# CS586- Software System Architecture Gas-Pump System Using Model Driven Architecture


Jaimin Sanghvi (A20344798)

Guided By: Prof. Bogdan Korel

DEPARTMENT OF COMPUTER SCIENCE, ILLINOIS INSTITUTE OF TECHNOLOGY

# Contents

# 1. ReadMe File

Name : Jaimin V Sanghvi

CWID: A20344798

Email Id: jsanghv2@hawk.iit.edu

Contact Number:+1(312)-647-5043

Professor: Bogdan Korel

Term: Spring 2015

## Steps to run a project

System Requirements: Java 1.6+

You can run the program using the following command on the command line. You must have a JRE or JDK installed on the machine.

**java –jar GasPumpSystem.jar**

Once program is running, you can select one of the three GasPumps. After selecting a GasPump, you must run the following operations in order to start operating the GasPump.

Activate() →if parameters that we are passing are greater than 0 than it activates the selected GasPump

Once this operation is done, than all other operations are available for selected gaspump..

# 2. Introduction

## Goal of the Project

Main goal of this project is to design three different GasPump components using a Model-Driven Architecture (MDA) and then implement these GasPump system using three different design patterns.

## Description of the Project

The system contains three different Gas-Pump named as Gas-Pump1, Gas-Pump2 and GasPump-3. Each of these components has their own set of operation. MDA architecture a simple software design approach. It provides set of instructions for the structuring of specifications, which are expressed as model.

To create MDA-EFSM architecture, we need to capture "meta-behavior" of all Gasp-Pumps and this can be achieved by following items.

- Identify Events
- Identify Actions
- State Diagram

In addition, I have used three different design patterns named as Abstract Factory Method, State Pattern and Strategy Pattern. Use of these pattern improved code reusability and code readability. By using MDA-EFSM architecture and object oriented design patterns, we can implement system with loose coupling and highly cohesive.

This model represents three different Gas-Pump components. All three Gas-Pump components are state-based components and are used to control a simple gas pumps. List of components with their operations are listed as below:

The **GasPump-1** component supports the following operations:
Activate (int a) // the gas pump is activated where a is the price of the 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
PayCash(int c) // pay for gas by cash, where c represents prepaid cash
StartPump() // start pumping gas
PumpGallon() // one gallon of gas is disposed
StopPump() // stop pumping gas

The **GasPump-2** 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

The **GasPump-3** component supports the following operations:
Activate (float a, float b) // the gas pump is activated where a is the price of regular gas
// and b is the price of premium gas per liter
Start() //start the transaction
PayCash(float 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
StartPump() // start pumping gas
PumpLiter() // one liter of gas is disposed
StopPump() // stop pumping gas
Receipt() // Receipt is requested
NoReceipt() // No receipt

# 3. Model-Driven Architecture of the component

## MDA –EFSM Model (meta-model) for the Gas-Pump Components

### 3.1 A list of events for the MDA-EFSM

- Activate()
- Start()
- PayCredit()
- PayCash()
- Reject()
- Cancel()
- Approved()
- StartPump()
- Pump()
- StopPump()
- SelectGas(int g)
- Receipt()
- NoReceipt(State s)

### 3.2 A list of actions for the MDA-EFSM

- 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
- SetW(int k) // set value for credit/cash flag
- 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) 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

## 3.3 A diagram of the MDA-EFSM

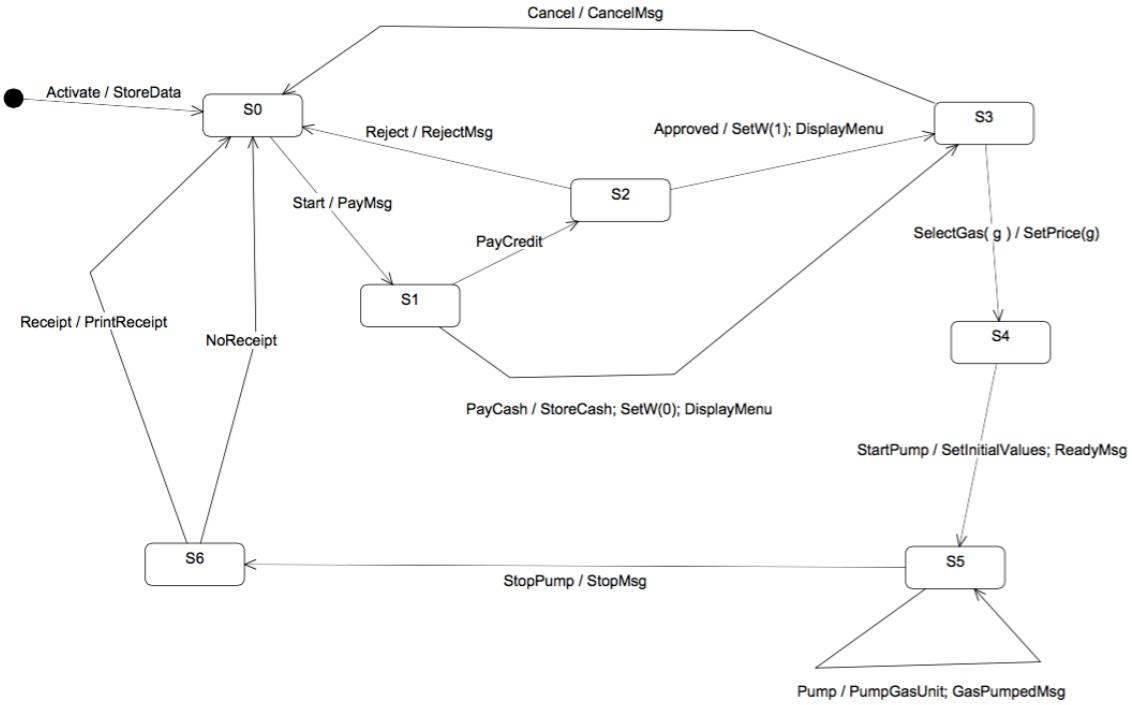



Figure 3-1 State Transition Diagram

## 3.4 Pseudo Code of MDA-EFSM System

**GasPump-1**

```
int cash, price, G, w ;
MDA-EFSM *m;
DataStore *d;
Activate(int a) {
        if (a>0) {
                d->temp_a=a;
        m->Activate()
        }
}

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

PayCredit() {
         m->PayCredit();
}
Reject() {
```

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

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

PayCash(int c) {
        if (c>0) {
                d->temp_c=c;
        m->PayCash();
}

StartPump() {
        m->SelectGas(1);
        m->StartPump();
}

PumpGallon() {
        if (d->w==1)
                m->Pump();
        else if (d->w==0) {
                if (d->cash<(d->G+1)*d->price) {
                        m->StopPump();
                        m->Receipt();
                }
                else
                        m->Pump();
                }
        }

StopPump() {
        m->StopPump();
        m->Receipt();
}
```

**GasPump-2**

```
MDA-EFSM *m;
DataStore *d;

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->PayCredit();
}

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();
}
```

**GasPump-3**

```
int cash, L, price;
MDA-EFSM *m;
DataStore *d;

Activate(float a, float b) {
        if ((a>0)&&(b>0)) {
```

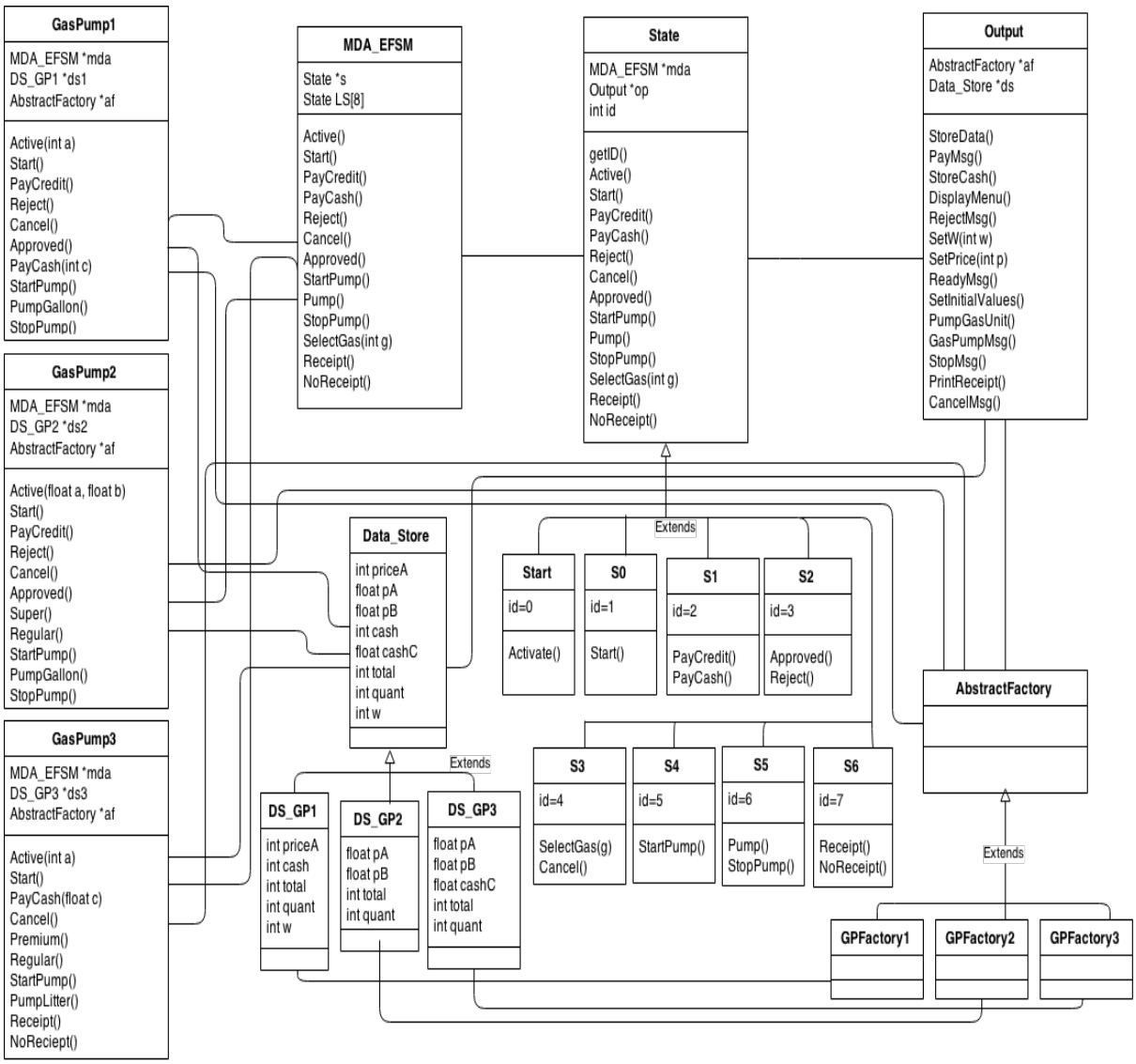
```
                d->temp_a=a;
                d->temp_b=b;
                m->Activate();
        }
}

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

PayCash(float c) {
         if (c>0) {
                d->temp_c=c;
                m->PayCash()
        }
}

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

Premium() {
        m->SelectGas(2);
}

Regular() {
        m->SelectGas(1);
}

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

PumpLiter() {
        if (d->cash<(d->L+1)*d->price)
                m->StopPump();
        else
                m->Pump()
}

StopPump() {
        m->StopPump();
}
Receipt() {
        m->Receipt();
}

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

# 4 Class diagram(s) of the MDA of the GasPump components. For each class in the class diagram(s) you should:

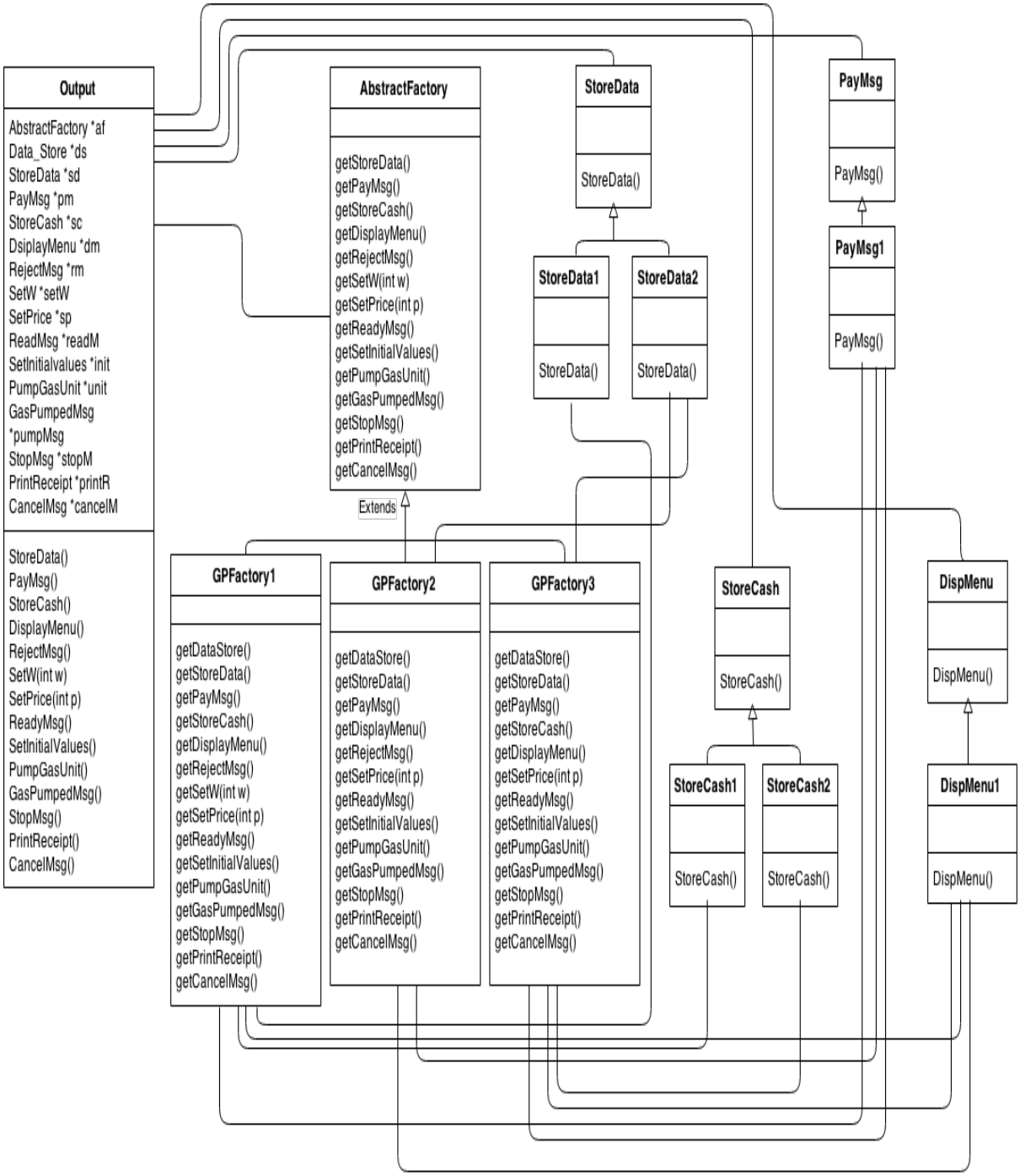
As system contains too many classes, I have divided my class diagram in three parts.

