Gas-Pump components using a Model-Driven Architecture (MDA)

CS-586: Software System Architecture

Project Report

Chintan Patel

A20329245

Model Driven Architecture of the GasPump Components

MDA-EFSM Events:

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

Operations of the Input Processor (GasPump-1)

```
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();
}
```

Notice:
cash: contains the value of cash deposited
price: contains the price of the selected gas
G: contains the number of gallons already pumped
w: cash/credit flag
cash ,
G, price, w are in the data store
m: is a pointer to the MDA-EFSM object
d: is a pointer to the Data Store object

Operations of the Input Processor (GasPump-2)

```
Activate(float a, float b) {
        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)
}
```

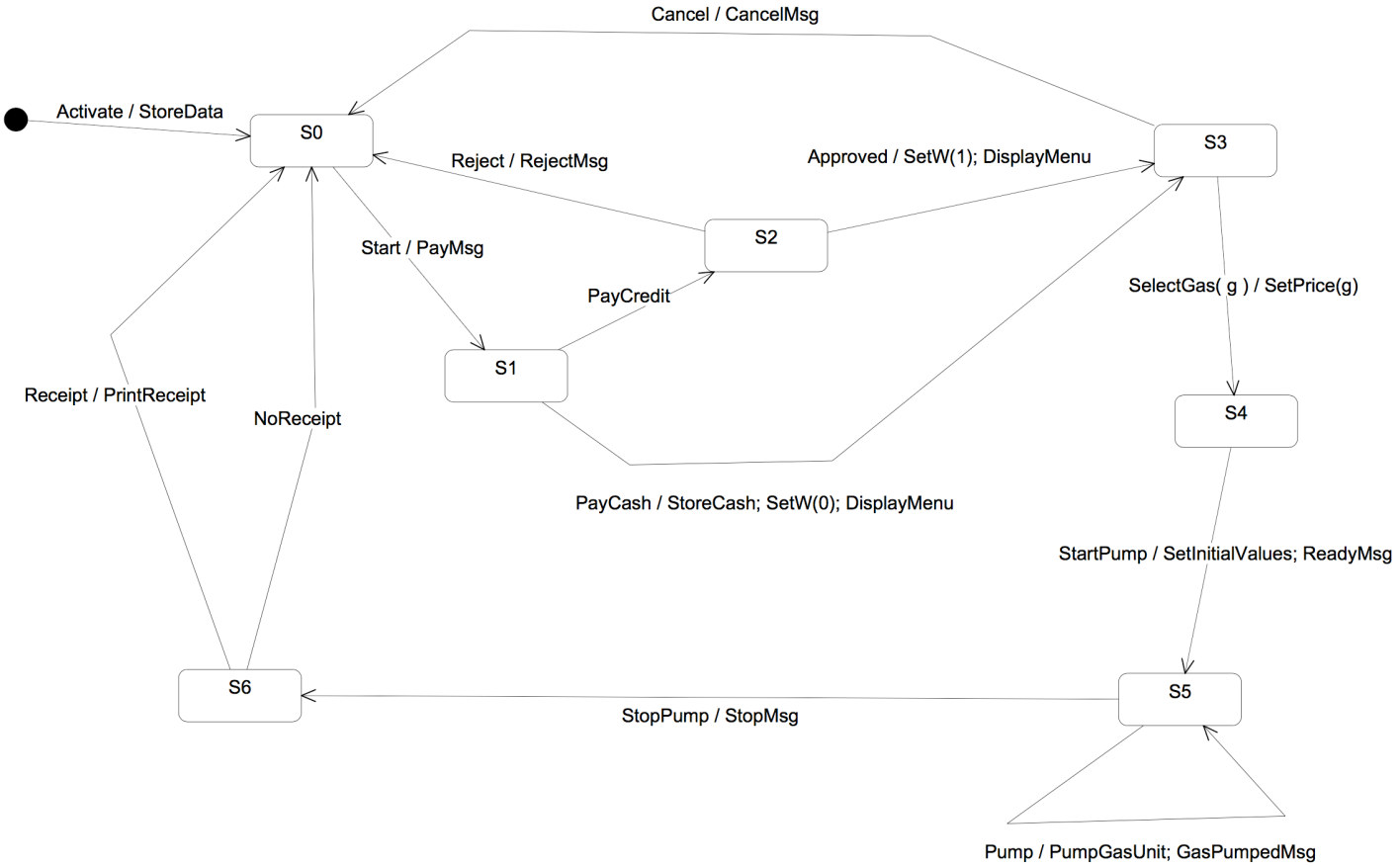
```
StartPump() {
        m->StartPump();
}

PumpGallon() {
        m->Pump();
}

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

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

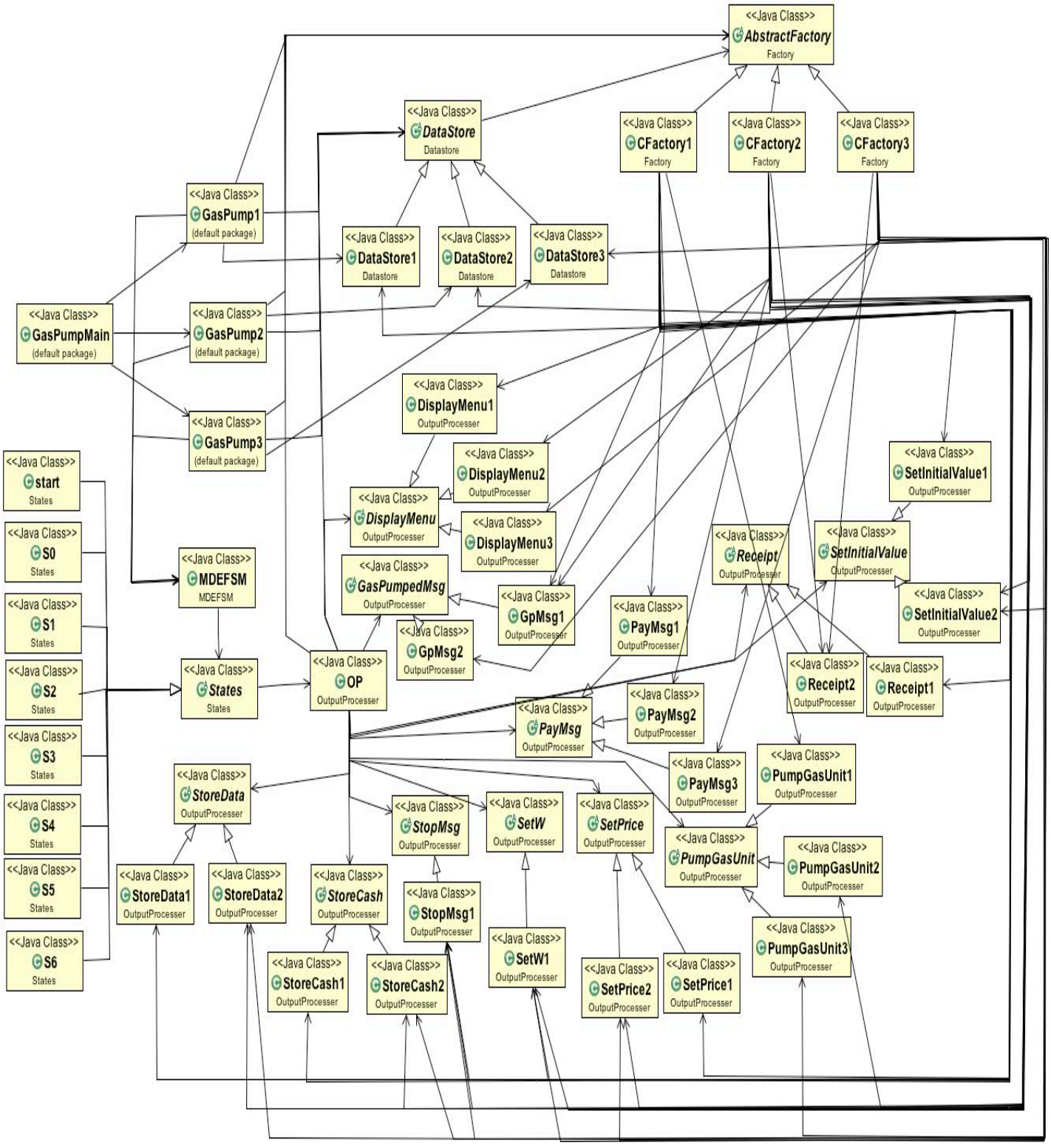
Operations of the Input Processor (GasPump-3)

