CS-586 SOFTWARE SYSTEMARCHITECTURE

PROJECT REPORT

ABHISHEK VIJHANI

CWID- A20377670

1. Introduction

This project is based on the coursework of CS586 Software System Architecture. This Project includes
various patterns used in software design and implements them.

Overview of the Project

It includes a GasPump system which performs different operations and actions based on the user's interest
There are two GasPump Components: GasPump - 1 and GasPump - 2

GasPump - 1 Component Supports the following operations:

Activate (float a, float b) // the gas pump is activated where a is the price of the Regular gas // and b is the price of Super gas per gallon

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

Super() // Super gas is selected

Regular() // Regular gas is selected

StartPump() // start pumping gas

PumpGallon() // one gallon of gas is disposed

StopPump() // stop pumping gas

GasPump - 1 Component Supports the following operations:

Activate (int a, int b, int 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 Super gas per liter

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

Super() // Super gas is selected

StartPump() // start pumping gas

PumpLiter() // one liter of gas is disposed

Stop() // stop pumping gas

Receipt() // Receipt is requested

NoReceipt() // No receipt

Both GasPump components are state-based components and are used to control simple gas pumps. Users can pay by cash or a credit card. The gas pump may dispose different types of the gasoline. The price of the gasoline is provided when the gas pump is activated.

Goal of the Project

The goal of this project is to design two different GasPump components using the Model-Driven Architecture (MDA) and then implement these GasPump components based on this design.

Model Driven Architecture of the GasPump Components:

MDA-EFSM Events:

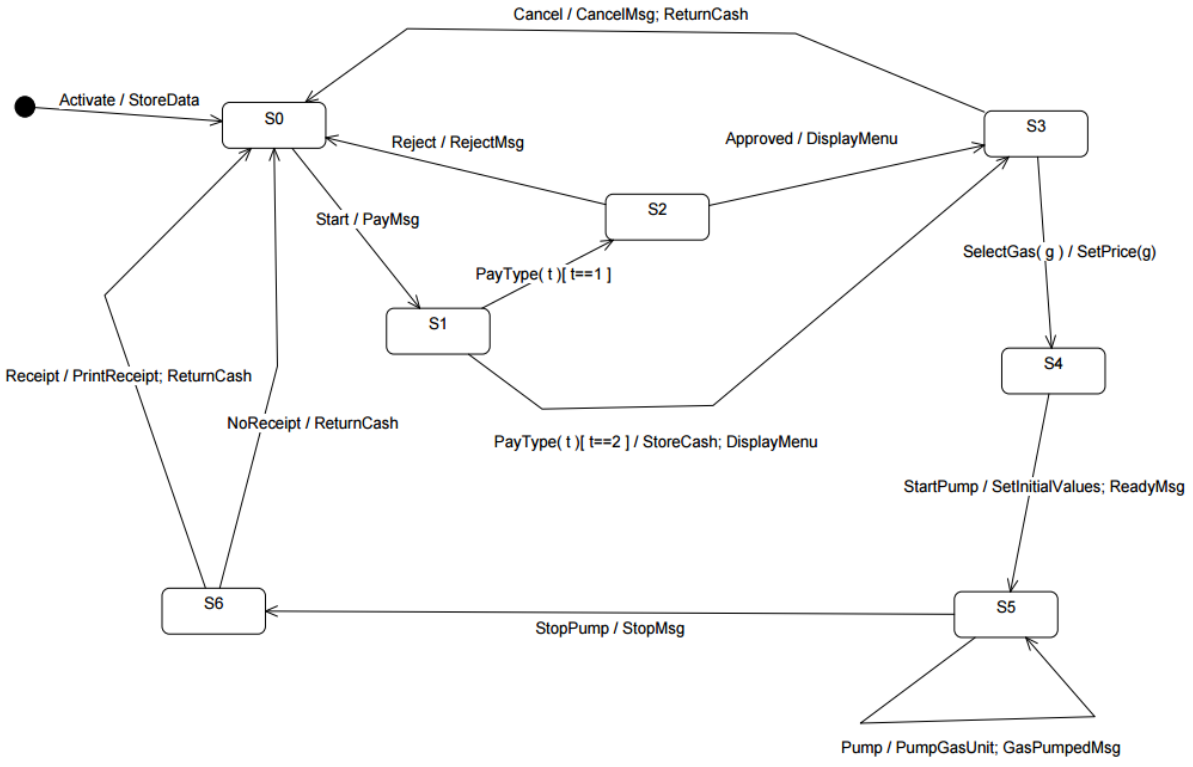
Activate()
Start()
PayType(int t) //credit: t=1; cash: t=2
Reject()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int g)

Receipt()
NoReceipt()

MDA-EFSM Actions:

StoreData // 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
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? msg (optionally)
PrintReceipt // print a receipt
CancelMsg // displays a cancellation message
ReturnCash // returns the remaining cash

A State Diagram of MDA-EFSM Component:



Pseudocode for all Operations of the Input Processor GasPump1 and GasPump2.

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

```
Approved()
{
m->Approved();
}
Super()
{
m->SelectGas(2)
}
Regular()
{
m->SelectGas(1)
}
StartPump()
{
m->StartPump();
}
PumpGallon()
{
m->Pump();
}
StopPump()
{
m->StopPump();
m->Receipt();
}
```

Notice:

m: is a pointer to the MDA-EFSM object

d: is a pointer to the Data Store object

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()
        }
}
Start()
{
m->Start();
}
PayCash(float c)
{
if (c>0)
        {
        d->temp_cash=c;
        m->PayType(2)
        }
}
Cancel()
{
```

```
m->Cancel();
}
Super()
{
m->SelectGas(2);
}
Premium()
{
m->SelectGas(3);
}
Regular()
{
m->SelectGas(1);
}
StartPump()
{
m->StartPump();
}
PumpLiter()
{
if (d->cashL+1)*d->price)
{
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

L: contains the number of liters already pumped

cash, L, 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 GasPump components