In the first part, I have designed all input classes with state and abstract factory design pattern. In second and third part, I have shown all action classes with strategy design pattern.

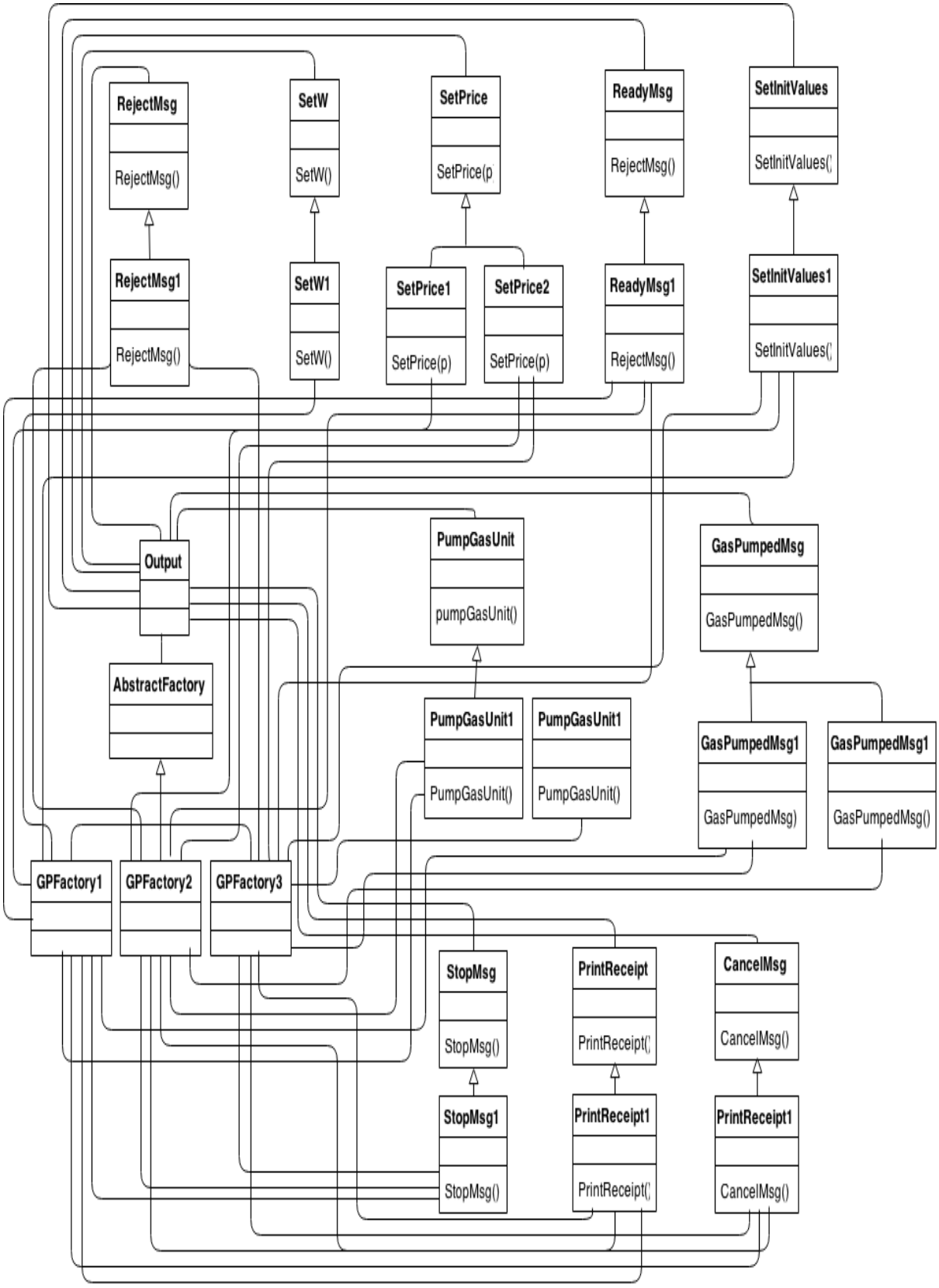
As explained in previous point, the first part of class diagram is as below:




**Class Diagram : Part1 – It includes all input classes with state and abstract factory pattern. Setter and Getter methods of Data_Store class are shown in pseudo code.**




**Class Diagram : Part2 –** It includes output class and action with strategy pattern. In this part, I have designed complete output and factory classes with pointers and methods.




**Class Diagram : Part3 –** It includes output class and action classes with strategy pattern. In this part, I have put only blocks of output and abstract factory classes.

## 4.1 Describe the purpose of the class

| 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. |
| **GasPump3** | GasPump3 class handles input events and executes appropriate actions from the input and call method of MDA_EFSM. |
| **Data_Store** | It stores centralized dependent data of the system. |
| **DS_GP1** | It stores independent data for GasPump-1 component. |
| **DS_GP2** | It stores independent data for GasPump-2 component. |
| **DS_GP3** | It stores independent data for GasPump-3 component. |
| **MDA_EFSM** | 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 reciect payment operations. After performing these operation it will changed state from S2 to S3. |
| **S3** | During this state, user can take decision to continue an operation by swithching 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 liiter 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 recipet or not. If yes then system generate reciept 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. |
| **AbstractFactory** | It provides defination of each concrete factory. |

| | |
|---|---|
| **GPFactory1** | It provides families of independent objets for GasPump1 component actions. |
| **GPFactory2** | It provides families of independent objets for GasPump2 component actions. |
| **GPFactory3** | It provides families of independent objets for GasPump3 component actions. |
| **StoreData** | It stores price(s) for the gas from the temporary data store |
| **PayMsg** | It displays a type of payment method |
| **DispMenu** | 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 |
| **SetInitValues** | It sets value of G (or L) and totla 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 |

## 4.2 Specify responsibilities of each operations supported by each class

| **Components** | **Responsibilities** |
|---|---|
| **Activate()** | It activates a pump by taking price of the gas per gallon. For GasPump1, It takes one integer value for regular gas. For GasPump2, It takes two float values for regular and super/premium gas. |
| **Start()** | It starts pump and prints available payment options. For GasPump1, It prints payment options like payByCash and payByCredit. For GasPump2, It prints payCredit option and For GasPump3, 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 valids. |
| **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 initiallizes 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 receipt. |

## 4.3 Describe the purpose of main attributes of each class

| **Classes** | **Attributes** |
|---|---|
| **GasPump** | GasPump1 gp1;<br>GasPump2 gp2;<br>GasPump3 gp3;<br>MDA_EFSM mda; |
| **Data_Store** | int priceA;<br>float priceA;<br>float priceB;<br>int cash;<br>float cash; |
| **MDA_EFSM** | State s; |
| **Output** | AbstractFactory af; |

| | |
|---|---|
| **AbstractFactory** | GPFacotry1 gf1;<br>GPFacotry2 gf2;<br>GPFactory3 gf3; |

# 5 Dynamics. Provide sequence diagrams of two Scenarios:

## 5.1 Sequence Diagram for Scenario1

Scenario-I should show how one gallon of gas (paid by credit) is disposed in GasPump-1, i.e., the following sequence of operations is issued: Activate(4), Start(), PayCredit(), Approved(), StartPump(), PumpGallon(), StopPump()

Activate(4)

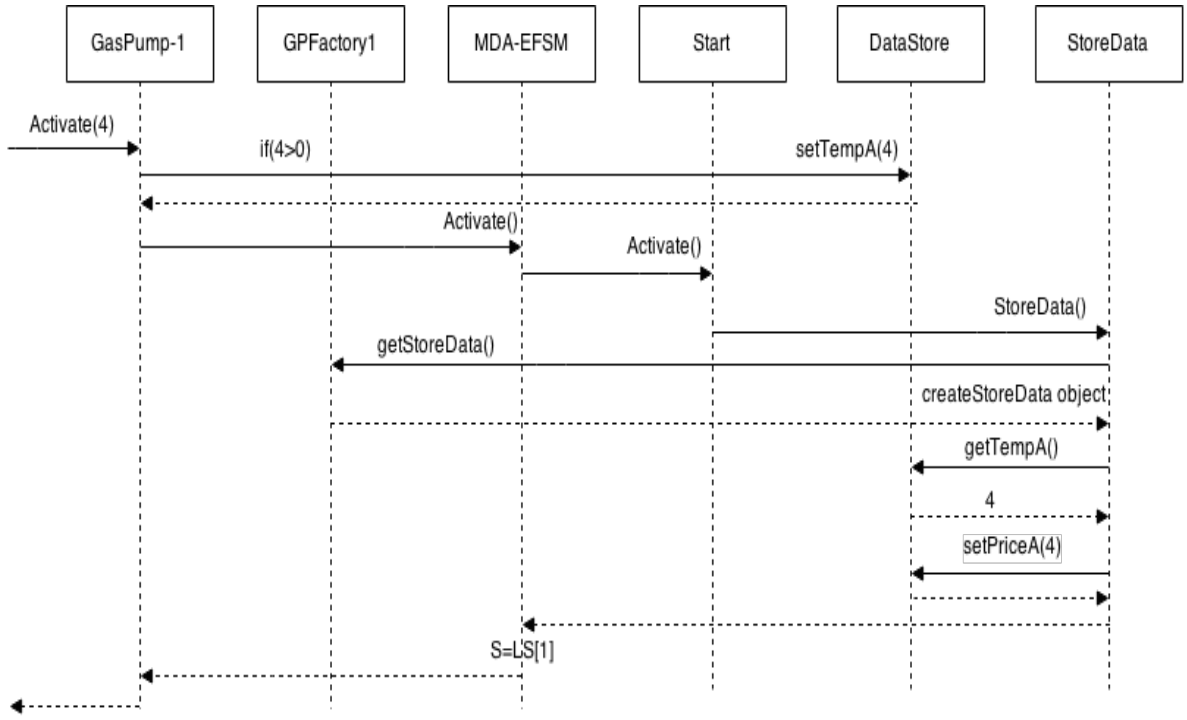



Figure 5-1 Sequence Diagram (Activate(4))

Start()

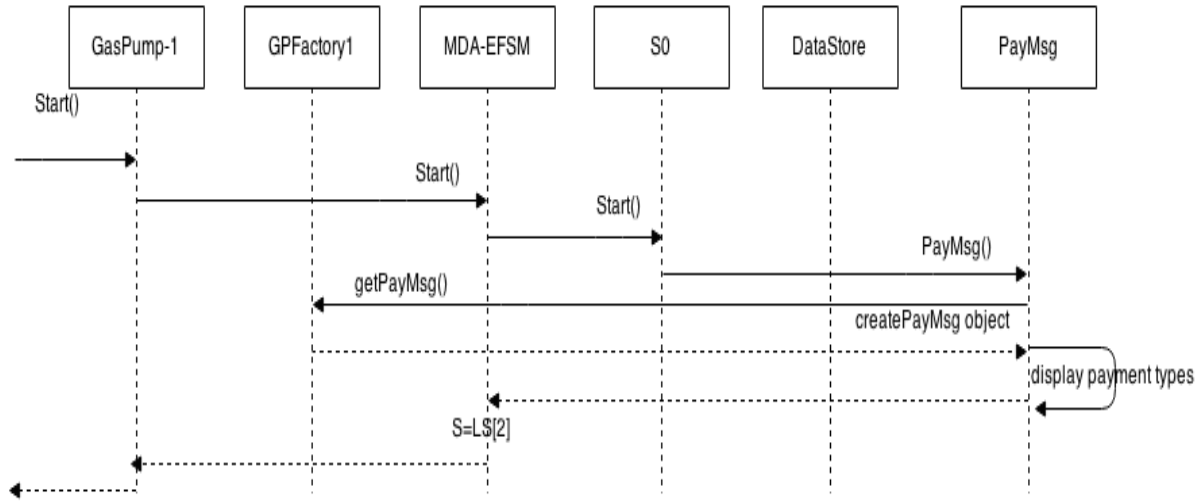



Figure 5-2 Sequence Diagram (Start())

PayCredit()

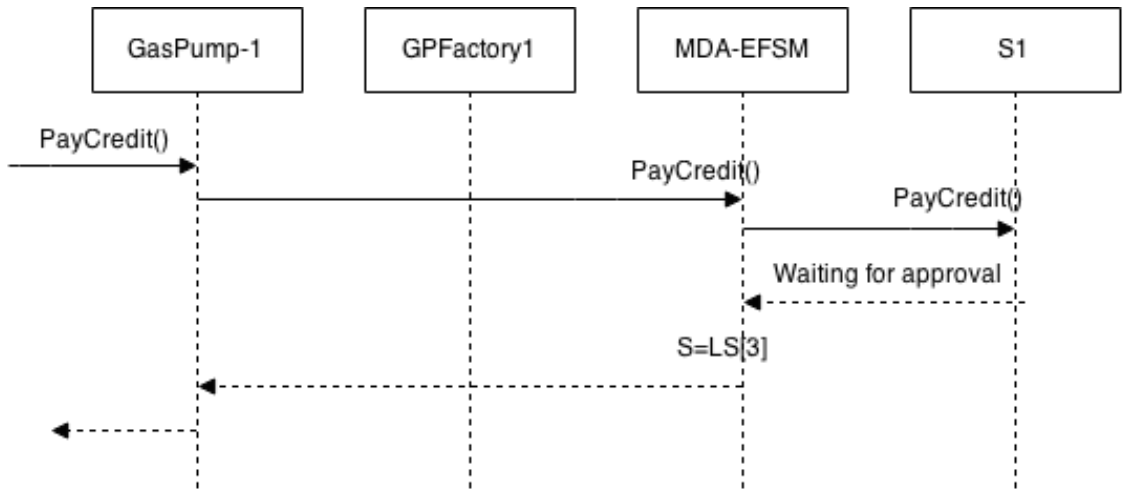



Figure 5-3 Sequence Diagram (PayCredit())

Approved()

