CS 589
SOFTWARE TESTING & ANALYSIS
PROJECT REPORT
Illinois Institute of Technology

Submitted by-
Devansh Goel
A20490554

December 5, Fall 2022

Table of Contents

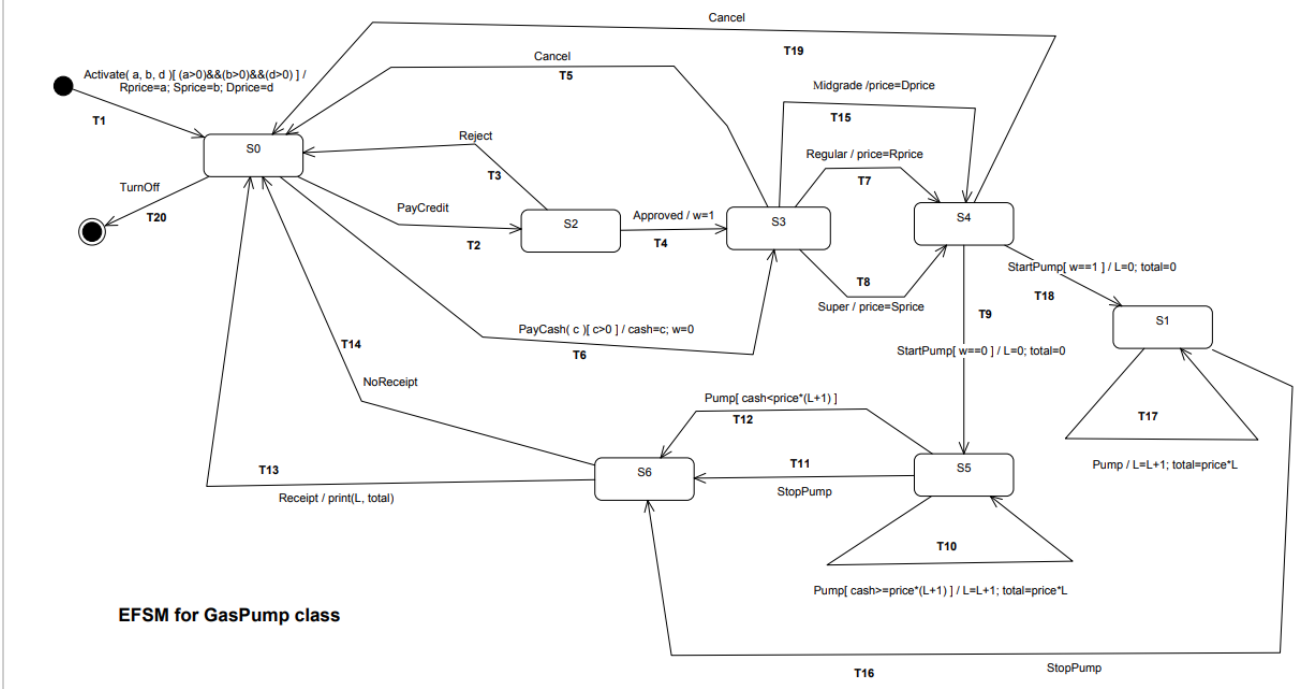
Introduction

The goal of this project is to test GasPump class that exhibits state behavior specified by the EFSM model. The GasPump class is a state-based class that is used to control a simple gas pump. Users can pay by cash or with a credit card. The gas pump disposes three types of gasoline: Regular, Super, and Midgrade. The price of each type of gasoline is provided when the gas pump is activated.

In this project the goal is to test the provided implementation (source code) of the GasPump class. The following testing methods are used:

1) Model-Based Testing
2) Default Transition Testing
3) Multiple Condition Testing

EFSM Model Used



EFSM model consists of states and their respective transitions. In the above model there are 8 states:-

States	**Operations**
1. Start 2. S0 3. S1 4. S2 5. S3 6. S4 7. S5 8. S6	GasPump() int Activate (float a, float b, float d) int PayCredit() int Reject() int Cancel() int Approved() int PayCash(float c) int Regular() int Super() int Midgrade() int StartPump() int Pump() int StopPump() int NoReceipt() int Receipt() int TurnOff()

Model-Based Testing of the GasPump class

Used the provided EFSM model to test the GasPump class. Designed test cases for the GasPump class so that all 2-transition sequences testing criterion (all transition-pairs) is satisfied based on the provided EFSM, i.e., all 2- transition sequences are executed during testing. Pairs are shown as (incoming, outgoing).

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

Transition Pairs
(T1, T2), (T1, T6), (T1, T20) (T3, T2), (T3, T6), (T3, T20) (T5, T2), (T5, T6), (T5, T20) (T13, T2), (T13, T6), (T13, T20) (T14, T2), (T14, T6), (T14, T20) (T19, T2), (T19, T6), (T19, T20)
(T17, T16), (T17, T17) (T18, T16), (T18, T17)
(T2, T3), (T2, T4)
(T4, T5), (T4, T7), (T4, T8), (T4, T15) (T6, T5), (T6, T7), (T6, T8), (T6, T15)
(T7, T9), (T7, T18), (T7, T19) (T8, T9), (T8, T18), (T8, T19) (T15, T9), (T15, T18), (T15, T19)
(T9, T10), (T9, T11), (T9, T12) (T10, T10), (T10, T11), (T10, T12)
(T11, T13), (T11, T14) (T12, T13), (T12, T14) (T16, T13), (T16, T14)

There are in total of 53 pairs.

