CS586 Project

A20372828

Chris Mathew Dani

INTRODUCTION

This project's goal is to implement different software architecture styes. This includes various patterns used in software design. GasPump components are state-based components and are used to control simple gas pumps. Users can pay by cash or a credit card and is selectable by option. The gas pump may dispose different types of the gasoline. The price of the gasoline is provided when the gas pump is activated and can be integer or a floating point value. The detailed behavior of GasPump components is specified using he EFSM.

OVERVIEW OF PROJECT

It includes Gaspump system which performs different operations and actions based on the user's interest

There are two GasPump components: GasPump -1 and GasPump -2

The GasPump-1 component supports the following operations:

1. **Activate (float a, float b)**: gas pump is activated where a is the price of the Regular gas and b is the price of Super gas per gallon
2. **Start()** :start the transaction
3. **PayCredit()** :pay for gas by a credit card
4. **Reject()** : credit card is rejected
5. **Cancel()** : cancel the transaction
6. **Approved()** : credit card is approved
7. **Super()** : Super gas is selected
8. **Regular()** : Regular gas is selected
9. **StartPump()** : start pumping gas
10. **PumpGallon()** : one gallon of gas is disposed
11. **StopPump()** : stop pumping gas

The GasPump-2 component supports the following operations:

1. **Activate (int a, int b, int c)**: the gas pump is activated where a is the price of Regular gas, b is the price of Premium gas and c is the price of Super gas per liter
2. **Start()** :start the transaction
3. **PayCash(int c)** :pay for gas by cash, where c represents prepaid cash
4. **Cancel()** : cancel the transaction
5. **Premium()** :Premium gas is selected
6. **Regular()** : Regular gas is selected
7. **Super()** : Super gas is selected

8. **StartPump()** : start pumping gas
9. **PumpLiter()** : one liter of gas is disposed
10. **Stop()** : stop pumping gas
11. **Receipt()** : Receipt is requested
12. **NoReceipt()** : No receipt

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

Aspects that vary between two GasPump components:

a. Types of gasoline disposed

b. Types of payment

c. Display menu(s)

d. Messages

e. Receipts

f. Operation names and signatures

g. Data types

h. etc.

MODEL-DRIVEN ARCHITECTURE OF THE GASPUMP COMPONENTS

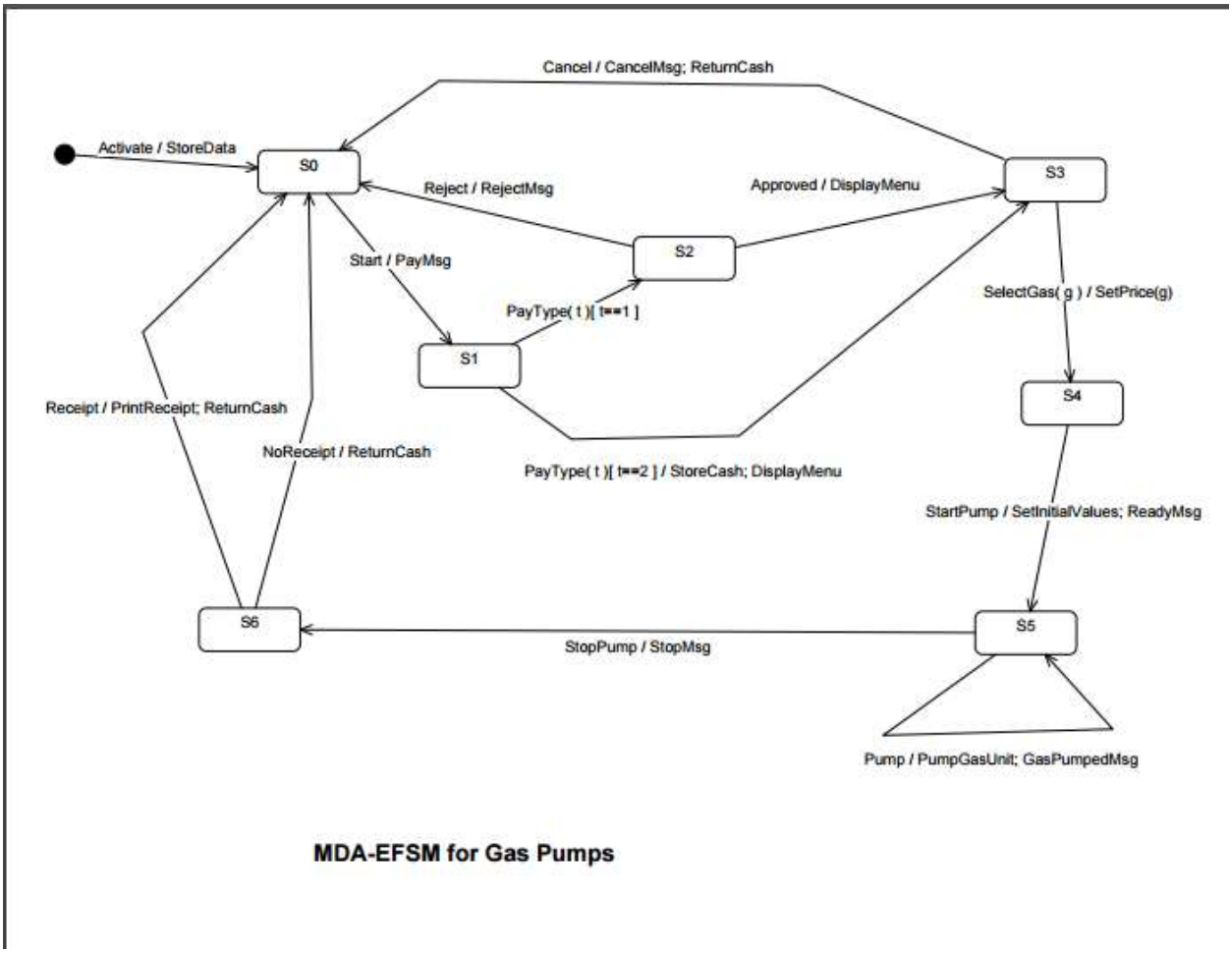
MDA-EFSM Events:

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

MDA-EFSM Actions:

- **StoreData** : stores price(s) for the gas from the temporary data store
- **PayMsg** : displays a type of payment method
- **StoreCash** : stores cash from the temporary data store
- **DisplayMenu** : display a menu with a list of selections
- **RejectMsg** : displays credit card not approved message
- **SetPrice(int g)** : set the price for the gas identified by g identifier
- **ReadyMsg** : displays the ready for pumping message
- **SetInitialValues** : set G (or L) and total to 0
- **PumpGasUnit** : disposes unit of gas and counts # of units disposed
- **GasPumpedMsg** : displays the amount of disposed gas
- **StopMsg** : stop pump message and receipt? msg (optionally)
- **PrintReceipt** : print a receipt
- **CancelMsg** : displays a cancellation message
- **ReturnCash** : returns the remaining cash

A STATE DIAGRAM OF THE MDA-EFSM GASPUMP COMPONENT:



Pseudocode for all Operations of the Input Processor GasPump1 & GasPump2

Operations of the Input Processor (GasPump-1)

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

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

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

d: is a pointer to the Data Store object

Operations of the Input Processor (GasPump-2)

```
Activate(int a, int b, int c) {
 if ((a>0)&&(b>0)&&(c>0)) {
 d->temp_a=a;
 d->temp_b=b;
 d->temp_c=c
```

```
 m->Activate()
 }
}
Start() {
 m->Start();
}
PayCash(float c) {
 if (c>0) {
 d->temp_cash=c;
 m->PayType(2)
 }
}
Cancel() {
 m->Cancel();
}
Super() {
 m->SelectGas(2);
}
Premium() {
 m->SelectGas(3);
}
Regular() {
 m->SelectGas(1);
}
StartPump() {
 m->StartPump();
}
PumpLiter() {
 if (d->cash<(d->L+1)*d->price)
 m->StopPump();
 else m->Pump()
}
Stop() {
 m->StopPump();
}
Receipt() {
 m->Receipt();
}
```

NoReceipt() {

m->NoReceipt();

