**Concordia University**



**SOEN 6441  
Advanced Programming Practices**

**Fall 2010**

**Professor Pankaj Kamthan**

**Incredible Prime Root (IPR)**

**Artifact 4 Evaluation**

**Team B**

**Baljinder Kaur Multani**

**Jalal Noureddine**

**Sergey Khechumov**

**Table of Contents**

1. User Acceptance Testing 3

2. JUnit Testing 7

3. Conclusion 8

**1. User Acceptance Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario** | **User expectation** | **Actual result** | **Notes** | **Result** |
| ***1. Access*** |  |  |  |  |
| 1.1 The user can run the program easily | Double-click | Double-click works if the Java platform is installed on the system and is the default program to run these files. Otherwise user has to open the program with Java manually or through command line | The program can be run using the command “java IPR.jar” if Java is installed on the system under any operating system | The user uses Windows operating system and has Java installed and .jar files are assigned to it  Result was **OK** |
| ***2. Program*** |  |  |  |  |
| 2.1 [ User Story ] IPR-US-INPUT-01  The user should be able to input a prime number for root calculation | Input a prime number: 7  A valid prime number | No errors received | User story’s constraints must be valid | Result was **OK** |
|  | Input a prime number: 14  An invalid prime number  An error should be received | Error received |  | Result was **OK** |
| 2.2 [ User Story ] IPR-US-INPUT-02  The user should be able to input a number that will indicate a root degree | Input a natural number: 7  A Valid number | No errors received | User story’s constraints must be valid | Result was **OK** |
|  | Input negative number: -2  An invalid natural number  An error should be received | Error received |  | Result was **OK** |
| 2.3 [ User Story ] IPR-US-INPUT-03  The user should be able to input a number to indicate an accuracy of the result of this program | Input natural number: 4 | No errors received | User story’s constraints must be valid | Result was **OK** |
|  | Input string: “four”  An error should be received | Error received |  | Result was **OK** |
| 2.4 [ User Story ] IPR-US-OUTPUT-01  The user should have an option to save the result of the program into a file | User can choose to save the result (XML or TXT) through file-chooser box | That box will appear asking the user where to save the file |  | Result was **OK** |
|  | User created the XML file “result”  On the next program run, the user again saved the file as “result.xml” in the same location | The program overwrites the previous file any notification | A message box prompting the user for confirmation should have appeared | Result was **NOT OK** |
| 2.5 [ User Story ] IPR-US-OUTPUT-02  The result should be displayed on the screen | User input a large accuracy | The program displays the result and user can scroll horizontally |  | Result was **OK** |
|  | User wants to see a title of the result field and better name instead of “RUN” | The program shows “RUN” and there is no title for the result field. | However, tooltips are available but requires more effort from the user (hover mouse over and wait few seconds for the explanation to appear) | Result was **NOT OK** |
| 2.6 [ User Story] IPR-US-PROCESS-01  The time limit of operation should not exceed 1 second per digit of the result | The user wants to get results in a quick way (not waiting so long) | “The computation could take more time” message appears when the computation may take time. Anyways, the computation time never was longer than 1 second per digit. |  | Result was **OK** |
| 2.7 [ User Story] IPR-US-PROCESS-02  The user should be able to run the program on different operating systems | The user uses Windows and wants to run the program under his or her operating system | Java Virtual Machine (JVM) helps to run Java programs on different operating system platforms | The user should have Java installed on his or her operating system in order to run the program | Result was **OK** |
| 2.8 [ User Story] IPR-US-DESIGN-01  The user should be able to use a program in interacting with a graphical user interface | The user expect an easy-to-use GUI interface | The GUI interface is easy to use; Tooltips are used; Help module is available | A command line version that takes few arguments could be implemented as well (program will dramatically decrease in size and launch time) | Result was **OK** |
| 2.9 [ User Story] IPR-US-DESIGN-02  The user must have an access to a help section | The user wants to have a help module in case something is not clear | The program uses tooltips to display additional information  A help module and question marks (?) are available  Error message boxes are displayed at appropriate times |  | Result was **OK** |
|  | The user didn’t want to have HTML-based help files because another program (user agent) should be launched. | The help module is HTML-based and the default user agent will be launched by default. |  | Result was **NOT OK** |
| 2.10 [ User Story] IPR-US-DESIGN-03  The user should have an access to a functionality of the program by a main menu to facilitate the use of a program | The user wants to have a menu in which various functions are displayed | The program uses JMenu menu bar in order to perform a variety of functions |  | Result was **OK** |
| 2.11 [ User Story] IPR-US-DESIGN-04  The user has to view a simple short description of a GUI widget when he hovers over it in order to facilitate the use of the program | The user wants to view additional information in case something is not clear | The program uses tooltips to display additional information for most of the GUI widgets |  | Result was **OK** |
| ***3. Calculation*** |  |  |  |  |
| 3.1 The user wants valid rounded results | User expects to have a valid rounded result | If the message “The computation could take more time” appears, the result is not rounding | TODO: Fix the Digit By Digit rounding problem | Result was **NOT OK** |

## Table 1: User acceptance testing results

**2. JUnit Testing**

We used JUnit 4 to test our program. We used decision source coverage for conditions when it was possible. We decided that in our case it is sufficient to have acceptable results.

|  |  |  |
| --- | --- | --- |
| **TestCase** | **Class** | **Description** |
| CheckForPrimalityTest\_Prime | CheckForPrimality | Check the behaviour of the program in case if the number provided is prime. |
| CheckForPrimalityTest\_NotPrime | CheckForPrimality | Check the behaviour of the program in case if the number provided is not prime. |
| NewtonTest | Newton | Check some difficult conditions. |
| AccuracyTest | Accuracy | Check a difficult condition. |
| DigitByDigitPrepareRadicandTest | DigitByDigit | Check if the radicand is preparing in the good manner. |
| DegreeTest | Degree | Check a difficult condition. |
| RadicandTest | Radicand | Check a difficult condition. |
| RootNTest | RootN | Check some difficult conditions. |
| SaveResultTest | SaveResult | Check some difficult conditions. |

*Table 2: Test cases descriptions*

We didn’t have any positive JUnit test result; so, we consider the source code “accepted”.

**3. Conclusion**

There are some positive test case results in acceptance testing, but there are not in JUnit test cases. So, we should evaluate a time to fix these problems and to fix them in acceptable time.