CS 586

Project Report

Name: Aditya Shivkant Niture
CWID: A20521495

Introduction:

This document describes the coursework of CS586 by implementing different patterns for the given project.

1. MDA-EFSM model for the Gas-Pump components

List of Meta Events for the MDA-EFSM

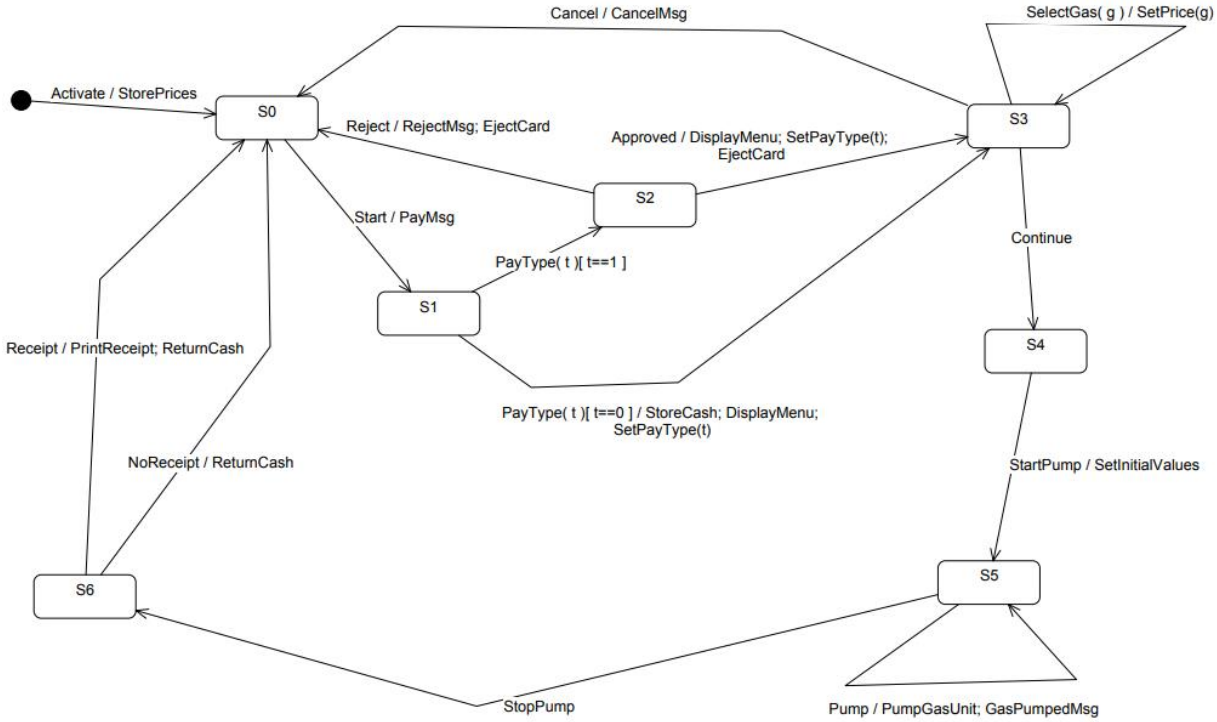
MDA-EFSM Events:

Activate()
Start()
PayType(int t) //credit: t=1; cash: t=0;
Reject()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int g) // Regular: g=1; Diesel: g=2; Premium: g=3
Receipt()
NoReceipt()
Continue()

MDA-EFSM Actions:

StorePrices() // stores price(s) for the gas from the temporary data store
PayMsg() // displays a type of payment method
StoreCash() // stores cash from the temporary data store
DisplayMenu() // display a menu with a list of selections
RejectMsg() // displays credit card not approved message
SetPrice(int g) // set the price for the gas identified by g identifier as in SelectGas(int g);
SetInitialValues() // set G (or L) and total to 0;
PumpGasUnit() // disposes unit of gas and counts # of units disposed and computes Total
GasPumpedMsg() // displays the amount of disposed gas
PrintReceipt() // print a receipt
CancelMsg() // displays a cancellation message
ReturnCash() // returns the remaining cash
SetPayType(t) // Stores pay type t to variable w in the data store
EjectCard() // Card is ejected

A state diagram of MDA-EFSM for GasPumps



MDA-EFSM for Gas Pumps

Pseudo-code of all operations of Input Processors of GasPump 1 and GasPump-2

Operations of the Input Processor(GasPump-1)

```
Activate(int a) {
 if (a>0) {
 d->temp_a=a;
 m->Activate()
 }
}
Start() {
 m->Start();
}
PayCash(int c) {
 if (c>0) {
 d->temp_c=c;
 m->PayType(0)
 }
}
PayCredit() {
```

```
 m->PayType(1);
}
Reject() {
 m->Reject();
}
Approved() {
 m-> Approved();
}
Cancel() {
 m->Cancel();
}
StartPump() {
 m->Continue()
 m->StartPump();
}
Pump() {
if (d->w==1) m->Pump()
else if (d->cash < d->price*(d->L+1)) {
 m->StopPump();
 m->Receipt(); }
 else m->Pump()
}
StopPump() {
 m->StopPump();
 m->Receipt();
}
```

Notice:
cash: contains the value of cash deposited
price: contains the price of the gas L: contains the number of liters already pumped
w: pay type flag (cash: w=0; credit: w=1)
cash,
L, price, w: are in the data store
m: is a pointer to the MDA-EFSM object d: is a pointer to the Data Store object

