Project

CS589 – SOFTWARE TESTING AND ANALYSIS

Prateek Poste

A20329636

CS- 589 Fall: 2015 Project Report

Table of Contents

1. Introduction: 3

2. EFSM Model: 4

3. Model Based Testing: 5
Observations: 8

4. Default/Ghost Transition Testing: 9

5. Multiple Condition Testing and Branch Testing: 15
Non Executable Cases (N.E): 20

6. Test Suite: 22

7. Results of the Test Suite: 24

8. Conclusions: 37

9. Source Code and Test Drivers: 38

10. Steps to Execute the Test Driver: 51

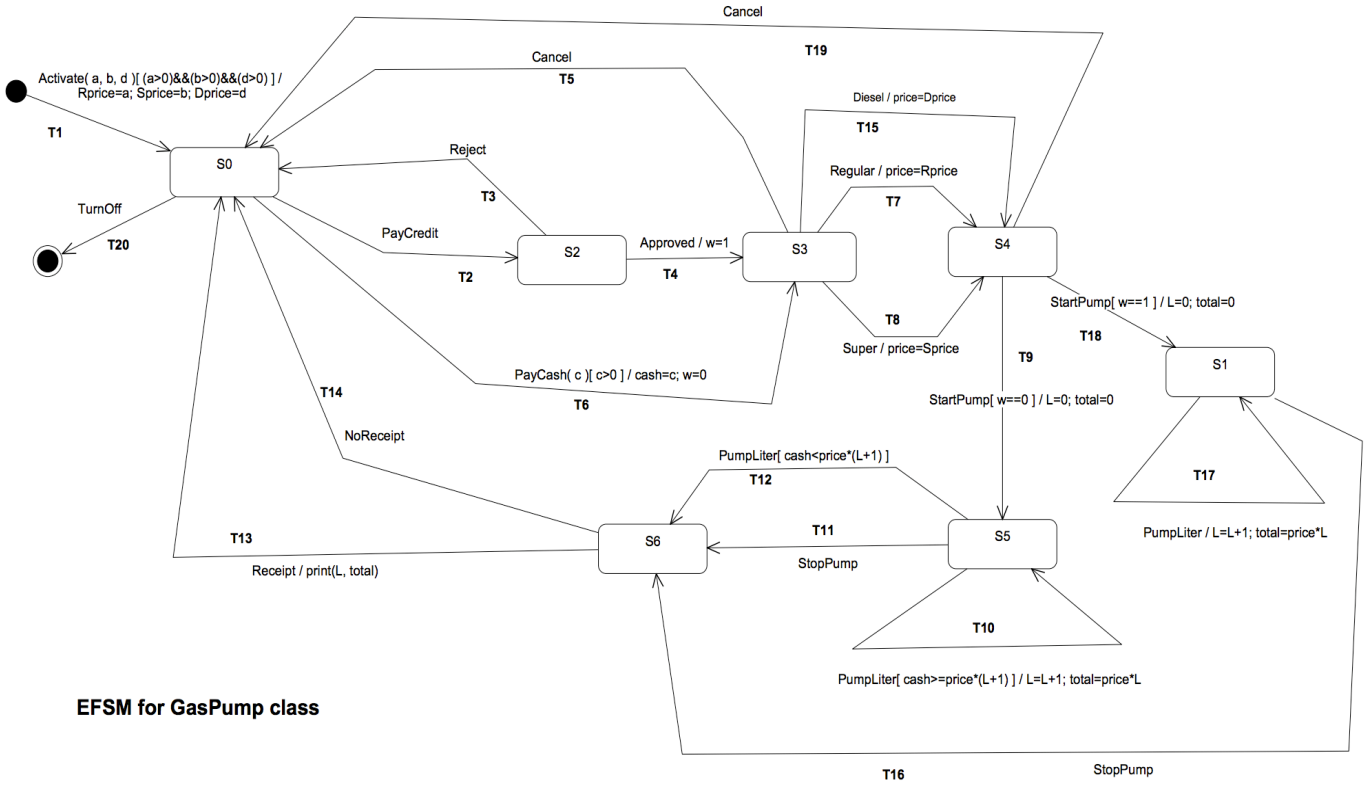
1. Introduction:

The sole purpose of this project is to test the GasPump class with three different testing techniques which are Model-based testing to show that 2-transition sequence testing is satisfied, Testing Default/Ghost transitions to show that the default transition testing is satisfied and the Multiple-condition Testing to show that multiple condition testing and also branch testing is satisfied.

We have to develop test cases based on the mentioned testing techniques and test them using a test driver that we have to develop for testing purpose. The test cases for the Model-based testing are based on the incoming and outgoing transitions of each state. These pairs can be identified easily using the EFSM diagram provided. To test the default/ghost transitions develop test cases for each state to check if any unnecessary functional calls are made to any other functions other than its own outgoing transitions. The test cases for the Multiple-condition testing are based on the source code of the program where we check for conditional statements. Both branch testing and multiple condition testing should be tested.

There might be some non-executable conditions and when these conditions are encountered we have to note them down and give reasoning for why it is not an executable condition. To test all the test cases we have to develop a test driver which should execute the methods of the GasPump class when called. In addition to the Test Driver also Testing-Oriented methods can be introduced to access the private variables of a class without altering the source code.

2. EFSM Model



EFSM for GasPump class

3. Model Based Testing:

The Model Based Testing involved 2 pair transition testing based on the EFSM of the *GasPump class* which was provided. Initially all the pairs in all the states were identified. Pairs were made on the base (incoming, outgoing). Then test cases were designed for them and the outcome of these test cases are mentioned below.

States	Incoming/ Outgoing	Transitions
S0	Incoming	T1, T3, T5, T13, T14, T19
	Outgoing	T2, T6, T20
S1	Incoming	T17, T18
	Outgoing	T16, T17
S2	Incoming	T2
	Outgoing	T3, T4
S3	Incoming	T4, T6
	Outgoing	T5, T7, T8, T15
S4	Incoming	T7, T8, T15
	Outgoing	T9, T18, T19
S5	Incoming	T9, T10
	Outgoing	T10, T11, T12
S6	Incoming	T11, T12, T16
	Outgoing	T13, T14

Test cases that cover all the transition pairs of the EFSM Diagram:

Test #	Test Case	Transitions Covered
Test #1	Activate(2.99, 3.99, 5.99), PayCredit(), Approved(), Super(), StartPump(), StopPump(), NoReceipt(),PayCash(5.99), Super(), StartPump(), PumpLiter(), StopPump(), NoReceipt(), TurnOff()	T1, T2, T4, T8, T18, T16, T14, T6, T8, T9, T10, T11, T14, T20
Test #2	Activate(2.99, 3.99, 5.99), PayCash(1), Diesel(), StartPump(), StopPump(), Receipt(), PayCredit(), Reject(), PayCash(1), Regular(), StartPump(), PumpLiter(), Receipt(), PayCash(5.99), Cancel(), TurnOff()	T1, T6, T15, T9, T11, T13, T2, T3, T6, T7, T9, T12, T13, T6, T5, T20
Test #3	Activate(2.99, 3.99, 5.99), PayCash(25.99), Regular(), Cancel(), PayCash(25.99), Super(), Cancel(), TurnOff()	T1, T6, T7, T19, T6, T8, T19, T20

Test #4	Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), PayCash(25.99), Cancel(), PayCredit(), Reject(), PayCredit(), Approved(), Diesel(), StartPump(), StopPump(), NoReceipt(), PayCredit(), Reject(), TurnOff()	T1, T2, T3, T6, T5, T2, T3, T2, T4, T15, T18, T16, T14, T2, T3, T20
Test #5	Activate(2.99, 3.99, 5.99), PayCash(11.99), Cancel(), PayCash(11.99), Diesel(), Cancel(), PayCredit(), Approved(), Super(), Cancel(), PayCash(), Super(), StartPump(), PumpLiter(), PumpLiter(), PumpLiter(), NoReceipt(), TurnOff()	T1, T6, T5, T6, T15, T19, T2, T4, T8, T19, T6, T8, T9, T10, T10, T12, T14, T20
Test #6	Activate(2.99, 3.99, 5.99), TurnOff()	T1, T20
Test #7	Activate(2.99, 3.99, 5.99), PayCredit(), Approved(), Regular(), StartPump(), PumpLiter(), PumpLiter(), StopPump(), Receipt(), TurnOff()	T1, T2, T4, T7, T18, T17, T17, T16, T13, T20
Test #8	Activate(2.99, 3.99, 5.99), PayCredit(), Approved(), Cancel(), TurnOff()	T1, T2, T4, T5, T20
Test #9	Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), TurnOff()	T1, T2, T3, T20

