Java Project Program : The 'Program code debug version to final version' program documentation file.

Version : 0.1. Demo

Made by : Jácint Papp

2023-01-27

Table of Contents

0. The introduction to the coding and documenting style :

0. The information required before :

There are several books written in documenting and coding style topics, please feel free to look for them.

1. The purpose of the file :

This file contains information regarding my documenting and coding style and the file is also the documentation of the 'Program code debug version to final version" program.

2. The base frame :

The documentation's '.odt' and '.pdf' files should start with a cover / title page and should be followed by the 'Table of contents' and then followed by the 'Topics'.

The documentation's '.txt' files should start with a title in a comment type row, usually with a // for stating the full name of the file.

After that there should be an empty indentation row and then there should be a row similar to the first row, which should state the 'Main Purpose' of the file.

After that there should be an empty indentation row and then the name of the creator of the file and then there should be two empty indentation rows.

Then there should be a row for describing the version number of the document and then there should be the date of creation and the changed content description.

After that there should be two empty indentation rows, following that, there should be the content listed in a 'Main Topic' and 'Sub Topic' list style.

Every written sentence in the documentation file(s) and in the code file(s) should be fit into one line and every line should be separated by at least one empty indentation row.

3. The documentation format :

Apart from the program's code itself, the documentation should be done in '.dia', '.docx', '.eps', '.json', '.md', '.odt', '.pdf', '.txt', '.xmi', '.xml' formats.

0. What is a 'Main Topic' :

A 'Main Topic' should contain the purpose of the paragraph in a short sentence.

A 'Main Topic' name should be followed by a : and an example value like :

0. The introduction to the coding and documenting style :

A 'Main Topic' should start from the left side.

A 'Main Topic' should be separated by at least two empty indentation rows above and below from the other document parts.

A 'Main Topic' should be able to contain 'Sub Topic(s)'.

Every 'Main Topic' should be numbered and the numbering should start from 0.

1. What is a 'Sub Topic' :

A 'Sub Topic' should be subdivision of the 'Main Topic' or purpose of the paragraph in a short sentence.

A 'Sub Topic' name should be followed by a : and an example value like :

0. What is a 'Main Topic' :

A 'Sub Topic' should start further right from the 'Main Topic'.

A 'Sub Topic' should be separated by at least two empty indentation rows above and below from the 'Main Topic' or from another 'Sub Topic'.

A 'Sub Topic' should be able to contain 'Segment(s)'.

Every 'Sub Topic' should be numbered and the numbering should start from 0.

2. What is a 'Segment' :

A 'Segment' should be the description of the paragraph that was mentioned in the 'Main Topic' or in the 'Sub Topic'.

A 'Segment' should not have a name.

A 'Segment' should start further right from the 'Sub Topic'.

A 'Segment' should be separated by at least one empty row above and below from the 'Main Topic' or from another 'Sub Topic' or from another 'Segment'.

//As you read these sentences you may notice this document is written in that format.

3. What is a 'Special Part' :

Some documented parts / segments should contain special characters like : '//' and this should be used for indicating a name definition or small description or a comment after the '//'.

There should be the '(' and ')' characters used for further description.

Some names should appear in the document(s) between the ' and the ' characters.

All of the special parts can be combined.

4. The 'Code' format :

0. The 'Comment' format :

The 'Comment' should be a general short description of the process or a short description about a method.

The 'Comment' should be always left aligned and should follow the 'Documentation Format'.

The 'Comment' code should be indented by one empty indentation row from below and from above from the program code.

1. The 'Class' format :

The 'Class' naming convention should be that every letter should be lowercase.

The 'Class' names should use abbreviation(s) if the naming word(s) are longer than four letters.

The 'Class' names should start with the 'filejava' character sequence and should be followed by the class main purpose and an example value like :

filejavamodudmain0

The 'Class' writing should follow the 'Documentation Format".

The 'Class' description 'Comment' field should contain the purpose of the class and the version of the file.

2. The 'Method' format :

The 'Method' commands should be written in one line and should be separated by an empty indentation row from the other lines.