Test cases

Test #	Test Case	Transitions	Pairs covered
1	Activate (2,3,4), PayCredit(), Approved(), Super(), StartPump(), Pump(), Pump(), StopPump(), Reciept(), PayCash(4), Cancel(), TurnOff().	T1, T2, T4, T8, T18, T17, T17, T16, T13, T6, T5, T20	(T1,T2), (T2,T4), (T4,T8), (T8,T18), (T18,T17), (T17,T17), (T17,T16), (T16,T13), (T13, T6), (T6, T5), (T5, T20)
2	Activate(2,3,4), PayCash(10), Regular(), StartPump(), Pump(), Pump(), StopPump(), NoReceipt(), PayCredit(), Reject(), PayCredit(), Approved(), Midgrade(), StartPump(), StopPump(), NoReceipt(), TurnOff()	T1, T6, T7, T9, T10, T10, T11, T14, T2, T3, T2, T4, T15, T18, T16, T14, T20	(T1,T6), (T6,T7), (T7,T9), (T9,T10), (T10,T10), (T10,T11), (T11,T14), (T14,T2), (T2,T3), (T3,T2), (T2,T4), (T4,T15), (T15,T18), (T18,T16), (T16,T14), (T14, T20)
3	Activate (2,3,4), TurnOff()	T1, T20	(T1,T20)
4	Activate(2,3,4), PayCredit(), Reject(), PayCash(20), Cancel(), PayCredit(), Reject(), PayCredit(), Approved(), Midgrade(), StartPump(), StopPump(), NoReceipt(), PayCredit(), Reject(), TurnOff()	T1, T2, T3, T6, T5, T2, T3, T2, T4, T15, T18, T16, T14, T2, T3, T20	(T1,T2), (T2,T3), (T3,T6), (T6,T5), (T5,T2), (T2,T3), (T3,T2), (T2,T4), (T4,T15), (T15,T18), (T18,T16), (T16,T14), (T14,T2), (T2,T3), (T3,T20)
5	Activate (2,3,4), PayCash(10), Super(), Cancel(), PayCash(10), Midgrade(), StartPump(), StopPump(), Receipt(), PayCredit(), Approved(), Cancel(), PayCash(10), Cancel(), TurnOff()	T1, T6, T8, T19, T6, T15, T9, T11, T13, T2, T4, T5, T6, T5, T20	(T1,T6), (T6,T8), (T8,T19), (T19,T6), (T6,T15), (T15,T9), (T9,T11), (T11,T13), (T13,T2), (T2,T4), (T4,T5), (T5,T6), (T6,T5), (T5,T20)
6	Activate(2,3,4), PayCredit(), Approved(), Regular(), Cancel(), PayCredit(), Approved(), Regular(), StartPump(), StopPump(), NoReceipt(), PayCash(10), Cancel(), TurnOff()	T1, T2, T4, T7, T19, T2, T4, T7, T18, T16, T14, T6, T5, T20	(T1,T2), (T2,T4), (T4,T7), (T7,T19), (T19,T2), (T2,T4), (T4,T7), (T7,T18), (T18,T16), (T16,T14), (T14,T6), (T6,T5), (T5,T20)
7	Activate(2,3,4), PayCash(2), Super(), StartPump(), Pump(), Receipt(), TurnOff()	T1, T6, T8, T9, T12, T13, T20	(T1,T6), (T6,T8), (T8,T9), (T9,T12), (T12,T13), (T13,T20)
8	Activate(2,3,4), PayCredit(), Approved(), Midgrade(), Cancel(), PayCash(5), Super(), StartPump(), Pump(), Pump(), NoReceipt(), PayCash(5), Midgrade(), Cancel(), TurnOff()	T1, T2, T4, T15, T19, T6, T8, T9, T10, T12, T14, T6, T15, T19, T20	(T1,T2), (T2,T4), (T4,T15), (T15,T19), (T19,T6), (T6,T8), (T8,T9), (T9,T10), (T10,T12), (T12,T14), (T14,T6), (T6,T15), (T15,T19), (T19,T20)

Testing Default Transitions (Ghost)

Default transitions are also known as Ghost transitions. These transitions have no effect on the state of the system and no system variable is affected as well. Default transitions and the test cases are shown below.

States	**Default Transitions**
Start	Activate (float a, float b, float d)[(a<=0)||(b<=0)||(d<=0)] PayCredit() Reject() Cancel() Approved() PayCash(float c) Regular() Super() Midgrade() StartPump() Pump() StopPump() NoReceipt() Receipt() TurnOff()
S0	Activate (float a, float b, float d) Reject() Cancel() Approved() PayCash(float c)[c<=0] Regular() Super() Midgrade() StartPump() Pump() StopPump() NoReceipt() Receipt()
S1	Activate (float a, float b, float d) PayCredit() Reject() Cancel() Approved() PayCash(float c) Regular() Super() Midgrade() StartPump() NoReceipt() Receipt()

	TurnOff()
S2	Activate (float a, float b, float d) PayCredit() Cancel() PayCash(float c) Regular() Super() Midgrade() StartPump() Pump() StopPump() NoReceipt() Receipt() TurnOff()
S3	Activate (float a, float b, float d) PayCredit() Reject() Approved() PayCash(float c) StartPump() Pump() StopPump() NoReceipt() Receipt() TurnOff()
S4	Activate (float a, float b, float d) PayCredit() Reject() Approved() PayCash(float c) Regular() Super() Midgrade() Pump() StopPump() NoReceipt() Receipt() TurnOff()
S5	Activate (float a, float b, float d) PayCredit() Reject() Cancel() Approved() PayCash(float c)

	Regular() Super() Midgrade() StartPump() NoReceipt() Receipt() TurnOff()
S6	Activate (float a, float b, float d) PayCredit() Reject() Cancel() Approved() PayCash(float c) Regular() Super() Midgrade() StartPump() Pump() StopPump() TurnOff()

Test Cases

State	Test #	Test Case
Start	9	Activate(-2, 3, 4), Activate(2, -3, 4), Activate(2, 3, -4), Activate(0, 3, 4), Activate(2, 0, 4), Activate(2, 3, 0), PayCredit(), Reject(), Cancel(), Approved(), PayCash(10), Regular(), Super(), Midgrade(), StartPump(), Pump(), StopPump(), NoReceipt(), Receipt(), TurnOff(), ***Activate(1, 2, 3), TurnOff()***
S0	10	***Activate(2, 3, 4),*** Activate(2,3,4), Reject(), Cancel(), Approved(), PayCash(-10), PayCash(0), Regular(), Super(), Midgrade(), StartPump(), Pump(), StopPump(), NoReceipt(), Receipt(), ***TurnOff()***
S1	11	***Activate(2,3,4), PayCredit(), Approved(), Super(), StartPump(),*** Activate(2,3,4), PayCredit(), Reject(), Cancel(), Approved(), PayCash(10), Regular(), Super(), Midgrade(), StartPump(), NoReceipt(), Receipt(), TurnOff(), ***StopPump(), Receipt(), TurnOff()***
S2	12	***Activate(2,3,4), PayCredit(),*** Activate(2,3,4), PayCredit(), Cancel(), PayCash(10), Regular(), Super(), Midgrade(), StartPump(), Pump(), StopPump(), NoReceipt(), Receipt(), TurnOff(), ***Reject(), TurnOff()***
S3	13	***Activate(2,3,4), PayCredit(), Approved****(),* Activate(2,3,4), PayCredit(), Reject(), Approved(), PayCash(10), StartPump(), Pump(), StopPump(), NoReceipt(), Receipt(), TurnOff(), ***Cancel(), TurnOff()***

