İZMİR UNIVERSITY OF ECONOMICS

SE 318 – SOFTWARE VERIFICATION AND VALIDATION

VENDING MACHINE

FURKAN KAVLAK

HATICE ERTUĞRUL

SIMGE GÜÇLÜKOL

UNİT TEST DOCUMENT

Version *2.0*

03/05/2017

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Furkan Kavlak	11/04/2017	Simge Güçlükol	12/04/2017	Test Case draft
2.0	Hatice Ertuğrul	02/05/2017	Furkan Kavlak	03/05/2017	Test Case review

TABLE OF CONTENTS

1 INTRODUCTION

1.1 PURPOSE OF THE TEST CASE DOCUMENT

The purpose of the test case document is testing functional requirements of Vending Machine. In this document all test cases of Vending Machine can be seen and it allows to finding errors at early stages.
The Test Case document documents the functional requirements of the ***Vending Machine*** test case. 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.

1.2 CONSTRAINTS

To use Junit , Java and Eclipse should be known. Also finding right function to test test cases is important. Sometimes there is no suitable way to test test cases.

2 UNIT TEST FRAMEWORK: *JUNIT*

In this project Junit is used as a unit test framework. This framework is for Java Programming Language.This framework is important in the development of test driven development. Junit is an open source framework and it allows people to write and run repeatable tests.

3 TEST CASES

Test Case 1	
Test Definition	
AddCoin(): It takes 1TL,5TL and 10TL and checks whether sum of them 16.	
Expected Value	
16	
Actual Value	
16	
Result of Test Case	***successful***

Test Case 2	
Test Definition	
AddCoinBalanceNotEqual():It takes 1TL,5TL and 10TL and checks whether sum of them 20	
Expected Value	
<Not> 20	
Actual Value	
16	
Result of Test Case	***successful***

Test Case 3	
Test Definition	
GiveProduct():Customer puts 20TL and takes Cola. It checks whether product is Cola.	
Expected Value	
“Cola”	
Actual Value	
“Cola”	
Result of Test Case	***successful***

Test Case 4	
Test Definition	
GiveProductNotEqual():Customer gives 20TL and takes Cola. It checks whether product is Chocolate.	
Expected Value	
<Not> “Chocolate”	
Actual Value	
“Cola”	
Result of Test Case	***successful***

Test Case 5	
Test Definition	
InsufficentBalance():Customer gives 1TL and wants to take Cola. Test checks that “Insufficient balance” message is seen on the screen	
Expected Value	
“Insufficient Balance”	
Actual Value	
“Insufficient Balance”	
Result of Test Case	***successful***

Test Case 6	
Test Definition	
EnoughBalance():Customer gives 20TL and wants to take Cola. Test checks "Insufficient Balance" message is not seen on the screen	
Expected Value	
<Not> "Insufficient Balance"	
Actual Value	
"Cola"	
Result of Test Case	***successful***

Test Case 7	
Test Definition	
GiveChange():Customer gives 5TL and takes Cola then wants to change. Test checks that whether change is 2.5TL	
Expected Value	
2.5TL	
Actual Value	
2.5TL	
Result of Test Case	***successful***

Test Case 8	
Test Definition	
GiveChangeNotEqual():Customer gives 5TL and takes Cola then wants to change. Test checks that whether change is 100TL	
Expected Value	
<Not> 100TL	
Actual Value	
2.5TL	
Result of Test Case	***successful***

Test Case 9	
Test Definition	
CheckBalance():Customer gives 10TL and takes Cola. Test checks that whether remaining coin is 7.5TL	
Expected Value	
7.5TL	
Actual Value	
7.5TL	
Result of Test Case	***successful***

Test Case 10	
Test Definition	
CheckBalanceNotEqual():Customer gives 10TL and takes Cola. Test checks that whether remaining coin is 10TL	
Expected Value	
<Not> 10TL	
Actual Value	
7.5TL	
Result of Test Case	***successful***

CONCLUSION

TO CONCLUDE THERE ARE 10 TEST CASES IN THIS DOCUMENT AND ALL OF THESE TEST CASES ARE SUCCESSFUL.