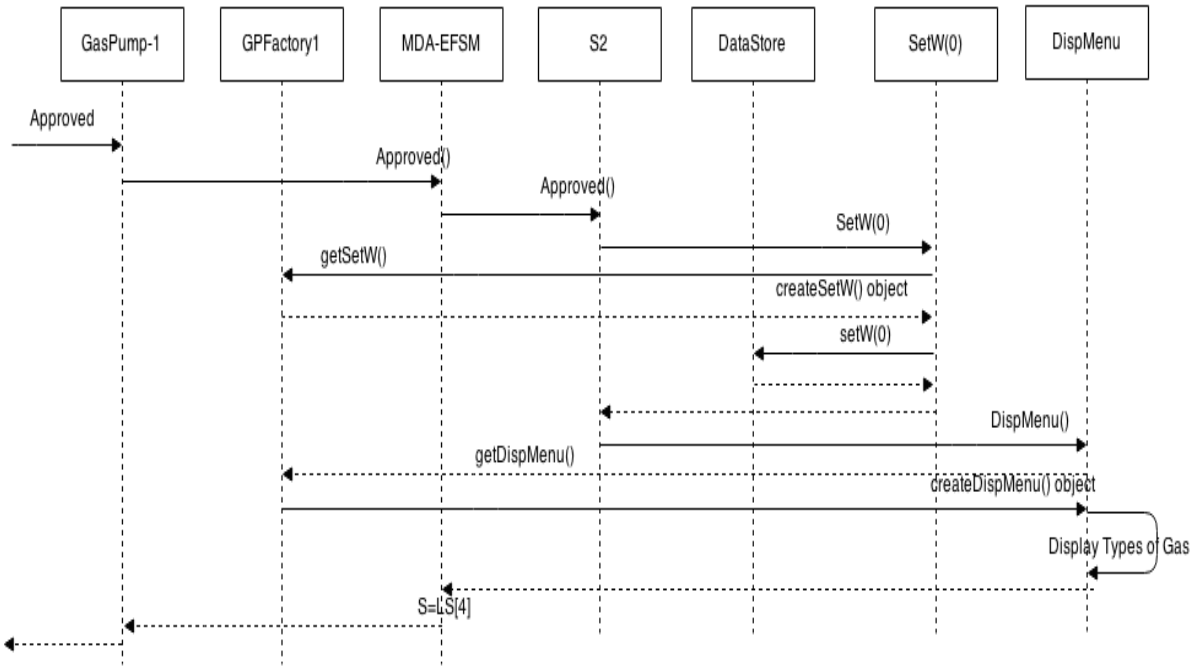



Figure 5-4 Sequence Diagram (Approved())

## StartPump()

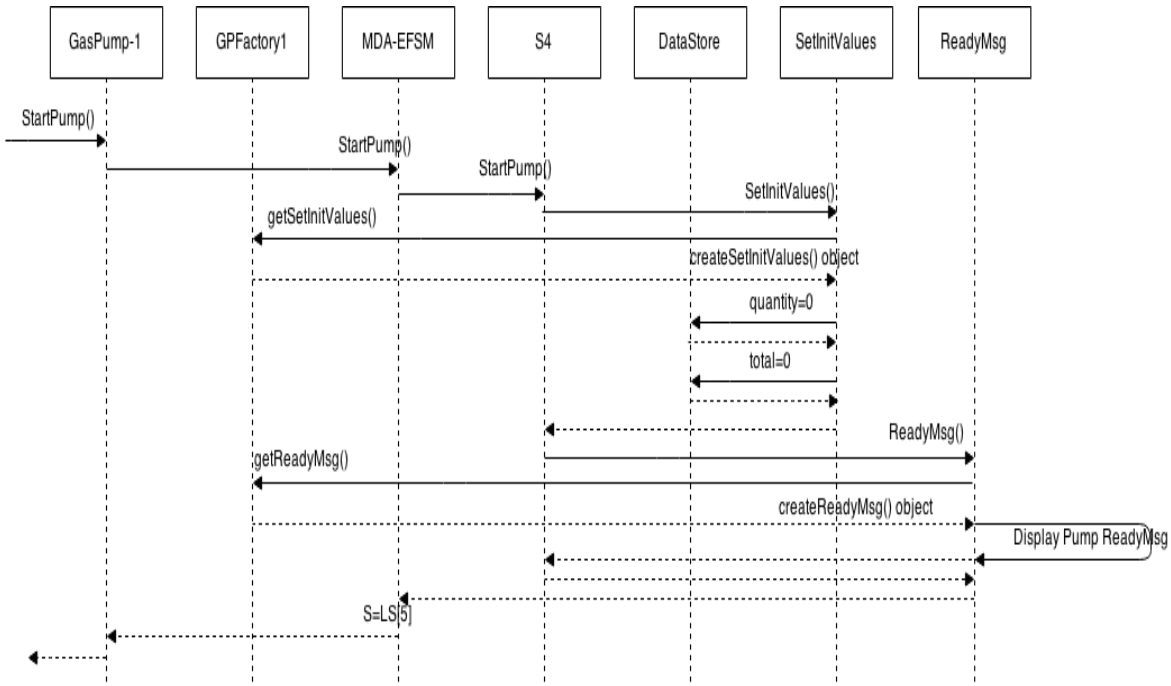



Figure 5-5 Sequence Diagram (StarPump())

## PumpGallon()

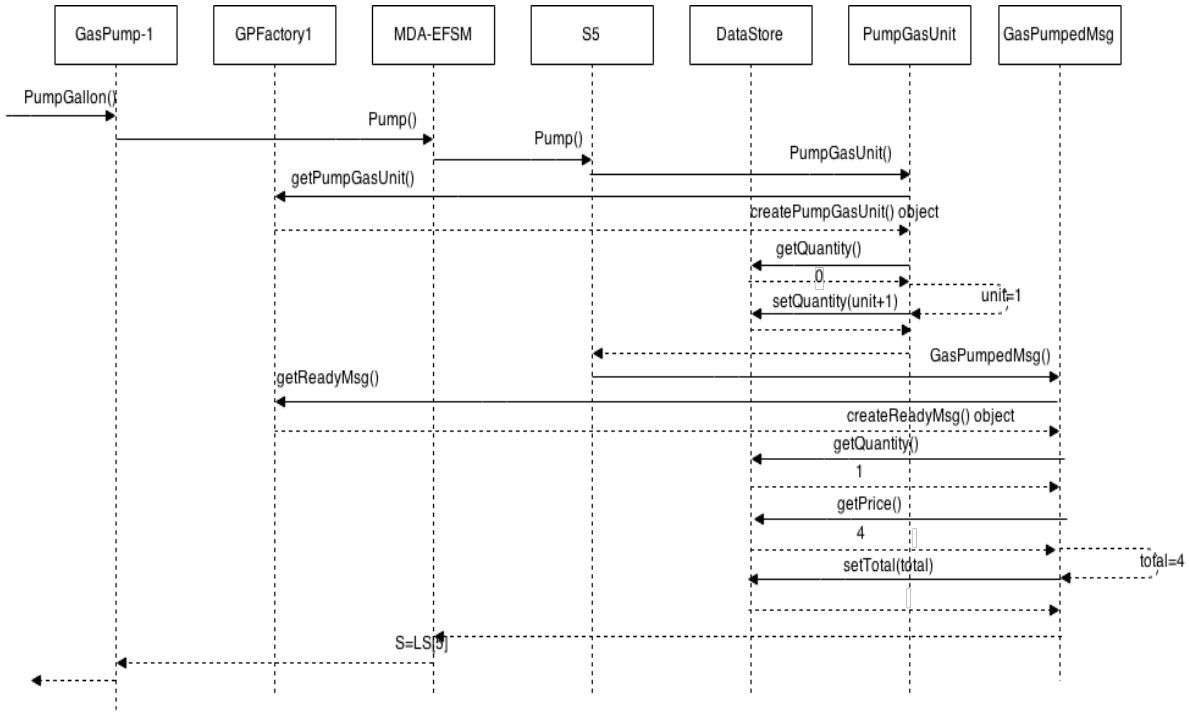



Figure 5-6 Sequence Diagram (PumpGallon())

StopPump()

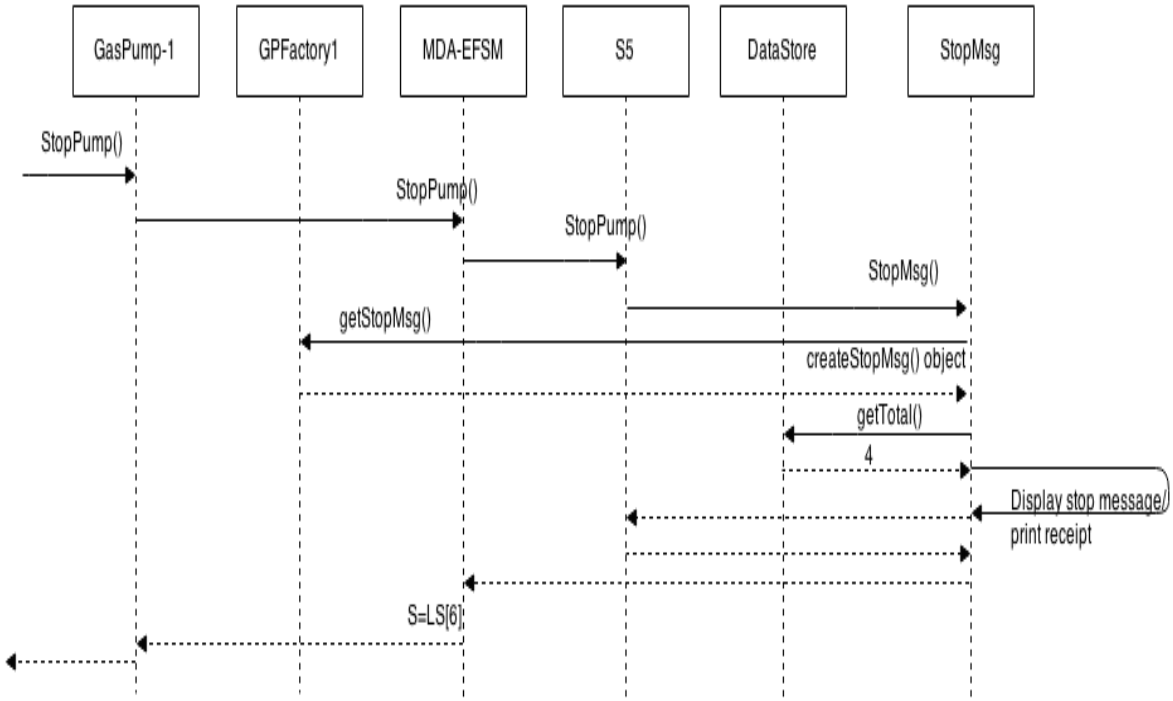



Figure 5-7 Sequence Diagram (StopPump())

## 5.2 Sequence Diagram for Scenario2

Scenario-II should show how one liter of gas (paid by cash) is disposed in GasPump-3, i.e., the following sequence of operations is issued: Activate(3.1, 4.5), Start(), PayCash(6.2), Premium(), StartPump(), PumpLiter(), PumpLiter(), NoReceipt()

Activate(3.1,4.5)

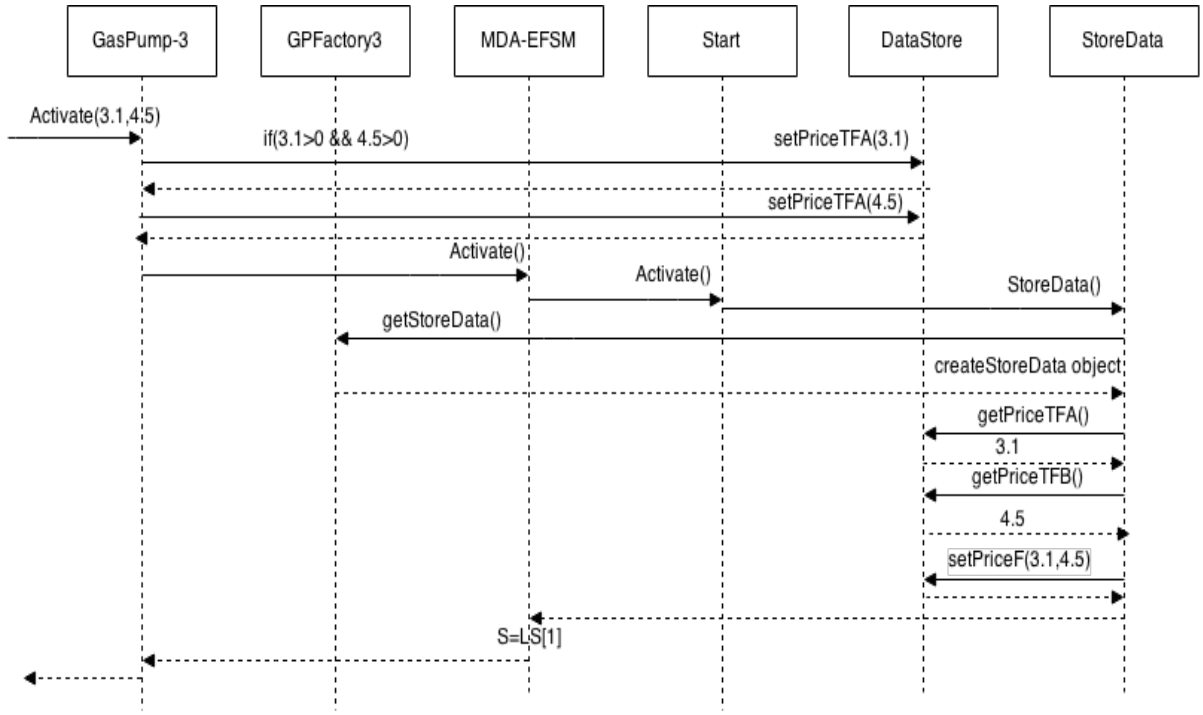



Figure 5-8 Sequence Diagram (Activate(3.1,4.5))

Start()

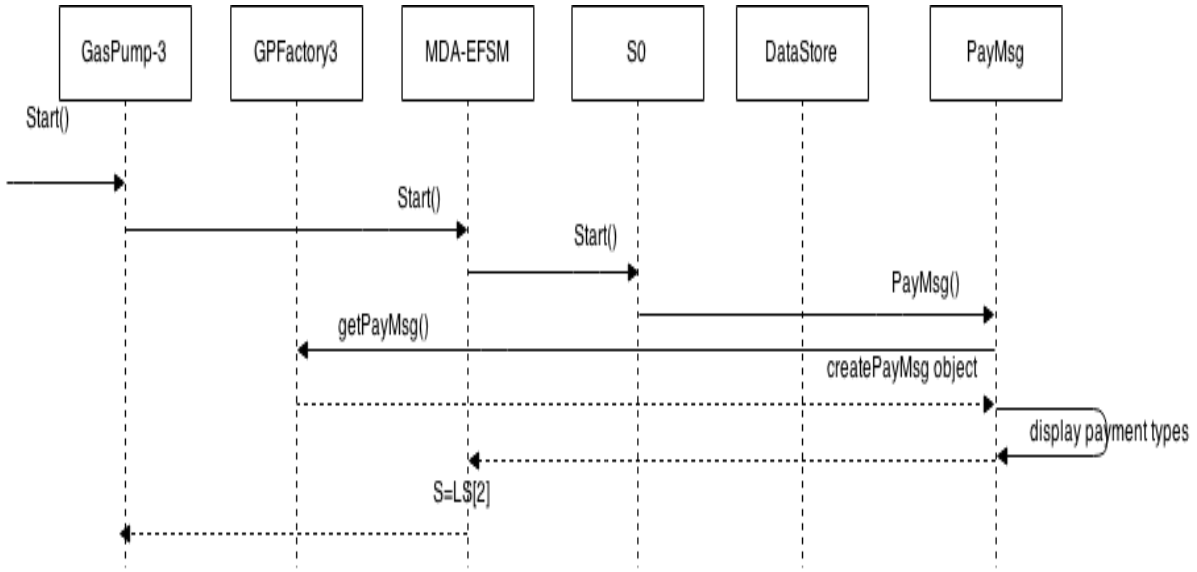



Figure 5-9 Sequence Diagram (Start())