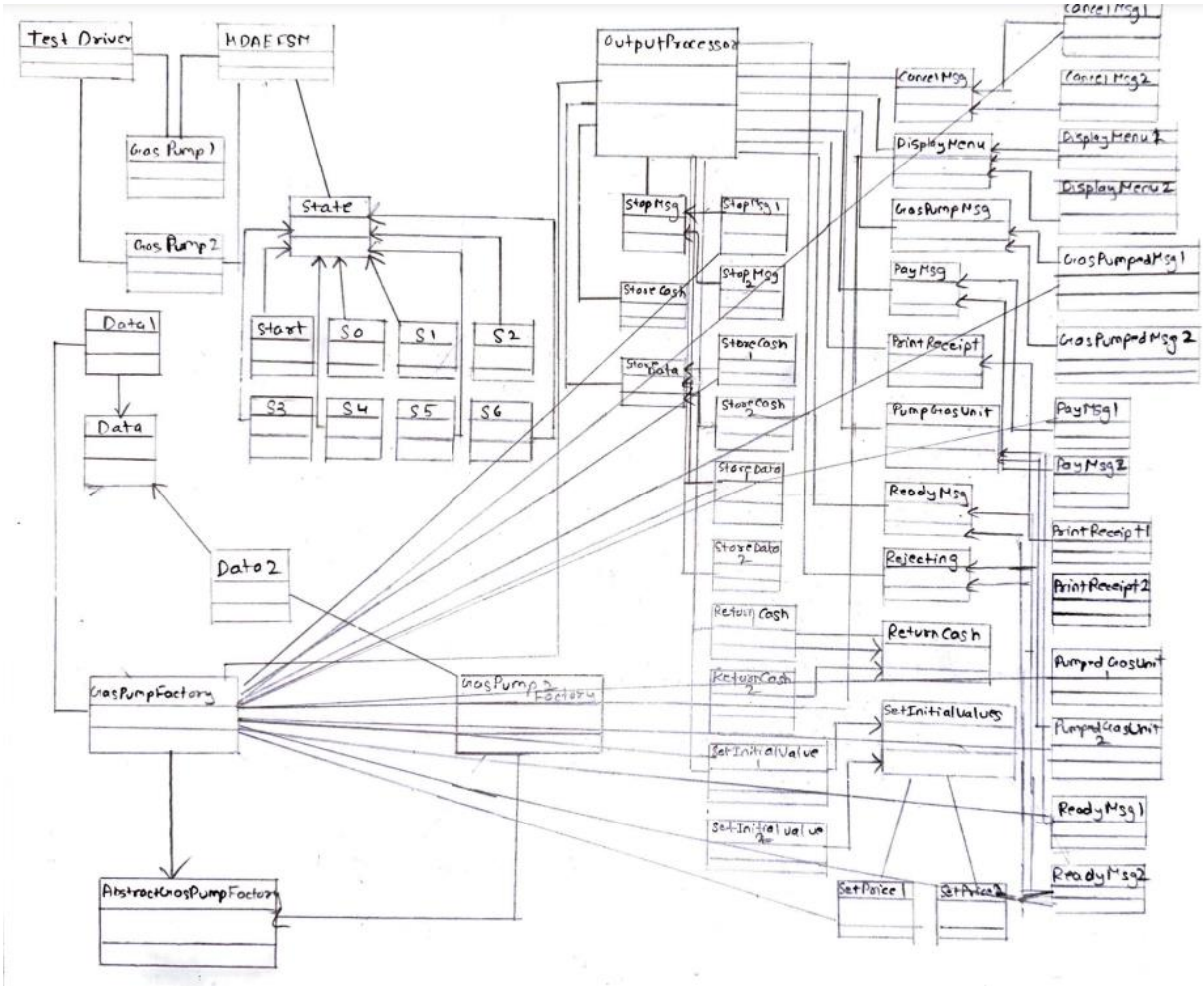
Operations of the Input Processor(GasPump-2)

```
Activate(float a, float b, float c) {
 if ((a>0)&&(b>0)&&(c>0)) {
 d->temp_a=a;
 d->temp_b=b;
 d->temp_c=c
 m->Activate()
 }
}
PayCash(int c) {
 if (c>0) {
 d->temp_cash=c;
 m->PayType(0)
 }
}
```

```
Start() {
 m->Start();
}
}
Cancel() {
 m->Cancel();
}
Diesel() {
 m->SelectGas(2);
 m->Continue();
}
Premium() {
 m->SelectGas(3);
 m->Continue();
}
Regular() {
 m->SelectGas(1);
 m->Continue();
}
StartPump() {
 m->StartPump();
}
PumpGallon() {
if (d->cash < d->price*(d->G+1))
 m->StopPump();
else m->Pump()
}
Stop() {
 m->StopPump();
}
Receipt() {
 m->Receipt();
}
NoReceipt() {
 m->NoReceipt();
}
```

Notice:
cash: contains the value of cash deposited
price: contains the price of the selected gas
G: contains the number of Gallons already
Pumped
cash, G, price are in the data store
m: is a pointer to the MDA-EFSM object
d: is a pointer to the Data Store object

2.Class diagram(s) of the MDA of the Gas-Pump components.



3. For each class in the class diagram(s)

a. Describe the purpose of the class, i.e., responsibilities.
Input Processor:

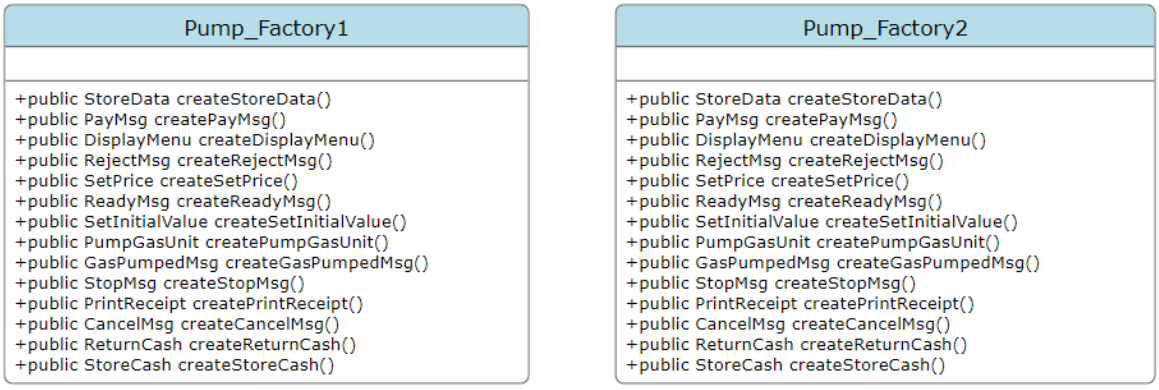
-------Diagram------

Here User can select either of the Gas-pumps and the Pump1Factory or Pump2factory is created along with the objects and it accepts the input data given by the user and passes it to the specific gas pump selected by the user.

GasPump1: Contains Methods related to GasPump1
GasPump2: Contains Methods related to GasPump2

ConcreteFactory: It returns the objects of strategy classes Data Store of respective GasPump1 and GasPump2.

