PROJECT REPORT

<u>Software Systems Architectures (CS-586-01)</u>

Name: Sona Shree Reddy Gutha

CWID: A20532599

1. MDA-EFSM model for the GP components

a) A list of meta events for the MDA-EFSM

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()

b) A list of meta actions for the MDA-EFSM with their descriptions

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

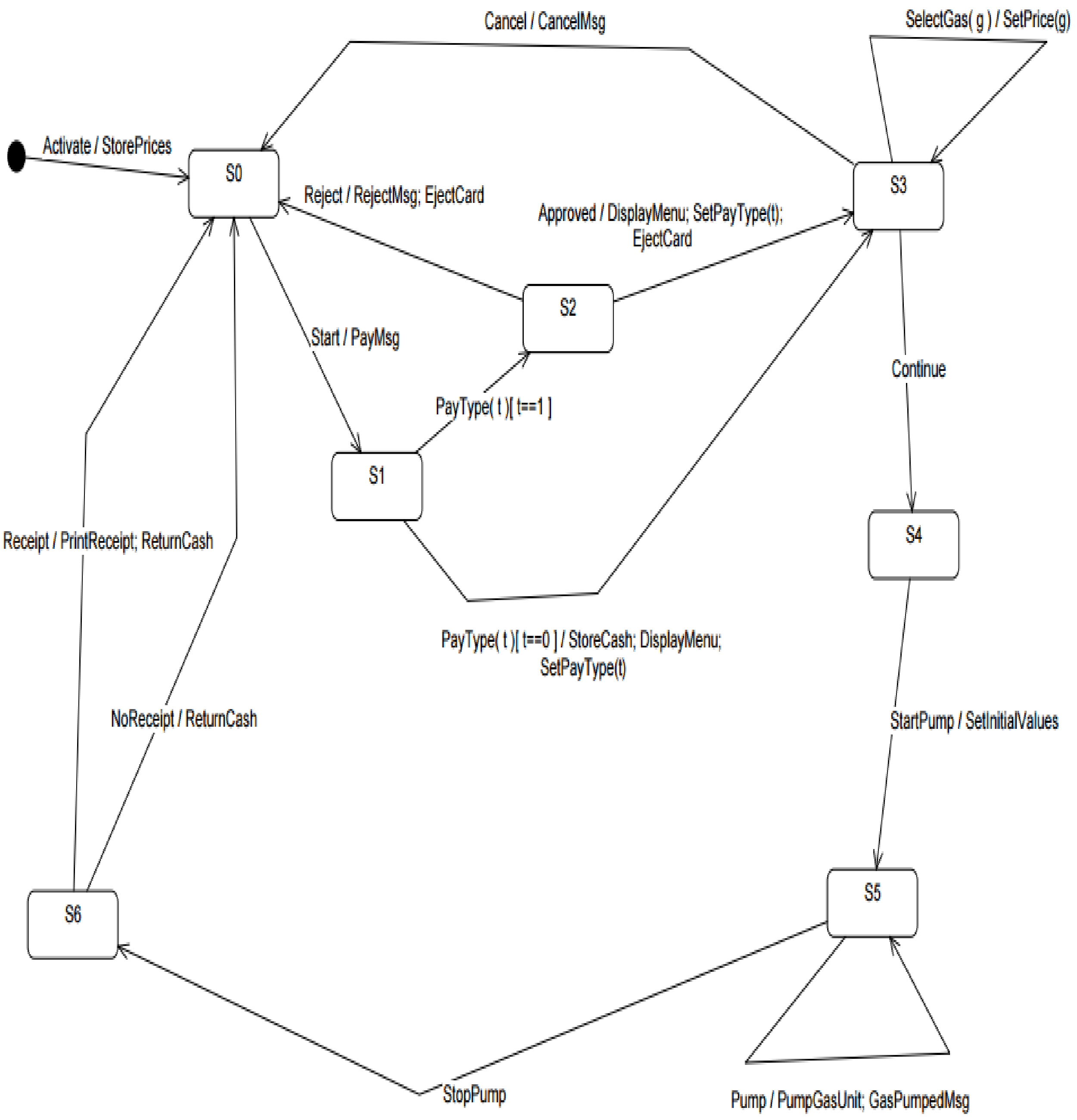
CancelMsg() // displays a cancellation message

ReturnCash() // returns the remaining cash

SetPayType(t) // Stores pay type t to variable w in the data store

EjectCard() // Card is ejected

c. A state diagram of the MDA-EFSM


Cancel / CancelMsg
SelectGas(g) / SetPrice(g)
Activate / StorePrices
S0
Reject / RejectMsg; EjectCard
Approved / DisplayMenu; SetPayType(t); EjectCard
S3
S2
Start / PayMsg
Continue
PayType(t)[t==1]
S1
S4
PayType(t)[t==0] / StoreCash; DisplayMenu; SetPayType(t)
Receipt / PrintReceipt; ReturnCash
NoReceipt / ReturnCash
StartPump / SetInitialValues
S5
S6
StopPump
Pump / PumpGasUnit; GasPumpedMsg

MDA-EFSM for Gas Pumps

d. Pseudo-code of all operations of Input Processors of Gas Pump: GP-1 and GP-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

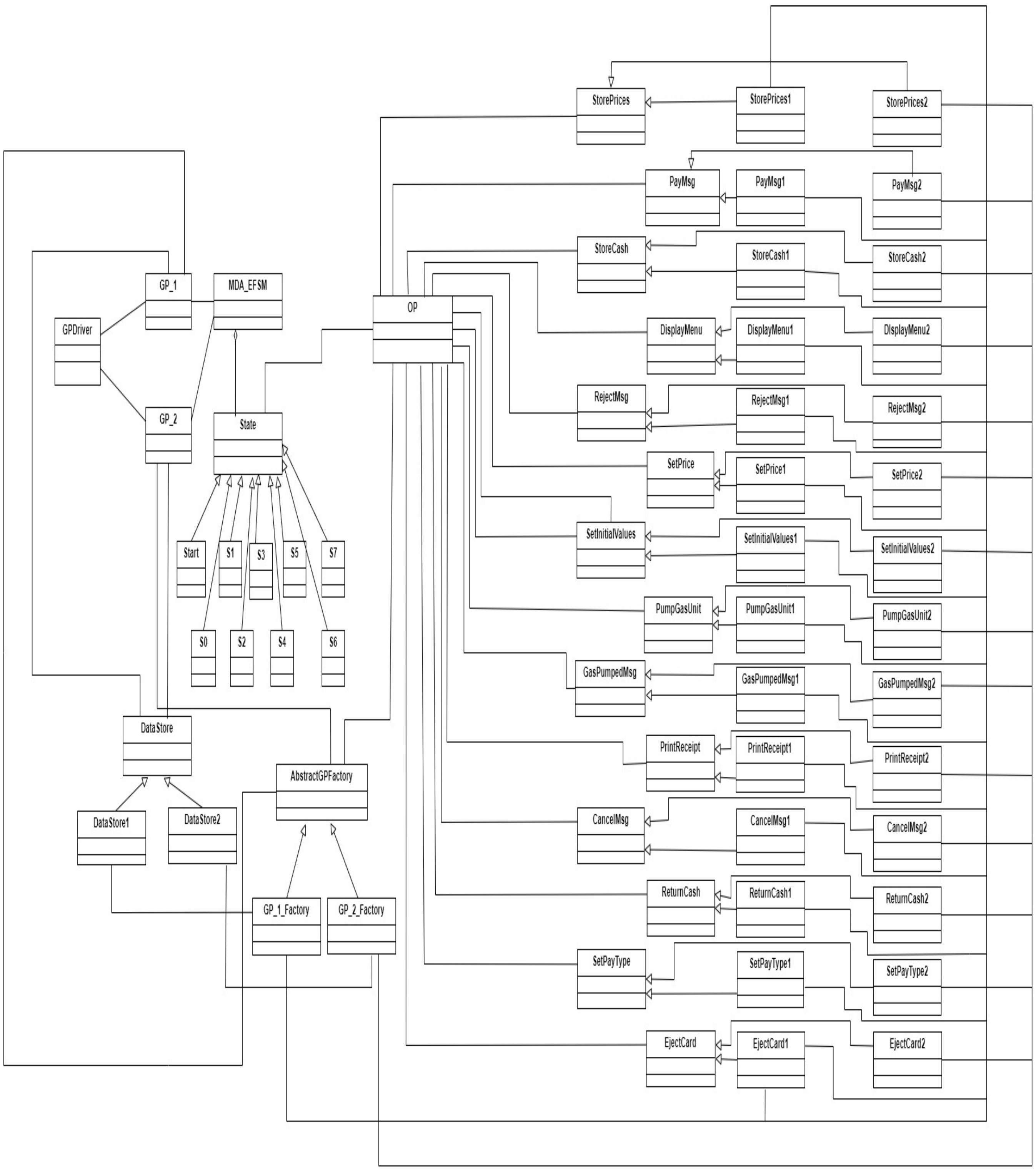
G: contains the number of Gallons already pumped

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 using state, strategy and abstract factory pattern



Note: For the ease if readability of the class diagram, all variables and methods of the classes are mentioned in detail in section 3.

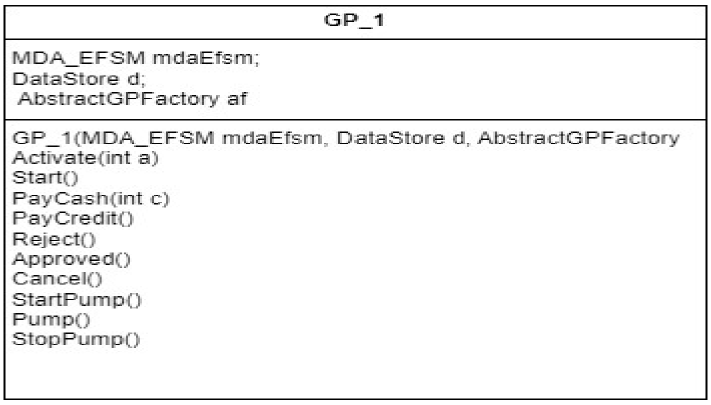
3) Classes Information

Class ‘**GPDriver’**:

This provides options for the user to select between gaspump1 and gaspump2. The gp1 or gp2 factory classes are created which takes the user input and performs the required operations.

Class ‘**GP_1’**:

This class consists of the methods to perform specified actions of gaspump 1



```
GP_1
MDA_EFSM mdaEfsm;
DataStore d;
 AbstractGPFactory af
GP_1(MDA_EFSM mdaEfsm, DataStore d, AbstractGPFactory
Activate(int a)
Start()
PayCash(int c)
PayCredit()
Reject()
Approved()
Cancel()
StartPump()
Pump()
StopPump()
```

Operations:

GP_1() is the constructor for GP1 which takes MDAEfsm, data store and abstract factory as parameters

Activate (int a) // the gas pump is activated where a is the price of the gas per liter

Start() //start the transaction

PayCredit() // pay for gas by a credit card

Reject() // credit card is rejected

Cancel() // cancel the transaction

Approved() // credit card is approved

PayCash(int c) // pay for gas by cash, where c represents prepaid cash

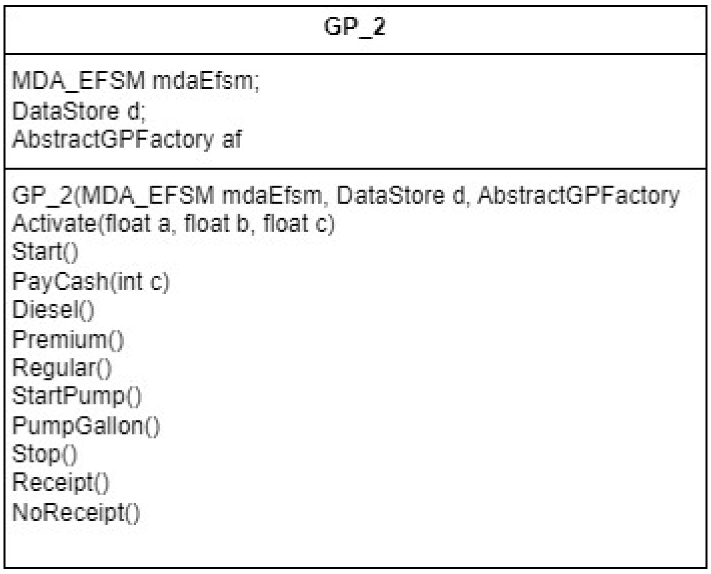
StartPump() // start pumping gas

Pump() // one liter of gas is disposed

StopPump() // stop pumping gas

Class ‘**GP-2**’:

This class consists of the methods to perform specified actions of gaspump 1



GP_2

MDA_EFSM mdaEfsm;
DataStore d;
AbstractGPFactory af

GP_2(MDA_EFSM mdaEfsm, DataStore d, AbstractGPFactory
Activate(float a, float b, float c)
Start()
PayCash(int c)
Diesel()
Premium()
Regular()
StartPump()
PumpGallon()
Stop()
Receipt()
NoReceipt()

Operations:

GP_2() is the constructor for GP2 which takes MDAEfsm, data store and abstract factory as parameters

Activate (float a, float b, float c) // the gas pump is activated where a is the price of Regular gas, b is

//the price of Premium gas and c is the price of Diesel per Gallon

Start() //start the transaction

PayCash(int c) // pay for gas by cash, where c represents prepaid cash

Cancel() // cancel the transaction

Premium() // Premium gas is selected

Regular() // Regular gas is selected

Diesel() // Diesel gas is selected

StartPump() // start pumping gas

PumpGallon() // one Gallon of gas is disposed

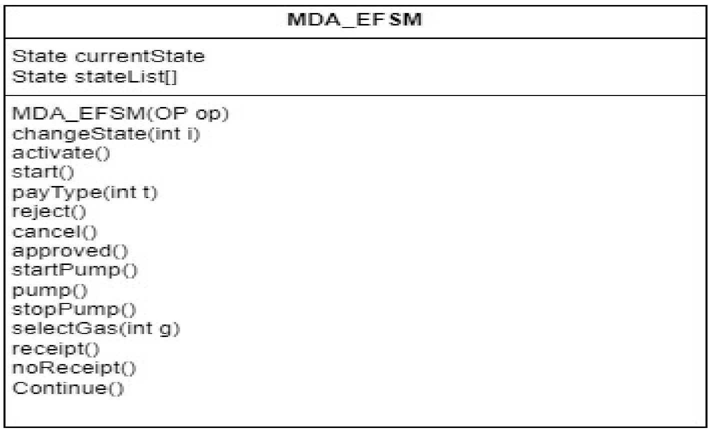
Stop() // stop pumping gas

Receipt() // Receipt is requested

NoReceipt() // No receipt

Class 'MDA_EFSM':

This class contains all the platform independent operations.