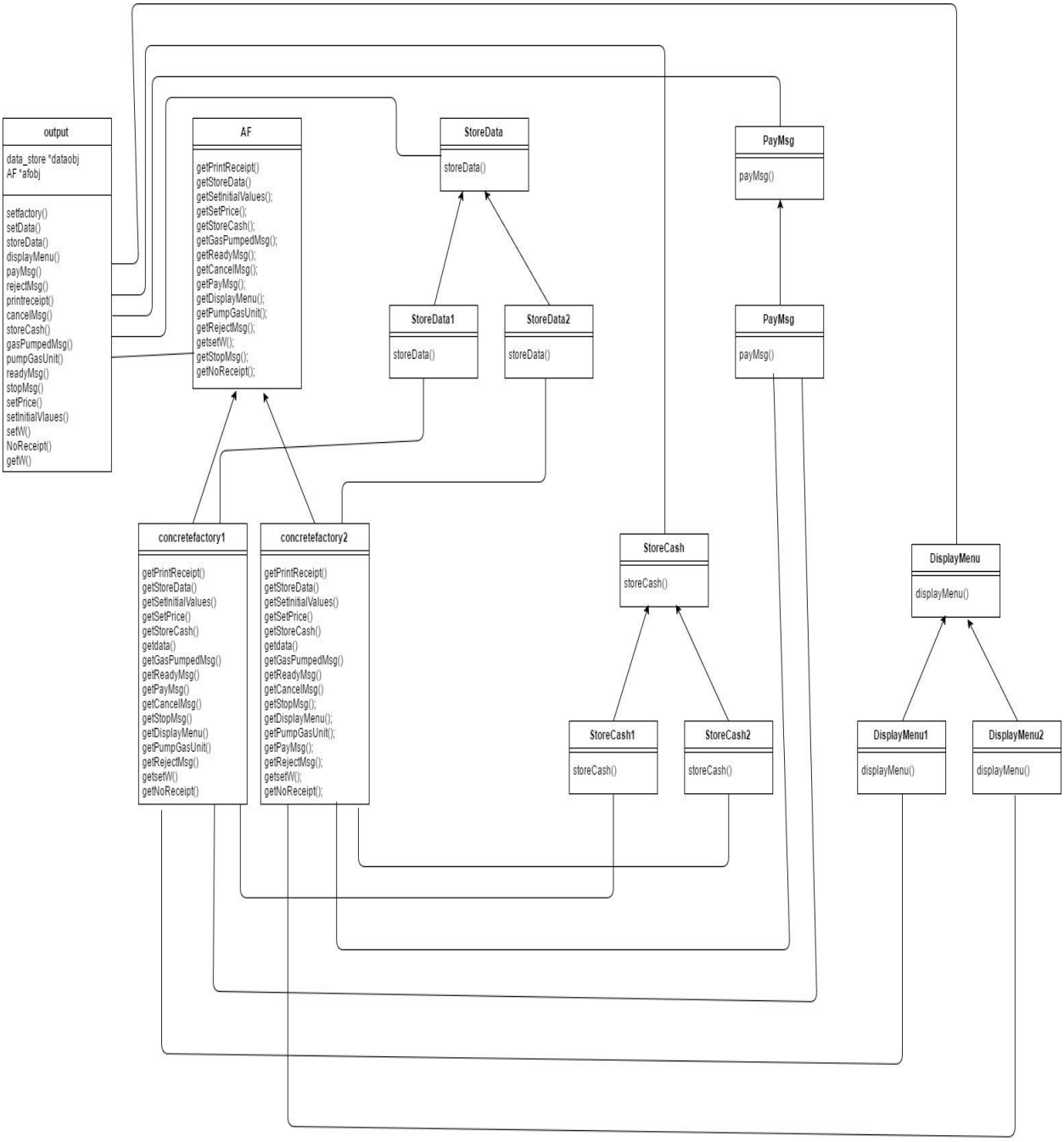
a) State Pattern

b) Strategy Pattern

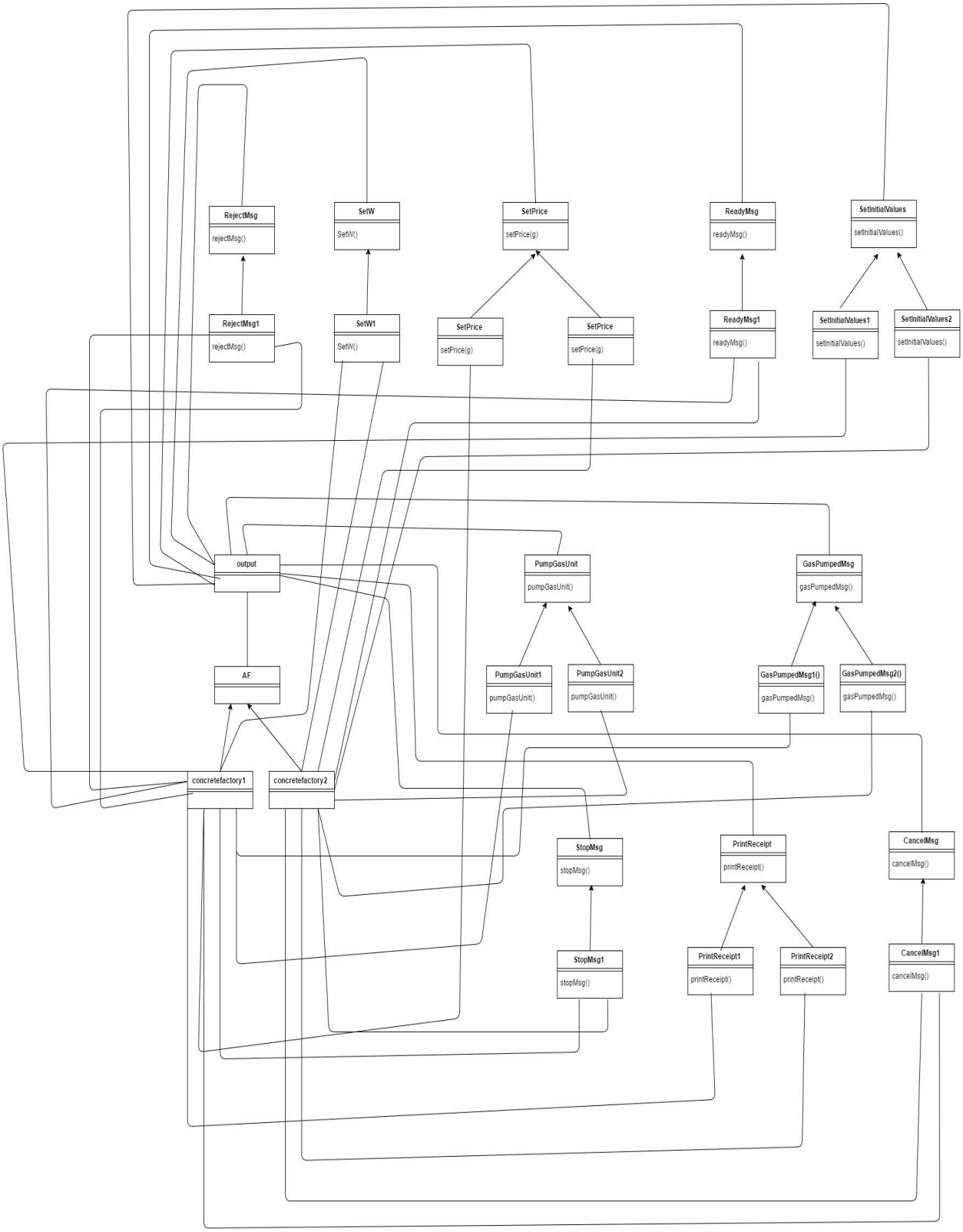
c) Abstract factory Pattern

State Pattern

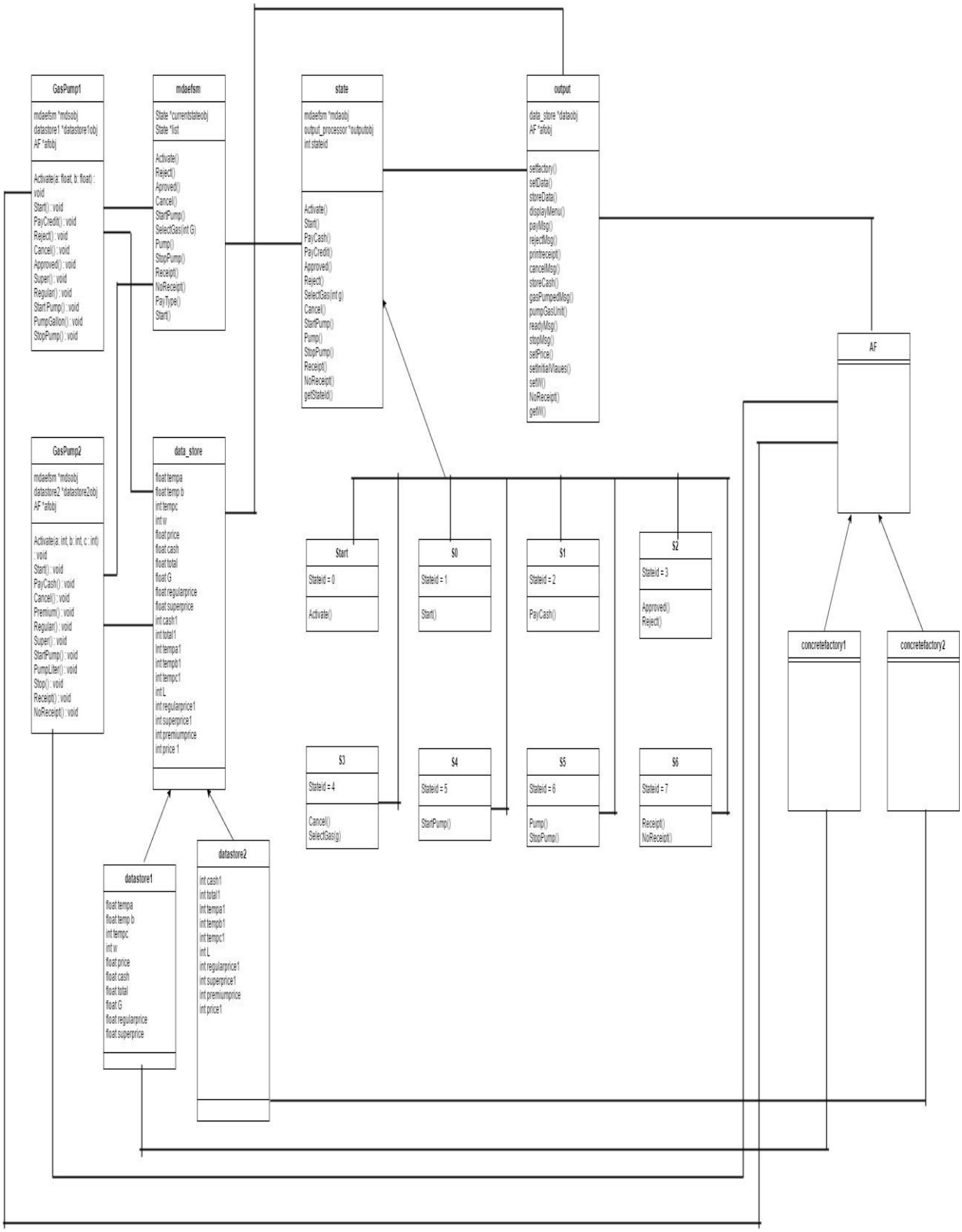
State Diagram



Strategy Pattern



Abstract Factory Pattern



avijhani@hawk.iit.edu
CWID – A20377670

3. Purpose of the Classes

Components	Responsibilities
GasPump1	GasPump1 class handles input events and executes appropriate actions from the input and call method of MDA_EFSM.
GasPump2	GasPump2 class handles input events and executes appropriate actions from the input and call method of MDA_EFSM.
data	It stores centralized dependent data of the system.
datastore1	It stores independent data for GasPump-1 component.
datastore2	It stores independent data for GasPump-2 component.
mdaefsm	MDA-EFSM will take care of changing states based on the actual events issued by user.
State	It is a part of state design pattern. State class is an interface which defines an action and concrete classes implement the state interface.
Start	This concrete class of state pattern activates a pump.
S0	This state will start pump.
S1	This state will check the payment method like pay by cash or credit card. After performing this operation, it will changed state from S1 to S2.
S2	This state performs approved and reject payment operations. After performing these operation, it will change state from S2 to S3.
S3	During this state, user can take decision to continue an operation by

	switching from state S3 to S4 or cancel the chain and get back to S0 state.
S4	It starts the pump to fill gas with initial value of gas unit and total prices and then switch to S5 state.
S5	It pumps gas in gallon or liter and increments counter of filled gas. After filling a gas, it stops a pump and redirect to next state S6.
S6	Its provide an option to user that he wants receipt or not. If yes, then system generate receipt with total unit of filled gas and prices and go back to S0 state. Otherwise it will go back to S0 state without generating any receipt.
AF	It provides definition of each concrete factory.
concretefactory1	It provides families of independent objects for GasPump1 component actions.
concretefactory2	It provides families of independent objects for GasPump2 component actions.
StoreData	It stores price(s) for the gas from the temporary data store
PayMsg	It displays a type of payment method
DisplayMenu	It displays a menu with a list of selections
StoreCash	It stores cash from the temporary data store
RejectMsg	It displays credit card not approved message
SetW	It sets a value for credit/cash flag
SetPrice	It sets price for the gas identified by g identifier
ReadyMsg	It displays the ready for pumping message

SetInitialValues	It sets value of G (or L) and total to 0
PumpGasUnit	It disposes unit of gas and counts # of units disposed
GasPumpedMsg	It displays the amount of disposed gas
StopMsg	It displays stop pump message and ask for receipt (optionally)
PrintReceipt	It prints a receipt with filled gas counter and total amount
CancelMsg	It displays a cancellation message if transaction has been cancelled