```
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();
}
```

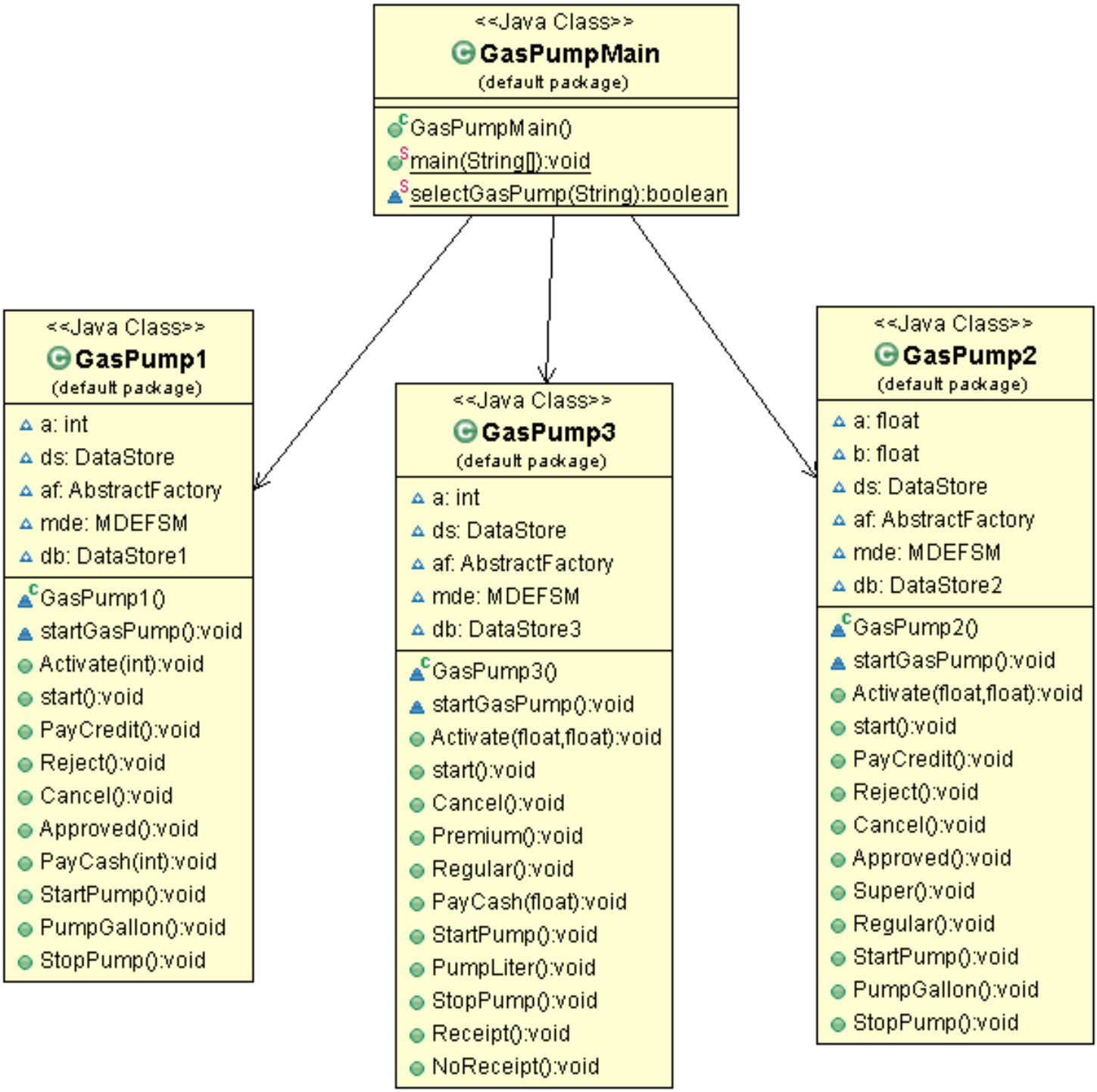
Notice:
cash: contains the value of cash deposited
price: contains the price of the selected gas
L: contains the number of liters already pumped
cash , L, price are in the data store
m: is a pointer to the MDA-EFSM object
d: is a pointer to the Data Store object


Cancel / CancelMsg
Activate / StoreData
S0
Reject / RejectMsg
Approved / SetW(1); DisplayMenu
S3
S2
Start / PayMsg
PayCredit
SelectGas(g) / SetPrice(g)
S1
Receipt / PrintReceipt
NoReceipt
S4
PayCash / StoreCash; SetW(0); DisplayMenu
StartPump / SetInitialValues; ReadyMsg
S6
S5
StopPump / StopMsg
Pump / PumpGasUnit; GasPumpedMsg

Class Diagram for Model Driven Architecture of the GasPump Components



Class Diagram : Part-1



Purpose, Attributes & Responsibilities for class operations and parameters

- Class : GasPumpMain

Purpose	This is a main class for GasPump system to select gaspumps.
Variables & Initialization	g1, g2, g3 are pointers to GasPump-1,2,3
main()	given choices for gas pumps
selectGasPump()	switch case for selecting gas pump

- Class: GasPump-1

Purpose	Provide all operations for cGasPump-1
Variables & Initialization	a is an integer gas price for gaspump-1 ds is a pointer for DataStore af is a pointer for AbstractFactory mde is a pointer of MDEFSM db is a pointer of dataStore1
startGasPump()	list of all operations supported by gas-pump1 and switch case for select them
All other methods	Operations supported by gaspump-1 and pseudo code given in MDAEFSM

- Class: GasPump-2

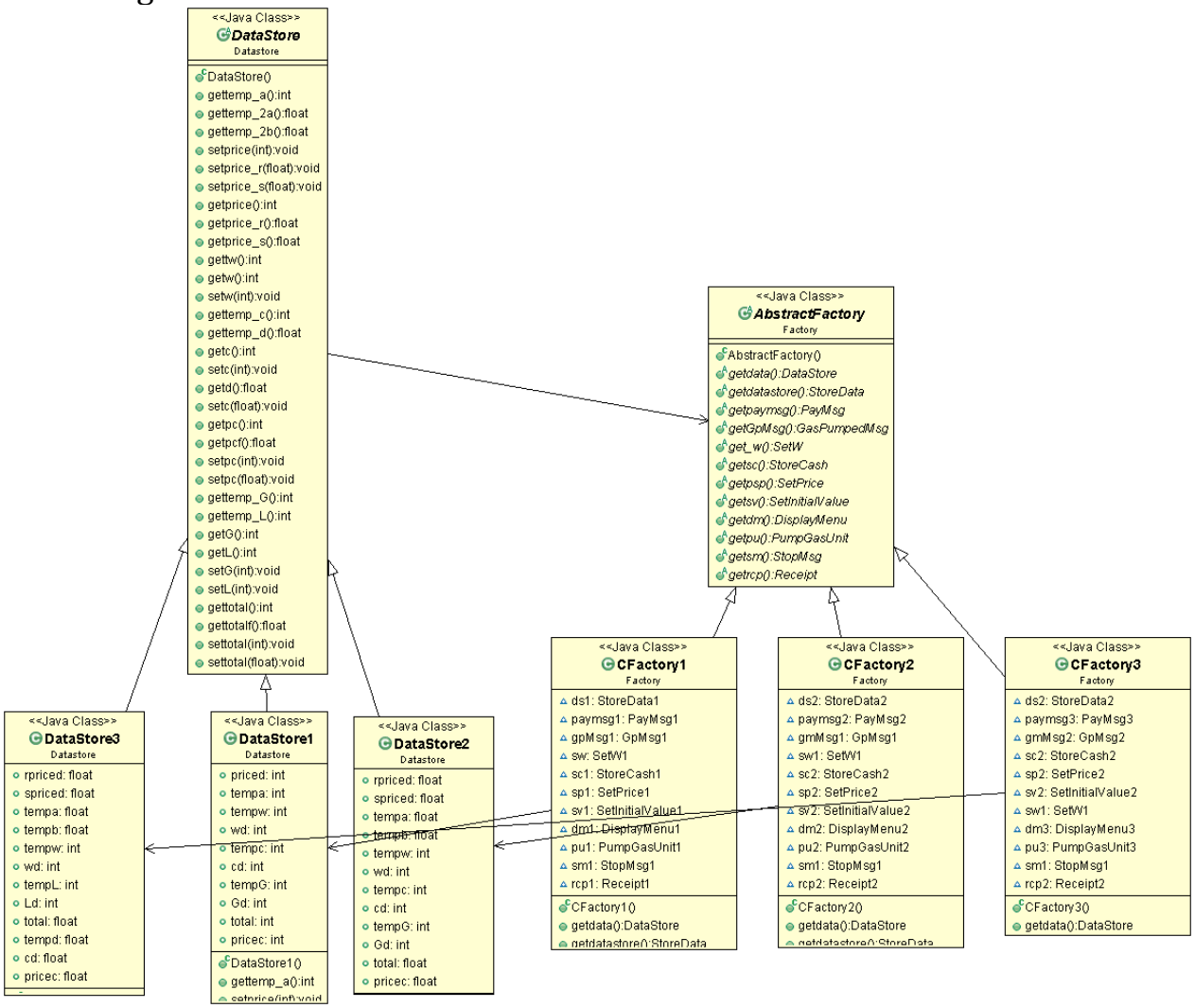
Purpose	Provide all operations for cGasPump-2
Variables & Initialization	a ,b is an float gas price for regular and super gas of gaspump-2 ds is a pointer for DataStore af is a pointer for AbstractFactory mde is a pointer of MDEFSM db is a pointer of dataStore1
startGasPump()	list of all operations supported by gas-pump2 and switch case for select them
All other methods	Operations supported by gaspump-2 and pseudo code given in MDAEFSM

- Class: GasPump-3

Purpose	Provide all operations for cGasPump-3

Variables & Initialization	a , b is a float gas price for super and premium gases of gaspump-3 ds is a pointer for DataStore af is a pointer for AbstractFactory mde is a pointer of MDEFSM db is a pointer of dataStore1
startGasPump()	list of all operations supported by gas-pump3 and switch case for select them
All other methods	Operations supported by gaspump-3 and pseudo code given in MDAEFSM

Class Diagram: Part-2



- Class: DataStore-1

Purpose	Provide all operations for to store data provided in GasPump-1 including gas price initial values and final total
Variables & Initialization	af is a pointer for AbstractFactory mde is a pointer of MDEFSM price is a gas price total is final total of gas pumped
All methods	All methods are getter-setter for data in gaspump-1 execution

- Class: DataStore -2

Purpose	Provide all operations for to store data provided in GasPump-2 including gas price initial values and final total
Variables & Initialization	af is a pointer for AbstractFactory mde is a pointer of MDEFSM rprice is a regular gas price in float sprice is a super gas price inf loat total is final total of gas pumped
All other methods	All methods are getter-setter for data in gaspump-2 execution

- Class: DataStore -3

Purpose	Provide all operations for to store data provided in GasPump-3 including gas price initial values and final total
Variables & Initialization	af is a pointer for AbstractFactory mde is a pointer of MDEFSM rprice is a regular gas price in float sprice is a premium gas price in float total is final total of gas pumped
All other methods	All methods are getter-setter for data in gaspump-3 execution

- Class: CFactory-1

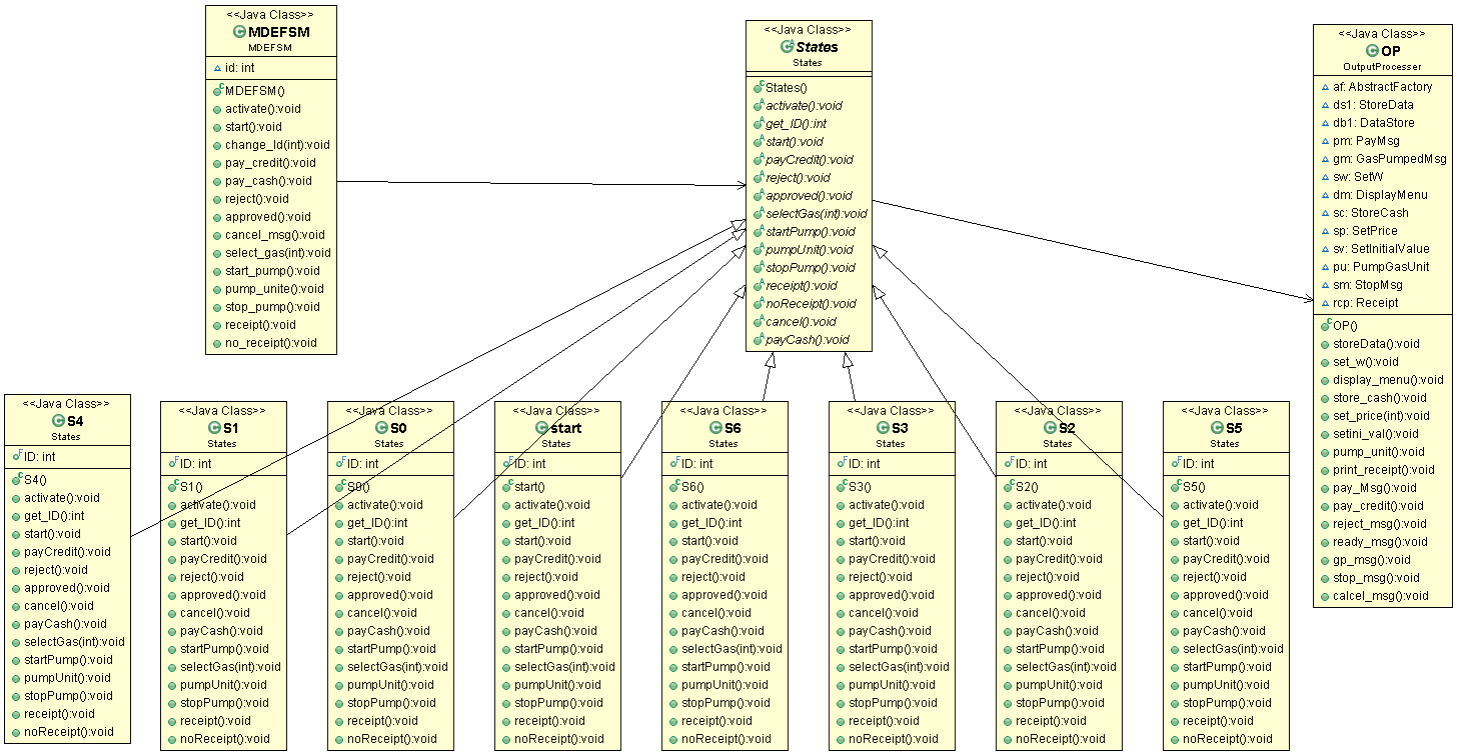