S4	14	***Activate(2,3,4), PayCredit(), Approved(), Regular(),*** Activate(2,3,4), PayCredit(), Reject(), Approved(), PayCash(10), Regular(), Super(), Midgrade(), Pump(), StopPump(), NoReceipt(), Receipt(), TurnOff(), ***Cancel(), TurnOff()***
S5	15	***Activate(2,3,4), PayCash(10), Super(), StartPump(),*** Activate(2,3,4), PayCredit(), Reject(), Cancel(), Approved(), PayCash(10), Regular(), Super(), Midgrade(), StartPump(), NoReceipt(), Receipt(), TurnOff(), ***StopPump(), Receipt(), TurnOff()***
S6	16	***Activate(2,3,4), PayCash(10), Super(), StartPump(), StopPump(),*** Activate(2,3,4), PayCredit(), Reject(), Cancel(), Approved(), PayCash(10), Regular(), Super(), Midgrade(), StartPump(), Pump(), StopPump(), TurnOff(), ***NoReceipt(), TurnOff()***

Methods in **bold** are used to reach a particular state and then turn the GasPump off.

Multiple Condition testing

Considering predicate for Activate()

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

(k == -1)	**(a > 0)**	**(b > 0)**	**(d > 0)**	**Test Case Covered**
T	T	T	T	Test #17
T	T	T	F	Test #17
T	T	F	T	Test #17
T	T	F	F	Test #17
T	F	T	T	Test #17
T	F	T	F	Test #17
T	F	F	T	Test #17
T	F	F	F	Test #17
F	T	T	T	Test #17
F	T	T	F	Test #17
F	T	F	T	Test #17
F	T	F	F	Test #17
F	F	T	T	Test #17
F	F	T	F	Test #17
F	F	F	T	Test #17
F	F	F	F	Test #17

PayCredit()

if (k == 0)

(k == 0)	**Test Case Covered**
T	Test #17
F	Test #17

Reject()

if (k == 2)

(k == 2)	**Test Case Covered**
T	Test #17
F	Test #17

Cancel()

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

(k == 3)	**(k == 4)**	**Test Case Covered**
T	T	Non-Executable
T	F	Test #17
F	T	Test #17
F	F	Test #17

Approved()

if (k == 2)

(k == 2)	**Test Case Covered**
T	Test #17
F	Test #17

if (w == 0)

(w == 0)	Test Case Covered
T	Test #17
F	Test #17

PayCash()

if (k == 0)

(k == 0)	Test Case Covered
T	Test #17
F	Test #17

Regular()

if (k == 3)

(k == 3)	Test Case Covered
T	Test #17
F	Test #17

Midgrade()

if (k == 3)

(k == 3)	Test Case Covered
T	Test #17
F	Test #17

Super()

if (k == 3)

(k == 3)	Test Case Covered
T	Test #17
F	Test #17

StartPump()

if (k == 4)

(k == 4)	Test Case Covered
T	Test #18
F	Test #18

Pump()

if (k == 5)

(k == 5)	Test Case Covered
T	Test #18
F	Test #18

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

(w == 1)	(cash > price * (L + 1))	(w == 0)	Test Case Covered
T	T	T	Non-Executable
T	T	F	Non-Executable
T	F	T	Non-Executable
T	F	F	Test #21
F	T	T	Test #18

F	T	F	Non-Executable
F	F	T	Test #18
F	F	F	Non-Executable

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

(w == 0)	**(cash < price * (L + 1))**	**Test Case Covered**
T	T	Test #18
T	F	Test #22
F	T	Non-Executable
F	F	Non-Executable

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

(w == 0)	**(total < cash)**	**Test Case Covered**
T	T	Test #18
T	F	Test #18
F	T	Non-Executable
F	F	Non-Executable

StopPump()

if (k == 5)

(k == 5)	**Test Case Covered**
T	Test #19
F	Test #18

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

(w == 0)	**(total < cash)**	**Test Case Covered**
T	T	Test #19
T	F	Test #20
F	T	Non-Executable
F	F	Test #21

NoReceipt()

if (k == 6)

(k == 6)	**Test Case Covered**
T	Test #18
F	Test #17

Receipt()

if (k == 6)

(k == 6)	**Test Case Covered**

TurnOff()

if (k == 0)

(k == 0)	Test Case Covered
T	Test #17
F	Test #17

T	Test #18
F	Test #17

Test Cases

Test #	Test Case
17	Activate(-2, -3, -4), Activate(-2, -3, 4), Activate(-2, 3, -4), Activate(2, -3, -4), Activate(-2, 3, 4), Activate(2, -3, 4), Activate(2, 3, -4), Activate(2, 3, 4), Activate(-2, -3, -4), Activate(-2, -3, 4), Activate(-2, 3, -4), Activate(2, -3, -4), Activate(-2, 3, 4), Activate(2, -3, 4), Activate(2, 3, -4), Activate(2, 3, 4), PayCredit(), PayCredit(), TurnOff(), Reject(), Reject(), Approved(), Reject(), Approved(), Approved(),PayCredit(), PayCash(20), Cancel(), Regular(), Super(), Midgrade(), Approved(), Regular(), Cancel(), PayCash(20), Super(), Cancel(), PayCash(20), Midgrade(), Cancel(), PayCash(20), Cancel(), Receipt(), NoReceipt(), TurnOff()
18	Activate(4.5, 5.5, 6.5), StartPump(), Pump(), PayCredit(), Approved(), Cancel(), PayCash(10.5), Regular(), StartPump(), Pump(), Pump(), Pump(), StopPump(), Receipt(), PayCash(-10.5), Super(), StartPump(), Pump(), NoReceipt(), TurnOff()
19	Activate(4.5, 5.5, 6.5), PayCash(10.5), Regular(), StartPump(), Pump(), Pump(), StopPump(), Receipt(), Super(), StartPump(), Midgrade(), TurnOff()
20	Activate(4.5, 5.5, 6.5), PayCash(-20), Regular(), StartPump(), StopPump(),

	Receipt(), Super(), StartPump(), Regular(), TurnOff()
21	Activate(4.5, 5.5, 6.5), PayCredit(), Approved(), Regular(), StartPump(), Pump(), StopPump(), NoReceipt(), Super(), Regular(), Midgrade(), Pump(), TurnOff()
22	Activate(4.5, 5.5, 6.5), PayCash(5.5), Super(), StartPump(), Pump(), StopPump(), NoReceipt(), TurnOff()

Non-Executable Test cases and Explanation

Some Non-Executable test cases are observed in Multiple conditioning. These are mentioned below.

