Ingeniería de Software II

Estándares de nomenclatura.

09/10/2015

Equipo 4 “La Roca 2”

Tabla de contenido

[Estándares de documentación interna. 3](#_Toc432191386)

[Bloques de documentación interna de la plantilla. 3](#_Toc432191387)

[Módulo de Documentación Bloquear 4](#_Toc432191388)

[Estándar de codificación. 5](#_Toc432191389)

[Nomenclatura 5](#_Toc432191390)

[Paquetes 5](#_Toc432191391)

[Nombre de interfaces 5](#_Toc432191392)

[Nombre de las clases 5](#_Toc432191393)

[Métodos 6](#_Toc432191394)

[Variables 6](#_Toc432191395)

[Constantes 6](#_Toc432191396)

[Comentarios 6](#_Toc432191397)

[Sentencias 7](#_Toc432191398)

[Sentencias return 7](#_Toc432191399)

[Líneas en blanco 7](#_Toc432191400)

[Paréntesis 8](#_Toc432191401)

[Longitud de la línea 8](#_Toc432191402)

[Convenios de nombrado 8](#_Toc432191403)

[Estándar de codificación para los controles FXML. 9](#_Toc432191404)

[Regla 9](#_Toc432191405)

[Controles más utilizados. 10](#_Toc432191406)

[Directrices de codificación 11](#_Toc432191407)

[Extensiones de los ficheros 11](#_Toc432191408)

[Organización de los ficheros 11](#_Toc432191409)

[Tabulación 12](#_Toc432191410)

[Comentarios 12](#_Toc432191411)

[Declaraciones 13](#_Toc432191412)

[Sentencias 14](#_Toc432191413)

[Espacios en blanco 14](#_Toc432191414)

[Convenciones de nombres 15](#_Toc432191415)

[Estándar de codificación de Base de datos. 16](#_Toc432191416)

[Nomenclatura para la base de datos. 16](#_Toc432191417)

[Nomenclatura para las tablas. 16](#_Toc432191418)

[Nomenclatura para las vistas. 16](#_Toc432191419)

[Nomenclatura para las columnas. 17](#_Toc432191420)

[Nomenclatura para los atributos. 17](#_Toc432191421)

[Nomenclatura para los triggers. 17](#_Toc432191422)

# Estándares de documentación interna.

El realizar correctamente la documentación interna para mejorar la legibilidad de un módulo de software. El SISPEG ha acordado que un archivo que contiene uno o más módulos de software o un archivo script de Shell debe tener un bloque de comentario en su comienzo que contenga la siguiente información básica:

* El nombre del autor que creó el archivo
* La fecha de creación del archivo
* Grupo de desarrollo del autor (por ejemplo HSEB, HSMB)
* Descripción (descripción general de los efectos de los módulos)

El SISEPG tiene un conjunto de elementos que se debe seguir independientemente del lenguaje de programación que se utiliza. Para poder tener una fácil compresión de la lectura y mantener los programas la sangría debe utilizarse para:

* En el cuerpo de una sentencia de control, como un lazo o una instrucción select.
* En el cuerpo de una sentencia condicional.
* En un nuevo bloque de alcance.

En general, sangría por tres o cuatro espacios se considera que es adecuado. Una vez que el programador elige el número de espacios de sangría es importante que se aplique de forma coherente esta cantidad.

## Bloques de documentación interna de la plantilla.

Los ejemplos utilizan un comentario estilo C, que también es utilizado C ++ y Java.

* Autor:
* Fecha de Creación del archivo:
* Grupo de Desarrollo:
* Descripción:

## Módulo de Documentación Bloquear

Este bloque de documentación precede a cada módulo en el archivo de origen.

Los elementos opcionales se denotan mediante el uso de paréntesis.

* Nombre del Módulo:
* Autor:
* Fecha de Creación del módulo:
* Descripción:
* Constante y Macro Suplentes:
* Nombre Descripción
* Valor de retorno:
* Fecha desarrollador Acción
* Archivos necesarios / Bases de datos
* Supuestos específicos del sistema operativo en su caso:
* Variables locales:
* Descripción Nombre del encabezado del archivo
* Modificación Historia:
* Tipo Descripción
* Códigos de error / Excepciones:
* Modificación Historia:
* Fecha desarrollador Acción
* Archivos necesarios / Bases de datos:
* Las rutinas no sistema llamado:
* Los argumentos de llamadas:
* Nombre Entrada / Salida Tipo Descripción
* Tipo Nombre Descripción
* Variables globales utilizados:
* Tipo Nombre Origen Descripción

# Estándar de codificación.

## Nomenclatura

El idioma para nombrar a las clases, variables y constantes será una mezcla de inglés y el idioma propio del programador. Aquella codificación que por estandarización este en inglés, se mantendrá así por convenio.