Purpose	This is a concrete factory for gaspump-1 and is used to handle the creation of class objects specific for GasPump-1. This factory creates a specific data store associated with the factory.
Variables & Initialization	ds1 is a pointer to the data store that will be common across all of the classes that access data that the factory creates CFactoty1() is the class constructor and is fired upon creation.
All methods	All methods are return pointers to specific strategy class for GasPump-1

- Class: CFactory -2

Purpose	This is a concrete factory for gaspump-2 and is used to handle the creation of class objects specific for GasPump-2. This factory creates a specific data store associated with the factory.
Variables & Initialization	ds2 is a pointer to the data store that will be common across all of the classes that access data that the factory creates CFactoty2() is the class constructor and is fired upon creation.
All other methods	All methods are return pointers to specific strategy class for GasPump-2

- Class: CFactory -3

Purpose	This is a concrete factory for gaspump-3 and is used to handle the creation of class objects specific for GasPump-3. This factory creates a specific data store associated with the factory.
Variables & Initialization	ds3 is a pointer to the data store that will be common across all of the classes that access data that the factory creates CFactoty3() is the class constructor and is fired upon creation.
All other methods	All methods are return pointers to specific strategy class for GasPump-2



➢ Class: MDEFSM

Purpose	This class represents the common functionality or platform independent logic of all of its clients (GasPump-1,2&3). The separation of this functionality allows for reduced effort during maintenance and when new clients are implemented.
Variables & Initialization	State[] is a array of pointer to state machine . MDAEFSM() is a constructor and it's fired upon creation.
All methods	All methods are call state machine methods as per operation selected on system.

➢ Class: States

Purpose	State class which manages the states by tracking a pointer to the current state class of the EFSM and forwarding calls onto the next state class.
Variables & Initialization	Op is a pointer of output processor
All other	All Methods are abstract supported by given states in system of

methods	GasPump

- Class: start

Purpose	This is the Start State, any execution path leaving this state are implemented here and only executed when the state machine is in this state.
Variables & Initialization	ID represents next state to be executed.
Activate()	Activate gas-pump and store data of that
All other methods	Gives error message of invalid state

- Class: S0

Purpose	This is the S0 state after start , any execution path leaving this state are implemented here and only executed when the state machine is in this state.
Variables & Initialization	ID represents next state to be executed.
pay_msg()	give payment methods available for gaspump
All other methods	Gives error message of invalid state

- Class: S1

Purpose	This is the S1 State, any execution path leaving this state are implemented here and only executed when the state machine is in this state.
Variables & Initialization	ID represents next state to be executed.
payCash()	payment selected by cash and give display menu for available gas type to select
payCredit()	payment selected by credit card
All other methods	Gives error message of invalid state

- Class: S2

Purpose	This is the S2 State, any execution path leaving this state are implemented here and only executed when the state machine is in this state.
Variables & Initialization	ID represents next state to be executed.
Approved()	Approve credit card and give display menu with available gas types for GasPump
Reject()	Reject credit card and give rejection message
All other methods	Gives error message of invalid state