States:

State: S0

	Incoming	*Outgoing*	*Test Case #*
S0	T1	T2	9, 8, 7, 4, 2
	T1	T6	5, 3, 1
	T1	T20	4
	T3	T2	4
	T3	T6	4
	T3	T20	9, 4
	T5	T2	6
	T1	T2	9, 8, 7, 4, 2
	T1	T6	5, 3, 1
	T5	T6	5
	T5	T20	8, 2
	T13	T2	2
	T13	T6	2
	T13	T20	7
	T14	T2	4
	T14	T6	1
	T14	T20	5, 1
	T19	T2	5

State: S1

	Incoming	*Outgoing*	*Test Case #*
S1	T17	T16	7
	T17	T17	7
	T18	T16	4, 1
	T18	T17	7

State: S2

	Incoming	*Outgoing*	*Test Case #*
S2	T2	T3	9, 4
	T2	T4	8, 7, 5, 4, 2

State: S3

	Incoming	*Outgoing*	*Test Case #*
S3	T4	T5	8
	T4	T7	7, 2
	T4	T8	5
	T4	T15	4, 2
	T6	T5	5, 4, 2
	T6	T7	3
	T6	T8	3, 1
	T6	T15	5

State: S4

	Incoming	*Outgoing*	*Test Case #*
S4	T7	T9	2
	T7	T18	7
	T7	T19	3
	T8	T9	5, 1
	T8	T18	1
	T8	T19	3
	T15	T9	2
	T15	T18	4
	T15	T19	5

State: S5

	Incoming	*Outgoing*	*Test Case #*
S5	T9	T10	5, 1
	T9	T11	2
	T9	T12	2
	T10	T10	5
	T10	T11	1
	T10	T12	5

State: S6

	Incoming	*Outgoing*	*Test Case #*
S6	T11	T13	2
	T11	T14	1
	T12	T13	2
	T12	T14	5
	T16	T13	7
	T16	T14	4, 1

Observations:

Following transition pairs are solely depending on the value of "**w**":

CASE 1:

If **w=1:**

- T15 → T18
- T7 → T18
- T8 →T18

These pairs will execute when the value of **w=1**, and T15 → T9, T7 → T9 and T8 →T9 will not execute when **w=1.**

CASE 2:

If **w=0:**

- T15 → T9
- T7 → T9
- T8 →T9

These pairs will execute when the value of **w=0**, and T15 → T18, T7 → T18 and T8 →T18 will not execute when **w=0.**

4. Default/Ghost Transition Testing:

Default transitions also known as Ghost Transitions are the ones which are not seen in the EFSM model but still can be executed. The Execution of these transactions doesn't affect the state of the system and on their execution no system variable is affected. Additional test cases have been designed and included in the test suit TS.txt. These tests are covered and their reports are as follows.

Test Cases:

Each test case is covering each state of the EFSM model:

Test Case #	*Test Case*
10	Activate(2.99, 3.99, 5.99), Reject(), Approved(), Cancel(), PayCash(-25.99), Regular(), Super(), StartPump(), PumpLiter(), StopPump(), PumpLiter(), Receipt(), NoReceipt(), Diesel(), StopPump(), PumpLiter(), StartPump(), Cancel()
11	Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), Approved(), Cancel(), PayCash(-25.99), Regular(), Super(), StartPump(), PumpLiter(), StopPump(), PumpLiter(), Receipt(), NoReceipt(), Diesel(), StartPump(), Cancel(), TurnOff()
12	Activate(2.99, 3.99, 5.99), PayCredit(), Cancel(), PayCash(-25.99), Regular(), Super(), StartPump(), PumpLiter(), StopPump(), PumpLiter(), Receipt(), NoReceipt(), Diesel(), StopPump(), PumpLiter(), StartPump(), Cancel(), TurnOff()
13	Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), Approved(), PayCash(-25.99), StartPump(), PumpLiter(), StopPump(), PumpLiter(), Receipt(), NoReceipt(), StopPump(), PumpLiter(), StartPump(), Cancel(), TurnOff()
14	Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), Approved(), Cancel(), PayCash(-25.99), Regular(), Super(), PumpLiter(), StopPump(), PumpLiter(), Receipt(), NoReceipt(), Diesel(), StopPump(), PumpLiter(), TurnOff()
15	Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), Approved(), Cancel(), PayCash(-25.99), Regular(), Super(), StartPump(), Receipt(), NoReceipt(), Diesel(), StopPump(), PumpLiter(), StartPump(), Cancel(), TurnOff()
16	Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), Approved(), Cancel(), PayCash(-25.99), Regular(), Super(), StartPump(), PumpLiter(), StopPump(), PumpLiter(), Diesel(), StopPump(), PumpLiter(), StartPump(), Cancel(),TurnOff()

State: S0

Function	Transition	Test Case Covered
Activate(2.99, 3.99, 5.99)	T1	Test Case #10
Activate(2.99, 3.99, 5.99)	T1	Test Case #10
Reject()	T3	Test Case #10
Approved()	T4	Test Case #10
Cancel()	T5	Test Case #10
PayCash(-25.99)	T6	Test Case #10
Regular()	T7	Test Case #10
Super()	T8	Test Case #10
Startpump()	T9	Test Case #10
PumpLiter()	T10	Test Case #10
StopPump()	T11	Test Case #10
PumpLiter()	T12	Test Case #10
Receipt()	T13	Test Case #10
NoReceipt()	T14	Test Case #10
Diesel()	T15	Test Case #10
StopPump()	T16	Test Case #10
PumpLiter()	T17	Test Case #10
StartPump()	T18	Test Case #10
Cancel()	T19	Test Case #10

State: S1

Function	Transition	Test Case Covered
Activate(2.99, 3.99, 5.99)	T1	Test Case #11
PayCredit()	T2	Test Case #11
Approved()	T4	Test Case #11
Regular()	T7	Test Case #11
StartPump()	T18	Test Case #11
Activate(2.99, 3.99, 5.99)	T1	Test Case #11
Cancel()	T5	Test Case #11
Reject()	T3	Test Case #11

PayCash(-25.99)	T6	Test Case #11
Regular()	T7	Test Case #11
Super()	T8	Test Case #11
PumpLiter()	T10	Test Case #11
StartPump()	T9	Test Case #11
StopPump()	T11	Test Case #11
Receipt()	T13	Test Case #11
NoReceipt()	T14	Test Case #11
Diesel()	T15	Test Case #11
StartPump()	T18	Test Case #11
Cancel()	T19	Test Case #11
TurnOff()	T20	Test Case #11

State: S2

Function	Transition	Test Case Covered
Activate(2.99, 3.99, 5.99)	T1	Test Case #12
PayCredit()	T2	Test Case #12
Activate(2.99, 3.99, 5.99)	T1	Test Case #12
PayCredit()	T2	Test Case #12
PayCash(-25.99)	T6	Test Case #12
Cancel()	T5	Test Case #12
Regular()	T7	Test Case #12
Super()	T8	Test Case #12
StartPump()	T9	Test Case #12
PumpLiter()	T10	Test Case #12
StopPump()	T11	Test Case #12
PumpLiter()	T12	Test Case #12
Receipt()	T13	Test Case #12
NoReceipt()	T14	Test Case #12
Diesel()	T15	Test Case #12
StopPump()	T16	Test Case #12
PumpLiter()	T17	Test Case #12
StartPump()	T18	Test Case #12
Cancel()	T19	Test Case #12
TurnOff()	T20	Test Case #12

State: S3

Function	Transition	Test Case Covered
Activate(2.99, 3.99, 5.99)	T1	Test Case #13
PayCredit()	T2	Test Case #13
Approved()	T4	Test Case #13
Activate(2.99, 3.99, 5.99)	T1	Test Case #13
PayCredit()	T2	Test Case #13
Reject()	T3	Test Case #13
PayCash(-25.99)	T6	Test Case #13
StartPump()	T9	Test Case #13
PumpLiter()	T10	Test Case #13
StopPump()	T11	Test Case #13
PumpLiter()	T12	Test Case #13
Receipt()	T13	Test Case #13
NoReceipt()	T14	Test Case #13
StopPump()	T16	Test Case #13
PumpLiter()	T17	Test Case #13
StartPump()	T18	Test Case #13
Cancel()	T19	Test Case #13
TurnOff()	T20	Test Case #13