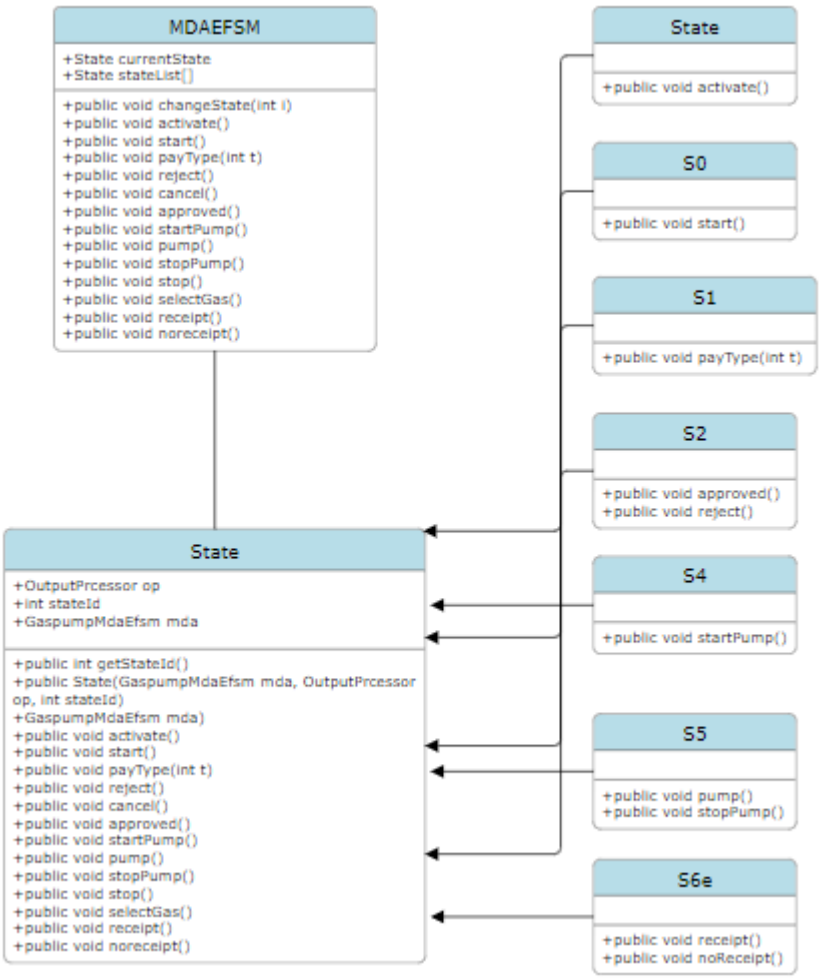
Pump_Factory1: Pump_Factory2:



For all state changes this is the main class and it contains list of states and its current state and operations are explained in the source code as same and explained in detail in the below class diagram attached.

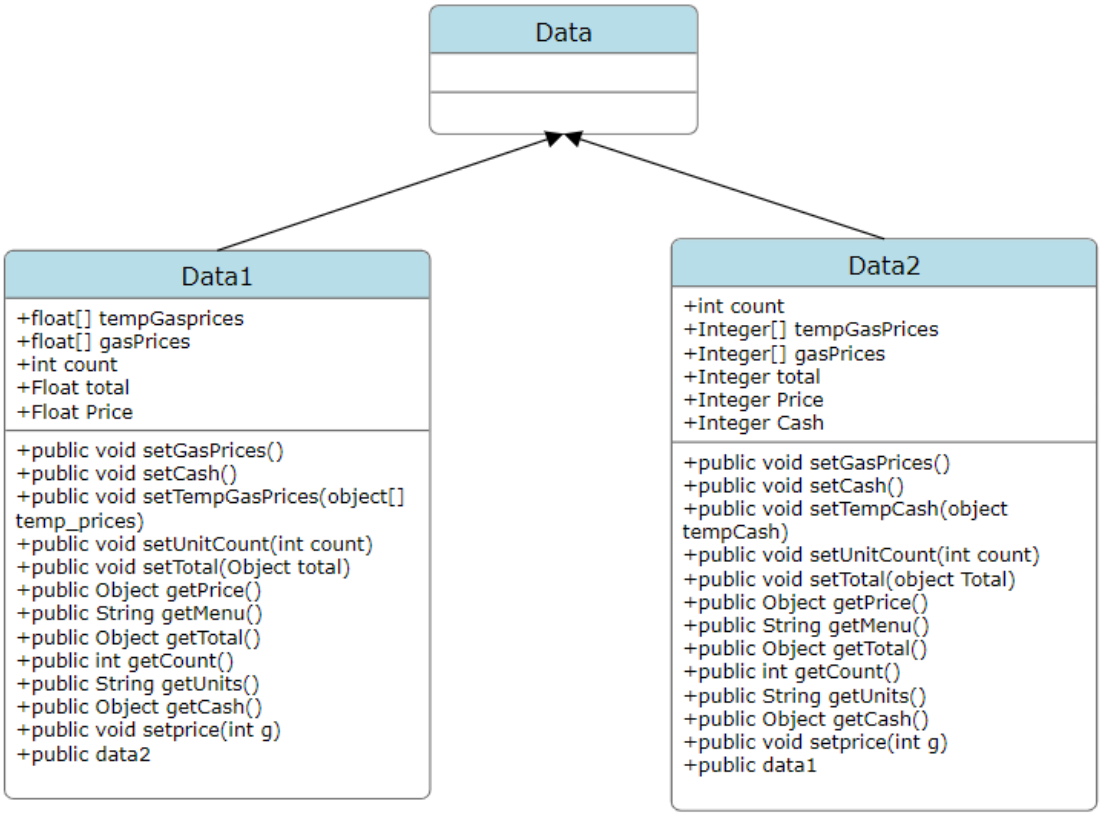
List of States:

State	- [0]
S0	- [1]
S1	- [2]
S2	- [3]
S3	- [4]
S4	- [5]
S5	- [6]
S6	- [7]