- Class: S3

Purpose	This is the S3 State, any execution path leaving this state are implemented here and only executed when the state machine is in this state.
Variables & Initialization	ID represents next state to be executed.
Cancel_msg	cancel transaction and give message
All other methods	Gives error message of invalid state

- Class: S4

Purpose	This is the S4 State, any execution path leaving this state are implemented here and only executed when the state machine is in this state.
Variables & Initialization	ID represents next state to be executed.
startPump()	start pump with initial values and give ready message
All other methods	Gives error message of invalid state

- Class: S5

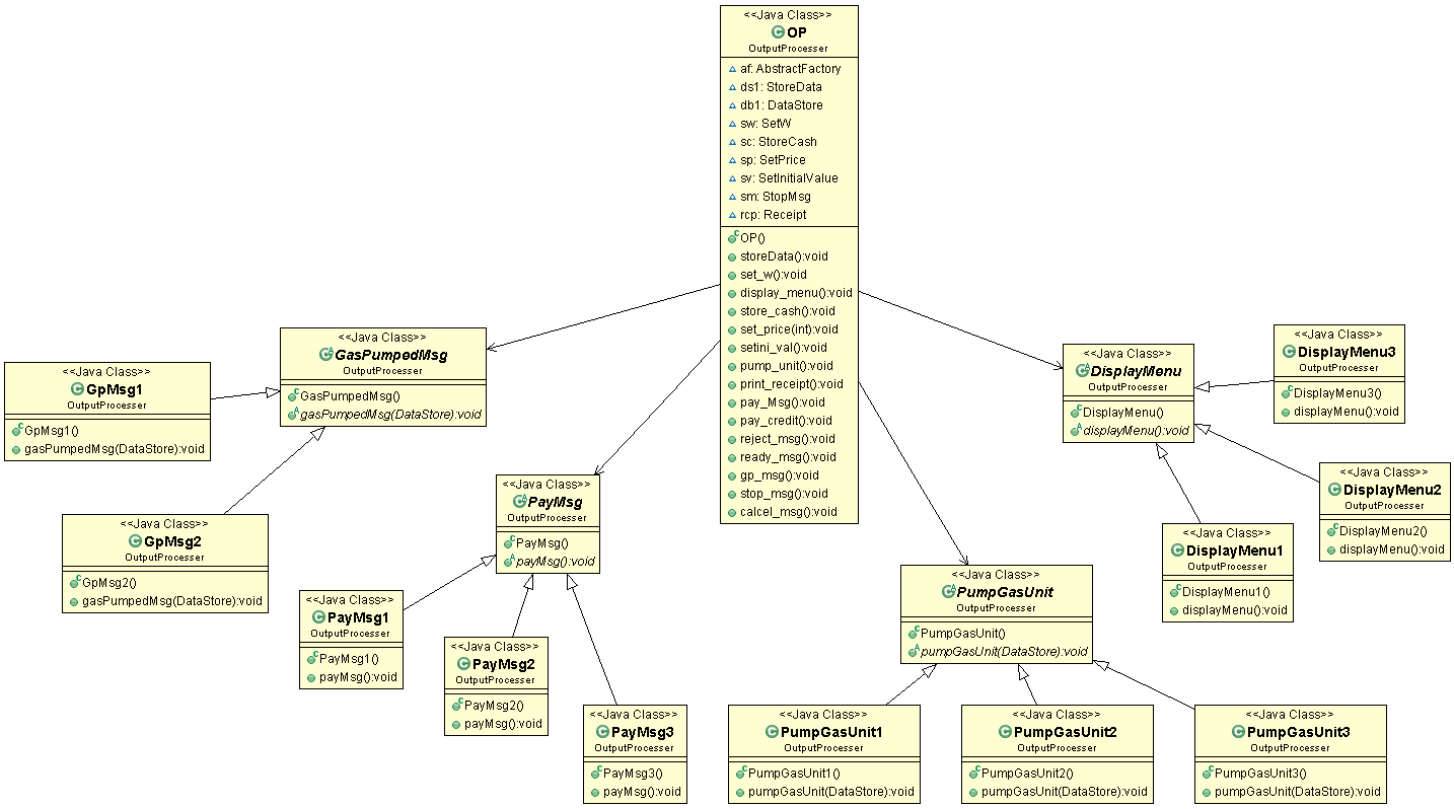
Purpose	This is the S5 State, any execution path leaving this state are implemented here and only executed when the state machine is in this state.
Variables & Initialization	ID represents next state to be executed.
stopPump()	stop pump and give message of stop
pumpUnit()	calculate pump unit and give message of pumped gas
All other methods	Gives error message of invalid state

- Class: S6

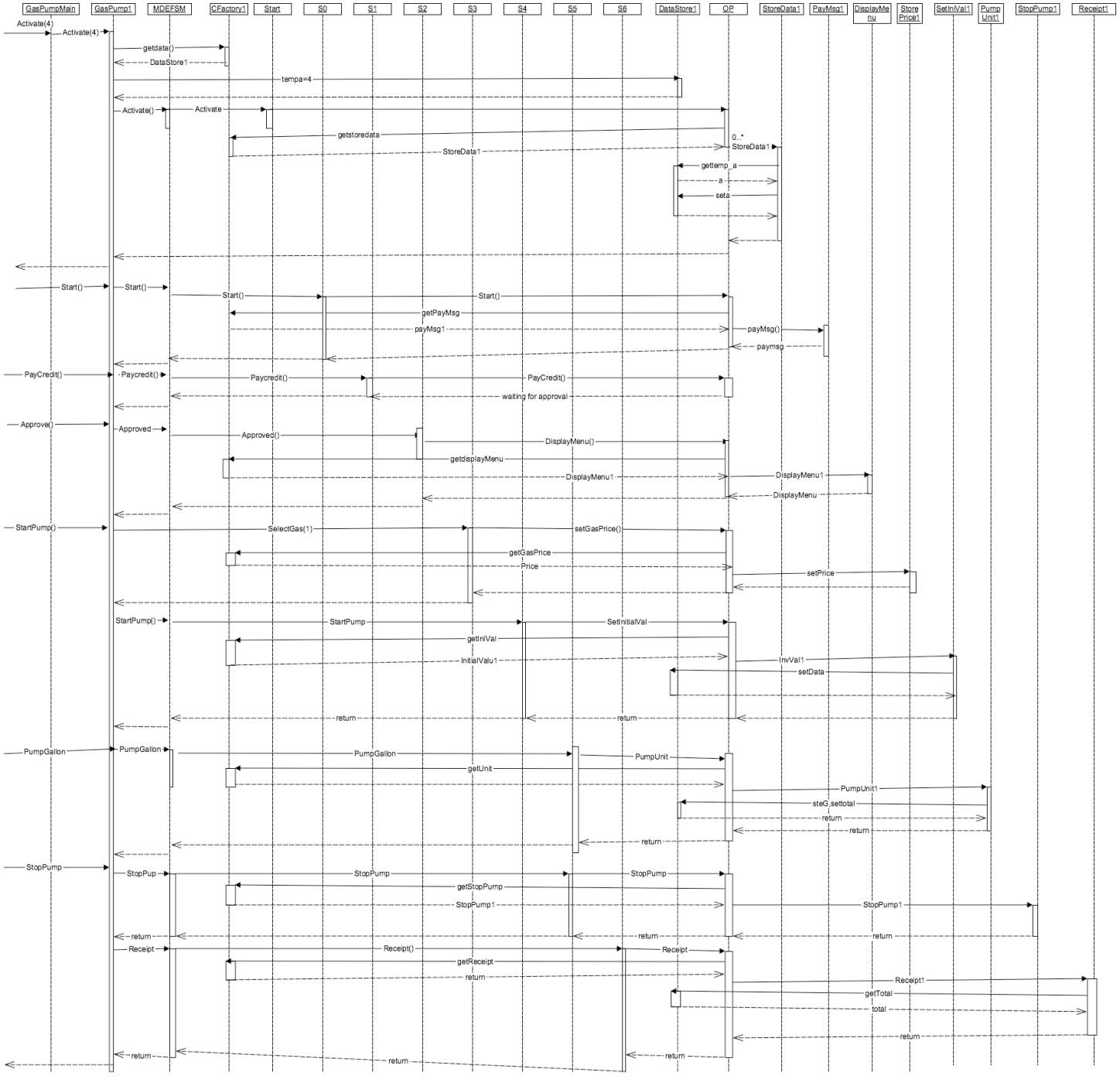
Purpose	This is the S6 State, any execution path leaving this state are implemented here and only executed when the state machine is in this state.
Variables & Initialization	ID represents next state to be executed.
NoReceipt()	No receipt given
Receipt()	Print receipt with total amount
All other methods	Gives error message of invalid state

- Class:OP

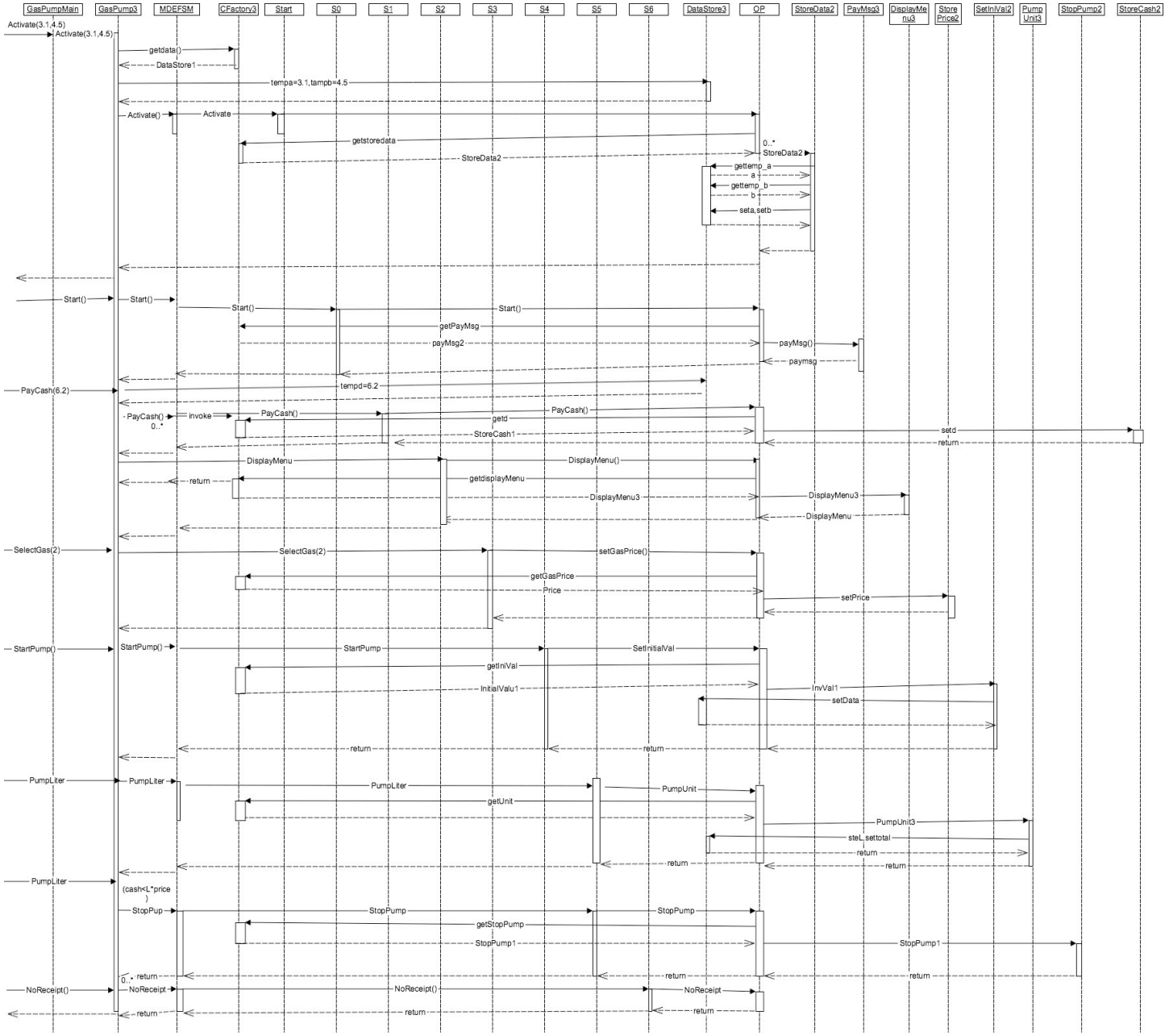
Purpose	This class represents the Output Processor of the MDA and is the client of the various action strategies. Each action having more than one strategy has an abstract class which is associated to the OP, while actions having only one strategy are directly associated with the OP.
Variables & Initialization	af is a pointer of Abstract Factory db1 is a pointer of DataStore OP() is a constructor
All methods	All methods are call action for specific strategy.


<<Java Class>>
OP
OutputProcesser
af: AbstractFactory
ds1: StoreData
db1: DataStore
sw: SetW
sc: StoreCash
sp: SetPrice
sv: SetInitialValue
sm: StopMsg
rcp: Receipt
OP()
storeData():void
set_w():void
display_menu():void
store_cash():void
set_price(int):void
setini_val():void
pump_unit():void
print_receipt():void
pay_Msg():void
pay_credit():void
reject_msg():void
ready_msg():void
gp_msg():void
stop_msg():void
calcel_msg():void
<<Java Class>>
GasPumpedMsg
OutputProcesser
GasPumpedMsg()
gasPumpedMsg(DataStore):void
<<Java Class>>
GpMsg1
OutputProcesser
GpMsg1()
gasPumpedMsg(DataStore):void
<<Java Class>>
GpMsg2
OutputProcesser
GpMsg2()
gasPumpedMsg(DataStore):void
<<Java Class>>
PayMsg
OutputProcesser
PayMsg()
payMsg():void
<<Java Class>>
PayMsg1
OutputProcesser
PayMsg1()
payMsg():void
<<Java Class>>
PayMsg2
OutputProcesser
PayMsg2()
payMsg():void
<<Java Class>>
PayMsg3
OutputProcesser
PayMsg3()
payMsg():void
<<Java Class>>
DisplayMenu
OutputProcesser
DisplayMenu()
displayMenu():void
<<Java Class>>
DisplayMenu3
OutputProcesser
DisplayMenu3()
displayMenu():void
<<Java Class>>
DisplayMenu2
OutputProcesser
DisplayMenu2()
displayMenu():void
<<Java Class>>
DisplayMenu1
OutputProcesser
DisplayMenu1()
displayMenu():void
<<Java Class>>
PumpGasUnit
OutputProcesser
PumpGasUnit()
pumpGasUnit(DataStore):void
<<Java Class>>
PumpGasUnit1
OutputProcesser
PumpGasUnit1()
pumpGasUnit(DataStore):void
<<Java Class>>
PumpGasUnit2
OutputProcesser
PumpGasUnit2()
pumpGasUnit(DataStore):void
<<Java Class>>
PumpGasUnit3
OutputProcesser
PumpGasUnit3()
pumpGasUnit(DataStore):void

Sequence Diagram for Model Driven Architecture of the GasPump -1 Components



Sequence Diagram for Model Driven Architecture of the GasPump -2 Components



Source Code for Model Driven Architecture of the GasPump -1 Components

GasPumpMain Class