Operations:

MDA_EFSM(): Constructor which takes output processor as the parameter.

changeState(int i): This is responsible to change the state from one to another.

Activate: Activates the selected gaspump

Start: starts the selected gaspump

PayType(int t): sets the payment type to cash or credit based on the t value

Reject(): This rejects the credit card when not approved

Cancel(): This is responsible to cancel the transaction

Approved(): This method is responsible to approve the credit card transactions

startPump(): This is responsible to start the selected gas pump

pump(): This is responsible to pump gas and keep track of the amount of gas pumped and the total cost.

Stoppump(): This is responsible to stop the gas disposal

Selectgas(int g): This method is used to select the type of the gas

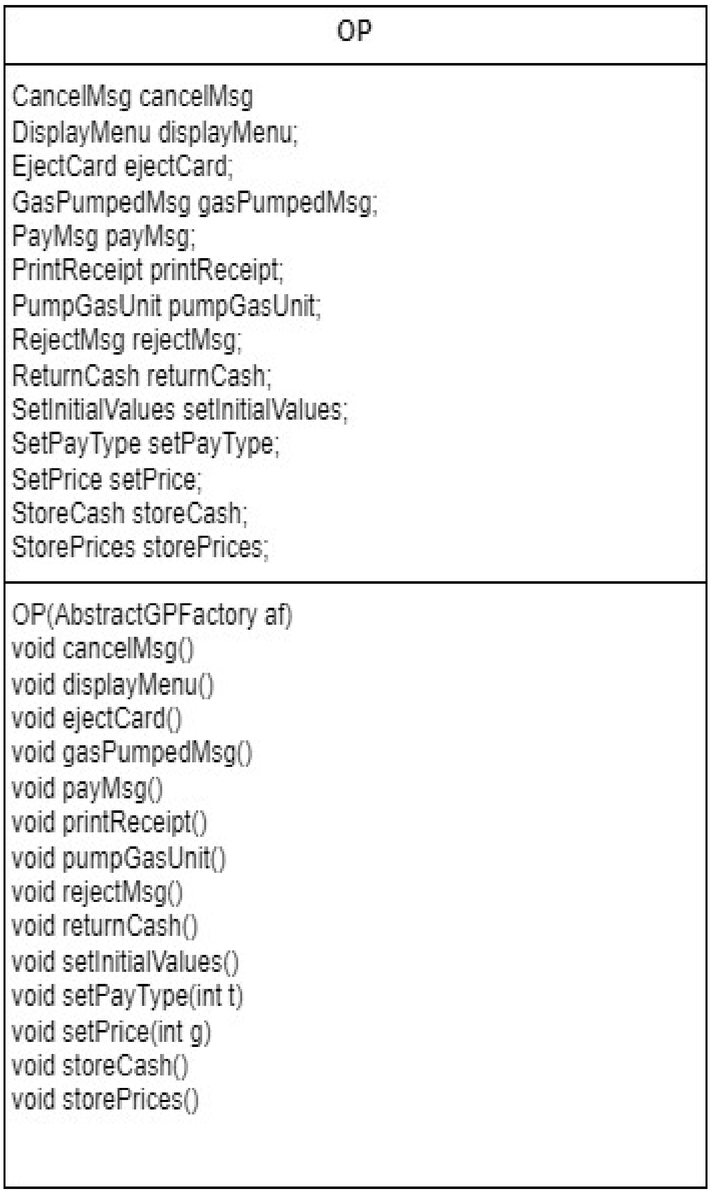
Receipt: This generates the receipt with the details

noReceipt: This does not generate any receipt

Continue: Continues the operations

Class ‘OP’:

This is the output processor. Each actions have 2 strategies for 2 gas components. This class gets the abstract factory based on which particular strategy operation is performed. If the abstract factory is for gaspump 1 the actions related to gaspump 1 strategies are executed. Similarly for GP2 If the abstract factory is for gaspump 2 the actions related to gaspump 2 strategies are executed



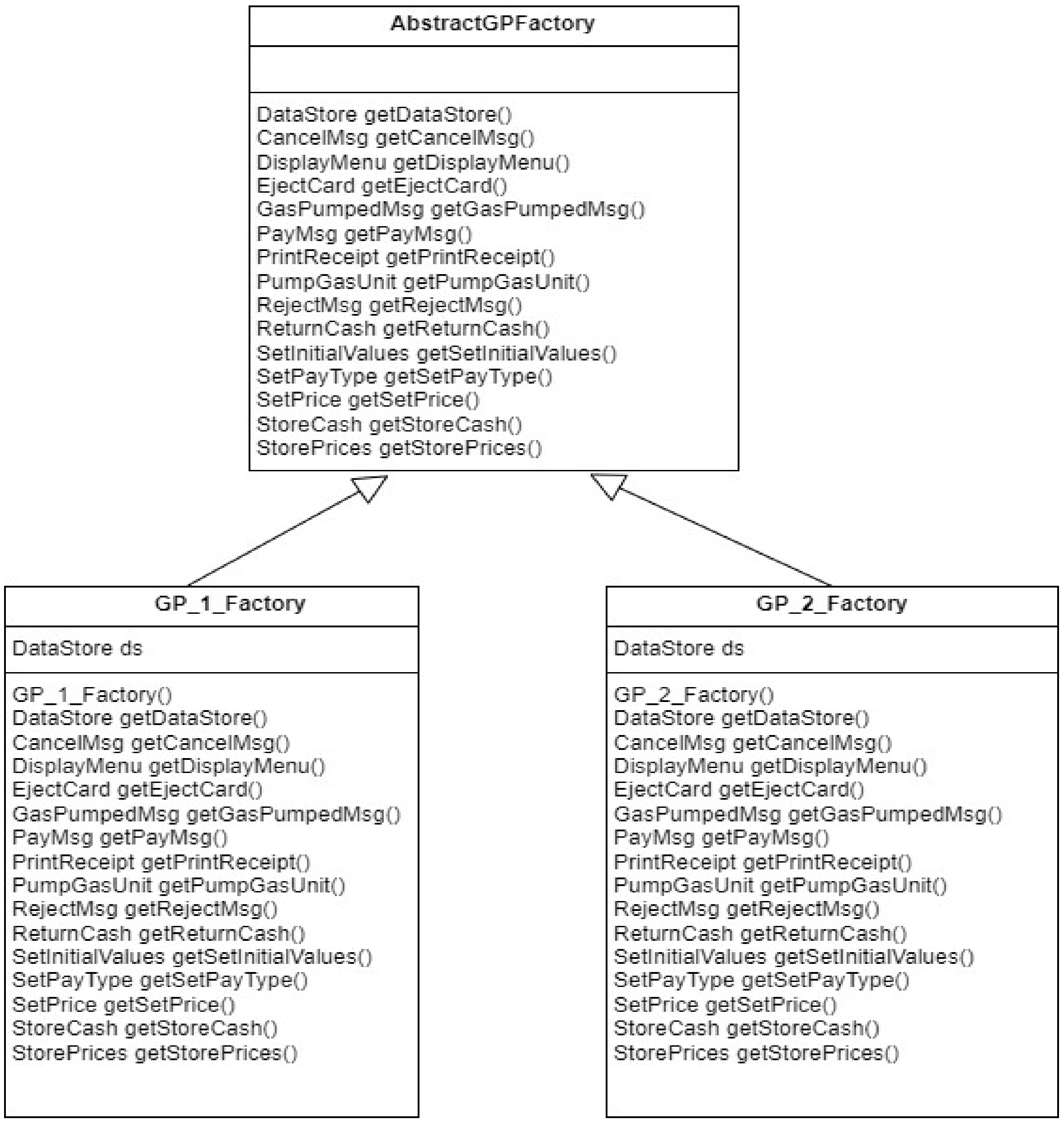
```
OP

CancelMsg cancelMsg
DisplayMenu displayMenu;
EjectCard ejectCard;
GasPumpedMsg gasPumpedMsg;
PayMsg payMsg;
PrintReceipt printReceipt;
PumpGasUnit pumpGasUnit;
RejectMsg rejectMsg;
ReturnCash returnCash;
SetInitialValues setInitialValues;
SetPayType setPayType;
SetPrice setPrice;
StoreCash storeCash;
StorePrices storePrices;

OP(AbstractGPFactory af)
void cancelMsg()
void displayMenu()
void ejectCard()
void gasPumpedMsg()
void payMsg()
void printReceipt()
void pumpGasUnit()
void rejectMsg()
void returnCash()
void setInitialValues()
void setPayType(int t)
void setPrice(int g)
void storeCash()
void storePrices()
```

This part of the main class diagram above uses **abstract factory pattern**.

Class **AbstractGPFactory**: This class is the abstract class for the factory classed. It contains the abstract methods which are implanted by the factory classes.

Class **GP_1_Factory**: Factory class for then gaspump 1 component which initializes the data store for the gaspump 1 and implements the abstract methods.

Class **GP_2_Factory**: Factory class for then gaspump 2 component which initializes the data store for the gaspump 2 and implements the abstract methods.


AbstractGPFactory
DataStore getDataStore()
CancelMsg getCancelMsg()
DisplayMenu getDisplayMenu()
EjectCard getEjectCard()
GasPumpedMsg getGasPumpedMsg()
PayMsg getPayMsg()
PrintReceipt getPrintReceipt()
PumpGasUnit getPumpGasUnit()
RejectMsg getRejectMsg()
ReturnCash getReturnCash()
SetInitialValues getSetInitialValues()
SetPayType getSetPayType()
SetPrice getSetPrice()
StoreCash getStoreCash()
StorePrices getStorePrices()
GP_1_Factory
DataStore ds
GP_1_Factory()
DataStore getDataStore()
CancelMsg getCancelMsg()
DisplayMenu getDisplayMenu()
EjectCard getEjectCard()
GasPumpedMsg getGasPumpedMsg()
PayMsg getPayMsg()
PrintReceipt getPrintReceipt()
PumpGasUnit getPumpGasUnit()
RejectMsg getRejectMsg()
ReturnCash getReturnCash()
SetInitialValues getSetInitialValues()
SetPayType getSetPayType()
SetPrice getSetPrice()
StoreCash getStoreCash()
StorePrices getStorePrices()
GP_2_Factory
DataStore ds
GP_2_Factory()
DataStore getDataStore()
CancelMsg getCancelMsg()
DisplayMenu getDisplayMenu()
EjectCard getEjectCard()
GasPumpedMsg getGasPumpedMsg()
PayMsg getPayMsg()
PrintReceipt getPrintReceipt()
PumpGasUnit getPumpGasUnit()
RejectMsg getRejectMsg()
ReturnCash getReturnCash()
SetInitialValues getSetInitialValues()
SetPayType getSetPayType()
SetPrice getSetPrice()
StoreCash getStoreCash()
StorePrices getStorePrices()

This part of the main class diagram uses **state pattern**.

Class State:

This class takes MDAEFSM and op as a parameters contains all the methods for both the gas pumps which are implemeted in different states based on the state diagram.

Here there are 8 states over all.

Class Start:

This class sets the initial state

Class S0:

This class activates the gas pump and changes the state from 0 to 1

Class S1:

This class starts the gas pump and changes the state to 2 or 3 based on the payment type.

Class S2:

This class either approves or rejects the payment and changes the state to 3 or 0.

Class S3:

This class provides operations to select gas or cancel the transaction or continue the operation and changes the state to 0 in case of cancel, 4 in case of continue or 3 for the select gas operation

Class S4:

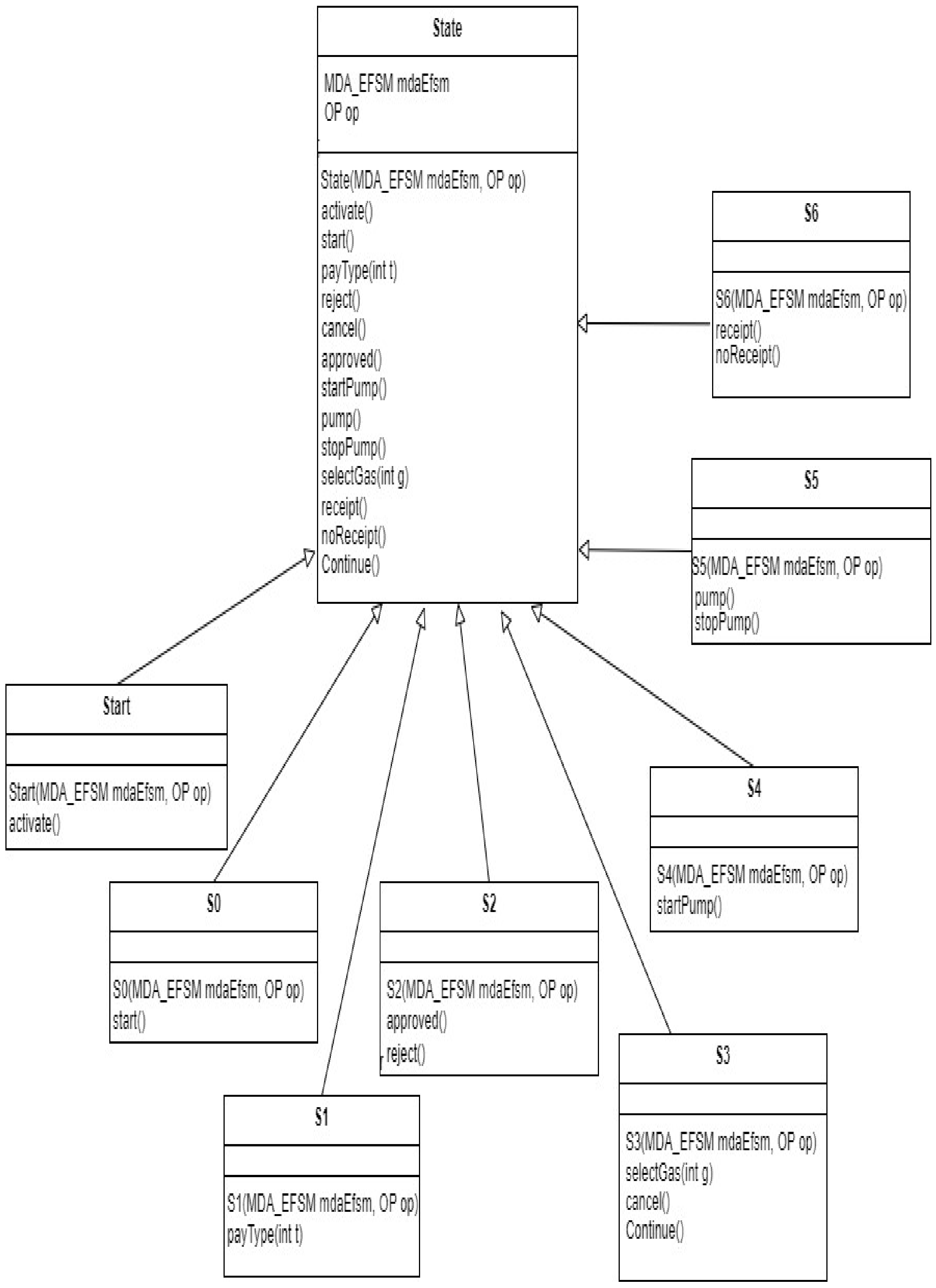
This class provides operations to start the gas pump. It changes the state from 4 to 5

Class S5:

This class provides operations to pump the gas or stop pump. In case of stop pump, this class changes the state to 6.