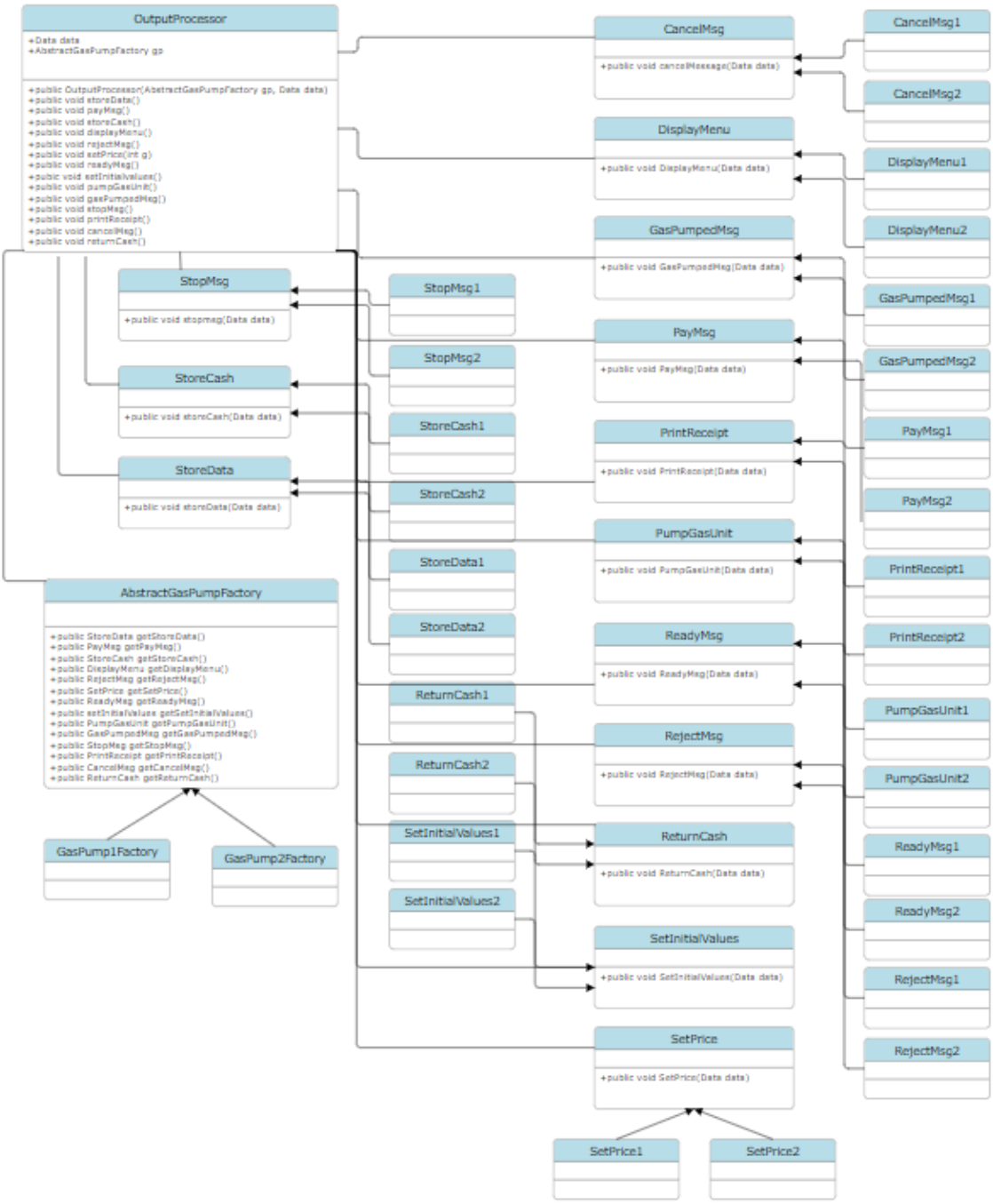


Data Class:

The methods and attributes are listed in the class diagram as below. Data 1 & Data 2 – This will contain variables related to GasPump1 and GasPump2



OutputProcessor: The Operations of OutputProcessor are explained in the source code below is the class diagram with the list of attributes and methods. This class gets the object from the concrete factory(GasPump1Factory,GasPump2factory) and performs the associated operation in the Strategy Pattern classes w.r.t Gaspump1 and gasPump2



b. Describe the responsibility of each operation supported by each class.

pump1 Class:

Activate(float): Activates gas-pump 1 with initial price per gallon.
Start(): Starts the transaction process.
PayCredit(): Allows the customer to pay using a credit card.
Reject(): Notifies the customer that their credit card is rejected.
Cancel(): Cancels the transaction and resets the system.
Approved(): Notifies the customer that their credit card is approved.
Regular(): Selects regular gas type.

StartPump(): Starts pumping gas.
PumpGallon(): Dispenses one gallon of gas.
StopPump(): Stops pumping gas.

pump2 Class:

Activate(int, int, int): Activates gas-pump 2 with initial parameters.
Start(): Starts the transaction process.
PayCash(float): Allows the customer to pay with cash.
Cancel(): Cancels the transaction and resets the system.
Super(): Selects super gas type.
Premium(): Selects premium gas type.
Regular(): Selects regular gas type.
StartPump(): Starts pumping gas.
PumpLiter(): Dispenses one liter of gas.
Stop(): Stops pumping gas.
Receipt(): Prints a receipt for the transaction.
NoReceipt(): Completes the transaction without printing a receipt.

