Software System Architecture

"Project Report on Gas Pump"

Submitted To:
Prof. Bogdan Korel

By
Parva Udaykumar Bhayani
A20396637

Spring 2018
Illinois Institute of Technology

1. MDA-EFSM model for the GasPump components:

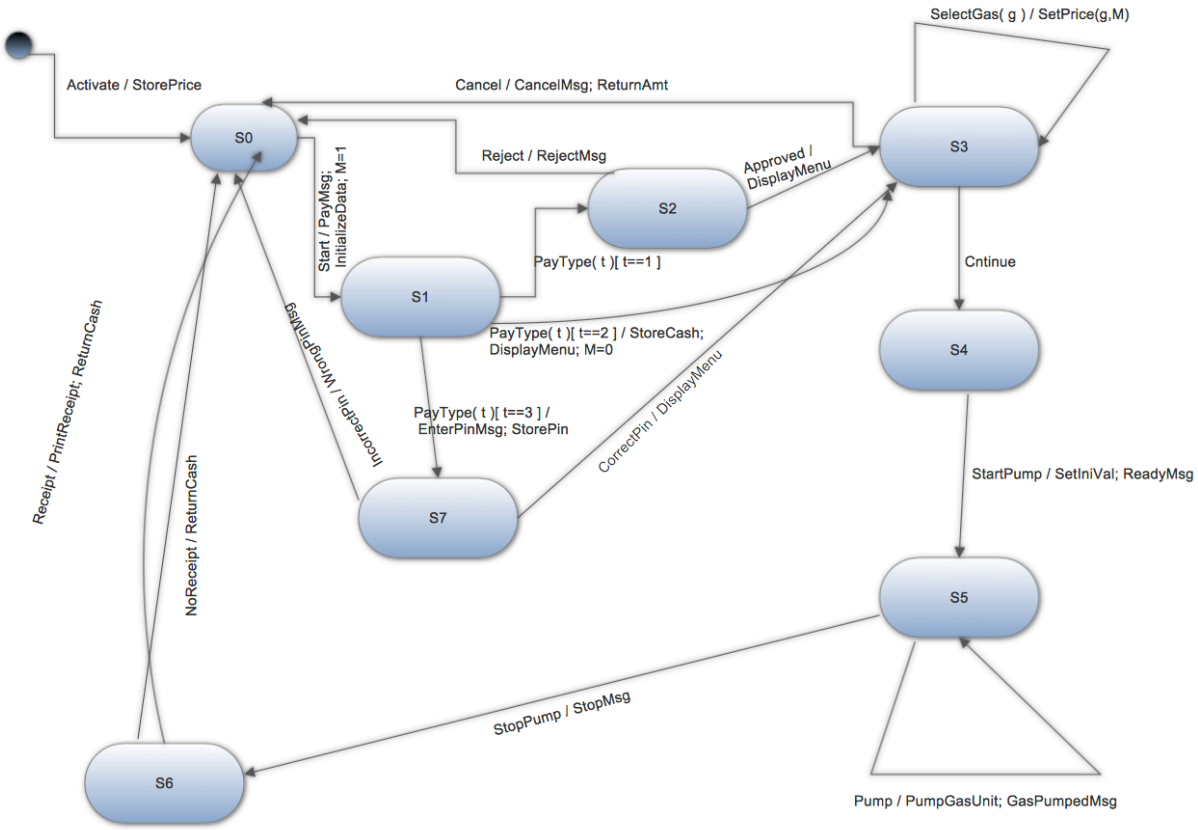
(i) A list of meta events for the MDA-EFSM

Activate()
Start()
PayType(int t) //credit: t=1; cash: t=2; debit: t=3
Reject()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int g)
Receipt()
NoReceipt()
CorrectPin()
IncorrectPin()
Continue()

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

StorePrice - stores price(s) for the gas from the temporary data store
PaymentMsg - 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, int M) - set the price for the gas identified by g identifier as in SelectGas(int g); if M=1, the price may be increased
ReadyMsg - displays the ready for pumping message
SetInitialValues - set G (or L) and total to 0;
PumpGasUnit - disposes unit of gas and counts # of units disposed
GasPumpedMsg - displays the amount of disposed gas
StopMsg - stop pump message and receipt
PrintReceipt - print a receipt
CancelMsg - displays a cancellation message
ReturnAmt - returns the remaining cash
WrongPinMsg - displays incorrect pin message
StorePin - stores the pin from the temporary data store
EnterPinMsg - displays a message to enter pin
SetIniVal - set the value of price and cash to 0

(iii) A state diagram of the MDA-EFSM



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

Operations of the Input Processor (GasPump-1)

```
Activate(float a, float b)
{
        if ((a>0)&&(b>0))
         {
                d->temp_a=a;
                d->temp_b=b;
                m->Activate()
        }
}
```

```
Start()
{
        m->Start();
}

PayCredit()
{
        m->PayType(1);
}
Reject()
{
        m->Reject();
}
PayDebit(string p)
 {
        d->temp_p=p;
        m->PayType(3);
}
Pin(string x)
{

        if (d->pin==x)
                m->CorrectPin()
        else
                m->InCorrectPin();
}
Cancel()
{
        m->Cancel();
}
Approved()
 {
        m->Approved();
}
Diesel() {
        m->SelectGas(4)
}
Regular()
{
        m->SelectGas(1)
}
StartPump()
{
          if (d->price>0)
        {
                m->Continue();
```

```
                m->StartPump();
        }
}


PumpGallon()
{
        m->Pump();
}
StopPump()
{
        m->StopPump();
        m->Receipt();
}
FullTank()
{
        m->StopPump();
        m->Receipt();
}
```

Notice:
m: is a pointer to the MDA-EFSM object
d: is a pointer to the Data Store object

Operations of the Input Processor of GasPump-2:

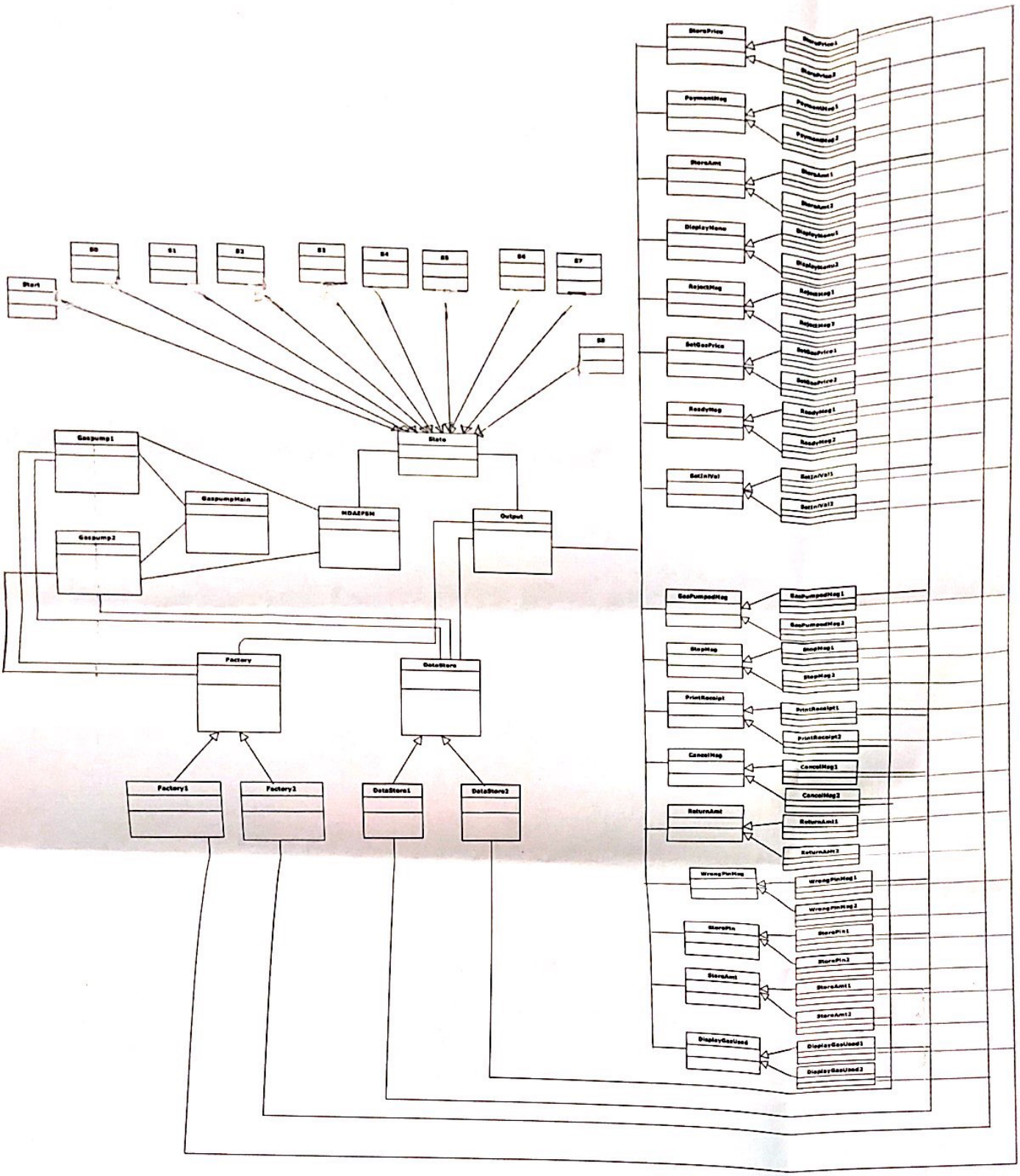
```
Activate(int a, int b, int c)
{

        if ((a>0)&&(b>0)&&(c>0))
        {
                d->temp_a=a;
                d->temp_b=b;
                d->temp_c=c
                m->Activate()
                m->Start()
        }
}
PayCash(float c)
{
        if (c>0)
        {

                d->temp_cash=c;
                m->PayType(2)
        }
}
```

```
PayCredit()
{
        m->PayType(1);
 }

Reject()
{
        m->Reject();
        m->Start();
}
Approved()
{

        m-> Approved();
}
Cancel()
{
        m->Cancel();
        m->Start()
}
Super()
{
         m->SelectGas(2);
        m->Continue();
 }
Premium()
{
        m->SelectGas(3);
        m->Continue();
}
Regular()
{
        m->SelectGas(1);
        m->Continue();
}
StartPump()
{
         m->StartPump();
}
PumpLiter()
{

        if (d->cash>0)&&(d->cash < d->price*(d->L+1))
                m->StopPump();
        else
```

```
        m->Pump()
}
Stop()
{
        m->StopPump();
}
Receipt()
{
        m->Receipt();
        m->Start();
}
NoReceipt()
{
         m->NoReceipt();
        m->Start();
}
```

Notice:
cash: contains the value of cash deposited
price: contains the price of the selected
gas L: contains the number of liters already pumped
cash , L, price are in the data store
m: is a pointer to the MDAEFSM object
d: is a pointer to the Data Store object

2. Class diagram(s) of the MDA of the GasPump components.



Note :

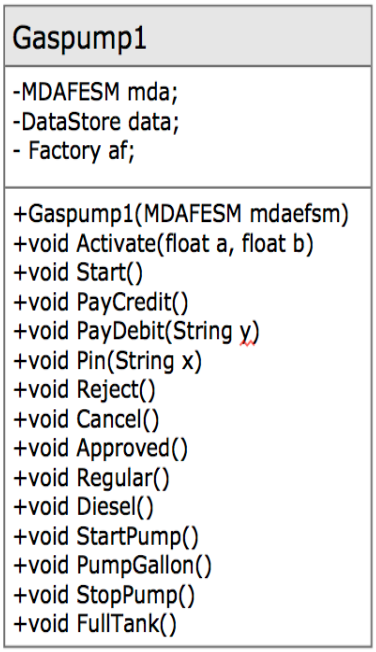
All methods of the classes are discussed further. Functionality of each method of each classes are mentioned seperately.

