Normativa de código Transport4Future

1 Introducción

Este documento sirve como referencia a la normativa de código por la que se rige la empresa para su desarrollo de software de manera eficiente y sin problemas.

Un archivo de código en Java solo se describe como perteneciente al *Estilo de Transport4Future* si y solo si cumple las reglas que contiene este documento.

1.1 Terminología

Dentro de cada apartado se hará referencia a si es una norma, cuyo contenido es obligatorio; o una recomendación, la cual no es obligatoria en determinadas situaciones.

Para la mayoría de ellas se utilizará el plugin Checkstyle, en cuyo caso, se especificará aquí cuales utilizan dicha herramienta mediante.

2 Organización de Archivos

1) Norma: Cada clase deberá de estar en un fichero distinto
2) Norma: En la cabecera se deberá dar crédito a la empresa Transport for Future de la siguiente manera:
   //Transport4Future
   //Checks Java source code for adherence to a set of rules.
   //Copyright (C) 2020
3) Norma: La gestión de archivos se hará haciendo uso de la plataforma Gitlab:

https://pds.sel.inf.uc3m.es/users/sign_in

   Cada colaborador dispondrá de una cuenta que le permitirá compartir su código en el

   proyecto pertinente.

4) Norma: No se permitirá importar usando la notación “*”, evitando el exceso de código sin utilizar.
5) Recomendación: No se deberán importar paquetes que no se utilicen o que hayan sido importados en una línea de código anterior.
6) Norma: Los nombres de los paquetes comenzarán por una minúscula.

3 Nombres y Variables

1) Norma: Las constantes no podrán contener letras minúsculas.
2) Norma: Las variables estáticas comenzarán por letra minúscula.
3) Norma: Las variables de las clases comenzarán por letra minúscula.

4) Recomendación: Todo número que no se encuentre entre -1 y 2 se considerará como constante.
5) Recomendación: Se utilizará *this* en los nombres de los campos de la clase para distinguir entre variables locales de una función de los campos de una clase.
6) Norma: Constantes deberán hacer uso de "L" y no de "I" para evitar confusiones y errores innecesarios.
7) Norma: cada variable se inicializará en una línea, para evitar sobrecarga en la lectura del código.

4 Métodos

1) Norma: Los métodos y sus parámetros comenzarán por letra minúscula.
2) Norma: Los métodos recibirán como máximo 7 parámetros.
3) Norma: La longitud máxima de los métodos será de 150 caracteres.
4) Recomendación: se seguirá una identación de valor 4 como norma general.
5) Norma: se deberá simplificar al máximo los métodos que devuelvan valores booleanos, como por ejemplo:

```
    if (valid()):
  return false;
else
  return true;
```

deberá escribirse como

```
return !valid();
```

6) Recomendación: las clases que sobreescriban el método *equals()* también deberán sobrescribir el método *hashCode()* para evitar errores innecesarios.

5 Excepciones

1) Norma: Las excepciones deberán ser mostradas por la aplicación.
2) Norma: Las variables y métodos locales no deberán esconder campos definidos en la misma clase.

6 Otros

1) Norma: Los archivos tendrán un contenido de máximo 1950 caracteres.
2) Norma: Las líneas de código tendrán una longitud máxima de 150 caracteres.
3) Norma: *switch* deberá hacer uso de *default.*
4) Norma: se seguirá la norma general de no dejar espacios en blanco entre los caracteres "<" y ">".
5) Norma: los operadores comunes como "==" se pondrán en la nueva línea en caso de necesidad de continuar el código en otra línea.
6) Norma: sentencias especiales como *for, while* e *if* dejarán un espacio antes del siguiente símbolo.

7) Norma: los operadores comunes como “==” deberán dejar un espacio a cada lado para dar limpieza a la lectura del código.