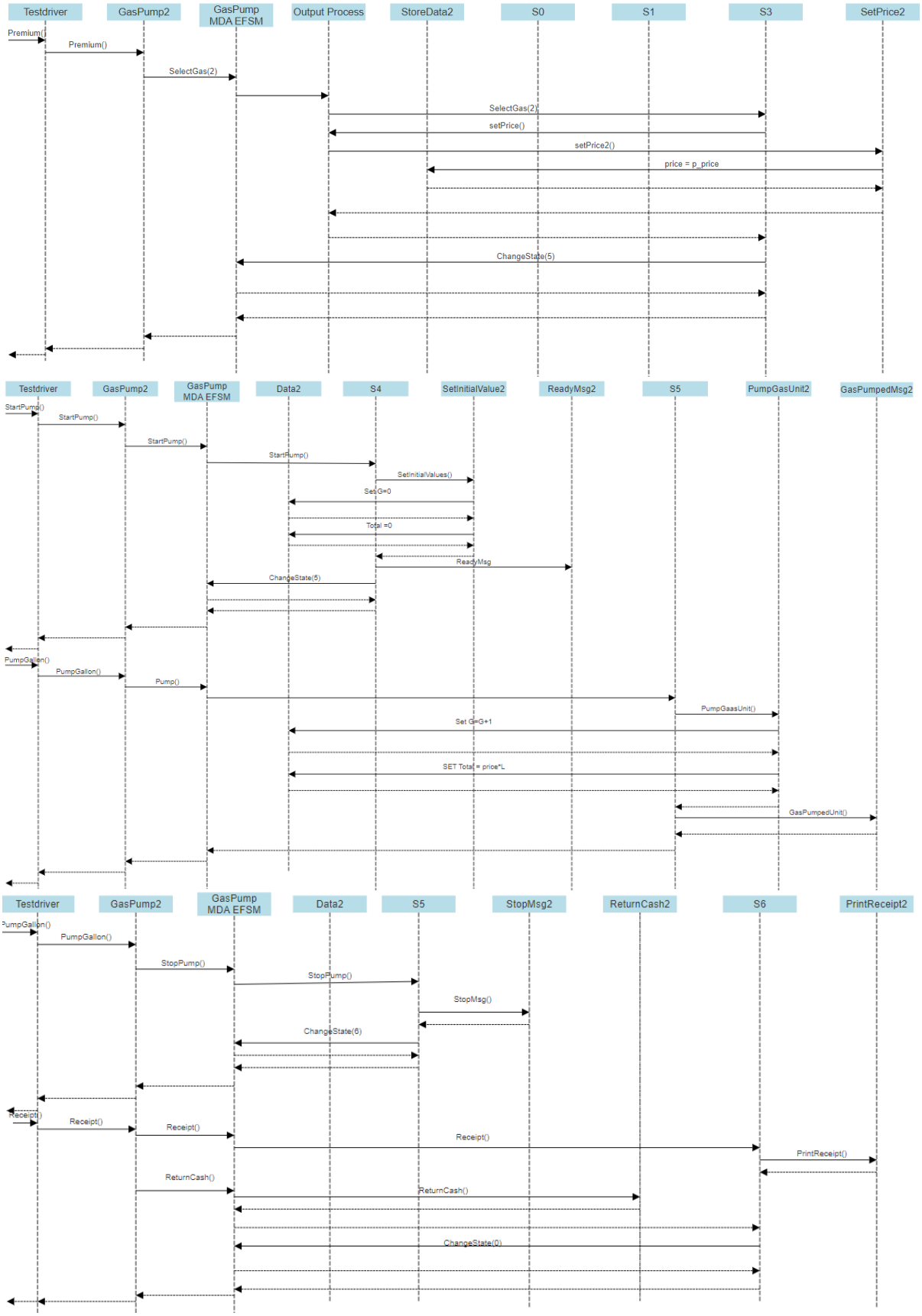
state Class (and its subclasses like **s1,s2,s3**):

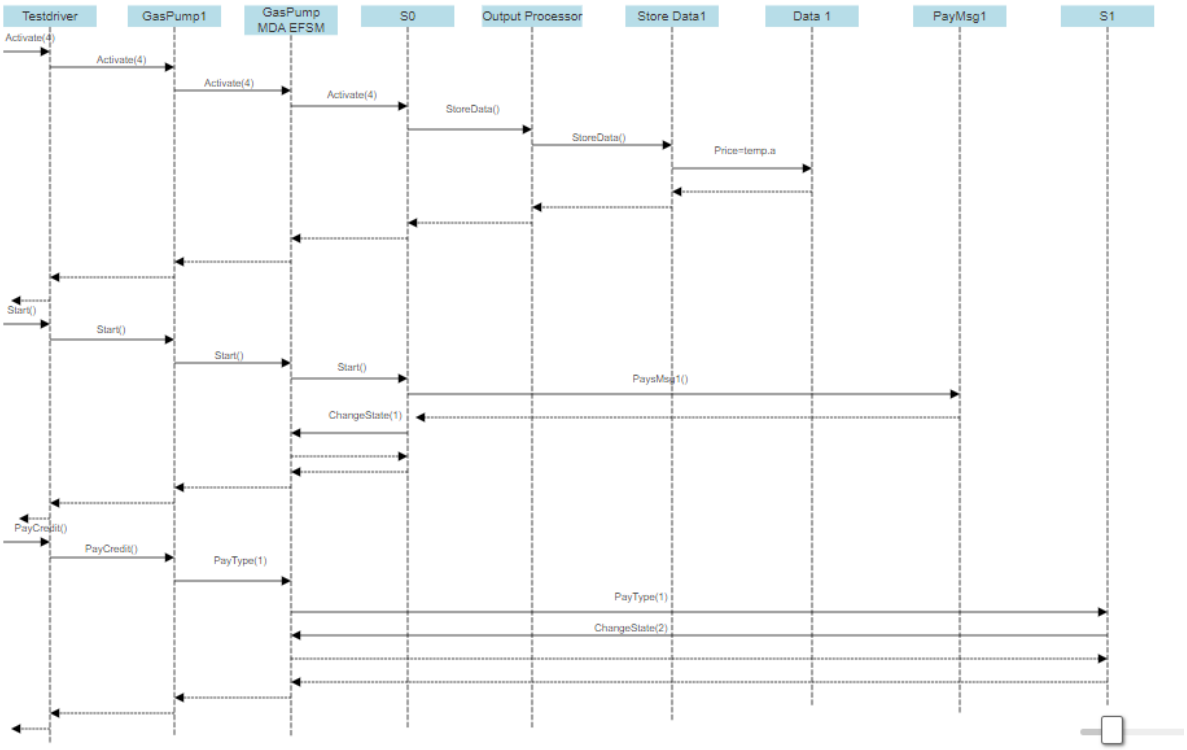
Cancel(): Handles cancellation of the transaction.
SelectGas(int): Handles selection of gas type.
StartPump(): Starts the process of pumping gas.
Pump(): Handles the pumping of gas.
StopPump(): Stops the process of pumping gas.
Receipt(): Prints a receipt for the transaction.
NoReceipt(): Completes the transaction without printing a receipt.
Start(): Starts the transaction process.
PayType(int): Handles the selection of payment type.
Activate(): Handles the activation of the gas pump.