Ejemplos: Insert, update, delete, drop, créate, list, set y get.

## Paquetes

Todos los paquetes se escribirán en minúsculas y sin utilizar caracteres especiales.

Ejemplos: modelo, vista y controlador o en inglés model, view and controller

## Nombre de interfaces

Los nombres de interfaces utilizaran el prefijo I y estarán compuestos por palabras con la primer letra en mayúscula (CamelCase). Se debe evitar el uso de abreviaciones que dificulten la comprensión del código. El sufijo I significa interfaz.

Ejemplos: ICoche, ILavadora, IVentas, IEmpleado, IAlumnos

## Nombre de las clases

Los nombres de las clases deben ser una combinación de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúscula (CamelCase). Se deben mantener los nombres de las clases simples y descriptivas, se deben usar palabras completas y evitar las abreviaturas. Si el nombre de la clase de conforma de dos a mas palabras lo que se hará es escribir la primer letra en mayúscula, seguida de la segunda palabra con la primer letra en mayúscula.

Ejemplos: Profesor, NombreAlumno, ApellidoPaterno, ApellidoMaterno, Edad

## Métodos

Los métodos deberán ser verbos en infinitivo en mayúsculas y minúsculas con la primera letra en minúsculas (lowerCamelCase). No se permiten caracteres especiales.

Ejemplos: get, set

## Variables

Los nombres de las variables reciben la misma estructura que para los métodos, para ellos utilizaremos (LowerCamelCase), empezando por una minúscula seguida de la segunda palabra en mayúscula. En este caso también se deben evitar las abreviaciones. También se evitará en la medida de lo posible la utilización de caracteres especiales.

Ejemplos: Correctos: diaCalculo, fechaIncoporacion

## Constantes

Los nombres de constantes de clases deberían escribirse todo en mayúsculas con las palabras separadas por subrayados ("\_"). Todas serán declaradas como public static final.

Ejemplo: public static final String PROPERTY\_URL\_SERVICIO = "urlServicio";

## Comentarios

Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de la clase. El uso abusivo de los comentarios es desaconsejable, principalmente por el trabajo extra necesario para su correcto mantenimiento. Es preferible rediseñar el código para una mejor compresión del mismo. Se tienen que evitar el uso de caracteres especiales dentro de los comentarios. Si se desea comentar una línea de código se utilizara doble diagonal seguida del comentario, pero en cambio si se desea comentar un párrafo completo el párrafo entre diagonal y asterisco de ahí el párrafo y se cierra con un asterisco y una diagonal.

Ejemplos: // mi comentario

/\*  
Aquí dentro se escribe todo el párrafo   
que se desee comentar  
\*/

## Sentencias

Las normas básicas son:

* Una sentencia por línea de código.
* Todo bloque de sentencias va entre llaves, aunque sea una sola sentencia después de un if.

Ejemplo de la declaración de los bucles for : for (int i = 0; i < condicion ; i++)

Son obligatorias las tres condiciones del bucle for: inicialización, condición de finalización y actualización del valor de la variable de avance. La variable de avance del bucle nunca podrá ser modificada dentro del propio bucle.

## Sentencias return

Una sentencia return con un valor no debe usar paréntesis a menos que hagan el valor de retorno más obvio de alguna manera.

Ejemplos: return; return miDiscoDuro.size();

## Líneas en blanco

Las líneas en blanco mejoran la facilidad de lectura separando secciones de código que están lógicamente relacionadas.

## Paréntesis

En general es una buena idea usar paréntesis en expresiones que implican distintos operadores para evitar problemas con el orden de precedencia de los operadores. Incluso sí parece claro el orden de precedencia de los operadores, podría no ser así para otros. No se debe asumir que otros programadores conozcan el orden de precedencia.

Ejemplos: if ((a == b) && (c == d))

## Longitud de la línea

Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

Ejemplos para uso en la documentación deben tener una longitud inferior, generalmente no más de 70 caracteres.

## Convenios de nombrado

Los convenios de nombrado hacen los programas más entendibles facilitando su lectura. También pueden dar información sobre la función de un identificador, por ejemplo, cuando es una constante, un paquete, o una clase, que puede ser útil para entender el código.

# **Estándar de codificación para los controles FXML**.

## Regla

Prefijo más nombre de la variable (CamelCase).