Class S6:

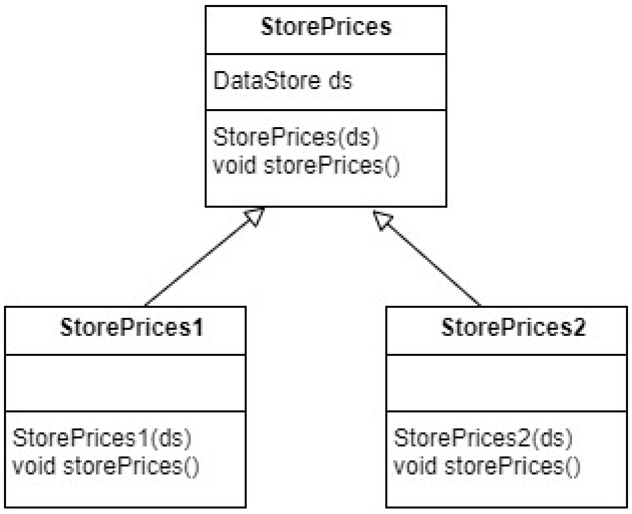
This class provides oprations to print receipt or no receipt. This changes the state to initial state


State
MDA_EFSM mdaEfsm
OP op
State(MDA_EFSM mdaEfsm, OP op)
activate()
start()
payType(int t)
reject()
cancel()
approved()
startPump()
pump()
stopPump()
selectGas(int g)
receipt()
noReceipt()
Continue()
S6
S6(MDA_EFSM mdaEfsm, OP op)
receipt()
noReceipt()
S5
S5(MDA_EFSM mdaEfsm, OP op)
pump()
stopPump()
Start
Start(MDA_EFSM mdaEfsm, OP op)
activate()
S4
S4(MDA_EFSM mdaEfsm, OP op)
startPump()
S0
S0(MDA_EFSM mdaEfsm, OP op)
start()
S2
S2(MDA_EFSM mdaEfsm, OP op)
approved()
reject()
S3
S3(MDA_EFSM mdaEfsm, OP op)
selectGas(int g)
cancel()
Continue()
S1
S1(MDA_EFSM mdaEfsm, OP op)
payType(int t)

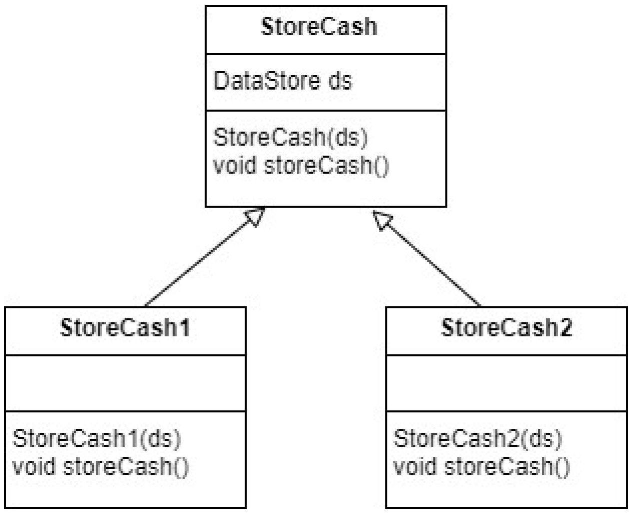
This below part of the main class diagram uses **strategy pattern**.

Here the data structures with different implementation for gaspump1 and gaspump2 are kept in different class, which dynamically sets the algorithm to use in case of runtime. In case of gas pumps, based on which object is created for the gas pump, those strategies are related to the particular gas pumps are executed

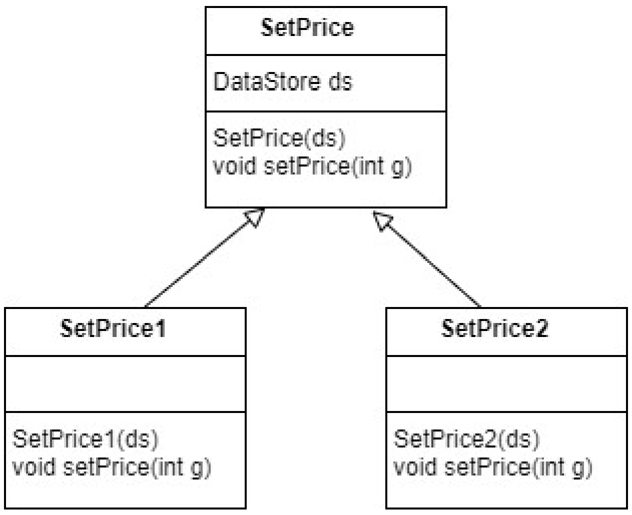
StorePrices is the abstract class for which are implemented by **StorePrices1** and **StorePrices2**. These strategies store the price of then gas entered by the user.



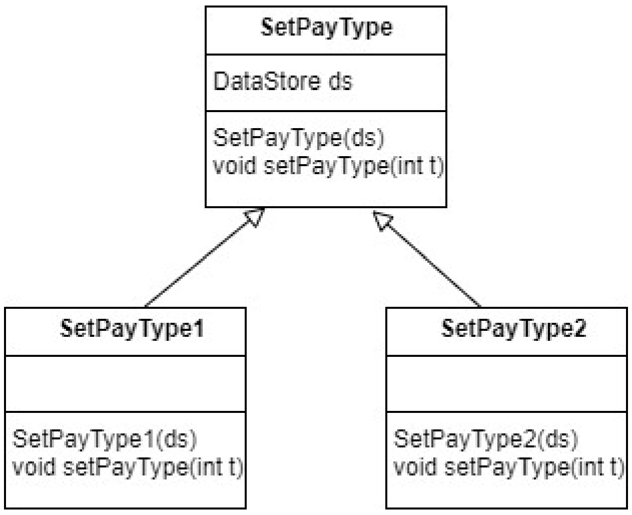
StoreCash is the abstract class for which are implemented by **StoreCash1** and **StoreCash2**. These strategies store the cash inserted into the machine by customer



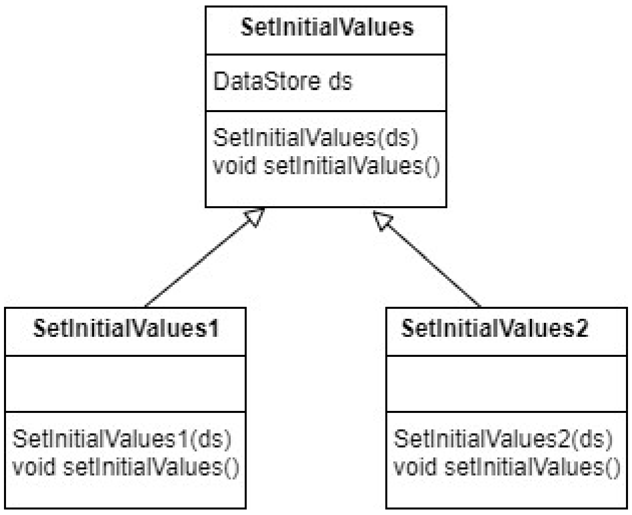
SetPrice is the abstract class for which are implemented by **SetPrice1** and **SetPrice2**. These strategies are used to set the price of the gas



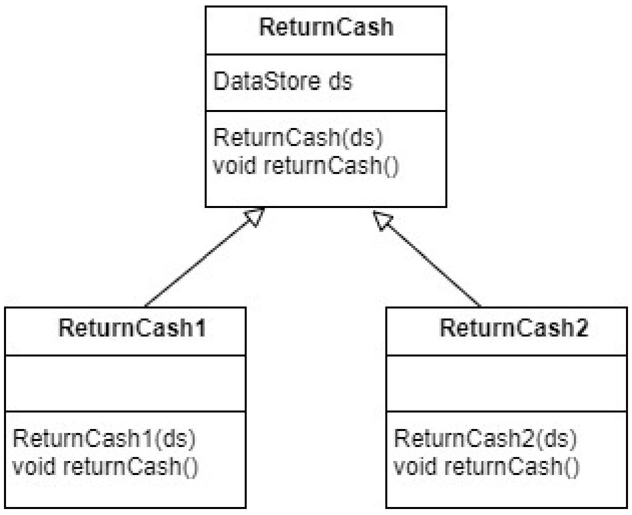
SetPayType is the abstract class for which are implemented by **SetPaytype1** and **SetPayType2**. These strategies set the type of payment.



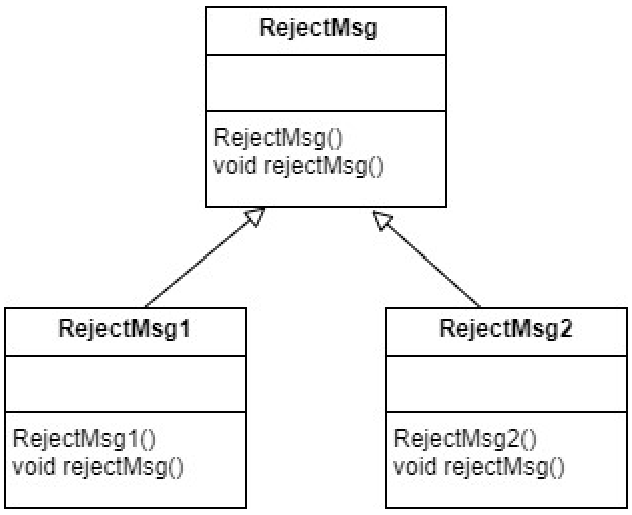
SetInitialValues is the abstract class for which are implemented by **SetInitialValues1** and **SetInitialValues2**. These strategies set the initial values of the gaspump.



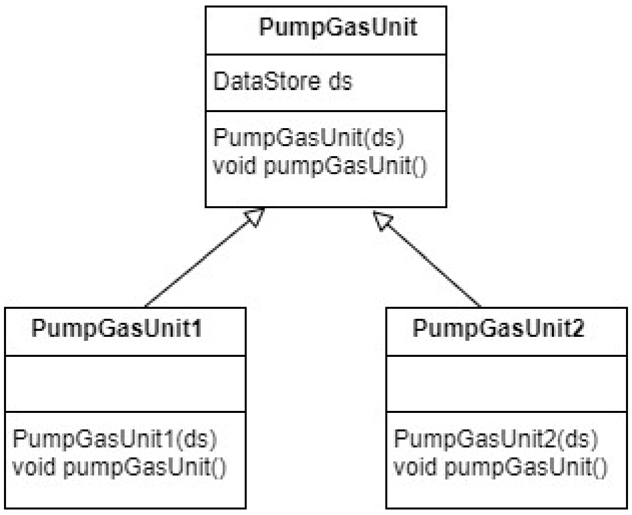
ReturnCash is the abstract class for which are implemented by **ReturnCash1** and **ReturnCash2**. These strategies return the cash to the customer.



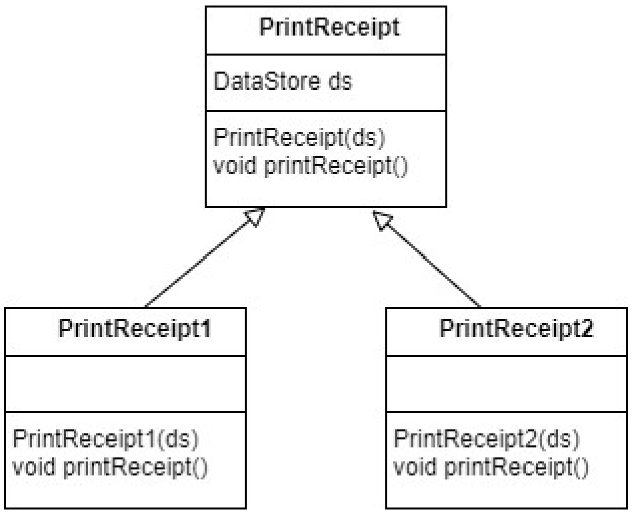
RejectMsg is the abstract class for which are implemented by **RejectMsg1** and **RejectMsg2**. These strategies reject and display a message when payment is failed.