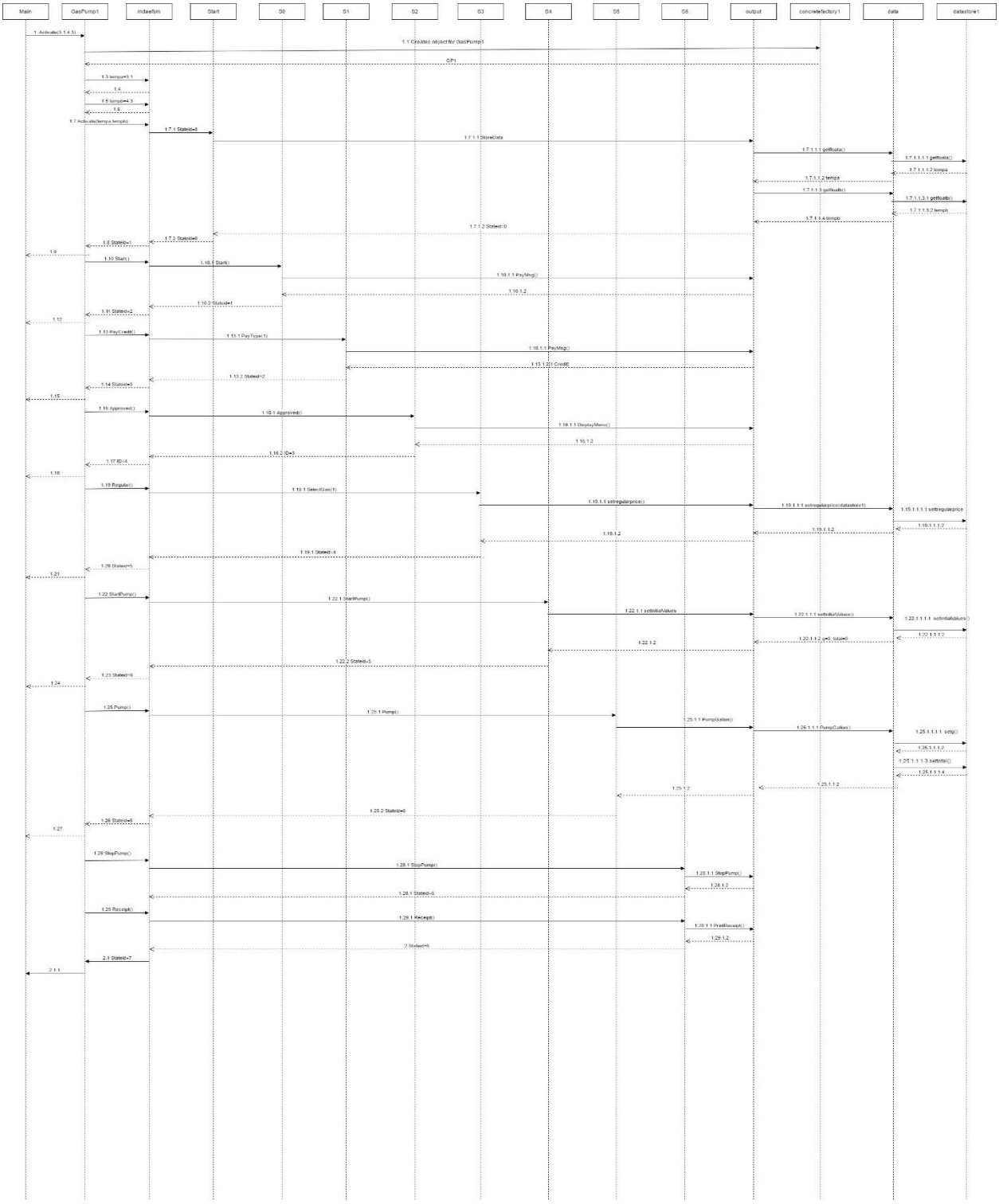
Responsibilities of each Operation Supported by each Class

Components	**Responsibilities**
Activate()	It activates a pump by taking price of the gas per gallon. For GasPump1, it takes two float value for regular and super gas. For GasPump2, it takes 3 integer values for regular and super/premium gas.
Start()	It starts pump and prints available payment options. For GasPump1, it prints payCredit option and For GasPump2, it prints payCash option only.
PayCredit()	It stores selected payment option and wait for an approval.
PayCash()	It stores selected payment option and store cash in temp value.
Approved()	It approves credit card if it is valid.
Reject()	It declines credit card option if it doesn't valid.
Cancel()	It cancels the transaction.
SelectGas(int g)	This operation set the final price of selected gas. It retrieves temporary

	value and stores in final price variable.
StartPump()	It initializes value of quantity(G or L) and total value for further pump operations.
Pump()	It pumps gas and count the quantity of filled gas. It also calculates total value of filled gas.
StopPump()	It stops pump and prints stop pump message. For GasPump1 and GasPump2, it also prints receipt with total value of filled gas.
Receipt()	It prints receipt with filled gas quantity and total values.
NoReceipt()	It will not generate the receipt

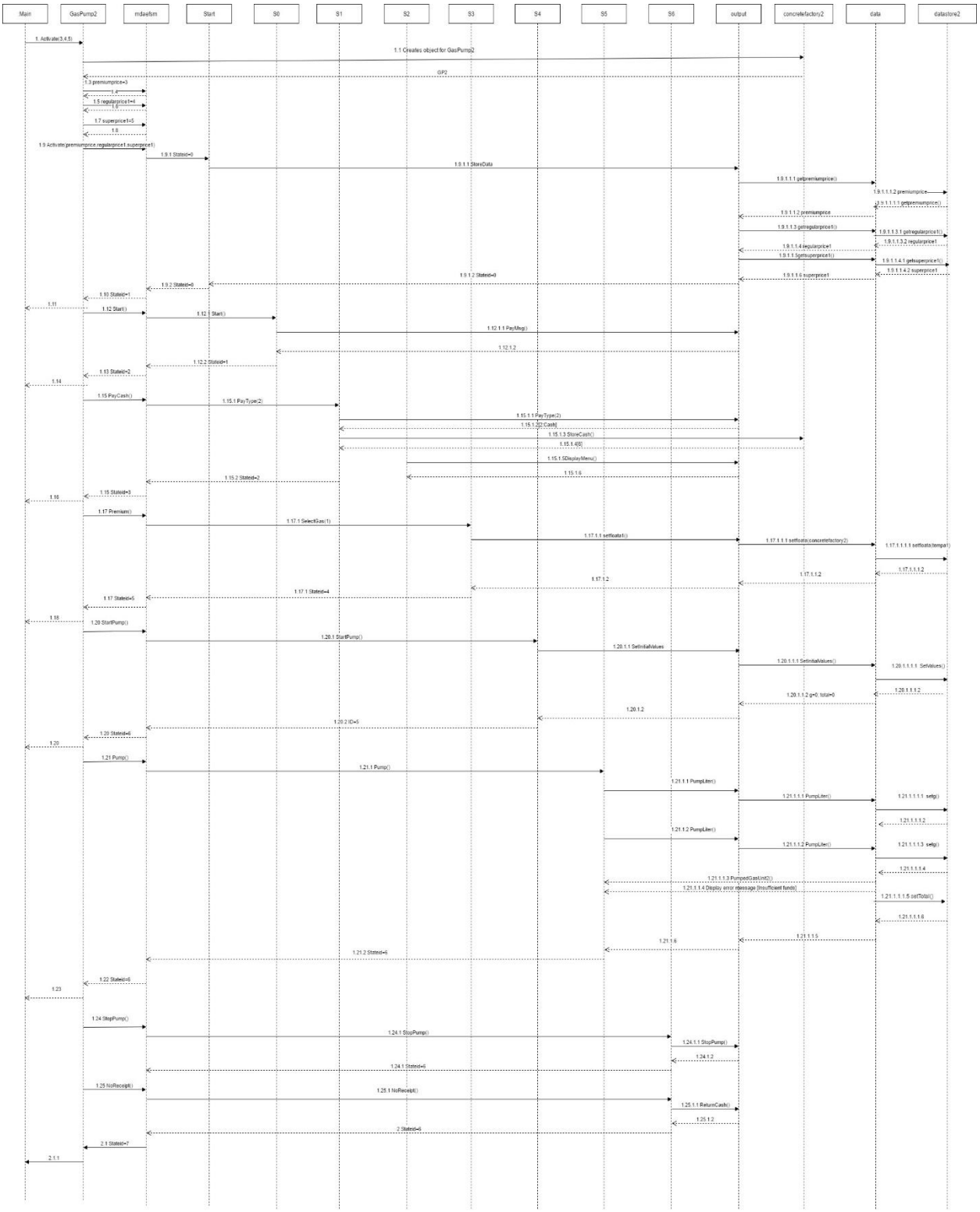
4. Sequence Diagram

Scenario-I should show how one gallon of Regular gas is disposed in GasPump-1, i.e., the following sequence of operations is issued: Activate(3.1, 4.3), Start(), PayCredit(), Approved(), Regular(), StartPump(), PumpGallon(), StopPump()



avijhani@hawk.iit.edu
CWID – A20377670

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), Start(), PayCash(6), Premium(), StartPump(), PumpLiter(), PumpLiter(), NoReceipt()



5. mdaefsm

This class implements the mdaefsm module of mdaefsm. It signifies the platform independent model component. It is accountable for invoking various actions based on the events triggered by the input processor. Implementation of mdaefsm is based on ***STATE PATTERN***. It stores reference to output_processor to invoke actions and it also stores reference to all the states of the GasPump system.