State: S4

Function	Transition	Test Case Covered
Activate(2.99, 3.99, 5.99)	T1	Test Case #14
PayCredit()	T2	Test Case #14
Approved()	T4	Test Case #14
Super()	T8	Test Case #14
Activate(2.99, 3.99, 5.99)	T1	Test Case #14
Reject()	T3	Test Case #14
Approved()	T4	Test Case #14
Cancel()	T5	Test Case #14
PayCash(-25.99)	T6	Test Case #14
Regular()	T7	Test Case #14
Super()	T8	Test Case #14
PumpLiter()	T10	Test Case #14

StopPump()	T11	Test Case #14
PumpLiter()	T12	Test Case #14
Receipt()	T13	Test Case #14
NoReceipt()	T14	Test Case #14
Diesel()	T15	Test Case #14
StopPump()	T16	Test Case #14
PumpLiter()	T17	Test Case #14
TurnOff()	T20	Test Case #14

State: S5

Function	Transition	Test Case Covered
Activate(2.99, 3.99, 5.99)	T1	Test Case #15
PayCash(20)	T6	Test Case #15
Super()	T8	Test Case #15
StartPump()	T9	Test Case #15
Activate(2.99, 3.99, 5.99)	T1	Test Case #15
PayCredit()	T2	Test Case #15
Reject()	T3	Test Case #15
Approved()	T4	Test Case #15
Cancel()	T5	Test Case #15
PayCash(-25.99)	T6	Test Case #15
Regular()	T7	Test Case #15
Super()	T8	Test Case #15
StartPump()	T9	Test Case #15
Receipt()	T13	Test Case #15
NoReceipt()	T14	Test Case #15
Diesel()	T15	Test Case #15
StopPump()	T16	Test Case #15
PumpLiter()	T17	Test Case #15
StartPump()	T18	Test Case #15
Cancel()	T19	Test Case #15
TurnOff()	T20	Test Case #15

State: S6

Function	Transition	Test Case Covered
Activate(2.99, 3.99, 5.99)	T1	Test Case #16
PayCredit()	T2	Test Case #16
Approved()	T4	Test Case #16
Diesel()	T15	Test Case #16
StartPump()	T18	Test Case #16
StopPump()	T16	Test Case #16
Activate(2.99, 3.99, 5.99)	T1	Test Case #16
Reject()	T3	Test Case #16
Approved()	T4	Test Case #16
Cancel()	T5	Test Case #16
PayCash(-25.99)	T6	Test Case #16
Regular()	T7	Test Case #16
Super()	T8	Test Case #16
StartPump()	T9	Test Case #16
PumpLiter()	T10	Test Case #16
StopPump()	T11	Test Case #16
PumpLiter()	T12	Test Case #16
Diesel()	T15	Test Case #16
StopPump()	T16	Test Case #16
PumpLiter()	T17	Test Case #16
StartPump()	T18	Test Case #16
Cancel()	T19	Test Case #16
TurnOff()	T20	Test Case #16

5. Multiple Condition Testing and Branch Testing:

Test Case #	Test Case
17	Activate(2.99, 3.99, 6), PayCash(6), Diesel(), StartPump(), PumpLiter(), StopPump(), NoReceipt(), TurnOff()
18	Activate(2.99, 5, 5.99), PayCash(5), Regular(), StartPump(), StopPump(), NoReceipt(),TurnOff()
19	Activate(2, 8, 10), PayCash(10), Diesel(), StartPump(), PumpLiter(), PumpLiter(), Receipt(), TurnOff()
20	Activate(8, 9, 10), PayCredit(), TurnOff(), Approved(), Diesel(), StartPump(), PumpLiter(), StopPump(), NoReceipt(), TurnOff()
21	Activate(4, 5, 6), PayCredit(), Cancel(), Regular(), Approved(), Super(), Cancel(), PayCash(5), Cancel(), NoReceipt(), TurnOff()
22	Activate(4, 5, 6), PayCredit(), Approved(), Cancel(), PayCash(-5), PayCash(10), PayCash(-5), PayCash(5), Regular(), Super(), Diesel(), StartPump(), StartPump(), PumpLiter(), PumpLiter(), PumpLiter(), PumpLiter(), StopPump(), Receipt(), Receipt(), TurnOff()
23	Activate(-2.99, -3.99, -4.99), Activate(-2.99, -3.99, 4.99), Activate(-2.99, 3.99, -4.99), Activate(-2.99, 3.99, 4.99), Activate(2.99, -3.99, -4.99), Activate(2.99, -3.99, 4.99), Activate(2.99, 3.99, -4.99), Activate(2.99, 3.99, 4.99), Activate(-2.99, -3.99, -4.99), Activate(-2.99, -3.99, 4.99), Activate(-2.99, 3.99, -4.99), Activate(-2.99, 3.99, 4.99), Activate(2.99, -3.99, -4.99), Activate(2.99, -3.99, 4.99), Activate(2.99, 3.99, -4.99), Activate(2.99, 3.99, 4.99), PayCredit(), PayCredit(), Reject(), Reject(), Approved(), TurnOff()

a) Activate():

if((k == –1) && (a>0) && (b>0) && (d>0))

k == –1	a>0	b>0	d>0	Test Case Covered
T	F	F	F	Test Case #23
T	F	F	T	Test Case #23
T	F	T	F	Test Case #23
T	F	T	T	Test Case #23
T	T	F	F	Test Case #23
T	T	F	T	Test Case #23
T	T	T	F	Test Case #23
T	T	T	T	Test Case #23
F	F	F	F	Test Case #23
F	F	F	T	Test Case #23

F	F	T	F	Test Case #23
F	F	T	T	Test Case #23
F	T	F	F	Test Case #23
F	T	F	T	Test Case #23
F	T	T	F	Test Case #23
F	T	T	T	Test Case #23

b) PayCredit():

if(k==0)

k==0	Test Case Covered
T	Test Case #22
F	Test Case #22

c) Reject():

if(k==2)

k==2	Test Case Covered
T	Test Case #23
F	Test Case #23

d) Approved():

if(k==2)

k==2	Test Case Covered
T	Test Case #22
F	Test Case #23

e) Cancel():

if(k==3 || k==4)

k==3	k==4	Test Case Covered
T	F	Test Case #22
T	T	Non-Executable
F	F	Test Case #21
F	T	Test Case #21

if (w==0)

w==0	Test Case Covered
T	Test Case #21

F	Test Case #22

f) PayCash():

if(k==0 && c>0)

k==0	c>0	Test Case Covered
T	F	Test Case #22
T	T	Test Case #22
F	F	Test Case #22
F	T	Test Case #22

g) Regular():

if(k==3)

k==3	Test Case Covered
T	Test Case #22
F	Test Case #21

h) Super():

if(k==3)

k==3	Test Case Covered
T	Test Case #21
F	Test Case #22

i) Diesel():

if(k==3)

k==3	Test Case Covered
T	Test Case #20
F	Test Case #22

j) StartPump():

if(k==4)

k==4	Test Case Covered
T	Test Case #22
F	Test Case #22

k) PumpLiter():

if(k==5)

k==5	Test Case Covered
T	Test Case #22
F	Test Case #22

if(w==1 || ((cash>=price*(L+1)) && (w==0)))

w==1	cash>=price*(L+1)	w==0	Test Case Covered
T	F	F	Test Case #20
T	F	T	Non-Executable
T	T	F	Non-Executable
T	T	T	Non-Executable
F	F	F	Non-Executable
F	F	T	Test Case #22
F	T	F	Non-Executable
F	T	T	Test Case #22

if((w==0) && (cash<price*(L+1))

w==0	cash<price*(L+1)	Test Case Covered
T	F	Non-Executable
T	T	Test Case #22
F	F	Non-Executable
F	T	Non-Executable

if((w==0) && (total<cash))

w==0	total<cash	Test Case Covered
T	F	Test Case #19
T	T	Test Case #22
F	F	Non-Executable
F	T	Non-Executable

l) StopPump():

if(k==5)

k==5	Test Case Covered
T	Test Case #20
F	Test Case #22

if((w==0) && (total<cash))

w==0	total<cash	Test Case Covered
T	F	Test Case #17
T	T	Test Case #18
F	F	Test Case #20
F	T	Non-Executable

m) NoReceipt():

if(k==6)

k==5	Test Case Covered
T	Test Case #20
F	Test Case #21

n) Receipt():

if(k==6)

k==5	Test Case Covered
T	Test Case #22

F	Test Case #22

o) TurnOff():

if(k==0)

k==5	Test Case Covered
T	Test Case #23
F	Test Case #20

Non-Executable Case(N.E.):