```
import java.util.Scanner;


public class GasPumpMain {

	GasPump1 gp1;
	GasPump2 gp2;
	GasPump3 gp3;

	public static void main(String []args) {
		boolean c = true;
		while(c) {
			System.out.println("\n\n**********Welcome to
GasPump**********\n");
			System.out.println("Select GasPump from below Chocies \n ");
			System.out.println("1.  GasPump-1 \n"
					+"2.  GasPump-2 \n"
					+"3.  GasPump-3 \n"
					+"4.  Exit \n");

			Scanner sc = new Scanner(System.in);
			System.out.println("Enter your choice : \n");
			String gp = sc.next();
			if(gp=="4") {
				System.out.println("\n **********GasPump
Stopped**********");
				c = false;
			}
			else {
				c=selectGasPump(gp);
			}
		}
	}
```

```
	/**
	 * Select from available options of GasPump
	 * @param gp
	 */
	static boolean selectGasPump(String gp) {

		switch(gp) {
		case "1":
			GasPump1 gp1 = new GasPump1();
			gp1.startGasPump();
			break;
		case "2":
			GasPump2 gp2 = new GasPump2();
			gp2.startGasPump();
			break;
		case "3":
			GasPump3 gp3 = new GasPump3();
			gp3.startGasPump();
			break;
		case "4":
			System.out.println("\n **************GasPump
Stopped***************");
			return false;
		default:
			System.out.println("Invalid choice \n");
			break;
		}
		return true;
	}
}
```

- GasPump-1 Class

