# myTaxyService®

## Code Inspection

Matteo Greselin matr. 773600 ,
Seyeed Mohsen Kashfi Haghighi matr.859085

Version 1.0

# Contents

# 1.Classes and Methods

## Class:

**In our document there is just on class and the name is:**

1. InstallationConfigurator

## Methods:

**Methods that are assigned to our group:**

1. unConfigure ( final PropertySheet propSheet , final boolean validateFlag ) - Start Line: 134
2. configureUpdatetool ( ) - Start Line: 201
3. unconfigureUpdatetool ( ) - Start Line: 265
4. updateConfigFile ( ) - Start Line: 442

# 2.Functional role

The four methods which are assigned to us are included in a class called InstallationConfigurator. The name suggest that it should allow the configuration of the application during the installation process. With a quikly analysis we can see that this class contains different methods regarding to its scope: configure (), getConcurrentConfigutation (), unconfigure (), configureGlassfish (), configureUpdatetool () and so on. Methods are divided respectively to the specific task that they should compute. Comments are presented but probably are not completely satisfaction.

# 3.Issues

## Checklist:

### 1)

There are some of errors about the meaning of method variables, class variables...
Lines (78, 163, 164, 167)

### 2)

Naming convention about one-character variable is respected. In the assigned parts we have found it five times (lines 155, 250, 283, 305, 473). Each time it is used for temporary throwaway variables.

### 3)

No error has been found in case of naming convention.

### 4)

Not violations found with respect to Interface name.

### 5)

Some ambiguity in Method names has found for example: the word Updatetool in lines (201, 265, 363) are written like Updatetool, on the other hand in line ( 578) is written like UpdateTool.

### 6)

In InstallationConfigurator class we have different variables but only LOGGER is defined with all caps letters.

### 7)

No issue has been found.

## 8)

Codes in the most parts have 4 spaces on each line for indentation and 8 spaces when the code of the same line must be divided in two lines. We can find violation in the following block: - if block 449 - 459 - try - catch block 466 - 479

## 9)

There are some lines that use tabs for example(467, 471, 473, 450).

## 10)

In the assigned parts is used the Kernighan and Ritchie style (the first brace is on the same line of the instruction that opens the new block)

## 11)

No fault has been found about braces in If or try-catch or ...

## 12)

We cannot find any violation but the structure is not perfectly clear: on the top of the code, between line 62 and line 78 we have variables declarations mixed with constructor and no comments that describe those parts.

## 13)

There are lots of lines that exceed 80 characters and it seems that they are practical for example(122, 145, 146, 159, 206, 207, 321, 322, 323, 324, 344, 391, 449, 454, 456, 504, 505, 506, 525, 528, 535, 538, 559, 566, 570, 571, 588, 589, 591, 604).

## 14)

The following lines are too long (more than 120 characters) : (145, 146, 159, 449, 454, 456).

## 15)

There are some error that has found in wrapping line, line break after comma or an operator for example(159, 205, 206, 207, 214, 222, 224, 438, 449, 454, 456, 504, 505, 506, 525, 528, 535, 538, 557, 559, 564, 566, 570, 571, 588, 591, 598, 600, 604)

## 16)

Higher-level breaks are correctly used.

## 17)

These lines are not align with the begining of previous expression, so it becomes difficult for the reader. The lines are(82, 106, 119, 149, 156, 166, 167, 168, 177, 178, 179, 180, 181, 182).

**18)**

Comments have been used but they aren't completly clear. They should be more frequent. About variables, we don't understand why only configSuccessfull is commented. And also: -Comment at line 77 isn't clear. -For example updateConfigFile (line 442) hasn't documentation comment. -Also installationConfigurator (not assigned to us) hasn't documentation comment. -For the methods unConfigure could be usefull other documentation comment.

**19)**

Comments are completely reasonable and there is no faults in them.

**20)**

Out source file respect the condition: there is only one class/interface into it (one class in our case).

**21)**

There is just one public class and it is ok with no fault.

**22)**

**23)**

**24)**

First non-comment line is package statements and import statements follow it, as required.

**25)**

Method setConfigData starts in line 70 is declared in wrong position. it should be moved to a line after constructor. Static part is declared before variable gWaitCount, so the variable should be moved before the static part. Constant LOGGER in line 66 should be movied after variables.

