Estándar de Desarrollo

Versión 1.1

Historia de revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Autor |
| 21/10/2019 | 1.0 | Primera versión | Julio  Rolando |
| 22/10/2019 | 1.1 | Correcciones estéticas menores. | Tito |
| 25/10/2019 | 1.2 | Versión revisada por QA y corregida. | Erik Manchego |
|  |  |  |  |

Contenido

[1. Convenciones Generales 3](#_Toc144304578)

[1.1. Nomenclatura 3](#_Toc144304579)

[1.2. Comentarios 3](#_Toc144304580)

[1.3. Estructura 3](#_Toc144304581)

[1.3.1. Estructura de los fuentes 4](#_Toc144304582)

[1.3.2. Estructura de bloques 4](#_Toc144304583)

[2. Convenciones Específicas 4](#_Toc144304584)

[2.1. Variables 5](#_Toc144304585)

[2.2. Constantes 5](#_Toc144304586)

[2.3. Indentación 5](#_Toc144304587)

[3. Otros 5](#_Toc144304588)

1. Convenciones Generales

Para la definición del estándar de implementación se tomará como referencia Java Code Conventions de SUN Microsystems.

Se detallarán a continuación las convenciones que se consideran más relevantes. Tener en cuenta que las convenciones aquí descritas puede diferir de lo especificado en las convenciones Java.

Por más información referirse a <http://java.sun.com/docs/codeconv/>.

* 1. Nomenclatura

Los estándares de Nomenclatura de objetos se detallan a continuación:

* Los nombres de paquetes, clases y métodos deben estar en inglés.
* Los nombres de clases y archivos deben estar en notación Pascal, esto es, la primera letra con mayúscula y las siguientes en minúscula, en caso de ser un nombre compuesto respetar esta convención para cada palabra. (Ej.: ThisIsAClass.java)
* Los nombres de interfaces deben seguir la convención anterior agregando la letra I (mayúscula) al comienzo.
* En caso que la clase represente un controlador, el nombre debe seguir la convención Pascal precedida por una letra C (mayúscula). (Ej.: CUsers)
* Los enumerados deben definirse con notación Pascal. Ej.: public enum { Type1, Type2 }
* Los nombres de las operaciones deben ser verbos y seguir la convención Camel, esto es, la primera letra en minúscula, Ej.: runFast().
* Los nombres para los paquetes deben seguir la notación Pascal y deben comenzar siempre por el nombre del proyecto, como en el siguiente ejemplo: FeedIt.PackageName
* Los nombres de las excepciones deben terminar con Exception. Ej.: FeedItException.
  1. Comentarios

Los estándares de la utilización de comentarios son los siguientes:

* Todos los comentarios que se refieran a métodos, variables, clases, etc, deben seguir el estándar JavaDoc.
* Se debe incluir un comentario de inicio del archivo antes de la definición de clases, que provea información acerca de las funcionalidades que provee.
* Se debe incluir un comentario para cada método de la clase, que explique a grandes rasgos su cometido.
* Cuando se finaliza la implementación de un componente se recomienda eliminar todo código comentado que dificulte la lectura del mismo.
* Los comentarios deben ser en español.
  1. Estructura
     1. Estructura de los fuentes

La estructura de los fuentes deberá seguir el siguiente orden:

1. Comentario de inicio.
2. Especificación del paquete.
3. Importación de librerías.
4. Declaración de la clase o interfaz.
5. En caso de tratarse de una clase, definición de atributos.
6. Constructores
7. Métodos agrupados por funcionalidad, comenzando por los get() y set().
   * 1. Estructura de bloques

Las llaves de inicio de bloque deben colocarse al final de la sentencia que lo inicia. La llave de fin no debe estar en la misma línea que la última sentencia del bloque y debe estar tabulada a la misma altura que la sentencia de inicio.

Ejemplos de estructuras:

**If**

if (cond) {

instr;

}else if (cond) {

instr;

}else {

instr;

}

**Try-Catch**

try {

instr;

}catch (ClassException e) {

instr;

}

1. Convenciones Específicas

En todos los casos se debe cumplir que el nombre asignado sea descriptivo de la funcionalidad a la cuál hacer referencia. Evitar utilizar abreviaciones para los nombres, éstas tienden a dificultar la mantenibilidad.

* 1. Variables

Los nombres de las variables y atributos deben seguir la convención Camel.

Se deben definir siempre al inicio del bloque en donde se utilizan.

También se deben inicializar al momento de la definición.

Se permite el uso de variables de una sola letra en minúscula para índices, ejemplo en un for. Evitar el uso de la letra ele minúscula (“l”).

* 1. Constantes

Las constantes se definirán con todas las letras en mayúsculas. En caso de ser nombre compuesto separar por un guión bajo (“\_”). Los comentarios deben ir a la derecha de la definición de la misma como se ve en el ejemplo:

public static final int MY\_BDATE = 10; // Esto es un comentario

* 1. Indentación

A continuación se detallan los estándares de Identación:

* La indentación se realizará con uso de tabuladores.
* Evitar utilizar espacios para indentar.
* Hacer uso de las herramientas provistas por Eclipse para el formato automático de código.
* Las sentencias muy largas, de manera que no entren en pantalla, deben separarse en varias líneas. Referirse, por ejemplos, al punto 4.2 de la documentación de Java.

1. Otros

Para mayor seguridad en el seguimiento de estos estándares se utilizarán herramientas de revisión automática de código provistas en pluggins (lo estábamos escribiendo así, es correcto?) para Eclipse.

Comentarios:

* Se podría agregar algún Anexo que facilite la aplicación de los estándares a los implementadores (convención Camel, JavaDoc, y algún otro estándar que se mencione).
* No se olviden de enviarle a Silvana los términos a incluir (si corresponde) en el Glosario.
* Actualicen el Índice pues me está dando error las referencias, supongo será por el control de Cambios.
* Les copio algo que se incluyó en otro informe (Vean si puede aplicar o no), por lo menos me parece relevante indicar el autor (no responsable) de cada clase:

/\*\*

\* Documentación describiendo la clase

\* Puede ser en formato HTML.

\*

\* @author <nombre del autor original>

\* @see <vinculo>

Beneficios:

* 1. • Establecer de forma clara cual es la funcionalidad de la Clase o Interfase.
  2. • Describir brevemente cuales son los principales servicios que presta.
  3. • Especificar a que módulo pertenece dentro de la arquitectura.
  4. • De forma opcional describir principales mecanismos, estructuras o algoritmos utilizados en la clase, cuando el programador lo crea conveniente.

La persona que crea una clase se debe identificar utilizando la declaración @author <nombre de autor>. De la misma manera que enriquezca la documentación con el uso de referencias utilizando la declaración @see, la cual en la generación es mapeada a un vínculo dinámico html.

Esta prohibido el uso de la declaración @version, debido a que el versionado lo va a manejar el CVS y su declaración puede ser fuente de confusiones.