Output_processor

This class implements the output processor module of the MDA. It provides the model with an interface to invoke actions based on various events. ***STRATEGY PATTERN*** has been followed for invoking actions of GasPump1 or GsPump2. It stores references to all the actions (represented by different classes). Most of the actions have two executions. One specific to GasPump-1 and other specific to GasPump-2.

abstract_factory

This is an Abstract Factory class for different GasPump Components. It contains two implementations: concretefactory1 and concretefactory2. It follows the ***ABSTRACT FACTORY PATTERN***

6. Source Code

Package abstract_factory

Class AF.java

```
package abstract_factory;

import strategy.PayMsg;

import strategy.StoreCash;

import strategy.DisplayMenu;

import strategy.RejectMsg;

import strategy.SetW;

import strategy.SetPrice;

import strategy.ReadyMsg;

import strategy.SetInitialValues;

import strategy.PumpGasUnit;
```

```
import strategy.GasPumpedMsg;
import strategy.StopMsg;
import strategy.PrintReceipt;
import strategy.CancelMsg;
import strategy.StoreData;
import strategy.NoReceipt;
public abstract class AF {
   public abstract PrintReceipt getPrintReceipt();
   public abstract StoreData getStoreData();
   public abstract SetInitialValues getSetInitialValues();
   public abstract SetPrice getSetPrice();
   public abstract StoreCash getStoreCash();
   public abstract GasPumpedMsg getGasPumpedMsg();
   public abstract ReadyMsg getReadyMsg();
   public abstract CancelMsg getCancelMsg();
   public abstract PayMsg getPayMsg();
   public abstract DisplayMenu getDisplayMenu();
   public abstract PumpGasUnit getPumpGasUnit();
   public abstract RejectMsg getRejectMsg();
   public abstract SetW getsetW();
   public abstract StopMsg getStopMsg();
   public abstract NoReceipt getNoReceipt();  }
```

Class concretefactory1.java

```
package abstract_factory;
import strategy.StoreData;
import strategy.StoreData1;
```

```
import strategy.PayMsg;
import strategy.StoreCash;
import strategy.DisplayMenu;
import strategy.RejectMsg;
import strategy.SetW;
import strategy.SetPrice;
import strategy.ReadyMsg;
import strategy.SetInitialValues;
import strategy.PumpGasUnit;
import strategy.GasPumpedMsg;
import strategy.StopMsg;
import strategy.PrintReceipt;
import data_store.data;
import data_store.datastore1;
import strategy.CancelMsg;
import strategy.NoReceipt;
import strategy.NoReceipt1;
import strategy.PayMsg1;
import strategy.StoreCash1;
import strategy.DisplayMenu1;
import strategy.RejectMsg1;
import strategy.SetW1;
import strategy.SetPrice1;
import strategy.ReadyMsg1;
import strategy.SetInitialValues1;
import strategy.PumpGasUnit1;
import strategy.GasPumpedMsg1;
```

```
import strategy.StopMsg1;
import strategy.PrintReceipt1;
import strategy.CancelMsg1;

public class concretefactory1 extends AF
{
    datastore1 datastore1obj= new datastore1();
        public PrintReceipt getPrintReceipt()
   {
                PrintReceipt printobj =new PrintReceipt1();
                printobj.setdata(datastore1obj);
            return printobj;
   }
        public StoreData getStoreData()
   {
                StoreData storedataobj =new StoreData1();
                storedataobj.setdata(datastore1obj);
                return storedataobj;
   }
        public SetInitialValues getSetInitialValues()
   {
                SetInitialValues intialvalueobj = new SetInitialValues1();
      intialvalueobj.setdata(datastore1obj);
      return intialvalueobj;
   }
        public SetPrice getSetPrice()
   {
```

```
            SetPrice setpriceobj = new SetPrice1();
        setpriceobj.setdata(datastore1obj);
        return setpriceobj;
}
    public StoreCash getStoreCash()
 {
            StoreCash storecashobj = new StoreCash1();
        storecashobj.setdata(datastore1obj);
        return storecashobj;
}
    public data getdata()
    {
        return datastore1obj;
    }

     public GasPumpedMsg getGasPumpedMsg()
{
            GasPumpedMsg gaspumpedmsgobj = new GasPumpedMsg1();
        gaspumpedmsgobj.setdata(datastore1obj);
        return gaspumpedmsgobj;

}
     public ReadyMsg getReadyMsg()
{
            ReadyMsg readymsgobj = new ReadyMsg1();
        readymsgobj.setdata(datastore1obj);
        return readymsgobj;
```

```
}
        public PayMsg getPayMsg()
{
                PayMsg paymsgobj = new PayMsg1();
    paymsgobj.setdata(datastore1obj);
    return paymsgobj;

}
        public CancelMsg getCancelMsg()
        {
                CancelMsg ccancelmsgobj = new CancelMsg1();
    ccancelmsgobj.setdata(datastore1obj);
    return ccancelmsgobj;
        }
        public StopMsg getStopMsg()
        {
                StopMsg stopmsgobj = new StopMsg1();
    stopmsgobj.setdata(datastore1obj);
    return stopmsgobj;
        }
        public DisplayMenu getDisplayMenu()
         {
                DisplayMenu displaymenuobj = new DisplayMenu1();
    displaymenuobj.setdata(datastore1obj);
    return displaymenuobj;
         }
         public PumpGasUnit getPumpGasUnit()
```

```
{
        PumpGasUnit pumpgasunitobj = new  PumpGasUnit1();
    pumpgasunitobj.setdata(datastore1obj);
    return pumpgasunitobj;
}

public RejectMsg getRejectMsg()
{
       RejectMsg rejectmsgobj = new RejectMsg1();
    rejectmsgobj.setdata(datastore1obj);
    return rejectmsgobj;
}

public SetW getsetW()
{
        SetW setwobj = new SetW1();
    setwobj.setdata(datastore1obj);
    return setwobj;
}
public NoReceipt getNoReceipt()
{
       NoReceipt noreceiptobj = new NoReceipt1();
    noreceiptobj.setdata(datastore1obj);
    return noreceiptobj;
}

}
```

Class concretefactory2.java

```
package abstract_factory;


import strategy.StoreData;
import strategy.StoreData2;
import strategy.PayMsg;
import strategy.StoreCash;
import strategy.DisplayMenu;
import strategy.RejectMsg;
import strategy.SetW;
import strategy.SetPrice;
import strategy.ReadyMsg;
import strategy.SetInitialValues;
import strategy.PumpGasUnit;
import strategy.GasPumpedMsg;
import strategy.StopMsg;
import strategy.PrintReceipt;
import data_store.data;


import data_store.datastore2;
import strategy.CancelMsg;
import strategy.NoReceipt;
import strategy.NoReceipt1;
import strategy.PayMsg1;
import strategy.StoreCash2;
import strategy.DisplayMenu2;
import strategy.RejectMsg1;
```

```
import strategy.SetW1;
import strategy.SetPrice2;
import strategy.ReadyMsg1;
import strategy.SetInitialValues1;
import strategy.PumpGasUnit2;
import strategy.GasPumpedMsg2;
import strategy.StopMsg1;
import strategy.PrintReceipt2;
import strategy.CancelMsg1;

public class concretefactory2 extends AF
{
    datastore2 datastore2obj= new datastore2();
        public PrintReceipt getPrintReceipt()
  {

            PrintReceipt printrecieptobj =new PrintReceipt2();
            printrecieptobj.setdata(datastore2obj);

        return printrecieptobj;
  }
        public StoreData getStoreData()
  {
            StoreData storedataobj =new StoreData2();
            storedataobj.setdata(datastore2obj);
            return storedataobj;
  }
```

```
        public SetInitialValues getSetInitialValues()
{
                SetInitialValues initialvalueobj = new SetInitialValues1();
                initialvalueobj.setdata(datastore2obj);
   return initialvalueobj;
}
        public SetPrice getSetPrice()
{
                SetPrice setpriceobj = new SetPrice2();
                setpriceobj.setdata(datastore2obj);
            return setpriceobj;
}
        public StoreCash getStoreCash()
 {
                  StoreCash storecashobj = new StoreCash2();
                  storecashobj.setdata(datastore2obj);
            return storecashobj;
 }
        public data getdata()
        {
            return datastore2obj;
        }

         public GasPumpedMsg getGasPumpedMsg()
         {
                 GasPumpedMsg gaspumpdmsgobj = new GasPumpedMsg2();
            gaspumpdmsgobj.setdata(datastore2obj);
```

```
        return gaspumpdmsgobj;

    }
    public ReadyMsg getReadyMsg()
    {
            ReadyMsg readymsgobj = new ReadyMsg1();
        readymsgobj.setdata(datastore2obj);
        return readymsgobj;

    }
    public PayMsg getPayMsg()
    {
            PayMsg paymsgobj = new PayMsg1();
paymsgobj.setdata(datastore2obj);
return paymsgobj;

    }
    public CancelMsg getCancelMsg()
    {
            CancelMsg cancelmsgobj = new CancelMsg1();
            cancelmsgobj.setdata(datastore2obj);
return cancelmsgobj;
    }
    public StopMsg getStopMsg()
    {
            StopMsg stopmsgobj = new StopMsg1();
            stopmsgobj.setdata(datastore2obj);
```

```
return stopmsgobj;
    }
    public DisplayMenu getDisplayMenu()
    {
            DisplayMenu displaymenuobj = new DisplayMenu2();
            displaymenuobj.setdata(datastore2obj);
return displaymenuobj;
     }
      public PumpGasUnit getPumpGasUnit()
      {
            PumpGasUnit pumpgasunitobj = new  PumpGasUnit2();
            pumpgasunitobj.setdata(datastore2obj);
          return pumpgasunitobj;
      }

      public RejectMsg getRejectMsg()
      {
           RejectMsg rejectmsgobj = new RejectMsg1();
           rejectmsgobj.setdata(datastore2obj);
          return rejectmsgobj;
      }
      public SetW getsetW()
      {
            SetW setwobj = new SetW1();
            setwobj.setdata(datastore2obj);
          return setwobj;
      }
```

```
        public NoReceipt getNoReceipt()
        {
            NoReceipt noreceiptobj = new NoReceipt1();
            noreceiptobj.setdata(datastore2obj);
           return noreceiptobj;
        }

}
```

Package data_store

Class data.java

```
package data_store;

public abstract class data
{
                public  void setinta(int a)
                {}
                public  void setintc(int a)
                {}
                public void setfloatc(float c)
                {}
                public float getfloatc()
                {       return 0;  }
                public void setfloatc1(int c)
                {}
                public int getfloatc1()
                {       return 0;  }
                public  void setintW(int a)
                {}
                public  int getintW()
                { return 0; }
                public int getinta()
                { return 0; }
                public  int getintc()
                {       return 0; }
                public  void setprice(float a)
                {}
                public  void setprice1(int a)
```

```
{}
public  int getprice1()
{          return 0; }
public  void setcash(float a)
{}
public  int getcash1()
{          return 0; }
public  void setcash1(int c)
{}
public  void setG(float a)
{}
public  float getprice()
{          return 0; }
public  float getcash()
{ return 0; }
public  float getG()
{ return 0; }
public float gettotal()
{ return 0; }
public  void settotal(float a)
{}
public int gettotal1()
{ return 0; }
public  void settotal1(int a)
{}
 public  void setfloata(float a)
 {}
public  void setfloatb(float a)
{}
public  float getfloata()
{          return 0; }
public float getfloatb()
{          return 0; }
public void setsuperprice(float a)
{}
public float getsuperprice()
{          return 0; }
public void setsuperprice1(int c)
{}
public int getsuperprice1()
{          return 0; }

public void setregularprice(float a)
```

```
        {}
        public float getregularprice()
        {       return 0; }
        public void setpremiumprice(int a)
        {}
        public void setregularprice1(int a)
        {}
        public int getregularprice1()
        {       return 0; }
        public int getpremiumprice()
        { return 0; }
        public int getL()
        {return 0; }
        public void setL(int a)
        {}
        public int getfloata1() {

                return 0;
        }
        public void setfloata1(int a)
        {}
        public int getfloatb1() {

                return 0;
        }
        public void setfloatb1(int b)
        {}
}
```