**26)**

Class installationConfigurator includes all methods necessary for the installation and each one compute a specific part about installation process.

**27)**

There is no long method or big classes and this class is free of duplicate and no capsulation is occured.

**28)**

With respect to assigned methods, variables and class have the right visibility.

**29)**

Variable gWaitcount is declared wrongly and it should be declared in the spicific method that is used. Variables wsShortMgr, folderName and modifiedInstallDir should be declared publicaly.

**30)**

Each time that is required a new object, in the assigned parts is called a costructor

**31)**

There are some objects that are not initialized such as( 63, 64, 78, 270, 291)

**32)**

Not all variables are inizialize where they are declarated, someone becouse needed some parameters, for example productRef, jdkHome or configData. Someone should be inizialized where they are declarated, for example LOGGER, configSuccessfull or gWaitCount. Reguarding local variables we have detected the following situations: folderName is inizialized. shoutdownCommand isn't inizialized becouse it depends to results of if-condition. shoutdownCommandArray is inizialized. configCommand isn't inizialized becouse it depends to results of if-condition. configCommandArray is inizialized. Other local variables are out of assigned methods.

**33)**

Some variables are not declared in the proper position and they should be at the begining of the block such as(

**34)**

In assigned part there aren't methods call that required a lot of parameters.

**35)**

All of methods return value are used properly and no error has been found.

**36)**

The only method that return something is the first one : unConfigure. It require that method return a type 'ResultRecord' and it is correctly defined in line 159.

**37)**

There are no off-by-one error in array indexing.

**38)**

In assigned methods we have only the creation of some array, they aren't manipulated. In other methods of InstallationConfigurator class there are some cases where arrays are manipulated (createServerShortCuts, unconfigureGlassfish, unpackJars) but indexes are correclty defined so has been prevented the out-of-bound situation.

**39)**

No faults have been found in terms of constructor call when a new array item is desired.

**40)**

We havechecked taht all object (including strings) are compared with ”equals” and not with ’==’

**41)**

Some errors has been found related to the output spelling and grammer such as(232, 414, 480).

**42)**

Error messages, as exceptions, are comprehensive but they don't provide guidance as how to correct the problem. They only inform users on what has been the problem.

**43)**

There is a fault in line 414 and can not be determine that what is the output.

**44)**

Implementation avoids brutish programming as required

**45)**

No error has been found in the order of operators and paranthesis.

**46)**

In assigned parts has been found a proper use of parenthesis to avoid operator precedence problems

**47)**

There is no division in order to make zero denomination.

## 48)

In assigned methods there aren't integer arithmetic, plus sign is used for concatenate different strings. Other operators present in the class are boolean operator

## 49)

All of comparison and boolean operators are correct.

## 50)

command .getMessage() inserts in LOGGER.log() in each catch brach will return a details message about the exception that has occurred. The implementation presents a style where exceptions are caught and user is informed about it but software don't compute any alternative task. The only block where is done something different is try-catch block at lines 466-478: if an exception is caught, system will proceed to create a new jdkHome file.

## 51)

There is no implicit type conversion in our class.

## 52)

Relevant exeption are cought but nothing of them is computed in details. In the assigned parts there are only general exception that inform in formal way about the problem but in the first try-catch block we can see (lines 104/105) a comment that explain "Don't do anything as major error detection, is handled throughout this class where appropriate and fatal." so we can suppose that all exception are managed.

## 53)

After each catch there should be an appropriate action and LOGGER is not count as an action. So error lines are(156, 251, 284, 306, 332, 353, 495, 544, 573, 608)

## 54)

There aren't switch statements in assigned parts or in the entire class..

## 55)

There is no switch statement.

## 56)

There aren't loops in assigned methods

## 57)

No Faults have been detected.

## 58)

with relation to assigned parts the only one that includes actions on a file is the last, more precisely the try-catch block on lines 483-396. Here the file is opened, saved and closed and alse exceptions are managed. In order to guarantuee a correct closing for the file, could be usefull insert a try-finally block and insert the closing of the file in the finally branch. Finally brach will be compute also if will be detected an exception, so the file will always closed.

## 59)

There is no End Of File Condition.

## 60)

Same things that are explained at points

# 4.Additional Materials

In this project we often use dropbox as an Application which allow us to share the tasks which have been done.