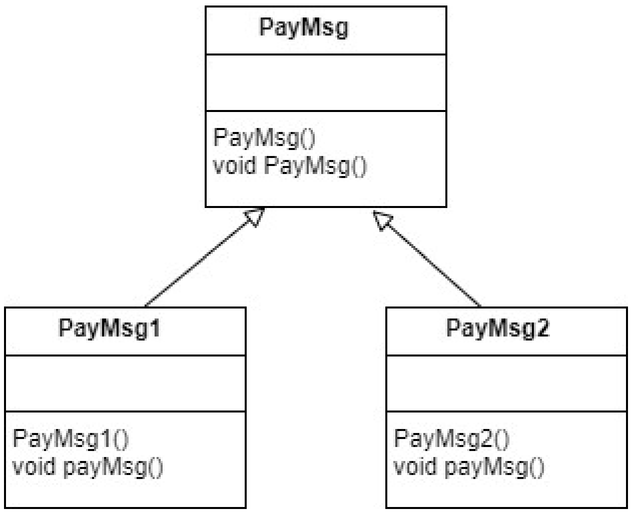
PumpGasUnit is the abstract class for which are implemented by **PumpGasUnit1** and **PumpGasUnit2**. These strategies store the information about the amount of gas pumped and total price incurred.



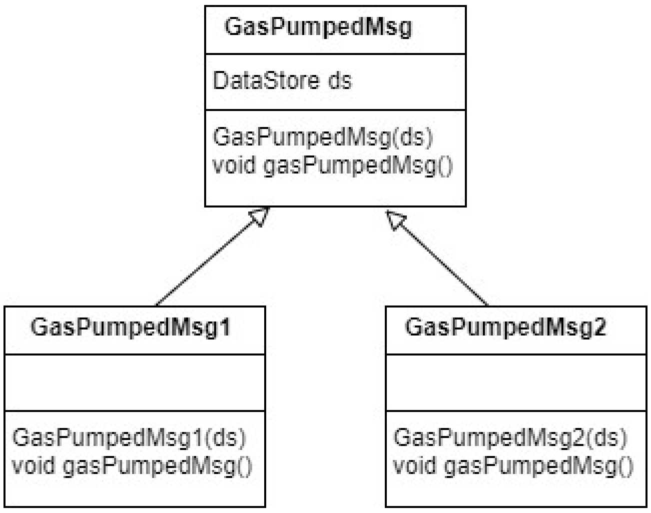
PrintReceipt is the abstract class for which are implemented by **PrintReceipt1** and **PrintReceipt2**. These strategies prints the receipt.



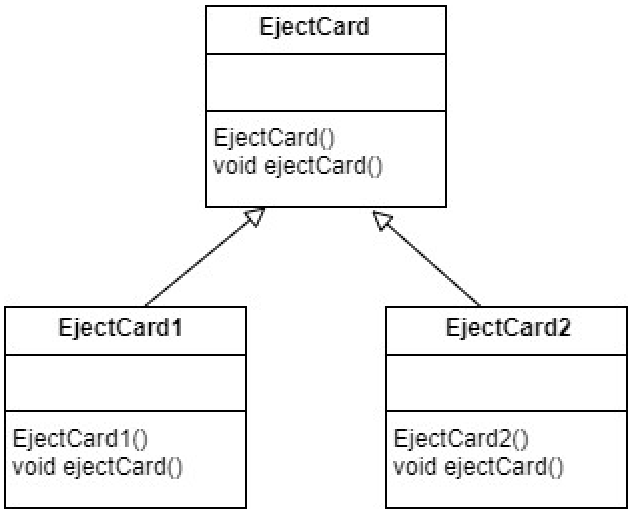
PayMsg is the abstract class for which are implemented by **PayMsg1** and **PayMsg2**. These strategies are used to display the payment message for the gaspumps



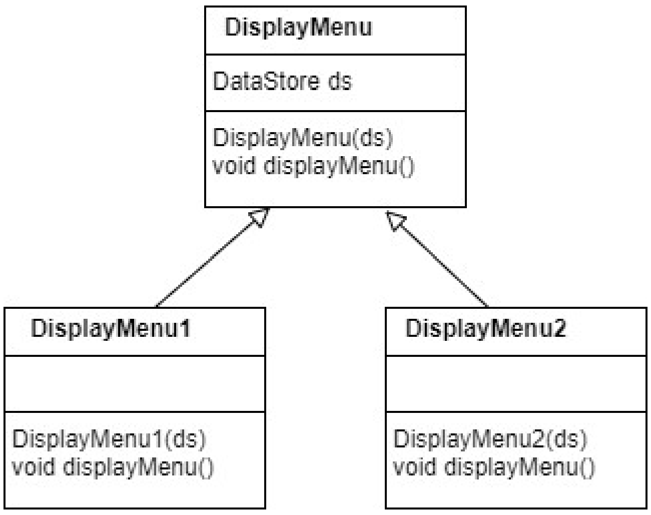
GasPumpedMsg is the abstract class for which are implemented by **GasPumpedMsg1** and **GasPumpedMsg2**. These strategies display the information about the amount of gas pumped.



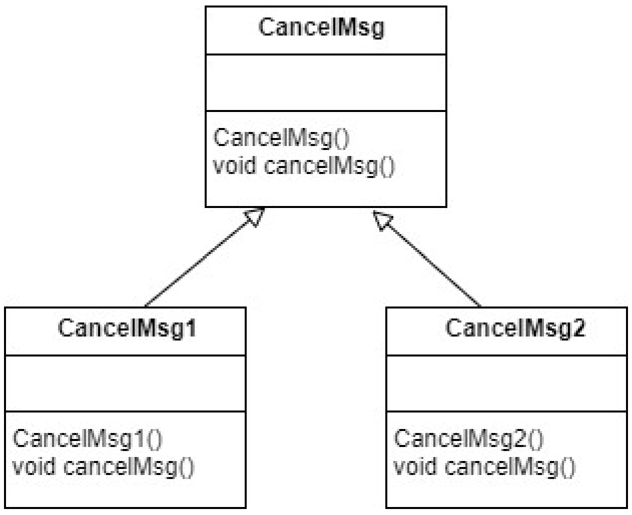
EjectCard is the abstract class for which are implemented by **EjectCard1** and **EjectCard2**. These strategies eject the card in case of payment completion or failure



DisplayMenu is the abstract class for which are implemented by **DisplayMenu1** and **DisplayMenu2**. These strategies display the menu for the gas pumps.



CancelMsg is the abstract class for which are implemented by **CancelMsg1** and **CancelMsg2**. These strategies cancel the transaction and displays a message

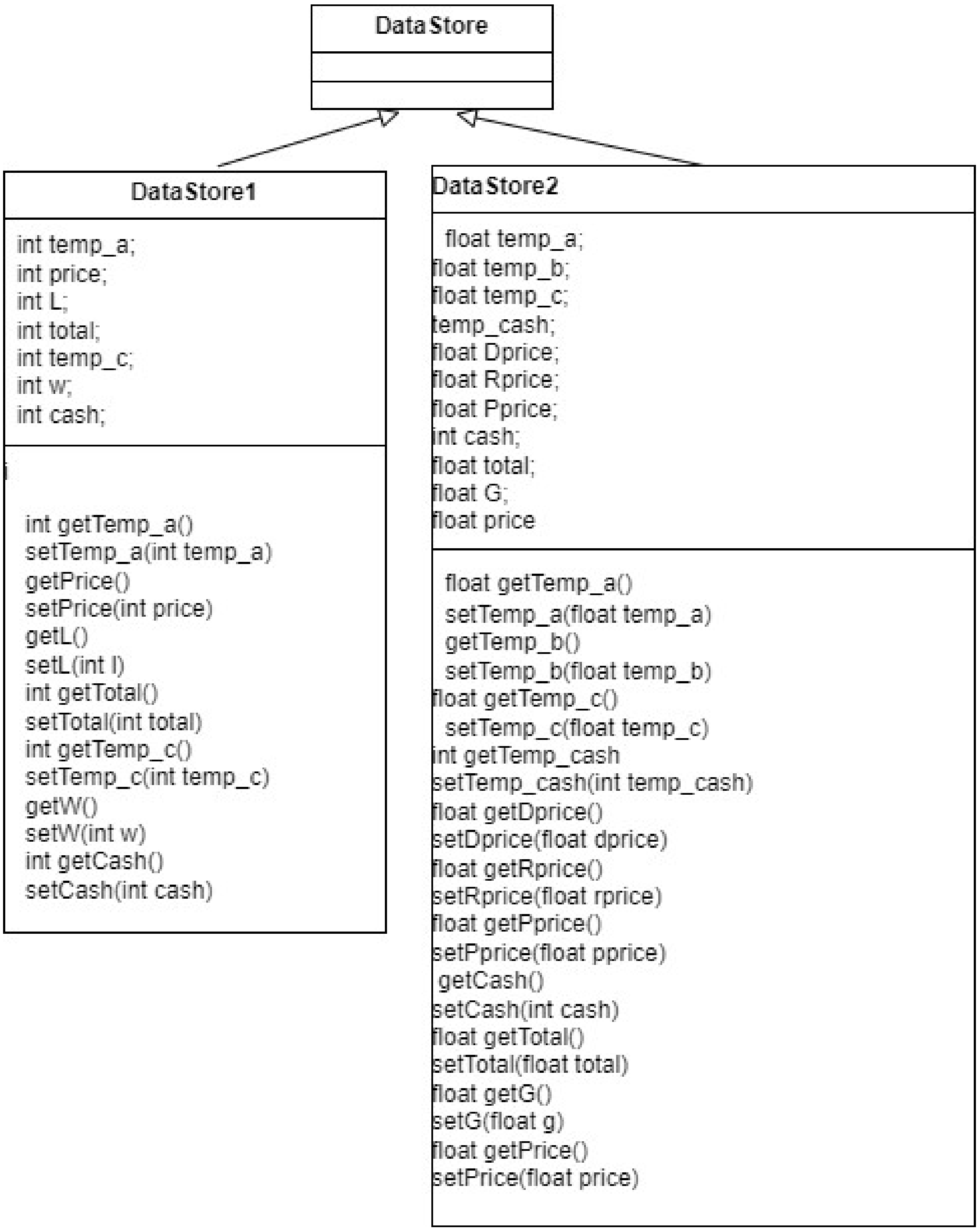


Class **DataStore**: is an abstract class for the data stores for the gaspump 1 and gaspump2.

DataStore1 is the data class for the GP1.

DaraStore2 is the data class for the GP2.

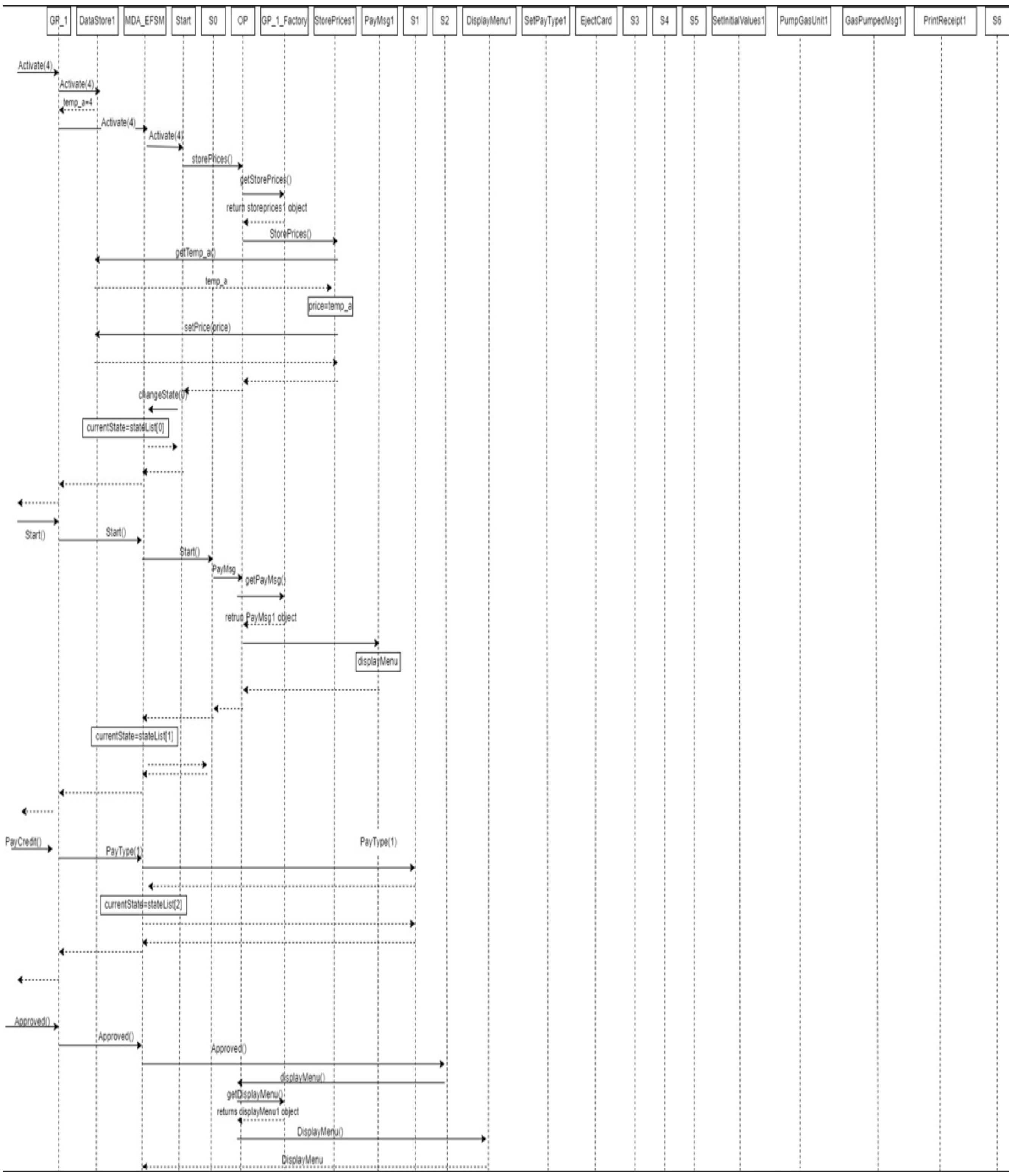
These classes consist of all the data required for GP1 and GP2.