Function: Cancel()

Case	**Function**	**Conditions**			**Result**
	Cancel()	**(k == 3)**	**(k == 4)**		
Case#1		T	T		Non-Executable
	Pump()				

Reason:-

Case#1: Value of k cannot be 3 and 4 simultaneously.

Function: Pump()

Case	**Function**	**Conditions**			**Result**
	Pump()				
		(w == 1)	**(cash > price * (L + 1))**	**(w == 0)**	
Case#2		T	T	T	Non-Executable
Case#3		T	T	F	Non-Executable
Case#4		T	F	T	Non-Executable
Case#5		F	T	F	Non-Executable
Case#6		F	F	F	Non-Executable
		(w == 0)	**(cash < price * (L + 1))**		
Case#7		F	T		Non-Executable
Case#8		F	F		Non-Executable
		(w == 0)	**(total < cash)**		
Case#9		F	T		Non-Executable
Case#10		F	F		Non-Executable

Reason:-

Case#2: Value of w cannot be 1 and 0 simultaneously.

Case#3: w=1 indicates that payment is done by credit card and cash>price*(L+1) is true indicates that cash has been accepted. Hence, both conditions cannot happen simultaneously.

Case#4: Value of w cannot be 1 and 0 simultaneously.

Case#5: The default value of w is 0. In this case w = 0 and w = 1 both are false. Not possible.

Case#6: The default value of w is 0. In this case w = 0 and w = 1 both are false. Not possible.

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

Case#8: : In this case, w=0 is false that means payment is done by card i.e. w=1 so it will execute the else condition.

Case#9: : w=0 is false means payment is done through card so the function will never reach at that point to execute.

Case#10: w=0 is false that means payment is done by card so the function will never reach at that point to execute.

Function: StopPump()

Case	**Function**	**Conditions**			**Result**
	StopPump()				
		(w == 0)	**(total < cash)**		
Case#11		F	T		Non-Executable

Case#11: w=0 is false that means payment is done by card so the function will never reach at that point to execute.

Test Suite

Test#1: Activate 2 3 4 PayCredit Approved Super StartPump Pump Pump StopPump Receipt PayCash 4 Cancel TurnOff

Test#2: Activate 2 3 4 PayCash 10 Regular StartPump Pump Pump StopPump NoReceipt PayCredit Reject PayCredit Approved Midgrade StartPump StopPump NoReceipt TurnOff

Test#3: Activate 2 3 4 TurnOff

Test#4: Activate 2 3 4 PayCredit Reject PayCash 20 Cancel PayCredit Reject PayCredit Approved Midgrade StartPump StopPump NoReceipt PayCredit Reject TurnOff

Test#5: Activate 2 3 4 PayCash 10 Super Cancel PayCash 10 Midgrade StartPump StopPump Receipt PayCredit Approved Cancel PayCash 10 Cancel TurnOff

Test#6: Activate 2 3 4 PayCredit Approved Regular Cancel PayCredit Approved Regular StartPump StopPump NoReceipt PayCash 10 Cancel TurnOff

Test#7: Activate 2 3 4 PayCash 2 Super StartPump Pump Receipt TurnOff

Test#8: Activate 2 3 4 PayCredit Approved Midgrade Cancel PayCash 5 Super StartPump Pump Pump NoReceipt PayCash 5 Midgrade Cancel TurnOff

Test#9: Activate -2 3 4 Activate 2 -3 4 Activate 2 3 -4 Activate 0 3 4 Activate 2 0 4 Activate 2 3 0 PayCredit Reject Cancel Approved PayCash 10 Regular Super Midgrade StartPump Pump StopPump NoReceipt Receipt TurnOff Activate 1 2 3 TurnOff

Test#10: Activate 2 3 4 Activate 2 3 4 Reject Cancel Approved PayCash -10 PayCash 0 Regular Super Midgrade StartPump Pump StopPump NoReceipt Receipt TurnOff

Test#11: Activate 2 3 4 PayCredit Approved Super StartPump Activate 2 3 4 PayCredit Reject Cancel Approved PayCash 10 Regular Super Midgrade StartPump NoReceipt Receipt TurnOff StopPump Receipt TurnOff

Test#12: Activate 2 3 4 PayCredit Activate 2 3 4 PayCredit Cancel PayCash 10 Regular Super Midgrade StartPump Pump StopPump NoReceipt Receipt TurnOff Reject TurnOff

Test#13: Activate 2 3 4 PayCredit Approved Activate 2 3 4 PayCredit Reject Approved PayCash 10 StartPump Pump StopPump NoReceipt Receipt TurnOff Cancel TurnOff

Test#14: Activate 2 3 4 PayCredit Approved Regular Activate 2 3 4 PayCredit Reject Approved PayCash 10 Regular Super Midgrade Pump StopPump NoReceipt Receipt TurnOff Cancel TurnOff

Test#15: Activate 2 3 4 PayCash 10 Super StartPump Activate 2 3 4 PayCredit Reject Cancel Approved PayCash 10 Regular Super Midgrade StartPump NoReceipt Receipt TurnOff StopPump Receipt TurnOff

Test#16: Activate 2 3 4 PayCash 10 Super StartPump StopPump Activate 2 3 4 PayCredit Reject Cancel Approved PayCash 10 Regular Super Midgrade StartPump Pump StopPump TurnOff NoReceipt TurnOff

Test#17: Activate -2 -3 -4 Activate -2 -3 4 Activate -2 3 -4 Activate 2 -3 -4 Activate -2 3 4 Activate 2 -3 4 Activate 2 3 -4 Activate 2 3 4 Activate -2 -3 -4 Activate -2 -3 4 Activate -2 3 -4 Activate 2 -3 -4 Activate -2 3 4 Activate 2 -3 4 Activate 2 3 -4 Activate 2 3 4 PayCredit PayCredit TurnOff Reject Reject Approved Reject Approved Approved PayCredit PayCash 20 Cancel Regular Super Midgrade Approved Regular Cancel PayCash 20 Super Cancel PayCash 20 Midgrade Cancel PayCash 20 Cancel Receipt NoReceipt TurnOff