}

Notice:

cash: contains the value of cash deposited

price: contains the price of the selected gas

L: contains the number of liters already

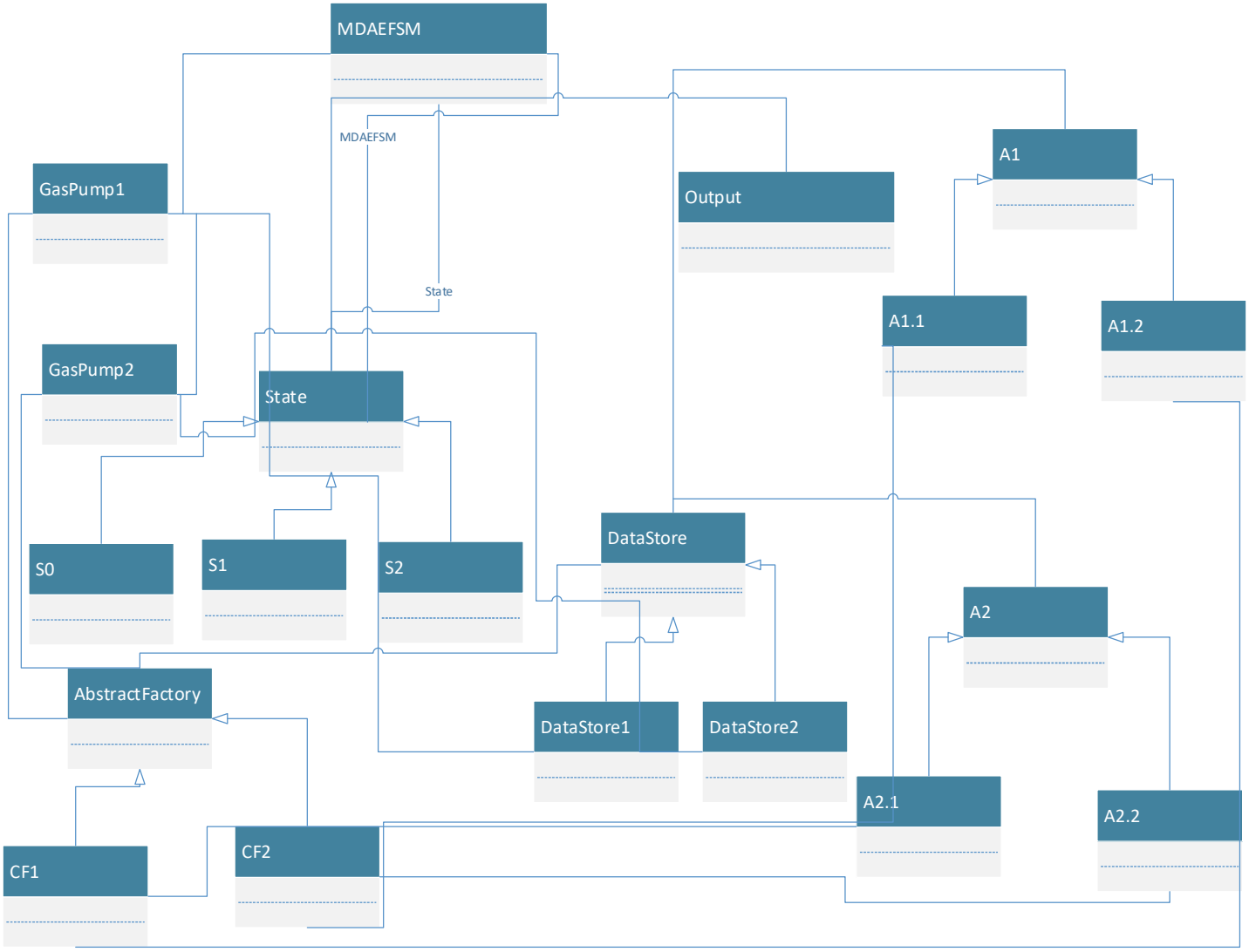
pumped

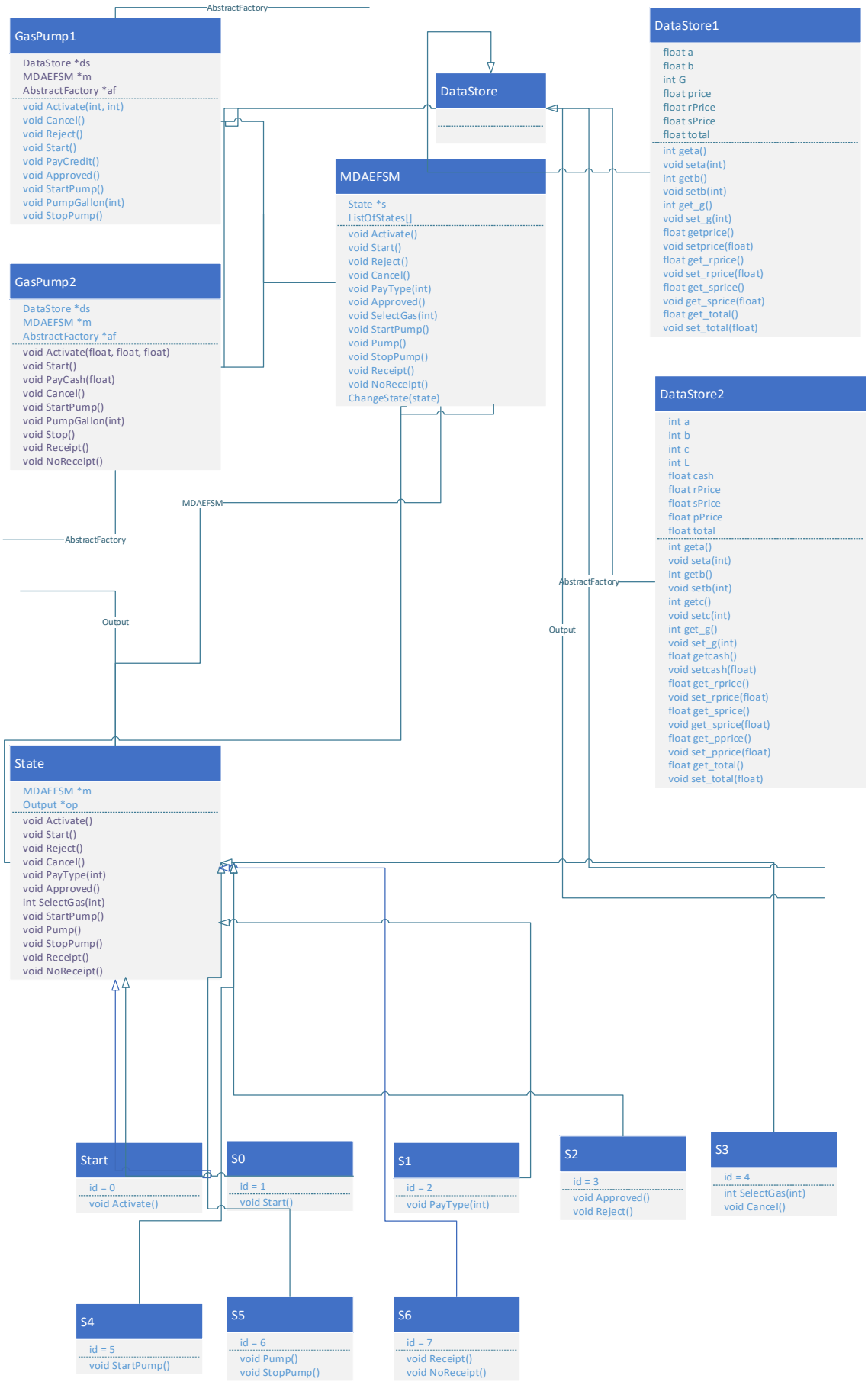
cash , L, price are in the data store

m: is a pointer to the MDA-EFSM object

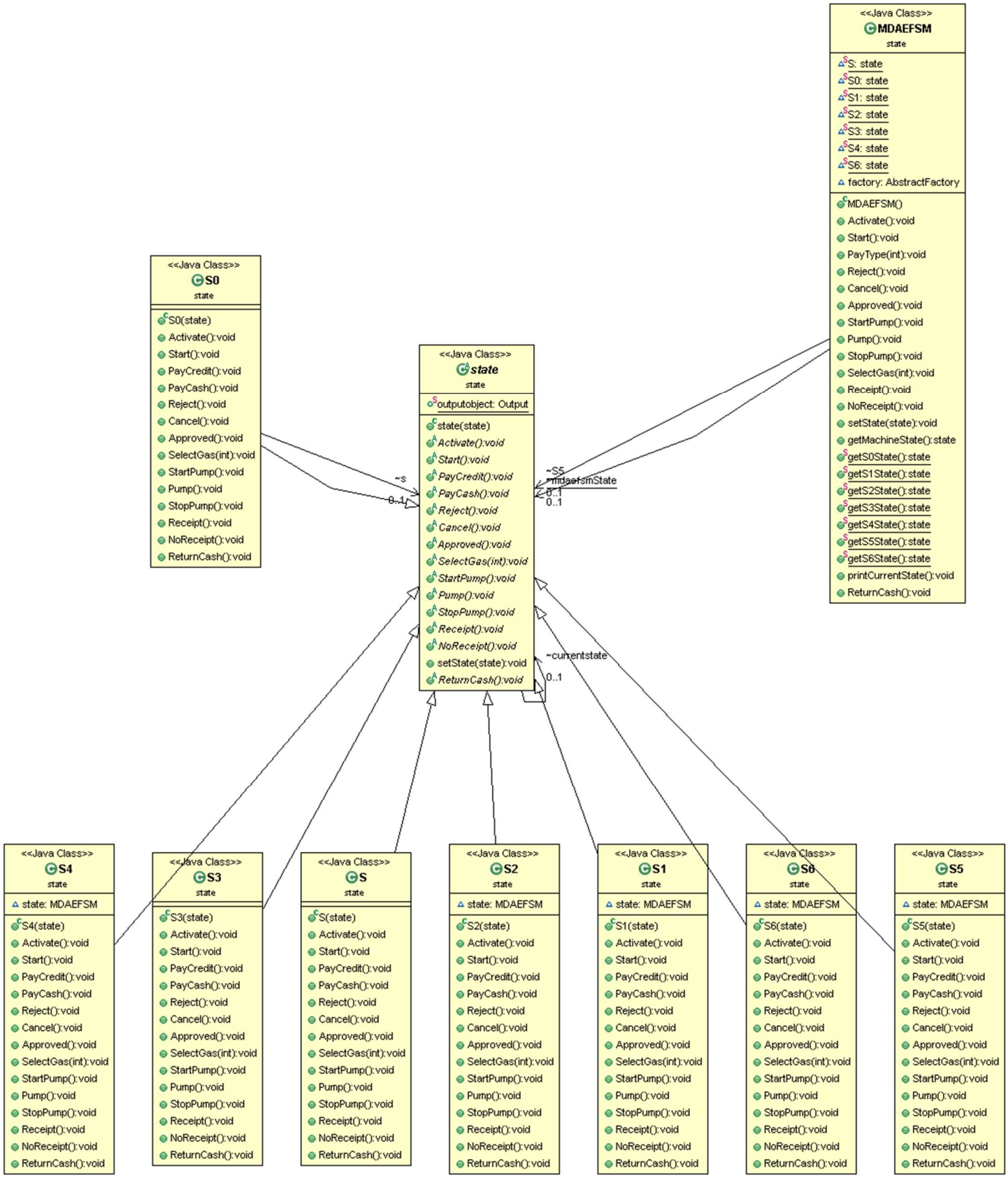
d: is a pointer to the Data Store object