In Multiple Condition testing we observe some Non-Executable Cases. They are as Follows:

Case	Function	Conditions			Test Case Covered
	Cancel()	**k==3**	**k==4**		
Case# 1		T	T		N.E.
	PumpLiter()	**w ==1**	**cash>=price*(L+1)**	**w==0**	
Case# 2		F	F	F	N.E.
Case# 3		F	T	F	N.E.
Case# 4		T	F	T	N.E.
Case# 5		T	T	F	N.E.
Case# 6		T	T	T	N.E.
		w == 0	**cash < price*(L+1)**		
Case# 7		F	F		N.E.
Case# 8		F	T		N.E.
Case# 9		T	F		N.E.
		w==0	**total<cash**		
Case# 10		F	F		N.E.
Case# 11		F	T		N.E.
	StopPump()	**w == 0**	**total < cash**		
Case# 12		F	T		N.E.

Description of the Non-Executable Cases:

CASE 1: Value of "**k**" cannot be 3 and 4 at the same time.

CASE 2: The initial value of w=0 so it is not possible that the value of w should be other that 0 or 1, so in this case the value of "**w**" is neither 0 nor 1 which is not possible.

CASE 3: The initial value of w=0 so it is not possible that the value of w should be other that 0 or 1, so in this case the value of "**w**" is neither 0 nor 1 which is not possible.

CASE 4: The value of "**w**" cannot be 0 and 1 at the same time.

CASE 5: In this case, w=1 means the payment is done by credit card and at the same time cash>=price*(L+1) is true means it is accepting cash also which is not possible. So, this case is not possible.

CASE 6: The value of "**w**" cannot be 0 and 1 at the same time.

CASE 7: In this case, w=0 is false that means payment is done by card w=1 so it will never go into the else condition.

CASE 8: In this case, w=0 is false that means payment is done by card w=1 so it will never go into the else condition.

CASE 9: This is because when w = 0, and cash <price(L+1) is false, it means cash>=price(L+1) is true. When this is the case the execution never reaches the else point.

CASE 10: In this case, w=0 is false that means payment is done by card so the function will never reach at that point to execute.

CASE 11: In this case, w=0 is false that means payment is done by card so the function will never reach at that point to execute.

CASE 12: In this case, w=0 is false that means payment is done by card so the function will never reach at that point to execute.

6. Test Suite:

Test#1: Activate 2.99 3.99 5.99PayCash 25.99 Super StartPump StopPump NoReceipt PayCash 25.99 Super StartPump PumpLiter StopPump NoReceipt TurnOff

Test#2: Activate 2.99 3.99 5.99 PayCredit Approved Diesel StartPump StopPump Receipt PayCredit Approved Regular StartPump PumpLiter Receipt PayCash 25.99 Cancel TurnOff

Test#3: Activate 2.99 3.99 5.99 PayCash 25.99 Regular Cancel PayCash 25.99 Super Cancel TurnOff

Test#4: Activate 2.99 3.99 5.99 PayCredit Reject PayCash 25.99 Cancel PayCredit Reject PayCredit Approved Diesel StartPump StopPump NoReceipt PayCredit Reject TurnOff

Test#5: Activate 2.99 3.99 5.99 PayCash 11.99 Cancel PayCash 11.99 Diesel Cancel PayCredit Approved Super Cancel PayCash 6 Super StartPump PumpLiter PumpLiter PumpLiter NoReceipt TurnOff

Test#6: Activate 2.99 3.99 5.99 TurnOff

Test#7: Activate 2.99 3.99 5.99 PayCredit Approved Regular StartPump PumpLiter PumpLiter StopPump Receipt TurnOff

Test#8: Activate 2.99 3.99 5.99 PayCredit Approved Cancel TurnOff

Test#9: Activate 2.99 3.99 5.99 PayCredit Reject TurnOff

Test#10: Activate 2.99 3.99 5.99 Activate 2.99 3.99 5.99 Reject Approved Cancel PayCash -25.99 Regular Super StartPump PumpLiter StopPump PumpLiter Receipt NoReceipt Diesel StopPump PumpLiter StartPump Cancel

Test#11: Activate 2.99 3.99 5.99 PayCredit Approved Regular StartPump Activate 2.99 3.99 5.99 Cancel Reject Cancel PayCash -25.99 Regular Super PumpLiter StartPump StopPump Receipt NoReceipt Diesel StartPump Cancel TurnOff

Test#12: Activate 2.99 3.99 5.99 PayCredit Activate 2.99 3.99 5.99 PayCredit PayCash -25.99 Cancel Regular Super StartPump PumpLiter StopPump PumpLiter Receipt NoReceipt Diesel StopPump PumpLiter StartPump Cancel TurnOff

Test#13: Activate 2.99 3.99 5.99 PayCredit Approved Activate 2.99 3.99 5.99 PayCredit Reject PayCash -25.99 StartPump PumpLiter StopPump PumpLiter Receipt NoReceipt StopPump PumpLiter StartPump Cancel TurnOff

Test#14: Activate 2.99 3.99 5.99 PayCredit Approved Super Activate 2.99 3.99 5.99 Reject Approved Cancel PayCash -25.99 Regular Super PumpLiter StopPump PumpLiter Receipt NoReceipt Diesel StopPump PumpLiter TurnOff

Test#15: Activate 2.99 3.99 5.99 PayCash Super StartPump Activate 2.99 3.99 5.99 PayCredit Reject Approved Cancel PayCash -25.99 Regular Super StartPump Receipt NoReceipt Diesel StopPump PumpLiter StartPump Cancel TurnOff

Test#16: Activate 2.99 3.99 5.99 PayCredit Approved Diesel StartPump StopPump Activate 2.99 3.99 5.99 Reject Approved Cancel PayCash -25.99 Regular Super StartPump PumpLiter StopPump PumpLiter Diesel StopPump PumpLiter StartPump Cancel TurnOff

Test#17: Activate 2.99 3.99 6 PayCash 6 Diesel StartPump PumpLiter StopPump NoReceipt TurnOff

Test#18: Activate 2.99 5 5.99 PayCash 5 Regular StartPump StopPump NoReceipt TurnOff

Test#19: Activate 2.99 3.99 10 PayCash 10 Diesel StartPump PumpLiter PumpLiter Receipt TurnOff

Test#20: Activate 8 9 10 PayCredit TurnOff Approved Diesel StartPump PumpLiter StopPump NoReceipt TurnOff

Test#21: Activate 4 5 6 PayCredit Cancel Regular Approved Super Cancel PayCash 5 Cancel NoReceipt TurnOff

Test#22: Activate 4 5 6 PayCredit Approved Cancel PayCash -5 PayCash 10 PayCash -5 PayCash 5 Regular Super Diesel StartPump StartPump PumpLiter PumpLiter PumpLiter PumpLiter StopPump Receipt Receipt TurnOff

Test#23: Activate -2.99 -3.99 -4.99 Activate -2.99 -3.99 4.99 Activate -2.99 3.99 -4.99 Activate -2.99 3.99 4.99 Activate 2.99 -3.99 -4.99 Activate 2.99 -3.99 4.99 Activate 2.99 3.99 -4.99 Activate 2.99 3.99 4.99 Activate -2.99 -3.99 -4.99 Activate -2.99 -3.99 4.99 Activate -2.99 3.99 -4.99 Activate -2.99 3.99 4.99 Activate 2.99 -3.99 -4.99 Activate 2.99 -3.99 4.99 Activate 2.99 3.99 -4.99 Activate 2.99 3.99 4.99 PayCredit PayCredit Reject Reject Approved TurnOff

$$

7. Results of the Test Suite:

Test #1: Activate(2.99, 3.99, 5.99), PayCredit(), Approved(), Super(), StartPump(), StopPump(), NoReceipt(),PayCash(25.99), Super(), StartPump(), PumpLiter(), StopPump(), NoReceipt(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCredit()	T2	1	1
Approved()	T4	1	1
Super()	T8	1	1
StartPump()	T9	1	1
StopPump()	T11	1	1
NoReceipt()	T14	1	1
PayCash(25.99)	T6	1	1
Super()	T8	1	1
StartPump()	T9	1	1
PumpLiter()	T10	1	1
StopPump()	T11	1	1
NoReceipt()	T14	1	1
TurnOff()	T20	1	1