Class datastore1.java

```
package data_store;

public class datastore1 extends data {


        static float tempa;
        static float tempb;
        static int tempc;
        static int W;
        static float price;
        static float cash;
        static float total;
        static float G;
```

```
static float regularprice;
static float superprice;
public void setfloata(float a)
{
        tempa =a;
}
public void setfloatb(float b)
{
        tempb=b;
}
public float getfloata()
{
        return tempa;
}
public float getfloatb()
{
        return tempb;
}
public void setintc(int c)
{
         tempc =c;
}
public int getintc()
{
        return tempc;
}

public void settotal(float c)
{
         total =c;
}
public float gettotal()
{
        return total;
}

public void setprice(float y)
{
        price=y;
}
public float getprice()
{
  return price;
```

```
        }
        public void setcash(float a)
        {
                cash =a;
        }
        public float getcash()
        {
                return cash;
        }
    public void setG(float y)
        {
           G=y;
        }
    public float getG()
        {
           return G;
        }
    public void setsuperprice(float y)
        {
                superprice=y;
        }
    public float getsuperprice()
        {
          return superprice;
        }
    public void setregularprice(float y)
        {
                regularprice=y;
        }
    public float getregularprice()
        {
          return regularprice;
        }
    public void setintW(int t)
        {
                W =t;
        }
        public int getintW()
        {
                return W;
        }
}
```

Class datastore2.java

```
package data_store;

public class datastore2 extends data
{
	static int tempa1;
	static int tempb1;
	static int tempc1;
	static int L;
	static int regularprice1;
	static int premiumprice;
	static int superprice1;
	static int cash1;
	static int total1;
	static int price1;
	public int getprice1()
	{
		return price1;
	}
	 public void setprice1(int y)
	{
			price1=y;
	}
	 public void setfloata1(int a)
	{
		tempa1 =a;
	}
	 public int getfloata1()
		{
			return tempa1;
		}
	public void setfloatb1(int b)
	{
		tempb1=b;
	}
	public int getfloatb1()
	{
		return tempb1;
	}
	public void setfloatc1(int c)
	{
		tempc1=c;
	}
```

```
public int getfloatc1()
{
        return tempc1;
}

public void setL(int a)
{
         L=a;
}
public int getL()
{
         return L;
}
 public void setpremiumprice(int a)
{
                premiumprice=a;
}
public int getpremiumprice()
{
        return premiumprice;
}
public void setregularprice1(int a)
{
                regularprice1=a;
}
public int getregularprice1()
{
        return regularprice1;
}
public void setsuperprice1(int c)
{
                superprice1=c;
}
public int getsuperprice1()
{
        return superprice1;
}
public void setcash1(int a)
{
                cash1=a;
}
public int getcash1()
{
```

```
		return cash1;
	}
	public int gettotal1()
	{
		return total1;
	}
	public void settotal1(int a)
	{
			total1=a;
	}

}
```

Package driver

Class maindriver.java

```
package driver;


import mdaefsm.mdaefsm;

import output_processor.output;

import java.io.IOException;

import java.util.Scanner;

import abstract_factory.AF;

import abstract_factory.concretefactory1;

import abstract_factory.concretefactory2;

import data_store.data;

import input_processor.GasPump1;

import input_processor.GasPump2;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import state.Start;

import state.S0;
```

```
import state.S1;
import state.S2;
import state.S3;
import state.S4;
import state.S5;
import state.S6;



public class maindriver
{
        static Scanner input=new Scanner(System.in);
        static BufferedReader buf=new BufferedReader(new InputStreamReader(System.in));
        public static void main(String[] args) throws NumberFormatException, IOException
  {
    System.out.println("\n ***Gas Pump Model*** ");
                System.out.println("\n          Select the Gas Pump ");
                System.out.println("\n 1. GasPump-1 ");
                System.out.println("\n 2. GasPump-2 ");
                System.out.println("\n Enter your choice ");
                int choice = input.nextInt();
    switch(choice)
    {
          case 1:
          {
                GasPump1 gaspump1obj = new GasPump1(); //gaspump1 represents the object
of Gas Pump 1
                Start s0obj  = new Start();//s0obj represents the object Of state 0
```

```
S0 s1obj = new S0();//s1obj represents the object Of state 1
S1 s2obj = new S1();//s2obj represents the object Of state 2
S2 s3obj = new S2();//s3obj represents the object Of state 3
S3 s4obj = new S3();//s4obj represents the object Of state 4
S4 s5obj = new S4();//s5obj represents the object Of state 5
S5 s6obj = new S5();//s6obj represents the object Of state 6
S6 s7obj = new S6();//s7obj represents the object Of state 7
mdaefsm mdaobj = new mdaefsm();
output outputobj = new output();
concretefactory1 concretefactory1obj = new concretefactory1();
data d_o;
d_o = concretefactory1obj.getdata();
gaspump1obj.setMDA(mdaobj); //Setting the mda with the mda object created
gaspump1obj.setfactory(concretefactory1obj);// Setting the object of the concrete factory 1
gaspump1obj.setdata(d_o);
s0obj.setoutput(outputobj);
s0obj.setStateId(0);
s1obj.setoutput(outputobj);
s1obj.setStateId(1);
s2obj.setoutput(outputobj);
s2obj.setStateId(2);
s3obj.setoutput(outputobj);
s3obj.setStateId(3);
s4obj.setoutput(outputobj);
s4obj.setStateId(4);
s5obj.setoutput(outputobj);
```

```
s5obj.setStateId(5);
s6obj.setoutput(outputobj);
s6obj.setStateId(6);
s7obj.setoutput(outputobj);
s7obj.setStateId(7);
outputobj.setData(d_o);
outputobj.setfactory(concretefactory1obj);
mdaobj.setStates(s0obj);
mdaobj.setStatesList(s0obj,s1obj,s2obj,s3obj,s4obj,s5obj,s6obj,s7obj);
String input=null;
int ch;
while(true)
{
    System.out.println("\n\n Please Select the option from the Available Choices for GasPump-1 ");
                                        System.out.println("\n 0.\t Activate (Regular,Super) ");
                                        System.out.println("\n 1.\t Start ");
                                        System.out.println("\n 2.\t PayCredit ");
                                        System.out.println("\n 3.\t Reject");
                                        System.out.println("\n 4.\t Cancel ");
                                        System.out.println("\n 5.\t Approved  ");
                                        System.out.println("\n 6.\t Super ");
                                        System.out.println("\n 7.\t Regular ");
                                        System.out.println("\n 8.\t StartPump");
                                        System.out.println("\n 9.\t PumpGallon");
                                        System.out.println("\n 10.\t StopPump");
                                        System.out.println("\n 11.\t Quit");
```

```
//                                      System.out.println("\n 12.\t NoReceipt()");
                                        System.out.println("\n Press any other key to exit \n\n ");
                                        input=buf.readLine();
            ch=Integer.parseInt(input);
            switch(ch)
            {
                                                case 0: System.out.println(" \n\n Enter the value of a to activate");
                        float a=Float.parseFloat(buf.readLine());
                        System.out.println("\n\n Enter the value of b to activate");
                        float b=Float.parseFloat(buf.readLine());
                        gaspump1obj.Activate(a,b);      // Calls method Activate in Gas Pump 1
                        break;

                                                case 1: gaspump1obj.Start();      // Calls method Start in Gas Pump 1
                    break;

                                                case 2: gaspump1obj.PayCredit();  // Calls method PayCredit in Gas Pump 1
                        break;

                                                case 3: gaspump1obj.Reject();     // Calls method Reject in Gas Pump 1
                    break;
```

```
                                        case 4: gaspump1obj.Cancel();      // Calls
method Reject in Gas Pump 1
                                                break;

                                        case 5: gaspump1obj.Approved();    // Calls
method Approved in Gas Pump 1
                                                break;

                                        case 6:gaspump1obj.Super();        // Calls
method Super in Gas Pump 1
                                                break;

                                        case 7:gaspump1obj.Regular();      // Calls
method Regular in Gas Pump 1
                                                break;

                                        case 8: gaspump1obj.StartPump();   // Calls
method StartPump in Gas Pump 1
                                                break;

                                        case 9: gaspump1obj.PumpGallon();  // Calls
method PumpGallon in Gas Pump 1
                                                break;

                                        case 10: gaspump1obj.StopPump();   // Calls
method StopPump in Gas Pump 1
                                                        gaspump1obj.Receipt();     //
Calls method Receipt in Gas Pump 1
                                                        break;
```

```
                        case 11:System.exit(0);      // Calls method Receipt in Gas Pump 1
                          break;
//
//                        case 12:gaspump1obj.NoReceipt();   // Calls method NoReceipt in Gas Pump
1
//                        break;
                      default:
                   System.out.println("\n Please enter Correct Option ");
               }
            }
           }

          case 2:
          {
              GasPump2 gaspump2obj = new GasPump2(); //gaspumpobj2 represents the object of
Gas Pump 2
                   Start s0obj  = new Start();//s0obj represents the object Of state 0
               S0 s1obj = new S0();//s1obj represents the object Of state 1
               S1 s2obj = new S1();//s2obj represents the object Of state 2
               S2 s3obj = new S2();//s3obj represents the object Of state 3
               S3 s4obj = new S3();//s4obj represents the object Of state 4
               S4 s5obj = new S4();//s5obj represents the object Of state 5
               S5 s6obj = new S5();//s6obj represents the object Of state 6
               S6 s7obj = new S6();//s7obj represents the object Of state 7
               mdaefsm mdaobj = new mdaefsm();
               output outputobj = new output();
               concretefactory2 concretefactory2obj = new concretefactory2();
               data d_o;
```

```
d_o = concretefactory2obj.getdata();
gaspump2obj.setMDA(mdaobj);//set the mda with the mda object created
gaspump2obj.setfactory(concretefactory2obj);
gaspump2obj.setdata(d_o);
s0obj.setoutput(outputobj);
s0obj.setStateId(0);
s1obj.setoutput(outputobj);
s1obj.setStateId(1);
s2obj.setoutput(outputobj);
s2obj.setStateId(2);
s3obj.setoutput(outputobj);
s3obj.setStateId(3);
s4obj.setoutput(outputobj);
s4obj.setStateId(4);
s5obj.setoutput(outputobj);
s5obj.setStateId(5);
s6obj.setoutput(outputobj);
s6obj.setStateId(6);
s7obj.setoutput(outputobj);
s7obj.setStateId(7);
outputobj.setData(d_o);
outputobj.setfactory(concretefactory2obj);
mdaobj.setStates(s0obj);
mdaobj.setStatesList(s0obj,s1obj,s2obj,s3obj,s4obj,s5obj,s6obj,s7obj);
String input=null;
int ch;
while(true)
```

```
        {
            System.out.println("\n Select your option from the Available choices for GasPump-2 ");
                                        System.out.println("\n 0.\t Activate (Regular, Premium, Super)");
                                        System.out.println("\n 1.\t Start()");
                                        System.out.println("\n 2.\t PayCash()");
                                        System.out.println("\n 3.\t Cancel()");
                                        System.out.println("\n 4.\t Premium()");
                                        System.out.println("\n 5.\t Regular()");
                                        System.out.println("\n 6.\t Super()");
                                        System.out.println("\n 7.\t StartPump()");
                                        System.out.println("\n 8.\t PumpLiter()");
                                        System.out.println("\n 9.\t Stop()");
                                        System.out.println("\n 10.\t Receipt()");
                                        System.out.println("\n 11.\t NoReceipt()");
                                        System.out.println("\n 12.\t Quit");
                                        System.out.println("\n Press any other key to exit \n\n ");
                                        input=buf.readLine();
            ch=Integer.parseInt(input);
            switch(ch)
            {
              case 0: System.out.println(" \n\n Enter the value of a to activate");
                      int a=Integer.parseInt(buf.readLine());
                      System.out.println(" \n\n Enter the value of b to activate");
                      int b=Integer.parseInt(buf.readLine());
                      System.out.println(" \n\n Enter the value of c to activate");
```

```
                int c=Integer.parseInt(buf.readLine());
                gaspump2obj.Activate(a,b,c);      //Calls method Activate
                break;

case 1: gaspump2obj.Start();   // Calls method Start
                break;

case 2: System.out.println("\n Enter the cash amount ");
                int c2=Integer.parseInt(buf.readLine());
                gaspump2obj.PayCash(c2);  //Calls method PayCash
                break;

case 3: gaspump2obj.Cancel();   //Calls method Cancel
                break;

case 4: gaspump2obj.Premium();  //Calls method Premium
                break;

case 5: gaspump2obj.Regular();   //Calls method Regular
                break;

case 6: gaspump2obj.Super();   //Calls method Super
                        break;

case 7: gaspump2obj.StartPump();   //Calls method StartPump
                break;
```

```
                case 8:gaspump2obj.PumpLiter();    //Calls method PumpLiter
                        break;

                case 9: gaspump2obj.Stop();       //Calls method Stop
                        break;

                case 10:gaspump2obj.Receipt();     //Calls method Receipt
                        break;

                case 11:gaspump2obj.NoReceipt();    //Calls method NoReceipt
                        break;
                case 12:System.exit(0);
                                        break;
                default:
                        System.out.println("\n Please Enter Correct Option. \n ");
              }
            }
          }
          }
    }
}
```