Test#18: Activate 4.5 5.5 6.5 StartPump Pump PayCredit Approved Cancel PayCash 10.5 Regular StartPump Pump Pump Pump StopPump Receipt PayCash -10.5 Super StartPump Pump NoReceipt TurnOff

Test#19: Activate 4.5 5.5 6.5 PayCash 10.5 Regular StartPump Pump Pump StopPump Receipt Super StartPump Midgrade TurnOff

Test#20: Activate 4.5 5.5 6.5 PayCash -20 Regular StartPump StopPump Receipt Super StartPump Regular TurnOff

Test#21: Activate 4.5 5.5 6.5 PayCredit Approved Regular StartPump Pump StopPump NoReceipt Super Regular Midgrade Pump TurnOff

Test#22: Activate 4.5 5.5 6.5 PayCash 5.5 Super StartPump Pump StopPump NoReceipt TurnOff

$$

Test Suite results

Test#1

Activate 2 3 4 PayCredit Approved Super StartPump Pump Pump StopPump Receipt PayCash 4 Cancel TurnOff

Method	Expected Output	Actual Output
Activate 2 3 4	1	1
PayCredit	1	1
Approved	1	1
Super	1	1
StartPump	1	1
Pump	1	1
Pump	1	1
StopPump	1	1
Receipt	1	1
PayCash 4	1	1
Cancel	1	1
TurnOff	1	1

Result: Pass

Test#2

Test#2: Activate 2 3 4 PayCash 10 Regular StartPump Pump Pump StopPump NoReceipt PayCredit Reject PayCredit Approved Midgrade StartPump StopPump NoReceipt TurnOff

Method	Expected Output	Actual Output
Activate 2 3 4	1	1
PayCash 10	1	1
Regular	1	1
StartPump	1	1
Pump	1	1
Pump	1	1
StopPump	1	1
NoReceipt	1	1
PayCredit	1	1
Reject	1	1
PayCredit	1	1
Approved	1	1
Midgrade	1	1
StartPump	1	1
StopPump	1	1
NoReceipt	1	1
TurnOff	1	1

Result: Pass

Test#3: Activate 2 3 4 TurnOff

Method	Expected Output	Actual Output
Activate 2 3 4	1	1
TurnOff	1	1

Result: Pass

Test#4: Activate 2 3 4 PayCredit Reject PayCash 20 Cancel PayCredit Reject PayCredit Approved Midgrade StartPump StopPump NoReceipt PayCredit Reject TurnOff

Method	Expected Output	Actual Output
Activate 2 3 4	1	1
PayCredit	1	1
Reject	1	1
PayCash 20	1	1
Cancel	1	1
PayCredit	1	1
Reject	1	1
PayCredit	1	1
Approved	1	1
Midgrade	1	1
StartPump	1	1
StopPump	1	1
NoReceipt	1	1
PayCredit	1	1
Reject	1	1
TurnOff	1	1

Result: Pass

Test#5: Activate 2 3 4 PayCash 10 Super Cancel PayCash 10 Midgrade StartPump StopPump Receipt PayCredit Approved Cancel PayCash 10 Cancel TurnOff

Method	Expected Output	Actual Output
Activate 2 3 4	1	1
PayCash 10	1	1
Super	1	1
Cancel	1	1
PayCash 10	1	1
Midgrade	1	1
StartPump	1	1

StopPump	1	1
Receipt	1	1
PayCredit	1	1
Approved	1	1
Cancel	1	1
PayCash 10	1	1
Cancel	1	1
TurnOff	1	1

Result: Pass

Test#6: Activate 2 3 4 PayCredit Approved Regular Cancel PayCredit Approved Regular StartPump StopPump NoReceipt PayCash 10 Cancel TurnOff

Method	**Expected Output**	**Actual Output**
Activate 2 3 4	1	1
PayCredit	1	1
Approved	1	1
Regular	1	1
Cancel	1	1
PayCredit	1	1
Approved	1	1
Regular	1	1
StartPump	1	1
StopPump	1	1
NoReceipt	1	1
PayCash 10	1	1
Cancel	1	1
TurnOff	1	1

Result: Pass

Test#7: Activate 2 3 4 PayCash 2 Super StartPump Pump Receipt TurnOff

Method	**Expected Output**	**Actual Output**
Activate 2 3 4	1	1
PayCash 2	1	1
Super	1	1
StartPump	1	1
Pump	1	1
Receipt	1	1
TurnOff	1	1

Test#8: Activate 2 3 4 PayCredit Approved Midgrade Cancel PayCash 5 Super StartPump Pump Pump NoReceipt PayCash 5 Midgrade Cancel TurnOff

Method	Expected Output	Actual Output
Activate 2 3 4	1	1
PayCredit	1	1
Approved	1	1
Midgrade	1	1
Cancel	1	1
PayCash 5	1	1
Super	1	1
StartPump	1	1
Pump	1	1
Pump	1	1
NoReceipt	1	1
PayCash 5	1	1
Midgrade	1	1
Cancel	1	1
TurnOff	1	1

Result: Pass

Test#9: Activate -2 3 4 Activate 2 -3 4 Activate 2 3 -4 Activate 0 3 4 Activate 2 0 4 Activate 2 3 0 PayCredit Reject Cancel Approved PayCash 10 Regular Super Midgrade StartPump Pump StopPump NoReceipt Receipt TurnOff Activate 1 2 3 TurnOff

Method	Expected Output	Actual Output
Activate -2 3 4	0	0
Activate 2 -3 4	0	0
Activate 2 3 -4	0	0
Activate 0 3 4	0	0
Activate 2 0 4	0	0
Activate 2 3 0	0	0
PayCredit	0	0
Reject	0	0
Cancel	0	0
Approved	0	0
PayCash 10	0	0
Regular	0	0
Super	0	0
Midgrade	0	0
StartPump	0	0
Pump	0	0
StopPump	0	0

NoReceipt	0	0
Receipt	0	0
TurnOff	0	0
Activate 1 2 3	1	1
TurnOff	1	1

Result: Pass

Test#10: Activate 2 3 4 Activate 2 3 4 Reject Cancel Approved PayCash -10 PayCash 0 Regular Super Midgrade StartPump Pump StopPump NoReceipt Receipt TurnOff