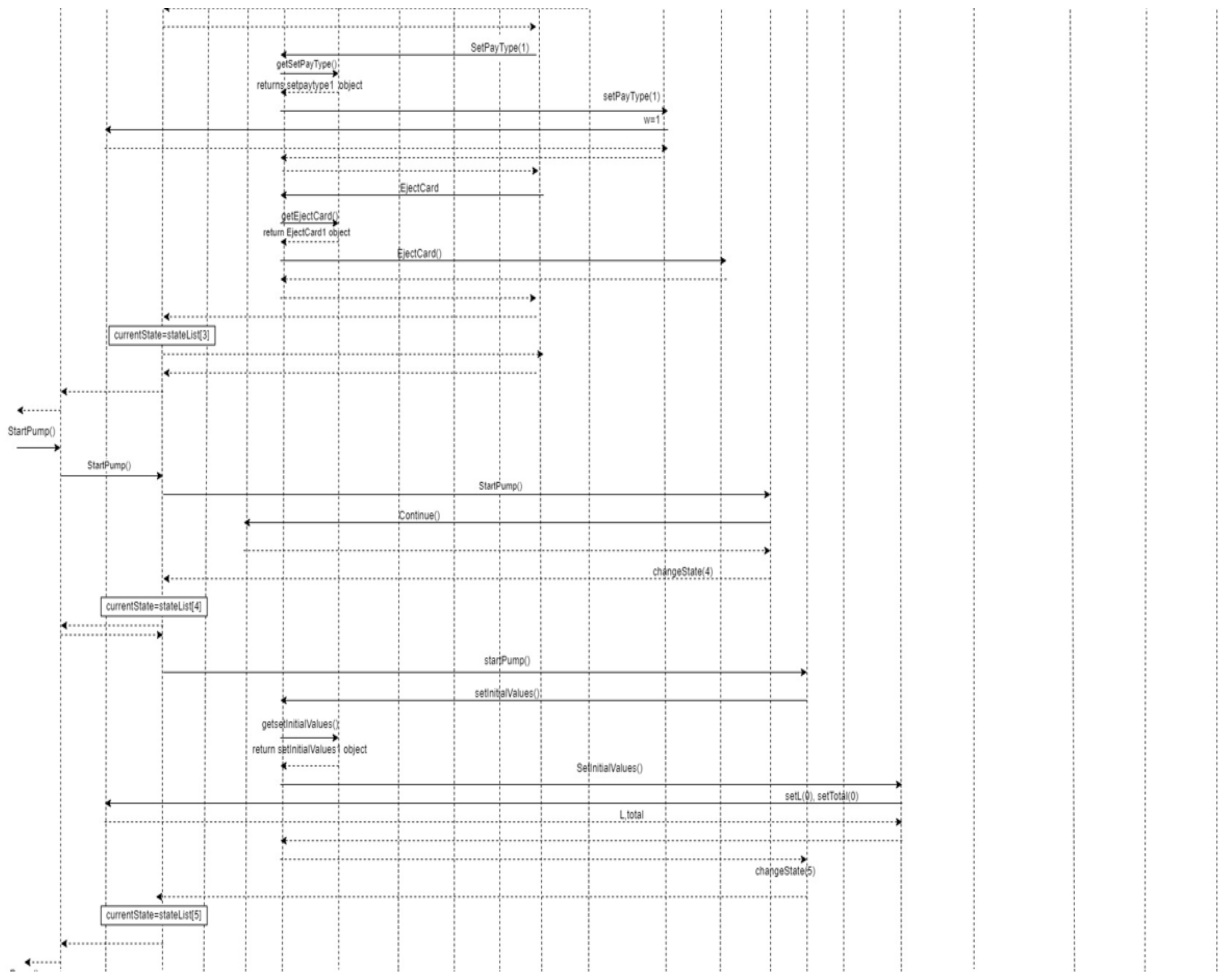


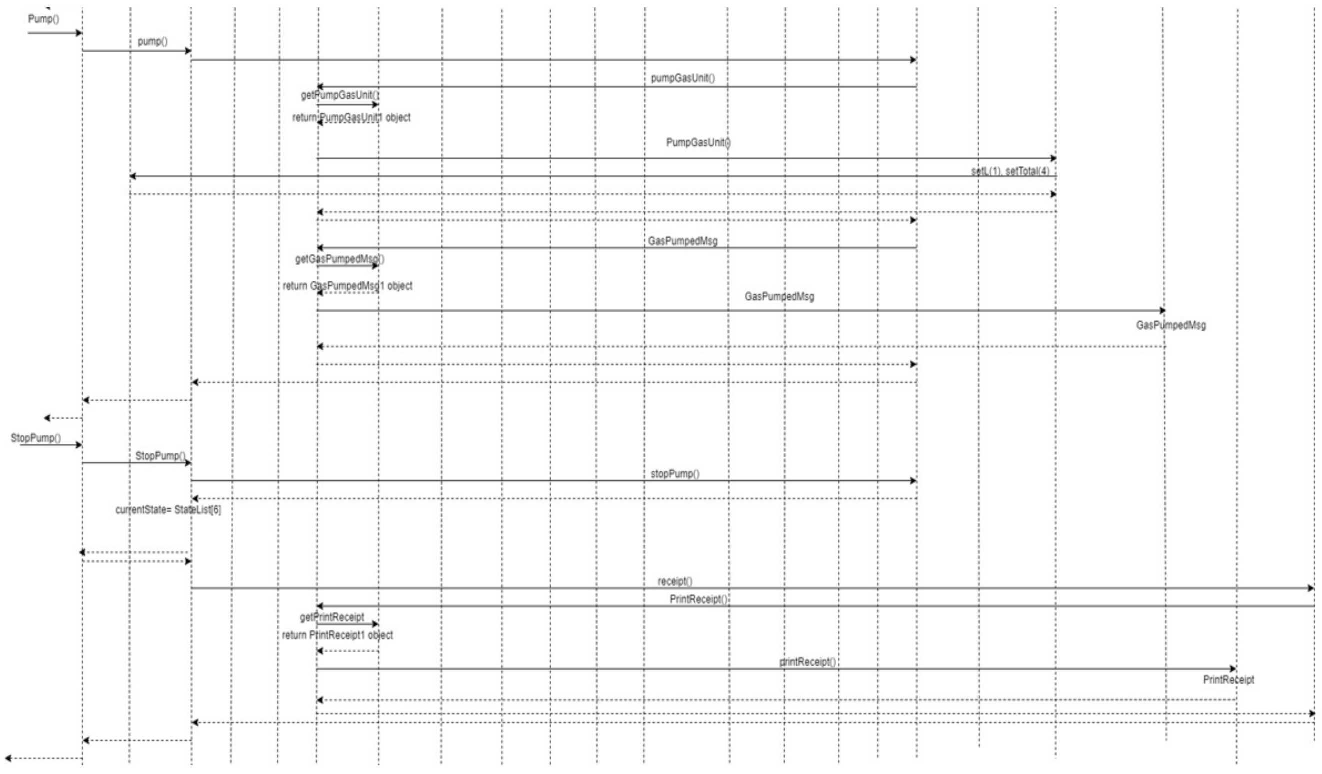
4) Sequence diagrams:

a) Scenario-I should show how one liter of gas is disposed in the Gas Pump GP-1 component, i.e., the following sequence of operations is issued:

Activate(4), Start(), PayCredit(), Approved(), StartPump(), Pump(), StopPump()

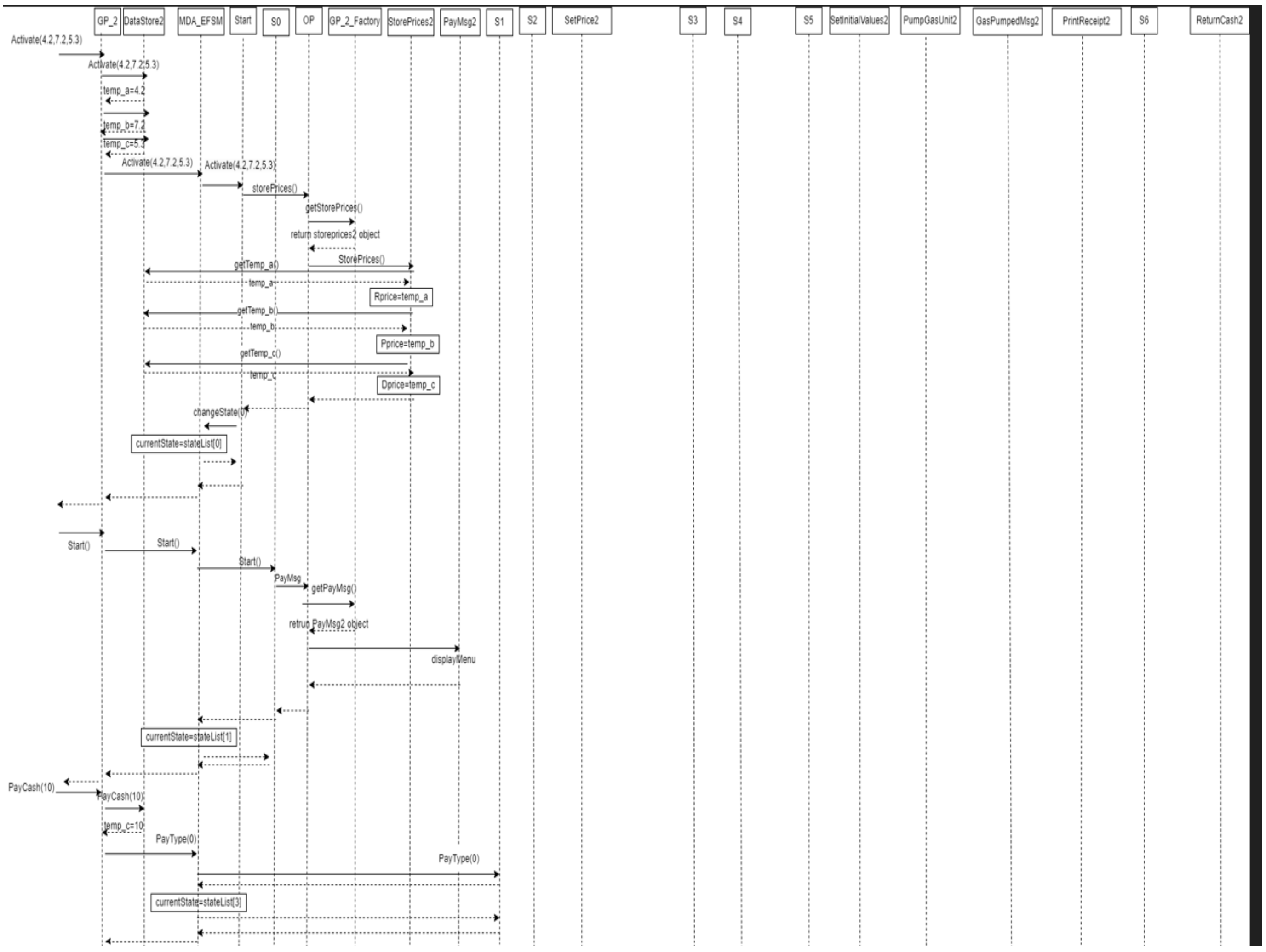


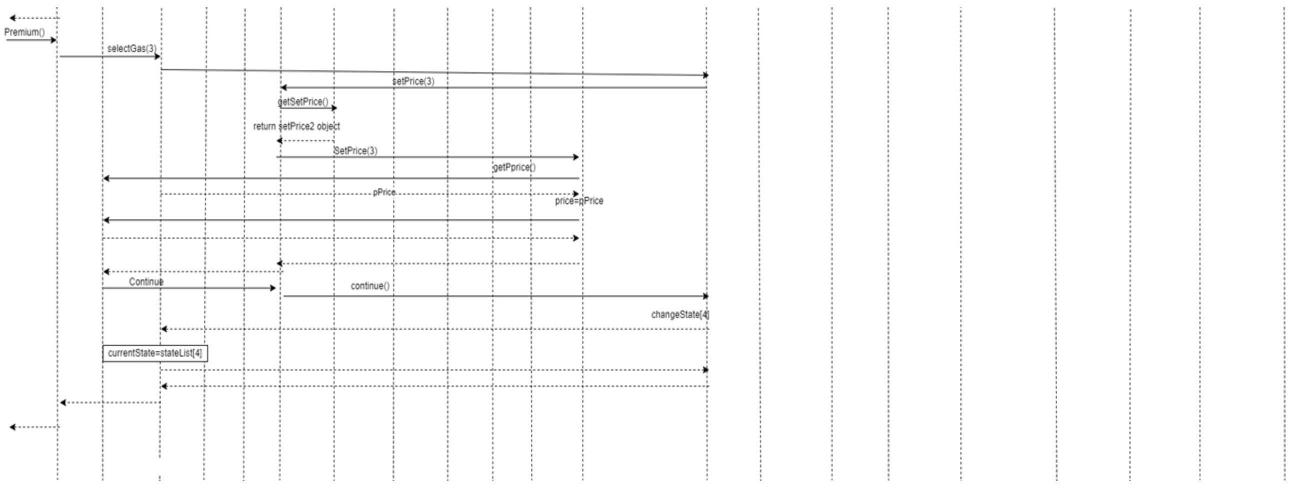

SetPayType(1)
getSetPayType()
returns setpaytype1 object
setPayType(1)
w=1
EjectCard
getEjectCard()
return EjectCard1 object
EjectCard()
currentState=stateList[3]
StartPump()
StartPump()
StartPump()
Continue()
changeState(4)
currentState=stateList[4]
startPump()
setInitialValues()
getsetInitialValues()
return setInitialValues1 object
SetInitialValues()
setL(0), setTotal(0)
L,total
changeState(5)
currentState=stateList[5]