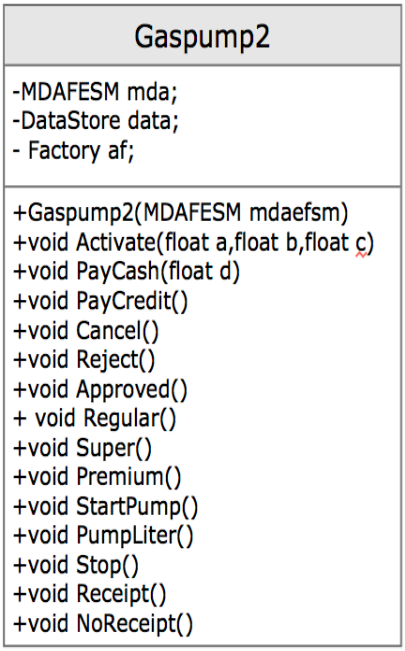
3. For each class in the class diagram(s) you should:

a. Describe the purpose of the class, i.e., responsibilities.

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

GasPump1 and GasPump2 Classes:





Gaspump1 Class:

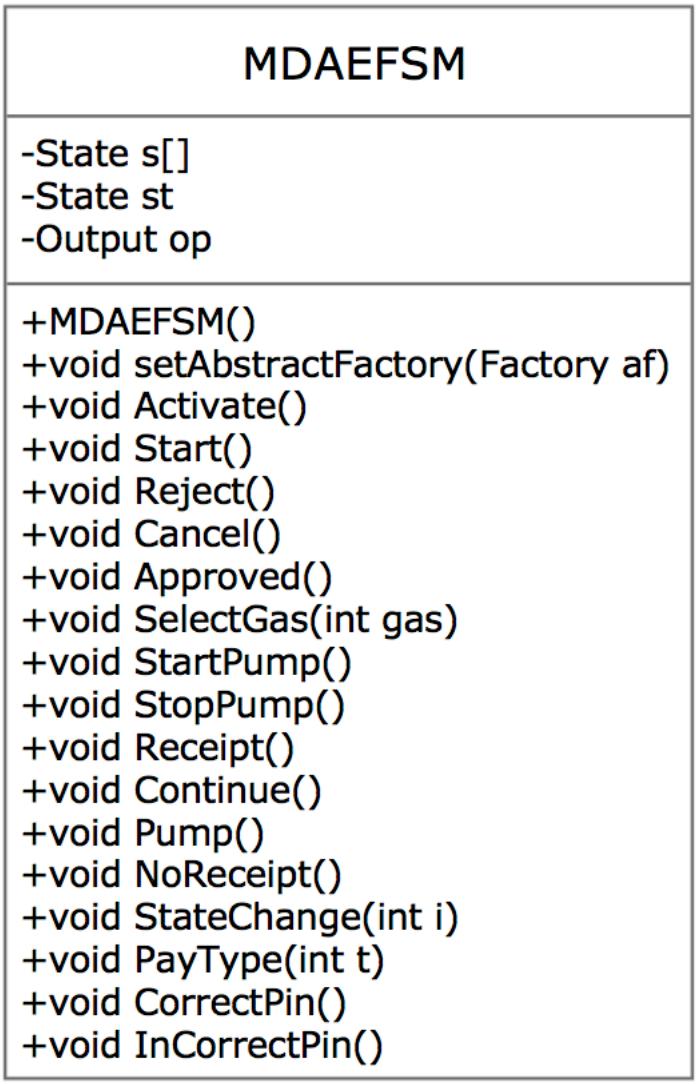
Purpose	Provides all the functionalities which are specific to Gaspump1
Gaspump1(MDAEFSM mdaefsm)	It sets the constructor for the MDAEFSM class, Factory1 and Factory(Abstract Factory)
Activate(float a, float b)	Calls Activate function in MDAFFSM class. GasPump is activated where a is price of Regular and b is the price of Diesel gas per gallon
Start()	Start the transaction
PayCredit()	Pay for gas by credit card
PayDebit(String y)	Pay for gas by debit card and set the actual pin of the card
Pin(String x)	User is asked to enter pin to make transaction successful
Reject()	Credit card is rejected
Cancel()	Cancel the transaction
Approved()	Credit card approved

Regular()	Regular gas selected
Diesel()	Diesel gas selected
StartPump	Start Pumping Gas
PumpGallon()	Every time user select this, 1 Gallon Gas is disposed
StopPump()	Stop pumping gas
FullTank()	Stop pumping gas because tank is full.

Gaspump2 Class:

Purpose	Provides all the functionalities which are specific to Gaspump2
Gaspump2(MDAEFSM mdaefsm)	It sets the constructor for the MDAEFSM class, Factory2 and Factory(Abstract Factory)
Activate(float a, float b, float c)	Pump is activated where a is price of Regular gas and b is the price of Premium gas and c is price of Super gas per liter
PayCash(float d)	Pay for gas by cash , d is the amount given by the user
PayCredit()	Pay for gas by credit card
Cancel()	Cancel the transaction
Reject()	Credit card is rejected
Approved()	Credit card approved
Regular()	Regular gas selected
Super()	Super gas selected
Premium()	Premium gas selected
StartPump()	Start Pumping Gas
PumpLiter()	Every time user select this, 1 Liter Gas is disposed
Stop()	Stop pumping gas
Receipt()	Receipt is requested
NoReceipt()	No receipt

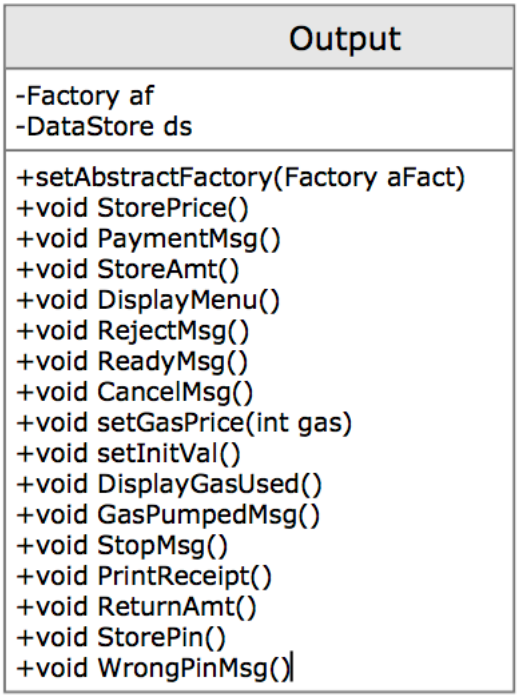
MDAEFSM Class:



Purpose	Contains all platform independent events. It is associated with Output and will perform actions based on current state.
MDAEFSM()	It creates a new state that has the pointers set to all the states individually
setAbstractFactory(Factory af)	It is abstract method
Activate()	Activate the pump
Start()	Start the pump
PayType(int t)	Three types of payment methods i.e By Cash , By Credit Card or By Debit Card
Reject()	Credit Card Rejected

Approved()	Credit card approved
SelectGas(int gas)	To select type of gas
StartPump()	Start pumping gas
StopPump()	Stop pumping gas
Cancel()	Cancel the transaction
Continue()	Continue where no more selection of gas is allowed
Pump()	Pump gas
NoReceipt()	No receipt
Receipt()	Receipt requested
StateChange(int i)	Changes state from one state to another
CorrectPin()	Debit card approved
InCorrectPin()	Debit card rejected

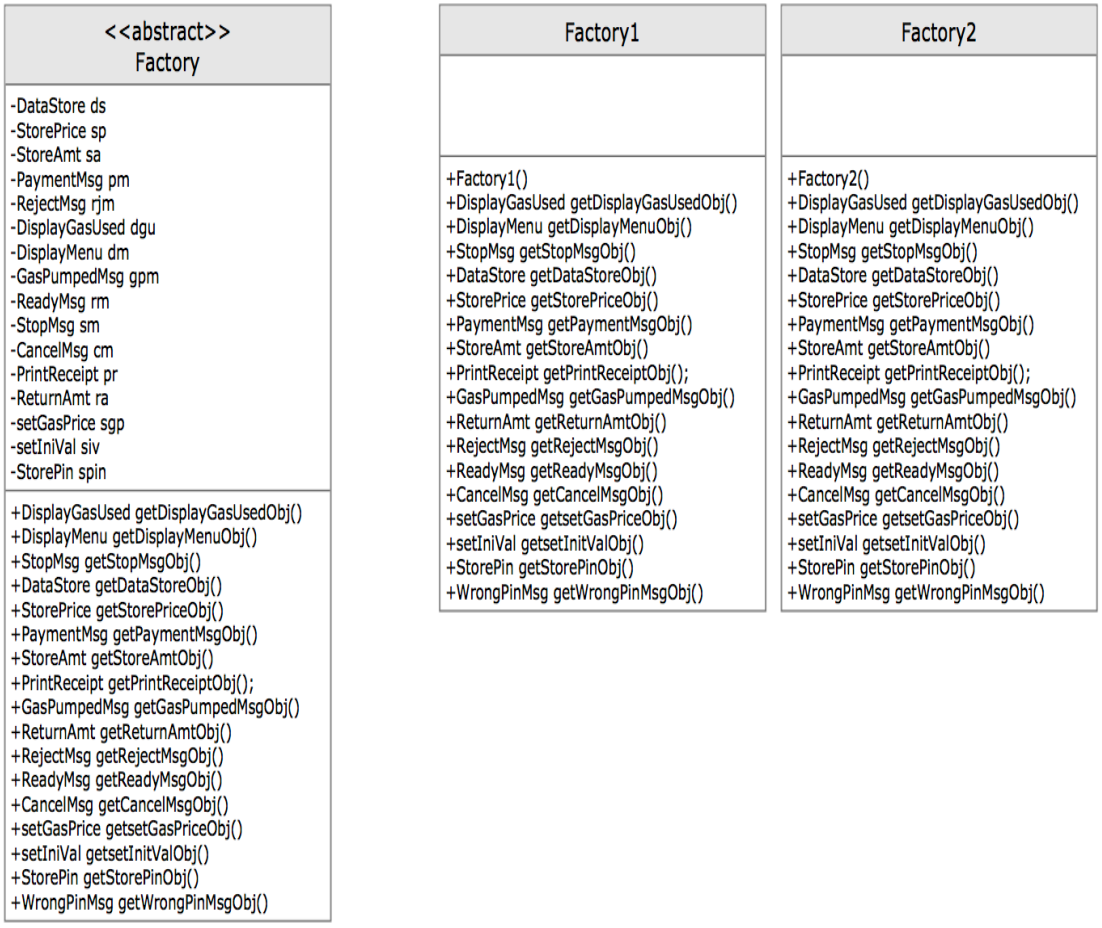
Output Class:



Purpose	Contains all platform specific actions. Each actions have 2 strategies where each is for different Gaspump components
setAbstractFactory(Factory aFact)	It is abstract method
StorePrice()	Stores the price of the gas

PaymentMsg()	Will display the type of payment method
StoreAmt()	Will store the cash amount from the temporary data store
DisplayMenu()	Will display menu with list of different gases
RejectMsg()	Will Display Credit card reject message
ReadyMsg()	Will display ready to pump message
CancelMsg()	Will display Gas selection cancelled message
setGasPrice(int gas)	Will set price of gas identified by gas identifier
setInitVal()	Will set initial value of gas unit: G or L to 0
DisplayGasUsed()	Will display total unit of gas used
GasPumpedMsg()	Will display Gas Pumped Message
StopMsg()	Will Display Pump stopped message
PrintReceipt()	Will print the receipt of amount paid
ReturnAmt()	Will return the cash amount remaining after the total is deducted.
StorePin()	Will Store Pin of debit card from temporary data store
WrongPinMsg()	Will Display Wrong Pin Msg

AbstractFactory and its Factory Classes



Factory Class:

Purpose	Used to create Factory Classes and group them together. Will also expose functions and member variables specific to Gaspump component that are achieved by different combinations
DataStore getDataStoreObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
DisplayGasUsed getDisplayGasUsedObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
DisplayMenu getDisplayMenuObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
StopMsg getStopMsgObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
StorePrice getStorePriceObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
PaymentMsg getPaymentMsgObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
StoreAmt getStoreAmtObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
PrintReceipt getPrintReceiptObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
GasPumpedMsg getGasPumpedMsgObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
ReturnAmt getReturnAmtObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
RejectMsg getRejectMsgObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
ReadyMsg getReadyMsgObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
CancelMsg getCancelMsgObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
setGasPrice getsetGasPriceObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
setIniVal getsetInitValObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
StorePin getStorePinObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes
WrongPinMsg getWrongPinMsgObj()	It is an abstract method that is implemented in Factory1 and Factory2 Classes