HIGH LEVEL DESIGN

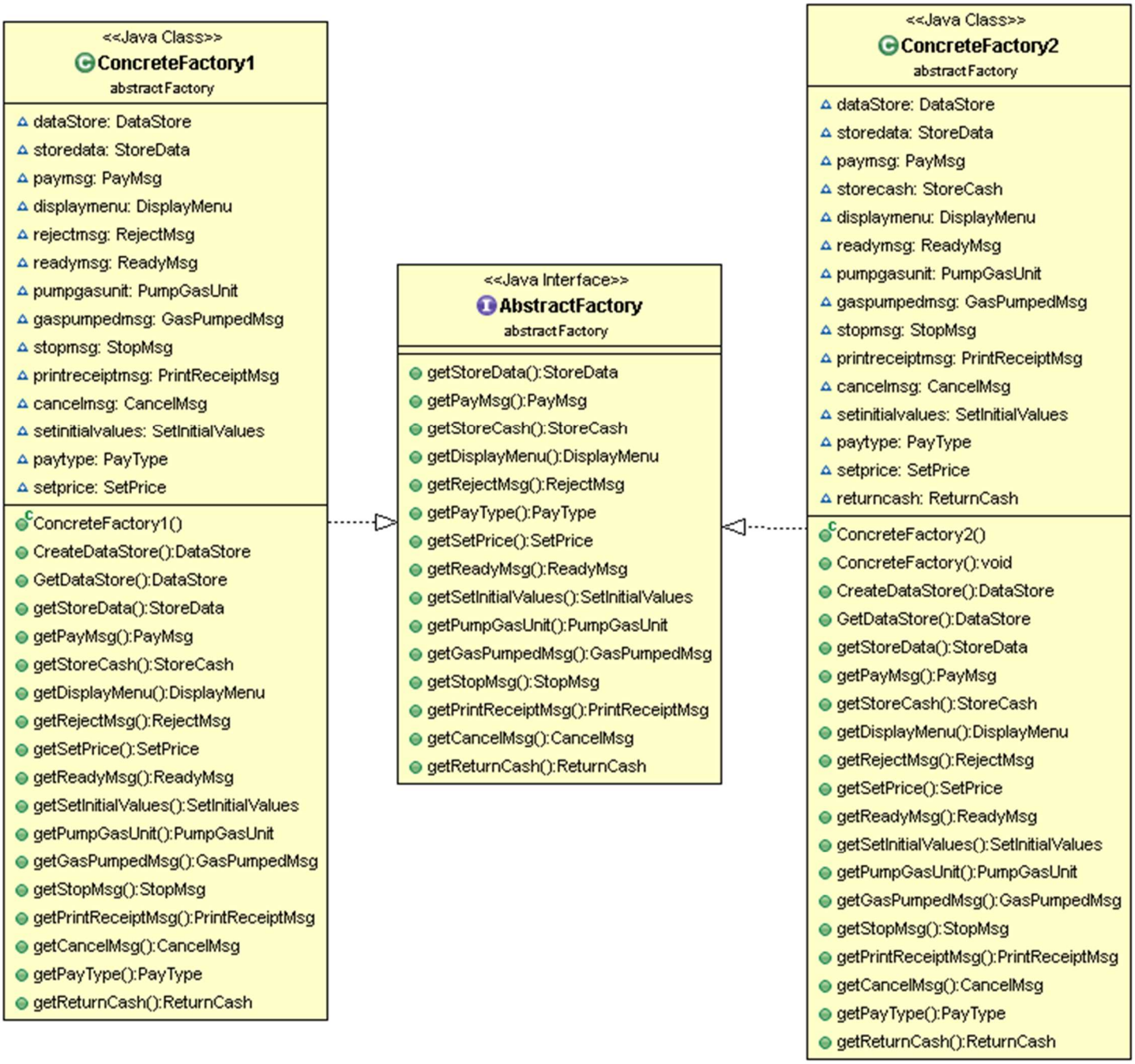



AbstractFactory
GasPump1
DataStore *ds
MDAEFSM *m
AbstractFactory *af
void Activate(int, int)
void Cancel()
void Reject()
void Start()
void PayCredit()
void Approved()
void StartPump()
void PumpGallon(int)
void StopPump()
DataStore
DataStore1
float a
float b
int G
float price
float rPrice
float sPrice
float total
int geta()
void seta(int)
int getb()
void setb(int)
int get_g()
void set_g(int)
float getprice()
void setprice(float)
float get_rprice()
void set_rprice(float)
float get_sprice()
void get_sprice(float)
float get_total()
void set_total(float)
MDAEFSM
State *s
ListOfStates[]
void Activate()
void Start()
void Reject()
void Cancel()
void PayType(int)
void Approved()
void SelectGas(int)
void StartPump()
void Pump()
void StopPump()
void Receipt()
void NoReceipt()
ChangeState(state)
GasPump2
DataStore *ds
MDAEFSM *m
AbstractFactory *af
void Activate(float, float, float)
void Start()
void PayCash(float)
void Cancel()
void StartPump()
void PumpGallon(int)
void Stop()
void Receipt()
void NoReceipt()
DataStore2
int a
int b
int c
int L
float cash
float rPrice
float sPrice
float pPrice
float total
int geta()
void seta(int)
int getb()
void setb(int)
int getc()
void setc(int)
int get_g()
void set_g(int)
float getcash()
void setcash(float)
float get_rprice()
void set_rprice(float)
float get_sprice()
void get_sprice(float)
float get_pprice()
void set_pprice(float)
float get_total()
void set_total(float)
MDAEFSM
AbstractFactory
Output
AbstractFactory
Output
State
MDAEFSM *m
Output *op
void Activate()
void Start()
void Reject()
void Cancel()
void PayType(int)
void Approved()
int SelectGas(int)
void StartPump()
void Pump()
void StopPump()
void Receipt()
void NoReceipt()
Start
id = 0
void Activate()
S0
id = 1
void Start()
S1
id = 2
void PayType(int)
S2
id = 3
void Approved()
void Reject()
S3
id = 4
int SelectGas(int)
void Cancel()
S4
id = 5
void StartPump()
S5
id = 6
void Pump()
void StopPump()
S6
id = 7
void Receipt()
void NoReceipt()

State pattern:



Factory:



Source :

CancelMsg

```
package strategy;

public abstract class CancelMsg {

	public abstract void cancelmsg();

}
```

CancelMsgGP

```
package strategy;

public class CancelMsgGP extends CancelMsg {

	@Override
	public void cancelmsg() {

		System.out.println("\n  !!Transaction Cancelled!! \n");
		System.out.println("\n  Please Start Again: \n");

	}

}
```

DisplayMenu_GP1

```
package strategy;

public class DisplayMenu_GP1 extends DisplayMenu {
	public void DisplayMenu()
	{
		System.out.println("Gas Pump1:: Select Gas: \n" );
		System.out.println("> Regular \n" );
		System.out.println("> Super \n" );
		System.out.println("Press 8 for Regular and 7 for Super \n");
	}

}
```

DisplayMenu_GP2

```
package strategy;

public class DisplayMenu_GP2 extends DisplayMenu {

	@Override
	public void DisplayMenu() {

		System.out.println("Gas Pump2:: Select Gas: \n" );
		System.out.println("> Premium \n" );
		System.out.println("> Regular \n" );
		System.out.println("> Super \n" );

		System.out.println("Press 5 for Premium 6 for Regular and 7 for Super \n");
	}

}
```

DisplayMenu

```
package strategy;

public abstract class DisplayMenu {
	public abstract void DisplayMenu();

}
```

GasPumpuedMsg_GP1

```
package strategy;


import dataStore.DataStore;

import dataStore.DataStore1;
```

```
public class GasPumpedMsg_GP1 extends GasPumpedMsg {

	public void GasPumpedMsg(DataStore dataStore){
		System.out.println(((DataStore1)dataStore).get_G() + " Gallon(s) Pumped"  );
		System.out.println("\nPress 10 to Continue Pumping:\n");
	}

}
```