## PayCash(6.2)

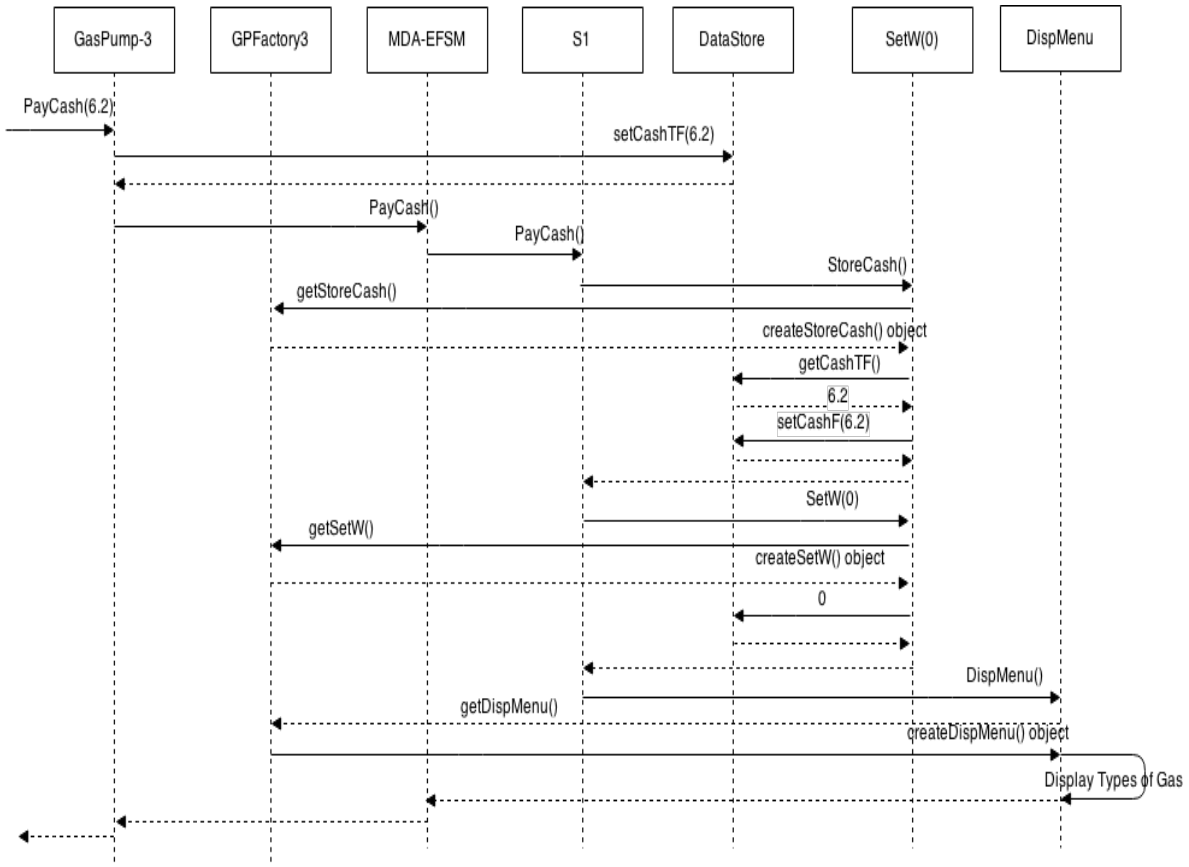



Figure 5-10 Sequence Diagram (PayCash(6.2))

## Premiun()

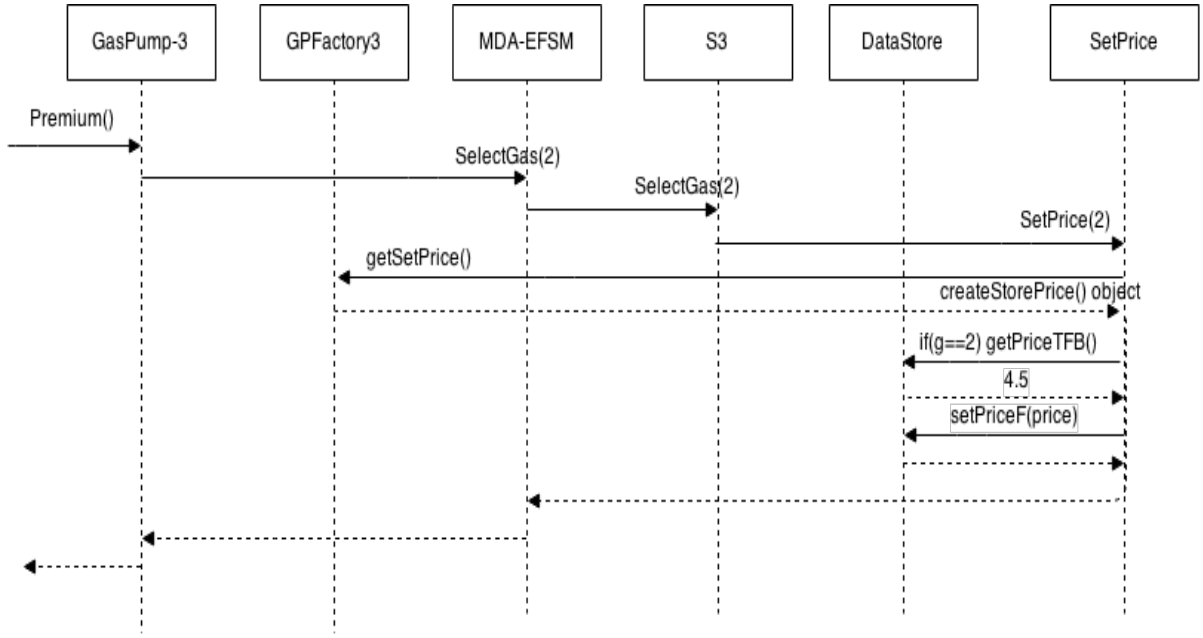



Figure 5-11 Sequence Diagram (Premium())

StartPump()

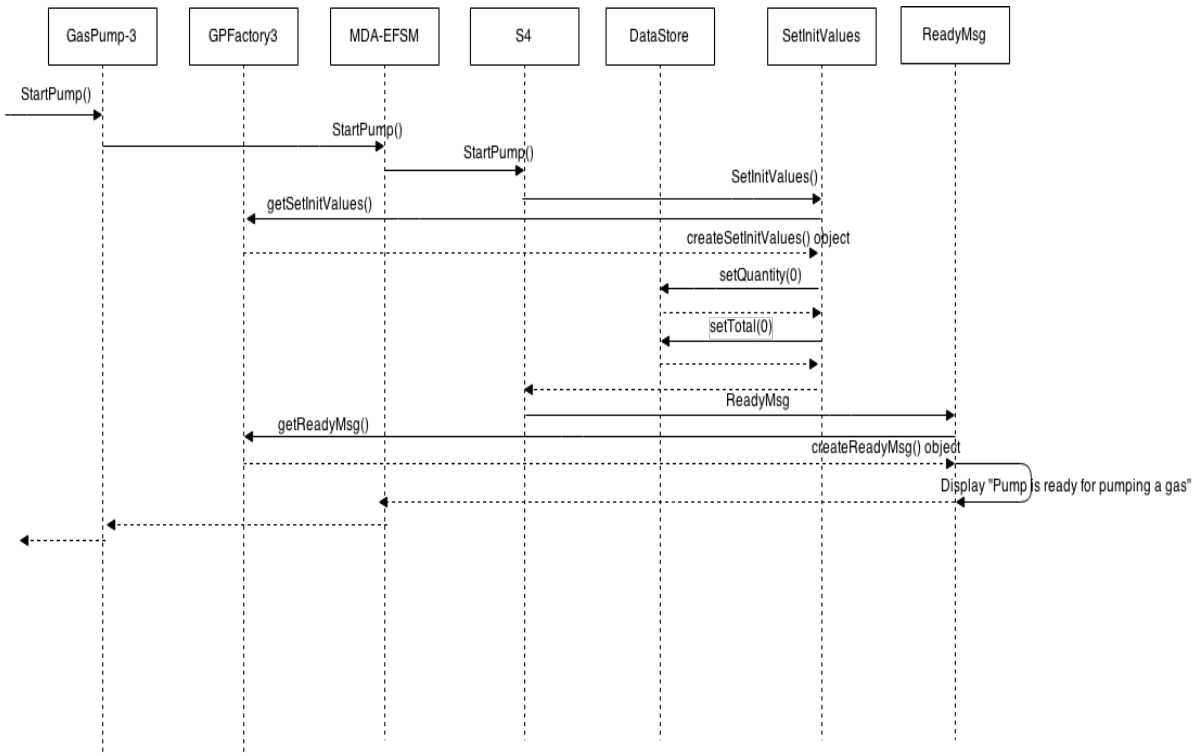



Figure 5-12 Sequence Diagram (StartPump())

NoReceipt()

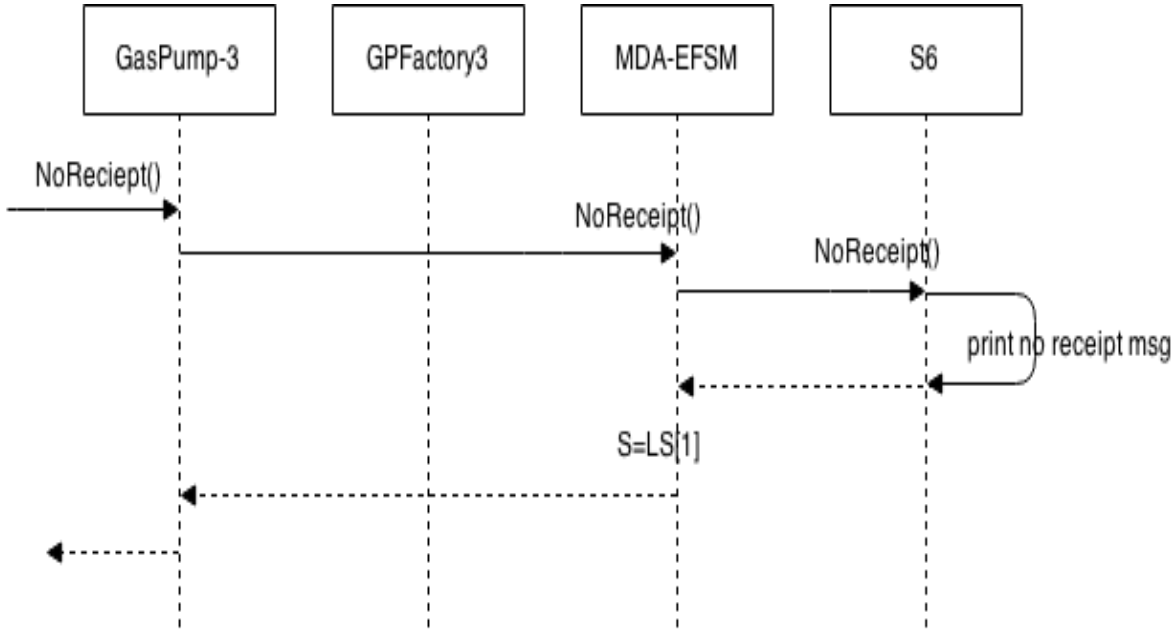



Figure 5-13 Sequence Diagram (NoReceipt())

PumpLitter()

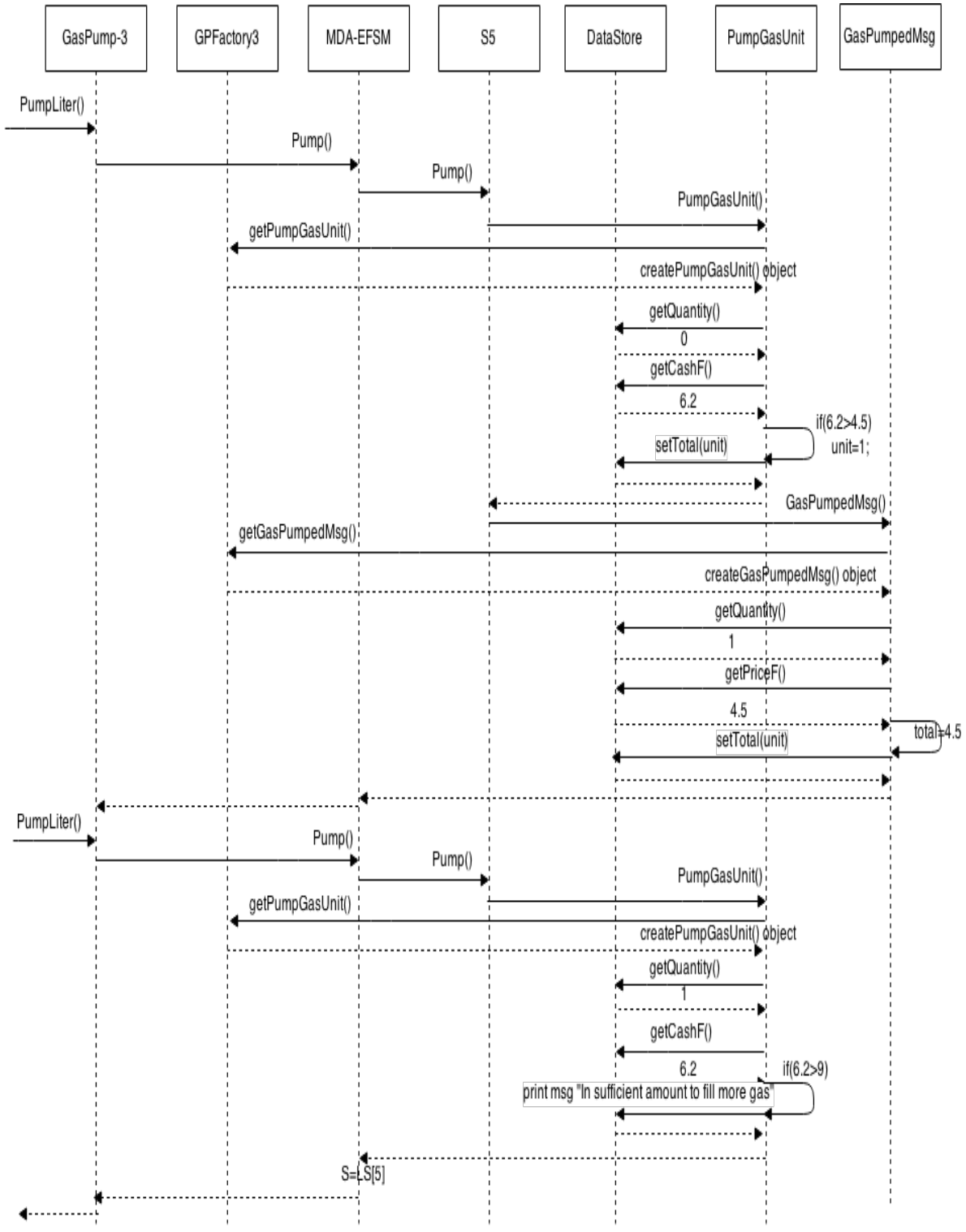



**Figure 5-14 Sequence Diagram (NoReceipt())**

# 6 Source Code

### Package gaspumpsystem.main // main class

**Main.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.main;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.logging.Level;
import java.util.logging.Logger;
import gaspumpsystem.ip.*;