```
/*
 * GasPump-1 Implementation Class
 */
import java.util.Scanner;

import Datastore.DataStore1;
import Datastore.DataStore;
import Factory.AbstractFactory;
```

```
import Factory.CFactory1;
import MDEFSM.MDEFSM;


public class GasPump1 {
      int a;
      DataStore ds;
      AbstractFactory af;
      MDEFSM mde;
      DataStore1 db;
      GasPump1()
      {
            af= new CFactory1();
            mde =new MDEFSM();
            DataStore.af=af;
      }
      /*
       * Start GasPump-1 and list all Possible operations supported by it.
       */
      void startGasPump() {
            System.out.println("\n\n*****************GasPump-1*****************\n");

            System.out.println("The GasPump-1 component supports the following
operations: \n"
                        + "         1. Activate (int a) \n"      // the gas pump is
activated where a is the price of the gas per gallon
                        + "         2. Start()  \n"        //start the transaction
                        + "         3. PayCredit()  \n"          // pay for gas by a
credit card
                        + "         4. Reject()  \n"       // credit card is rejected
                        + "         5. Cancel()  \n"       // cancel the transaction
                        + "         6. Approved()  \n"           // credit card is
approved
                        + "         7. PayCash(int c)  \n"       // pay for gas by cash,
where c represents prepaid cash
                        + "         8. StartPump()  \n"          // start pumping gas
                        + "         9. PumpGallon()  \n"         // one gallon of gas is
disposed
                        + "         10. StopPump()  \n"          // stop pumping gas
                        + "         11. ExitPump()  \n");        //Exit from Pump1


            System.out.println("Execution Started for GasPump-1 \n");

            boolean cp = true;
            while(cp) {
                  System.out.println("\n---> Select Operations :\n 1.Activate  "
                                                +" 2.Start  "
                                                +" 3.PayCredit  "
                                                +" 4.Reject  "
                                                +" 5.Cancel  "
                                                +" 6.Approved  "
                                                +" 7.PayCash  "
                                                +" 8.StartPump  "
                                                +" 9.PumpGallon  "
```

```
                                        +" 10.StopPump  "
                                        +" 11.Exit \n ");
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter your choice : ");
                int op = sc.nextInt();

                if(op == 11) {
                      System.out.println("GasPump-1 Exit \n");
                      break;
                }
                switch(op)
                {
                      case 1:
                            int a ;
                            System.out.println("\n Operation:  Activate(int a) \n  ");
                            System.out.println("Enter Value for a (Gas Price) : ");
                            try {
                                  a=sc.nextInt();
                            }
                            catch(Exception e) {
                                  a=0;
                                  System.out.print("\nError!! -  Gas Price must be in
Integer for GasPump-1");
                            }

                            Activate(a);
                            break;
                      case 2:
                            System.out.println("\n Operation:  Start() \n  ");
                            start();
                            break;
                      case 3:
                            System.out.println("\n Operation:  PayCredit() \n  ");
                            PayCredit();
                            break;
                      case 4:
                            System.out.println("\n Operation:  Reject() \n  ");
                            Reject();
                            break;
                      case 5:
                            System.out.println("\n Operation:  Cancel() \n  ");
                            Cancel();
                            break;
                      case 6:
                            System.out.println("\n Operation:  Approved() \n  ");
                            Approved();
                            break;
                      case 7:
                            System.out.println("\n Operation:  PayCash(int c) \n  ");
                           System.out.println("\n Please Enter Cash Amount : ");
                           Scanner pc =new Scanner(System.in);
                           int c = pc.nextInt();
                           if(c>0) {
                             PayCash(c);
                           }
```

```
                              else {
                                System.out.println("!!! Plese Enter Valid Cash Payment
!!!!");
                              }
                              break;
                        case 8:
                              System.out.println("\n Operation:  StartPump() \n  ");
                              StartPump();
                              break;
                        case 9:
                              System.out.println("\n Operation:  PumpGallon() \n  ");
                              PumpGallon();
                              break;
                        case 10:
                              System.out.println("\n Operation:  StopPump() \n  ");
                              StopPump();
                              break;
                        case 11:
                              cp = false;
                              break;
                        default:
                              System.out.println("Invalid Choice");
                        break;
                  }
            }
      }

      /*
       * Activate GasPump1 with Gas Price is a
       */
      public void Activate(int a) {
            if(a > 0){
                  db=(DataStore1) af.getdata();
                  db.tempa=a;
                  mde.activate();
                  System.out.println("\n ******GasPump 1 is Activated****** ");
            }
            else {
                  System.out.println("\n Gas Price must be greater than 0");
            }
      }

      /*
       * Start GasPump1
       */
      public void start() {
            mde.start();
      }
      /*
       * Payment by Credit
       */
      public void PayCredit() {
            mde.pay_credit();
      }
      /*
```

```
	 * Reject Credit card
	 */
	public void Reject() {
		mde.reject();
	}
	/*
	 * Cancel Transaction
	 */
	public void Cancel() {
		mde.cancel_msg();
	}
	/*
	 * Approve Credit card
	 */
	public void Approved() {
		db=(DataStore1) af.getdata();
		db.tempw=1;
		mde.approved();
	}
	/*
	 * Payment by cash
	 */
	public void PayCash(int c) {
		db=(DataStore1) af.getdata();
		db.tempw=0;
		db.tempc=c;
		mde.pay_cash();
	}
	/*
	 * Start Pumping for gas
	 */
	public void StartPump() {
		System.out.println("\n******GasPump Start******");
		mde.select_gas(1);
		db=(DataStore1) af.getdata();
		db.tempG=0;
		mde.start_pump();
	}
	/*
	 * Pumping of Gas is running upto manually select stop pump or cash becomes less
then 1 gallon price
	 */
	public void PumpGallon() {
		System.out.println("\n******Pumping for Gas Started from GasPump-1******");
		db=(DataStore1) af.getdata();
		if(db.getw()==1) {
			mde.pump_unite();

		}
		else if (db.getc()<((db.getG()+1)*db.getpc())) {
			mde.stop_pump();
			mde.receipt();
		}
		else {
			mde.pump_unite();
```

```
		}
	}
	/*
	 * Stop pumping of gas
	 */
	public void StopPump() {
		mde.stop_pump();
		mde.receipt();
	}
}
```

➢ GasPump-2 Class

```
/*
 * GasPump-2 Implementation Class
 */
import java.util.Scanner;

import Datastore.DataStore2;
import Datastore.DataStore;
import Factory.AbstractFactory;
import Factory.CFactory2;
import MDEFSM.MDEFSM;


public class GasPump2 {
	float a,b;
	DataStore ds;
	AbstractFactory af;
	MDEFSM mde;
	DataStore2 db;
	/*
	 * Constructor for GasPump2
	 */
	GasPump2(){
		af= new CFactory2();
		mde =new MDEFSM();
		DataStore.af=af;
	}
	/*
	 * List all Possible operations for GasPump2
	 */
	void startGasPump() {
		System.out.println("*****************GasPump-2*****************\n");

		System.out.println("The GasPump-2 component supports the following
operations: \n"
				+ "1. Activate (float a, float b) \n"  // the 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()  \n"          //start the transaction
				+ "3. PayCredit()  \n"      // pay for gas by a credit card
				+ "4. Reject()  \n"         // credit card is rejected
```

```
                        + "5. Cancel()  \n"          // cancel the transaction
                        + "6. Approved()  \n"        // credit card is approved
                        + "7. Super()  \n"           // Super gas is selected
                        + "8. Regular() \n"     // Regular gas is selected
                        + "9. StartPump()  \n"       // start pumping gas
                        + "10. PumpGallon()  \n"          // one gallon of gas is
disposed
                        + "11. StopPump()  \n"       // stop pumping gas
                        + "12. ExitPump()  \n");          //Exit from Pump1