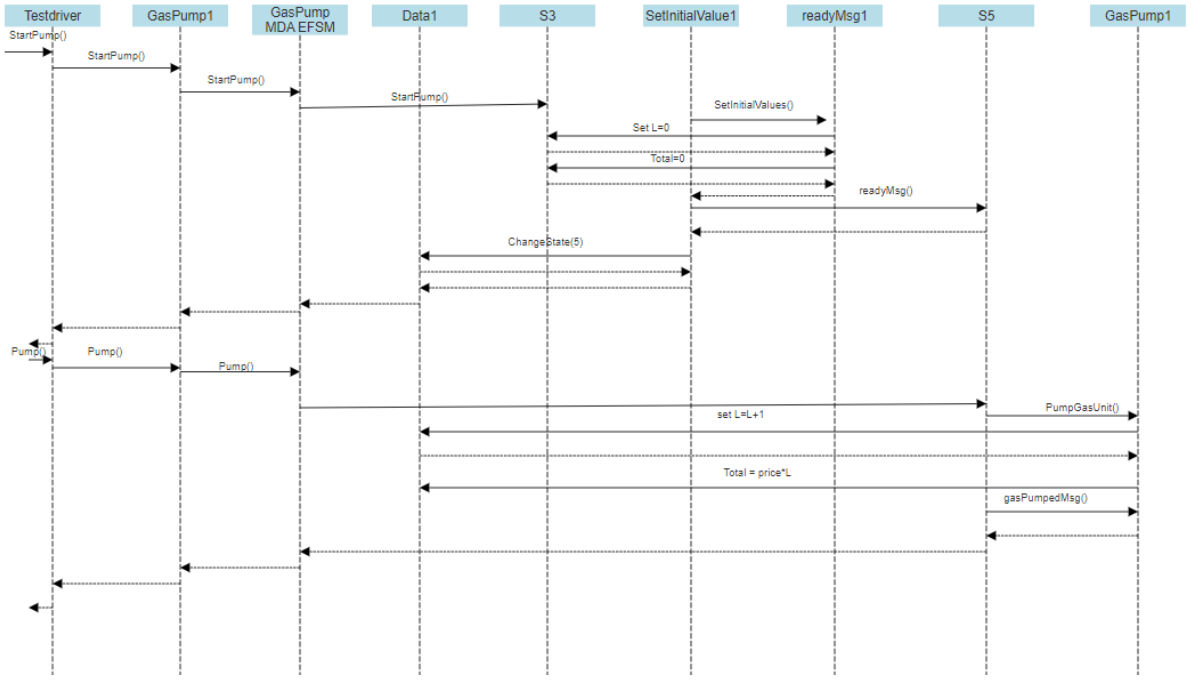
Pump_Factory Class:

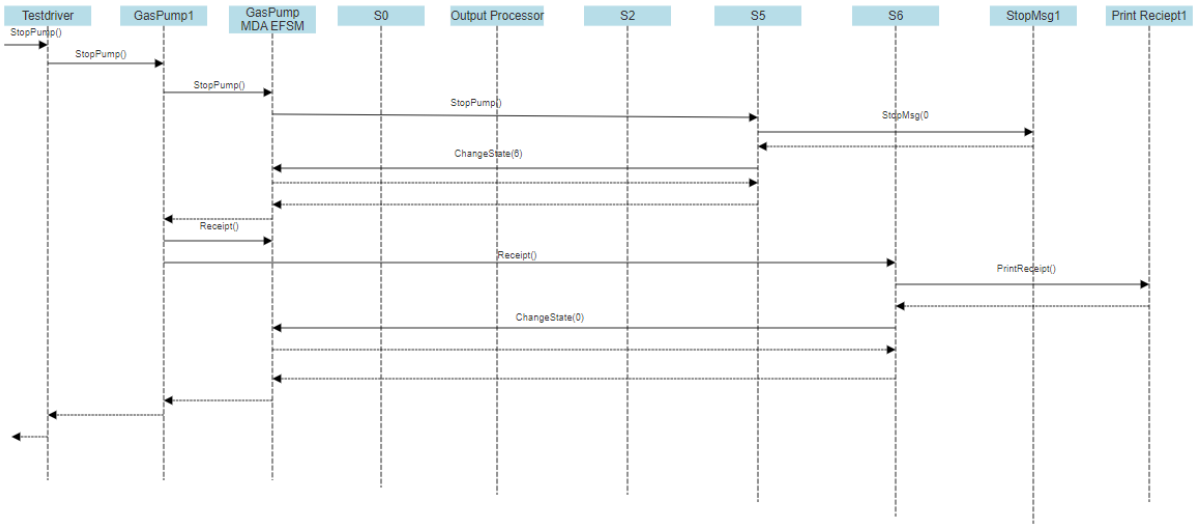
createStoreCash(): Creates an instance of a class responsible for handling cash-related operations.
createDisplayMenu(): Creates an instance of a class responsible for displaying the menu.
createGasPumpedMsg(): Creates an instance of a class responsible for displaying messages related to gas pumping.
createStopMsg(): Creates an instance of a class responsible for displaying stop messages.
createPumpGasUnit(): Creates an instance of a class responsible for pumping gas units.
createGasPumpedMsg(): Creates an instance of a class responsible for displaying gas pumped messages.
createSetPrice(): Creates an instance of a class responsible for setting gas prices.
createReadyMsg(): Creates an instance of a class responsible for displaying ready messages.
createReturnCash(): Creates an instance of a class responsible for returning cash.
createWrongPinMsg(): Creates an instance of a class responsible for displaying wrong PIN messages.