/**
 *
 * @author jaiminsanghvi
 */
public class Main {
//GasPump Objects created

   GasPump1 objGp1;
   GasPump2 objGP2;
   GasPump3 objGP3;

   int selectedPump = 0;

   private Main() {
      try {
         // Open input stream to read data from console
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         //Display program header
         System.out.println("CS-586 Final Project\nby Jaimin Sanghvi\nCWID:
A20344798\nTerm : Spring 2015");
         System.out.println("\n");
         do {
            System.out.println("*** Select GasPump ***\n----------------------");
            System.out.println("1. GasPump-1\n2. GasPump-2\n3. GasPump-3\n");

            //Choose a GasPump
            System.out.println("Choose GasPump:");

            //Store user's selection
            selectedPump = Integer.parseInt(br.readLine());
            switch (selectedPump) {
               case 1:
                  //  if GasPump-1 is selected, go to GasPump-1
                  System.out.println("GasPump-1 is selected");
                  objGp1 = new GasPump1();       //Create GasPump-1 object
                  objGp1.run();              //Start working in GasPump-1
```

```
                break;
            case 2:
                //  if GasPump-2 is selected, go to GasPump-2
                System.out.println("GasPump-2 is selected");
                objGP2 = new GasPump2();         //Create GasPump-2 object
                objGP2.run();               //Start working in GasPump-2
                break;
            case 3:
                //if GasPump-3 is selected, go to GasPump-3
                System.out.println("\tGasPump-3 is selected");
                objGP3 = new GasPump3();        //Create GasPump-3 object
                objGP3.run();                //Start working in GasPump-3
                break;
default:
                System.out.println("Please choose correct GasPump");
          }
        } while (selectedPump < 1 || selectedPump > 4); // check wether user have select
gaspump from 1,2,3
      } catch (IOException ex) {
         Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
         System.out.println("\n IO exception in Main Class : " + ex);
      }
   }

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      // TODO code application logic here
      Main objGas = new Main(); //
   }
}
```

## Package gaspumpsystem.ip // input classes

**GasPump1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.ip;

import gaspumpsystem.datastore.Data_Store;
import gaspumpsystem.factory.*;
import gaspumpsystem.mdaefsm.MDA_EFSM;
import gaspumpsystem.op.Output;
import gaspumpsystem.state.State;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author jaiminsanghvi
 */
public class GasPump1 {

   public static AbstractFactory af = new GPFactory1();    // create GasFactory-1 object
   int userChoice = 0;
   int paymentOption = 0;
   int price = 0, cash = 0;

   public void run() {

      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

      MDA_EFSM mda = new MDA_EFSM();                  // create MDA_EFSM object
      Data_Store ds = af.getDataStore();              // create Data_Store object
      State.setOutput(new Output(af, ds));

      //  Print GasPump-1 menu
      System.out.println("GasPump-1 : Menu Of Operations");
      System.out.println("1. Active(int a)");
      System.out.println("2. Start()");
      System.out.println("3. PayCredit()");
      System.out.println("4. Reject()");
      System.out.println("5. Cancel()");
      System.out.println("6. Approved()");
      System.out.println("7. PayCash(int c)");
      System.out.println("8. StartPump()");
      System.out.println("9. PumpGallon()");
      System.out.println("10. StopPump()");
      System.out.println("0. Quit from the system");

      System.out.println("\n\n");
```

```
    System.out.println("Please select an operation");

    do {
      try {
        //  display list of operations
        System.out.println();
        System.out.println("GasPump-1 Execution");
        System.out.println("Select an Operation:");
        System.out.println("1-Activate,2-Start,3-PayCredit,4-Reject,5-Cancel,6-Approved,7-
PayCash,8-StartPump,9-PumpGallon,10-StopPump, 0-Quit");

        //Get user's choice
        String selectedChoice = br.readLine();
        userChoice = Integer.parseInt(selectedChoice);

        switch (userChoice) {
          case 1:
            //  Activate option selected
            System.out.println("Operation: Active(int a)");
            System.out.println("Enter value of parameter a : ");
            price = Integer.parseInt(br.readLine());    // read price input from user
            if (price > 0) {
              ds.setTempA(price);        // store temp value to data store
              mda.active();
            } else {
              System.out.println("Invalid value. Values should be greater than 0.");
            }
            break;
          case 2:
            // Start option is selected
            System.out.println("Operation: Start()");
            mda.start();
            break;
          case 3:
            // PayCredit option is selected
            System.out.println("Operation : PayCredit()");
            mda.payCredit();
            break;
          case 4:
            // Reject option is selected
            System.out.println("Operation : Reject()");
            mda.Reject();
            break;
          case 5:
            // Cancel option is selected
            System.out.println("Operation : Cancel()");
            mda.Cancel();
            break;
          case 6:
            // Approved option is selected
            System.out.println("Operation : Approved()");
            mda.Approved();
            break;
          case 7:
            // PayCash option is selected
            System.out.println("Operation : PayCash(int c)");
```

```
                System.out.println("Enter value of parameter c :");
                cash = Integer.parseInt(br.readLine());
                ds.setTempCash(cash);
                mda.payCash();
                break;
            case 8:
                // StartPump is selected
                System.out.println("Operation : StartPump()");
                mda.selectGas(1);
                mda.StartPump();
                break;
            case 9:
                // PumpGallon is selected
                System.out.println("Operation : PumpGallon()");
                mda.Pump();
                break;
            case 10:
                // Stop Pump is selected
                System.out.println("Operation : StopPump()");
                mda.StopPump();
                mda.Reciept();
                break;

          }
        } catch (IOException ie) {
            Logger.getLogger(GasPump1.class.getName()).log(Level.SEVERE, null, ie);
        }
      } while (userChoice != 0);

  }

}
```

**GasPump2.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.ip;

import gaspumpsystem.datastore.Data_Store;
import gaspumpsystem.factory.AbstractFactory;
import gaspumpsystem.factory.GPFactory2;
import gaspumpsystem.mdaefsm.MDA_EFSM;
import gaspumpsystem.op.Output;
import gaspumpsystem.state.State;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
```

```
 * @author jaiminsanghvi
 */
public class GasPump2 {

   public static AbstractFactory af = new GPFactory2();
   int userChoice = 0;
   int paymentOption = 0;
   float priceA = 0, priceB = 0;

   public void run() {

      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

      MDA_EFSM mda = new MDA_EFSM();
      Data_Store ds = af.getDataStore();
      State.setOutput(new Output(af, ds));

      //  Print GasPump-3 menu
      System.out.println("GasPump-2 : Menu Of Operations");
      System.out.println("1. Active(float a, float b)");
      System.out.println("2. Start()");
      System.out.println("3. PayCredit()");
      System.out.println("4. Reject()");
      System.out.println("5. Cancel()");
      System.out.println("6. Approved()");
      System.out.println("7. Super()");
      System.out.println("8. Regular()");
      System.out.println("9. StartPump()");
      System.out.println("10. PumpGallon()");
      System.out.println("11. StopPump()");
      System.out.println("0. Quit from the system");

      System.out.println("\n\n");

      System.out.println("Please select an operation");

      do {
        try {
          //  display list of operations
          System.out.println();
          System.out.println("GasPump-2 Execution");
          System.out.println("Select an Operation:");
          System.out.println("1-Activate,2-Start,3-PayCredit,4-Reject,5-Cancel,6-Approved,7-
Super,8-Regular,9-StartPump,10-PumpGallon,11-StopPump, 0-Quit");

          //Get user's choice
          userChoice = Integer.parseInt(br.readLine());

          switch (userChoice) {
            case 1:
              //  Activate option selected
              System.out.println("Operation: Active(float a, float b)");
              System.out.println("Enter value of parameter a : ");
              priceA = Float.parseFloat(br.readLine());
              System.out.println("Enter value of parameter b : ");
              priceB = Float.parseFloat(br.readLine());
```

```
            if (priceA > 0 && priceB > 0) {
                ds.setPriceTFA(priceA);
                ds.setPriceTFB(priceB);
                mda.active();
            } else {
                System.out.println("Invalid value. Values should be greater than 0.");
            }
            break;
        case 2:
            // Start option is selected
            System.out.println("Operation: Start()");
            mda.start();
            break;
        case 3:
            // PayCredit option is selected
            System.out.println("Operation : PayCredit()");
            mda.payCredit();
            break;
        case 4:
            // Reject option is selected
            System.out.println("Operation : Reject()");
            mda.Reject();
            break;
        case 5:
            // Cancel option is selected
            System.out.println("Operation : Cancel()");
            mda.Cancel();
            break;
        case 6:
            System.out.println("Operation : Approved()");
            mda.Approved();
            break;
        case 7:
            // Super option is selected
            System.out.println("Operation : Super()");
            mda.selectGas(2);
            break;
        case 8:
            // Regulare gas option is selected
            System.out.println("Operation : Regular()");
            mda.selectGas(1);
            break;
        case 9:
            // StartPump option is selected
            System.out.println("Operation : StartPump()");
            mda.StartPump();
            break;
        case 10:
            // PumpGallon option is selected
            System.out.println("Operation : PumpGallon()");
            mda.Pump();
            break;
        case 11:
            // StopPump option is selected
            System.out.println("Operation : StopPump()");
            mda.StopPump();
```

```
                mda.Reciept();
                break;
            }
        } catch (IOException ie) {
            Logger.getLogger(GasPump1.class.getName()).log(Level.SEVERE, null, ie);
        }
    } while (userChoice != 0);

  }
}
```

**GasPump3.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.ip;

import gaspumpsystem.datastore.Data_Store;
import gaspumpsystem.factory.AbstractFactory;
import gaspumpsystem.factory.GPFactory2;
import gaspumpsystem.factory.GPFactory3;
import gaspumpsystem.mdaefsm.MDA_EFSM;
import gaspumpsystem.op.Output;
import gaspumpsystem.state.State;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author jaiminsanghvi
 */
public class GasPump3 {

   public static AbstractFactory af = new GPFactory3();
   int userChoice = 0;
   int paymentOption = 0;
   float priceA = 0, priceB = 0;
   float cash = 0;

   public void run() {

      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

      MDA_EFSM mda = new MDA_EFSM();
      Data_Store ds = af.getDataStore();
      State.setOutput(new Output(af, ds));

      //  Print GasPump-3 menu
      System.out.println("GasPump-3 : Menu Of Operations");
      System.out.println("1. Active(float a, float b)");
      System.out.println("2. Start()");
```

```
System.out.println("3. PayCash(float c)");
System.out.println("4. Cancel()");
System.out.println("5. Premium()");
System.out.println("6. Regular()");
System.out.println("7. StartPump()");
System.out.println("8. PumpLiter()");
System.out.println("9. StartPump()");
System.out.println("10. Reciept()");
System.out.println("11. NoReceipt()");
System.out.println("0. Quit from the system");

System.out.println("\n\n");

System.out.println("Please select an operation");

do {
  try {
    //  display list of operations
    System.out.println();
    System.out.println("GasPump-3 Execution");
    System.out.println("Select an Operation:");
    System.out.println("1-Activate,2-Start,3-PayCash(float c),4-Cancel,5-Premium,6-
Regular,7-StartPump,8-PumpLiter,9-StopPump,10-Receipt,11-NoReciept, 0-Quit");

    //Get user's choice
    userChoice = Integer.parseInt(br.readLine());

    switch (userChoice) {
      case 1:
        //  Activate option selected
        System.out.println("Operation: Active(float a, float b)");
        System.out.println("Enter value of parameter a : ");
        priceA = Float.parseFloat(br.readLine());
        System.out.println("Enter value of parameter b : ");
        priceB = Float.parseFloat(br.readLine());
        if (priceA > 0 && priceB > 0) {
          ds.setPriceTFA(priceA);
          ds.setPriceTFB(priceB);
          mda.active();
        } else {
          System.out.println("Invalid value. Values should be greater than 0.");
        }
        break;
      case 2:
        // Start option is selected
        System.out.println("Operation: Start()");
        mda.start();
        break;
      case 3:
        // PayCash option is selected
        System.out.println("Operation : PayCash(float c)");
        System.out.println("Enter value of parameter c :");
        cash = Float.parseFloat(br.readLine());
        ds.setCashTF(cash);
        mda.payCash();
        break;
```

```
                case 4:
                    // Cancel option is selected
                    System.out.println("Operation : Cancel()");
                    mda.Cancel();
                    break;
                case 5:
                    // Premium option is selected
                    System.out.println("Operation : Premium()");
                    mda.selectGas(2);
                    break;
                case 6:
                    // Regular option is selected
                    System.out.println("Operation : Regular()");
                    mda.selectGas(1);
                    break;
                case 7:
                    // Start pump
                    System.out.println("Operation : StartPump()");
                    mda.StartPump();
                    break;
                case 8:
                    // Fill gas in liter unit
                    System.out.println("Operation : PumpLiter()");
                    mda.Pump();
                    break;
                case 9:
                    // StopPump
                    System.out.println("Operation : StopPump()");
                    mda.StopPump();
                    break;
                case 10:
                    // Print receipt with number of liiter gas and total price
                    System.out.println("Operation : Receipt()");
                    mda.Reciept();
                    break;
                case 11:
                    // Not print a receipt
                    System.out.println("Operation : NoReceipt()");
                    break;

            }
        } catch (IOException ie) {
            Logger.getLogger(GasPump1.class.getName()).log(Level.SEVERE, null, ie);
        }
    } while (userChoice != 0);
  }
}
```

