Grado en Ingeniería Informática

Principios de Desarrollo de Software

Curso 2018-2019

Grupo 82

Ejercicio Guiado – 2

NORMATIVAS Y PROPIEDAD COLECTIVA DE CÓDIGO

Rafael Rus Rus – 100363907

Abel Rodríguez Pérez – 100363855

Grupo 1

Normativa de codificación

Además de la normativa de codificación en .pdf se ha proporcionado la misma en .xml

P1: Estándar para la Organización de Archivos

Archivos de Código Fuente

- Se debe determinar en todo momento el formato de leyenda de derechos de autor que deberán tener todas las clases y los métodos mediante un comentario Javadoc. Es decir, será necesario colocar un comentario Javadoc al inicializar una clase o un método del código fuente donde será obligatorio escribir la etiqueta que muestre el autor del mismo.

```
/**
 * @author L
 */
public class LICENSE {
```

La utilización del comentario Javadoc será necesaria como estándar de la industria para documentar en todas las clases y los métodos el nombre del desarrollador, lo cual facilita la posterior compresión del código y el mantenimiento de la autoría de las diferentes partes del código fuente. También se deberá dejar por conveniencia una línea de separación entre los comentarios Javadoc y el anterior elemento del código:

```
import java.util.Calendar;
import java.util.Date;

/**
 * @author L
 */
public class LICENSE {
```

- Además de la etiqueta @author, los métodos tendrán que incluir de forma obligatoria en sus comentarios Javadoc (siempre que sea necesario) las etiquetas @return, @throws y @param. Estas dos últimas deberán llevar algo de texto para describirlo (la longitud máxima del texto será la misma que el resto de líneas del código).

```
/**
 * @author L
 * @return LIC
 * @param LIC ...
 */
public static LICENSE REVOKELICENSE(LICENSE LIC) {
return LIC;
}
```

Cabe destacar que cada uno de los parámetros estará en una línea del comentario Javadoc distinta para facilitar su posterior compresión.

- Las etiquetas de los comentarios Javadoc tienen que seguir el orden que se ha establecido: @author, @version, @return, @throws, @exception y @param. Al igual que éstas, se han dejado algunas etiquetas que podría utilizar cualquiera de los miembros del equipo de trabajo a su libre albedrio: @see, @since, @serialField, @serialData y @deprecated. Se deberá seguir este orden, si se quiere añadir una de estas etiquetas en los comentarios Javadoc.

- El control de versiones de todos los ficheros fuente (que contengan código fuente) y todas sus librerías y clases se gestionará a través del servicio GitLab, de tal forma que los cambios realizados en el código fuente por cualquiera de los dos miembros del equipo se comentaran y publicarán en el propio GitLab a través de Sourcetree.

Fichero de Clases

- Se deberá incluir una cabecera en cada fichero de clase que servirá en otras cosas para aportar información resumida de la misma, así como para tenerla en cuenta con el fin de facilitar su revisión/actualización posterior. En la cabecera de todas las clases tiene que estar escrito lo siguiente:

```
// Checkstyle: Copyright (C) 2019  L
// Version: 8.12

package org.galactic.empire.secret.software.licensing;
```

No deberá faltar ningún dato concreto de la cabecera, y las dos líneas del código tendrán que estar separadas por un salto de línea. La primera servirá como dato de copyright de la aplicación utilizada (es un ejemplo no real) y la segunda línea será la versión utilizada.

P2: Estándar para Nombres y Variables

Clases y Miembros de Clases

- Con el fin de facilitar su comprensión y su detención, los nombres de todos los métodos sean del tipo que sean (públicos o privados), deberán ser llamados con todas las letras mayúsculas y contener entre 1 y 20 caracteres.

- Al igual que la normativa anterior, también los nombres de todas las clases deberán ser llamados con letras mayúsculas y contener entre 1 y 20 caracteres. No podrán contener por regla ningún otro carácter (ni letras minúsculas ni ningún carácter numérico).

- Los nombres de todos los parámetros (a excepción de aquellos que se van a utilizar en las funciones de error) deberán ser llamados con letras mayúsculas y deberán contener entre 1 y 20 caracteres. Al igual que los parámetros pedidos por las funciones, los throws también deberán ser llamados con letras mayúsculas y no permitirán más de 20 caracteres.

- Los nombres de todos los miembros de las clases deberán ser nombrados con letras mayúsculas y contener entre 1 y 20 caracteres.

- Las variables locales del código fuente también seguirán ciertas reglas de escritura: deberán ser llamadas con letras minúsculas (todos los caracteres y no se admiten otro tipo) y tendrán que contener entre 3 y 10 caracteres. Así podrán ser distinguidas más fácilmente del resto de elementos del código fuente.

Visibilidad

- Deberá haber una línea de separación entre las declaraciones de variables, entre las declaraciones de paquetes y entre las declaraciones de clases, tal y como se muestra a continuación:

```
private String STATIONNAME;

private Date CREATIONDATE;

private Date EXPIRATIONDATE;

private boolean REVOKED;

private String SIGNATURE;
```

Esto facilitará la visibilidad a la hora de declarar los diferentes elementos del código fuente y servirá de apoyo a la siguiente normativa.

- En todo momento deberá haber una instrucción por línea del código fuente (en todas las clases y métodos). Esto será clave para la visualización de la práctica por parte de los integrantes del grupo de trabajo.

- Las declaraciones de todos los paquetes tendrán que estar separadas por 1 línea tanto por arriba como por abajo del código fuente. Así será más fácil separar los paquetes de la cabecera utilizada obligatoriamente en todas las clases y de las declaraciones de todos los imports.

- Se admitirá una longitud máxima de línea de 500 caracteres para facilitar a los miembros del equipo de trabajo su libre escritura del código y no tener un tope de longitud de línea muy bajo.

P3: Estándar para Métodos

Estructura de los Métodos

- La longitud máxima de un método ha de ser de 50 líneas con el fin de que los miembros del equipo no se extiendan mucho al definir los métodos del código fuente. También se podrán escribir comentarios en los métodos en cualquier, salvo encima de los comentarios Javadoc de los mismos (ya que deben tener una línea vacía por arriba).

Indentación y llaves

- Las líneas del código no pueden contener ningún carácter de tabulación en ninguna de las líneas del código. De esta forma, se bloqueará cualquier normativa de indentación que se considere innecesaria con respecto al nivel de las líneas del código y se eliminarán los problemas relacionados con la sangría. Además, todo el código estará arrastrado al lado izquierdo facilitando así su compresión:

```
public String GETSIGNATURETOKEN() throws LICENSINGEXCEPTION {
MessageDigest mda;
try {
mda = MessageDigest.getInstance("MD5");
} catch (NoSuchAlgorithmException e) {
throw new LICENSINGEXCEPTION("Error: no such hashing algorithm.")
}
String input =  SECRETSALT + "-" + SIGNATURE;
mda.update(input.getBytes(StandardCharsets.UTF_8));
byte[] digest = mda.digest();
String hex = String.format("%064x", new BigInteger(1, digest));
return hex;
}
```

P4: Estándar para el Tratamiento de Excepciones

- Con el fin de detectar algunos errores en el código y tratar las excepciones se ha establecido una normativa en los *for*, de forma que las variables de control que se utilizan en las mismas no pueden ser modificadas dentro del propio código del bucle *for*.