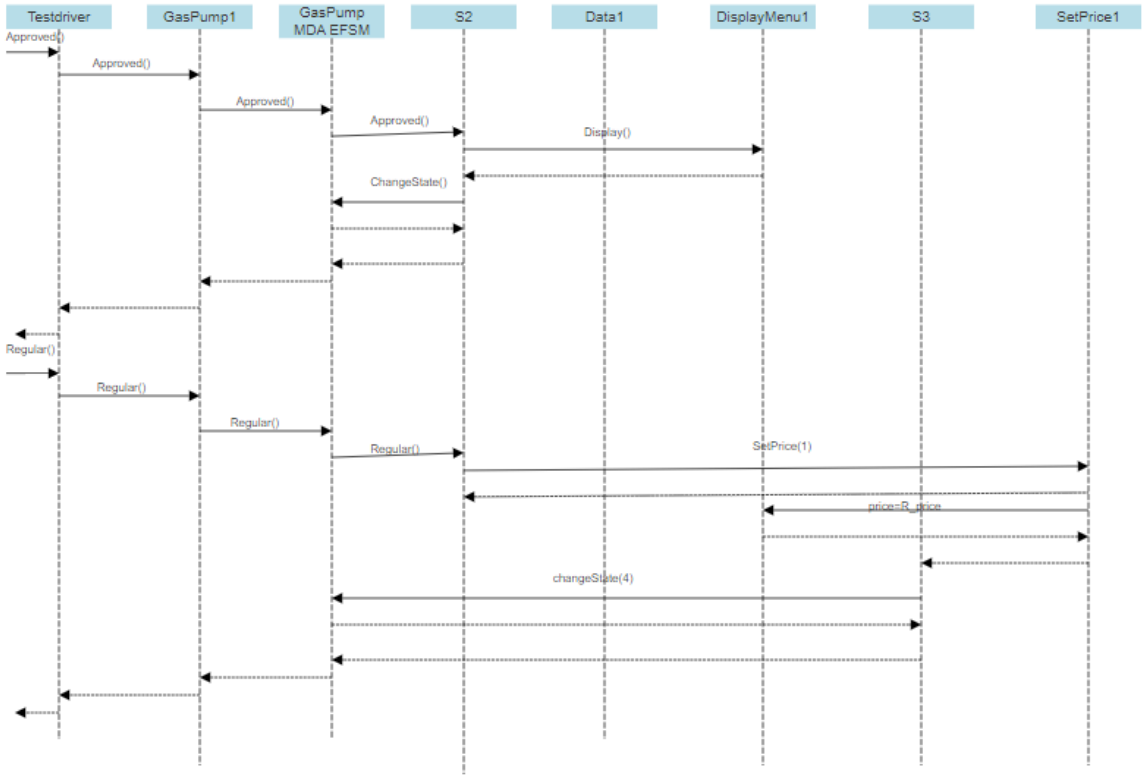
4. Dynamics. Provide two sequence diagrams for two Scenarios:

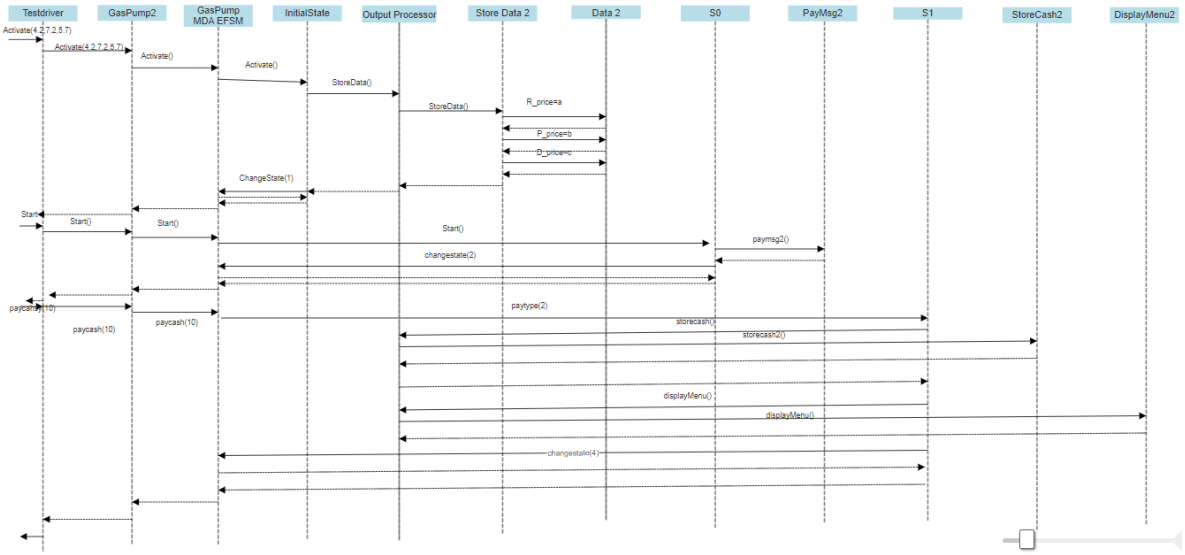



Testdriver
GasPump1
GasPump MDA EFSM
S0
Output Processor
Store Data1
Data 1
PayMsg1
S1
Activate(4)
Activate(4)
Activate(4)
Activate(4)
StoreData()
StoreData()
Price=temp.a
Start()
Start()
Start()
Start()
PaysMsg1()
ChangeState(1)
PayCredit()
PayCredit()
PayType(1)
PayType(1)
ChangeState(2)


Testdriver
GasPump1
GasPump MDA EFSM
Data1
S3
SetInitialValue1
readyMsg1
S5
GasPump1
StartPump()
StartPump()
StartPump()
StartPump()
SetInitialValues()
Set L=0
Total=0
readyMsg()
ChangeState(5)
Pump()
Pump()
Pump()
PumpGasUnit()
set L=L+1
Total = price*L
gasPumpedMsg()


Testdriver
GasPump1
GasPump MDA EFSM
S0
Output Processor
S2
S5
S6
StopMsg1
Print Reciept1
StopPump()
StopPump()
StopPump()
StopPump()
StopMsg()
ChangeState(6)
Receipt()
Receipt()
PrintReceipt()
ChangeState(0)


Testdriver
GasPump1
GasPump MDA EFSM
S2
Data1
DisplayMenu1
S3
SetPrice1
Approved()
Approved()
Approved()
Approved()
Display()
ChangeState()
Regular()
Regular()
Regular()
Regular()
SetPrice(1)
price=R_price
changeState(4)


Testdriver
GasPump2
GasPump MDA EFSM
InitialState
Output Processor
Store Data 2
Data 2
S0
PayMsg2
S1
StoreCash2
DisplayMenu2
Activate(4,2,7,2,5,7)
Activate(4,2,7,2,5,7)
Activate()
Activate()
StoreData()
StoreData()
R_price=a
P_price=b
D_price=c
ChangeState(1)
Start
Start()
Start()
Start()
paymsg2()
changestate(2)
paytype(2)
paycash(10)
paycash(10)
paycash(10)
storecash()
storecash2()
displayMenu()
displayMenu()
changestate(4)

Project Executables

I have uploaded a runnable jar file on the blackboard. Steps to run the jar file as below.

1.Copy the .jar file to your desktop

2.Open “Command Prompt” and navigate to your desktop folder

(eg: cd..

cd..

cd users //select the appropriate user

cd Desktop

ls //to make sure that the given jar file is in the desktop)

3.Run the following command to make the jar file running as expected.