## Package gaspumpsystem.state // state classes

**State.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.state;

import gaspumpsystem.op.Output;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class State {

   private int id;
   protected static Output op;

   public abstract void Active();

   public abstract void Start();

   public abstract void PayCredit();

   public abstract void PayCash();

   public abstract void Approved();

   public abstract void Reject();

   public abstract void SelectGas(int g);

   public abstract void Cancel();

   public abstract void StartPump();

   public abstract void Pump();

   public abstract void StopPump();

   public abstract void Receipt();

   public abstract void NoReceipt();

   public void setID(int id) {
      this.id = id;
   }

   public int getID() {
      return id;
   }

   public static void setOutput(Output op) {
```

```
        State.op = op;
    }
}
```

**Start.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.state;

/**
 *
 * @author jaiminsanghvi
 */
public class Start extends State {

    @Override
    public void Active() {
        op.StoreData();     // call StoreData action for Active event
        System.out.println("GasPump is activated. Now you can proceed further for other
operations.");
    }

    @Override
    public void Start() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void PayCredit() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void PayCash() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Approved() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Reject() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
```

```
    public void SelectGas(int g) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Cancel() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StartPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Pump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StopPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Receipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void NoReceipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

}
```

S0.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.state;

/**
 *
 * @author jaiminsanghvi
 */
public class S0 extends State {
```

```
    @Override
    public void Active() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Start() {      // start pump
        op.PayMsg();           // call PayMsg action for Start event
    }

    @Override
    public void PayCredit() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void PayCash() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Approved() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Reject() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void SelectGas(int g) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Cancel() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StartPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Pump() {
```

```
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StopPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Receipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void NoReceipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

}
```

**S1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.state;

/**
 *
 * @author jaiminsanghvi
 */
public class S1 extends State {

    @Override
    public void Active() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Start() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void PayCredit() {
        System.out.println("Waiting for an approval.");
    }

    @Override
```

```
    public void PayCash() {
        op.StoreCash();     // call StoreCash action for PayCash event
        op.SetW(0);         // call SetW action for PayCash event
        op.DisplayMenu();   // call DisplayMenu action for PayCash event
    }

    @Override
    public void Approved() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Reject() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void SelectGas(int g) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Cancel() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StartPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Pump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StopPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Receipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void NoReceipt() {
```

```
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

}
```

**S2.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.state;

/**
 *
 * @author jaiminsanghvi
 */
public class S2 extends State {

  @Override
  public void Active() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public void Start() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public void PayCredit() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public void PayCash() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public void Approved() {
    op.SetW(1);       // call SetW action for Approved event
    op.DisplayMenu();  // call DisplayMenu for Approved event
  }

  @Override
  public void Reject() {
    op.RejectMsg();    // call RejectMsg for Reject event
  }

  @Override
```

```
    public void SelectGas(int g) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Cancel() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StartPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Pump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StopPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Receipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void NoReceipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

}
```

S3.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.state;

/**
 *
 * @author jaiminsanghvi
 */
public class S3 extends State {
```

```
    @Override
    public void Active() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Start() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void PayCredit() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void PayCash() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Approved() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Reject() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void SelectGas(int g) {
        op.SetPrice(g);     // call SelectPrice action for SelectGas event
    }

    @Override
    public void Cancel() {
        op.CancelMsg();     // call CancelMsg action for Cancel event
    }

    @Override
    public void StartPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Pump() {
```

```
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StopPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Receipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void NoReceipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

}
```

**S4.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.state;

/**
 *
 * @author jaiminsanghvi
 */
public class S4 extends State {

    @Override
    public void Active() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Start() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void PayCredit() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }
```

```
    @Override
    public void PayCash() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Approved() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Reject() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void SelectGas(int g) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Cancel() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StartPump() {
        op.setInitValues(); // call setInitValues action for StartPump event
        op.ReadyMsg();      // call ReadyMsg action for StartPump event
    }

    @Override
    public void Pump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StopPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Receipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void NoReceipt() {
```

```
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

}
```

**S5.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.state;

/**
 *
 * @author jaiminsanghvi
 */
public class S5 extends State {

  @Override
  public void Active() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public void Start() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public void PayCredit() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public void PayCash() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public void Approved() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public void Reject() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }
```

```
    @Override
    public void SelectGas(int g) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Cancel() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StartPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Pump() {
        op.PumpGasUnit();   // call PumpGasUnit action for Pump event
        op.GasPumpedMsg();  // call GasPumpedMsg action for Pump event
    }

    @Override
    public void StopPump() {
        op.StopMsg();       // call StopMsg action for SopPump event
    }

    @Override
    public void Receipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void NoReceipt() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

}
```

**S6.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.state;

/**
 *
 * @author jaiminsanghvi
 */
public class S6 extends State {
```

```
    @Override
    public void Active() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Start() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void PayCredit() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void PayCash() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Approved() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Reject() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void SelectGas(int g) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Cancel() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StartPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
```

```
    public void Pump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void StopPump() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void Receipt() {
        op.PrintReceipt();  // call PrintReceipt action for Receipt event
    }

    @Override
    public void NoReceipt() {
        System.out.println("Exit with No Reciept.");
    }
}
```

## Package gaspumpsystem.mdaefsm // mda-efsm classes

### MDA_EFSM.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.mdaefsm;

import gaspumpsystem.state.*;

/**
 *
 * @author jaiminsanghvi
 */
public class MDA_EFSM {
//array of states.

   private State[] LS = new State[]{
      new Start(), //id 0
      new S0(), //id 1
      new S1(), //id 2
      new S2(), //id 3
      new S3(), //id 4
      new S4(), //id 5
      new S5(), //id 6
      new S6() //id 7
   };

   private State s = LS[0];

   public void active() {
      int curId = s.getID();      //get current state id

      if (curId == 0) {
         //  calls the state's active method
         s.Active();     //  call Active() method of the current start state
         switch (curId) {
            case 0:
               s = LS[1];
               s.setID(1);
               System.out.println("Changed state from start to S0");
               break;
            default:
               break;
         }
      } else {
         System.out.println("\n Caution : You can't active pump from middle operation.");
      }
   }

   public void start() {
      int curId = s.getID();      //get current state id
      if (curId == 1) {
```

```
        // calls the state's start method
        s.Start();      // call Start()
        switch (curId) {
           case 1:
              s = LS[2];
              s.setID(2);
              System.out.println("Changed state from S0 to S1");
              break;
           default:
              break;
        }
     } else {
        System.out.println("\n Caution : You have to first activate the GasPump. \n");
     }
  }

  public void payCredit() {
     int curId = s.getID();  // get current state id

     if (curId == 2) {
        //  calls the state's payCredit method
        s.PayCredit();  // call payCredit()
        switch (curId) {
           case 2:
              s = LS[3];
              s.setID(3);
              System.out.println("Changed state from S1 to S2");
              break;
           default:
              break;
        }
     } else {
        System.out.println("\n Caution : You can't PayByCredit unit you start a pump. \n");
     }
  }

  public void payCash() {
     int curId = s.getID();  // get current state id

     if (curId == 2) {
        //  calls the state's payCash method
        s.PayCash();    // call payCash()
        switch (curId) {
           case 2:
              s = LS[4];
              s.setID(4);
              System.out.println("Changed state from S1 to S2");
              break;
           default:
              break;
        }
     } else {
        System.out.println("\n Caution : You can't PayByCash unit you start a pump. \n");
     }
  }
```

```
  public void Approved() {
    int curId = s.getID();  // get current state id

    if (curId == 3) {
      //  calls the state's approved method
      s.Approved();   // call Approved()
      switch (curId) {
        case 3:
          s = LS[4];
          s.setID(4);
          System.out.println("Changed state from S2 to S4");
          System.out.println("Credit card approved");
          break;
        default:
          break;
      }
    } else {
      System.out.println("\n Caution : You can't approve payment by Credit option unit you
select PayCredit option. \n");
    }
  }

  public void Reject() {
    int curId = s.getID();     // get current state id

    if (curId == 3) {
      //  calls the state's reject method
      s.Reject();    // call Reject()
      switch (curId) {
        case 3:
          s = LS[1];
          s.setID(1);
          System.out.println("Changed state from S2 to S0");
          break;
        default:
          break;
      }
    } else {
      System.out.println("\n Caution : You can't reject payment by Credit option unit you select
PayCredit option. \n");
    }
  }

  public void Cancel() {
    int curId = s.getID();  // get current state id

    if (curId == 4) {
      //  calls the state's cancel method
      s.Cancel();    // call Cancel()
      switch (curId) {
        case 4:
          s = LS[1];
          s.setID(1);
          System.out.println("Changed state from S3 to S0");
          break;
        default:
```

```
                break;
            }
        } else {
            System.out.println("\n Caution : You can't cancel the operation until you payment or card
has been approved. \n");
        }
    }

    public void selectGas(int i) {
        int curId = s.getID();      // get current state id

        if (curId == 4) {
            //  calls the state's SelectGas method
            s.SelectGas(i); // call SelectGas
            switch (curId) {
                case 4:
                    s = LS[5];
                    s.setID(5);
                    System.out.println("Changed state from S3 to S4");
                    break;
                default:
                    break;
            }
        } else {
            System.out.println("\n Caution : You can't select the operation until you payment or card
has been approved. \n");
        }
    }

    public void StartPump() {
        int curId = s.getID();  // get current state id

        if (curId == 5) {
            //  calls the state's start pump method
            s.StartPump();  // call StartPump()
            switch (curId) {
                case 5:
                    s = LS[6];
                    s.setID(6);
                    System.out.println("Changed state from S4 to S5");
                    break;
                default:
                    break;
            }
        } else {
            System.out.println("Caution : you can't start pump without selecting a gas.");
        }
    }

    public void Pump() {
        int curId = s.getID();  // get current state id

        if (curId == 6) {
            //  calls the state's PumpGalllon method
            s.Pump();   // call Pump()
            switch (curId) {
```

```
            case 6:
               s = LS[6];
               s.setID(6);

               System.out.println("Changed state from S5 to S5");
               break;
            default:
               break;
         }
      } else {
         System.out.println("Caution : you can't pump gas without starting a pump.");
      }
   }

   public void StopPump() {
      int curId = s.getID();  // get current state id
      if (curId == 6) {
         //  calls the state's StopPump method
         s.StopPump();   // call StopPump()
         switch (curId) {
            case 6:
               s = LS[7];
               s.setID(7);
               System.out.println("Changed state from S5 to S6");
               break;
            default:
               break;
         }
      } else {
         System.out.println("Caution : you can't stop pump without stating pump.");
      }
   }

   public void Reciept() {
      int curId = s.getID();  // get current state id
      if (curId == 7) {
         //  calls the state's Receipt method
         s.Receipt();      // call Receipt
         switch (curId) {
            case 7:
               s = LS[1];
               s.setID(1);
               System.out.println("Changed state from S6 to S0");
               break;
            default:
               break;
         }
      } else {
         System.out.println("Caution : you can't print a receipt without pumping a gas.");
      }
   }

   public void NoReciept() {
      int curId = s.getID();     // get current state id
      if (curId == 7) {
         //  calls the state's NoReceipt method
```

```
            s.NoReceipt();      // call NoReceipt();
            switch (curId) {
               case 7:
                  s = LS[1];
                  s.setID(1);
                  System.out.println("Changed state from S6 to S0");
                  break;
               default:
                  break;
            }
         } else {
            System.out.println("Caution : you can't print a receipt without pumping a gas.");
         }
      }
   }
```

## Package gaspumpsystem.datastore // datastore classes

### Data_Store.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.datastore;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class Data_Store {

   // defination of setter and getter methods for datastore class
   // setter and getter methods of gasPrice and cash for GasPump-1
   public abstract void setPriceA(int a);

   public abstract int getPriceA();

   public abstract void setTempA(int a);

   public abstract int getTempA();

   public abstract void setTempCash(int c);

   public abstract int getTempCash();

   public abstract void setCash(int c);

   public abstract int getCash();

   //set prices of gas and cash mehods for GasPump-2 and GasPump-3
   public abstract void setPriceTFA(float a);

   public abstract float getPriceTFA();

   public abstract void setPriceTFB(float b);

   public abstract float getPriceTFB();

   public abstract void setPriceF(float a);

   public abstract float getPriceF();

   public abstract void setCashF(float c);

   public abstract float getCashF();

   public abstract void setCashTF(float c);

   public abstract float getCashTF();
```

```
    //setter and getter methods defination of W, Quantity and Total of Filled Gas
    public abstract void setW(int w);

    public abstract int getW();

    public abstract void setQuantity(int q);

    public abstract int getQuantity();

    public abstract void setTotal(float total);

    public abstract float getTotal();
}
```

**DS_GP1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.datastore;

/**
 *
 * @author jaiminsanghvi
 */
public class DS_GP1 extends Data_Store {

    private static int priceA;
    private static int tempPrice, tempCash;
    private static int cash;
    private static int w;
    private static int quantity;
    private static float total;

    @Override
    public void setPriceA(int a) {  // set price of A
        DS_GP1.priceA = a;
    }

    @Override
    public int getPriceA() {       // get price of A
        return priceA;
    }

    @Override
    public void setCash(int c) {    // set cash
        DS_GP1.cash = c;
    }

    @Override
    public int getCash() {        // get cash
        return cash;
    }

    @Override
```

```
    public void setTempA(int tempPrice) {   // set temporary price value
        this.tempPrice = tempPrice;
    }

    @Override
    public int getTempA() {              // get temporary price value
        return tempPrice;
    }

    @Override
    public void setCashF(float c) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public float getCashF() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setTempCash(int c) {    // set temp cash value
        DS_GP1.tempCash = c;
    }

    @Override
    public int getTempCash() {        // get temp cash value
        return tempCash;
    }

    @Override
    public void setW(int w) {         // set value of W
        DS_GP1.w = w;
    }

    @Override
    public int getW() {              // get value of W
        return w;
    }

    @Override
    public void setQuantity(int q) {    // set filled gas quantity
        DS_GP1.quantity = q;
    }

    @Override
    public int getQuantity() {        // get fiiled gas quantity
        return quantity;
    }

    @Override
    public void setTotal(float total) { // set total amount of filled gas
        DS_GP1.total = total;
    }
```

```
    @Override
    public float getTotal() {          // get total amount of filled gas
        return total;
    }

    @Override
    public void setPriceTFA(float a) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public float getPriceTFA() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setPriceTFB(float b) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public float getPriceTFB() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setPriceF(float a) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public float getPriceF() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setCashTF(float c) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public float getCashTF() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }
}
```

**DS_GP2.java**

```
/*
```

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.datastore;

/**
 *
 * @author jaiminsanghvi
 */
public class DS_GP2 extends Data_Store {

    private static float priceTFA;
    private static float priceTFB;
    private static float priceF;
    private static int w;
    private static int quantity;
    private static float total;

    @Override
    public void setPriceA(int a) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public int getPriceA() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setCash(int c) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public int getCash() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setCashF(float c) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public float getCashF() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
```

```
    public void setPriceF(float a) {    // set final price of selected gas
        this.priceF = a;
    }