Test #2: Activate(2.99, 3.99, 5.99), PayCash(1), Diesel(), StartPump(), StopPump(), Receipt(), PayCredit(), Reject(), PayCash(1), Regular(), StartPump(), PumpLiter(), Receipt(), PayCash(25.99), Cancel(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCash(1)	T6	1	1
Diesel()	T15	1	1
StartPump()	T9	1	1

StopPump()	T11	1	1
Receipt()	T13	1	1
PayCredit()	T2	1	1
Reject()	T3	1	1
PayCash(1)	T6	1	1
Regular()	T7	1	1
StartPump()	T9	1	1
PumpLiter()	T12	1	1
Receipt()	T13	1	1
PayCash(25.99)	T6	1	1
Cancel()	T5	1	1
TurnOff()	T20	1	1

Test #3: Activate(2.99, 3.99, 5.99), PayCash(25.99), Regular(), Cancel(), PayCash(25.99), Super(), Cancel(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCash(25.99)	T6	1	1
Regular()	T7	1	1
Cancel()	T19	1	1
PayCash(25.99)	T6	1	1
Super()	T8	1	1
Cancel()	T19	1	1
TurnOff()	T20	1	1

Test #4: Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), PayCash(25.99), Cancel(), PayCredit(), Reject(), PayCredit(), Approved(), Diesel(), StartPump(), StopPump(), NoReceipt(), PayCredit(), Reject(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCredit()	T2	1	1
Reject()	T3	1	1
PayCash(25.99)	T6	1	1

Cancel()	T5	1	1
PayCredit()	T2	1	1
Reject()	T3	1	1
PayCredit()	T2	1	1
Approved()	T4	1	1
Diesel()	T15	1	1
StartPump()	T18	1	1
StopPump()	T16	1	1
NoReceipt()	T14	1	1
PayCredit()	T2	1	1
Reject()	T3	1	1
Turnoff()	T20	1	1

Test #5: Activate(2.99, 3.99, 5.99), PayCash(11.99), Cancel(), PayCash(11.99), Diesel(), Cancel(), PayCredit(), Approved(), Super(), Cancel(), PayCash(), Super(), StartPump(), PumpLiter(), PumpLiter(), PumpLiter(), NoReceipt(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCash(11.99)	T6	1	1
Cancel()	T5	1	1
PayCash(11.99)	T6	1	1
Diesel()	T15	1	1
Cancel()	T19	1	1
PayCredit()	T2	1	1
Approved()	T4	1	1
Super()	T8	1	1
Cancel()	T19	1	1
PayCash(6)	T6	1	1
Super()	T8	1	1
StartPump()	T9	1	1
PumpLiter()	T10	1	1
PumpLiter()	T10	1	1
PumpLiter()	T12	1	1
NoReceipt()	T14	1	1
TurnOff()	T20	1	1

Test #6: Activate(2.99, 3.99, 5.99), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
TurnOff()	T20	1	1

Test #7: Activate(2.99, 3.99, 5.99), PayCredit(), Approved(), Regular(), StartPump(), PumpLiter(), PumpLiter(), StopPump(), Receipt(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCredit()	T2	1	1
Approved()	T4	1	1
Regular()	T7	1	1
StartPump()	T18	1	1
PumpLiter()	T17	1	1
PumpLiter()	T17	1	1
StopPump()	T16	1	1
Receipt()	T13	1	1
TurnOff()	T20	1	1

Test #8: Activate(2.99, 3.99, 5.99), PayCredit(), Approved(), Cancel(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCredit()	T2	1	1
Approved()	T4	1	1
Cancel()	T5	1	1
TurnOff()	T20	1	1

Test #9: Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCredit()	T2	1	1
Reject()	T3	1	1
TurnOff()	T20	1	1

Test #10: Activate(2.99, 3.99, 5.99), Activate(2.99, 3.99, 5.99), Reject(), Approved(), Cancel(), PayCash(-25.99), Regular(), Super(), StartPump(), PumpLiter(), StopPump(), PumpLiter(), Receipt(), NoReceipt(), Diesel(), StopPump(), PumpLiter(), StartPump(), Cancel()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
Activate(2.99, 3.99, 5.99)	T1	0	0
Reject()	T3	0	0
Approved()	T4	0	0
Cancel()	T5	0	0
PayCash(-25.99)	T6	0	0
Regular()	T7	0	0
Super()	T8	0	0
StartPump()	T9	0	0
PumpLiter()	T10	0	0
StopPump()	T11	0	0
PumpLiter()	T12	0	0
Receipt()	T13	0	0
NoReceipt()	T14	0	0
Diesel()	T15	0	0
StopPump()	T16	0	0
PumpLiter()	T17	0	0
StartPump()	T18	0	0
Cancel()	T19	0	0

Test #11: Activate(2.99, 3.99, 5.99), PayCredit(), Approved(), Regular(), StartPump(), Activate(2.99, 3.99, 5.99), Cancel(), Reject(), Cancel(), PayCash(-25.99), Regular(), Super(), PumpLiter(), StartPump(), StopPump(), Receipt(), NoReceipt(), Diesel(), StartPump(), Cancel(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCredit()	T2	1	1
Approved()	T4	1	1
Regular()	T7	1	1
StartPump()	T18	1	1
Activate(2.99, 3.99, 5.99)	T1	0	0
Cancel()	T5	0	0
Reject()	T3	0	0
PayCash(-25.99)	T6	0	0
Regular()	T7	0	0
Super()	T8	0	0
PumpLiter()	T10	0	0
Startpump()	T9	0	0
StopPump()	T11	0	0
Receipt()	T13	0	0
NoReceipt()	T14	0	0
Diesel()	T15	0	0
StartPump()	T18	0	0
Cancel()	T19	0	0
TurnOff()	T20	0	0

Test #12: Activate(2.99, 3.99, 5.99), PayCredit(), Activate(2.99, 3.99, 5.99), PayCredit(), PayCash(-25.99), Cancel(), Regular(), Super(), StartPump(), PumpLiter(), StopPump(), PumpLiter(), Receipt(), NoReceipt(), Diesel(), StopPump(), PumpLiter(), StartPump(), Cancel(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCredit()	T2	1	1

Activate(2.99, 3.99, 5.99)	T1	0	0
PayCredit()	T2	0	0
PayCash(-25.99)	T6	0	0
Cancel()	T5	0	0
Regular()	T7	0	0
Super()	T8	0	0
StartPump()	T9	0	0
PumpLiter()	T10	0	0
StopPump()	T11	0	0
PumpLiter()	T12	0	0
Receipt()	T13	0	0
NoReceipt()	T14	0	0
Diesel()	T15	0	0
StopPump()	T16	0	0
PumpLiter()	T17	0	0
StartPump()	T18	0	0
Cancel()	T19	0	0
TurnOff()	T20	0	0

Test #13: Activate(2.99, 3.99, 5.99), PayCredit(), Approved(),Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), PayCash(-25.99), StartPump(), PumpLiter(), StopPump(), PumpLiter(), Receipt(), NoReceipt(), StopPump(), PumpLiter(), StartPump(), Cancel(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCredit()	T2	1	1
Approved()	T4	1	1
Activate(2.99, 3.99, 5.99)	T1	0	0
PayCredit()	T2	0	0
Reject()	T3	0	0
PayCash(-25.99)	T6	0	0
Startpump()	T9	0	0
PumpLiter()	T10	0	0
StopPump()	T11	0	0
PumpLiter()	T12	0	0
Receipt()	T13	0	0
NoReceipt()	T14	0	0
StopPump()	T16	0	0
PumpLiter()	T17	0	0
StartPump()	T18	0	0