GasPumpedMsg_GP2

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore1;
import dataStore.DataStore2;

public class GasPumpedMsg_GP2 extends GasPumpedMsg {

	@Override
	public void GasPumpedMsg(DataStore dataStore) {

		System.out.println(((DataStore2)dataStore).get_L() + " Liters(s) Pumped"  );
		System.out.println("\nPress 9 to Continue Pumping:\n");
	}

}
```

GasPumpedMsg

```
package strategy;

import dataStore.DataStore;

public abstract class GasPumpedMsg {

	public abstract void GasPumpedMsg(DataStore dataStore);

}
```

GasPumpedMsg_GP1

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore1;


public class GasPumpedMsg_GP1 extends GasPumpedMsg {
```

```
    public void GasPumpedMsg(DataStore dataStore){
        System.out.println(((DataStore1)dataStore).get_G() + " Gallon(s) Pumped"  );
        System.out.println("\nPress 10 to Continue Pumping:\n");
    }

}
```

Abstract Class PayMsg

```
package strategy;

public abstract class PayMsg {

        public abstract void PayMsg();



}
```

Class PayMsg_GP1

```
package strategy;

public class PayMsg_GP1 extends PayMsg {

        @Override
        public void PayMsg() {

                System.out.println("\n  Pay by CREDIT \n");
                System.out.println("Press 3 to Pay \n");
        }

}
```

Class PayMsg_GP2

```
package strategy;

public class PayMsg_GP2 extends PayMsg {

        @Override
        public void PayMsg() {

                System.out.println("\n  Pay by CASH \n");
                System.out.println("Press 3 to Pay \n");

        }
```

```
}
```

Class PayType

```
package strategy;

public class PayType {

}
```

Class PayTypeGP

```
package strategy;

public class PaytypeGP extends PayType {

}
```

Abstract classPrintReceiptMsg

```
package strategy;

import dataStore.DataStore;

public abstract class PrintReceiptMsg {


	public abstract void PrintReceiptMsg(DataStore dataStore);

}
```

Class PrintReceiptMsg_GP1

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore1;


public class PrintReceiptMsg_GP1 extends PrintReceiptMsg {
	public void PrintReceiptMsg(DataStore dataStore) {

		System.out.println("Gas Pump 1:\n");
		System.out.println("Total Amount: $" + ((DataStore1)dataStore).get_Total() );
		System.out.println("Quantity:"+ (((DataStore1)dataStore).get_G()-1) + " Gallon(s)" );
	}

}
```

Class PrintReceiptMsg_GP2

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore1;
import dataStore.DataStore2;

public class PrintReceiptMsg_GP2 extends PrintReceiptMsg {

	@Override
	public void PrintReceiptMsg(DataStore dataStore) {

      System.out.println("Gas Pump 2:\n");

			System.out.println("\nTotal Amount: $" + ((DataStore2)dataStore).get_Total() );
			System.out.println("Quantity:"+ (((DataStore2)dataStore).get_L()-1) + " Liter(s)" );
	}

}
```

Abstract class PumpGasUnit

```
package strategy;

public abstract class PumpGasUnit {

	public abstract void PumpGasUnit();
}
```

Class PumpGasUnit_GP1

```
package strategy;

public class PumpGasUnit_GP1 extends PumpGasUnit {

	public void PumpGasUnit()
	{
		System.out.println("\nGas Pump 1:: 1 Gallon pumped \n" );
	}

}
```

Class PumpGasUnit_GP2

```
package strategy;

public class PumpGasUnit_GP2 extends PumpGasUnit {
```

```
	@Override
	public void PumpGasUnit() {

		System.out.println("\nGas Pump 2:: 1 Liter is Pumped \n" );

	}

}
```

Class ReadyMsg

```
package strategy;

public abstract class ReadyMsg {

	public abstract void ReadyMsg();

}
```

Class ReadyMsgGP

```
package strategy;

public class ReadyMsgGP extends ReadyMsg {

	public void ReadyMsg(){
		System.out.println("\n  Gas Pump is Ready. ");
		System.out.println("  Start Pumping: \n");
	}

}
```

Abstract class RejectMsg

```
package strategy;

public abstract class RejectMsg {

  public abstract void RejectMsg();
}
```

Class RejectMsgGP

```
package strategy;

public class RejectMsgGP extends RejectMsg {

	@Override
```

```
	public void RejectMsg() {

		System.out.println("\n  !!Credit Card Rejected!! \n");
		System.out.println("\n  Please Start Again: \n");

	}

}
```

Abstract class ReturnCash

```
package strategy;

import dataStore.DataStore;

public abstract class ReturnCash {

	public abstract void ReturnCash(DataStore datastore);

}
```

Class ReturnCashGP

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore1;
import dataStore.DataStore2;

public class ReturnCashGP extends ReturnCash{

	@Override
	public void ReturnCash(DataStore datastore) {

		int cash =((DataStore2)datastore).get_cash();
		int unit_price = ((DataStore2)datastore).get_Price();
		int liters = ((DataStore2)datastore).get_L() - 2;

		int s = cash - (unit_price * liters);

		System.out.println("the amount of cash to be returned :$"+ s);

	}

}
```

Class SetInitialValues

```
package strategy;

import dataStore.DataStore;

public abstract class  SetInitialValues {

	public abstract void SetInitialValues(DataStore dataStore);
}
```

Class SetInitialValues_GP1

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore1;

public class SetInitialValues_GP1 extends SetInitialValues {


	public void SetInitialValues(DataStore dataStore) {

		((DataStore1)dataStore).set_G(0);
		((DataStore1)dataStore).set_Total(0);
	}
}
```

Class SetInitialValues_GP2

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore1;
import dataStore.DataStore2;

public class SetInitialValues_GP2 extends SetInitialValues {

	@Override
	public void SetInitialValues(DataStore dataStore) {

		((DataStore2)dataStore).set_L(0);
		((DataStore2)dataStore).set_Total(0);
	}

}
```

Abstract class SetPrice

```
package strategy;

import dataStore.DataStore;

public  abstract class SetPrice {

	public abstract void SetPrice(DataStore dataStore);
}
```

Class SetPrice_GP1

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore1;

public class SetPrice_GP1 extends SetPrice {


	public void SetPrice(DataStore dataStore)
	{
		System.out.println("\nGas Pump 1:: Gas Price: $" + ((DataStore1)dataStore).get_Price() );

	}
}
```

Class SetPrice_GP2

```
package strategy;

import dataStore.DataStore;

public class SetPrice_GP2 extends SetPrice {

	@Override
	public void SetPrice(DataStore dataStore) {

	}

}
```

Abstract class StopMsg

```
package strategy;

public abstract class StopMsg {
```

```
	public abstract void StopMsg();

}
```

Class StopMsgGP

```
package strategy;

public class StopMsgGP extends StopMsg {

	public void StopMsg(){
		System.out.println("\n  Gas Pump has stopped!! \n");
	}
}
```

Abstract class StoreCash

```
package strategy;

import dataStore.DataStore;

public abstract class StoreCash {

	public abstract void StoreCash(DataStore dataStore);

}
```

Class StoreCash_GP2

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore2;

public class StoreCash_GP2 extends StoreCash {

	@Override

		public void StoreCash(DataStore dataStore)
		{
			((DataStore2)dataStore).set_cash();
			System.out.println("Gas Pump 3:: Cash Stored: $" + ((DataStore2)dataStore).get_cash() );
		}
}
```

Abstract class StoreData

```
package strategy;
```

```
import dataStore.DataStore;

public abstract class StoreData {

	public abstract void StoreData(DataStore dataStore);

}
```

class StoreData_GP1

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore1;

public class StoreData_GP1 extends StoreData {

	public void StoreData(DataStore dataStore)
	{
		((DataStore1)dataStore).set_superprice();
		((DataStore1)dataStore).set_regularprice();
	}
}
```

Class StoreData_GP2

```
package strategy;

import dataStore.DataStore;
import dataStore.DataStore2;

public class StoreData_GP2 extends StoreData {

	@Override
	public void StoreData(DataStore dataStore) {

		((DataStore2)dataStore).set_premium_price();
		((DataStore2)dataStore).set_super_price();
		((DataStore2)dataStore).set_regular_price();

	}

}
```

AbstractFactory

```
package abstractFactory;
```

```
import strategy.*;
 public interface AbstractFactory {

        //abstract factory class, concrete factory 1 & 2 implement these methods

        public StoreData getStoreData();

        public PayMsg getPayMsg();

        public StoreCash getStoreCash();

        public DisplayMenu getDisplayMenu();

        public RejectMsg getRejectMsg();

        public PayType getPayType();

        public SetPrice getSetPrice();

        public ReadyMsg getReadyMsg();

        public SetInitialValues getSetInitialValues();

        public PumpGasUnit getPumpGasUnit();

        public GasPumpedMsg getGasPumpedMsg();

        public StopMsg getStopMsg();

        public PrintReceiptMsg getPrintReceiptMsg();

        public CancelMsg getCancelMsg();

        public ReturnCash getReturnCash();

}
```