The 'Method' component(s) or code segment(s) should be tabulated by four spaces.

The 'Method' should have the maximum of 2 calling parameters and an example value like :

public static ArrayList<String> meth0PublStatArraListStri(ArrayList<String> locaArraListStriPara0)

The 'Method' should use the 'Try-Catch Block' for catching any expected and unexpected error(s).

The 'Method' should return feedback messages to the user by the 'Logging' or by the 'Standard Output'.

The 'Method' logic block closing bracket should be in the same row as the last command of the logic block.

3. The naming conventions :

Every naming should start with a lowercase letter and at every new word should start with an uppercase letter except for class names and file names and folder names.

Every naming should state the type and the purpose of the 'Variable' or 'Class'.

Every sub part of the name should be maximum of 4 characters.

Every name should have the maximum of 120 characters length.

The 'Variable' name should be starting with either 'loca' or with 'glob' character sequence .

The 'Variable' name should contain the modification access type, if it is package type, then it should not state anything.

Example program code for the mentioned principles :

```
public static boolean meth0PublStatBool(String locaStriPara0) throws Exception {
   try {
      File locaFileStat0=new File(locaStriPara0);
      if (locaFileStat0.exists()) {
         return true;}
      return false;}}
   catch (Exception locaExceCatcExce0) {
      filejavamoduloggproclogg.meth0PublStatBool("ERROR - EXCEPTION in filejavamoduexam.meth0PublStatBool.");
      return false;}}
```

1. The specifications of the program :

0. The program's version history :

Version number	Short description	Change maker	Date
Version 0.1.	Core functions added.	Jácint Papp	2023-01-27

1. The program's task general description :

This software is called 'Program code debug version to final version' and it is considered a demo program.

This program should be able to make copies of the selected program code file’s and create new folders.

The program should run on 'Debian 10' or newer version(s) of this system and it also should run on 'Windows 10' or newer version(s) of this system.

Keep in mind this is a demo program for a programming showcase only, meaning it is not a fully completed or finished program and it may contain error(s) in the code or in the documentation file(s).

2. The installation of the program :

In this example the software itself is a '.java' file and it should be copied to the designated folder on 'Windows' systems like : 'C:/Myprograms/filejavamodu.java' or on 'Debian' systems like : '/home/test/filejavamodu.java'.

3. The uninstallation of the program :

The first step of the uninstallation should be the deletion of the program '.java' file from its installed location and an example value like : 'C:/MyPrograms/filejavamodu.java'.

2. The 'Main' module's description :

0. The 'Main' module's basic information and class :

This module is responsible for the 'Main' related processes.

This class methods should use the '**System.err.println**' command for logging.

A class is needed for a process named '**Module Main Process Main**' for starting the program.

1. The Module Main Process Main :

This process should contain the method for the program start.

0 :It should create the selected folder(s) and copy the selected file(s).

3. The program's life cycle progress :

In general the program's life cycle logging is important, it should be even used as a backlog and from these data the planning and the programming phase speed can be also measured.

2023-01-27 : Planned the program's specifications and the main function and created the file and also created the necessary file(s) related to the documentation.

4. References :

Java programming language : https://www.java.com/en/

Open Java Development Kit : https://openjdk.org/

5. Meanings :

'Class' : A template in the programming language for defining methods and variables.

'Comment' : A part of the code / documentation for clarifying purpose.

'Documentation Format' : The general principles for the documentation handling and writing.

'Java' : An object oriented programming language.

'Main Purpose' : The goal of the file / project / topic / segment.

'Main Topic' : A large part of the project that should be divided into smaller parts.

'Method' : A programmed part that contains the steps for the procedure in the class.

'Module' : A summary name of one function part of the program.

'Sub Topic' : A smaller, better defined part of the 'Main Topic'.

'Segment' : The description of a 'Main Topic' or a 'Sub Topic'.

'Standard Output' : A stream where the program writes the output and it is usually visible on the screen.

'Try-Catch Block' : A programmed part that should specify handlers for different exceptions.

'Variable' : A value, what depending on the conditions, should be able to change during the program's run.