ESTANDAR DE CODIFICACIÓN

JAVA FX

30 DE NOVIEMBRE DE 2020

BRANDON TRUJILLO, MARIO DORANTES
Principios de construcción de software

Contenido

1. Introducción

El siguiente documento muestra las reglas de codificación que se seguirán para realizar el desarrollo del sistema de control de calidad de la Universidad Veracruzana (SGC). Dicho documento es generado como parte del proyecto de la EE Principios de Construcción de Software, para que, de esta forma, el código resultante del software mencionado anteriormente se apegue estrictamente a este estándar.

Lo anterior mencionado será desarrollado en el lenguaje de programación JAVA FX, utilizando el IDE Neatbeans para escribir el código y la herramienta Java FX secene builder para la construcción de las interfaces de usuario.

2. Nombres

En esta sección se indicará como debe de ser el nombrado de los elementos que puedan aparecer en el código

2.1. Métodos

Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas (CamelCase). No se permiten caracteres especiales. El nombre ha de ser lo suficientemente descriptivo, no importando la longitud del mismo. Ejemplo:

- Correcto: validarCamposCorrectos
- Incorrecto: validarCamp
- Incorrecto: validarcampos
- Incorrecto: validaciónDeCampos

2.2. Variables

Los nombres de las variables tanto de instancia como estáticas reciben el mismo tratamiento que para los métodos, con la salvedad de que aquí sí importa más la relación entre la regla mnemónica y la longitud del nombre. Ejemplo:

- Correctos: diaCalculo, fechaIncoporacion
- Incorrectos: dC, DCal, fI, FI¿

Se evitará en la medida de lo posible la utilización de caracteres especiales, así como nombre sin ningún tipo de significado funcional.

Las excepciones son las variables utilizadas en bucles for, para esos casos se permite utilizar i, j, k, l y siempre en ese orden de anidamiento. El primer bucle siempre será el que tenga la variable i como iterador. (Esta variable se definirá para el bucle en cuestión).

2.3. 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";

2.4. Clases

Los nombres de clases deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas (CamelCase). Debemos intentar mantener los nombres de clases simples y descriptivos.

Debemos usar palabras completas y evitar acrónimos y abreviaturas (se permiten DAO, DTO, URL, HTML, etc.). Si la clase cumpliese algún patrón determinado o tuviese una funcionalidad específica es recomendable definirlo en el nombre.

2.5. Paquetes

Por defecto todos los paquetes deben escribirse con su primera letra minúscula y usando mayúsculas al iniciar cada palabra nueva (camelCase). Ejemplo:

- ventanasDeUsuario

2.6. clases de FXML

Todas las interfaces de usuario deben de ser realizadas usando Java FX, por tanto, los nombres de las clases que contengan interfaces deben contener el prefijo FXML al iniciar el nombre, en mayúsculas seguido del nombre de la interfaz, usando camelCase. Ejemplo:

- Correcto: FXMLLogin
- Correcto: FXMLRegistarDocente
- Incorrecto: RegistrarDocente
- Incorrecto: fxmlRegistrarAlumno

3. Comentarios

Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de la clase. Se tiene, por tanto, que evitar referencias al diseño funcional de la misma. 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, así como el uso de cajas u otro tipo de gráfico creado mediante códigos ASCII.

El formato de los diferentes tipos de comentarios se especificará a continuación:

3.1. Comentarios de documentación

Como norma es obligatorio proporcionar un comentario de documentación por cada clase (normal o de Java FX) o interface. Los comentarios de este tipo constaran de:

- Autor – Nombre de la persona que se encarga de crear y editar la clase
- Fecha – la fecha en la que se creó el componente

La estructura es:

/

** el comentario va aquí*

** aquí*

**y termina así */*

3.2. Comentarios de una sola línea

Dan retroalimentación sobre algún componente dentro del código, ojo, retroalimentación, en ningún caso un comentario va a explicar lo que hace un bloque de código. La estructura es la siguiente:

// aquí va el comentario

3.3. Comentarios de varias líneas

De igual forma se utilizan para dar retroalimentación a un bloque de código, la diferencia es que no caben en una sola línea, por tanto, se usan más. La estructura es:

/ inicia aquí*

Continua

.

.

*Termina así */*

4. Declaraciones

Para la declaración de las variables se utiliza una declaración de cada vez y no se permiten dejar variables locales sin inicializar. La codificación correcta sería:

private int entero = 0;

La declaración de las variables locales a una clase, método o bloque de código se realizan al principio del mismo y no justo antes de necesitarse la utilización de la variable.

La única excepción a esta regla son las variables que gestionan los bucles for. Las variables de avance de bucles for no podrán ser modificadas de ninguna manera fuera de la propia sentencia del bucle.

La duplicidad de los nombres de variables en diferentes niveles dentro de la misma clase se tiene que evitar.

5. Sentencias

Normas básicas son:

- Una sentencia por línea de código.
- Todo bloque de sentencias entre llaves.
- Después de cada sentencia, el corchete de apertura debe ir en la misma línea de la sentencia y el de cierre una línea después del último fragmento de código.

Para el caso de los for son obligatorias las tres condiciones del bucle: inicialización, condición de finalización y actualización del valor de la variable de avance y la variable de avance del bucle nunca podrá ser modificada dentro del propio bucle.

6. Identación y tamaños de líneas

La identación deberá aplicarse a toda estructura que esté lógicamente contenida una dentro de otra. Esta será de 4 espacios. De esta forma, se vera todo más claro. Ejemplo:

```
8       public static void main(String[] args) {
9           int suma = 2 + 2;
10          System.out.println("La suma es " + suma);
11      }
12  }
```

Las líneas no tendrán en ningún caso demasiados caracteres que impidan que se pueda leer en una pantalla. Un número máximo recomendable suele estar entre unos 70 y 90 caracteres, incluyendo los espacios de identación. Si una línea debe ocupar más caracteres, tiene que dividirse en dos o más líneas. Para dividir una línea en varias, utiliza los siguientes principios:

- Tras una coma.
- Antes de un operador, que pasará a la línea siguiente.
- Una construcción de alto nivel (por ejemplo, una expresión con paréntesis).
- La nueva línea deberá alinearse con una identación lógica, respecto al punto de ruptura

Ejemplos:

Dividir tras una coma:

funcion(expresionMuuuuyLarga1,

expresionMuuuyyyyLarga2,

expresionMuuuyyyLarga3);

Mantener la expresión entre paréntesis en la misma línea:

```
nombreLargo = nombreLargo2*
        (nombreLargo3 + nombreLArgo4)+
       4*nombreLargo5;
```

7. Espaciado

Se debe usar una sola línea en blanco para separar:

- Declaración del paquete
- Declaraciones de clase
- Constructores
- Métodos

y puede ser usado para separar grupos lógicos de:

- declaraciones de importación
- campos
- declaraciones