ComcreteFactory1

```
package abstractFactory;
import dataStore.DataStore;
import dataStore.DataStore1;
import strategy.CancelMsg;
```

```
import strategy.CancelMsgGP;
import strategy.DisplayMenu;
import strategy.DisplayMenu_GP1;
import strategy.GasPumpedMsg;
import strategy.GasPumpedMsg_GP1;
import strategy.PayMsg;
import strategy.PayMsg_GP1;
import strategy.PayType;
import strategy.PaytypeGP;
import strategy.PrintReceiptMsg;
import strategy.PrintReceiptMsg_GP1;
import strategy.PumpGasUnit;
import strategy.PumpGasUnit_GP1;
import strategy.ReadyMsg;
import strategy.ReadyMsgGP;
import strategy.RejectMsg;
import strategy.RejectMsgGP;
import strategy.ReturnCash;
import strategy.SetInitialValues;
import strategy.SetInitialValues_GP1;
import strategy.SetPrice;
import strategy.SetPrice_GP1;
import strategy.StopMsg;
import strategy.StopMsgGP;
import strategy.StoreCash;
import strategy.StoreData;
import strategy.StoreData_GP1;

public class ConcreteFactory1 implements AbstractFactory {

        //this implements the methods in abstract factory
        DataStore dataStore = new DataStore1();
        StoreData storedata = new StoreData_GP1();
        PayMsg paymsg = new PayMsg_GP1();

        DisplayMenu displaymenu = new DisplayMenu_GP1();
        RejectMsg rejectmsg = new RejectMsgGP();
        ReadyMsg readymsg = new ReadyMsgGP();
        PumpGasUnit pumpgasunit = new PumpGasUnit_GP1();
        GasPumpedMsg gaspumpedmsg = new GasPumpedMsg_GP1();
        StopMsg stopmsg = new StopMsgGP();
```

```
PrintReceiptMsg printreceiptmsg = new PrintReceiptMsg_GP1();
CancelMsg cancelmsg = new CancelMsgGP();
SetInitialValues setinitialvalues = new SetInitialValues_GP1();
PayType paytype = new PaytypeGP();
SetPrice setprice = new SetPrice_GP1();

public DataStore CreateDataStore() {
        return (this.dataStore);
}

public DataStore GetDataStore() {
        return this.dataStore;
}

@Override
public StoreData getStoreData() {
        return this.storedata;
}

@Override
public PayMsg getPayMsg() {
        return this.paymsg;
}

@Override
public StoreCash getStoreCash() {
        return null;
}

@Override
public DisplayMenu getDisplayMenu() {
        return this.displaymenu;
}

@Override
public RejectMsg getRejectMsg() {
        return this.rejectmsg;
}

@Override
public SetPrice getSetPrice() {
```

```
        return this.setprice;
}

@Override
public ReadyMsg getReadyMsg() {
        return this.readymsg;
}

@Override
public SetInitialValues getSetInitialValues() {
        return this.setinitialvalues;
}

@Override
public PumpGasUnit getPumpGasUnit() {
        return this.pumpgasunit;
}

@Override
public GasPumpedMsg getGasPumpedMsg() {
        return this.gaspumpedmsg;
}

@Override
public StopMsg getStopMsg() {
        return this.stopmsg;
}

@Override
public PrintReceiptMsg getPrintReceiptMsg() {
        return this.printreceiptmsg;
}

@Override
public CancelMsg getCancelMsg() {
        return this.cancelmsg;
}

@Override
public PayType getPayType() {
        return null;
```

```
	}

	@Override
	public ReturnCash getReturnCash() {
		return null;
	}

}
```

ConcreteFactory2

```
package abstractFactory;

import dataStore.DataStore;
import dataStore.DataStore1;
import dataStore.DataStore2;
import strategy.StoreCash_GP2;
import strategy.CancelMsg;
import strategy.CancelMsgGP;
import strategy.DisplayMenu;
import strategy.DisplayMenu_GP1;
import strategy.DisplayMenu_GP2;
import strategy.GasPumpedMsg;
import strategy.GasPumpedMsg_GP1;
import strategy.GasPumpedMsg_GP2;
import strategy.PayMsg;
import strategy.PayMsg_GP1;
import strategy.PayMsg_GP2;
import strategy.PayType;
import strategy.PaytypeGP;
import strategy.PrintReceiptMsg;
import strategy.PrintReceiptMsg_GP1;
import strategy.PrintReceiptMsg_GP2;
import strategy.PumpGasUnit;
import strategy.PumpGasUnit_GP1;
import strategy.PumpGasUnit_GP2;
import strategy.ReadyMsg;
import strategy.ReadyMsgGP;
import strategy.RejectMsg;
import strategy.RejectMsgGP;
import strategy.ReturnCash;
import strategy.ReturnCashGP;
import strategy.SetInitialValues;
```

```
import strategy.SetInitialValues_GP1;
import strategy.SetInitialValues_GP2;
import strategy.SetPrice;
import strategy.SetPrice_GP1;
import strategy.SetPrice_GP2;
import strategy.StopMsg;
import strategy.StopMsgGP;
import strategy.StoreCash;
import strategy.StoreData;
import strategy.StoreData_GP1;
import strategy.StoreData_GP2;

public class ConcreteFactory2 implements AbstractFactory{

	DataStore dataStore = null;
	StoreData storedata = null;
	PayMsg paymsg = null;
	StoreCash storecash = null;
	DisplayMenu displaymenu = null;
	ReadyMsg readymsg = null;
	PumpGasUnit pumpgasunit = null;
	GasPumpedMsg gaspumpedmsg = null;
	StopMsg stopmsg = null;
	PrintReceiptMsg printreceiptmsg = null;
	CancelMsg cancelmsg = null;
	SetInitialValues setinitialvalues  = null;
	PayType paytype = null;
	SetPrice setprice = null;
	ReturnCash returncash = null;

	public ConcreteFactory2(){
		dataStore = new DataStore2();
		storedata= new StoreData_GP2();
		paymsg = new PayMsg_GP2();
		storecash = new StoreCash_GP2();
		displaymenu = new DisplayMenu_GP2();
		readymsg = new ReadyMsgGP();
		pumpgasunit = new PumpGasUnit_GP2();
		gaspumpedmsg = new GasPumpedMsg_GP2();
		stopmsg = new StopMsgGP();
		printreceiptmsg = new PrintReceiptMsg_GP2();
```

```
            cancelmsg = new CancelMsgGP();
            setinitialvalues = new SetInitialValues_GP2();
            paytype = new PaytypeGP();
            setprice = new SetPrice_GP2();
            returncash = new ReturnCashGP();
}
public void ConcreteFactory() {

}

public DataStore CreateDataStore() {
            return (dataStore);
}

public DataStore GetDataStore() {
            return dataStore;
}

@Override
public StoreData getStoreData() {

            return storedata;
}

@Override
public PayMsg getPayMsg() {

            return paymsg;
}

@Override
public StoreCash getStoreCash() {

            return storecash;
}

@Override
public DisplayMenu getDisplayMenu() {

            return displaymenu;
}
```

```
@Override
public RejectMsg getRejectMsg() {

        return null;
}

@Override
public SetPrice getSetPrice() {

        return setprice;
}

@Override
public ReadyMsg getReadyMsg() {

        return readymsg;
}

@Override
public SetInitialValues getSetInitialValues() {

        return setinitialvalues;
}

@Override
public PumpGasUnit getPumpGasUnit() {

        return pumpgasunit;
}

@Override
public GasPumpedMsg getGasPumpedMsg() {

        return gaspumpedmsg;
}

@Override
public StopMsg getStopMsg() {

        return stopmsg;
```

```
        }

        @Override
        public PrintReceiptMsg getPrintReceiptMsg() {

                return printreceiptmsg;
        }

        @Override
        public CancelMsg getCancelMsg() {

                return cancelmsg;
        }

        @Override
        public PayType getPayType() {

                return null;
        }

        @Override
        public ReturnCash getReturnCash() {

                return returncash;
        }

}
```

Abstract Class DataStore

```
package dataStore;

//datastore Abstract class , will be inherited in datastore1 and 2

public abstract class DataStore {

}
```

Class DataStore1

```
package dataStore;

public class DataStore1 extends DataStore {

        //those accessed from the outside are public
```

```
int W; // Flag for Cash and Credit Payment Options
int G; // Number of Gallons Needed
float price; // Price of gas
float totalPrice; // Total amount
float sprice;// Variable for keeping super gas price
float rprice;// Variable for keeping regular gas price

int g; // Quantity in gallons temp storage
float total; // Total amount temp storage
int w; // Temp flag for Cash and Credit

public float temp_a; // Temp storage for super gas unit price
public float temp_b; // Temporary Variable for regular gas unit price

public float set_superprice()
{
	return this.sprice = this.temp_a;
}

public float set_regularprice()
{
	return this.rprice = this.temp_b;
}

public void set_W() {
	this.W = this.w;

}

public int get_W() {
	return this.W;
}



public void set_G(int g) {
	this.G = this.g;
}

public int get_G() {
	return this.G = this.G + 1;
```

```
	}

	public float get_Price() {
		return this.price;
	}

	public void set_sprice() {

		this.price = this.sprice;

	}

	public void set_rprice() {

		this.price = this.rprice;

	}
	public void set_Total(int total) {

		this.totalPrice = this.total;

	}

	public float get_Total() {
		// The total amount is  price * Quantity in Gallons)
		return this.total = this.price * this.G;
	}

}
```