            System.out.println("Execution Started for GasPump-2 \n");

            boolean cp = true;
            while(cp) {
                  System.out.println("\n Operations :\n 1.Activate  "
                                                +"2.Start  "
                                                +"3.PayCredit  "
                                                +"4.Reject  "
                                                +"5.Cancel  "
                                                +"6.Approved  "
                                                +"7.Super  "
                                                +"8.Regular  "
                                                +"9.StartPump  "
                                                +"10.PumpGallon  "
                                                +"11.StopPump  "
                                                +"12.Exit \n ");
                  Scanner sc = new Scanner(System.in);
                  System.out.println("Enter your choice : \n");
                  int op = sc.nextInt();

                  if(op == 12) {
                        System.out.println("GasPump-2 Exit \n");
                        break;
                  }
                  switch(op)
                  {
                        case 1:
                              float a, b  ;
                              System.out.println("\n Operation:  Activate(a,b) \n  ");
                              System.out.println("Enter Price of Regular Gas : ");
                              try {
                                    a=sc.nextFloat();
                              //    break;
                              }
                              catch(Exception e) {
                                    a=0;
                                    System.out.print("Gas Price must be in float for
GasPump-2");
                              }
                              System.out.println("Enter Price of Supar Gas : ");
                              try {
                                    b=sc.nextFloat();
                              //    break;
                              }
```

```
                        catch(Exception e) {
                              b=0;
                              System.out.print("Gas Price must be in float for
GasPump-2");
                        }

                        Activate(a,b);
                        break;
                  case 2:
                        System.out.println("\n Operation:  start() \n  ");
                        start();
                        break;
                  case 3:
                        System.out.println("\n Operation:  PayCredit() \n  ");
                        PayCredit();
                        break;
                  case 4:
                        System.out.println("\n Operation:  Reject() \n  ");
                        Reject();
                        break;
                  case 5:
                        System.out.println("\n Operation:  Cancel() \n  ");
                        Cancel();
                        break;
                  case 6:
                        System.out.println("\n Operation:  Approved() \n  ");
                        Approved();
                        break;
                  case 7:
                        System.out.println("\n Operation:  Super() \n  ");
                       Super();
                       break;
                  case 8:
                        System.out.println("\n Operation:  Regular() \n  ");
                       Regular();
                       break;
                  case 9:
                        System.out.println("\n Operation:  StartPump() \n  ");
                        StartPump();
                        break;
                  case 10:
                        PumpGallon();
                        break;
                  case 11:
                        System.out.println("\n Operation:  StopPump() \n  ");
                        StopPump();
                        break;
                  case 12:
                        cp = false;
                        break;
                  default:
                        System.out.println("Invalid Choice");
                  break;
            }
      }
```

```
	}
	public void Activate(float a, float b) {
		if(a > 0 && b>0){
			db=(DataStore2) af.getdata();
			db.tempa=a;
			db.tempb=b;
			mde.activate();
		}
		else {
			System.out.println("Gas Price must be greater than 0");
		}
	}
	public void start() {
		mde.start();
	}
	public void PayCredit() {
		mde.pay_credit();
	}
	public void Reject() {
		mde.reject();
	}
	public void Cancel() {
		mde.cancel_msg();
	}
	public void Approved() {
		db=(DataStore2) af.getdata();
		db.tempw=1;
		mde.approved();
	}
	public void Super() {
		mde.select_gas(2);
	}
	public void Regular() {
		mde.select_gas(2);
	}
	public void StartPump() {
		System.out.println("\n******Gas Pump Start******");
		db=(DataStore2) af.getdata();
		db.tempG=0;
		mde.start_pump();
	}
	public void PumpGallon() {
		System.out.println("\n******Pumping for Gas Started from GasPump-1******");
		db=(DataStore2) af.getdata();
		mde.pump_unite();
	}
	public void StopPump() {
		mde.stop_pump();
		mde.receipt();
	}
}
```

➢ GasPump-3 Class

```
/*
 * GasPump-3 Implementation Class
 */
import java.util.Scanner;

import Datastore.DataStore3;
import Datastore.DataStore;
import Factory.AbstractFactory;
import Factory.CFactory3;
import MDEFSM.MDEFSM;


public class GasPump3 {
	int a;
	DataStore ds;
	AbstractFactory af;
	MDEFSM mde;
	DataStore3 db;
	GasPump3()
	{
		af = new CFactory3();
		mde =new MDEFSM();
		DataStore.af=af;
	}
	void startGasPump() {
		System.out.println("*****************GasPump-3*****************\n");

		System.out.println("The GasPump-3 component supports the following
operations: \n"
				+ "1. Activate (int a) \n"		// the gas pump is activated
where a is the price of the gas per gallon
				+ "2. Start()  \n"			//start the transaction
				+ "3. PayCash(float c)  \n"		// pay for gas by cash, where
c represents prepaid cash
				+ "4. Cancel()  \n"			// cancel the transaction
				+ "5. Premium()  \n"   //Premium gas is Selected
				+ "6. Regular()  \n"			// Regular gas is selected
				+ "7. StartPump()  \n"		// start pumping gas
				+ "8. PumpLiter()  \n"		// one gallon of gas is disposed
				+ "10. StopPump()  \n"		// stop pumping gas
				+ "11. Receipt()   \n"		//Receipt is requested
				+ "12. NoReceipt()  \n"    //No Receipt
				+ "13. ExitPump()  \n");			//Exit from Pump1


		System.out.println("Execution Started for GasPump-3 \n");

		boolean cp = true;
		while(cp) {
			System.out.println("\n Operations :\n 1.Activate  "
									+"2.Start  "
									+"3.PayCash  "
```

```
                                        +"4.Cancel  "
                                        +"5.Regular  "
                                        +"6.Premium  "
                                        +"7.StartPump  "
                                        +"8.PumpLiter  "
                                        +"9.StopPump  "
                                        +"10.Receipt  "
                                        +"11.No Receipt  "
                                        +"12.Exit \n ");
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter your choice : \n");
                int op = sc.nextInt();

                if(op == 12) {
                      System.out.println("GasPump-3 Exit \n");
                      break;
                }
                switch(op)
                {
                      case 1:
                            float a, b  ;
                            System.out.println("\n Operation:  Activate(a,b) \n  ");
                            System.out.println("Enter Price of Regular Gas : ");
                            try {
                                  a=sc.nextFloat();
                            //    break;
                            }
                            catch(Exception e) {
                                  a=0;
                                  System.out.print("Gas Price must be in float for
GasPump-2");
                            }
                            System.out.println("Enter Price of Premium Gas : ");
                            try {
                                  b=sc.nextFloat();
                            //    break;
                            }
                            catch(Exception e) {
                                  b=0;
                                  System.out.print("Gas Price must be in float for
GasPump-2");
                            }

                            Activate(a,b);
                            break;
                      case 2:
                            System.out.println("\n Operation:  start() \n  ");
                            start();
                            break;
                      case 3:
                            System.out.println("\n Operation:  PayCash(c) \n  ");
                            System.out.println("\n Please Enter Cash Amount : ");
                           Scanner pc =new Scanner(System.in);
                           float c = pc.nextFloat();
                           if(c>0) {
```

```
                              PayCash(c);
                            }
                            else {
                              System.out.println("!!! Plese Enter Valid Cash Payment
!!!!");
                            }
                            break;
                      case 4:
                            System.out.println("\n Operation:  Cancel() \n  ");
                            Cancel();
                            break;
                      case 5:
                            System.out.println("\n Operation:  Regular() \n  ");
                            Regular();
                            break;
                      case 6:
                            System.out.println("\n Operation:  Premium() \n  ");
                            Premium();
                            break;
                      case 7:
                            System.out.println("\n Operation:  StartPump() \n  ");
                            StartPump();
                           break;
                      case 8:
                            System.out.println("\n Operation:  PumpLiter() \n  ");
                            PumpLiter();
                            break;
                      case 9:
                            System.out.println("\n Operation:  StopPump() \n  ");
                            StopPump();
                            break;
                      case 10:
                            System.out.println("\n Operation:  Receipt() \n  ");
                            Receipt();
                            break;
                      case 11:
                            System.out.println("\n Operation:  NoReceipt() \n  ");
                            NoReceipt();
                            break;
                      case 12:
                            cp = false;
                            break;
                      default:
                            System.out.println("Invalid Choice");
                      break;
                  }
            }


//          if(op == 1) {
//                System.out.println("Operation :  Acivate(int a) \n");
//                System.out.println("Enter value for a : ");
//                int a = sc.nextInt();
//          }
      }
```