Cancel()	T19	0	0
TurnOff()	T20	0	0

Test #14: Activate(2.99, 3.99, 5.99), PayCredit(), Approved(), Super(), Activate(2.99, 3.99, 5.99), Reject(), Approved(), Cancel(), PayCash(-25.99), Regular(), Super(), PumpLiter(), StopPump(), PumpLiter(), Receipt(), NoReceipt(), Diesel(), StopPump(), PumpLiter(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCredit()	T2	1	1
Approved()	T4	1	1
Super()	T8	1	1
Activate(2.99, 3.99, 5.99)	T1	0	0
Reject()	T3	0	0
Approved()	T4	0	0
Cancel()	T5	0	0
PayCash(-25.99)	T6	0	0
Regular()	T7	0	0
Super()	T8	0	0
PumpLiter()	T10	0	0
StopPump()	T11	0	0
PumpLiter()	T12	0	0
Receipt()	T13	0	0
NoReceipt()	T14	0	0
Diesel()	T15	0	0
StopPump()	T16	0	0
PumpLiter()	T17	0	0
TurnOff()	T20	0	0

Test #15: Activate(2.99, 3.99, 5.99), PayCash(20), Super(), StartPump(), Activate(2.99, 3.99, 5.99), PayCredit(), Reject(), Approved(), Cancel(), PayCash(-25.99), Regular(), Super(), StartPump(), Receipt(), NoReceipt(), Diesel(), StopPump(), PumpLiter(), StartPump(), Cancel(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCash(20)	T6	1	1
Super()	T8	1	1
StartPump()	T9	1	1
Activate(2.99, 3.99, 5.99)	T1	0	0
PayCredit()	T2	0	0
Reject()	T3	0	0
Approved()	T4	0	0
Cancel()	T5	0	0
PayCash(-25.99)	T6	0	0
Regular()	T7	0	0
Super()	T8	0	0
Startpump()	T9	0	0
Receipt()	T13	0	0
NoReceipt()	T14	0	0
Diesel()	T15	0	0
StopPump()	T16	0	0
PumpLiter()	T17	0	0
StartPump()	T18	0	0
Cancel()	T19	0	0
TurnOff()	T20	0	0

Test #16: Activate(2.99, 3.99, 5.99), PayCredit(), Approved(), Diesel(), StartPump(), StopPump(), Activate(2.99, 3.99, 5.99), Reject(), Approved(), Cancel(), PayCash(-25.99), Regular(), Super(), StartPump(), PumpLiter(), StopPump(), PumpLiter(), Diesel(), StopPump(), PumpLiter(), StartPump(), Cancel(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 5.99)	T1	1	1
PayCredit()	T2	1	1
Approved()	T4	1	1
Diesel()	T15	1	1
StartPump()	T18	1	1
StopPump()	T16	1	1
Activate(2.99, 3.99, 5.99)	T1	0	0
Reject()	T3	0	0
Approved()	T4	0	0
Cancel()	T5	0	0
PayCash(-25.99)	T6	0	0
Regular()	T7	0	0
Super()	T8	0	0
StartPump()	T9	0	0
PumpLiter()	T10	0	0
StopPump()	T11	0	0
PumpLiter()	T12	0	0
Diesel()	T15	0	0
StopPump()	T16	0	0
PumpLiter()	T17	0	0
StartPump()	T18	0	0
Cancel()	T19	0	0
TurnOff()	T20	0	0

Test #17: Activate(4, 5, 6), PayCash(6), Diesel(), StartPump(), PumpLiter(), StopPump(), NoReceipt(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 3.99, 6)	T1	1	1
PayCash(6)	T6	1	1
Diesel()	T15	1	1
StartPump()	T9	1	1
PumpLiter()	T10	1	1
StopPump()	T11	0	0
NoReceipt()	T14	1	1
TurnOff()	T20	1	1

Test #18: Activate(4, 5, 6), PayCash(5), Regular(), StartPump(), StopPump(), NoReceipt(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2.99, 5, 5.99)	T1	1	1
PayCash(5)	T6	1	1
Regular()	T7	1	1
StartPump()	T9	1	1
StopPump()	T11	1	1
NoReceipt()	T14	1	1
TurnOff()	T20	1	1

Test #19: Activate(2, 8, 10), PayCash(10), Diesel(), StartPump(), PumpLiter(), PumpLiter(), Receipt(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(2, 8, 10)	T1	1	1
PayCash(10)	T6	1	1
Diesel()	T15	1	1
StartPump()	T9	1	1
PumpLiter()	T10	1	1

PumpLiter()	T12	1	1
Receipt()	T13	1	1
TurnOff()	T20	1	1

Test #20: Activate(8, 9, 10), PayCredit(), TurnOff(), Approved(), Diesel(), StartPump(), PumpLiter(), StopPump(), NoReceipt(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(8, 9, 10)	T1	1	1
PayCredit()	T2	1	1
TurnOff()	T20	0	0
Approved()	T4	1	1
Diesel()	T15	1	1
StartPump()	T18	1	1
PumpLiter()	T17	1	1
StopPump()	T16	1	1
NoReceipt()	T14	1	1
TurnOff()	T20	1	1

Test #21: Activate(4, 5, 6), PayCredit(), Cancel(), Regular(), Approved(), Super(), Cancel(), PayCash(5), Cancel(), NoReceipt(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(4, 5, 6)	T1	1	1
PayCredit()	T2	1	1
Cancel()	T5	0	0
Regular()	T7	0	0
Approved()	T4	1	1
Super()	T8	1	1
Cancel()	T19	1	1
PayCash(5)	T6	1	1
Cancel()	T5	1	1
NoReceipt()	T14	0	0
TurnOff()	T20	1	1

Test #22: Activate(4, 5, 6), PayCredit(), Approved(), Cancel(), PayCash(-5), PayCash(10), PayCash(-5), PayCash(5), Regular(), Super(), Diesel(), StartPump(), StartPump(), PumpLiter(), PumpLiter(), PumpLiter(), PumpLiter(), StopPump(), Receipt(), Receipt(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output
Activate(4, 5, 6)	T1	1	1
PayCredit()	T2	1	1
Approved()	T4	1	1
Cancel()	T5	1	1
PayCash(-5)	T6	0	0
PayCash(10)	T6	1	1
PayCash(-5)	T6	0	0
PayCash(5)	T6	0	0
Regular()	T7	1	1
Super()	T8	0	0
Diesel()	T15	0	0
StartPump()	T9	1	1
StartPump()	T9	0	0
PumpLiter()	T10	1	1
PumpLiter()	T10	1	1
PumpLiter()	T10	1	1
PumpLiter()	T10	0	0
StopPump()	T11	0	0
Receipt()	T13	1	1
Receipt()	T13	0	0
TurnOff()	T20	1	1

Test #23: Activate(-2.99, -3.99, -4.99), Activate(-2.99, -3.99, 4.99), Activate(-2.99, 3.99, -4.99), Activate(-2.99, 3.99, 4.99), Activate(2.99, -3.99, -4.99), Activate(2.99, -3.99, 4.99), Activate(2.99, 3.99, -4.99), Activate(2.99, 3.99, 4.99), Activate(-2.99, -3.99, -4.99), Activate(-2.99, -3.99, 4.99), Activate(-2.99, 3.99, -4.99), Activate(-2.99, 3.99, 4.99), Activate(2.99, -3.99, -4.99), Activate(2.99, -3.99, 4.99), Activate(2.99, 3.99, -4.99), Activate(2.99, 3.99, 4.99), PayCredit(), PayCredit(), Reject(), Reject(), Approved(), TurnOff()

Test Result: PASS

Function Name	Transition	Expected Output	Actual Output

Activate(-2.99, -3.99, -4.99)	T1	0	0
Activate(-2.99, -3.99, 4.99)	T1	0	0
Activate(-2.99, 3.99, -4.99)	T1	0	0
Activate(-2.99, 3.99, 4.99)	T1	0	0
Activate(2.99, -3.99, -4.99)	T1	0	0
Activate(2.99, -3.99, 4.99)	T1	0	0
Activate(2.99, 3.99, -4.99)	T1	0	0
Activate(2.99, 3.99, 4.99)	T1	1	1
Activate(-2.99, -3.99, -4.99)	T1	0	0
Activate(-2.99, -3.99, 4.99)	T1	0	0
Activate(-2.99, 3.99, -4.99)	T1	0	0
Activate(-2.99, 3.99, 4.99)	T1	0	0
Activate(2.99, -3.99, -4.99)	T1	0	0
Activate(2.99, -3.99, 4.99)	T1	0	0
Activate(2.99, 3.99, -4.99)	T1	0	0
Activate(2.99, 3.99, 4.99)	T1	0	0
PayCredit()	T2	1	1
PayCredit()	T2	0	0
Reject()	T3	1	1
Reject()	T3	0	0
Approved()	T4	0	0
TurnOff()	T20	1	1

8. Conclusion:

Testing the GasPump class is the goal that I would do try to accomplish in this project. Testing if the GasPump class has any errors or bugs and reporting them is our role.