Class DataStore2

```
package dataStore;

public class DataStore2 extends DataStore {

	//those accessed from the outside are public

	public int temp_a;
	public int temp_b;
	public int temp_c;

	int pprice;
```

```
int sprice;
int rprice;
public int Price;
public int total;
int Total;
int l;
public int L;
int w;
int W;
public int temp_cash;
public int cash;

public int set_premium_price()
{
	return this.pprice = this.temp_a;
}

public int set_super_price()
{
	return this.sprice = this.temp_b;
}

public int set_regular_price()
{
	return  rprice = this.temp_c;
}

public void set_pprice()
{
	this.Price = this.pprice;
}

public void set_sprice()
{
	this.Price = this.sprice;
}

public void set_rprice()
{
	this.Price = this.rprice;
}
```

```
public void set_Total(int total) {

		this.Total = this.total;

}

public int get_Total() {
		// The total amount is price * Quantity in Liters
		return this.total = this.Price * this.L;
}

public int set_L(int l) {
		return this.L = this.l;
}

public int get_L() {

		return this.L = this.L + 1;

}

public void set_W() {
		this.W = this.w;

}

public int get_W() {
		return this.W;
}

public int get_Price() {

		return this.Price;
}

public int set_cash() {

		return this.cash = this.temp_cash;
}
```

```
	public int get_cash() {

		return this.cash;
	}
}
```

GasPump1

```
package gaspump;

import dataStore.DataStore;
import dataStore.DataStore1;

import state.MDAEFSM;

public class GasPump1 {


		MDAEFSM mdaefsmObj = null; // Pointer to MDA-EFSM object

		DataStore dataStore = null; // Pointer to DataStore

		public GasPump1(MDAEFSM mdaefsmObj, DataStore dataStore) {

			//constructor for gaspump
			this.mdaefsmObj = mdaefsmObj;
			this.dataStore = dataStore;
		}

		public void Activate(float superprice, float regularprice) {

			//check the prices and activate the gas pump
			if (superprice > 0 && regularprice > 0)
				((DataStore1) dataStore).temp_a = superprice;
			  ((DataStore1) dataStore).temp_b = regularprice;

			mdaefsmObj.Activate();
		}


		public void Start() {
```

```
        //start the gas pump
        mdaefsmObj.Start();
}

public void PayCredit() {

        //set the payment type
        mdaefsmObj.PayType(1);
        System.out.println("Received card details \n 4 -> Reject , 5-> Cancel , 6-> Approved\n");
}

public void Reject() {

        //reject the payment
        mdaefsmObj.Reject();

}

public void Cancel() {

        //cancel the operation
        mdaefsmObj.Cancel();

}

public void Approved() {

        //approve the payment
        mdaefsmObj.Approved();

}

public void Regular() {

        //select the regular gas type and set variable
        ((DataStore1) dataStore).set_rprice();
        mdaefsmObj.SelectGas(1);

        System.out.println("Regular Gas selected\n");

}
```

```
        public void Super() {

                //select the super gas type and set variable
                ((DataStore1) dataStore).set_sprice();
                mdaefsmObj.SelectGas(2);

                System.out.println("Super Gas selected\n");

        }

        public void StartPump() {

                //start the pump
                mdaefsmObj.StartPump();
        }

        public void Pump() {

                //pump one gallon of gas
                mdaefsmObj.Pump();
        }

        public void StopPump() {

                //stop the pump and generate receipt
                mdaefsmObj.StopPump();
                mdaefsmObj.Receipt();

        }



    }
```

GasPumP2

```
package gaspump;

import dataStore.DataStore;
import dataStore.DataStore2;
import state.MDAEFSM;
```

```
public class GasPump2 {

	MDAEFSM mdaefsmObj = null; // Pointer to MDA-EFSM
	DataStore dataStore = null; // Pointer to DataStore

	public GasPump2(MDAEFSM efsm, DataStore dataStore) {

		//constructor to set the MDA-EFSM and DataStore object
		this.mdaefsmObj = efsm;
		this.dataStore = dataStore;
	}

	public void Activate(int premiumprice, int superprice, int regularprice) {

		//check prices and then activate
		if (premiumprice >0 && superprice > 0 && regularprice >0 ){
			((DataStore2) dataStore).temp_a = premiumprice;
			((DataStore2) dataStore).temp_b = superprice;
			((DataStore2) dataStore).temp_c = regularprice;
			mdaefsmObj.Activate();
		}

	}

	public void Start() {

		//start the gas pump
		mdaefsmObj.Start();
	}

	public void PayCash(int cash) {

		//check cash and set the payment type
		if (cash > 0){
			((DataStore2) dataStore).temp_cash = cash;
			mdaefsmObj.PayType(2);
		}

	}
```

```
public void Cancel() {

        //cancel the operation
        mdaefsmObj.Cancel();

}

public void Regular() {

        //select the regular gas type
        ((DataStore2) dataStore).set_rprice();
        mdaefsmObj.SelectGas(1);

        System.out.println("Regular Gas selected\n");

}

public void Premium() {

        //select the premium gas type
        ((DataStore2) dataStore).set_pprice();
        mdaefsmObj.SelectGas(2);

        System.out.println("Premium Gas selected\n");

}

public void Super() {

        //select the super gas type
        ((DataStore2) dataStore).set_sprice();
        mdaefsmObj.SelectGas(3);

        System.out.println("Super Gas selected\n");

}

public void StartPump() {

        //start the pump
        mdaefsmObj.StartPump();
```

```
	}

	public void Pump() {

		//pump one liter of gas
		if (((DataStore2) dataStore).cash < (((DataStore2) dataStore).L + 1)
					* ((DataStore2) dataStore).Price) {
			mdaefsmObj.StopPump();
			System.out.println("Pump Stopped, Insufficient Cash\n");
		}
		else
			mdaefsmObj.Pump();

	}

	public void StopPump() {

		//stop the gas pumping
		mdaefsmObj.StopPump();
	}

	public void Receipt() {

		//generate receipt
		mdaefsmObj.Receipt();
	}

	public void NoReceipt() {

		//proceed without receipt
		mdaefsmObj.NoReceipt();
	}
}
```