Function	**Expected Output**	**Actual Output**
Activate 2 3 4	1	1
Activate 2 3 4	0	0
Reject	0	0
Cancel	0	0
Approved	0	0
PayCash -10	0	1
PayCash 0	0	0
Regular	0	1
Super	0	0
Midgrade	0	0
StartPump	0	1
Pump	0	1
StopPump	0	0
NoReceipt	0	1
Receipt	0	0
TurnOff	1	1

Result: Fail

Reason: Predicates of PayCash() do not check the value of cash entered. Defect Detected

Test#11: Activate 2 3 4 PayCredit Approved Super StartPump Activate 2 3 4 PayCredit Reject Cancel Approved PayCash 10 Regular Super Midgrade StartPump NoReceipt Receipt TurnOff StopPump Receipt TurnOff

Test#11:		
Function	**Expected Output**	**Actual Output**
Activate 2 3 4	1	1
PayCredit	1	1
Approved	1	1
Super	1	1
StartPump	1	1

Activate 2 3 4	0	0
PayCredit	0	0
Reject	0	0
Cancel	0	0
Approved	0	0
PayCash 10	0	0
Regular	0	0
Super	0	0
Midgrade	0	0
StartPump	0	0
NoReceipt	0	0
Receipt	0	0
TurnOff	0	0
StopPump	1	1
Receipt	1	1
TurnOff	1	1

Result: Pass

Test#12: Activate 2 3 4 PayCredit Activate 2 3 4 PayCredit Cancel PayCash 10 Regular Super Midgrade StartPump Pump StopPump NoReceipt Receipt TurnOff Reject TurnOff

Test#12:		
Function	**Expected Output**	**Actual Output**
Activate 2 3 4	1	1
PayCredit	1	1
Activate 2 3 4	0	0
PayCredit	0	0
Cancel	0	0
PayCash 10	0	0
Regular	0	0
Super	0	0
Midgrade	0	0
StartPump	0	0
Pump	0	0
StopPump	0	0
NoReceipt	0	0
Receipt	0	0
TurnOff	0	0
Reject	1	1
TurnOff	1	1

Result: Pass

Test#13: Activate 2 3 4 PayCredit Approved Activate 2 3 4 PayCredit Reject Approved PayCash 10 StartPump Pump StopPump NoReceipt Receipt TurnOff Cancel TurnOff

Test#13:		
Function	**Expected Output**	**Actual Output**
Activate 2 3 4	1	1
PayCredit	1	1
Approved	1	1
Activate 2 3 4	0	0
PayCredit	0	0
Reject	0	0
Approved	0	0
PayCash 10	0	0
StartPump	0	0
Pump	0	0
StopPump	0	0
NoReceipt	0	0
Receipt	0	0
TurnOff	0	0
Cancel	1	1
TurnOff	1	1

Result: Pass

Test#14: Activate 2 3 4 PayCredit Approved Regular Activate 2 3 4 PayCredit Reject Approved PayCash 10 Regular Super Midgrade Pump StopPump NoReceipt Receipt TurnOff Cancel TurnOff

Test#14:		
Function	**Expected Output**	**Actual Output**
Activate 2 3 4	1	1
PayCredit	1	1
Approved	1	1
Regular	1	1
Activate 2 3 4	0	0
PayCredit	0	0
Reject	0	0
Approved	0	0
PayCash 10	0	0
Regular	0	0
Super	0	0
Midgrade	0	0
Pump	0	0
StopPump	0	0
NoReceipt	0	0
Receipt	0	0
TurnOff	0	0
Cancel	1	1

TurnOff	1	1

Result: Pass

Test#15: Activate 2 3 4 PayCash 10 Super StartPump Activate 2 3 4 PayCredit Reject Cancel Approved PayCash 10 Regular Super Midgrade StartPump NoReceipt Receipt TurnOff StopPump Receipt TurnOff

Test#15:		
Function	**Expected Output**	**Actual Output**
Activate 2 3 4	1	1
PayCash 10	1	1
Super	1	1
StartPump	1	1
Activate 2 3 4	0	0
PayCredit	0	0
Reject	0	0
Cancel	0	0
Approved	0	0
PayCash 10	0	0
Regular	0	0
Super	0	0
Midgrade	0	0
StartPump	0	0
NoReceipt	0	0
Receipt	0	0
TurnOff	0	0
StopPump	1	1
Receipt	1	1
TurnOff	1	1

Result: Pass

Test#16: Activate 2 3 4 PayCash 10 Super StartPump StopPump Activate 2 3 4 PayCredit Reject Cancel Approved PayCash 10 Regular Super Midgrade StartPump Pump StopPump TurnOff NoReceipt TurnOff

Test#16:		
Function	**Expected Output**	**Actual Output**
Activate 2 3 4	1	1
PayCash 10	1	1
Super	1	1
StartPump	1	1
StopPump	1	1
Activate 2 3 4	0	0
PayCredit	0	0

Reject	0	0
Cancel	0	0
Approved	0	0
PayCash 10	0	0
Regular	0	0
Super	0	0
Midgrade	0	0
StartPump	0	0
Pump	0	0
StopPump	0	0
TurnOff	0	0
NoReceipt	1	1
TurnOff	1	1

Result: Pass

Test#17: Activate -2 -3 -4 Activate -2 -3 4 Activate -2 3 -4 Activate 2 -3 -4 Activate -2 3 4 Activate 2 -3 4 Activate 2 3 -4 Activate 2 3 4 Activate -2 -3 -4 Activate -2 -3 4 Activate -2 3 -4 Activate 2 -3 -4 Activate -2 3 4 Activate 2 -3 4 Activate 2 3 -4 Activate 2 3 4 PayCredit PayCredit TurnOff Reject Reject Approved Reject Approved Approved PayCredit PayCash 20 Cancel Regular Super Midgrade Approved Regular Cancel PayCash 20 Super Cancel PayCash 20 Midgrade Cancel PayCash 20 Cancel Receipt NoReceipt TurnOff

Test#17:		
Function	**Expected Output**	**Actual Output**
Activate -2 -3 -4	0	0
Activate -2 -3 4	0	0
Activate -2 3 -4	0	0
Activate 2 -3 -4	0	0
Activate -2 3 4	0	0
Activate 2 -3 4	0	0
Activate 2 3 -4	0	0
Activate 2 3 4	1	1
Activate -2 -3 -4	0	0
Activate -2 -3 4	0	0
Activate -2 3 -4	0	0
Activate 2 -3 -4	0	0
Activate -2 3 4	0	0
Activate 2 -3 4	0	0
Activate 2 3 -4	0	0
Activate 2 3 4	0	0
PayCredit	1	1
PayCredit	0	0
TurnOff	0	0
Reject	1	1