Java -jar Filename.jar

Select type of GasPump:

1. GP1
2. GP2

1

****

GP-1 Menu Of Operations

0. Activate(int a)
1. Start
2. PayCredit
3. PayCash(int c)
4. Reject
5. Cancel
6. Approved
7. StartPump
8. Pump
9. StopPump
111. Quit the Program

****

0

Activate

Enter the price parameter a:

4

GasPump1 activated successfully!

****

GP-1 Menu Of Operations

0. Activate(int a)
1. Start
2. PayCredit
3. PayCash(int c)
4. Reject
5. Cancel
6. Approved
7. StartPump
8. Pump
9. StopPump
111. Quit the Program

****

1

Start

Thank you for choosing GasPump-1

Please select payment type

****

GP-1 Menu Of Operations

0. Activate(int a)
1. Start
2. PayCredit
3. PayCash(int c)
4. Reject
5. Cancel
6. Approved
7. StartPump

8. Pump
9. StopPump
111. Quit the Program
****
2
PayCredit
****
GP-1 Menu Of Operations
0. Activate(int a)
1. Start
2. PayCredit
3. PayCash(int c)
4. Reject
5. Cancel
6. Approved
7. StartPump
8. Pump
9. StopPump
111. Quit the Program
****
6
Approved
CREDIT CARD APPROVED
Gasoline [$4/liter]
Select (5) to cancel
****
GP-1 Menu Of Operations
0. Activate(int a)
1. Start
2. PayCredit
3. PayCash(int c)
4. Reject
5. Cancel
6. Approved
7. StartPump
8. Pump
9. StopPump
111. Quit the Program
****
7
StartPump
READY
****
GP-1 Menu Of Operations
0. Activate(int a)
1. Start
2. PayCredit
3. PayCash(int c)
4. Reject
5. Cancel
6. Approved

7. StartPump
8. Pump
9. StopPump
111. Quit the Program
****
8
Pump
Pumped 1 Liter of gas
Total gallons pumped: 1
****
GP-1 Menu Of Operations
0. Activate(int a)
1. Start
2. PayCredit
3. PayCash(int c)
4. Reject
5. Cancel
6. Approved
7. StartPump
8. Pump
9. StopPump
111. Quit the Program
****
9
StopPump
STOPPING PUMP ...
1 Liter of gasoline @ $4/Liter
Total: $4
****
GP-1 Menu Of Operations
0. Activate(int a)
1. Start
2. PayCredit
3. PayCash(int c)
4. Reject
5. Cancel
6. Approved
7. StartPump
8. Pump
9. StopPump
111. Quit the Program
****

```
Select type of GasPump:
1. GP1
2. GP2
2
**********
GP-2 Menu Of Operations
0. Activate(float a, float b, float c)
1. Start
2. PayCash(int c)
3. Cancel
4. Regular
5. Premium
6. Diesel
7. StartPump
8. PumpGallon
9. Stop
10. Receipt
11. NoReceipt
111. Quit the Program
**********
0
Activate(float a, float b, float c)
Enter the price parameter a:
4.2
Enter the price parameter b:
7.2
Enter the price parameter c:
5.3
GasPump2 activated successfully!
**********
GP-2 Menu Of Operations
0. Activate(float a, float b, float c)
1. Start
2. PayCash(int c)
3. Cancel
4. Regular
5. Premium
6. Diesel
7. StartPump
8. PumpGallon
9. Stop
10. Receipt
11. NoReceipt
111. Quit the Program
**********
1
Start
**********
GP-2 Menu Of Operations
0. Activate(float a, float b, float c)
```

1. Start
2. PayCash(int c)
3. Cancel
4. Regular
5. Premium
6. Diesel
7. StartPump
8. PumpGallon
9. Stop
10. Receipt
11. NoReceipt
111. Quit the Program
****
2
PayCash(int c)
Insert cash (enter $ amount):
10
Amount of cash inserted: $10
Please select gas type:
(4) Regular [$4.2/Gallon]
(5) Premium [$7.2/Gallon]
(6) Diesel [$5.3/Gallon]
Select (3) to cancel
****
GP-2 Menu Of Operations
0. Activate(float a, float b, float c)
1. Start
2. PayCash(int c)
3. Cancel
4. Regular
5. Premium
6. Diesel
7. StartPump
8. PumpGallon
9. Stop
10. Receipt
11. NoReceipt
111. Quit the Program
****
5
Premium
Select (7) to start the pump
****
GP-2 Menu Of Operations
0. Activate(float a, float b, float c)
1. Start
2. PayCash(int c)
3. Cancel
4. Regular
5. Premium
6. Diesel

7. StartPump
8. PumpGallon
9. Stop
10. Receipt
11. NoReceipt
111. Quit the Program
****
7
StartPump
READY
Select (8) to dispense 1 Gallon of Premium gasoline
Select (9) to stop
****
GP-2 Menu Of Operations
0. Activate(float a, float b, float c)
1. Start
2. PayCash(int c)
3. Cancel
4. Regular
5. Premium
6. Diesel
7. StartPump
8. PumpGallon
9. Stop
10. Receipt
11. NoReceipt
111. Quit the Program
****
8
PumpGallon
Pumped 1 gallon of Premium gasoline
Total gallon pumped: 1.0
****
GP-2 Menu Of Operations
0. Activate(float a, float b, float c)
1. Start
2. PayCash(int c)
3. Cancel
4. Regular
5. Premium
6. Diesel
7. StartPump
8. PumpGallon
9. Stop
10. Receipt
11. NoReceipt
111. Quit the Program
****
8
PumpGallon
STOPPING PUMP ...

****
GP-2 Menu Of Operations
0. Activate(float a, float b, float c)
1. Start
2. PayCash(int c)
3. Cancel
4. Regular
5. Premium
6. Diesel
7. StartPump
8. PumpGallon
9. Stop
10. Receipt
11. NoReceipt
111. Quit the Program
****
10
PrintReceipt
****
1.0 Gallon of Premium gasoline @ $7.2/gallon
Total: $7.2
Cash Inserted: $10
Cash Returned: $2.80
****
Cash to return: $2.80
Returning $2.80
****
GP-2 Menu Of Operations
0. Activate(float a, float b, float c)
1. Start
2. PayCash(int c)
3. Cancel
4. Regular
5. Premium
6. Diesel
7. StartPump
8. PumpGallon
9. Stop
10. Receipt
11. NoReceipt
111. Quit the Program
****