Test plan

1 Introduction

This document is a plan of testing “FastCalc” application. The application must meet the requirements described in Requirements Document. In addition, the application should have no bugs and must have no critical bugs. Tests aim to validate and verify the application. The toolset and resources amount is limited. The testing team (that is, a group of people who is involved in testing) consists of a user and a developer. The time resources are limited to 12 hours, whereas estimated time is 8 hours.

2 Risk Issues

2.1 Description

“FastCalc”, complex calculator application is tested. Complex calculator named “Fastcalc” aims to be an easy in use and quite effective app to do simple operations with complex numbers. It does basic arithmetic operations with two complex operands. All operations are performed on floating point numbers. (See more in Requirements Document’s Introduction).

2.2 Quality attributes

The application should perform precise calculations fast and be easy to use for any potential user i.e. student.

3 Risk Issues

At the moment, the only known risk issue known is environment, in which application is launched and run. This includes assembly of hardware and software aspects.

4 Aspects

4.1 Multiplatform support

As requirement document states, the application should be able to run on any desktop platform, that supports JVM.

4.2 User Interface

The graphical user interface is to be compared with a scheme specified in the Requirement Document Appendix A (Figure 1 and Figure 2).

4.3 Usability

This aspect determines how easy it is for user to use the application. Apparently, it is important that the users have no issues launching the application and working with it. It is important how easy it is for users to get acquainted with the application and get used to it.

4.4 Calculations results

The results of complex number calculation are to be valid and precise enough (precision of three digits after the point is mentioned in Requirements Document 3.2.1). This aspect is the keystone of the application and must be tested very carefully. All of operations (see Requirements Document 3.1) must be tested. Exponential form from number form and vice versa deduction must be tested.

4.5 Performance

Under performance of the application result evaluation time is meant. As specified in Requirements Document 3.2.2, any complex number operation must take less than 1 second.

5 Test approaches

5.1 User tests

To test aspects 4.2, 4.3 user tests will be used, because those aspects have pure subjective nature. The aspect 4.1 will be also tested by user tests, as no other approaches can be considered here. Developer will test aspect 4.1, while a potential user outside the developers' team will test 4.2 and 4.3.

5.2 Unit tests

To test aspects 4.4 and 4.5, unit tests are used. For this purpose, toolset "ScalaTest" is used.

6 Pass/Fail Criteria

6.1 User test scenario

User tests should test use-case "calculate" mentioned in UseCase diagram. The event flow should be the same as mentioned in this diagram. User must get sure that buttons are clickable and that clicking them results in supposed action.

6.1 Pass criteria

As for aspects 4.5 and 4.4, the criteria is already specified (in Requirements Document). As for aspect 4.1, as resources for tests are limited, launching the application on two desktop platforms, that represent different platform families (i.e. Windows and Unix), is enough.

Usability (aspect 4.3) test is considered passed if user, who have never seen this application before, can understand how to use it and has idea of what do buttons do within 2 minutes using application.

User interface (aspect 4.2) test is considered passed if the count and relative positions of UI elements are the same as in Appendix A of Requirements Document and all text in app is readable.

7 Conclusion

Tests mentioned in plan should must prove that the application meets both functional and non-functional requirements (see Requirements Document 3.1 and 3.2) and is valid for any use.