Reject	0	0
Approved	0	0
Reject	0	0
Approved	0	0
Approved	0	0
PayCredit	1	1
PayCash 20	0	0
Cancel	0	0
Regular	0	0
Super	0	0
Midgrade	0	0
Approved	1	1
Regular	1	1
Cancel	1	1
PayCash 20	1	1
Super	1	1
Cancel	1	1
PayCash 20	1	1
Midgrade	1	1
Cancel	1	1
PayCash 20	1	1
Cancel	1	1
Receipt	0	0
NoReceipt	0	0
TurnOff	1	1

Result: Pass

Test#18: Activate 4.5 5.5 6.5 StartPump Pump PayCredit Approved Cancel PayCash 10.5 Regular StartPump Pump Pump Pump StopPump Receipt PayCash -10.5 Super StartPump Pump NoReceipt TurnOff

Test#18:		
Function	**Expected Output**	**Actual Output**
Activate 4.5 5.5 6.5	1	1
StartPump	0	0
Pump	0	0
PayCredit	1	1
Approved	1	1
Cancel	1	1
PayCash 10.5	1	1
Regular	1	1
StartPump	1	1
Pump	1	1
Pump	1	1

Pump	1	1
StopPump	0	0
Receipt	1	1
PayCash -10.5	0	1
Super	0	1
StartPump	0	1
Pump	0	1
NoReceipt	0	1
TurnOff	1	1

Result: Fail

Reason: The code accepts a negative value of cash but according to the given EFSM cash value should always be greater than 0. Hence this is a defect.

Test#19: Activate 4.5 5.5 6.5 PayCash 10.5 Regular StartPump Pump Pump StopPump Receipt Super StartPump Midgrade TurnOff

Test#19:		
Function	**Expected Output**	**Actual Output**
Activate 4.5 5.5 6.5	1	1
PayCash 10.5	1	1
Regular	1	1
StartPump	1	1
Pump	1	1
Pump	1	1
StopPump	1	1
Receipt	1	1
Super	0	0
StartPump	0	0
Midgrade	0	0
TurnOff	1	1

Result Pass

Test#20: Activate 4.5 5.5 6.5 PayCash -20 Regular StartPump StopPump Receipt Super StartPump Regular TurnOff

Test#20:		
Function	**Expected Output**	**Actual Output**
Activate 4.5 5.5 6.5	1	1
PayCash -20	0	1
Regular	0	1
StartPump	0	1
StopPump	0	1

Receipt	0	1
Super	0	0
StartPump	0	0
Regular	0	0
TurnOff	1	1

Result: Fail

Reason: The code accepts a negative value of cash but according to the given EFSM cash value should always be greater than 0. Hence this is a defect.

Test#21: Activate 4.5 5.5 6.5 PayCredit Approved Regular StartPump Pump StopPump NoReceipt Super Regular Midgrade Pump TurnOff

Test#21:		
Function	**Expected Output**	**Actual Output**
Activate 4.5 5.5 6.5	1	1
PayCredit	1	1
Approved	1	1
Regular	1	1
StartPump	1	1
Pump	1	1
StopPump	1	1
NoReceipt	1	1
Super	0	0
Regular	0	0
Midgrade	0	0
Pump	0	0
TurnOff	1	1

Result: Pass

Test#22: Activate 4.5 5.5 6.5 PayCash 5.5 Super StartPump Pump StopPump NoReceipt TurnOff

Test#22:		
Function	**Expected Output**	**Actual Output**
Activate 4.5 5.5 6.5	1	1
PayCash 5.5	1	1
Super	1	1
StartPump	1	1
Pump	1	0
StopPump	1	1
NoReceipt	1	1

TurnOff	1	1

Result: Fail

Reason: According to the EFSM the expected output is 1 as the condition of cash >= price * (L+1) is satisfied. But in the code the predicates for Pump method do not handle the case when cash == price * (L + 1). Hence a defect.

Failed Tests

Test #10

Function	**Expected Output**	**Actual Output**
Activate 2 3 4	1	1
Activate 2 3 4	0	0
Reject	0	0
Cancel	0	0
Approved	0	0
PayCash -10	0	1
PayCash 0	0	0
Regular	0	1
Super	0	0
Midgrade	0	0
StartPump	0	1
Pump	0	1
StopPump	0	0
NoReceipt	0	1
Receipt	0	0
TurnOff	1	1

Test#18

Test#18:		
Function	**Expected Output**	**Actual Output**
Activate 4.5 5.5 6.5	1	1
StartPump	0	0
Pump	0	0
PayCredit	1	1
Approved	1	1
Cancel	1	1
PayCash 10.5	1	1
Regular	1	1
StartPump	1	1

Pump	1	1
Pump	1	1
Pump	1	1
StopPump	0	0
Receipt	1	1
PayCash -10.5	0	1
Super	0	1
StartPump	0	1
Pump	0	1
NoReceipt	0	1
TurnOff	1	1

Test# 20

Test#20:		
Function	**Expected Output**	**Actual Output**
Activate 4.5 5.5 6.5	1	1
PayCash -20	0	1
Regular	0	1
StartPump	0	1
StopPump	0	1
Receipt	0	1
Super	0	0
StartPump	0	0
Regular	0	0
TurnOff	1	1

Test #22

Test#22:		
Function	**Expected Output**	**Actual Output**
Activate 4.5 5.5 6.5	1	1
PayCash 5.5	1	1
Super	1	1
StartPump	1	1
Pump	1	0
StopPump	1	1
NoReceipt	1	1
TurnOff	1	1

Conclusion

In this project three testing methods are used to test the GasPump class. Transition pair, Default/Ghost, and multiple condition testing. Expected values for all the test cases were calculated from the provided EFSM model for GasPump. There are in total of 22 valid test cases and 11 non executable conditions identified. All the non-executable conditions are encountered in the multiple condition testing. There are in total of 4 failed test cases, one from Default/Ghost testing and the rest from multiple condition testing. The defect occurs due to the methods PayCash() and Pump() present in the source code. PayCash(), according to the source code does not check whether the entered cash amount is greater than 0 which it should according to the EFSM. In the Pump() method the condition cash == price * (L+1) is never checked, hence incorrect output is obtained. All the valid test cases mentioned above are added to a test suite file TS.txt and are verified through a test suite checker file. The source code used in this project is added in the report along with relevant testing-oriented methods.