Factory1 Class:

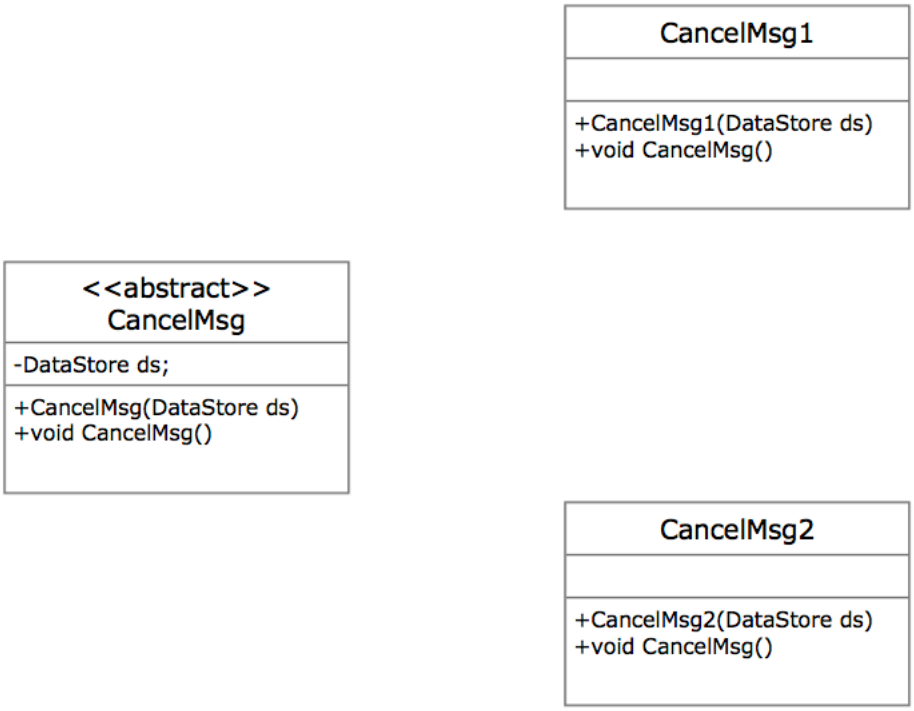
Purpose	Will extend Factory Class and is used to create all class objects specific to GasPump1
Factory1()	Helps in returning DataStore1 object
DataStore getDataStoreObj()	Returns object of DataStore1 Class
DisplayGasUsed getDisplayGasUsedObj()	Returns object of DisplayGasUsed1 Class
DisplayMenu getDisplayMenuObj()	Returns object of DisplayMenu1 Class

StopMsg getStopMsgObj()	Returns object of StopMsg1 Class
StorePrice getStorePriceObj()	Returns object of StorePrice1 Class
PaymentMsg getPaymentMsgObj()	Returns object of PaymentMsg1 Class
StoreAmt getStoreAmtObj()	Returns object of StoreAmt1 Class
PrintReceipt getPrintReceiptObj()	Returns object of PrintReceipt1 Class
GasPumpedMsg getGasPumpedMsgObj()	Returns object of GasPumpedMsg1 Class
ReturnAmt getReturnAmtObj()	Returns object of ReturnAmt1 Class
RejectMsg getRejectMsgObj()	Returns object of RejectMsg1 Class
ReadyMsg getReadyMsgObj()	Returns object of ReadyMsg1 Class
CancelMsg getCancelMsgObj()	Returns object of CancelMsg1 Class
setGasPrice getsetGasPriceObj()	Returns object of setGasPrice1 Class
setIniVal getsetInitValObj()	Returns object of setInitVal1 Class
StorePin getStorePinObj()	Returns object of StorePin1 Class
WrongPinMsg getWrongPinMsgObj()	Returns object of WrongPinMsg1 Class

Factory2 Class:

Purpose	Will extend Factory Class and is used to create all class objects specific to GasPump2
Factory1()	Helps in returning DataStore2 object
DataStore getDataStoreObj()	Returns object of DataStore2 Class
DisplayGasUsed getDisplayGasUsedObj()	Returns object of DisplayGasUsed2 Class
DisplayMenu getDisplayMenuObj()	Returns object of DisplayMenu2 Class
StopMsg getStopMsgObj()	Returns object of StopMsg2 Class
StorePrice getStorePriceObj()	Returns object of StorePrice2 Class
PaymentMsg getPaymentMsgObj()	Returns object of PaymentMsg2 Class
StoreAmt getStoreAmtObj()	Returns object of StoreAmt1 Class
PrintReceipt getPrintReceiptObj()	Returns object of PrintReceipt2 Class
GasPumpedMsg getGasPumpedMsgObj()	Returns object of GasPumpedMsg2 Class
ReturnAmt getReturnAmtObj()	Returns object of ReturnAmt2 Class
RejectMsg getRejectMsgObj()	Returns object of RejectMsg2 Class
ReadyMsg getReadyMsgObj()	Returns object of ReadyMsg2 Class
CancelMsg getCancelMsgObj()	Returns object of CancelMsg2 Class
setGasPrice getsetGasPriceObj()	Returns object of setGasPrice2 Class
setIniVal getsetInitValObj()	Returns object of setInitVal2 Class
StorePin getStorePinObj()	Returns object of StorePin2 Class
WrongPinMsg getWrongPinMsgObj()	Returns object of WrongPinMsg1 Class

CancelMsg, CancelMsg1 and CancelMsg2 Classes:



CancelMsg Class:

Purpose	Responsible to create the data objects for particular class, where data objects come from the abstract factory, called by the Output which defines Strategy pattern for CancelMsg
CancelMsg(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
CancelMsg()	It is an abstract method implemented in CancelMsg1 and CancelMsg2 Classes

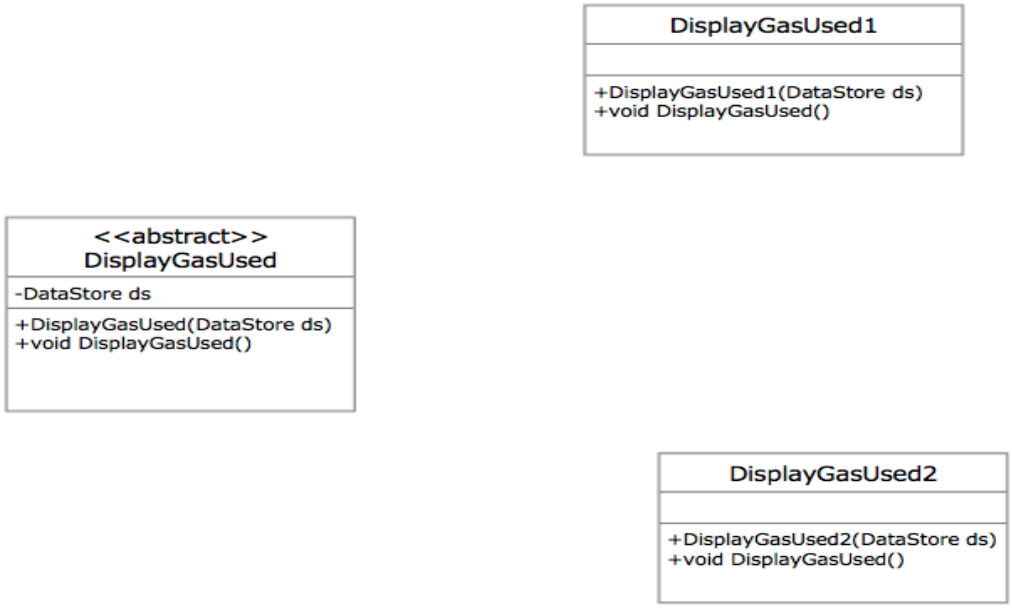
CancelMsg1 Class:

Purpose	Responsible for cancelling the transaction even after the payment is done. Called when Output recognizes object for GasPump1
CancelMsg1(DataStore ds)	Constructor that returns the DataStore object
CancelMsg()	Prints cancel message when the transaction is cancelled

CancelMsg2 Class:

Purpose	Responsible for cancelling the transaction even after the payment is done. Called when Output recognizes object for GasPump2
CancelMsg2(DataStore ds)	Constructor that returns the DataStore object
CancelMsg()	Prints cancel message when the selection for the gas is cancelled.

DisplayGasUsed, DisplayGasUsed1 and DisplayGasUsed2 Classes:



DisplayGasUsed Class

Purpose	Responsible to create data objects for particular class where data object come from abstract factory, called by the Output which defines Strategy pattern for DisplayGasUsed.
DisplayGasUsed(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
DisplayGasUsed()	It is an abstract method implemented in DisplayGasUsed1 and DisplayGasUsed2 Classes

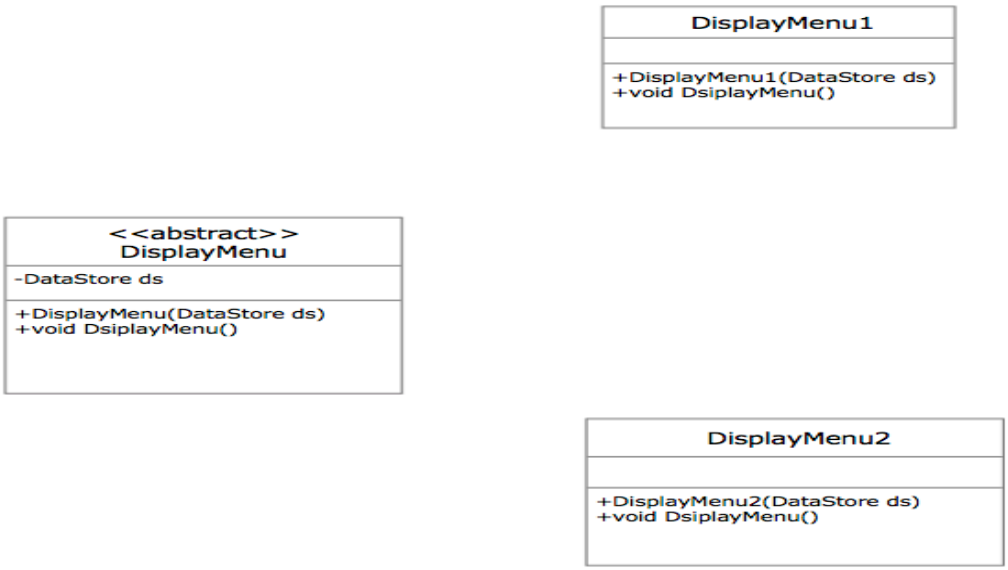
DisplayGasUsed1 Class**:**

Purpose	Calculates gallons of gas pumped. Class is called when the Output recognizes object for GasPump1
DisplayGasUsed1(DataStore ds)	Constructor that returns the DataStore object
DisplayGasUsed()	Responsible to set the gallon and calculate the amount

DisplayGasUsed2 Class**:**

Purpose	Calculates gallons of gas pumped. Class is called when the Output recognizes object for GasPump2
DisplayGasUsed2(DataStore ds)	Constructor that returns the DataStore object
DisplayGasUsed()	Responsible to set the Liter and calculate the amount

DisplayMenu, DisplayMenu1 and DisplayMenu2 Classes:



DisplayMenu Class:

Purpose	Responsible to create data objects for particular class where data object come from abstract factory.Called by the Output which defines Strategy pattern for DisplayMenu.
DisplayMenu(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
DisplayMenu()	It is an abstract method implemented in DisplayMenu1 and DisplayMenu2 Classes