Package input_processor

Class GasPump1.java

```
package input_processor;

import abstract_factory.AF;

import data_store.data;

import mdaefsm.mdaefsm;
```

```
public class GasPump1
{
	AF afobj;
	mdaefsm mdaobj;
	data dataobj;
	public void setMDA(mdaefsm x)
	{
		mdaobj = x;
	}
	public void setdata(data x)
	{
		dataobj = x;
	}
	public void setfactory(AF x)
	{
		afobj = x;
	}
  public void Activate(float a,float b)
  {
	if((a>0)&&(b>0))
	{
      dataobj.setfloata(a);
      dataobj.setfloatb(b);
      mdaobj.Activate();

	}
  }
```


avijhani@hawk.iit.edu
CWID – A20377670

```
public void Start()
{
   mdaobj.Start();
}
public void PayCredit()
{
   mdaobj.PayCredit();
}
public void Reject()
{
   mdaobj.Reject();
}
public void Cancel()
{
   mdaobj.Cancel();
}
public void Approved()
{
  mdaobj.Approved();

}
public void Super()
{
        mdaobj.SelectGas(2);
}
public void Regular()
{
```

```
        mdaobj.SelectGas(1);
    }
     public void StartPump()
    {
       mdaobj.StartPump();
    }
    public void PumpGallon()
    {
        mdaobj.Pump();
    }
    public void StopPump()
    {
       mdaobj.StopPump();

    }
    public void Receipt()
    {
        mdaobj.Receipt();
    }
    public void NoReceipt()
    {
        mdaobj.NoReceipt();
    }
}
```

Class GasPump2.java

```
package input_processor;
import abstract_factory.AF;
import data_store.data;
import mdaefsm.mdaefsm;
public class GasPump2
{
        AF afobj;
        mdaefsm mdaobj;
        data dataobj;

   public void setMDA(mdaefsm x)
        {
                mdaobj = x;
        }
        public void setdata(data x)
        {
                dataobj = x;
        }
        public void setfactory(AF x)
        {
                afobj = x;
        }
   public void Activate(int a,int b,int c)
   {
        if((a>0)&&(b>0)&&(c>0))
        {
      dataobj.setfloata1(a);
```

```
        dataobj.setfloatb1(b);
        dataobj.setfloatc1(c);
        mdaobj.Activate();

            }
    }
    public void Start()
    {
       mdaobj.Start();

    }
    public void PayCash(int c)
    {
            if(c>0)
            {
                    dataobj.setfloatc1(c);

            mdaobj.PayCash();
            }
    }
    public void Cancel()
    {
       mdaobj.Cancel();
    }
    public void Premium()
    {
       mdaobj.SelectGas(2);
```

```
}
public void Regular()
{
        mdaobj.SelectGas(1);
}
public void Super()
{
        mdaobj.SelectGas(3);
}
 public void StartPump()
{
   mdaobj.StartPump();
}
public void PumpLiter()
{

        int cash = dataobj.getcash1();

        int price =dataobj.getprice1();

        int L = dataobj.getL();
        if(cash<(L+1)*price)
        {
                mdaobj.StopPump();
        }
        else
        {
```

```
                mdaobj.Pump();
        }
    }
    public void Stop()
    {
      mdaobj.StopPump();
    }
    public void Receipt()
    {
        mdaobj.Receipt();
    }
    public void NoReceipt()
    {
        mdaobj.NoReceipt();
    }
}
```