    @Override
    public float getPriceF() {          // get final price of selected gas
        return priceF;
    }

    @Override
    public void setTempA(int a) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public int getTempA() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setTempCash(int c) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public int getTempCash() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setW(int w) {               // set value of w for different payment method
        this.w = w;
    }

    @Override
    public int getW() {                     // get value of w for different payment method
        return w;
    }

    @Override
    public void setQuantity(int q) {        // set quantity of filled gas
        this.quantity = q;
    }

    @Override
    public int getQuantity() {              // get quantity of filled gas
        return quantity;
    }

    @Override
    public void setTotal(float total) {     // set total value of filled gas
        DS_GP2.total = total;
```

```
    }

    @Override
    public float getTotal() {              // get total value filled gas
        return total;
    }

    @Override
    public void setPriceTFA(float a) {     // set temp value for regular gas
        this.priceTFA = a;
    }

    @Override
    public float getPriceTFA() {           // get temp value for regular gas
        return priceTFA;
    }

    @Override
    public void setPriceTFB(float b) {     // set temp value for super gas
        this.priceTFB = b;
    }

    @Override
    public float getPriceTFB() {           // set temp value for super gas
        return priceTFB;
    }

    @Override
    public void setCashTF(float c) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public float getCashTF() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

}
```

**DS_GP3.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.datastore;

/**
 *
 * @author jaiminsanghvi
 */
public class DS_GP3 extends Data_Store {

    private static float priceTFA;
```

```
    private static float priceTFB;
    private static float priceF;
    private static float cashF;
    private static float cashTF;
    private static int w;
    private static int quantity;
    private static float total;

    @Override
    public void setPriceA(int a) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public int getPriceA() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setCash(int c) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public int getCash() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setCashF(float c) {     // set cash value for pump-3
        this.cashF = c;
    }

    @Override
    public float getCashF() {          // get cash value for pump-3
        return cashF;
    }

    @Override
    public void setTempA(int a) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public int getTempA() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public void setTempCash(int c) {
```

```
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public int getTempCash() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
  }

  @Override
  public void setW(int w) {          // set value of w for different payment method
    this.w = w;
  }

  @Override
  public int getW() {               // get value of w for different payment method
    return w;
  }

  @Override
  public void setQuantity(int q) {    // set quantity of filled gas
    this.quantity = q;
  }

  @Override
  public int getQuantity() {         // get quantity of filled gas
    return quantity;
  }

  @Override
  public void setTotal(float total) { // set total amount of filled gas
    DS_GP3.total = total;
  }

  @Override
  public float getTotal() {          // get total amount of filled gas
    return total;
  }

  @Override
  public void setPriceTFA(float a) {     // set temp price of regular gas
    this.priceTFA = a;
  }

  @Override
  public float getPriceTFA() {       // get temp price of regular gas
    return priceTFA;
  }

  @Override
  public void setPriceTFB(float b) {  // set temp price of premium gas
    this.priceTFB = b;
  }

  @Override
```

```
    public float getPriceTFB() {       // get temp price of premium gas
        return priceTFB;
    }

    @Override
    public void setPriceF(float a) {       // set final price of gas
        this.priceF = a;
    }

    @Override
    public float getPriceF() {             // get final price of gas
        return priceF;
    }

    @Override
    public void setCashTF(float c) {       // set temporary cash value
        this.cashTF = c;
    }

    @Override
    public float getCashTF() {             // get temporary cash value
        return cashTF;
    }
}
```

## Package gaspumpsystem.factory//factory classes

## AbstractFactory.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.factory;

import gaspumpsystem.datastore.Data_Store;
import gaspumpsystem.op.*;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class AbstractFactory {

   // data store and action class methods defination for all concrete classes
   public static int flag;

   public abstract Data_Store getDataStore();

   public abstract StoreData getStoreData();

   public abstract PayMsg getPayMsg();

   public abstract StoreCash getStoreCash();

   public abstract DispMenu getDisplayMenu();

   public abstract RejectMsg getRejectMsg();

   public abstract SetW getSetW();

   public abstract SetPrice getSetPrice();

   public abstract ReadyMsg getReadyMsg();

   public abstract SetInitValues getSetInitValues();

   public abstract PumpGasUnit getPumpGasUnit();

   public abstract GasPumpedMsg getGasPumpedMsg();

   public abstract StopMsg getStopMsg();

   public abstract PrintReceipt getPrintReceipt();

   public abstract CancelMsg getCancelMsg();
}
```

**GPFactory1.java**

```
/*
```

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.factory;

import gaspumpsystem.datastore.DS_GP1;
import gaspumpsystem.datastore.Data_Store;

import gaspumpsystem.op.*;

/**
 *
 * @author jaiminsanghvi
 */
public class GPFactory1 extends AbstractFactory {

    // action methods implementation which provide objects to GasPump-1
    @Override
    public StoreData getStoreData() {
        StoreData sd = new StoreData1();
        sd.setDataObject(new DS_GP1());
        return sd;
    }

    @Override
    public PayMsg getPayMsg() {
        flag = 1;
        PayMsg pm = new PayMsg1();
        return pm;
    }

    @Override
    public StoreCash getStoreCash() {
        StoreCash sc = new StoreCash1();
        sc.setDataObject(new DS_GP1());
        return sc;
    }

    @Override
    public DispMenu getDisplayMenu() {
        DispMenu dm = new DispMenu1();
        return dm;
    }

    @Override
    public RejectMsg getRejectMsg() {
        RejectMsg rm = new RejectMsg1();
        return rm;
    }

    @Override
    public SetW getSetW() {
        SetW setW = new SetW1();
        setW.setDataObject(new DS_GP1());
        return setW;
```

```
}

@Override
public SetPrice getSetPrice() {
   SetPrice sp = new SetPrice1();
   sp.setDataObject(new DS_GP1());
   return sp;
}

@Override
public ReadyMsg getReadyMsg() {
   ReadyMsg readyM = new ReadyMsg1();
   return readyM;
}

@Override
public SetInitValues getSetInitValues() {
   SetInitValues init = new SetInitValues1();
   init.setDataObject(new DS_GP1());
   return init;
}

@Override
public PumpGasUnit getPumpGasUnit() {
   PumpGasUnit unit = new PumpGasUnit1();
   unit.setDataObject(new DS_GP1());
   return unit;
}

@Override
public GasPumpedMsg getGasPumpedMsg() {
   GasPumpedMsg pumpMsg = new GasPumpedMsg1();
   pumpMsg.setDataObject(new DS_GP1());
   return pumpMsg;
}

@Override
public StopMsg getStopMsg() {
   StopMsg stopM = new StopMsg1();
   return stopM;
}

@Override
public PrintReceipt getPrintReceipt() {
   PrintReceipt printM = new PrintReceipt1();
   printM.setDataObject(new DS_GP1());
   return printM;
}

@Override
public CancelMsg getCancelMsg() {
   CancelMsg cancelM = new CancelMsg1();
   return cancelM;
}

@Override
```

```
    public Data_Store getDataStore() {
        return new DS_GP1();
    }

}
```

**GPFactory2.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.factory;

import gaspumpsystem.datastore.DS_GP2;
import gaspumpsystem.datastore.Data_Store;
import gaspumpsystem.op.CancelMsg;
import gaspumpsystem.op.CancelMsg1;
import gaspumpsystem.op.DispMenu;
import gaspumpsystem.op.DispMenu1;

import gaspumpsystem.op.GasPumpedMsg;

import gaspumpsystem.op.GasPumpedMsg2;
import gaspumpsystem.op.PayMsg;
import gaspumpsystem.op.PayMsg1;
import gaspumpsystem.op.PrintReceipt;
import gaspumpsystem.op.PrintReceipt1;
import gaspumpsystem.op.PumpGasUnit;
import gaspumpsystem.op.PumpGasUnit1;
import gaspumpsystem.op.ReadyMsg;
import gaspumpsystem.op.ReadyMsg1;
import gaspumpsystem.op.RejectMsg;
import gaspumpsystem.op.RejectMsg1;
import gaspumpsystem.op.SetInitValues;
import gaspumpsystem.op.SetInitValues1;
import gaspumpsystem.op.SetPrice;
import gaspumpsystem.op.SetPrice2;
import gaspumpsystem.op.SetW;
import gaspumpsystem.op.SetW1;
import gaspumpsystem.op.StopMsg;
import gaspumpsystem.op.StopMsg1;
import gaspumpsystem.op.StoreCash;
import gaspumpsystem.op.StoreData;
import gaspumpsystem.op.StoreData2;

/**
 *
 * @author jaiminsanghvi
 */
public class GPFactory2 extends AbstractFactory {

    // action methods implementation which provide objects to GasPump-2
    @Override
    public StoreData getStoreData() {
        StoreData sd = new StoreData2();
```

```
        sd.setDataObject(new DS_GP2());
        return sd;
    }

    @Override
    public PayMsg getPayMsg() {
        flag = 2;
        PayMsg pm = new PayMsg1();
        return pm;
    }

    @Override
    public StoreCash getStoreCash() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    @Override
    public DispMenu getDisplayMenu() {
        DispMenu dm = new DispMenu1();
        return dm;
    }

    @Override
    public RejectMsg getRejectMsg() {
        RejectMsg rm = new RejectMsg1();
        return rm;
    }

    @Override
    public SetW getSetW() {
        SetW setW = new SetW1();
        setW.setDataObject(new DS_GP2());
        return setW;
    }

    @Override
    public SetPrice getSetPrice() {
        SetPrice sp = new SetPrice2();
        sp.setDataObject(new DS_GP2());
        return sp;
    }

    @Override
    public ReadyMsg getReadyMsg() {
        ReadyMsg readyM = new ReadyMsg1();
        return readyM;
    }

    @Override
    public SetInitValues getSetInitValues() {
        SetInitValues init = new SetInitValues1();
        init.setDataObject(new DS_GP2());
        return init;
    }
```

```
    @Override
    public PumpGasUnit getPumpGasUnit() {
        PumpGasUnit unit = new PumpGasUnit1();
        unit.setDataObject(new DS_GP2());
        return unit;
    }

    @Override
    public GasPumpedMsg getGasPumpedMsg() {
        GasPumpedMsg pumpMsg = new GasPumpedMsg2();
        pumpMsg.setDataObject(new DS_GP2());
        return pumpMsg;
    }

    @Override
    public StopMsg getStopMsg() {
        StopMsg stopM = new StopMsg1();
        return stopM;
    }

    @Override
    public PrintReceipt getPrintReceipt() {
        PrintReceipt printM = new PrintReceipt1();
        printM.setDataObject(new DS_GP2());
        return printM;
    }

    @Override
    public CancelMsg getCancelMsg() {
        CancelMsg cancelM = new CancelMsg1();
        return cancelM;
    }

    @Override
    public Data_Store getDataStore() {
        return new DS_GP2();
    }

}
```

**GPFactory3.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.factory;

import gaspumpsystem.datastore.DS_GP3;
import gaspumpsystem.datastore.Data_Store;
import gaspumpsystem.op.CancelMsg;
import gaspumpsystem.op.CancelMsg1;
import gaspumpsystem.op.DispMenu;
import gaspumpsystem.op.DispMenu1;
import gaspumpsystem.op.GasPumpedMsg;
```

```
import gaspumpsystem.op.GasPumpedMsg2;
import gaspumpsystem.op.PayMsg;
import gaspumpsystem.op.PayMsg1;
import gaspumpsystem.op.PrintReceipt;
import gaspumpsystem.op.PrintReceipt1;
import gaspumpsystem.op.PumpGasUnit;
import gaspumpsystem.op.PumpGasUnit2;
import gaspumpsystem.op.ReadyMsg;
import gaspumpsystem.op.ReadyMsg1;
import gaspumpsystem.op.RejectMsg;
import gaspumpsystem.op.RejectMsg1;
import gaspumpsystem.op.SetInitValues;
import gaspumpsystem.op.SetInitValues1;
import gaspumpsystem.op.SetPrice;
import gaspumpsystem.op.SetPrice2;
import gaspumpsystem.op.SetW;
import gaspumpsystem.op.SetW1;
import gaspumpsystem.op.StopMsg;
import gaspumpsystem.op.StopMsg1;
import gaspumpsystem.op.StoreCash;
import gaspumpsystem.op.StoreCash2;
import gaspumpsystem.op.StoreData;
import gaspumpsystem.op.StoreData2;

/**
 *
 * @author jaiminsanghvi
 */
public class GPFactory3 extends AbstractFactory {

    // action methods implementation which provide objects to GasPump-3
    @Override
    public StoreData getStoreData() {
        StoreData sd = new StoreData2();
        sd.setDataObject(new DS_GP3());
        return sd;
    }

    @Override
    public PayMsg getPayMsg() {
        flag = 3;
        PayMsg pm = new PayMsg1();
        return pm;
    }

    @Override
    public StoreCash getStoreCash() {
        StoreCash sc = new StoreCash2();
        sc.setDataObject(new DS_GP3());
        return sc;
    }

    @Override
    public DispMenu getDisplayMenu() {
        DispMenu dm = new DispMenu1();
        return dm;
```

```
}

@Override
public RejectMsg getRejectMsg() {
   RejectMsg rm = new RejectMsg1();
   return rm;
}

@Override
public SetW getSetW() {
   SetW setW = new SetW1();
   setW.setDataObject(new DS_GP3());
   return setW;
}

@Override
public SetPrice getSetPrice() {
   SetPrice sp = new SetPrice2();
   sp.setDataObject(new DS_GP3());
   return sp;
}

@Override
public ReadyMsg getReadyMsg() {
   ReadyMsg readyM = new ReadyMsg1();
   return readyM;
}

@Override
public SetInitValues getSetInitValues() {
   SetInitValues init = new SetInitValues1();
   init.setDataObject(new DS_GP3());
   return init;
}

@Override
public PumpGasUnit getPumpGasUnit() {
   PumpGasUnit unit = new PumpGasUnit2();
   unit.setDataObject(new DS_GP3());
   return unit;
}

@Override
public GasPumpedMsg getGasPumpedMsg() {
   GasPumpedMsg pumpMsg = new GasPumpedMsg2();
   pumpMsg.setDataObject(new DS_GP3());
   return pumpMsg;
}

@Override
public StopMsg getStopMsg() {
   StopMsg stopM = new StopMsg1();
   return stopM;
}

@Override
```

```
    public PrintReceipt getPrintReceipt() {
        PrintReceipt printM = new PrintReceipt1();
        printM.setDataObject(new DS_GP3());
        return printM;
    }

    @Override
    public CancelMsg getCancelMsg() {
        CancelMsg cancelM = new CancelMsg1();
        return cancelM;
    }

    @Override
    public Data_Store getDataStore() {
        return new DS_GP3();
    }

}
```

## Package gaspumpsystem.op // output classes

### Output.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.datastore.Data_Store;
import gaspumpsystem.factory.AbstractFactory;