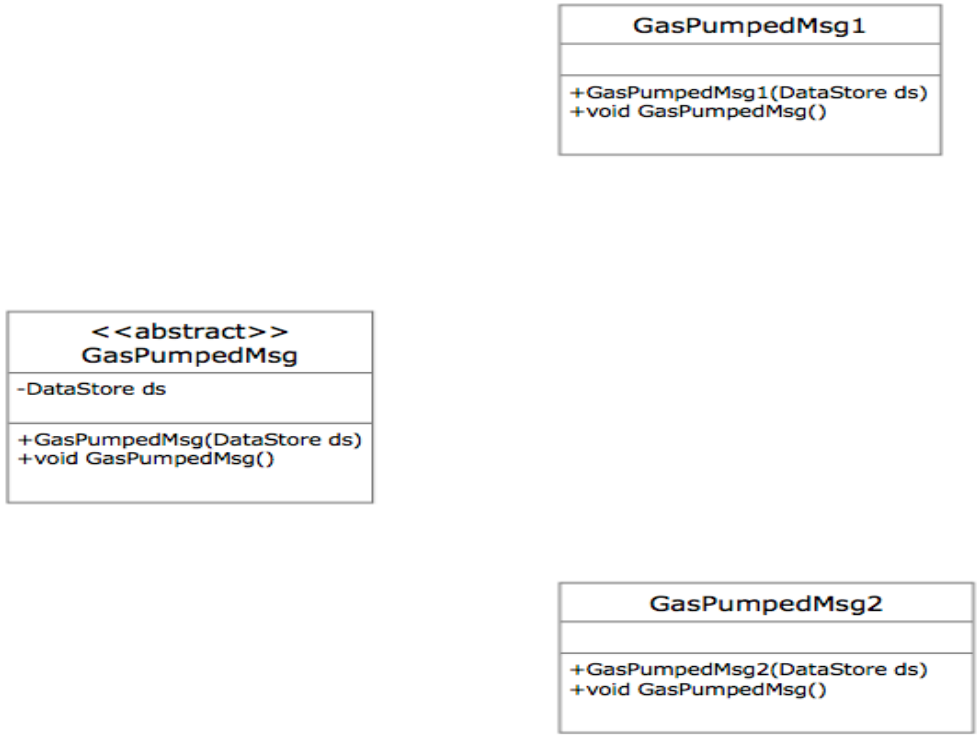
DisplayMenu1 Class:

Purpose	Responsible for Displaying Menu once the payment method is selected. Called when Output recognizes object for GasPump1
DisplayMenu1(DataStore ds)	Constructor that returns the DataStore object
DisplayMenu()	Prints menu of gas pump once the price is set and the payment type is selected.

DisplayMenu2 Class:

Purpose	Responsible for displaying menu once the payment method is selected, called when Output recognizes object for GasPump2
DisplayMenu2(DataStore ds)	Constructor that returns the DataStore object
DisplayMenu()	Prints menu of gas pump once the price is set and the payment type is selected.

GasPumpedMsg, GasPumpedMsg1 and GasPumpedMsg2



GasPumpedMsg Class:

Purpose	Responsible to create data objects for particular class where data object come from abstract factory, called by the Output which defines Strategy pattern for GasPumpedMsg.
GasPumpedMsg(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
GasPumpedMsg()	It is an abstract method implemented in GasPumpedMsg1 and GasPumpedMsg2 Classes

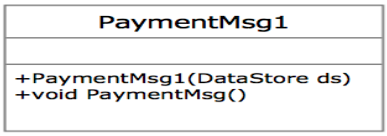
GasPumpedMsg1 Class:

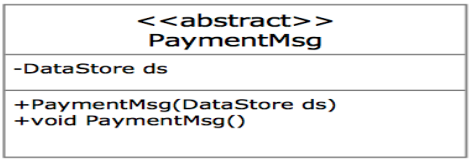
Purpose	Responsible for printing total gallon pumped, called when Output recognizes object for GasPump1
GasPumpedMsg(1DataStore ds)	Constructor that returns the DataStore object
GasPumpedMsg()	Prints Gas Pumped message when the pump stops

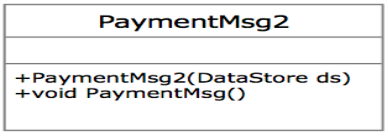
GasPumpedMsg2 Class:

Purpose	Responsible for printing total gallon pumped, called when Output recognizes object for GasPump2
GasPumpedMsg2(DataStore ds)	Constructor that returns the DataStore object
GasPumpedMsg()	Prints Gas Pumped message when the pump stops

PaymentMsg, PaymentMsg1 and PaymentMsg2 Classes:







PaymentMsg Class:

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for PaymentMsg
PaymentMsg(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
PaymentMsg()	It is an abstract method implemented in PaymentMsg1 and PaymentMsg2 Classes

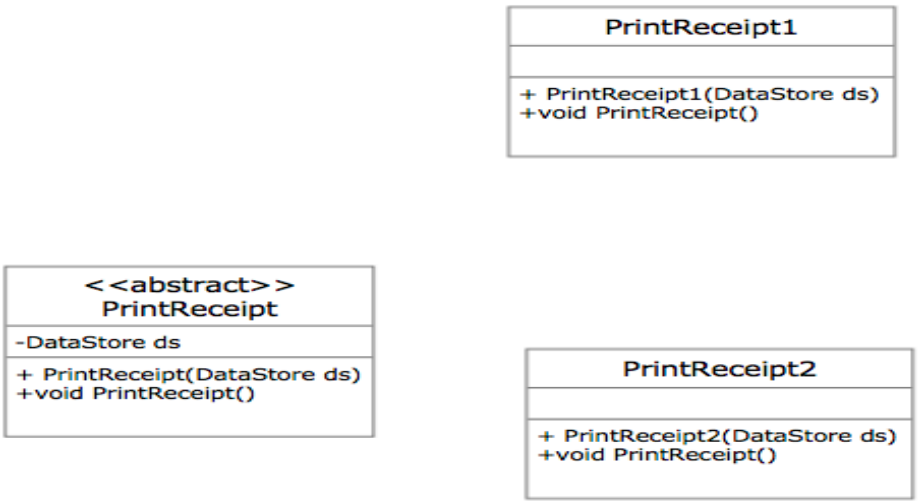
PaymentMsg1 Class:

Purpose	Allows user to select payment method once price has been set.Called when Output recognizes object for GasPump1
PaymentMsg1(DataStore ds)	Constructor that returns the DataStore object
PaymentMsg()	Prints payment type gaspump1 is providing.

PaymentMsg2 Class:

Purpose	Allows user to select payment method once price has been set.Called when Output recognizes object for GasPump2
PaymentMsg2(DataStore ds)	Constructor that returns the DataStore object
PaymentMsg()	Prints payment type gaspump2 is providing.

PrintReceipt, PrintReceipt1 and PrintReceipt2 Classes:



PrintReceipt Class

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for PrintReceipt
PrintReceipt(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
PrintReceipt()	It is an abstract method implemented in PrintReceipt1 and PrintReceipt2 Classes

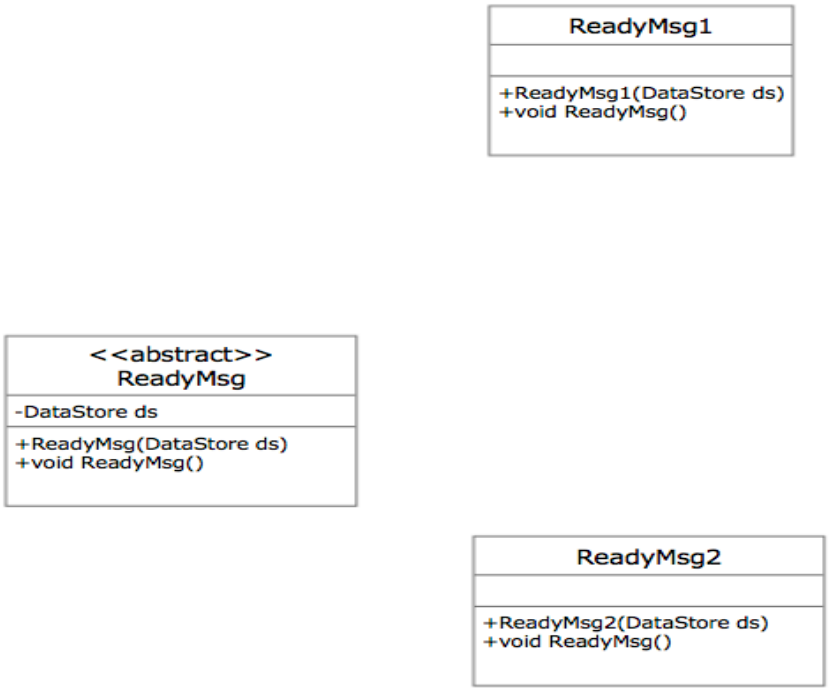
PrintReceipt1 Class

Purpose	Responsible for printing the receipt after pump stops. Called when Output recognizes object for GasPump1
PrintReceipt1(DataStore ds)	Constructor that returns the DataStore object
PrintReceipt()	Prints receipt once the gas pump stops

PrintReceipt2 Class

Purpose	Responsible for printing the receipt after pump stops. Called when Output recognizes object for GasPump2
PrintReceipt2(DataStore ds)	Constructor that returns the DataStore object
PrintReceipt()	Prints receipt once the gas pump stops and user choose receipt option

ReadyMsg, ReadyMsg1 and ReadyMsg2 Classes:



ReadyMsg Class

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for ReadyMsg.
ReadyMsg(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
ReadyMsg()	It is an abstract method implemented in ReadyMsg1 and ReadyMsg2 Classes

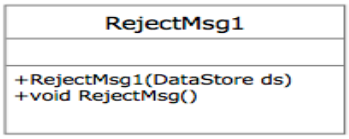
ReadyMsg1 Class

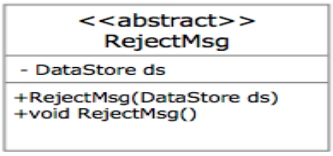
Purpose	Responsible for alerting the pump to start once gas type is selected. Called when Output recognizes object for GasPump1
ReadyMsg1(DataStore ds)	Constructor that returns the DataStore object
ReadyMsg()	Prints Ready message when pump is ready to start

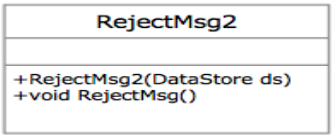
ReadyMsg2 Class

Purpose	Responsible for alerting the pump to start once gas type is selected. Called when Output recognizes object for GasPump2
ReadyMsg2(DataStore ds)	Constructor that returns the DataStore object
ReadyMsg()	Prints Ready message when pump is ready to start

RejectMsg, RejectMsg1 and RejectMsg2 Classes:







RejectMsg Class

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for RejectMsg.
RejectMsg(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
RejectMsg()	It is an abstract method implemented in RejectMsg1 and RejectMsg2 Classes

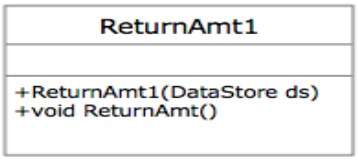
RejectMsg1 Class

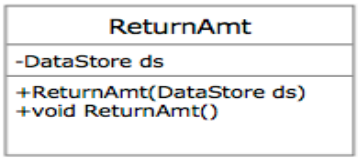
Purpose	Responsible to reject transaction if credit card not valid. Called when Output recognizes object for GasPump1
RejectMsg1(DataStore ds)	Constructor that returns the DataStore object
RejectMsg()	Prints reject message when credit card is not authorized

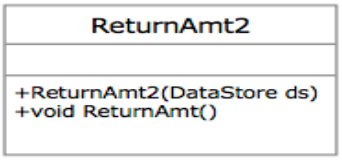
RejectMsg2 Class

Purpose	Responsible to reject transaction if credit card not valid. Called when Output recognizes object for GasPump2
RejectMsg2(DataStore ds)	Constructor that returns the DataStore object
RejectMsg()	Prints reject message when credit card is not authorized

ReturnAmt, ReturnAmt1 and ReturnAmt2 Classes:







ReturnAmt Class:

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for ReturnAmt.
ReturnAmt(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
ReturnAmt()	It is an abstract method implemented in ReturnAmt1 and ReturnAmt2 Classes

ReturnAmt1 Class:

Purpose	Called when Output recognizes object for GasPump1
ReturnAmt1(DataStore ds)	Constructor that returns the DataStore object
ReturnAmt()	No implementation

ReturnAmt2 Class:

Purpose	Called when Output recognizes object for GasPump2
ReturnAmt2(DataStore ds)	Constructor that returns the DataStore object
ReturnAmt()	Calculates total amount of remaining and returns it

setGasPrice, setGasPrice1 and setGasPrice2 Classes:

setGasPrice Class:

