

BCS3323 – SOFTWARE TESTING & MAINTENANCE

**Lab Assessment 1**

Lecture’s Name

ROZLINA BINTI MOHAMED

Member’s Name

KHAIRUNNISAK BINTI NOORDIN CB12018

NORUL HUSNA BINTI OTHMAN CB13008

NUR SYUHAIDAH BINTI ISMAIL CB13006

(i). Test Case

|  |  |  |  |
| --- | --- | --- | --- |
| CONDITION | Test ID |  | EXPECTED RESULT |
| KUIH SIPUT, negative price | 1 | -40 | Please insert 40 cent |
| KUIH SIPUT, exactly price | 2 | 50 | Successful trading |
| KEREPEK UBI, negative price | 3 | -40 | Please insert 50 cent |
| KEREPEK UBI, exactly price | 4 | 65 | Successful trading |
| TEMPEYEK, negative price | 5 | -40 | Please insert 60 cent |
| TEMPEYEK, exactly price | 6 | 45 | Successful trading |
| KEREPEK PISANG, negative price | 7 | -40 | Please insert 70 cent |
| KEREPEK PISANG, exacly price | 8 | 85 | Successful trading |

Test Case 0

To buy kuih siput please insert 40 cent

Test Case 1

To buy kuih siput please insert 50 cent

Test Case 2

Please pick up your Kuih Siput and enjoy

Test Case 3

To buy kerepek ubi please insert 50 cent

Test Case 4

To buy kerepek ubi please insert 65 rcent

Test Case 5

Please pick up your kerepek ubi and enjoy

Test Case 6

To buy Tempeyek please insert 60 cent

Test Case 7

To buy Tempeyek please insert 45 cent

Test Case 8

Please pick up your Tempeyek and enjoy

Test Case 9

To buy kerepek pisang please insert 70 cent

Test Case 10

To buy kerepek pisang please insert 85 cent

Test Case 11

Please pick up your kerepek pisang and enjoy

(ii). Unit test

package venmac04;

import java.util.Scanner;

public class VenMac02 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

Scanner keyboard = new Scanner(System.in);

Dispenser d = new Dispenser();

double item1 = 50;

double item2 = 60;

double item3 = 65;

double item4 = 85;

System.out.print("\n 1 - KUIH SIPUT - 50 CENTS ");

System.out.print("\n 2 - KEREPEK UBI - 60 CENTS ");

System.out.print("\n 3 - TEMPEYEK - 65 CENTS ");

System.out.print("\n 4 - KEREPEK PISANG - 85 CENTS ");

System.out.print("\nEnter an item number: ");

int item = keyboard.nextInt();

System.out.print("Enter the amount paid: ");

double paid = keyboard.nextDouble();

if (item == 2 || item == 4 )

{

if (paid >= item2)

{

item = d.getCount();

paid = d.getProductCost();

System.out.println("Thank you for buying item " + item + ", your change is $" + (paid-item2) + ". Please come again!");

}

if (paid < item2)

{

System.out.println("Please insert another " + "$" + (item2-paid));

}

}

else if (item == 1)

{

if (paid >= item1)

{

System.out.println("Thank you for buying item " + item + ", your change is $" + (paid-item1) + ". Please come again!");

}

if (paid < item1)

{

System.out.println("Please insert another " + "$" + (item1-paid));

}

}

else if (item == 3)

{

if (paid >= item3)

{

System.out.println("Thank you for buying item " + item + ", your change is $" + (paid-item3) + ". Please come again!");

}

if (paid < item3)

{

System.out.println("Please insert another " + "$" + (item3-paid));