/**
 *
 * @author jaiminsanghvi
 */
public class Output {

    // Object references for action classes
    AbstractFactory af;
    Data_Store ds;
    StoreData sd;
    PayMsg pm;
    StoreCash sc;
    SetW setW;
    RejectMsg rm;
    CancelMsg cancelM;
    SetPrice sp;
    SetInitValues init;
    DispMenu dm;
    PumpGasUnit unit;
    GasPumpedMsg pumpMsg;
    StopMsg stopM;
    PrintReceipt printR;
    ReadyMsg readyM;

    public Output(AbstractFactory af, Data_Store ds) {
        this.af = af;
        this.ds = ds;
    }

    public void StoreData() {
        sd = af.getStoreData();   //    get the StoreData object from AbstractFactory
        sd.StoreData();          //     call StoreData method of action class
        System.out.println("Your data has been stored.");
    }

    public void PayMsg() {
        pm = af.getPayMsg();      // get the PayMsg object from AbstractFactory ;
        pm.PayMsg();              // call PayMsg method of action class
    }

    public void StoreCash() {
        sc = af.getStoreCash();     // get the StoreCash object from AbstractFactory ;
```

```
    sc.StoreCash();          // call StoreCash method of action class
  }

  public void SetW(int wFlag) {
    setW = af.getSetW();       // get the SetW object from AbstractFactory ;
    setW.setW(wFlag);          // call setW method of action class
  }

  public void RejectMsg() {      // get the RejectMsg object from AbstractFactory ;
    rm = af.getRejectMsg();    // call RejectMsg method of action class
    rm.RejectMsg();
  }

  public void CancelMsg() {
    cancelM = af.getCancelMsg();   // get the Cancel object from AbstractFactory ;
    cancelM.CancelMsg();          // call CancelMsg method of action class
  }

  public void SetPrice(int g) {
    sp = af.getSetPrice();        // get the StorePrice object from AbstractFactory ;
    sp.SetPrice(g);              // call SetPrice method of action class
  }

  public void setInitValues() {
    init = af.getSetInitValues();  // get the SetInitValues object from AbstractFactory
    init.SetInitValues();          // call SetInitValues method of action class
  }

  public void DisplayMenu() {
    dm = af.getDisplayMenu();      // get the DisplayMenu object from AbstractFactory ;
    dm.DisplayMenu();             // call DisplayMenu method of action class
  }

  public void PumpGasUnit() {
    unit = af.getPumpGasUnit();    // get the PumpGasUnit object from AbstractFactory ;
    unit.PumpGasUnit();           // call PumpGasUnit method of action class
  }

  public void GasPumpedMsg() {
    pumpMsg = af.getGasPumpedMsg(); // get the GasPumpedMsg object from AbstractFactory
;
    pumpMsg.GasPumpedMsg();        // call GasPumpedMsg method of action class
  }

  public void StopMsg() {
    stopM = af.getStopMsg();       // get the StopMsg object from AbstractFactory ;
    stopM.StopMsg();              // call StopMsg method of action class
  }

  public void PrintReceipt() {
    printR = af.getPrintReceipt();  // get the PrintReceipt object from AbstractFactory ;
    printR.PrintReceipt();         // call PrintReceipt method of action class
  }

  public void ReadyMsg() {
    readyM = af.getReadyMsg();     // get the ReadyMsg object from AbstractFactory ;
```

```
      readyM.ReadyMsg();            // call ReadyMsg method of action class
   }
}
```

**StoreData.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.datastore.Data_Store;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class StoreData {

   protected Data_Store ds;

   public void setDataObject(Data_Store x) {
      ds = x;
   }

   // abstract method defination for StoreData action classes
   public abstract void StoreData();

}
```

**StoreData1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class StoreData1 extends StoreData {

   @Override
   public void StoreData() {
      int priceA = ds.getTempA(); // get temporary price
      ds.setPriceA(priceA);      // set temporary price
   }
}
```

**StoreData2.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class StoreData2 extends StoreData {

   @Override
   public void StoreData() {
      float priceA = ds.getPriceTFA();    // get temporary value of parameter A
      float priceB = ds.getPriceTFB();    // get temporary value of paraemeter B
      ds.setPriceF(priceA);              // set final price of parameter A
      ds.setPriceF(priceB);              // set final price of parameter B

   }

}
```

**StoreCash.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.datastore.Data_Store;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class StoreCash {

   protected static Data_Store ds;

   // abstract method defination for StoreCash action classes
   public abstract void StoreCash();

   public void setDataObject(Data_Store x) {
      ds = x;
   }
}
```

**StoreCash1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
```

```
 */
public class StoreCash1 extends StoreCash {

   @Override
   public void StoreCash() {
      if (ds != null) {
         int cash = ds.getTempCash();
         ds.setCash(cash);
         System.out.println("cash stored==" + cash);
      }
   }

}
```

**StoreCash2.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class StoreCash2 extends StoreCash {

   @Override
   public void StoreCash() {
      if (ds != null) {
         float cash = ds.getCashTF();
         ds.setCashF(cash);
         System.out.println("cash stored==" + cash);
      }
   }

}
```

**SetW.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.datastore.Data_Store;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class SetW {
```

```
    // abstract method defination for SetW action classes
    public abstract void setW(int w);

    protected Data_Store ds;

    public void setDataObject(Data_Store x) {
        ds = x;
    }
}
```

**SetW1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class SetW1 extends SetW {

    @Override
    public void setW(int w) {
        ds.setW(w);
    }

}
```

**StopMsg.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class StopMsg {

    // abstract method defination for StopMsg action classes

    public abstract void StopMsg();
}
```

**StopMsg1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
```

```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class StopMsg1 extends StopMsg {

    @Override
    public void StopMsg() {
        System.out.println("Pump has been stopped.");
    }

}
```

**SetPrice.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.datastore.Data_Store;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class SetPrice {

    // abstract method defination for SetW action classes
    public abstract void setW(int w);

    public abstract void SetPrice(int gasType);

    protected Data_Store ds;

    public void setDataObject(Data_Store x) {
        ds = x;
    }
}
```

**SetPrice1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;
```

```
/**
 *
 * @author jaiminsanghvi
 */
public class SetPrice1 extends SetPrice {

   @Override
   public void SetPrice(int gasType) {
      ds.setPriceA(ds.getTempA());
   }

   @Override
   public void setW(int w) {
      throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
   }

}
```

**SetPrice2.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class SetPrice2 extends SetPrice{

   @Override
   public void SetPrice(int g) {
      float price;
      if(g==1)
         price=ds.getPriceTFA();
      else
         price=ds.getPriceTFB();

      ds.setPriceF(price);

   }

   @Override
   public void setW(int w) {
      throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
   }

}
```

**SetInitValues.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.datastore.Data_Store;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class SetInitValues {

   // abstract method defination for SetW action classes
   public abstract void SetInitValues();

   protected Data_Store ds;

   public void setDataObject(Data_Store x) {
      ds = x;
   }
}
```

**SetInitValues1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class SetInitValues1 extends SetInitValues {

   @Override
   public void SetInitValues() {
      ds.setQuantity(0);
      ds.setTotal(0);
      System.out.println("Init Values : " + ds.getQuantity() + " " + ds.getTotal());
   }

}
```

**RejectMsg.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class RejectMsg {

   // abstract method defination for RejectMsg action classes
   public abstract void RejectMsg();
}
```

**RejectMsg1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class RejectMsg1 extends RejectMsg{
   @Override
   public void RejectMsg() {
      System.out.println("Credit card not approved.");    // print rejection message
   }
}
```

**ReadyMsg.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class ReadyMsg {

   // abstract method defination for ReadyMsg action classes

   public abstract void ReadyMsg();

}
```

**ReadyMsg1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class ReadyMsg1 extends ReadyMsg {

    @Override
    public void ReadyMsg() {
        System.out.println("Pump is ready for pumping a gas."); // print ready message
    }
}
```

**PumpGasUnit.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.datastore.Data_Store;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class PumpGasUnit {

    protected static Data_Store ds;

    // abstract method defination for PumpGasUnit action classes
    public abstract void PumpGasUnit();

    public void setDataObject(Data_Store x) {
        ds = x;
    }
}
```

**PumpGasUnit1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;
```

```
/**
 *
 * @author jaiminsanghvi
 */
public class PumpGasUnit1 extends PumpGasUnit{

   @Override
   public void PumpGasUnit() {
     int w= ds.getW();
     int cash ;
     int price ;
     int unit= ds.getQuantity()+1;

     if(w==1){
         ds.setQuantity(unit);   // set filled gas quanity
         System.out.println("Filled Gas Quntity : " +unit);
     } else if(w==0) {
        cash= ds.getCash();      // get temporary cash value
        price=ds.getPriceA();    // get price value
        if(cash>(price*unit)) {
           ds.setQuantity(unit);    // set quantity value
           System.out.println("Filled Gas Quntity : " +unit);
         }else {
           System.out.println("In sufficient amount to fill more gas");
         }
     }
   }

}
```

**PumpGasUnit2.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class PumpGasUnit2 extends PumpGasUnit {

   @Override
   public void PumpGasUnit() {
      int w = ds.getW();
      float cash;
      float price;
      int unit = ds.getQuantity() + 1;

      if (w == 0) {
         cash = ds.getCashF();
         price = ds.getPriceF();
         if (cash > (price * unit)) {
```

```
            ds.setQuantity(unit);
            System.out.println("Filled Gas Quntity : " + unit);
        } else {
            System.out.println("In sufficient amount to fill more gas");
        }
    }
  }

}
```

**PrintReceipt.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.datastore.DS_GP1;
import gaspumpsystem.datastore.Data_Store;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class PrintReceipt {

   protected static Data_Store ds;

   // abstract method defination for PrintReceipt action classes
   public abstract void PrintReceipt();

   public void setDataObject(Data_Store ds) {
      this.ds = ds;
   }
}
```

**PrintReceipt1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.factory.AbstractFactory;

/**
 *
 * @author jaiminsanghvi
 */
public class PrintReceipt1 extends PrintReceipt {

   @Override
```

```
   public void PrintReceipt() {

      float total = ds.getTotal();    // get total gas value
      if (AbstractFactory.flag == 1 || AbstractFactory.flag == 2) {
         System.out.println("\n\t****** RECEIPT ********");
         System.out.println("\tTotal Filled Gas Amount = " + total);
         System.out.println("\tGood Bye. Have a nice day !!!");
      } else if (AbstractFactory.flag == 3) {
         int quant = ds.getQuantity();
         System.out.println("\n\t****** RECEIPT ********");
         System.out.println("\tFilled Gas Quantity = " + quant);
         System.out.println("\tTotal Filled Gas Amount = " + total);
         System.out.println("\tGood Bye. Have a nice day !!!");
      }

   }

}
```

**PayMsg.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class PayMsg {

   // abstract method defination for PayMsg action classes
   public abstract void PayMsg();

}
```

**PayMsg1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.factory.AbstractFactory;

/**
 *
 * @author jaiminsanghvi
 */
public class PayMsg1 extends PayMsg{
```

```
    @Override
    public void PayMsg() {
        if (AbstractFactory.flag == 1) {
            System.out.println("Payment Option : ");
            System.out.println("1. PayByCash");
            System.out.println("2. PayByCredit");
        } else if(AbstractFactory.flag==2) {
            System.out.println("Payment Option : ");
            System.out.println("1. PayByCredit");
        } else if(AbstractFactory.flag==3) {
            System.out.println("Payment Option : ");
            System.out.println("1. PayByCash");
        }
    }

}
```

**GasPumpedMsg.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.datastore.Data_Store;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class GasPumpedMsg {

    protected static Data_Store ds;

    public void setDataObject(Data_Store x) {
        ds = x;        // set data object for output class
    }

    // abstract method defination for GasPumpMsg action classes
    public abstract void GasPumpedMsg();

}
```

**GasPumpedMsg1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
```

```
 *
 * @author jaiminsanghvi
 */
public class GasPumpedMsg1 extends GasPumpedMsg {

   @Override
   public void GasPumpedMsg() {
      int unit = ds.getQuantity();
      int price = ds.getPriceA();
      int cash = ds.getCash();
      int total;
      int w = ds.getW();

      if (w == 1) {
         total = unit * price;
         ds.setTotal(total);
         System.out.println("Total filled gas amount : " + total);
      } else if (w == 0) {
         if (cash > (price * unit)) {
            total = unit * price;
            ds.setTotal(total);
            System.out.println("Total filled gas amount : " + total);
         } else {
            System.out.println("In sufficient amount");
         }
      }
   }
}
```

**GasPumpedMsg2.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class GasPumpedMsg2 extends GasPumpedMsg {

   @Override
   public void GasPumpedMsg() {

      int unit = ds.getQuantity();
      float price = ds.getPriceF();
      float cash = 0;
      float total;
      int w = ds.getW();
      if (w == 1) {
         total = unit * price;
         ds.setTotal(total);
         System.out.println("Total filled gas amount : " + total);
      } else if (w == 0) {
         cash = ds.getCashF();
```

```
            if (cash > (price * unit)) {
                total = unit * price;
                ds.setTotal(total);
                System.out.println("Total filled gas amount : " + total);
            } else {
                System.out.println("In sufficient amount");
            }
        }
    }
}
```

**DispMenu.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class DispMenu {
    // abstract method defination for DispMenu action classes
    public abstract void DisplayMenu();
}
```

**DispMenu1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

import gaspumpsystem.factory.AbstractFactory;

/**
 *
 * @author jaiminsanghvi
 */
public class DispMenu1 extends DispMenu{
    @Override
    public void DisplayMenu() {
        if(AbstractFactory.flag==1) {
            System.out.println("Type of Gas Available : Regular");
        } else if(AbstractFactory.flag==2) {
            System.out.println("Type of Gas Available : \n1. Regular\n2. Super");
        } else if(AbstractFactory.flag==3) {
            System.out.println("Type of Gas Available : \n1. Regular\n2. Premium");
        }
    }
}
```

**CancelMsg.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
```

```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public abstract class CancelMsg {

   // abstract method defination for CancelMsg action classes

   public abstract void CancelMsg();
}
```

**CancelMsg1.java**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gaspumpsystem.op;

/**
 *
 * @author jaiminsanghvi
 */
public class CancelMsg1 extends CancelMsg {

   @Override
   public void CancelMsg() {
      System.out.println("Transaction cancelled.");
   }
}
```

# 7 Conclusions

The main aim of this project is to design three Gas Pump components using Model-Driven Architecture. This MDA_EFSM Model is decoupled from actual data and Implementation details. The System is designed using three different design patterns. In the State design pattern, I have used centralized pattern. It is used to recognize the present state of the system at any point of time and change state from one state to another state. Strategy design pattern is used to apply different functionalities of three different Gas Pump components at runtime. Abstract Factory design pattern is used to create families of object for different Gas-Pump components using concrete factory classes. This implementation helps to maintain the system with low coupling and high cohesive design.