MainDriver:

```
package mainDriver;
import java.util.Scanner;

import abstractFactory.ConcreteFactory1;
import abstractFactory.ConcreteFactory2;
import gaspump.GasPump1;
```

```
import gaspump.GasPump2;
import output.Output;
import state.MDAEFSM;
import state.state;

/*
 * @author Chris Mathew Dani
 * A20372828
 *
 */
public class Driver {

    public static int gasPumpType = 0;

    public static void main(String[] args) throws InterruptedException {
        System.out.println("Gas pump menu: (Enter the item #)");
        System.out.println("1. Gas Pump 1");
        System.out.println("2. Gas Pump 2");

        Scanner inpScan = new Scanner(System.in);
        int menuItem = inpScan.nextInt();
        if (menuItem == 1){

            gasPumpType= 1;
            ConcreteFactory1 factory = new ConcreteFactory1();
            MDAEFSM mdaefsmObj = new MDAEFSM();
            state.outputobject = new Output(factory, factory.GetDataStore());

            GasPump1 gasPump1 = new GasPump1(mdaefsmObj, factory.GetDataStore());

            System.out.println("Gas pump 1 functionality:");

            while(true){

                System.out.println("  Select the Operation #: ");
                System.out.println("Operations for Gas Pump 1");
                System.out.println("1 . Activate(float,float)");
                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(" Input: ");
                int menuItem1 = inpScan.nextInt();
```

```
if (menuItem1 == 12)
    break;

if (menuItem1 == 1){

    //ACTIVATE
    System.out.println("\n  Operation:  Activate(float a,float b)");
    System.out.println("   Enter the Unit Price for the Super Gas");
    float superprice  =  inpScan.nextFloat();
    System.out.println("   Enter the Unit  Price for the Regular Gas");
    float regularprice = inpScan.nextFloat();
    gasPump1.Activate(superprice,regularprice);

}
else if (menuItem1 == 2){

    System.out.println("  Operation:  Start()");
    gasPump1.Start();
}
else if (menuItem1 == 3){

    //PayCredit
    System.out.println("  Operation:  PayCredit()");
    gasPump1.PayCredit();
}
else if (menuItem1 == 4)
{
    //Reject;
    System.out.println("Operations  Reject()");
    gasPump1.Reject();

}
else if (menuItem1 ==5){
    //cancel
    System.out.println("Operations  Cancel()");
    gasPump1.Cancel();

}
else if (menuItem1 == 6){
    //approve
    System.out.println("Operations Approved()");
    gasPump1.Approved();
}
else if (menuItem1 == 7){
    //select regular gas
    System.out.println("  Operation:  Regular()");
    gasPump1.Regular();
}
else if (menuItem1 == 8){
    //selected super gas
```

```
        System.out.println("  Operation:  Super()");
        gasPump1.Super();
    }
    else if (menuItem1 == 9){
        //start pump
        System.out.println("  Operation:  StartPump()");
        gasPump1.StartPump();
    }
    else if (menuItem1 == 10){
        //pump gas
        System.out.println("  Operation:  Pump()");
        gasPump1.Pump();
    }
    else if (menuItem1 == 11){
        //stop the pump
        System.out.println("  Operation:  StopPump()");
        gasPump1.StopPump();
    }
}
}
else if (menuItem ==2)
{
    gasPumpType= 2;
    ConcreteFactory2 factory = new ConcreteFactory2();
    MDAEFSM mdaefsmObj = new MDAEFSM();
    state.outputobject = new Output(factory, factory.GetDataStore());

    GasPump2 gasPump2 = new GasPump2(mdaefsmObj, factory.GetDataStore());

    System.out.println("Gas pump 2");
    System.out.println("Operations for Gas Pump 2");

    while(true){
        System.out.println("  Select Operation: ");
        System.out.println("1 . Activate(int, int ,int)");
        System.out.println("2 . Start() ");
        System.out.println("3 . PayCash(int)");
        System.out.println("4 . Cancel()");
        System.out.println("5 . Premium()");
        System.out.println("6 . Regular()");
        System.out.println("7 . Super()  ");
        System.out.println("8 . StartPump()");
        System.out.println("9 . PumpLiter()");
        System.out.println("10. Stop()");
        System.out.println("11. Receipt()");
        System.out.println("12. NoReceipt()");

        int menuItem1 = inpScan.nextInt();
```

```
if (menuItem1 == 1){

    //ACTIVATE
    System.out.println("\n  Operation:  Activate(int a,int b,int c)");
    System.out.println("   Enter the Unit  Price for Premium Gas");
    int premiumprice  =  inpScan.nextInt();
    System.out.println("   Enter the Unit  Price for Super Gas");
    int superprice  =  inpScan.nextInt();
    System.out.println("   Enter the Unit  Price for Regular Gas");
    int regularprice = inpScan.nextInt();
    gasPump2.Activate(premiumprice,superprice,regularprice);

}
else if (menuItem1 == 2){

    System.out.println("  Operation:  Start()");
    gasPump2.Start();
}

else if (menuItem1 == 3){

    //PayCash
    System.out.println("  Operation:  PayCash()");
    System.out.println("Enter the amount of cash :");
    int cash = inpScan.nextInt();
    gasPump2.PayCash(cash);
}

else if (menuItem1 ==4){
    System.out.println("Operations  Cancel()");
    gasPump2.Cancel();
    //cancel
}
else if (menuItem1 == 5){
    //select regular gas
    System.out.println("  Operation:  Premium()");
    gasPump2.Premium();
}

else if (menuItem1 == 6){
    //select regular gas
    System.out.println("  Operation:  Regular()");
    gasPump2.Regular();
}
else if (menuItem1 == 7){
    //selected super gas
    System.out.println("  Operation:  Super()");
    gasPump2.Super();
}
else if (menuItem1 == 8){
```

```
                //start pump
                System.out.println("  Operation:  StartPump()");
                gasPump2.StartPump();
            }
            else if (menuItem1 == 9){
                //to pump gas
                System.out.println("  Operation:  Pump()");
                gasPump2.Pump();
            }
            else if (menuItem1 == 10){

                //stop pumping
                System.out.println("  Operation:  StopPump()");
                gasPump2.StopPump();
            }
            else if (menuItem1 == 11){

                //generate and print receipt
                System.out.println("  Operation:  Receipt()");
                gasPump2.Receipt();;
            }
            else if (menuItem1 == 12){

                //don't take receipt
                System.out.println("  Operation:  NoReceipt()");
                gasPump2.NoReceipt();
            }

        }
        }
        else{

            //retry choosing gas pump
            System.out.println("Invalid option, try again ");
        }

        inpScan.close();
    }

    }
```

Output:

```
package output;

import abstractFactory.AbstractFactory;
import dataStore.DataStore;
import strategy.CancelMsg;
import strategy.DisplayMenu;
import strategy.GasPumpedMsg;
import strategy.PayMsg;
```

```
import strategy.PrintReceiptMsg;
import strategy.PumpGasUnit;
import strategy.ReadyMsg;
import strategy.RejectMsg;
import strategy.SetInitialValues;
import strategy.SetPrice;
import strategy.StopMsg;
import strategy.StoreCash;
import strategy.StoreData;
import strategy.ReturnCash;

public class Output {

    AbstractFactory factoryInstance = null;
    DataStore dataStore = null;


    public Output(AbstractFactory factoryInstance, DataStore dataStore) {

        //Constructor to initialize the output object
        this.factoryInstance = factoryInstance;
        this.dataStore = dataStore;
    }

    public void StoreData() {

        //stored data process
        System.out.println("\nGas Pump activated ");
        StoreData storedata = factoryInstance.getStoreData();
        storedata.StoreData(dataStore);
    }

    public void PayMsg() {

        //payment message
        PayMsg paymsg = factoryInstance.getPayMsg();
        paymsg.PayMsg();
    }

    public void StoreCash() {
        //store cash
        StoreCash storecash = factoryInstance.getStoreCash();
        storecash.StoreCash(dataStore);

    }

    public void DisplayMenu() {
        //display the menu
        DisplayMenu displaymenu = factoryInstance.getDisplayMenu();
        displaymenu.DisplayMenu();
```

```
}

public void RejectMsg() {
    //rejection message
    RejectMsg rejectmsg = factoryInstance.getRejectMsg();
    rejectmsg.RejectMsg();
}

public void CancelMsg() {

    //cancellation message
    CancelMsg cancelmsg = factoryInstance.getCancelMsg();
    cancelmsg.cancelmsg();


}

public void SetPrice() {
    //set the price
    SetPrice setprice = factoryInstance.getSetPrice();
    setprice.SetPrice(dataStore);

}

public void SetInitialValues() {

    //setting initial values
    SetInitialValues setinitialvalues = factoryInstance.getSetInitialValues();
    setinitialvalues.SetInitialValues(dataStore);

}

public void ReadyMsg() {
    //ready message
    ReadyMsg readymsg = factoryInstance.getReadyMsg();
    readymsg.ReadyMsg();

}

public void PumpGasUnit() {
    //pump gas
    PumpGasUnit pumpgasunit = factoryInstance.getPumpGasUnit();
    pumpgasunit.PumpGasUnit();

}

public void GasPumpedMsg() {

    //pumped message
```

```
        GasPumpedMsg gaspumpedmsg = factoryInstance.getGasPumpedMsg();
        gaspumpedmsg.GasPumpedMsg(dataStore);

    }

    public void StopMsg() {

        //message to stop operation
        StopMsg stopmsg = factoryInstance.getStopMsg();
        stopmsg.StopMsg();

    }

    public void PrintReceiptMsg() {

        //printing the receipt
        System.out.println("\nReceipt: ");
        PrintReceiptMsg printreceiptmsg = factoryInstance.getPrintReceiptMsg();
        printreceiptmsg.PrintReceiptMsg(dataStore);

    }


    public void ReturnCash(){

        //return cash
        ReturnCash returncash = factoryInstance.getReturnCash();
        returncash.ReturnCash(dataStore);

    }

}
```

Class MDAEFSM

```
package state;

import abstractFactory.AbstractFactory;
import dataStore.DataStore;
import output.Output;

public class MDAEFSM {


        static state mdaefsmState = null;
        static state S,S0,S1,S2,S3,S4,S5,S6;

        AbstractFactory factory = null;

        public MDAEFSM() {
```

```
            S = new S(mdaefsmState);
            S0 = new S0(mdaefsmState);
            S1 = new S1(mdaefsmState);
            S2 = new S2(mdaefsmState);
            S3 = new S3(mdaefsmState);
            S4 = new S4(mdaefsmState);
            S5 = new S5(mdaefsmState);
            S6 = new S6(mdaefsmState);
            mdaefsmState = S;

    }

    public void Activate() {

            mdaefsmState.Activate();
            printCurrentState();
    }

    public void Start() {

            mdaefsmState.Start();
            printCurrentState();
    }

public void PayType(int t ) {

       if(t==1){
                   mdaefsmState.PayCredit();
                   printCurrentState();
       }else{
                   mdaefsmState.PayCash();
                   printCurrentState();
            }
    }


    public void Reject() {

            mdaefsmState.Reject();
            printCurrentState();

    }

    public void Cancel() {

            mdaefsmState.Cancel();
            printCurrentState();

    }
```

```
public void Approved() {

	mdaefsmState.Approved();
	printCurrentState();
}

public void StartPump() {

	mdaefsmState.StartPump();
	printCurrentState();
}

public void Pump() {

	mdaefsmState.Pump();
	printCurrentState();
}

public void StopPump() {

	mdaefsmState.StopPump();
	printCurrentState();
}

public void SelectGas(int g) {

	mdaefsmState.SelectGas(g);
	printCurrentState();
}

public void Receipt() {

	mdaefsmState.Receipt();
	printCurrentState();
}

public void NoReceipt() {

	mdaefsmState.NoReceipt();
	printCurrentState();
}

public void setState(state mdaefsmState) {
	this.mdaefsmState = mdaefsmState;
}

public state getMachineState() {
	return mdaefsmState;
}
```

```
	//return state objects
	public static state getS0State() {
		return S0;
	}

	public static state getS1State() {
		return S1;
	}

	public static state getS2State() {
		return S2;
	}

	public static state getS3State() {
		return S3;
	}

	public static state getS4State() {
		return S4;
	}

	public static state getS5State() {
		return S5;
	}

	public static state getS6State() {
		return S6;
	}

	public void printCurrentState() {
		System.out.println("\n" + "Current State : "
				+ mdaefsmState.getClass().getName() + "\n");
	}

	public void ReturnCash() {

		//customer return cash
		mdaefsmState.ReturnCash();
		printCurrentState();

	}


}
```