Package mdaefsm

Class mdaefsm.java

```
package mdaefsm;
import strategy.*;
import data_store.data;
import data_store.datastore2;
import output_processor.output;
import state.State;
public class mdaefsm extends PrintReceipt2
{
```

```
        int total1=0,cash=0,cr=0;


    State currentstateobj; //currentstateobj state is used to point to the current state object
    State[] list = new State[8];
    public void setStates(State a)
    {
    currentstateobj = a;
    }
    public void setStatesList( State a1, State a2, State a3, State a4, State a5, State a6,State a7,State
a8)//setting the states
    {
            list[0] = a1; //start state
            list[1] = a2; //S0 state
            list[2] = a3; //S1 state
            list[3] = a4; //S2 state
            list[4] = a5; //S3  state
            list[5] = a6; //S4 state
            list[6] = a7; //S5 state
            list[7] = a8; //S6 state
    }
    public void Activate()
    {
            int cur = currentstateobj.getStateId();
            System.out.print(cur);
            switch(cur)
            {
```

```
        case 0:
        {
                System.out.println("\n Gas Pump is now Activated");
                System.out.println("\n Changing State from Start to S0");
                currentstateobj.Activate();
         currentstateobj = list[1];
         break;
        }
        case 1: break;
        case 2: break;
        case 3: break;
        case 4: break;
        case 5: break;
        case 6: break;
        case 7: break;
        }
  }
  public void Start()
  {
        int cur = currentstateobj.getStateId();
        switch(cur)
        {
     case 0:break;
     case 1:
     {
        System.out.println("\n Gas Pump has now Started");
        System.out.println("\n Changing State from S0 to S1");
```

```
            currentstateobj.Start();
           currentstateobj = list[2];
           break;
       }
       case 2:  break;
       case 3:  break;
       case 4:  break;
       case 5:  break;
       case 6:  break;
       case 7:  break;

           }
   }

   public void PayCash()
   {
   int cur = currentstateobj.getStateId();
   switch(cur)
   {
       case 0:  break;
       case 1:  break;
       case 2:
       {
           System.out.println("\n You Selected Pay by Cash Option");
           System.out.println("\n Changing State from S1 to S3");
           currentstateobj.PayCash();
          currentstateobj = list[4];
```

```
    }
    case 3:  break;
    case 4:  break;
    case 5:  break;
    case 6:  break;
    case 7: break;
        }
}
public void PayCredit()
{
int cur = currentstateobj.getStateId();
switch(cur)
{
    case 0: break;
    case 1:  break;
    case 2:
    {
        System.out.println("\n You Selected Pay by Credit Card Option");
        System.out.println("\n Changing State from S1 to S2");
        currentstateobj.PayCredit();
       currentstateobj = list[3];
    }
    case 3:  break;
    case 4:  break;
    case 5:  break;
    case 6:  break;
    case 7:  break;
```

```
    }
}


    public void Approved()
    {
    int cur = currentstateobj.getStateId();
    switch(cur)
    {
        case 0:  break;
        case 1:  break;
        case 2:  break;
        case 3:
        {
            System.out.println("\n Your Credit Card has been Approved");
            System.out.println("\n Changing State from S2 to S3");
            System.out.println("\n Please Select the Gas");
            currentstateobj.Approved();
          currentstateobj = list[4];
          break;
        }
        case 4: break;
        case 5: break;
        case 6: break;
        case 7: break;
```

```
    }
}

public void Reject()
{
    int cur = currentstateobj.getStateId();
    switch(cur)
    {
        case 0:  break;
        case 1:  break;
        case 2:  break;
        case 3:
        {
            System.out.println("\n Your Credit Card has been Rejected");
            System.out.println("\n Changing State from S2 to S0");
            currentstateobj.Reject();
          currentstateobj = list[1];
          break;
        }
        case 4:break;
        case 5:break;
        case 6:break;
        case 7:break;
    }
}

public void Cancel()
```

```
{
int cur = currentstateobj.getStateId();
switch(cur)
{
   case 0:  break;
   case 1:  break;
   case 2:  break;
   case 3:  break;
   case 4:
   {
          currentstateobj.Cancel();
      currentstateobj = list[1];
      System.out.println("\n You Selected Cancel Option");
      System.out.println("\n Changing State from S3 to S0");
      break;
   }
   case 5:  break;
   case 6:  break;
   case 7:  break;
}
}
public void SelectGas(int g)
{
   int cur = currentstateobj.getStateId();
   switch(cur)
   {
      case 0: break;
```

```
        case 1: break;
        case 2: break;
        case 3:  break;
        case 4:
        {
            System.out.println("\n Gas selected");
            System.out.println("\n s3 to s4");
            System.out.println("\n Start the pump");
            currentstateobj.SelectGas(g);
           currentstateobj = list[5];
           break;
        }
        case 5: break;
        case 6: break;
        case 7: break;
    }
}
```

```
            public void StartPump()
            {
    int cur = currentstateobj.getStateId();
    switch(cur)
    {
        case 0:  break;
```

```
        case 1:  break;
        case 2:  break;
        case 3:  break;
        case 4:  break;
        case 5:
        {
            currentstateobj.StartPump();
           currentstateobj = list[6];
           System.out.println("\n The Pump has been Started");
           System.out.println("\n Changing State from S4 to S5");
           break;
        }
        case 6:  break;
        case 7:  break;
    }
}


            public void Pump()
            {
    int cur = currentstateobj.getStateId();
    switch(cur)
    {
        case 0:  break;
        case 1:  break;
        case 2:  break;
        case 3:  break;
        case 4:  break;
```

```
        case 5:  break;
        case 6:
        {
            System.out.println("\n The gas is being Pumped");
            System.out.println("\n Changing State from S5 to S5");
           currentstateobj.Pump();
           currentstateobj = list[6];
           break;
        }
        case 7:  break;
    }
}


            public void StopPump()
            {
    int cur = currentstateobj.getStateId();
    switch(cur)
    {
        case 0: break;
        case 1: break;
        case 2: break;
        case 3: break;
        case 4:  break;
        case 5:  break;
        case 6:
        {
             currentstateobj.StopPump();
```

```
            currentstateobj = list[7];
            System.out.println("\n The Pump has now been Stopped");
            System.out.println("\n Changing State from S5 to S6");

            total1=    PumpGasUnit.dataobj.gettotal1();
             cash=   StoreCash.dataobj.getcash1();

            break;
        }
        case 7:  break;
            }
}
            public void Receipt()
            {
    int cur = currentstateobj.getStateId();
    switch(cur)
    {
        case 0:  break;
        case 1:  break;
        case 2: break;
        case 3:  break;
        case 4:  break;
        case 5:  break;
        case 6: break;
        case 7:
        {
```

```
            System.out.println("Receipt");
            currentstateobj.Receipt();
          currentstateobj = list[0];
          System.out.println("\n The Receipt is being Generated");
          System.out.println("\n Changing State from S6 to S0");
          break;
      }

   }
}

          public void NoReceipt()
          {

   int cur = currentstateobj.getStateId();
   switch(cur)
   {
      case 0: break;
      case 1: break;
      case 2: break;
      case 3: break;

      case 4: break;
      case 5: break;
      case 6: break;
      case 7:
          {
```

```
                int cr=cash-total1;

                        System.out.println("Cash returned="+cr);


         //cr=cash-total1;
          currentstateobj.NoReceipt();
        currentstateobj = list[0];
        System.out.println("\n You Selected No Receipt Option");
        System.out.println("\n Changing State from S6 to S0");
        break;
         }

         }
         }

}
```

Package output_processor

Class output.java

```
package output_processor;

import abstract_factory.AF;
import data_store.data;
import output_processor.output;
import strategy.StoreData;
import strategy.PayMsg;
import strategy.StoreCash;
```

```
import strategy.RejectMsg;
import strategy.SetW;
import strategy.SetPrice;
import strategy.ReadyMsg;
import strategy.SetInitialValues;
import strategy.PumpGasUnit;
import strategy.GasPumpedMsg;
import strategy.StopMsg;
import strategy.CancelMsg;
import strategy.PrintReceipt;
import strategy.DisplayMenu;
import strategy.NoReceipt;
public class output
{
                static AF afobj;
        data dataobj;
        public void setfactory(AF af1)
        {
           afobj=af1;
        }
        public void setData(data d1)
        {
           dataobj=d1;
        }
        public static void storeData()
        {
```

```
            StoreData object;
            object=afobj.getStoreData();
            object.storeData();
    }
    public static void displayMenu()
    {
        DisplayMenu object;
        object=afobj.getDisplayMenu();
        object.displayMenu();
    }
    public static void payMsg()
    {
        PayMsg object;
        object=afobj.getPayMsg();
        object.payMsg();
    }
    public static void rejectMsg()
    {
        RejectMsg object;
        object=afobj.getRejectMsg();
        object.rejectMsg();
    }
    public static void printReceipt()
    {
        PrintReceipt object;
        object=afobj.getPrintReceipt();
        object.printReceipt();
```

```
    }
    public static void cancelMsg()
    {
        CancelMsg object;
        object=afobj.getCancelMsg();
        object.cancelMsg();
    }

    public static void gasPumpedMsg()
    {
        GasPumpedMsg object;
        object=afobj.getGasPumpedMsg();
        object.gasPumpedMsg();
    }
    public static void pumpGasUnit()
    {
        PumpGasUnit object;
            object=afobj.getPumpGasUnit();
            object.pumpGasUnit();
    }
public static void storeCash()
    {
        StoreCash object;
        object=afobj.getStoreCash();
        object.storeCash();

    }
```

```
public static void readyMsg()
{
        ReadyMsg object;
        object=afobj.getReadyMsg();
        object.readyMsg();
}
public static void stopMsg()
{
        StopMsg object;
        object=afobj.getStopMsg();
        object.stopMsg();
}

public static void setPrice(int g)
{
        SetPrice object;
        object=afobj.getSetPrice();
        object.setPrice(g);
}
public static void setInitialValues()
{

        SetInitialValues object;
        object=afobj.getSetInitialValues();
        object.setInitialValues();
}
public static void setW(int t)
```

```
        {
            SetW object;
            object=afobj.getsetW();
            object.setW(t);
        }
        public static void NoReceipt()
        {

            NoReceipt object;
            object=afobj.getNoReceipt();
            object.noReceipt();
        }
            public static int getW() {

                    return 0;
            }

}
```

Package state

Class S0.java

```
package state;
import output_processor.output;
public class S0 extends State
{
        @Override
   public void Start()
   {
    output.payMsg();
   }
}
```

Class S1.java

```
package state;
import output_processor.output;
public class S1 extends State
{
        @Override
   public void PayCash()
   {
    output.storeCash();
    output.setW(0);
    output.displayMenu();
    }
}
```

Class S2.java

```
package state;
import output_processor.output;
public class S2 extends State
{
        @Override
   public void Approved()
   {
     output.setW(1);
     output.displayMenu();

   }
   public void Reject()
   {
    output.rejectMsg();

   }
}
```

Class S3.java

```
package state;
import output_processor.output;

public class S3 extends State
{
        @Override
```

```
        public void Cancel()
   {
    output.cancelMsg();

   }
        public void SelectGas(int g)
   {
    output.setPrice(g);

   }

}
```

Class S4.java

```
package state;
import output_processor.output;

public class S4 extends State
{
        @Override
   public void StartPump()
   {
    output.setInitialValues();
    output.readyMsg();

   }
}
```

Class S5.java

```
package state;
import output_processor.output;
public class S5 extends State
{
        @Override
   public void Pump()
   {
                output.pumpGasUnit();
                output.gasPumpedMsg();
                output.storeCash();

   }
   public void StopPump()
```

```
    {

        output.stopMsg();
    }
}
```

Class S6.java

```
package state;
import output_processor.output;
public class S6 extends State
{
        @Override
    public void Receipt()
    {
     output.printReceipt();

    }
        public void NoReceipt()
    {
     output.NoReceipt();

    }
}
```

Class Start.java

```
package state;
import output_processor.output;
public class Start extends State
{
        @Override
    public void Activate()
    {
        System.out.println("\n \n The Gas pump is  Activated ");
        output.storeData();
    }
}
```

Class State.java

```
package state;
import output_processor.output;
public class State
```

```
{
        output outputobj;//object of the output processor
        int Stateid;
   public int getStateId()
   {
      return Stateid;
   }
   public void setoutput(output o)
   {
      outputobj = o;
   }
   public void setStateId(int a)
   {
      Stateid = a;
   }
   public void Activate(){}
   public void Start(){}
   public void PayCash(){}
   public void PayCredit(){}
   public void Approved(){}
   public void Reject(){}
   public void SelectGas(int g){}
   public void Cancel(){}
   public void StartPump(){}
   public void Pump(){}
   public void StopPump(){}
   public void Receipt(){}
   public void NoReceipt(){}

}
```

Package Strategy

Class CancelMsg.java

```
package strategy;
import data_store.data;
public abstract class CancelMsg
{
   data dataobj;
   public abstract void cancelMsg();
   public void setdata(data dt)
   {
      dataobj=dt;
```

```
    }
}
```