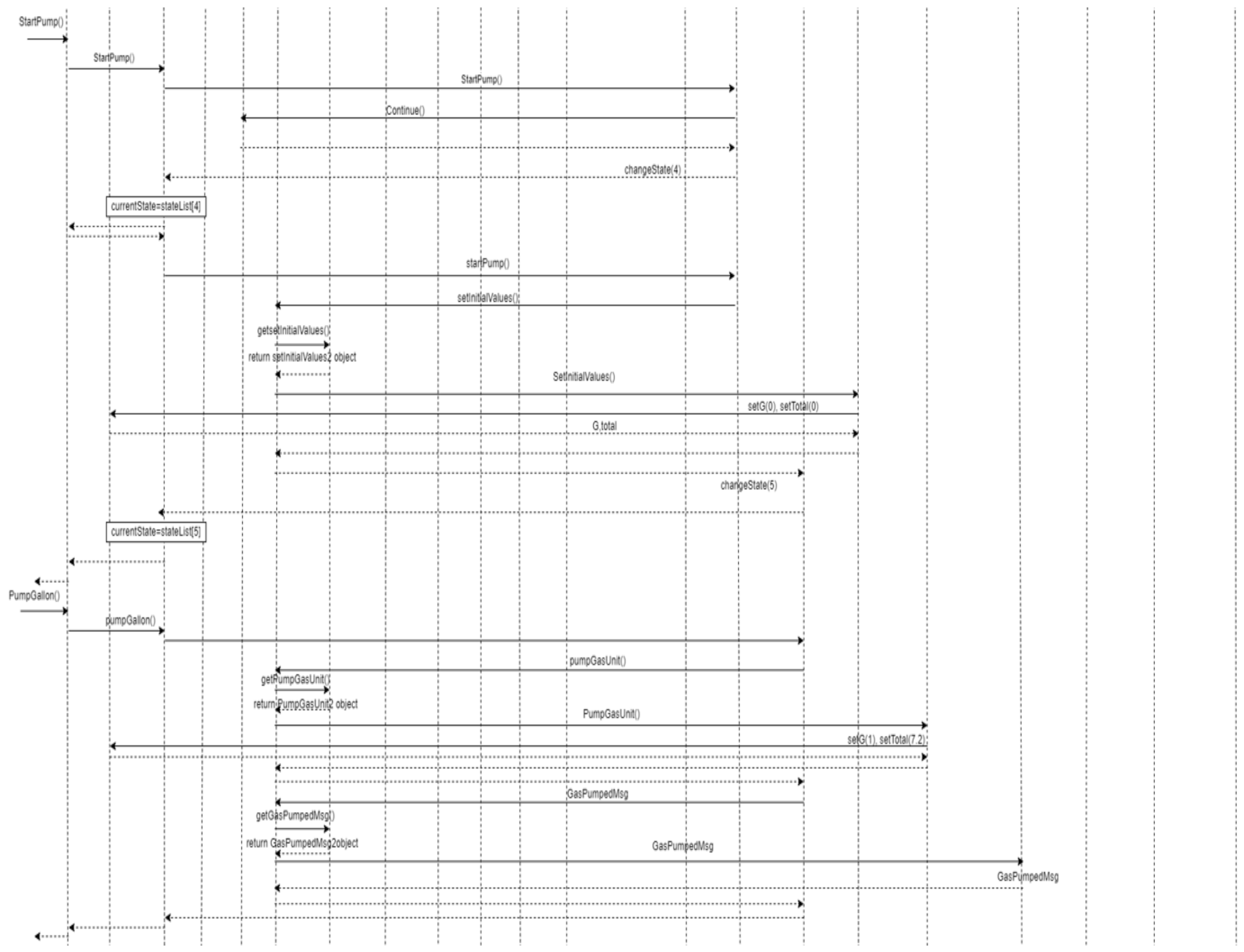

Pump()
pump()
pumpGasUnit()
getPumpGasUnit()
return PumpGasUnit1 object
PumpGasUnit()
setL(1), setTotal(4)
GasPumpedMsg
getGasPumpedMsg()
return GasPumpedMsg1 object
GasPumpedMsg
GasPumpedMsg
StopPump()
StopPump()
stopPump()
currentState= StateList[6]
receipt()
PrintReceipt()
getPrintReceipt
return PrintReceipt1 object
printReceipt()
PrintReceipt

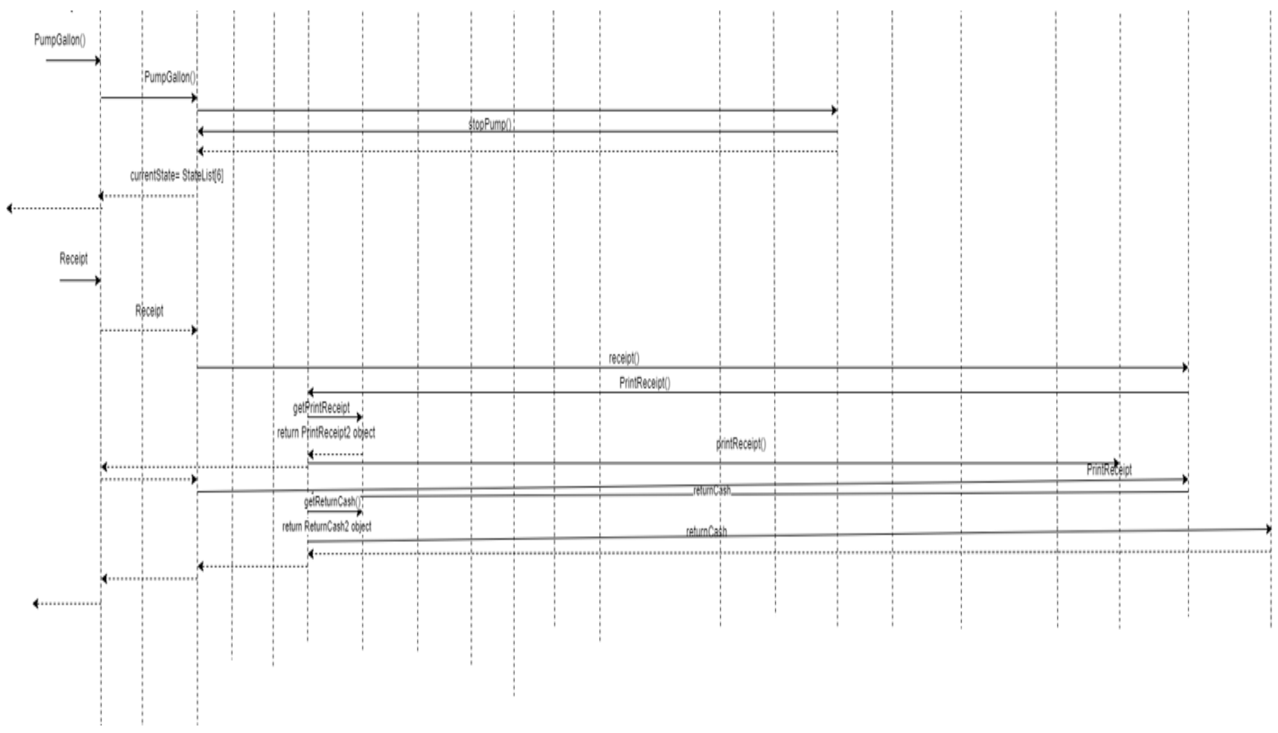
b)Scenario-II should show how one gallon of Premium gas is disposed in the Gas Pump GP-2 component, i.e., the following sequence of operations is issued:

Activate(4.2, 7.2, 5.3), Start(), PayCash(10), Premium(), StartPump(), PumpGallon(), PumpGallon(), Receipt()






StartPump()
StartPump()
StartPump()
Continue()
changeState(4)
currentState=stateList[4]
startPump()
setInitialValues()
getsetInitialValues()
return setInitialValues2 object
SetInitialValues()
setG(0), setTotal(0)
G,total
changeState(5)
currentState=stateList[5]
PumpGallon()
pumpGallon()
pumpGasUnit()
getPumpGasUnit()
return PumpGasUnit2 object
PumpGasUnit()
setG(1), setTotal(7.2)
GasPumpedMsg
getGasPumpedMsg()
return GasPumpedMsg2object
GasPumpedMsg
GasPumpedMsg

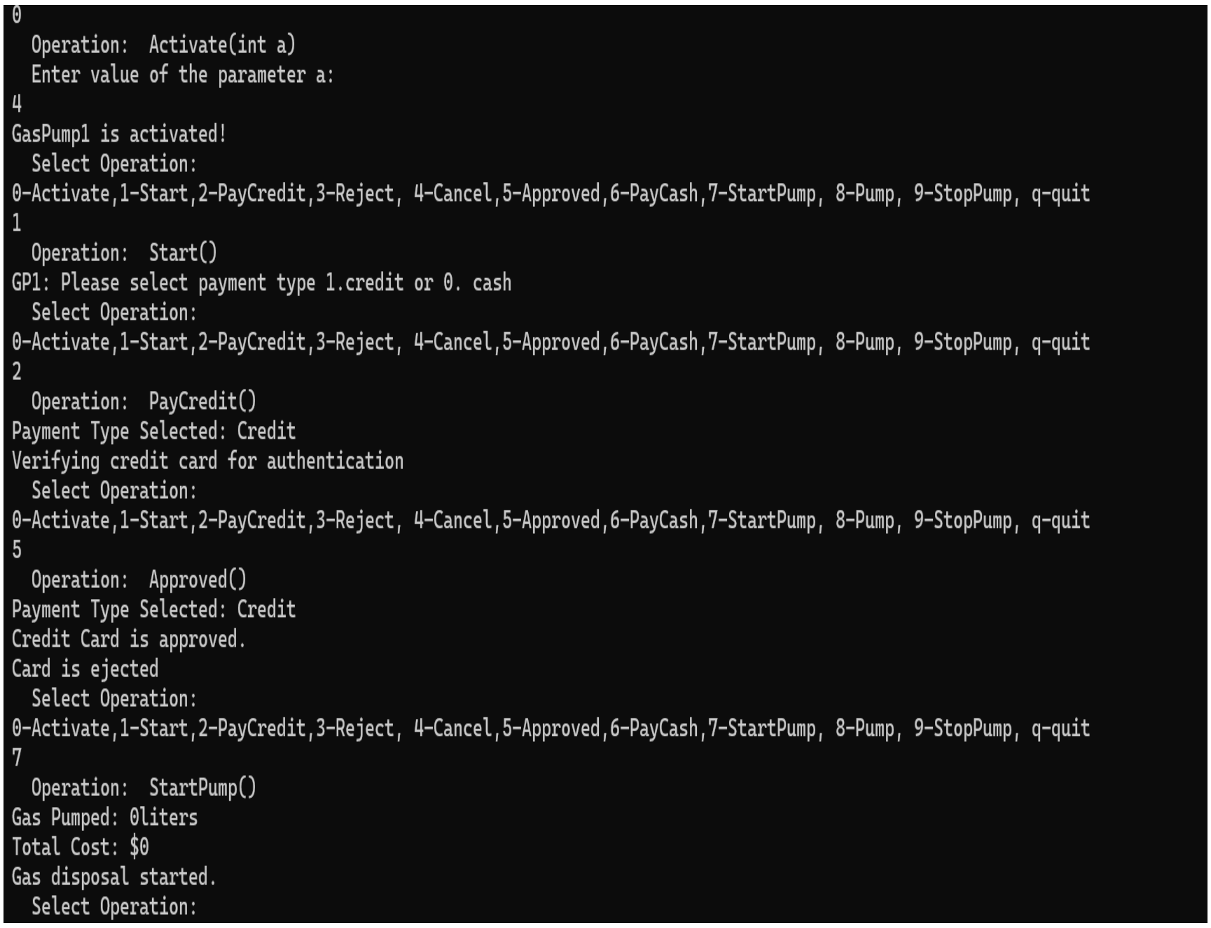

PumpGallon()
PumpGallon()
stopPump()
currentState= StateList[6]
Receipt
Receipt
receipt()
PrintReceipt()
getPrintReceipt
return PrintReceipt2 object
printReceipt()
PrintReceipt
returnCash
getReturnCash()
return ReturnCash2 object
returnCash

Testcases Execution:

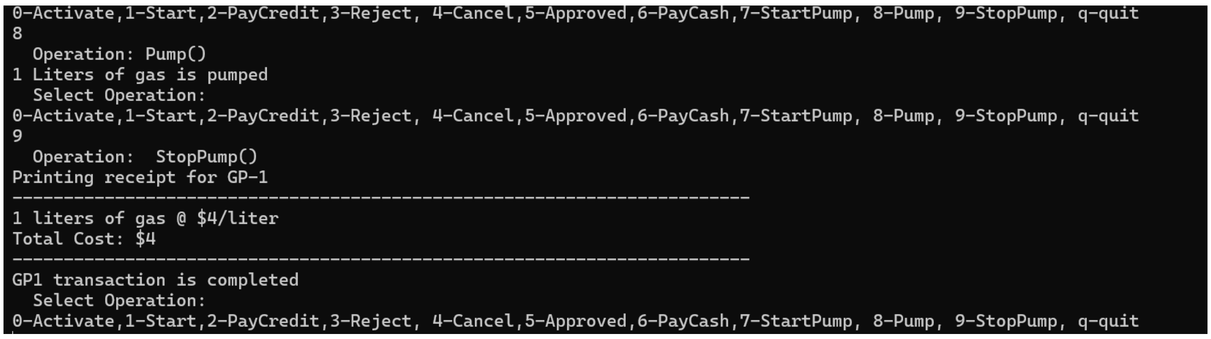
Gas Pump GP-1

Test #1

Activate(4), Start(), PayCredit(), Approved(), StartPump(), Pump(), StopPump() Expected result at the last operation: Total=$4



