se escribirán siempre en letras minúsculas para evitar que entren en conflicto con los nombres de clases o interfaces. El prefijo del paquete siempre corresponderá a un nombre de dominio de primer nivel, tal como: es, eu, org, com, net, etc.

El resto de componentes del paquete se nombrarán de acuerdo a las normas internas de organización de la empresa: departamento, proyecto, máquina, sección, organismo, área, etc.

Generalmente se suele utilizar el nombre de dominio de Internet en orden inverso. Cuando dicho nombre contenga un carácter "-", este se sustituirá por el carácter "_".

Ejemplos:

es.provincia.organismo1.festivaldecine

es.provincia.organismo2.vivienda

es.provincia.organismo3.juventud

es.provincia.organismo3.formacion

es.provincia.organismo3.gestionturistica

java.util.ArrayList

java.util.Date

java.util.Properties

javax.servlet.http.HttpServletRequest

javax.servlet.http.HttpServletResponse

Clases e interfaces

Los nombres de clases deben ser sustantivos y deben tener la primera letra en mayúsculas. Si el nombre es compuesto, cada palabra componente deberá comenzar con maýusculas.

Los nombres serán simples y descriptivos. Debe evitarse el uso de acrónimos o abreviaturas, salvo en aquellos casos en los que dicha abreviatura sea más utilizada que la palabra que representa (URL, HTTP, etc.).

Las interfaces se nombrarán siguiendo los mismos criterios que los indicados para las clases. Como norma general toda interfaz se nombrará con el prefijo "I" para diferenciarla de la clase que la implementa (que tendrá el mismo nombre sin el prefijo "I").

class Ciudadano

class OrganigramaDAO

class AgendaService

class IAgendaService

Métodos

Los métodos deben ser verbos escritos en minúsculas. Cuando el método esté compuesto por varias palabras cada una de ellas tendrá la primera letra en mayúsculas.

public void insertaUnidad(Unidad unidad);

public void eliminaAgenda(Agenda agenda);

public void actualizaTramite(Tramite tramite)

Variables

Las variables se escribirán siempre en minúsculas. Las variables compuestas tendrán la primera letra de cada palabra componente en mayúsculas.

Las variables nunca podrán comenzar con el carácter "_" o "$". Los nombres de variables deben ser cortos y sus significados tienen que expresar con suficiente claridad la función que desempeñan en el código. Debe evitarse el uso de nombres de variables con un sólo carácter, excepto para variables temporales.

Unidad unidad;

Agenda agenda;

Tramite tramite;

Constantes

Todos los nombres de constantes tendrán que escribirse en mayúsculas. Cuando los nombres de constantes sean compuestos las palabras se separarán entre sí mediante el carácter de subrayado "_".

int LONGITUD_MAXIMA;

int LONGITUD_MINIMA;