ESTÁNDAR PARA CODIFICACIÓN

Elaborado por:
Missael Hernández Rosado

Lic. Ingeniería de software | Principios de construcción de software
Facilitador: Juan Carlos Pérez Arriaga

Estándar de codificación

Propósito	Guiar el la codificación uniforme de programas escritos en Java
Encabezado del (la) programa / clase	Todos los archivos deben contener un encabezado con los siguientes datos: • Nombre del programador • Fecha de creación del archive • Ultima fecha de modificación del archive • Descripción breve de lo que el programa o clase hace

Formato del encabezado	`/*`
	` * Autor: Juan Pedro Pérez Vargas`
	` * Fecha de creación: 02/12/2016`
	` * Fecha de modificación: 09/12/2016`
	` * Descripción: Describe las características y`
	` *  comportamiento de una agenda`
	` */`

Instrucciones de reúso	Cada método debe contener, además de los comentarios de la implementación, comentarios referentes al uso de la función justo antes de la declaración del método a fin de servir como documentación, estos comentarios se fusionarán con los comentarios de Javadoc: Estos comentarios deben proveer la siguiente información: • Propósito de la definición del método • Limitaciones del método (en caso de existir) • Justificación y/o explicación breve del tipo de retorno del método y su uso.

Ejemplo de instrucciones de reúso	`/**`
	` * Ajusta la hora de inicio`
	` * @param horaInicio La hora en formato de 24 horas inicio a guardar`
	` * @throws java.lang.Exception`
	` */`

Identificadores	Todos los identificadores deben seguir a un nombre lógico y descriptivo de su función,
	• Los nombres de paquetes son todos en minúsculas, sin guiones bajos ni ningún carácter que actué como separador de palabras.
	• Los nombres de las clases siempre se escriben con la nomenclatura UpperCamelCase.
	  ○ Los nombres de las clases son normalmente sustantivos o frases compuestas por un adjetivo y un sustantivo
	• Los nombres de los método siguen la nomenclatura lowerCamelCase
	  ○ Los nombres de métodos son usualmente verbos u oraciones compuestas por un verbo en infinitivo y un sustantivo.
	• Los nombres de contantes deben ser definidas en mayúsculas completamente separando las palabras por un guio bajo.
	• Los atributos, parámetros y variables locales deben ser definidos en la nomenclatura lowerCamelCase
	Para un definición complete de camelCase diríjase a:
	https://google.github.io/styleguide/javaguide.html#s5.3-camel-case

Ejemplo de identificadores	`class Agenda {`
	`  private int Id;`
	`  private String nombrePropietario;`
	
	`  public getId () {}`
	`  public setId (int id) {`
	`  int variableLocal = 1;`
	`  }`
	`}`

Comentarios	• Documentar el código en la medida que cualquier lector pueda entender el código • Los comentarios deben explicar tanto el comportamiento como el propósito del código. • La variables deben ser comentadas únicamente en el caso de que el nombre no sea lo suficientemente descriptivo. Los comentarios de variables deben definirse como comentarios en línea. • Los comentarios deben aportar al lector información adicional al código, no describir su semántica.

Espacios en blanco	**Espacios en blanco verticales:**
	Una sola línea en blanco aparece:
	
	• Entre los miembros consecutivos (o iniciadores) de una clase: campos, constructores, métodos, clases anidadas, inicializadores estáticos, inicializadores de instancia.
	• Excepción: Una línea en blanco entre dos campos consecutivos (no teniendo otro código entre ellos) es opcional. Tales líneas en blanco se utilizan según sea necesario para crear agrupaciones lógicas de campos.
	• Dentro de cuerpos de los métodos, según sea necesario para crear agrupaciones lógicas de declaraciones.
	• Opcionalmente antes del primer miembro o después del último miembro de la clase.
	Espacios en blanco :
	Un único espacio ASCII aparecerá sólo en los siguientes lugares.
	
	• La separación de cualquier palabra reservada, como *if*, de un paréntesis de apertura (*(*) que le sigue en esa línea.
	• La separación de cualquier palabra reservada, como *else* o *catch*, de una llave de cierre (}) que lo precede en esa línea.
	• Antes de cualquier llave de apertura ({), con dos excepciones:
	  ○ SomeAnnotation ({A, b}) (no se utiliza ningún espacio)
	  ○ String [] [] x = {{ "foo"}}; (No se requiere ningún espacio entre {{)
	• A ambos lados de cualquier operador binario o ternario.
	• A ambos lados de la doble barra (//) que comienza un comentario al final de la línea. Aquí, múltiples espacios están permitidos.
	• Entre el tipo y la variable de una declaración: List<String> lista
	• Opcional justo dentro de los dos corchetes de un inicializador de matriz
	  ○ new int [] {5, 6} y new int [] { 5, 6 } son ambos válidos.

Indentación	Cada vez que un nuevo bloque se abre, el nivel de la sangría aumenta cuatro espacios. Cuando el bloque termine, la sangría se regresa al nivel de sangría anterior. El nivel de sangría se aplica tanto al código como a los comentarios en todo el bloque.

Ejemplo de indentación	`class Agenda {`
	`    private int Id;`
	`    private String nombrePropietario;`
	
	`    public getId () {}`
	`    public setId (int id) {`
	`        int variableLocal = 1;`
	`    }`
	`}`