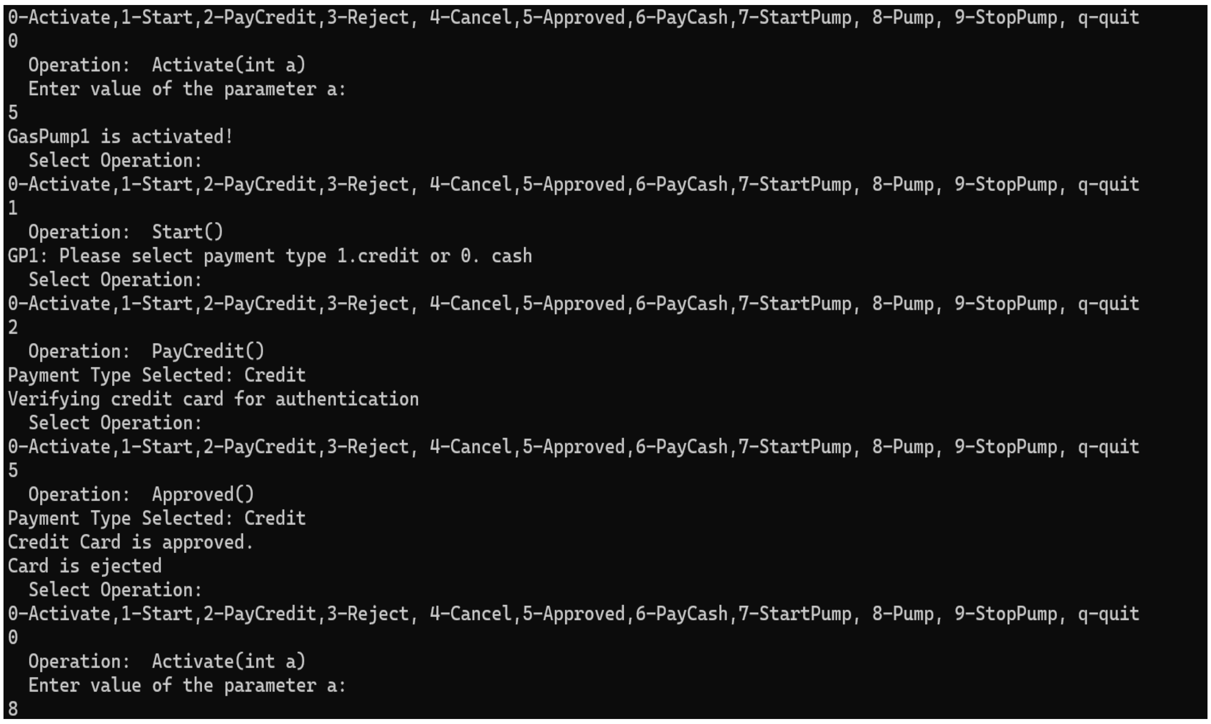
```
0
  Operation:  Activate(int a)
  Enter value of the parameter a:
4
GasPump1 is activated!
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
1
  Operation:  Start()
GP1: Please select payment type 1.credit or 0. cash
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
2
  Operation:  PayCredit()
Payment Type Selected: Credit
Verifying credit card for authentication
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
5
  Operation:  Approved()
Payment Type Selected: Credit
Credit Card is approved.
Card is ejected
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
7
  Operation:  StartPump()
Gas Pumped: 0liters
Total Cost: $0
Gas disposal started.
  Select Operation:
```



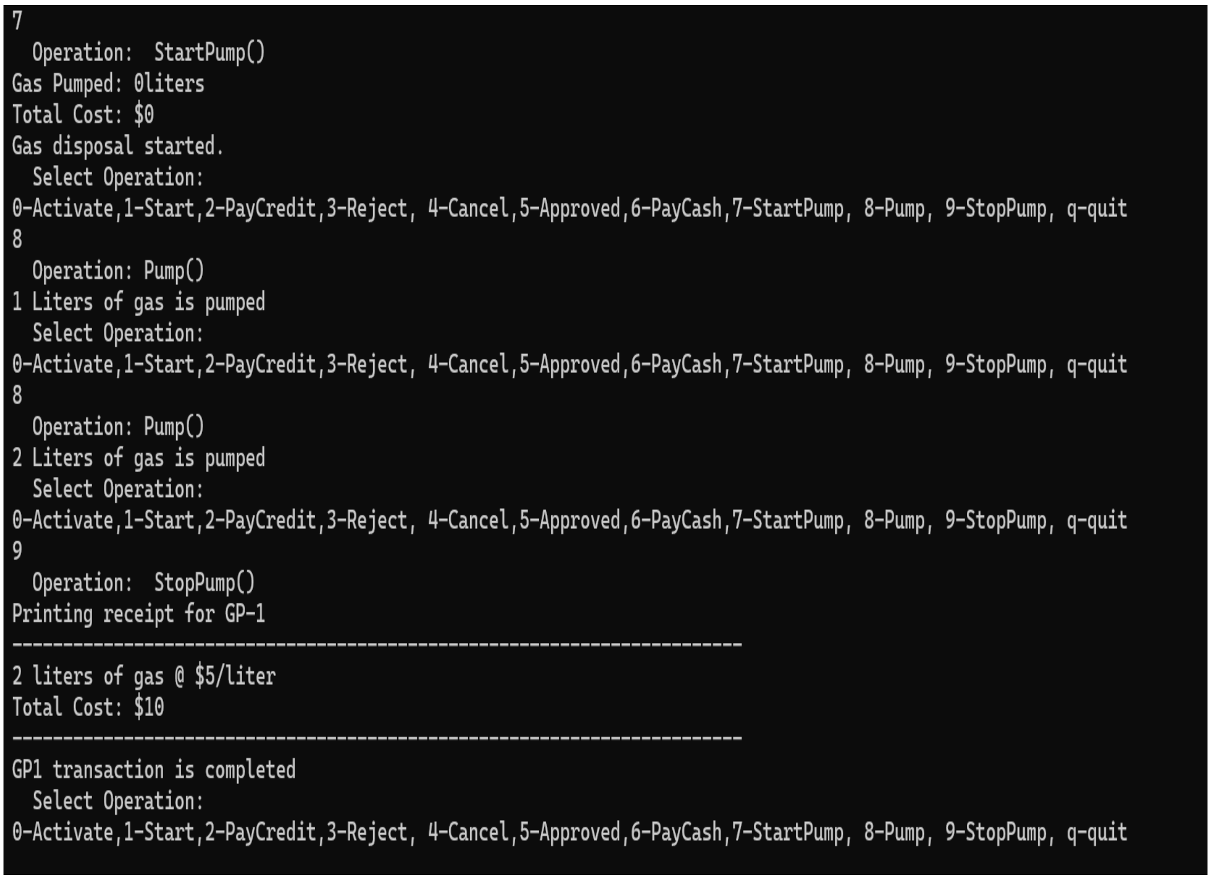
```
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
8
  Operation: Pump()
1 Liters of gas is pumped
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
9
  Operation:  StopPump()
Printing receipt for GP-1
------------------------------------------------------------------------
1 liters of gas @ $4/liter
Total Cost: $4
------------------------------------------------------------------------
GP1 transaction is completed
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
```

Test #2

Activate(5), Start(), PayCredit(), Approved(), Activate(8), StartPump(), Pump(), Pump(), StopPump()
Expected result at the last operation: Total=$10



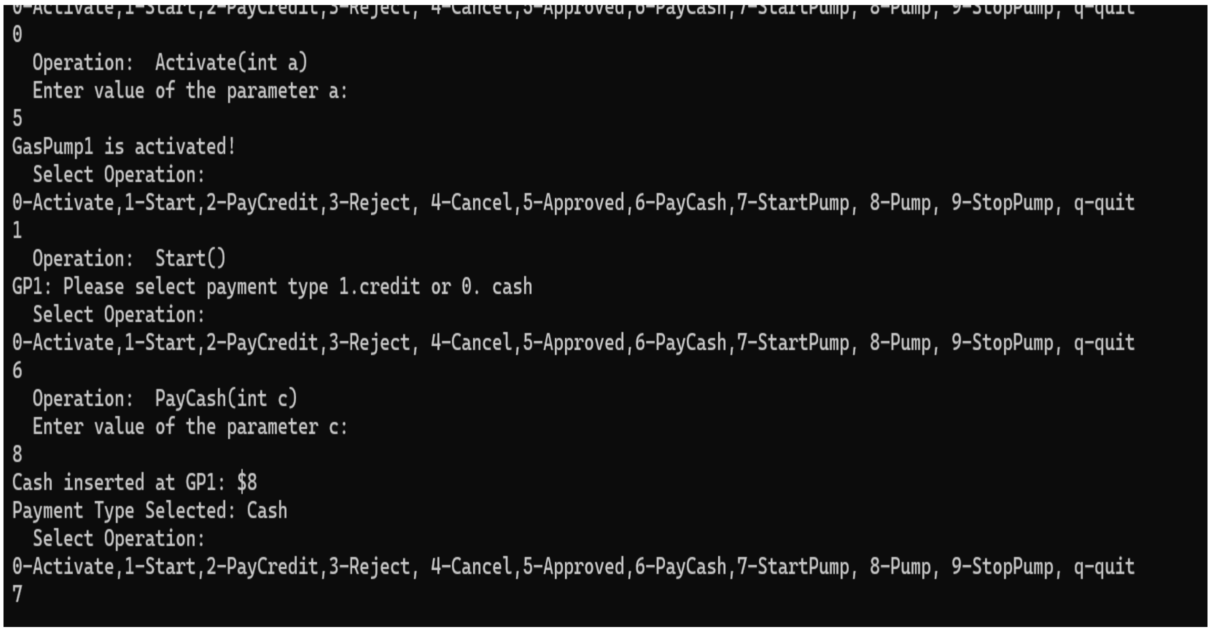
```
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
0
  Operation:  Activate(int a)
  Enter value of the parameter a:
5
GasPump1 is activated!
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
1
  Operation:  Start()
GP1: Please select payment type 1.credit or 0. cash
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
2
  Operation:  PayCredit()
Payment Type Selected: Credit
Verifying credit card for authentication
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
5
  Operation:  Approved()
Payment Type Selected: Credit
Credit Card is approved.
Card is ejected
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
0
  Operation:  Activate(int a)
  Enter value of the parameter a:
8
```



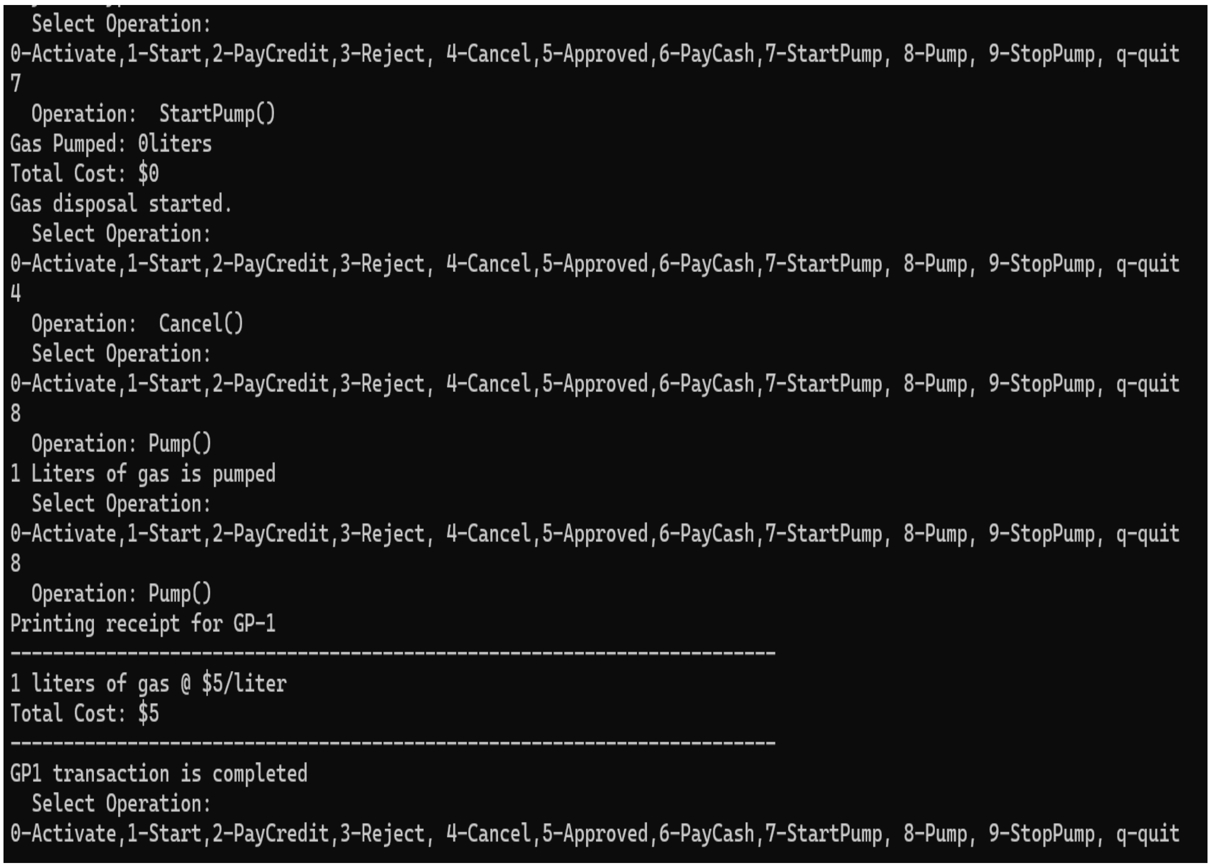
```
7
  Operation:  StartPump()
Gas Pumped: 0liters
Total Cost: $0
Gas disposal started.
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
8
  Operation: Pump()
1 Liters of gas is pumped
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
8
  Operation: Pump()
2 Liters of gas is pumped
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
9
  Operation:  StopPump()
Printing receipt for GP-1
-----------------------------------------------------------------------
2 liters of gas @ $5/liter
Total Cost: $10
-----------------------------------------------------------------------
GP1 transaction is completed
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
```

Test #3

Activate(5), Start(), PayCash(8), StartPump(), Cancel(), Pump(), Pump() Expected result at the last operation: Total=$5



```
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
0
  Operation:  Activate(int a)
  Enter value of the parameter a:
5
GasPump1 is activated!
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
1
  Operation:  Start()
GP1: Please select payment type 1.credit or 0. cash
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
6
  Operation:  PayCash(int c)
  Enter value of the parameter c:
8
Cash inserted at GP1: $8
Payment Type Selected: Cash
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
7
```