Purpose	Responsible to create data objects for particular class where data object come from abstract factory.Called by the Output which defines Strategy pattern for setGasPrice
setGasPrice(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two
setGasPrice(int gas)	It is an abstract method implemented in setGasPrice1 and setGasPrice2 Classes

setGasPrice1 Class:

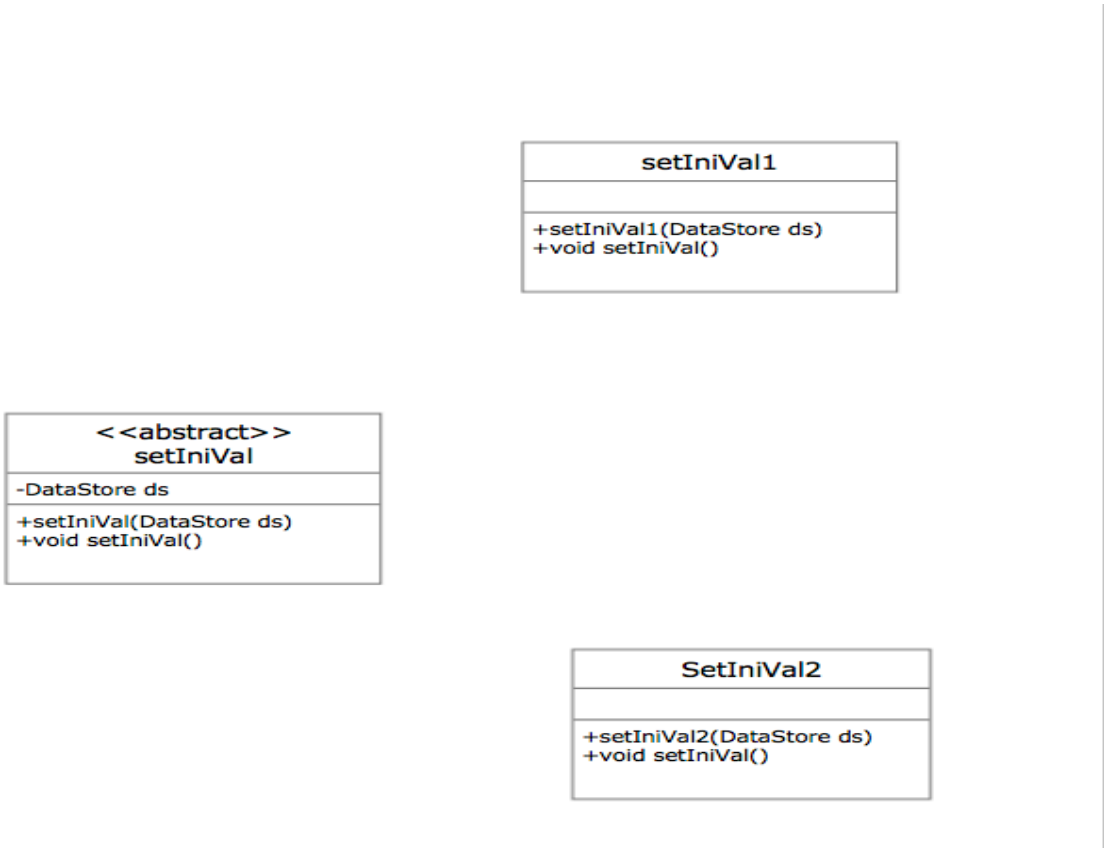
Purpose	Responsible for setting the price of selected gas.Called when Output recognizes object for GasPump1
setGasPrice1(DataStore ds)	Constructor that returns the DataStore object
setGasPrice(int gas)	Sets the price for a particular gas type

setGasPrice2 Class:

Purpose	Responsible for setting the price of selected gas.Called when Output recognizes object for GasPump2
setGasPrice2(DataStore ds)	Constructor that returns the DataStore object

setGasPrice(int gas)	Sets the price for a particular gas type , if payment type is credit card then price would be 1.1*price.

setInitVal, setInitVal1 and setInitVal2 Classes:



setIniVal Class:

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for setIniVal.
setIniVal(DataStore ds)	Reurns DataStore object and that object will be responsible to implement class depending on either objects of the two.
setIniVal()	It is an abstract method implemented in setIniVal1 and setIniVal2 Classes

setIniVal1 Class:

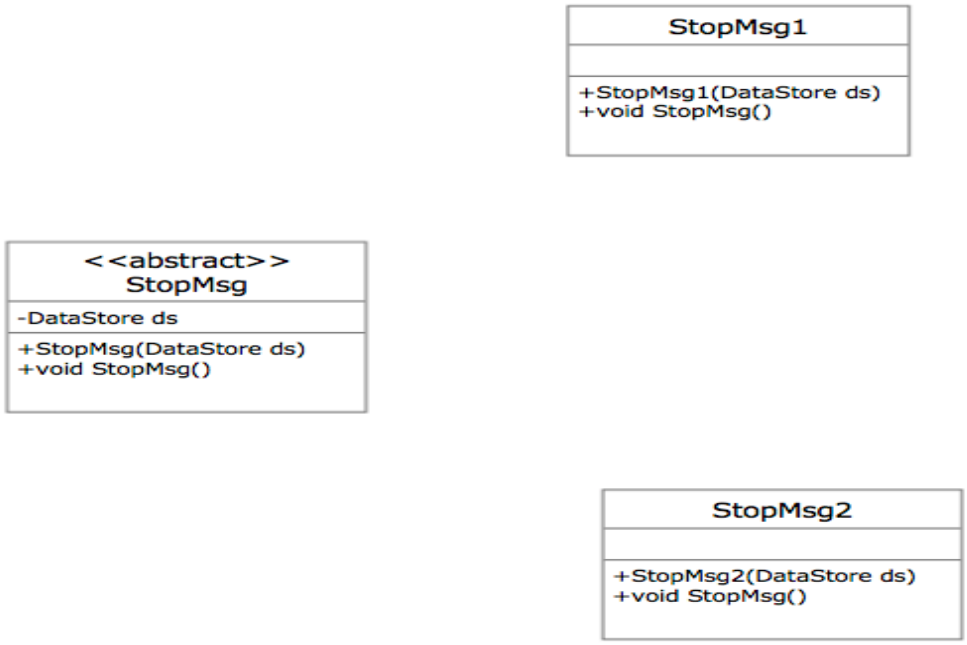
Purpose	Responsible for setting the initial values of gallon pumped and total to 0. Called when Output recognizes object for GasPump1
setIniVal1(DataStore ds)	Constructor that returns the DataStore object
setIniVal()	Sets the initial gallon pumped and total to 0.

setIniVal2 Class:

Purpose	Responsible for setting the initial values of gallon pumped and total to 0. Called when Output recognizes object for GasPump2

setIniVal2(DataStore ds)	Constructor that returns the DataStore object
setIniVal()	Sets the initial liter pumped and total to 0.

StopMsg, StopMsg1 and StopMsg2 Classes:



StopMsg Class:

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for StopMsg.
StopMsg(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
StopMsg()	It is an abstract method implemented in StopMsg1 and StopMsg2 Classes

StopMsg1 Class:

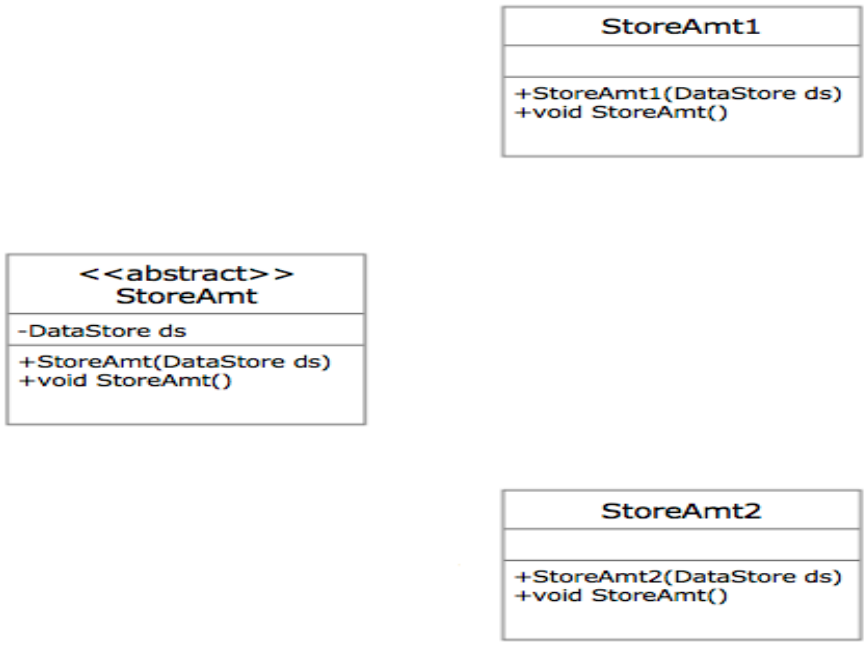
Purpose	Responsible for stopping the pump after gas is pumped. Called when Output recognizes object for GasPump1
StopMsg(DataStore ds)	Constructor that returns the DataStore object
StopMsg()	Prints the stop message when the pump stops

StopMsg2 Class:

Purpose	Responsible for stopping the pump after gas is pumped. Called when Output recognizes object for GasPump2
StopMsg(DataStore ds)	Constructor that returns the DataStore object

StopMsg()	Prints the stop message when the pump stops

StoreAmt, StoreAmt1 and StoreAmt2 Classes:



StoreAmt Class:

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for StoreAmt
StoreAmt(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
StoreAmt()	It is an abstract method implemented in StoreAmt1 and StoreAmt2 Classes

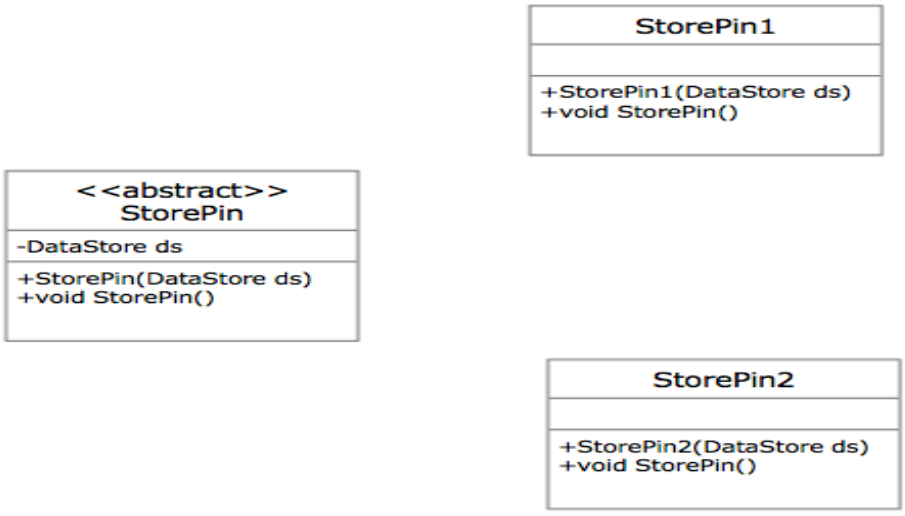
StoreAmt1 Class:

Purpose	Called when Output recognizes object for GasPump1
StoreAmt1(DataStore ds)	Constructor that returns the DataStore object
StoreAmt()	No implementation

StoreAmt2 Class:

Purpose	Responsible for storing the cash input by the user. Called when Output recognizes object for GasPump2
StoreAmt2(DataStore ds)	Constructor that returns the DataStore object
StoreAmt()	Responsible for storing cash value

StorePin , StorePin1, StorePin2 Class:



StorePin Class

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for StorePin
StorePin(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two
StorePin()	It is an abstract method implemented in StorePin1 and StorePin2 Classes

