İZMİR UNIVERSITY OF ECONOMICS

SE 318 – SOFTWARE VERIFICATION AND VALIDATION

OBSQUARE – QUIZ MAKER

BATUHAN CANLITÜRK

BEGÜM ÖZCEYLAN

ONUR DURUKAN

UNIT TEST DOCUMENT

Version *2.0*

05/01/2017

VERSION HISTORY

There are three version of test cases.

You can see our progress below.

Version 2.0 were applied on our software which was Quiz Maker version 2.0.

Version 1.0 was our first try for test cases. It was the initial test runs.

Version 1.1 was enhanced of version 1.0. There is a minor difference between them.

Version 2.0 is our official test cases for the project. As we said above, it was used on the Quiz Maker ver. 2.0.

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	*Onur Durukan*	*04/16/2017*	*Batuhan Canlıtürk*	*04/16/2017*	Test Cases draft
1.1	Begüm Özceylan	04/31/2017	Onur Durukan	04/31/2017	Drafts revision
2.0	Batuhan Canlıtürk	05/01/2017	Begüm Özceylan	05/01/2017	Final change on test cases

TABLE OF CONTENTS

1 INTRODUCTION

1.1 PURPOSE OF THE TEST CASE DOCUMENT

The Test Case document documents the functional requirements of the *Quiz Maker's* test cases. The intended audience is the project manager, project team, and testing team. Some portions of this document may on occasion be shared with the client/user and other stakeholder whose input/approval into the testing process is needed.

The Test Case documents are the main sources for keeping track of the test cases and for their results. They can be used for the future test.

1.2 CONSTRAINTS

Quiz Maker version 2.0 is based on Java. Therefore, JUnit were used during the test runs.

2 UNIT TEST FRAMEWORK: *JUNIT*

JUnit is a simple, open source framework to write and run repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks. JUnit features include:

- Assertions for testing expected results
- Test fixtures for sharing common test data
- Test runners for running tests

It is already situated on Eclipse.

3 TEST CASES

Test Case 1	
Test Definition	
Testing getAnswer method of Answer class	
Expected Value	
Set value : “A”	
Actual Value	
“A”	
Result of Test Case	***SUCCESSFUL***

Test Case 2	
Test Definition	
Testing getName method of Exam class	
Expected Value	
Set name: “Begum”	
Actual Value	
“Begum”	
Result of Test Case	***SUCCESSFUL***

Test Case 3	
Test Definition	
Testing getID method of Exam class	
Expected Value	
Set ID: 318	
Actual Value	
318	
Result of Test Case	***SUCCESSFUL***

Test Case 4	
Test Definition	
Testing getPaper method of Exam class	
Expected Value	
Set paper to other exam paper (exam1)	
Actual Value	
Exam1 paper	
Result of Test Case	***SUCCESSFUL***

Test Case 5	
Test Definition	
Testing getQuestions method of Exam class	
Expected Value	
Set questions to exam1's questions	
Actual Value	
Exam1 questions	
Result of Test Case	***SUCCESSFUL***

Test Case 6	
Test Definition	
Testing getQuestionOrder method of Exam class	
Expected Value	
Set questions to exam1's questions order	
Actual Value	
Exam1 questions order	
Result of Test Case	***SUCCESSFUL***

Test Case 7	
Test Definition	
Testing getMyAnswers method of Exam class	
Expected Value	
Set answers to A, B, C, D, and E respectively.	
Actual Value	
A, B, C, D, E	
Result of Test Case	***SUCCESSFUL***

Test Case 8	
Test Definition	
Testing solveExam method of Exam class	
Expected Value	
After the exam is prepared, it should provide environment for solving exam.	
Actual Value	
It runs and getting inputs correctly.	
Result of Test Case	***SUCCESSFUL***

Test Case 9	
Test Definition	
Testing prepareExam method of Exam class	
Expected Value	
Preparing exam correctly; 5 shuffled questions for the paper.	
Actual Value	
5 shuffled questions is ready.	
Result of Test Case	***SUCCESSFUL***

Test Case 10	
Test Definition	
Testing displayExam method of Exam class	
Expected Value	
It should print the questions to console.	
Actual Value	
All questions were printed.	
Result of Test Case	***SUCCESSFUL***

Test Case 11	
Test Definition	
Testing gradeExam method of Exam class	
Expected Value	
Exam should be graded according to my answers.	
Actual Value	
Grade was calculated correctly.	
Result of Test Case	***SUCCESSFUL***

Test Case 12	
Test Definition	
Testing getCorrect method of Question class	
Expected Value	
Set correct answer: “C”	
Actual Value	
“C”	
Result of Test Case	***SUCCESSFUL***

Test Case 13	
Test Definition	
Testing getName method of Question class	
Expected Value	
Set question name : “Question”	
Actual Value	
“Question”	
Result of Test Case	***SUCCESSFUL***

Test Case 14	
Test Definition	
Testing getAnswers method of Question class	
Expected Value	
Set first answer : “Adana”	
Actual Value	
“Adana”	
Result of Test Case	***SUCCESSFUL***

Test Case 15	
Test Definition	
Testing constructor of Answer class	
Expected Value	
Set answer: “Bolu” in constructor	
Actual Value	
“Bolu”	
Result of Test Case	***SUCCESSFUL***

Test Case 16	
Test Definition	
Testing constructor of Question class	
Expected Value	
Set name: “which one is above?” in constructor	
Actual Value	
“which one is above?”	
Result of Test Case	***SUCCESSFUL***

Test Case 17	
Test Definition	
Testing constructor of Exam class	
Expected Value	
Set ID: 318 in constructor	
Actual Value	
318	
Result of Test Case	***SUCCESSFUL***

Test Case 18	
Test Definition	
Testing default constructor of Answer class	
Expected Value	
Set answer: “no answer” in default constructor	
Actual Value	
“no answer”	
Result of Test Case	***SUCCESSFUL***

Test Case 19	
Test Definition	
Testing default constructor of Question class	
Expected Value	
Set name: “no question” in default constructor	
Actual Value	
“no question”	
Result of Test Case	***SUCCESSFUL***

Test Case 20	
Test Definition	
Testing default constructor of Exam class	
Expected Value	
Set name: “no name” in constructor	
Actual Value	
“no name”	
Result of Test Case	***SUCCESSFUL***

4 CONCLUSION

In the first phase, initial test cases were discussed and were designed according to our code. Enhancements were made and primary test cases were created. There are 20 test cases for the project. We tested the getter methods for the consistency. Along with the getters, setter methods were also checked implicitly. Both default and parametrized constructors were tested. Finally, all test cases passed successfully.