Button btnNombre.   
CheckBox chkNombre.  
ChoiceBox cheNombre.  
ColorPicker cpNombre.  
ComboBox cbNombre.  
DatePicker dpNombre.  
HTMLEditor heNombre.  
Hyperlink hlNombre.  
ImageView ivNombre.  
Label lblNombre.  
ListView lvNombre.  
MediaView mvNombre.  
MenuBar mbNombre.  
MenuButton mbtnNombre.  
Pagination paginadorNombre.  
PasswordField pwNombre.  
Progressbar pbNombre.  
ProgressIndicator pINombre.  
RadioButton rbtnNombre.  
ScrollBar sbhNombre.  
ScrollBar sbvNombre.  
Separator shNombre.  
Separator svNombre.  
Slider slhNombre.  
Slider slvNombre.  
SplitMenuButton smbtnNombre.  
TableColumn tcNombre.  
TableView tvNombre.  
TextArea txtaNombre.  
TextField txtNombre.  
ToggleButton tbtnNombre.  
TreeTableColumn ttcNombre.  
TreeTableView ttvNombre.  
TreeView twNombre.  
WebView wvNombre.

## Controles más utilizados.

|  |  |
| --- | --- |
| Control | Prefijos |
| Button | btn |
| TextField | txt |
| TextArea | txta |
| TableColumn | tc |
| TableView | tv |
| TreeView | trv |
| MenuBar | mb |
| ImageView | iv |
| ComboBox | cb |
| Pagination | paginador |
| Label | lbl |
| CheckBox | chk |

# Directrices de codificación

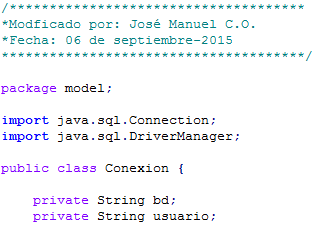
Las directrices de codificación son aquellas reglas aplicadas al código para que este sea “elegante”, es decir que sea fácilmente legible y por tanto facilite su corrección o modificación de ser necesaria en algún momento.

## Extensiones de los ficheros

Los ficheros utilizados para la codificación de tendrán las siguientes extensiones.

|  |  |
| --- | --- |
| .java | Fichero de código fuente. |
| .class | Fichero compilado de java. |
| .fxml | Archivo fxml para la el diseño de interfaces. |
| .css | Hoja de estilo para dar presentación a la interfaz. |
| .jar | Librería precompilada. |

## Organización de los ficheros



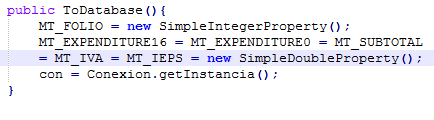
Los ficheros de código fuente deben tener el siguiente orden:

1. Comentarios del comienzo.
2. Sentencias package e import.
3. Declaración de clases de interfaces.

Una vez declarada la clase también es necesario organizar sus componentes internos siguiendo la siguiente jerarquía:

1. Comentarios de documentación de la clase.
2. Sentencia class o interfaces.
3. Comentario de implementación de la clase (opcional).
4. Variables de clase (“static”, primero públicas, después protegidas y al final privadas).
5. Variables de instancia (primero públicas, después protegidas y al final privadas).
6. Constructores.
7. Métodos.

## Tabulación

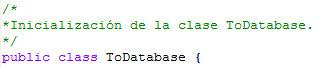


Se deben evitar las líneas de más de 80 caracteres, ya hasta este límite no son bien manejadas por algunas herramientas, cuando una expresión no entre en una línea romperla de acuerdo a los siguientes principios.

1. Romper después de una coma.
2. Romper antes de un operador (+, %, /…)
3. Alinear la nueva línea al nivel de la expresión inmediatamente anterior.

## Comentarios

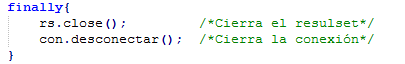
Los programas en Java tienen dos clases de comentarios; de implementación y de documentación, los primeros sirven de descripción para alguna implementación en particular y los segundos describen la especificación del código, así mismo pueden tener varios formatos.



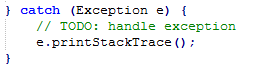
Comentario de bloque: Debe ir precedido por una línea en blanco que lo separe del resto del código.



Comentario de una línea: Son comentarios que de una única línea que se hacen en algún nivel del código fuente.

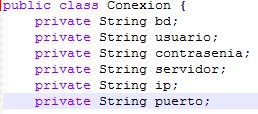


Comentarios de remolque: Son comentarios cortos en el mismo nivel que el código fuente y separados de las sentencias a la misma distancia.