StorePin1 Class

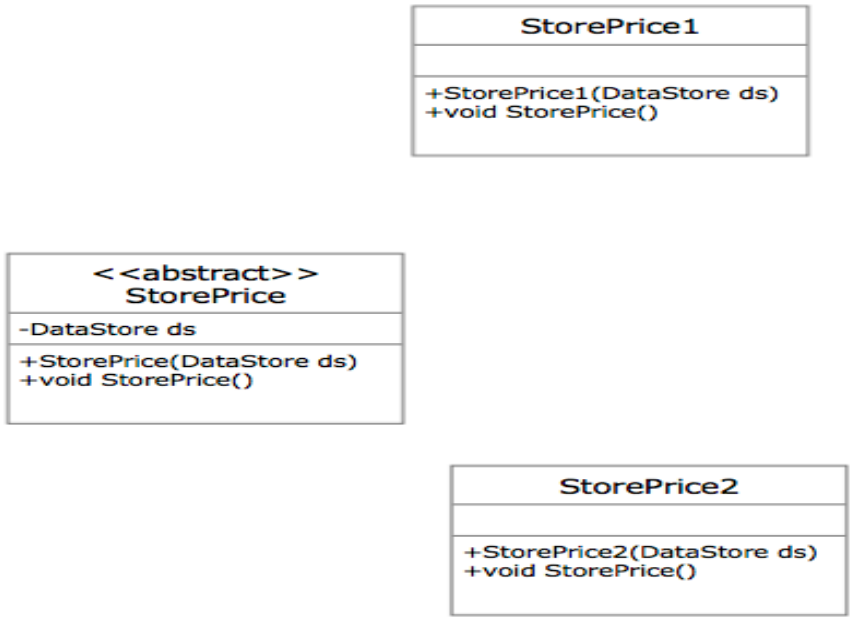
Purpose	Responsible for storing the pin input by the user. Called when Output recognizes object for GasPump1
StorePin(DataStore ds)	Constructor that returns the DataStore object
StorePin()	Responsible for storing Pin value of Debit Card

StorePin2 Class

Purpose	Called when Output recognizes object for GasPump2
StorePin(DataStore ds)	Constructor that returns the DataStore object

StorePin()	No implementation

StorePrice, StorePrice1 and StorePrice2 Classes:



StorePrice Class

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for StorePrice
StorePrice(DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
StorePrice()	It is an abstract method implemented in StorePrice1 and StorePrice2 Classes

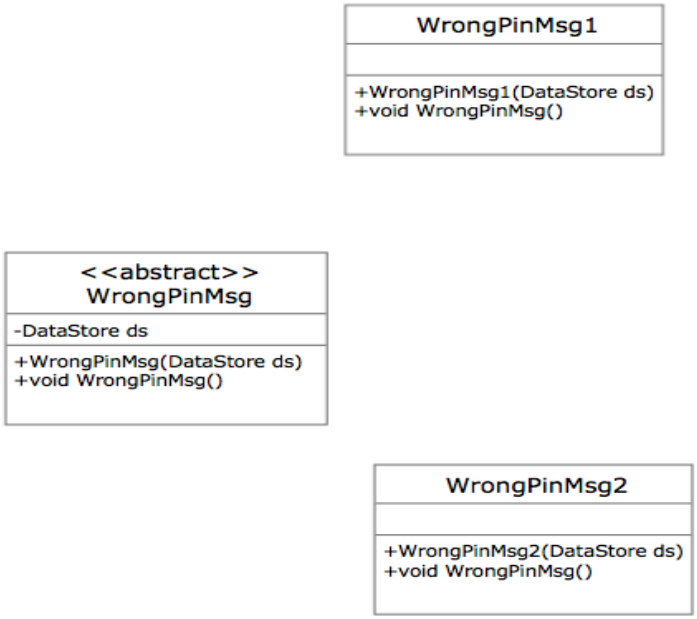
StorePrice1 Class

Purpose	Responsible for storing price of different gas.Called when Output recognizes object for GasPump1
StorePrice1(DataStore ds)	Constructor that returns the DataStore object
StorePrice()	Stores the price of the different gas

StorePrice2 Class

Purpose	Responsible for storing price of different gas.Called when Output recognizes object for GasPump2
StorePrice2(DataStore ds)	Constructor that returns the DataStore object
StorePrice()	Stores the price of the different gas

WrongPinMsg , WrongPinMsg1 and WrongPinMsg2 Classes:



WrongPinMsg Class

Purpose	Responsible to create data objects for particular class where data object come from abstract factory. Called by the Output which defines Strategy pattern for WrongPinMsg Class
WrongPinMsg (DataStore ds)	Returns DataStore object and that object will be responsible to implement class depending on either objects of the two.
WrongPinMsg()	It is an abstract method implemented in WrongPinMsg1 and WrongPinMsg2 Class

WrongPinMsg1 Class

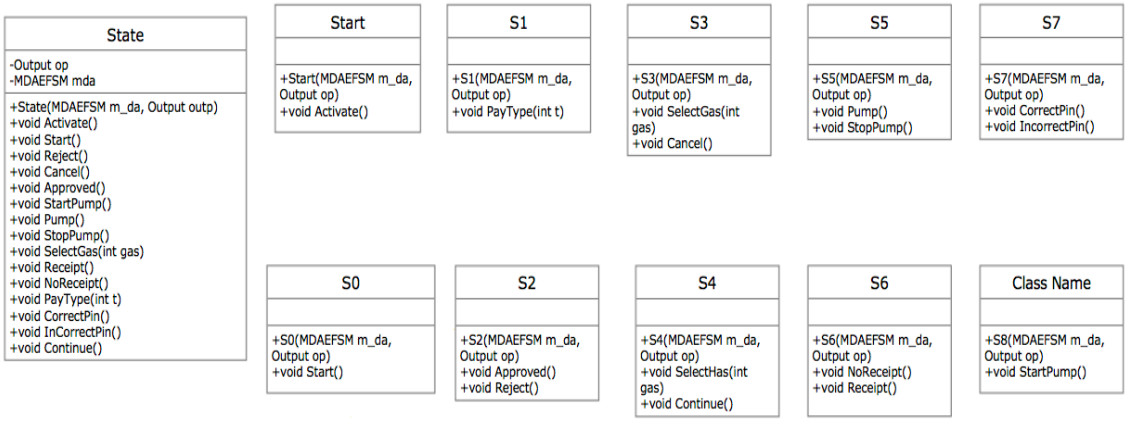
Purpose	Responsible for Message after wrong pin entered of different gas.Called when Output recognizes object for GasPump1
WrongPinMsg1(DataStore ds)	Constructor that returns the DataStore object
StorePrice()	Prints Wrong Pin Message

WrongPinMsg2 Class

Purpose	Responsible for Message after wrong pin entered of different gas.Called when Output recognizes object for GasPump2

WrongPinMsg2(DataStore ds)	Constructor that returns the DataStore object
WrongPinMsg()	No Implementation

State, Start, S0, S1, S2, S3, S4, S5, S6 and S7 Classes:



Start Class:

Purpose	It is State pattern implemented by MDAEFSM and calls Output for further implementation. MDAEFSM pointer can be used to change states.
State(MDAEFSM m_da, Output outp)	Sets MDAEFSM and Output Objects.
Activate()	It is an abstract method implemented in Start class
Start()	It is an abstract method implemented in S0 class
Reject()	It is an abstract method implemented in S2 class
Cancel()	It is an abstract method implemented in S3 class
Approved()	It is an abstract method implemented in S2 class
StartPump()	It is an abstract method implemented in S4 class
Pump()	It is an abstract method implemented in S5 class
StopPump()	It is an abstract method implemented in S5 class
SelectGas(int gas)	It is an abstract method implemented in S3 and S8 class
Receipt()	It is an abstract method implemented in S6class
NoReceipt()	It is an abstract method implemented in S6 class

PayType(int t)	It is an abstract method implemented in S1 class
CorrectPin()	It is an abstract method implemented in S7 class
InCorrectPin()	It is an abstract method implemented in S7 class
Continue()	It is an abstract method implemented in S4 class

Start Class

Purpose	It is the start of the state which is then transmitted to S0
Start(MDAEFSM m_da, Output outp)	Calls MDAEFSM and Output Objects from super class
Activate()	Calls StorePrice method and changes state from 0 to 1

Note: Other abstract methods of State Class are also defined in Start Class which only prints that they are user is at invalid state and cannot use that function that the current state.

S0 class:

Purpose	It is the 1st state which is then transmitted to S1
Start(MDAEFSM m_da, Output outp)	Calls MDAEFSM and Output Objects from super class
Start()	Calls PaymentMsg method and change state from 1 to 2.

Note: Other abstract methods of State Class are also defined in S0 Class which only prints that the user is at invalid state and cannot use that function at the current state

S1 Class:

Purpose	It is the 2nd state which is then transmitted to S2,S3 or S7
S1(MDAEFSM m_da, Output outp)	Calls MDAEFSM and Output Objects from super class
PayType(int t)	If user selects credit payment than state changes from 2 to 3, if user selects to pay with cash than calls storeAmt and displaymenu and state changes from 2 to 4 and is user selects to pay with debit than state changes from 2 to 8.

Note: Other abstract methods of State Class are also defined in S1 Class which only prints that they are user is at invalid state and cannot use that function that the current state.

S2 Class

Purpose	It is the 3rd state which is then transmitted to S3 or S0
S2(MDAEFSM m_da, Output outp)	Calls MDAEFSM and Output Objects from super class
Approved()	Calls DisplayMenu method and change state from 3 to 4.
Reject()	Calls RejectMsg method and change state from 3 to 1

Note: Other abstract methods of State Class are also defined in S2 Class which only prints that they are user is at invalid state and cannot use that function that the current state.

S3 Class

Purpose	It is the 4th state which is then transmitted to S4 or S0
S3(MDAEFSM m_da, Output outp)	Calls MDAEFSM and Output Objects from super class
SelectGas(int gas)	Calls setGasPrice method and change state from 4 to 5.
Cancel()	Calls CancelMsg method and change state from 4 to 1

Note: Other abstract methods of State Class are also defined in S3 Class which only prints that they are user is at invalid state and cannot use that function that the current state.

S4 Class

Purpose	It is the 5th state which is then transmitted to S5 or remains in S4
S4(MDAEFSM m_da, Output outp)	Calls MDAEFSM and Output Objects from super class
SelectGas(int gas)	Calls setGasPrice method and remains in same state.

Note: Other abstract methods of State Class are also defined in S4 Class which only prints that they are user is at invalid state and cannot use that function that the current state.

S5 Class

Purpose	It is the 6th state which is then transmitted to S6 or remains in S5
S5(MDAEFSM m_da, Output outp)	Calls MDAEFSM and Output Objects from super class
Pump()	Calls DisplayGasUsed and GasPumpedMsg method .
StopPump()	Calls StopMsg method and change state from 6 to 7.

Note: Other abstract methods of State Class are also defined in S5 Class which only prints that they are user is at invalid state and cannot use that function that the current state.

S6 Class

Purpose	It is the 7th state which is then transmitted to S0
S6(MDAEFSM m_da, Output outp)	Calls MDAEFSM and Output Objects from super class
Receipt()	Calls ReturnAmt and PrintReceipt method and change state from 7 to 1.
NoReceipt()	Calls ReturnAmt method and change state from 7 to 1

Note: Other abstract methods of State Class are also defined in S6 Class which only prints that they are user is at invalid state and cannot use that function that the current state.