We are asked to test the source code of GasPump class with three different types of testing methods namely Model-based testing to show that 2-transition sequence testing is satisfied, Testing Default/Ghost transitions to show that the default transition testing is satisfied and the Multiple-condition Testing to show that multiple condition testing and also branch testing is satisfied.

Given the source code I tried to go-through it so as to understand for what I actually have to write the test cases, after a clear understanding I developed a test driver to check and find the flaws in the program because we might not find all the bugs in the program even by thorough understanding of the code until we test the code. The test driver was written in Java which checks the functionality of all the methods in the class. But the test driver alone was not sufficient, so I had to introduce a few **Testing-Oriented Methods** to check the intermediate values of the private variables of the class. I introduced testing methods to check the values of **total, price, Sprice, Rprice, Dprice, cash paid, current state and liters** and see if the expected value is same as the actual value of these variables.

Transition pair testing was easy enough due to prior experience, I had to identify all the pairs based on the incoming and outgoing transitions for each state and identify test cases covering these transitions; but when it came to testing default/ghost transitions understanding the testing method took some time. After understanding this testing method I developed 7 test cases each for a state to help in

confirming if any unnecessary or irrelevant function calls were made to any other functions other than its own outgoing transitions that gave up the integrity of the code. The next testing method was multiple-condition testing which shed light on the non-executable conditions in the program. I identified quite a few non-executable conditions and also explained the reason behind it. Testing the PumpLiter method was the toughest while doing multiple condition testing because there were conditions in the method which had complex conditions to check. I completed this testing method and the next part was testing all the test cases using the test driver.

The testing activity of the class can be partially or fully automated. It can be partially automated by creating a test driver such that it can read a single test case from a file and execute that test case completely rather than manually mentioning each activity in the test case one by one. While it can also be fully automated by creating a test driver such that it can read all the test cases from a file and execute all the test cases one after another and return the results of the tests on the console itself or into a separate file. By this the testing process by the tester is either partially automated or completely automated and the tester has to only compare the expected results and actual results and report the same to the programmer.

9. Source Code and Test Drivers:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Scanner;

public class GasPump {
	private float Rprice;
	private float Sprice;
	private float Dprice;
	private int w;
	private float price;
	private int L;
	private float total;
	private float cash;
	private int k;

	public static void main(String args[]) throws Exception {
		// Creating an object for the GasPump class
		GasPump g = new GasPump();
		System.out.println("\n\t\t\t\tCS 589 : FALL 2015");
		System.out.println("\t\t\t  TESTING PROJECT - GASPUMP CLASS");
		System.out.println("\t\t\tPrateek Poste ---- A20329636");
		@SuppressWarnings("resource")
		Scanner keyIn = new Scanner(System.in);
		System.out.print("Press The ENTER KEY To Continue");
		keyIn.nextLine();
```

```
            // TO execute program if user give different type of input
            while (true)
            {
                  System.out.print("Test Driver for the GASPUMP Class\n");
                  try {
                 BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
                        int choice;
                        String s;
                        float x, y, z, d;
                        //GasPump Menu
                        while (true) {
                              System.out.println("\n*-------  GASPUMP MENU  ------
-*");
                              System.out.println("\t   0.Activate");
                              System.out.println("\t   1.PayCredit");
                              System.out.println("\t   2.Reject");
                              System.out.println("\t   3.Cancel");
                              System.out.println("\t   4.Approved");
                              System.out.println("\t   5.PayCash");
                              System.out.println("\t   6.Regular");
                              System.out.println("\t   7.Super");
                              System.out.println("\t   8.Diesel");
                              System.out.println("\t   9.StartPump");
                              System.out.println("\t  10.PumpLiter");
                              System.out.println("\t  11.StopPump");
                              System.out.println("\t  12.NoReceipt");
                              System.out.println("\t  13.Receipt");
                              System.out.println("\t  14.TurnOff");
                              System.out.println("\n *--- TESTING ORIENTED METHOD
---*");
                              System.out.println("\n\t  15.SHOW PRICE");
                              System.out.println("\t  16.TOTAL");
                              System.out.println("\t  17.CASH PAID");
                              System.out.println("\t  18.NUMBER OF LITERS
PUMPED");
                              System.out.println("\t  19.SHOW STATE");
                              System.out.println("\t  20.QUIT GASPUMP DRIVER");
                              System.out.println("\n Enter your Choice:");
                              s = br.readLine();

                              choice = Integer.parseInt(s);
```

```
                        switch (choice) { //To execute the user entered
choice
                        case 0:
                              System.out.print("\nEnter value of Regular
Price: ");
                              x = Float.parseFloat(br.readLine());
                              System.out.print("\nEnter value of Super Price:
");
                              y = Float.parseFloat(br.readLine());
                              System.out.print("\nEnter value of Diesel
Price: ");
                              d = Float.parseFloat(br.readLine());
                              //calling the Activate function by using the
class object
                              System.out.println("Value returned by method
is: " + g.Activate(x, y, d));
                              break;
                        case 1:
                              //calling the PayCredit function by using the
class object
                              System.out.println("Value returned by method
is: " + g.PayCredit());
                              break;
                        case 2:
                              //calling the Reject function by using the
class object
                              System.out.println("Value returned by method
is: " + g.Reject());
                              break;
                        case 3:
                               //calling the Cancel function by using the
class object
                              System.out.print("\nValue returned by method
is: " + g.Cancel());
                              break;
                        case 4:
                              //calling the Approved function by using the
class object
                              System.out.println("Value returned by method
is: " + g.Approved());
                              break;
                        case 5:
```

```
                                    System.out.print("\nEnter Amount you want to
pay?:");
                                    z = Float.parseFloat(br.readLine());
                                    //calling the PayCash function by using the
class object
                                    System.out.println("Value returned by method
is: " + g.PayCash(z));
                                    break;
                               case 6:
                                    //calling the Regular function by using the
class object
                                    System.out.println("Value returned by method
is: "+ g.Regular());
                                    break;
                               case 7:
                                     //calling the Super function by using the
class object
                                    System.out.println("Value returned by method
is: "+ g.Super());
                                    break;
                               case 8:
                                    //calling the Diesel function by using the
class object
                                    System.out.println("Value returned by method
is: " + g.Diesel());
                                    break;
                               case 9:
                                    //calling the StartPump function by using the
class object
                                    System.out.println("Value returned by method
is: " + g.StartPump());
                                    break;
                               case 10:
                                    //calling the PumpLiter function by using the
class object
                                    System.out.println("Value returned by method
is: " + g.PumpLiter());
                                    break;
                               case 11:
                                    //calling the StopPump function by using the
class object
                                    System.out.println("Value returned by method
```

```
is: "+ g.StopPump());
                                    break;
                               case 12:
                                    //calling the NoReceipt function by using the
class object
                                    System.out.println("Value returned by method
is: "+ g.NoReceipt());
                                    break;
                               case 13:
                                    //calling the Receipt function by using the
class object
                                    System.out.println("Value returned by method
is: " + g.Receipt());
                                    break;
                               case 14:
                                    //calling the TurnOff function by using the
class object
                                    int value =g.TurnOff();
                                    System.out.println("Value returned by method
is: "+ value);
                                    // We check the return value of the class
                                    if(value == 1)
                                    {
                                    //and if it is 1 we refresh all the variables
in the class.
                                    //This is done by creating a new object for the
class
                                    GasPump a1 = new GasPump();
                                    g = a1;
                                    }
                                    break;
                               case 15:
                                    //Testing Oriented Method to show value of
price
                                    System.out.println("The Price is: " +
g.show_price());
                                    break;
                               case 16:
                                    //Testing Oriented Method to show value of
total
                                    System.out.print("\nThe Total is: " +
g.show_total());
```