Source Code and Test Driver

```
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("SOFTWARE TESTING & ANALYSIS PROJECT - GASPUMP CLASS");
        System.out.println("\nCS 589 : FALL 2022");
        System.out.println("Devansh Goel - A20490554");
        @SuppressWarnings("resource")
        Scanner keyIn = new Scanner(System.in);
        System.out.print("Press The ENTER KEY To Continue");
        keyIn.nextLine();
        // To execute program if user gives 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.Midgrade");
                    System.out.println("\t   9.StartPump");
```

```
System.out.println(x: "\t   9.StartPump");
System.out.println(x: "\t  10.Pump");
System.out.println(x: "\t  11.StopPump");
System.out.println(x: "\t  12.NoReceipt");
System.out.println(x: "\t  13.Receipt");
System.out.println(x: "\t  14.TurnOff");
System.out.println(x: "\n *#### TESTING ORIENTED METHOD #####*");
System.out.println(x: "\n\t  15.SHOW PRICE");
System.out.println(x: "\t  16.TOTAL");
System.out.println(x: "\t  17.CASH PAID");
System.out.println(x: "\t  18.NUMBER OF GALLONS PUMPED");
System.out.println(x: "\t  19.SHOW STATE");
System.out.println(x: "\t  20.QUIT GASPUMP DRIVER");
System.out.println(x: "\n Enter your Choice:");
s = br.readLine();

choice = Integer.parseInt(s);
switch (choice) { //To execute the user entered choice
case 0:
    System.out.print(s: "\nEnter value of Regular Price: ");
    x = Float.parseFloat(br.readLine());
    System.out.print(s: "\nEnter value of Super Price: ");
    y = Float.parseFloat(br.readLine());
    System.out.print(s: "\nEnter value of Midgrade 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(s: "\nEnter Amount you want to pay: ");
    z = Float.parseFloat(br.readLine());
    //calling the PayCash function by using the class object
```

```
        //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 Midgrade function by using the class object
        System.out.println("Value returned by method is: " + g.Midgrade());
        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 Pump function by using the class object
        System.out.println("Value returned by method is: " + g.Pump());
        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 Gallons
                        System.out.print("\nNumber of Gallons: " + g.show_Gallons());
                        break;
                    case 19:
                        //Testing Oriented Method to show Current State
                        System.out.print("\nCurrent State: "+ g.show_state());
                        break;
                    case 20:
                        //Exit method
                        System.exit(status: 0);
                    default:
                        //Default Case
                        System.out.println(x: "Please Enter a Valid Input");
                        break;
                    }
                    System.out.println(x: "\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_Gallons()
{
    return L;
}
// testing oriented method
public int show_state()
{
    int s=k;
    if(k==5 && w==1)
    {s=1;}
    return s;
}

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(s: "GAS PUMP IS ON");
        System.out.print(s: "\n");
        return 1;
    } else {
        return 0;
    }

}

public final int PayCredit() {
    if (k == 0) {
        k = 2;
```

```
            System.out.print(s: "CHECKING CREDIT CARD.");
            System.out.print(s: "\n");
            return 1;
        } else {
            return 0;
        }
    }

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

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

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

```
            System.out.print(s: "\n");
            System.out.print(s: "c. MIDGRADE");
            System.out.print(s: "\n");
            return 1;
        } else {
            return 0;
        }
    }

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

    public final int Regular() {
        if (k == 3) {
            k = 4;
            System.out.print(s: "REGULAR IS SELECTED.");
            System.out.print(s: "\n");
            price = Rprice;
            return 1;
        } else {
            return 0;
        }
    }

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

```
            price = Sprice;
            return 1;
        } else {
            return 0;
        }
    }

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

    public final int StartPump() {
        if (k == 4) {
            k = 5;
            L = 0;
            total = 0;
            System.out.print(s: "PUMP IS READY TO DISPOSE ");
            System.out.print(s: "\n");
            System.out.print(s: "# OF GALLONS PUMPED: ");
            System.out.print(L);
            System.out.print(s: "\n");
            System.out.print(s: "TOTAL CHARGE: $");
            System.out.print(total);
            System.out.print(s: "\n");
            return 1;
        } else {
            return 0;
        }
    }

    public final int Pump() {
        if (k == 5) {
            if ((w == 1) || ((cash > price * (L + 1)) && (w == 0))) {
                L = L + 1;
                total = L * price;
                System.out.print(s: "# OF GALLONS PUMPED: ");
                System.out.print(L);
                System.out.print(s: "\n");
                System.out.print(s: "TOTAL CHARGE: $");
```

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

public final int StopPump() {
    if (k == 5) {
        k = 6;
        System.out.print(s: "PUMP STOPPED. ");
        System.out.print(s: "\n");
        System.out.print(s: "# OF GALLONS PUMPED: ");
        System.out.print(L);
        System.out.print(s: "\n");
        System.out.print(s: "TOTAL CHARGE: $");
        System.out.print(total);
        System.out.print(s: "\n");
        if ((w == 0) && (total < cash)) {
            System.out.print(s: "$");
            System.out.print(cash - total);
            System.out.print(s: " OF CASH IS RETURNED");
            System.out.print(s: "\n");
        }
```

```
            System.out.print(s: "DO YOU WANT A RECEIPT?");
            System.out.print(s: "\n");
            return 1;
        } else {
            return 0;
        }
    }

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

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

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

Steps to Execute Test Driver

There are 2 methods to execute the test driver

First Method (Using command prompt)

Steps

Pre-requisite: Java should be installed in the system

1. Open command prompt by clicking “windows button + R” and then enter “cmd”
2. Navigate to the folder named “CS 589 Project” where the GasPump.java file is present.
3. Run javac GasPump.java command to compile the code (this is optional as the code is precompiled and .class file is already present)
4. Run the java file using java GasPump

Second Method (Using STA.exe file)

Steps

Pre-requisite: Java should be installed in the system

1. Navigate to the folder named “CS 589 Project” where the STA.exe file is present.
2. Simply run the file named “STA.exe” which directly executes the test driver.

Note: if second method is not working, please execute the first method.