S7 Class

Purpose	It is the 8th state which is then transmitted to S3 or to S0
S7(MDAEFSM m_da, Output outp)	Calls MDAEFSM and Output Objects from super class
CorrectPin()	Calls DisplayMenu method and change state from 7 to 4.
InCorrectPin()	Change state from 7 to 1

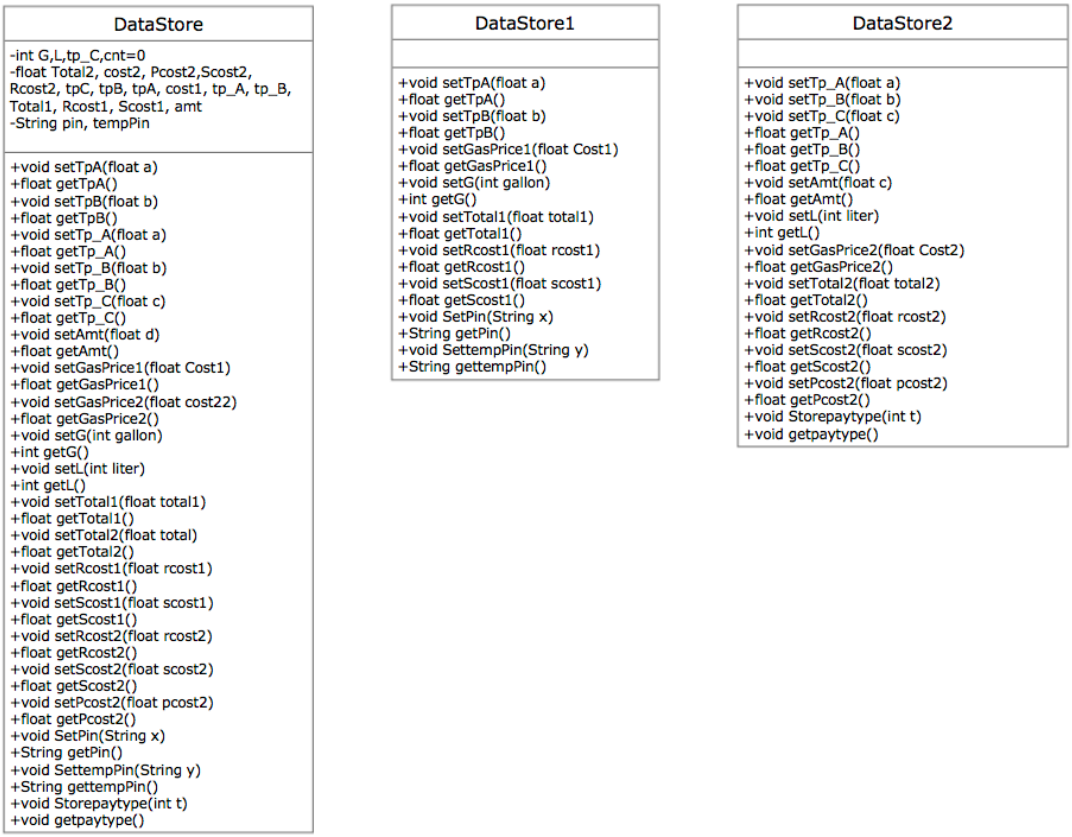
Note: Other abstract methods of State Class are also defined in S7 Class which only prints that they are user is at invalid state and cannot use that function that the current state.

S8 Class

Purpose	It is the 9th state which is then transmitted to S5
S8(MDAEFSM m_da, Output outp)	Calls MDAEFSM and Output Objects from super class
StartPump()	Calls setInitVal and ReadyMsg method and change state from 5 to 6.

Note: Other abstract methods of State Class are also defined in S8 Class which only prints that they are user is at invalid state and cannot use that function that the current state.

DataStore, DataStore1 and DataStore2 Classes:



DataStore Class:

Purpose: Decentralized State Pattern has been used so we need to keep DataStore Class seperate where all the data is stored. All variables and functions has been defined here that are required for different classes. It is implemented by DataStore1 and DataStore2 Classes

DataStore1 Class

Purpose	Responsible to set and get values required for the GasPump1
void setTpA(float a)	Sets the price of gastype=1
void setTpB(float b)	Gets the price of gastype=1
float getTpA()	Sets the price of gastype=4

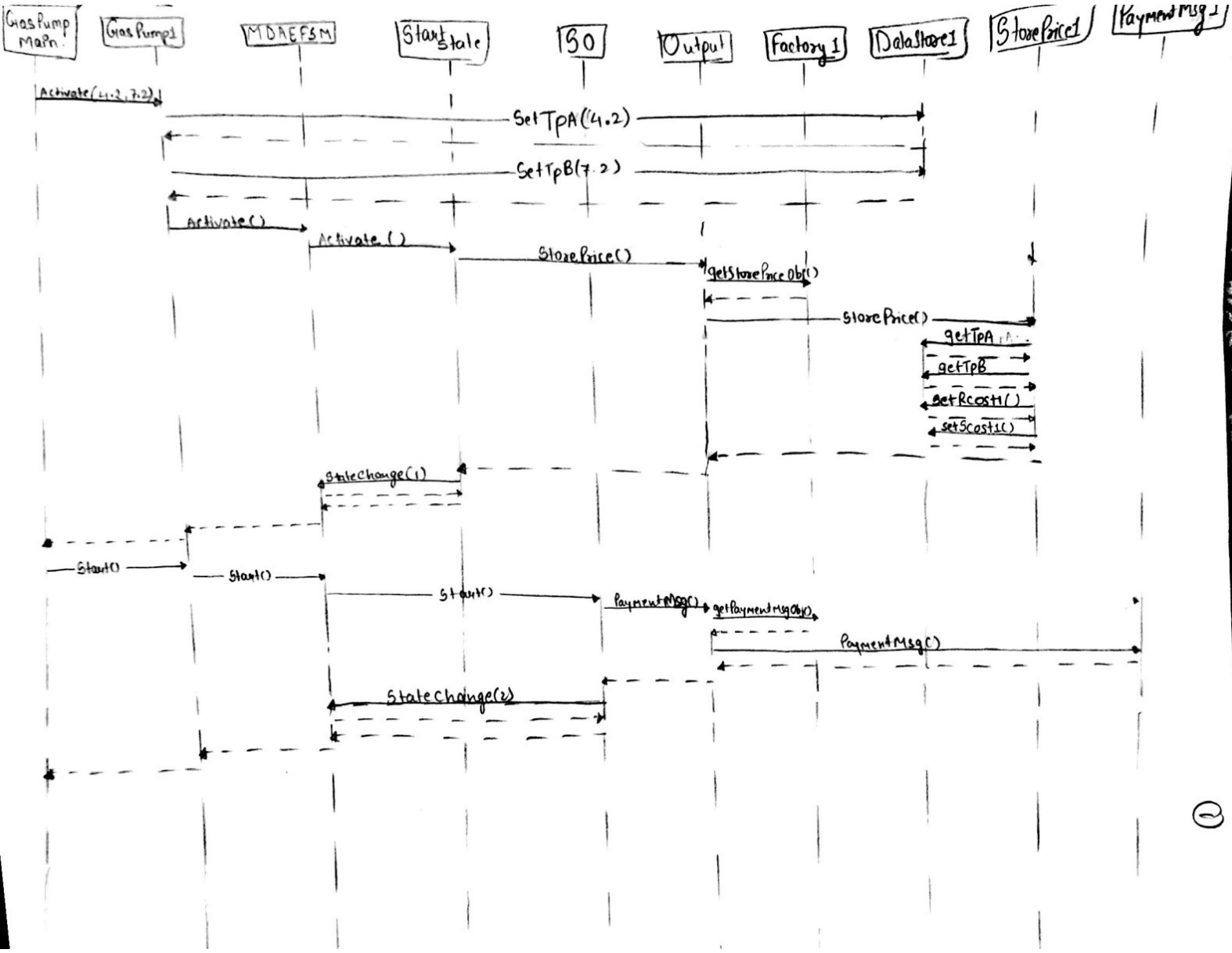
float getTpB()	Gets the price of gastype=4
void setGasPrice1(float Cost1)	Sets the price of the selected gastype
float getGasPrice1()	Gets the price of the selected gastype
void setG(int gallon)	Sets the total gallon pumped
int getG()	Gets the total gallon pumped
void setTotal1(float total1)	Sets the total amount charged for the gallon pumped
float getTotal1()	Gets the total amount charged for the gallon pumped
void setRcost1(float rcost1)	Sets the price of the regular gas
float getRcost1()	Gets the price of the regular gas
void setScost1(float scost1)	Sets the price of the Diesel gas
float getScost1()	Gets the price of the Diesel gas
void SetPin(String x)	Set the pin of the debit card in the database
String getPin()	Gets the pin of the debit card from the database
void Settemppin(String y)	Set the pin of the card from the user (temporary)
String gettempPin()	Gets the pin of the debit card entered by the user