}

}

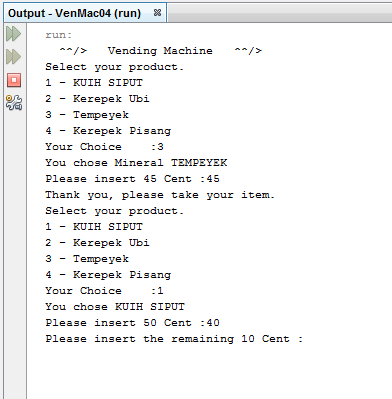
}

}

(iii). Decision Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Vending Machine test case | | Test ID’s | | | | | |
| #1 | #2 | #3 | #4 | #5 | #6 |
| Input Conditions | Select Item 1  (etc. kuih siput) | Y | Y | Y | Y | Y | N |
| Select Item 2  (etc. rempeyek) | N | N | N | N | Y | N |
| No. Of Item < 100 | Y | N | Y | N | N | N |
| Cost Of Item ≥ 50 cent | Y | N | Y | Y | N | N |
| Insert Coin 5 cent | Y | N | Y | N | N | N |
| Insert Coin 10 cent | N | N | N | N | Y | N |
| Insert Coin 20 cent | N | N | N | N | N | N |
| Insert Coin 50 cent | Y | N | Y | Y | Y | N |
| Output Conditions | Item Receive | [✓](http://en.wikipedia.org/wiki/%E2%9C%93) |  |  |  |  |  |
| Select Item |  | [✓](http://en.wikipedia.org/wiki/%E2%9C%93) |  | [✓](http://en.wikipedia.org/wiki/%E2%9C%93) | [✓](http://en.wikipedia.org/wiki/%E2%9C%93) | [✓](http://en.wikipedia.org/wiki/%E2%9C%93) |
| Insert Coin |  | [✓](http://en.wikipedia.org/wiki/%E2%9C%93) |  |  |  |  |
| Get Balance | [✓](http://en.wikipedia.org/wiki/%E2%9C%93) |  | [✓](http://en.wikipedia.org/wiki/%E2%9C%93) |  | [✓](http://en.wikipedia.org/wiki/%E2%9C%93) |  |

(iv). Unit Testing



package venmac04;

import java.util.Scanner;

/\*\*

\*

\* @author KAMI

\*/

public class VenMac04 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

Scanner console = new Scanner(System.in);

int siput = 100;

int ubi = 100;

int tempeyek = 75;

int pisang = 100;

System.out.println(" ^^/> Vending Machine ^^/>");

while (true) {

if ( siput == 0 && ubi == 0 && tempeyek == 0 && pisang == 0 ) {

System.out.println("Temporarily out of order");

break;

}

System.out.println("Select your product.");

System.out.println("1 - KUIH SIPUT ");

System.out.println("2 - Kerepek Ubi ");

System.out.println("3 - Tempeyek");

System.out.println("4 - Kerepek Pisang");

System.out.print("Your Choice :");

int YourChoice = console.nextInt();

while (YourChoice < 1 || YourChoice > 3) {

System.out.println("Please pick a valid product");

System.out.print("Your Choice :");

YourChoice = console.nextInt();

}

if (YourChoice == 1 && siput >0) {

System.out.println("You chose KUIH SIPUT");

siput--;

} else if (YourChoice == 1 && siput==0) {

System.out.println("The product is out of order, please pick something else");

}

if (YourChoice == 2 && ubi>0) {

System.out.println("You chose KEREPEK UBI");

ubi--;

} else if (YourChoice == 2 && ubi==0) {

System.out.println("The product is out of order, please pick something else");

}

if (YourChoice == 3 && tempeyek>0) {

System.out.println("You chose Mineral TEMPEYEK");

tempeyek--;

} else if (YourChoice == 3 && tempeyek==0) {

System.out.println("The product is out of order, please pick something else");

tempeyek--;

}

if (YourChoice == 4 && pisang>0) {

System.out.println("You chose KEREPEK PISANG");

pisang--;

} else if (YourChoice == 4 && pisang==0) {

System.out.println("The product is out of order, please pick something else");

pisang--;

}

if (YourChoice == 1 && siput >0){

int Prize = 50;

System.out.print("Please insert " + Prize + " Cent :");

int Coins = console.nextInt();

while (Coins < Prize) {

System.out.println("Please insert the remaining "+ (Prize - Coins) + " Cent :");

Coins += console.nextInt();

}

if (Coins == Prize) {

System.out.println("Thank you, please take your item.");

}

else if (Coins > 50) {

System.out.println("Thank you, please take your change ("+ (Coins - 50) + " Cent).");

}

else if (Coins < 0) {

System.out.println("Please insert a valid amount");

}

}

if (YourChoice == 2 && ubi >0){

int Prize = 65;

System.out.print("Please insert " + Prize + " Cent :");

int Coins = console.nextInt();

while (Coins < Prize) {

System.out.println("Please insert the remaining "+ (Prize - Coins) + " Cent :");

Coins += console.nextInt();

}

if (Coins == Prize) {

System.out.println("Thank you, please take your item.");

}

else if (Coins > 65) {

System.out.println("Thank you, please take your change ("+ (Coins - 65) + " Cent).");

}

else if (Coins < 0) {

System.out.println("Please insert a valid amount");

}

}

if (YourChoice == 3 && tempeyek >0){

int Prize = 45;

System.out.print("Please insert " + Prize + " Cent :");

int Coins = console.nextInt();

while (Coins < Prize) {

System.out.println("Please insert the remaining "+ (Prize - Coins) + " Cent :");

Coins += console.nextInt();

}

if (Coins == Prize) {

System.out.println("Thank you, please take your item.");

}

else if (Coins > 45) {

System.out.println("Thank you, please take your change ("+ (Coins - 45) + " Cent).");

}

else if (Coins < 0) {

System.out.println("Please insert a valid amount");