Class CancelMsg1.java

```
package strategy;
public class CancelMsg1 extends CancelMsg
{
    public void cancelMsg()
    {
            System.out.println("\n The operation has been cancelled \n");
    }

}
```

Class DisplayMenu.java

```
package strategy;
import data_store.data;
public abstract class DisplayMenu
{
  data dataobj;
  public abstract void displayMenu();
  public void setdata(data dt)
    {
       dataobj=dt;
    }
}
```

Class DisplayMenu1.java

```
package strategy;
public class DisplayMenu1 extends DisplayMenu
{
    public void displayMenu()
    {
    System.out.println("\n **********DISPLAY MENU*************");
            System.out.println("\n Select option 6 for Regular ");
            System.out.println("\n Select option 7 for Super");

    }
}
```

Class DisplayMenu2.java

```
package strategy;

public class DisplayMenu2 extends DisplayMenu
{
   public void displayMenu()
   {
   System.out.println("\n ***********DISPLAY MENU*************");
            System.out.println("\n Select option 4 for Premium");
            System.out.println("\n Select option 5 to Regular");
            System.out.println("\n Select option 6 to Super");

   }

}
```

Class GasPumpedMsg.java

```
package strategy;
import data_store.data;
public abstract class GasPumpedMsg{
  data dataobj;
   public abstract void gasPumpedMsg();

   public void setdata(data dt)
   {
         dataobj=dt;
   }
}
```

Class GasPumpedMsg1.java

```
package strategy;
public class GasPumpedMsg1 extends GasPumpedMsg
{
   public void gasPumpedMsg()
   {
         float g = dataobj.getG();
         System.out.printf("The Gas pump has sucessfully pumped units :" +g);
   }

}
```

Class GasPumpedMsg2.java

```
package strategy;
public class GasPumpedMsg2 extends GasPumpedMsg
{
   public void gasPumpedMsg()
   {
         float l = dataobj.getL();
         System.out.printf("The Gas pump has sucessfully pumped L  units:"+l);

   }

}
```

Class NoReceipt.java

```
package strategy;
import data_store.data;
public abstract class NoReceipt
{
         data dataobj;
   public abstract void noReceipt();

   public void setdata(data dt)
   {
         dataobj=dt;
   }
}
```

Class NoReceipt1.java

```
package strategy;
public class NoReceipt1 extends NoReceipt
{
   public void noReceipt()
   {

   }
}
```

Class PayMsg.java

```
package strategy;
import data_store.data;
public abstract class PayMsg
{
   data dataobj;
   public abstract void payMsg();

   public void setdata(data dt)
   {
         dataobj=dt;
   }
}
```

Class PayMsg1.java

```
package strategy;
public class PayMsg1 extends PayMsg
{
   public void payMsg()
   {

         System.out.println("Please select a payment mode from the list of available options
below");

   }

}
```

Class PrintReceipt.java

```
package strategy;
import data_store.data;
public abstract class PrintReceipt {
   public data dataobj;
   public abstract void printReceipt();

   public void setdata(data dt)
   {
      dataobj=dt;
   }
}
```

Class PrintReceipt1.java

```
package strategy;
public class PrintReceipt1 extends PrintReceipt
{
        @Override
   public void printReceipt()
   {
        float total = dataobj.gettotal();
        System.out.printf("The toatl amount that the gas that has been pumped is "+total);

   }

}
```

Class PrintReceipt2.java

```
package strategy;

public class PrintReceipt2 extends PrintReceipt
{
        int total1=0,cash=0,cr=0;
        @Override
   public void printReceipt()
   {

        System.out.println("Receipt");
        total1=dataobj.gettotal1();
        cash=dataobj.getcash1();
        cr=cash-total1;
        System.out.printf(" \n The total amount the gas has been pumped is \t "+total1);
        int L=dataobj.getL();
        System.out.printf(" \n The total Litres of Gas that has been pumped is \t "+L);
        System.out.println("\nCash Returned= "+cr);
   }

}
```

Class PumGasUnit.java

```
package strategy;
import data_store.data;
public abstract class PumpGasUnit
{
  public static data dataobj;
   public abstract void pumpGasUnit();
   public void setdata(data dt)
   {
      dataobj=dt;
   }
}
```

Class PumpGasUnit1.java

```
package strategy;

public class PumpGasUnit1 extends PumpGasUnit
{
        @Override
   public void pumpGasUnit()
   {
```

```
      float g=dataobj.getG();
      g=g+1;
      float total;
      float price = dataobj.getprice();
      total =price *g;
      dataobj.setG(g);
      dataobj.settotal(total);
     }
}
```

Class PumpGasUnit2.java

```
package strategy;
public class PumpGasUnit2 extends PumpGasUnit
{
        @Override
   public void pumpGasUnit()
   {
     int l=dataobj.getL();
     l=l+1;
     int total1;
     int price1 = dataobj.getprice1();
     total1 =price1*l;
     dataobj.setL(l);
     dataobj.settotal1(total1);
    }

}
```

Class ReadyMsg.java

```
package strategy;
import data_store.data;
public abstract class ReadyMsg
{
   data dataobj;
   public abstract void readyMsg();
   public void setdata(data dt)
   {
      dataobj=dt;
   }
}
```

Class ReadyMsg1.java

```
package strategy;

public class ReadyMsg1 extends ReadyMsg
{
   public void readyMsg()
   {

         System.out.println(" \n The Gas Pump is ready to pump; Continue to Pump..");
   }

}
```

Class RejectMsg.java

```
package strategy;
import data_store.data;
public abstract class RejectMsg
{
   data dataobj;
   public abstract void rejectMsg();
   public void setdata(data dt)
   {
      dataobj=dt;
   }
}
```

Class RejectMsg1.java

```
package strategy;

public class RejectMsg1 extends RejectMsg
{
   public void rejectMsg()
   {
          System.out.println("Sorry the action is Rejected.Choose an option.....");
   }

}
```

Class SetInitialValues.java

```
package strategy;
```

```
import data_store.data;
public abstract class SetInitialValues
{
        data dataobj;
   public abstract void setInitialValues();
   public void setdata(data dt)
   {
      dataobj=dt;
   }
}
```

Class SetInitialVlaues1.java

```
package strategy;
public class SetInitialValues1 extends SetInitialValues
{
        public void setInitialValues()
   {

      dataobj.setG(0);
      dataobj.settotal(0);
   }

}
```

Class SetInitialValues2.java

```
package strategy;
public class SetInitialValues2 extends SetInitialValues
{

        public void setInitialValues()
   {

      dataobj.setL(0);
      dataobj.settotal(0);
   }

}
```

Class SetPrice.java

```
package strategy;
import data_store.data;
```

```
public abstract class SetPrice
{
  data dataobj;
   public abstract void setPrice(int g);
   public void setdata(data dt)
   {
     dataobj=dt;
   }
}
```

Class SetPrice1.java

```
package strategy;
public class SetPrice1 extends SetPrice
{
          @Override
   public void setPrice(int g)
   {
     float a=dataobj.getregularprice();
     float b=dataobj.getsuperprice();
     if( g== 1)
     dataobj.setprice(a);
     else if (g == 2)
        dataobj.setprice(b);
   }

}
```

Class SetPrice2.java

```
package strategy;
public class SetPrice2 extends SetPrice
{
          @Override
   public void setPrice(int g)
   {
                    int a=dataobj.getregularprice1();
     int b=dataobj.getpremiumprice();
     int c=dataobj.getsuperprice1();
     if( g== 1)
     dataobj.setprice1(a);
     else if (g == 2)
     dataobj.setprice1(b);
     else
```

```
            dataobj.setprice1(c);
    }

}
```

Class SetW.java

```
package strategy;
import data_store.data;
public abstract class SetW
{
    data dataobj;
    public abstract void setW(int t);
    public void setdata(data dt)
    {
        dataobj=dt;
    }
}
```

Class SetW1.java

```
package strategy;
public class SetW1 extends SetW
{
            public void setW(int t)
    {
      dataobj.setintW(t);
    }


}
```

Class StopMsg.java

```
package strategy;
import data_store.data;
public abstract class StopMsg
{
    data dataobj;
    public abstract void stopMsg();
    public void setdata(data dt)
    {
        dataobj=dt;
    }
}
```

Class StopMsg1.java

```
package strategy;

public class StopMsg1 extends StopMsg
{
   public void stopMsg()
   {

         System.out.println("The Gas Pump has stopped Pumping");
         System.out.println("\n Choose to print the Receipt ");

   }

}
```

Class StoreCash.java

```
package strategy;
import data_store.data;
public abstract class StoreCash
{
   public static data dataobj;
   public abstract void storeCash();
   public void setdata(data dt)
   {
      dataobj=dt;
   }
}
```

Class StoreCash1.java

```
package strategy;
public class StoreCash1 extends StoreCash
{
   public void storeCash()
   {
      int c =dataobj.getintc();
      float d =(float)c;
      dataobj.setcash(d);
   }

}
```

Class StoreCash2.java

```
package strategy;

public class StoreCash2 extends StoreCash
{
   public void storeCash()
   {

      int c =dataobj.getfloatc1();

      dataobj.setcash1(c);

   }

}
```

Class StoreData.java

```
package strategy;
import data_store.data;
public abstract class StoreData
{
   data dataobj;
   public abstract void storeData();
   public void setdata(data dt)
   {
      dataobj=dt;
   }
}
```

Class StoreData1.java

```
package strategy;
public class StoreData1 extends StoreData
{

         @Override
   public void storeData()
   {
    float a, b;
    a=dataobj.getfloata();
    dataobj.setregularprice(a);
    b=dataobj.getfloatb();
    dataobj.setsuperprice(b);
```

```
    }

}
```

Class StoreData2.java

```
package strategy;
public class StoreData2 extends StoreData
{
        @Override
   public void storeData()
   {
    int a, b,c;
    a=dataobj.getfloata1();
    dataobj.setregularprice1(a);
    b=dataobj.getfloatb1();
    dataobj.setpremiumprice(b);
    c=dataobj.getfloatc1();
    dataobj.setsuperprice1(c);
   }

}
```