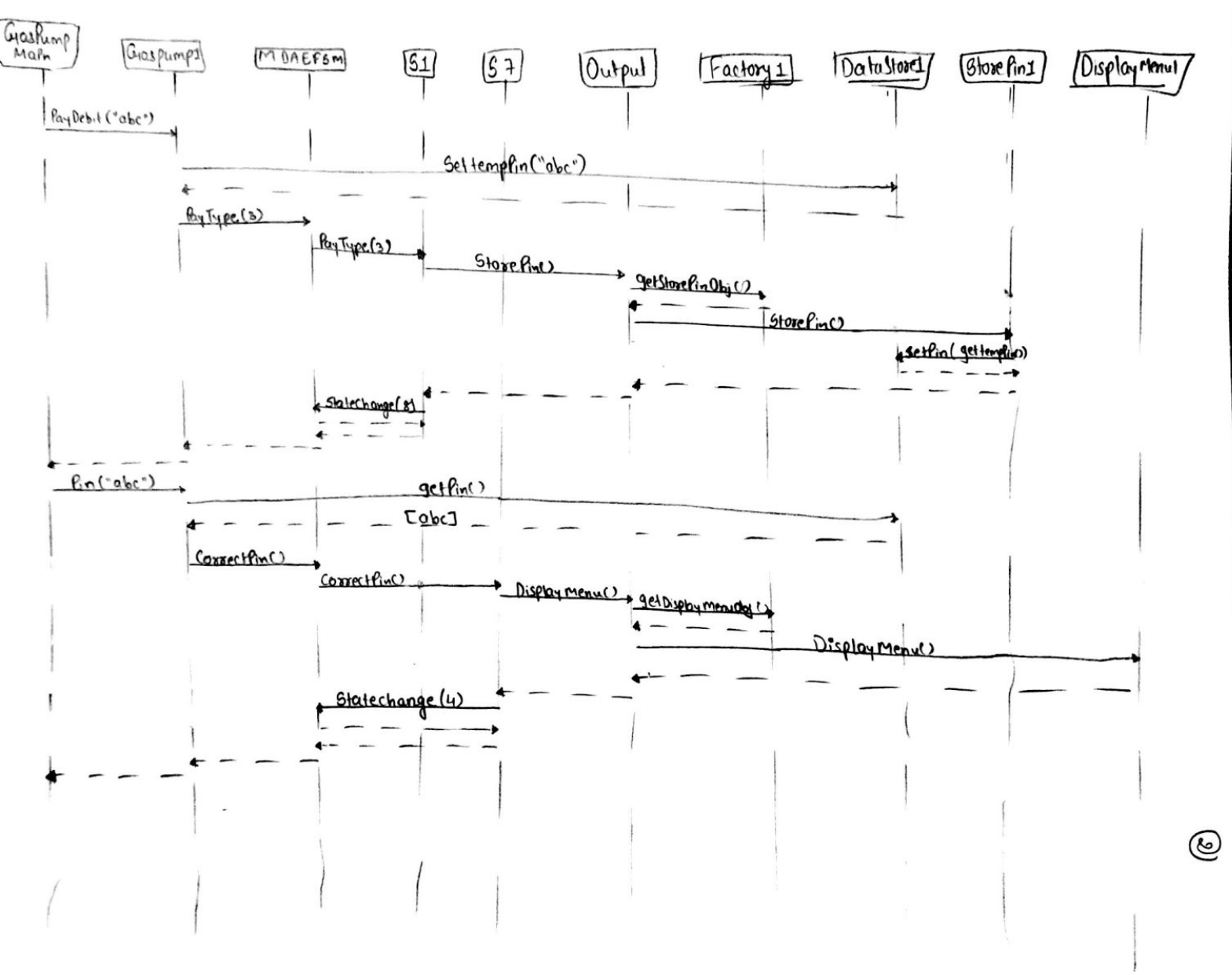
DataStore2 Class

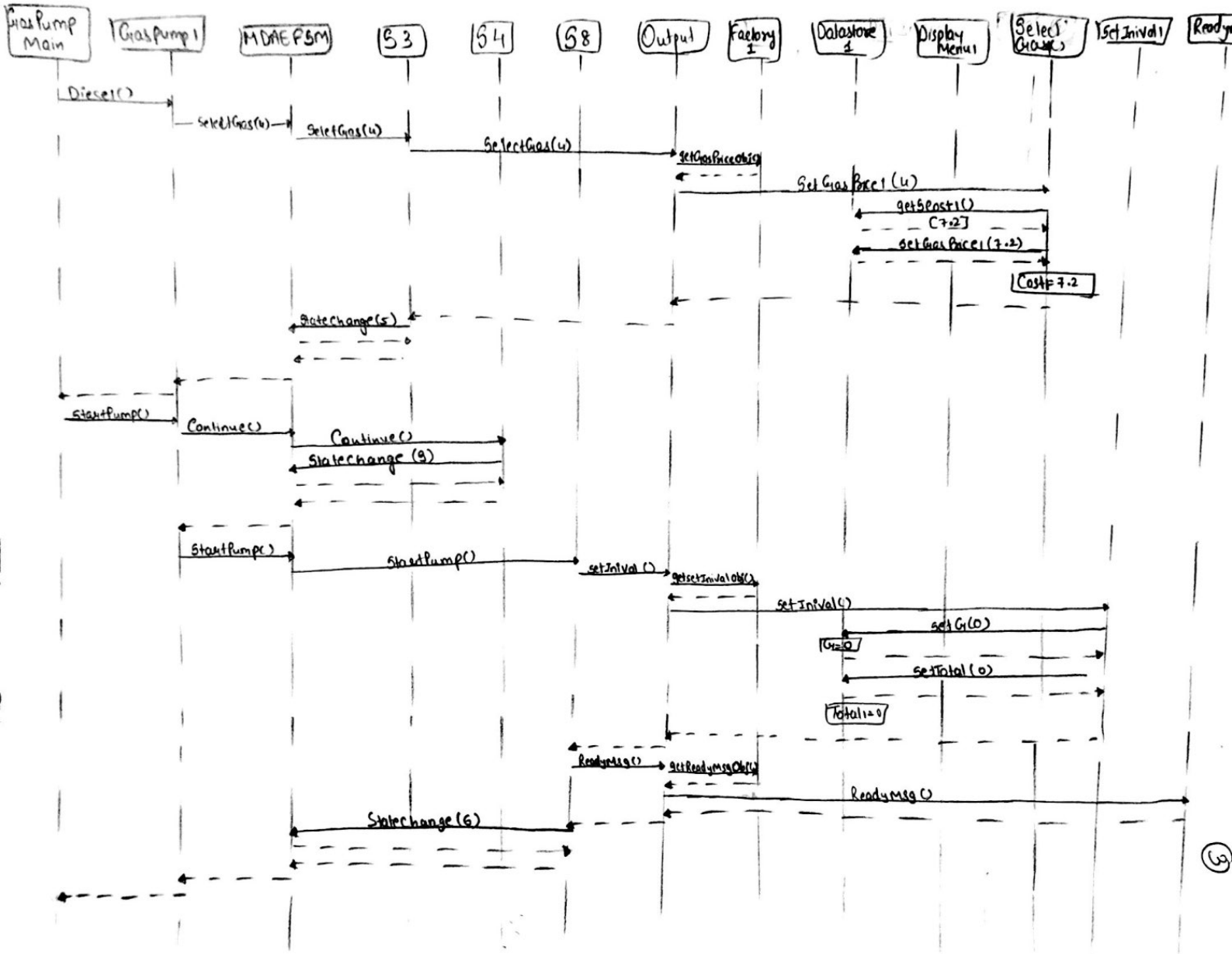
Purpose	Responsible to set and get values required for the GasPump2
void setTp_A(float a)	Sets the price of gastype=1
void setTp_B(float b)	Sets the price of gastype=2
void setTp_c(float c)	Sets the price of gastype=3
float gettp_A()	Gets the price of gastype=1
float gettp_B()	Gets the price of gastype=2
float gettp_C()	Gets the price of gastype=3
void setAmt(float c)	Sets the value of cash
float getAmt()	Gets the value of cash
void setL(int liter)	Sets the total liter pumped
int getL()	Gets the total liter pumped
void setGasPrice2(float Cost2)	Sets the price of the selected gastype
float getGasPrice2()	Gets the price of the selected gastype
void setTotal2(float total2)	Sets the total2 amount charged for the Liter pumped
float getTotal2()	Gets the total2 amount charged for the Liter pumped
void setRcost2(float rcost2)	Sets the price of the regular gas
float getRcost2()	Gets the price of the regular gas
void setScost2(float scost2)	Sets the price of the super gas
float getScost2()	Gets the price of the super gas
void setPcost2(float pcost2)	Sets the price of the Premium gas
float getPcost2()	Gets the price of the Premium gas
storepaytype(int t)	Sets the payment type user selected
getpaytype()	Returns the payment type user selected

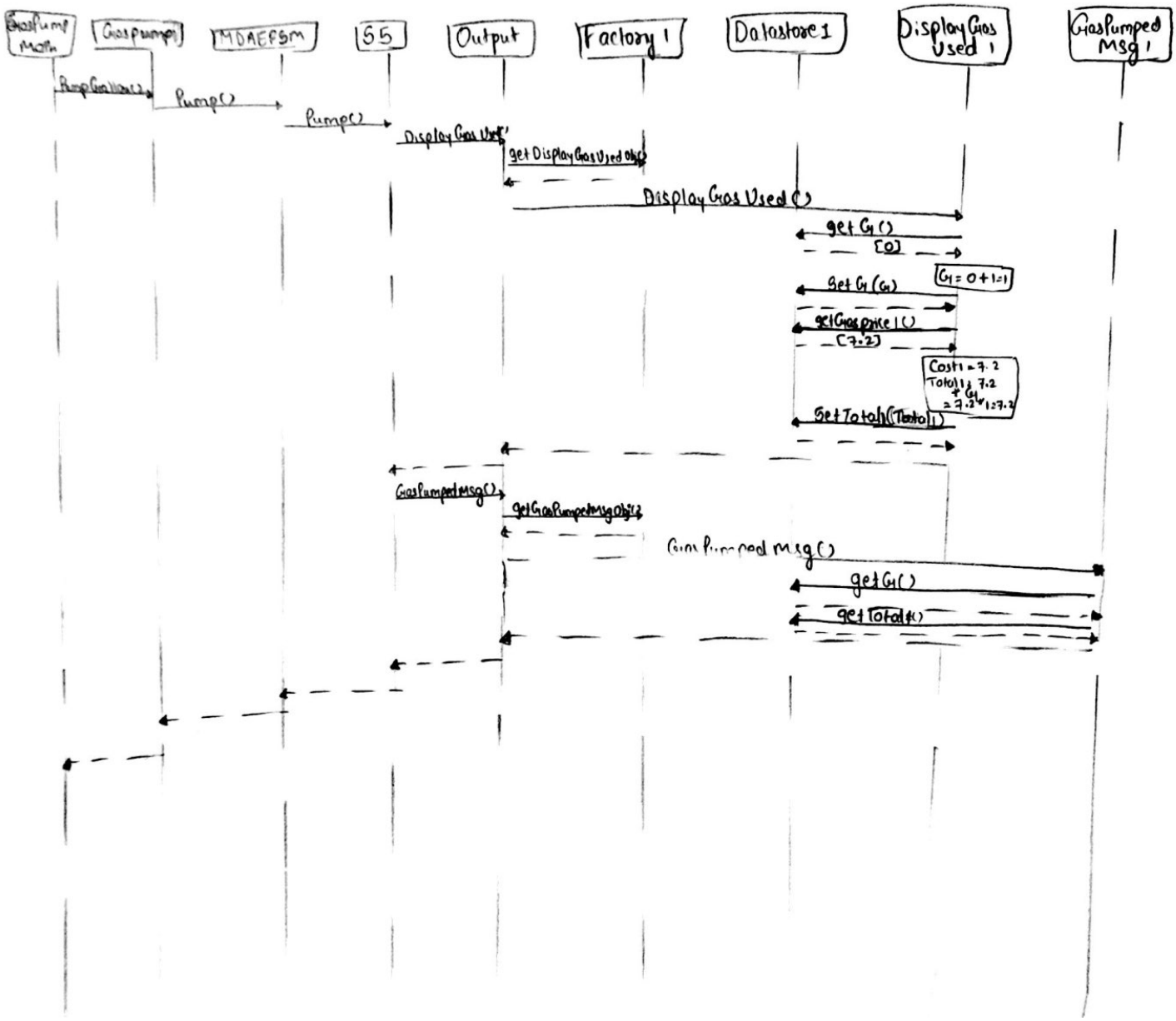
4 Dynamics. Provide two sequence diagrams for two Scenarios:

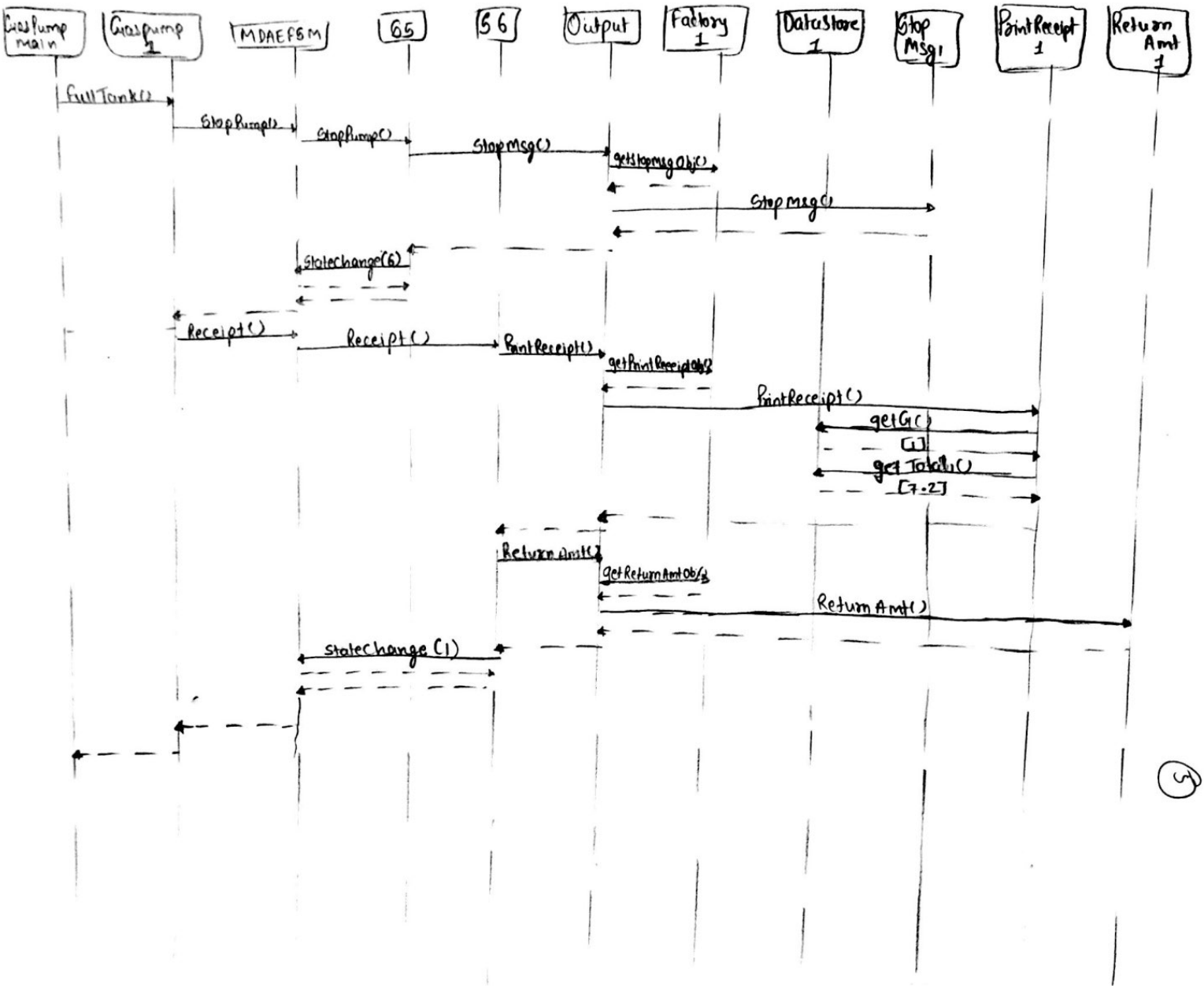
a. Scenario-I should show how one gallon of Diesel gas is disposed in GasPump-1, i.e., the following sequence of operations is issued: Activate(4.2, 7.2), Start(), PayDebit("abc"), Pin("abc"), Diesel(), StartPump(), PumpGallon(), FullTank()











b. Scenario-II should show how one liter of Premium gas is disposed in GasPump-2, i.e., the following sequence of operations is issued: Activate(3, 4, 5.2), PayCash(10), Premium(), StartPump(), PumpLiter(), PumpLiter(), NoReceipt()