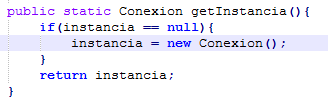


Comentario de fin de línea: Puede convertir en comentario una línea completa o parte de una línea.

## Declaraciones



Se recomienda una declaración por línea, así mismo no se recomienda declarar diferentes tipos de dato en el mismo nivel, esto además genera errores, para el caso de las variables también recomendable inicializarlas justo cuando son declaradas.



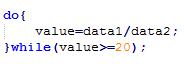
Al codificar clases e interfaces se siguen las siguientes reglas:

1. Ningún espacio en blanco entre el método y el paréntesis que lo abre.
2. La llave de apertura “{“ debe aparecer al final de la línea de sentencia que se declara.
3. La llave de salida “}” debe ajustarse al nivel de la sentencia correspondiente.
4. Los métodos se separan por líneas en blanco.

## Sentencias



Cada línea puede contender solo una sentencia, las sentencias por lo general se delimitan por el carácter “;”.



También existen sentencias compuesta que contienen listas de sentencias encerradas entre llaves { }.

## Espacios en blanco

Los espacios en blanco facilitan la lectura separando secciones del código que están lógicamente relacionadas.

Se debe usar dos líneas en blanco en las siguientes circunstancias:

1. Entre secciones de un fichero de código fuente.
2. Entre definiciones de clases de interfaces.

Se debe usar una línea en blanco siempre en las siguientes circunstancias.

1. Entre métodos.
2. Entre variables locales de un método y su primera sentencia.
3. Antes de un comentario de bloque o de línea.
4. Entre distintas secciones lógicas de un método.

## Convenciones de nombres

|  |  |  |
| --- | --- | --- |
| Tipo | Regla | Ejemplo |
| Paquete | Se debe escribir en minúscula, los subsecuentes componentes dependerán de la convención utilizada por la organización y que corresponderá a departamentos, directorios, proyectos o máquinas. | com.sun.eng  com.edu.lang |
| Clase | Deben corresponder a sustantivos y tener la primera letra en mayúsculas. | class Cliente; |
| Interfaces | La misma regla que las clases. | interface Almacen; |
| Métodos | Debe corresponder a verbos e iniciar con minúscula, cuando son compuestos, la segunda palabra deberá iniciar en mayúscula. | executeQuery; |
| Variables | Se debe evitar las variables de un solo carácter o que inicien con un carácter especial, exceptuando instancias y variables de clases o métodos iniciarán con minúscula. | int dato;  char name; |
| Constantes | Las variables declaradas como constantes deben ir siempre en mayúsculas. | ALTURA\_MAXIMA=15; |

# Estándar de codificación de Base de datos.

## Nomenclatura para la base de datos.

Para nombrar la base de datos se debe poner la primera letra en mayúscula y lo demás en minúsculas si consta de una sola palabra, **sin usar espacios en blanco.**

Por ejemplo *“Renta”.*

En caso de que el nombre de la base de datos conste de 2 o más palabras se deberá poner la primera letra de cada palabra en mayúscula, pero solo la primera, **sin usar espacios en blanco.**

Por ejemplo *“PrimeraRenta”*.

## Nomenclatura para las tablas.

Para nombrar las tablas debe ser en forma singular y debe representar lo que almacena la tabla, esta se escribirá en minúsculas.

*Por ejemplo “empleados”.*

Si consta de dos palabras la segunda palabra deberá iniciar con mayúscula.

*Por ejemplo “departamentoEmpleados”.*

## Nomenclatura para las vistas.

Para nombrar las vistas se utilizara la misma nomenclatura que las tablas pero usando el prefijo “**vw\_**”.

*Por ejemplo “vw\_empleados”.*

## Nomenclatura para las columnas.

Para nombrar las columnas debe ser de forma singular y representativa, de la misma manera que las tablas.

*Por ejemplo “nombreEmpleado”, “codigoEmpleado”.*

## Nomenclatura para los atributos.

Cada nombre de los atributos debe corresponder a una columna predeterminada para ellos, si el atributo corresponde al nombre del empleado esta debe empezar con mayúscula.

*Por ejemplo “Norma”.*

## Nomenclatura para los triggers.

Para nombrar los triggers se utilizará la misma regla de las tablas, pero con la siguiente instrucción, <TABLA>\_<OPERACION>[\_AUX]

*Por ejemplo “Customer\_Insert\_InsertInUsers”.*