```
                                    break;
                              case 17:
                                    //Testing Oriented Method to show value of cash
                                    System.out.print("\nThe Cash Paid is: "+
g.show_cash());
                                    break;
                              case 18:
                                    //Testing Oriented Method to show value of
Liters
                                    System.out.print("\nNumber of Liters: " +
g.show_Liters());
                                    break;
                              case 19:
                                    //Testing Oriented Method to show value of
Liters
                                    System.out.print("\nCurrent State: "+
g.show_state());
                                    break;
                              case 20:
                                    //Exit method
                                    System.exit(0);
                              default:
                                    //Default Case
                                    System.out.println("Enter Valid Input");
                                    break;
                              }
                              System.out.println("\nPress Enter to continue");
                              keyIn.nextLine();
                        }

                  } catch (Exception e)
                  {
                        System.out.println("Invalid Type:" + e);
                  }
            }
      }
      // testing oriented method
      public float show_price()
      {
            return price;
      }
      // testing oriented method
```

```
	public float show_total()
	{
		return total;
	}
	// testing oriented method
	public float show_cash()
	{
		return cash;
	}
	// testing oriented method
	public int show_Liters()
	{
		return L;
	}
	// testing oriented method
	public int show_state()
	{
		return k;
	}
	public GasPump() {
		Rprice = 0;
		Sprice = 0;
		Dprice = 0;
		w = 0;
		price = 0;
		L = 0;
		total = 0;
		cash = 0;
		k = -1;
	}

	public final int Activate(float a, float b, float d) {
		if ((k == -1) && (a > 0) && (b > 0) && (d > 0)) {
			k = 0;
			Rprice = a;
			Sprice = b;
			Dprice = d;
			System.out.print("GAS PUMP IS ON");
			System.out.print("\n");
			return 1;
		} else {
			return 0;
```

```
		}

	}

	public final int PayCredit() {
		if (k == 0) {
			k = 2;
			System.out.print("CHECKING CREDIT CARD.");
			System.out.print("\n");
			return 1;
		} else {
			return 0;
		}
	}

	public final int Reject() {
		if (k == 2) {
			k = 0;
			System.out.print("CREDIT CARD IS REJECTED.");
			System.out.print("\n");
			return 1;
		} else {
			return 0;
		}
	}

	public final int Cancel() {
		if ((k == 3) || (k == 4)) {
			k = 0;
			System.out.print("TRANSACTION IS CANCELLED.");
			System.out.print("\n");
			if (w == 0) {
				System.out.print("$");
				System.out.print(cash);
				System.out.print(" OF CASH IS RETURNED");
				System.out.print("\n");
			}
			return 1;
		} else {
			return 0;
		}
	}
```

```
	public final int Approved() {
		if (k == 2) {
			k = 3;
			w = 1;
			System.out.print("CREDIT CARD APPROVED.");
			System.out.print("\n");
			System.out.print("SELECT TYPE OF GASOLINE:");
			System.out.print("\n");
			System.out.print("a. REGULAR");
			System.out.print("\n");
			System.out.print("b. SUPER");
			System.out.print("\n");
			System.out.print("c. DIESEL");
			System.out.print("\n");
			return 1;
		} else {
			return 0;
		}
	}

	public final int PayCash(float c) {
		if ((k == 0) && (c > 0)) {
			k = 3;
			w = 0;
			cash = c;
			System.out.print("SELECT TYPE OF GASOLINE:");
			System.out.print("\n");
			System.out.print("a. REGULAR");
			System.out.print("\n");
			System.out.print("b. SUPER");
			System.out.print("\n");
			System.out.print("c. DIESEL");
			System.out.print("\n");
			return 1;
		} else {
			return 0;
		}
	}

	public final int Regular() {
		if (k == 3) {
```

```
                k = 4;
                System.out.print("REGULAR IS SELECTED.");
                System.out.print("\n");
                price = Rprice;
                return 1;
          } else {
                return 0;
          }
     }

     public final int Super() {
          if (k == 3) {
                k = 4;
                System.out.print("SUPER IS SELECTED.");
                System.out.print("\n");
                price = Sprice;
                return 1;
          } else {
                return 0;
          }
     }

     public final int Diesel() {
          if (k == 3) {
                k = 4;
                System.out.print("DIESEL IS SELECTED.");
                System.out.print("\n");
                price = Dprice;
                return 1;
          } else {
                return 0;
          }
     }

     public final int StartPump() {
          if (k == 4) {
                k = 5;
                L = 0;
                total = 0;
                System.out.print("PUMP IS READY TO DISPOSE ");
                System.out.print("\n");
                System.out.print("# OF LITERS PUMPED: ");
```

```
                System.out.print(L);
                System.out.print("\n");
                System.out.print("TOTAL CHARGE: $");
                System.out.print(total);
                System.out.print("\n");
                return 1;
          } else {
                return 0;
          }
     }

     public final int PumpLiter() {
          if (k == 5) {
                if ((w == 1) || ((cash >= price * (L + 1)) && (w == 0))) {
                     L = L + 1;
                     total = L * price;
                     System.out.print("# OF LITERS PUMPED: ");
                     System.out.print(L);
                     System.out.print("\n");
                     System.out.print("TOTAL CHARGE: $");
                     System.out.print(total);
                     System.out.print("\n");
                     System.out.print("CONTINUE PUMPING");
                     System.out.print("\n");
                     return 1;
                } else if ((w == 0) && (cash < price * (L + 1))) {
                     k = 6;
                     System.out.print("PUMP STOPPED. NOT SUFFICIENT FUNDS. ");
                     System.out.print("\n");
                     System.out.print("# OF LITERS PUMPED: ");
                     System.out.print(L);
                     System.out.print("\n");
                     System.out.print("TOTAL CHARGE: $");
                     System.out.print(total);
                     System.out.print("\n");
                     if ((w == 0) && (total < cash)) {
                          System.out.print("$");
                          System.out.print(cash - total);
                          System.out.print(" OF CASH IS RETURNED");
                          System.out.print("\n");
                     }
                     System.out.print("DO YOU WANT A RECEIPT?");
```

```
                    System.out.print("\n");
                    return 1;
               }
               ;
          }
          ;
          return 0;
     }

     public final int StopPump() {
          if (k == 5) {
               k = 6;
               System.out.print("PUMP STOPPED. ");
               System.out.print("\n");
               System.out.print("# OF LITERS PUMPED: ");
               System.out.print(L);
               System.out.print("\n");
               System.out.print("TOTAL CHARGE: $");
               System.out.print(total);
               System.out.print("\n");
               if ((w == 0) && (total < cash)) {
                    System.out.print("$");
                    System.out.print(cash - total);
                    System.out.print(" OF CASH IS RETURNED");
                    System.out.print("\n");
               }
               System.out.print("DO YOU WANT A RECEIPT?");
               System.out.print("\n");
               return 1;
          } else {
               return 0;
          }
     }

     public final int NoReceipt() {
          if (k == 6) {
               k = 0;
               System.out.print("NO RECEIPT IS PRINTED ");
               System.out.print("\n");
               return 1;
          } else {
               return 0;
```

```
        }
    }

    public final int Receipt() {
        if (k == 6) {
            k = 0;
            System.out.print("RECEIPT IS PRINTED: ");
            System.out.print("\n");
            System.out.print("# OF LITERS PUMPED: ");
            System.out.print(L);
            System.out.print("\n");
            System.out.print("TOTAL CHARGE: $");
            System.out.print(total);
            System.out.print("\n");
            return 1;
        } else {
            return 0;
        }
    }

    public final int TurnOff() {
        if (k == 0) {
            k = -2;
            System.out.print("GAS PUMP IS TURNED OFF ");
            System.out.print("\n");
            return 1;
        } else {
            return 0;
        }
    }
    }
```

10. Steps to Execute the Test Driver:

1. Double click the STAProject.exe to run the test driver which prompts a window which has all the operations of the GasPump class. We enter every operation of our test case in the prompt and check it with our expected results.

2. User will be prompted with a menu where all the operations of the class are present and we enter our operations from the test case and check if the return value is 0 or 1. If it is 1 then the operation is executed, else the operation is not executed.

3. If the test driver is not working properly, open command prompt by clicking "windows button + R" and then enter cmd in the text field. Navigate to the folder location where the test java file is present then re-compile the .java file and then run it using the java command.

   a. For compiling: javac GasPump.java ☾ Class file is created

   b. To run: java Gaspump

4. We can also execute the test driver from the .jar file provided. Open command prompt by clicking "windows button + R" and then enter cmd in the text field. Navigate to the folder location where the test java file is present then in the command prompt enter:

   a. java –jar STAProject.jar