Class S

```
package state;

import output.Output;
```

```
/**
 * State class performing activate function
 */
public class S extends state {

	public S(state s) {
		super(s);

	}

	@Override
	public void Activate() {

		outputobject.StoreData();
		MDAEFSM.mdaefsmState = MDAEFSM.getS0State();
	}

	@Override
	public void Start() {


	}

	@Override
	public void PayCredit() {


	}

	@Override
	public void PayCash() {


	}

	@Override
	public void Reject() {


	}

	@Override
	public void Cancel() {


	}

	@Override
	public void Approved() {
```

```
        }

        @Override
        public void SelectGas(int g) {


        }

        @Override
        public void StartPump() {


        }

        @Override
        public void Pump() {


        }

        @Override
        public void StopPump() {


        }

        @Override
        public void Receipt() {


        }

        @Override
        public void NoReceipt() {


        }

        @Override
        public void ReturnCash() {


        }

}
```

Class S0

```
package state;

public class S0 extends state {

	state s = null;

	public S0(state s) {
		super(s);
	}


	public void Activate() {


	}


	public void Start() {

		outputobject.PayMsg();
		MDAEFSM.mdaefsmState = (MDAEFSM.getS1State());
	}


	public void PayCredit() {


	}


	public void PayCash() {


	}


	public void Reject() {


	}


	public void Cancel() {


	}


	public void Approved() {
```

```
        }

        public void SelectGas(int g) {

        }

        public void StartPump() {

        }

        public void Pump() {

        }

        public void StopPump() {

        }

        public void Receipt() {

        }

        public void NoReceipt() {

        }

        @Override
        public void ReturnCash() {

        }

}
```

Class S1

```
package state;
```

```
public class S1 extends state {

	MDAEFSM state = null;

	public S1(state s) {
		super(s);

	}


	public void Activate() {


	}


	public void Start() {


	}


	public void PayCredit() {

		MDAEFSM.mdaefsmState = (state.getS2State());

	}


	public void PayCash() {

		outputobject.StoreCash();
		outputobject.DisplayMenu();
		MDAEFSM.mdaefsmState = (state.getS3State());

	}


	public void Reject() {


	}


	public void Cancel() {


	}
```

```
        public void Approved() {

        }

        public void SelectGas(int g) {

        }

        public void StartPump() {

        }

        public void Pump() {

        }

        public void StopPump() {

        }

        public void Receipt() {

        }

        public void NoReceipt() {

        }

        @Override
        public void ReturnCash() {

        }

}
```

Class S2

```
package state;

public class S2 extends state {
	public S2(state s) {
		super(s);
	}

	MDAEFSM state = null;

	@Override
	public void Activate() {


	}

	@Override
	public void Start() {


	}

	@Override
	public void PayCredit() {


	}

	@Override
	public void PayCash() {


	}

	@Override
	public void Reject() {

		outputobject.RejectMsg();
		MDAEFSM.mdaefsmState = (state.getS0State());

	}

	@Override
	public void Cancel() {


	}

	@Override
```

```
	public void Approved() {
		outputobject.DisplayMenu();
		MDAEFSM.mdaefsmState = (state.getS3State());

	}

	@Override
	public void SelectGas(int g) {


	}

	@Override
	public void StartPump() {


	}

	@Override
	public void Pump() {


	}

	@Override
	public void StopPump() {


	}

	@Override
	public void Receipt() {


	}

	@Override
	public void NoReceipt() {


	}

	@Override
	public void ReturnCash() {


	}
}
```

Class S3

```
package state;
import mainDriver.Driver;

public class S3 extends state {


	public S3(state s) {
		super(s);

	}

	@Override
	public void Activate() {


	}

	@Override
	public void Start() {


	}

	@Override
	public void PayCredit() {


	}

	@Override
	public void PayCash() {


	}

	@Override
	public void Reject() {


	}

	@Override
	public void Cancel() {

		if (Driver.gasPumpType == 1){
		outputobject.CancelMsg();
		MDAEFSM.mdaefsmState = (MDAEFSM.getS0State());
		}
		else if(Driver.gasPumpType == 2){
			outputobject.CancelMsg();
```

```
                outputobject.ReturnCash();
                MDAEFSM.mdaefsmState = (MDAEFSM.getS0State());
        }

}

@Override
public void Approved() {


}

@Override
public void SelectGas(int g) {

        outputobject.SetPrice();
        MDAEFSM.mdaefsmState = (MDAEFSM.getS4State());

}

@Override
public void StartPump() {


}

@Override
public void Pump() {


}

@Override
public void StopPump() {


}

@Override
public void Receipt() {


}

@Override
public void NoReceipt() {


}
```

```
        @Override
        public void ReturnCash() {


        }




}
```

Class S4

```
package state;

public class S4 extends state {

        public S4(state s) {
                super(s);
                //this.state = state;
        }

        MDAEFSM state = null;

        @Override
        public void Activate() {


        }

        @Override
        public void Start() {


        }

        @Override
        public void PayCredit() {


        }

        @Override
        public void PayCash() {


        }

        @Override
        public void Reject() {
```

```
}

@Override
public void Cancel() {


}

@Override
public void Approved() {


}

@Override
public void SelectGas(int g) {


}

@Override
public void StartPump() {

        outputobject.SetInitialValues();
        outputobject.ReadyMsg();
        MDAEFSM.mdaefsmState = (state.getS5State());

}

@Override
public void Pump() {


}

@Override
public void StopPump() {


}

@Override
public void Receipt() {


}

@Override
```

```
	public void NoReceipt() {


	}

	@Override
	public void ReturnCash() {


	}

}
```

Class S5

```
package state;

public class S5 extends state {

	public S5(state s) {
		super(s);
		//this.state = state;
	}

	MDAEFSM state = null;

	@Override
	public void Activate() {


	}

	@Override
	public void Start() {


	}

	@Override
	public void PayCredit() {


	}

	@Override
	public void PayCash() {


	}
```

```
@Override
public void Reject() {


}

@Override
public void Cancel() {


}

@Override
public void Approved() {


}

@Override
public void SelectGas(int g) {


}

@Override
public void StartPump() {


}

@Override
public void Pump() {

        outputobject.PumpGasUnit();
        outputobject.GasPumpedMsg();
        MDAEFSM.mdaefsmState = (state.getS5State());

}

@Override
public void StopPump() {

        outputobject.StopMsg();
        MDAEFSM.mdaefsmState = (state.getS6State());

}

@Override
public void Receipt() {
```

```
	}

	@Override
	public void NoReceipt() {


	}

	@Override
	public void ReturnCash() {


	}

}
```

Class S6

```
package state;

import mainDriver.Driver;

public class S6 extends state {

	public S6(state s) {
		super(s);
	}

	MDAEFSM state = null;


	@Override
	public void Activate() {


	}

	@Override
	public void Start() {


	}

	@Override
	public void PayCredit() {


	}

	@Override
```

```
public void PayCash() {


}

@Override
public void Reject() {


}

@Override
public void Cancel() {


}

@Override
public void Approved() {


}

@Override
public void SelectGas(int g) {


}

@Override
public void StartPump() {


}

@Override
public void Pump() {


}

@Override
public void StopPump() {

}



@Override
public void Receipt() {
```

```
		if(Driver.gasPumpType == 1){
			outputobject.PrintReceiptMsg();
		}
		else if (Driver.gasPumpType ==2){
			outputobject.PrintReceiptMsg();
			outputobject.ReturnCash();
		}
		MDAEFSM.mdaefsmState = (state.getS0State());
	}

	@Override
	public void NoReceipt() {

		outputobject.ReturnCash();
		MDAEFSM.mdaefsmState = (state.getS0State());

	}

	@Override
	public void ReturnCash() {


	}

}
```

Abstract Class State

```
package state;
import output.Output;
public abstract class state  {

	public static Output outputobject;
	state currentstate;
	public state( state s)
	{
		currentstate = s;
	}


	public abstract void Activate();

	public abstract void Start();

	public abstract void PayCredit();

	public abstract void PayCash();

	public abstract void Reject();
```

```
        public abstract void Cancel();

        public abstract void Approved();

        public abstract void SelectGas(int g);

        public abstract void StartPump();

        public abstract void Pump();

        public abstract void StopPump();

        public abstract void Receipt();

        public abstract void NoReceipt();

        public void setState(state s) {

                currentstate = s;

        }

        public abstract void ReturnCash();


}
```