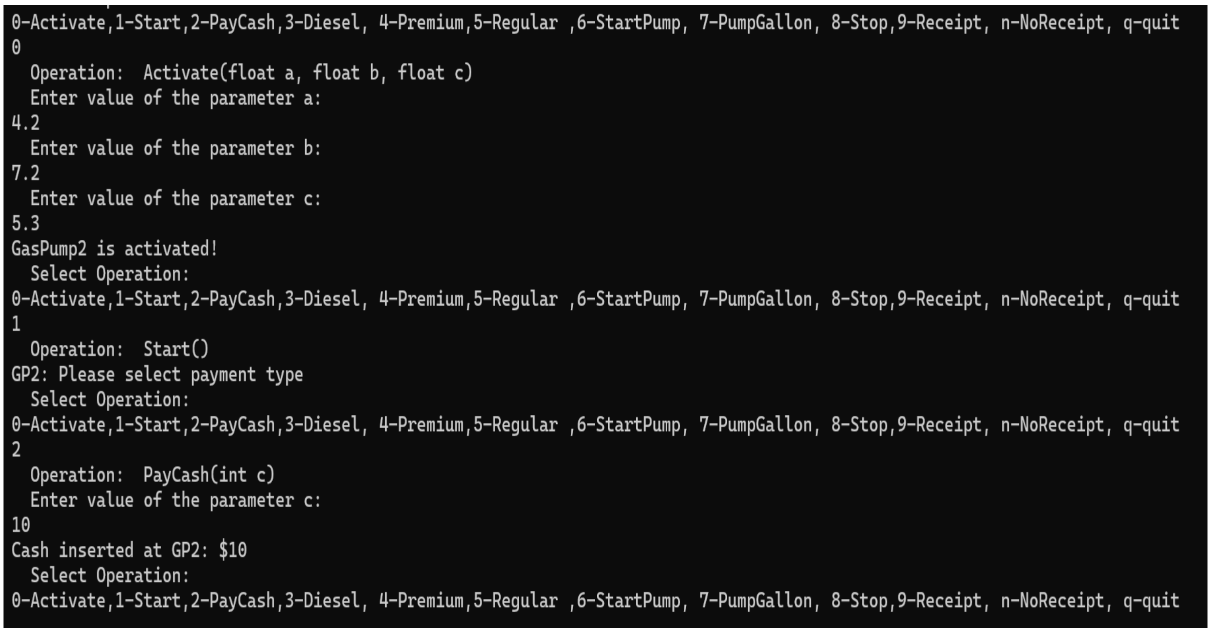


```
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
7
  Operation:  StartPump()
Gas Pumped: 0liters
Total Cost: $0
Gas disposal started.
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
4
  Operation:  Cancel()
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
8
  Operation: Pump()
1 Liters of gas is pumped
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
8
  Operation: Pump()
Printing receipt for GP-1
--------------------------------------------------------------------------
1 liters of gas @ $5/liter
Total Cost: $5
--------------------------------------------------------------------------
GP1 transaction is completed
  Select Operation:
0-Activate,1-Start,2-PayCredit,3-Reject, 4-Cancel,5-Approved,6-PayCash,7-StartPump, 8-Pump, 9-StopPump, q-quit
```

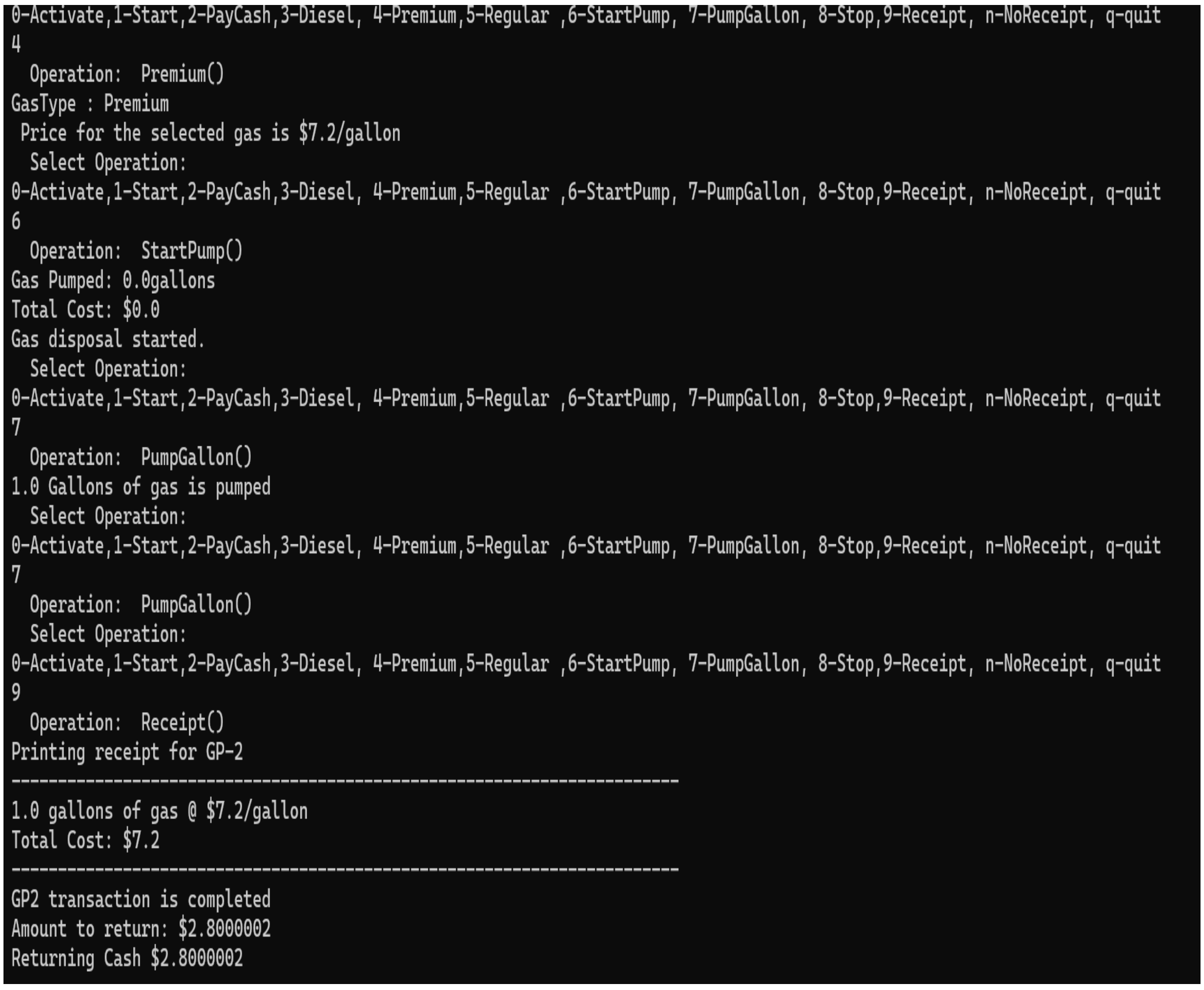
Gas Pump GP-2

Test #1

Activate(4.2, 7.2, 5.3), Start(), PayCash(10), Premium(), StartPump(), PumpGallon(), PumpGallon(), Receipt() Expected result at the last operation: Total=$7.2; Returned cash=$2.8



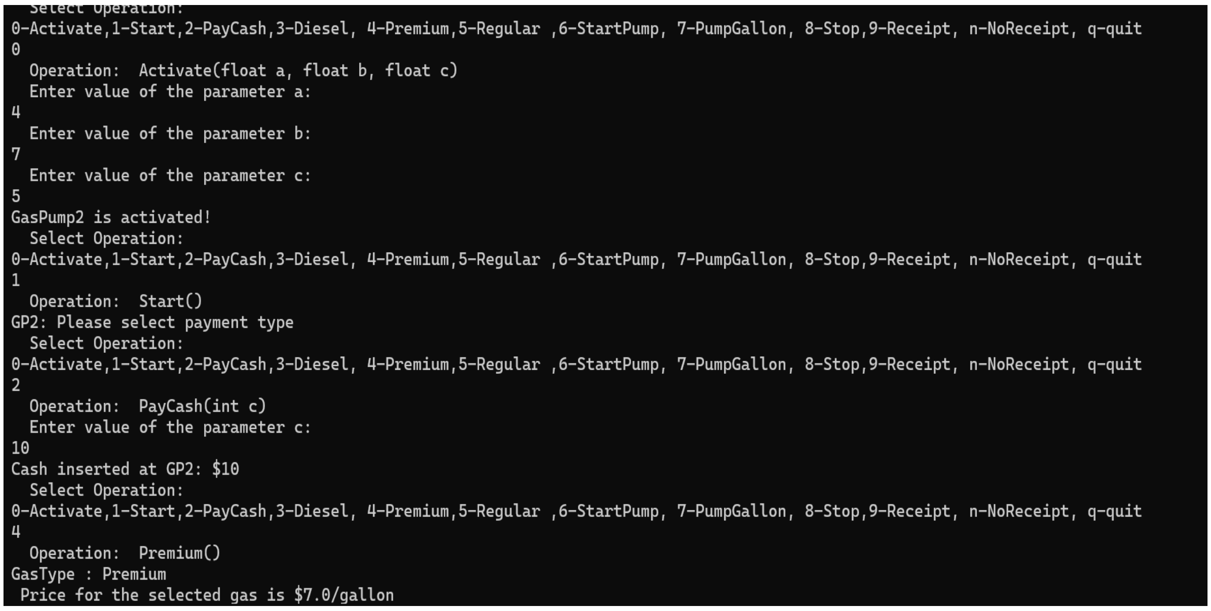
```
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
0
  Operation:  Activate(float a, float b, float c)
  Enter value of the parameter a:
4.2
  Enter value of the parameter b:
7.2
  Enter value of the parameter c:
5.3
GasPump2 is activated!
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
1
  Operation:  Start()
GP2: Please select payment type
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
2
  Operation:  PayCash(int c)
  Enter value of the parameter c:
10
Cash inserted at GP2: $10
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
```



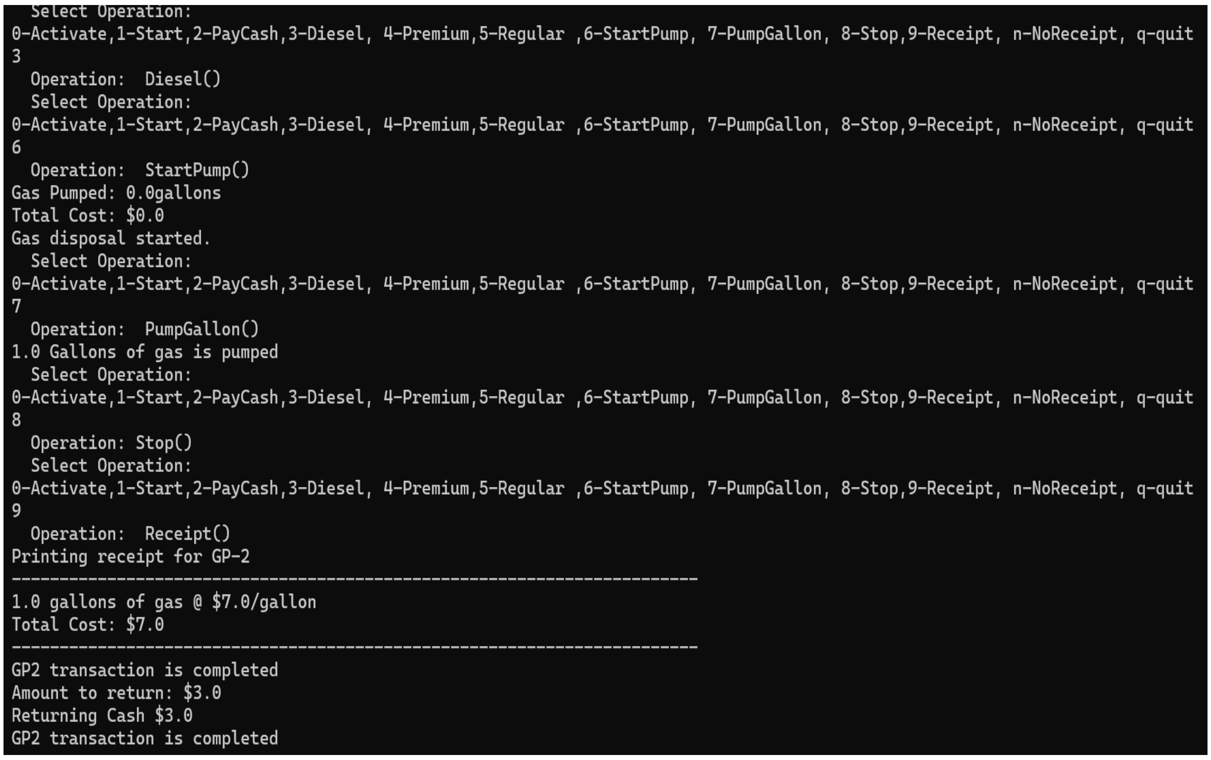
```
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
4
  Operation:  Premium()
GasType : Premium
 Price for the selected gas is $7.2/gallon
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
6
  Operation:  StartPump()
Gas Pumped: 0.0gallons
Total Cost: $0.0
Gas disposal started.
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
7
  Operation:  PumpGallon()
1.0 Gallons of gas is pumped
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
7
  Operation:  PumpGallon()
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
9
  Operation:  Receipt()
Printing receipt for GP-2
-----------------------------------------------------------------------
1.0 gallons of gas @ $7.2/gallon
Total Cost: $7.2
-----------------------------------------------------------------------
GP2 transaction is completed
Amount to return: $2.8000002
Returning Cash $2.8000002
```

Test #2

Activate(4, 7, 5), Start(), PayCash(10), Premium(), Diesel(), StartPump(), PumpGallon(), Stop(), Receipt() Expected result at the last operation: Total=$7; Returned cash=$3



```
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
0
  Operation:  Activate(float a, float b, float c)
  Enter value of the parameter a:
4
  Enter value of the parameter b:
7
  Enter value of the parameter c:
5
GasPump2 is activated!
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
1
  Operation:  Start()
GP2: Please select payment type
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
2
  Operation:  PayCash(int c)
  Enter value of the parameter c:
10
Cash inserted at GP2: $10
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
4
  Operation:  Premium()
GasType : Premium
 Price for the selected gas is $7.0/gallon
```



```
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
3
  Operation:  Diesel()
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
6
  Operation:  StartPump()
Gas Pumped: 0.0gallons
Total Cost: $0.0
Gas disposal started.
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
7
  Operation:  PumpGallon()
1.0 Gallons of gas is pumped
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
8
  Operation: Stop()
  Select Operation:
0-Activate,1-Start,2-PayCash,3-Diesel, 4-Premium,5-Regular ,6-StartPump, 7-PumpGallon, 8-Stop,9-Receipt, n-NoReceipt, q-quit
9
  Operation:  Receipt()
Printing receipt for GP-2
------------------------------------------------------------------------
1.0 gallons of gas @ $7.0/gallon
Total Cost: $7.0
------------------------------------------------------------------------
GP2 transaction is completed
Amount to return: $3.0
Returning Cash $3.0
GP2 transaction is completed
```