}

}

if (YourChoice == 4 && pisang >0){

int Prize = 85;

System.out.print("Please insert " + Prize + " Cent :");

int Coins = console.nextInt();

while (Coins < Prize) {

System.out.println("Please insert the remaining "+ (Prize - Coins) + " Cent :");

Coins += console.nextInt();

}

if (Coins == Prize) {

System.out.println("Thank you, please take your drink.");

}

else if (Coins > 45) {

System.out.println("Thank you, please take your change ("+ (Coins - 85) + " Cent)");

}

System.out.println("");

System.out.println("-------Welcome to the Vending Machine-------");

}

}}(v). SOLUTION

The flowchart is -



Information table

A = cost of item

B = price of item

Yes

Yes

Yes

No

No

No

**Select item**

**Read number of item**

**IF item available THEN**

**Define cost per item**

**ELSE**

**Select item**

**END IF**

**Display item cost**

**Insert coin**

**IF A > B THEN**

**Request success**

**ELSE**

**Insert coin**

**IF Coin balance available THEN**

**Get balance**

**ELSE**

**Receive Item**

**END IF**

**Statement coverage**

In this the test case is executed in such a way that every statement of the code is

executed at least once.

**Decision coverage**

Test coverage criteria requires enough test cases such that each condition in a decision

takes on all possible outcomes at least once, and each point of entry to a program or

subroutine is invoked at least once. That is, every branch (decision) taken each way,

true and false. It helps in validating all the branches in the code making sure that no

branch leads to abnormal behavior of the application.

**Path coverage**

In this the test case is executed in such a way that every path is executed at least once.

All possible control paths taken, including all loop paths taken zero, once, and multiple

(ideally, maximum) items in path coverage technique, the test cases are prepared based

on the logical complexity measure of a procedural design. In this type of testing every

statement in the program is guaranteed to be executed at least one time. Flow Graph,

Cyclomatic Complexity and Graph Metrics are used to arrive at basis path

**Statement coverage (SC)**

To calculate Statement Coverage, find out the shortest number of paths following

which all the nodes will be covered.

**1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13**

So statement coverage is 1.

**Decision coverage (DC)**

To calculate Branch Coverage, find out the minimum number of paths which will

ensure covering of all the edges.

**1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13**

**1 – 2 – 3 – 4 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 7- 8 - 9 – 10 – 12 – 13**

So decision coverage is 2

**Path coverage (PC)**

Path Coverage ensures covering of all the paths from start to end.

All possible paths are-

**1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13**

**1 – 2 – 3 – 4 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13**

**1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 7 – 8 – 9 – 10 – 11 – 12 – 13**

**1 – 2 – 3 – 4 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 7 – 8 – 9 – 10 – 11 – 12 – 13**

**1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 12 – 13**

**1 – 2 – 3 – 4 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 12 – 13**

**1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 7 – 8 – 9 – 10 – 12 – 13**

So path coverage is 7

Therefore, path coverage need the highest number of test case which is 7 compare to the other testing coverage.

|  |  |  |
| --- | --- | --- |
| Type of coverage | No of path | Percent of statement (%) |
| Statement coverage | 1 | (1/10)\*100 = 10 |
| Decision coverage | 2 | (2/10)\*100 = 20 |
| Path coverage | 7 | (7/10)\*100 =70 |

(vi). Often under-estimated and not much considered by developers, testing is however one of the key elements of software engineering, and a fundamental pillar without which no software can be put on the market with appropriate quality.

For the vending machine example, there are product is designed to work, gives plenty of cases

* Does it give the correct change
* How fast can it process the request
* What if an item is out of stock
* What if it is overfilled
* What if the user puts in too little money
* What if it is out of change

This is some advice from tester to programmer on how to Vending Machine should be implementing. There are several ideas:

1. Vending machine's program actual software from scratch using Java. There need to define the class for all of the products that are sold, the fields that must be met for each product to be dispensed, and the conditionals and loop fields to enable the customers to navigate their way through the menu. Include local variables allowing for refunding money and for exiting the vending machine program in case the customer changes his mind.
2. Withdraw cash by making selections and changes in vending machine program using the three scroll keys. The up and down arrows allow to scroll through selections, while the center button allows to confirm the selection. One action will need to take is to withdraw the cash. To do this, need to select the cash mode that will allow withdrawing the cash that the machine has earned.
3. Set desired prices of items by selecting the price mode. Once in the price mode input the items that will be sold in the vending machine, as well as their quantity. The stock list can be updated, so that items no longer stocked are removed and replaced with new items. If all products have the same price, select the multiple pricing functions. If the products have different prices, choose the individual pricing button. Use the up and down keys to increase or decrease the price.
4. Check the machine for errors. In error mode, the machine will display each machine function that is not functioning.