FEB 2023

Universidad Veracruzana

PROPUESTA ESTÁNDAR DE CODIFICACIÓN

JAVA

Principios de Construcción de Software

Fabiola Narayani Guzmán Figueroa
Mario Morales Portilla
Aneth Michelle Tamariz Moreno

Tabla de Contenido

Introducción

El desarrollo de software trae consigo varias implicaciones que deben considerarse, pues para que el producto sea comprensible, escalable, se le pueda dar mantenimiento y tenga una implementación correcta es necesario seguir un estándar de codificación para unificar el código del equipo de desarrollo y que a su vez facilite la lectura de este. Además, que gracias a estas técnicas se podrá obtener un rendimiento más eficiente en las aplicaciones, por ello a lo largo de este documento se definirá un estándar de codificación para Java que nos ayude a cumplir con los propósitos planteados.

Propósito

Ayudar a crear un código que sea fácil de entender, mantener y mejorar en cualquier momento, incluso que pueda ser escalable.

1. Reglas de nombrado.

Con el propósito de hacer que los programas sean más entendibles es necesario utilizar reglas de nombrado que nos ayuden a leer fácilmente, aunque toma tiempo crear buenos nombres pues deben revelar la intención de lo que estamos nombrando ya que utilizan nombres relacionados con el dominio de la solución y de significado claro. Las convenciones por seguir para el nombrado en este estándar incluyen el uso completo del idioma inglés a la hora de codificar por lo que los nombres mantendrán esta regla para describir las variables, métodos, constantes y clases.

- Las frases o acrónimos oficiales deben respetar la nomenclatura correspondiente de acuerdo con su tipo de estructura.
- No repetir nombres entre variables, métodos, clases, etc.
- Utilización nombres fáciles de pronunciar.
- Uso de prefijos en la capa gráfica y asociados, así como en métodos de acceso o mutators.
- Evitar abreviaciones, hay que ser lo más específicos posibles, pero sin caer en redundancias.
- Los verbos o frases que hagan alusiones a verbos solo deberán utilizarse en nombres de métodos, evitarlos en nombres de clases.
- Evitar el reemplazo de letras por símbolos como el uso del signo de dólar para reemplazar la letra S.

1.1 Nombrado de variables

El nombre de las variables deberá estar escrito en LowerCamelCase lo que significa que la primera y última letra de la palabra irán en minúsculas, la siguiente palabra iniciara con una letra mayúscula y las demás serán minúsculas, cabe mencionar que no hay espacios entre palabras.

Ejemplo:

```
int profileIndex = 0;
String customersName = ””;
```

1.2 Nombrado de constantes

El nombre de las constantes deberá estar escrito en UPPER_SNAKE_CASE donde todas las letras del nombre iran en mayúsculas y los espacios se representarán con un guion bajo (_).

Ejemplo:

```
final double PI_VALUE = 3.1416;
final float TAX_PERCENTAGE = 1.92;
```

1.3 Nombrado de métodos

El nombre de los métodos deberá estar escrito en owerCamelCase al igual que las variables.

Ejemplo:

```
public double calculateTotalPayment(int cost, int countryTax){
    return cost + (cost*countryTax);
}
```

1.4 Nombrado de clases

El nombre de las clases deberá estar escrito en UpperCamelCase donde la primera letra de cada palabra deberá ir con mayúsculas y las demás letras en minúsculas, al igual que el LowerCamelCase no se usarán espacios en este tipo de escritura.

Ejemplo:

```
public class MathStudent extends UvStudent{
    private string research=””;

    public void mathResearch(){
        System.out.println(name + “ researches ” + research);
   }
}
```

1.5 Nombrado de contadores en ciclos.

2. Estilo de código

A medida que el código va creciendo es complicado leer todo lo que se ha hecho, por lo que utilizar un estilo de código proveerá elegancia, estética y mayor comprensión a este. En esta

sección se abordan aspectos fundamentales como la indentación, espaciados, líneas en blanco, longitud máxima de las líneas en cuestión de caracteres.

2.1 Indentación.

La acción de utilizar sangrado en las líneas de código se conoce como indentar que sirve para indicar visualmente los niveles en los que nos entramos ya sea al interior de una función, ciclo o condicional.

2.2 Líneas en blanco y saltos de línea

2.3 Espaciados

3. Comentarios

4. Estructuras de control

En lenguajes como java solo se usarán prefijos métodos de acceso.

Usar 4 espacios para un nivel de indentacion, no se usará la tecla tab.

No dejar llaves sin contenido en sentencias multi-block (como if/else o try/catch/finally)

Tampoco se permitiran sentencias de codigo vacias en un While o If

Una sentencia por línea. Cada sentencia está seguida por un salto de línea

Una variable se usa para una función en específico (por ejemplo, una variable para el switch).

Las variables deben tener un inicializador.

Usar @overrride siempre que se sobreescriba un método.

Utilizar Javadoc cuando se implementen los métodos definidos en el DAO.

Los elementos en los archivos de Java deben organizarse en una secuencia estándar: class comments ■ package declaration ■ import statements ■ class declaration ■ static variables ■ instance variables ■ constructors ■ methods

No atrapar excepciones genéricas. Para ello se debe atrapar cada excepción por separado o arrojarla.

No atrapar ni lanzar excepciones de Null Pointer Exception. En su lugar, se debe asegurar de remover la causa de dicha excepción.

Si se necesita usar un bloque de comentario este deberá usar la siguiente forma:

```
/*
 * This is a block comment
```

```
 * it´s written in several lines
*/
```

Si solo se necesita usar una linea de comentario se añadira de la siguiente manera:

```
/* This a line comment */
```

Una variacion seria:

```
// This is a variation to write a line comment
```

Tambien se pueden usar lineas de comentarios al final de una sentencia de codigo:

```
…. // This is a comment at the end of the line
```

Código que se encuentre comentado se debera eliminar para la versión final del código.

Líneas de código de 80 caracteres máximo

Si la línea de código sobrepasa de los 80 caracteres se tendrá que hacer uso del *salto de línea:*

- Se hace un salto de línea después de una coma.
- Se hace un salto de línea después de un operador.
- Alinear la nueva linea a dos niveles de identacion despues del inicio de la linea separada.
- Intenta no romper una linea que tenga una expresion entre parentesis.

Si hay variables, métodos, asignaciones o variables que no estén siendo usadas se deberán eliminar para la ultima version del codigo.

Referencias