



SE 318 – SOFTWARE VERIFICATION AND VALIDATION

VENDING MACHINE

FURKAN KAVLAK

HATICE ERTUĞRUL

SİMGE GÜÇLÜKOL

UNIT TEST DOCUMENT

Version 3.0

12/05/2017

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
3.0	Furkan Kavlak	11/05/2017	Simge Güçlükol	12/05/2017	Completion of All Test Cases

TABLE OF CONTENTS

1 INTRODUCTION.....	4
1.1 Purpose of The Test Case Document.....	4
1.2 Constraints.....	4
2 UNIT TEST FRAMEWORK	4
3 TEST CASES.....	5
3.1 Test Case 1.....	5
3.2 Test Case 2.....	5
3.3 Test Case 3.....	5
3.4 Test Case 4.....	6
3.5 Test Case 5.....	6
3.6 Test Case 6.....	6
3.7 Test Case 7.....	7
3.8 Test Case 8.....	7
3.9 Test Case 9.....	7
3.10 Test Case 10.....	8
3.11 Test Case 11.....	8
3.12 Test Case 12.....	8
3.13 Test Case 13.....	9
3.14 Test Case 14.....	9
3.15 Test Case 15.....	9
3.16 Test Case 16.....	10
3.17 Test Case 17.....	10
3.18 Test Case 18.....	10
3.19 Test Case 19.....	11
3.20 Test Case 20.....	11
4 CONCLUSION.....	12

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	<i>successful</i>

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	<i>successful</i>

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	<i>successful</i>

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	<i>successful</i>

Test Case 5	
Test Definition	
InsufficientBalance():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	<i>successful</i>

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	<i>successful</i>

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	<i>successful</i>

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	<i>successful</i>

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	<i>successful</i>

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	<i>successful</i>

Test Case 11	
Test Definition	
CheckInStock():When the customer gives money and choose product, if product is available,there should be no “Out of Stock” message	
Expected Value	
<Not> “Out of Stock”	
Actual Value	
Cola	
Result of Test Case	<i>successful</i>

Test Case 12	
Test Definition	
CheckOutOfStock():When the customer gives money and choose product, if product is not available,there should be “Out of Stock” message	
Expected Value	
”Out of Stock”	
Actual Value	
“Out of Stock”	
Result of Test Case	<i>successful</i>

Test Case 13	
Test Definition	
AddCent():It takes 50 Cent and 25 Cent and checks whether sum of them is 75 Cent.	
Expected Value	
75 Cent	
Actual Value	
75 Cent	
Result of Test Case	<i>successful</i>

Test Case 14	
Test Definition	
AddCentNotEqual():It takes 50 Cent and 25 Cent and checks whether sum of them is not 50 Cent.	
Expected Value	
<Not> 50 Cent	
Actual Value	
75 Cent	
Result of Test Case	<i>successful</i>

Test Case 15	
Test Definition	
AddProduct(): It creates a machine and add a product which name is Redbull and check the first product of machine is Redbull.	
Expected Value	
"Redbull"	
Actual Value	
"Redbull"	
Result of Test Case	<i>successful</i>

Test Case 16	
Test Definition	
AddProductNotEqual(): It creates a machine and add a product which name is Redbull and check the first product of machine is not Fanta.	
Expected Value	
<Not> Fanta	
Actual Value	
Redbull	
Result of Test Case	<i>successful</i>

Test Case 17	
Test Definition	
GiveChangeBalanceZero(): It controls the balance is 0 after taking change.	
Expected Value	
0 TL	
Actual Value	
0 TL	
Result of Test Case	<i>successful</i>

Test Case 18	
Test Definition	
GiveChangeBalanceNotZero(): It controls the balance is not 2.5TL after taking change.	
Expected Value	
<Not> 2.5 TL	
Actual Value	
0 TL	
Result of Test Case	<i>successful</i>

Test Case 19	
Test Definition	
DefaultBalanceZero(): It adds a machine and check the default balance is 0 TL	
Expected Value	
0 TL	
Actual Value	
0 TL	
Result of Test Case	<i>successful</i>

Test Case 20	
Test Definition	
DefaultBalanceNotZero(): It adds a machine and check the default balance is not 5 TL	
Expected Value	
<Not> 5 TL	
Actual Value	
0 TL	
Result of Test Case	<i>successful</i>

CONCLUSION

**TO CONCLUDE THERE ARE 12 TEST CASES IN THIS DOCUMENT
AND ALL OF THESE TEST CASES ARE SUCCESSFUL.**

**THEREFORE THESE CODES CAN BE IMPLEMENTED FOR
VENDING MACHINE PROJECT.**