```
public void Activate(float a, float b) {
	if(a > 0 && b>0){
		db=(DataStore3) af.getdata();
		db.tempa=a;
		db.tempb=b;
		mde.activate();
	}
	else {
		System.out.println("Gas Price must be greater than 0");
	}
}
public void start() {
	mde.start();
}
public void Cancel() {
	mde.cancel_msg();
}
public void Premium() {
	mde.select_gas(2);
}
public void Regular() {
	mde.select_gas(1);
}

public void PayCash(float c) {
	db=(DataStore3) af.getdata();
	db.tempw=0;
	db.tempd=c;
	mde.pay_cash();
}
public void StartPump() {
	System.out.println("\n******GasPump Start******");
	db=(DataStore3) af.getdata();
	db.tempL=0;
	mde.start_pump();
}
public void PumpLiter() {
	System.out.println("\n******Pumping for Gas Started from GasPump-1******");
	db=(DataStore3) af.getdata();
	if (db.getd() < ((db.getL()+1)*db.getpcf())) {
		mde.stop_pump();
	}
	else {
		mde.pump_unite();
	}
}
public void StopPump() {
	mde.stop_pump();
}
public void Receipt() {
	mde.receipt();
}
public void NoReceipt() {
	mde.no_receipt();
}
```

```
}
```

➢ MDEFSM Class

```
/*
 * MDEFSM class for whole system
 */
package MDEFSM;

import States.S0;
import States.S1;
import States.S2;
import States.S3;
import States.S4;
import States.S5;
import States.S6;
import States.States;
import States.start;

public class MDEFSM {
	States[] state;
	int id=0;
	/*
	 * Constructor for MDEFSM including all states
	 */
	public MDEFSM() {
		state=new States[8];
		state[0]=new start();
		state[1]=new S0();
		state[2]=new S1();
		state[3]=new S2();
		state[4]=new S3();
		state[5]=new S4();
		state[6]=new S5();
		state[7]=new S6();
	}

	public void activate() {
		state[id].activate();
```

```
        if(state[id].get_ID()==0) {
                id=1;
        }
}
public void start() {
        state[id].start();
        if(state[id].get_ID()==1) {
                id=2;
        }
}
public void change_Id(int id) {
        this.id=id;
}
public void pay_credit() {
        state[id].payCredit();
        if(state[id].get_ID()==2){
                id=3;
        }
}
public void pay_cash() {
        state[id].payCash();
        if(state[id].get_ID()==2) {
                id=4;
        }
}
public void reject() {
        state[id].reject();
        if(state[id].get_ID()==3){
                id=1;
        }
}
public void approved() {
        state[id].approved();
        if(state[id].get_ID()==3) {
                id=4;
        }
}
public void cancel_msg() {
        state[id].cancel();
```

```
        if(state[id].get_ID()==4) {
                id=1;
        }
}
public void select_gas(int i) {
        state[id].selectGas(i);
        if(state[id].get_ID()==4) {
                id=5;
        }
}
public void start_pump() {
        state[id].startPump();
        if(state[id].get_ID()==5) {
                id=6;
        }
}
public void pump_unite() {
        state[id].pumpUnit();
        if(state[id].get_ID()==6) {
                id=6;
        }
}

public void stop_pump() {
        state[id].stopPump();
        if(state[id].get_ID()==6) {
                id=7;
        }
}
public void receipt() {
        state[id].receipt();
        if(state[id].get_ID()==7) {
                id=1;
        }
}
public void no_receipt() {
        state[id].noReceipt();
        if(state[id].get_ID()==7) {
                id=1;
```

```
            }
      }
}
```

➢ AbstractFactory Class

```
/*
 * AbstractFactory class for whole system
 */
package Factory;

import Datastore.DataStore;
import OutputProcesser.DisplayMenu;
import OutputProcesser.StoreData;
import OutputProcesser.GasPumpedMsg;
import OutputProcesser.PumpGasUnit;
import OutputProcesser.PayMsg;
import OutputProcesser.Receipt;
import OutputProcesser.StoreCash;
import OutputProcesser.StopMsg;
import OutputProcesser.SetPrice;
import OutputProcesser.SetInitialValue;
import OutputProcesser.SetW;

public abstract class AbstractFactory {
      public abstract DataStore getdata();
      public abstract StoreData getdatastore();
      public abstract PayMsg getpaymsg();
      public abstract GasPumpedMsg getGpMsg();
    public abstract SetW get_w();
    public abstract StoreCash getsc();
    public abstract SetPrice getpsp();
    public abstract SetInitialValue getsv();
    public abstract DisplayMenu getdm();
      public abstract PumpGasUnit getpu();
    public abstract StopMsg getsm();
    public abstract Receipt getrcp();
}
```

Class : CFactory-1

```
/*
 * Concrete factory class for GasPump-1
 */
package Factory;

import Datastore.DataStore;
import Datastore.DataStore1;
import OutputProcesser.DisplayMenu;
import OutputProcesser.DisplayMenu1;
```

```
import OutputProcesser.SetW;
import OutputProcesser.StoreData;
import OutputProcesser.StoreData1;
import OutputProcesser.GasPumpedMsg;
import OutputProcesser.GpMsg1;
import OutputProcesser.PumpGasUnit;
import OutputProcesser.PumpGasUnit1;
import OutputProcesser.PayMsg;
import OutputProcesser.Receipt;
import OutputProcesser.Receipt1;
import OutputProcesser.StoreCash;
import OutputProcesser.StoreCash1;
import OutputProcesser.StopMsg;
import OutputProcesser.StopMsg1;
import OutputProcesser.SetPrice;
import OutputProcesser.SetPrice1;
import OutputProcesser.SetInitialValue;
import OutputProcesser.SetInitialValue1;
import OutputProcesser.PayMsg1;
import OutputProcesser.SetW1;


public class CFactory1 extends AbstractFactory {
      DataStore1 db1;
      StoreData1 ds1;
      PayMsg1 paymsg1;
      GpMsg1 gpMsg1;
      SetW1 sw;
      StoreCash1 sc1;
      SetPrice1 sp1;
      SetInitialValue1 sv1;
      DisplayMenu1 dm1;
      PumpGasUnit1 pu1;
      StopMsg1 sm1;
      Receipt1 rcp1;
      /*
       * Concrete Factory for GasPump1
       */
      public CFactory1() {
            db1 =new DataStore1();
          ds1=new StoreData1();
          paymsg1=new PayMsg1();
          gpMsg1 = new GpMsg1();
          sw=new SetW1();
          sc1=new StoreCash1();
          sp1=new SetPrice1();
          sv1=new SetInitialValue1();
          dm1=new DisplayMenu1();
          pu1=new PumpGasUnit1();
          sm1=new StopMsg1();
          rcp1=new Receipt1();
      }
      public DataStore getdata() {
      return db1;
      }
```

```
@Override
public StoreData getdatastore() {
	// TODO Auto-generated method stub
	return ds1;
}
@Override
public PayMsg getpaymsg() {
	// TODO Auto-generated method stub
	return paymsg1;
}
@Override
public SetW get_w() {
	// TODO Auto-generated method stub
	return sw;
}
@Override
public StoreCash getsc() {
	// TODO Auto-generated method stub
	return sc1;
}
@Override
public SetPrice getpsp() {
	// TODO Auto-generated method stub
	return sp1;
}
@Override
public SetInitialValue getsv() {
	// TODO Auto-generated method stub
	return sv1;
}
@Override
public DisplayMenu getdm() {
	// TODO Auto-generated method stub
	return dm1;
}

@Override
public PumpGasUnit getpu() {
	// TODO Auto-generated method stub
	return pu1;
}
@Override
public StopMsg getsm() {
	// TODO Auto-generated method stub
	return sm1;
}
@Override
public Receipt getrcp() {
	// TODO Auto-generated method stub
	return rcp1;
}
@Override
public GasPumpedMsg getGpMsg() {
	// TODO Auto-generated method stub
	return gpMsg1;
```

```
      }


}
```

Class : CFactory-2

```
^^
/*
 * Concrete factory class for GasPump-2
 */
package Factory;

import Datastore.DataStore2;
import Datastore.DataStore;
import OutputProcesser.DisplayMenu;
import OutputProcesser.DisplayMenu2;
import OutputProcesser.SetW;
import OutputProcesser.SetW1;
import OutputProcesser.StopMsg1;
import OutputProcesser.StoreData;
import OutputProcesser.StoreData2;
import OutputProcesser.GasPumpedMsg;
import OutputProcesser.GpMsg1;
import OutputProcesser.PumpGasUnit;
import OutputProcesser.PumpGasUnit2;
import OutputProcesser.PayMsg;
import OutputProcesser.Receipt;
import OutputProcesser.Receipt2;
import OutputProcesser.StoreCash;
import OutputProcesser.StoreCash2;
import OutputProcesser.StopMsg;
import OutputProcesser.SetPrice;
import OutputProcesser.SetPrice2;
import OutputProcesser.SetInitialValue;
import OutputProcesser.SetInitialValue2;
import OutputProcesser.PayMsg2;

public class CFactory2 extends AbstractFactory{
   DataStore2 db2;
   StoreData2 ds2;
   PayMsg2 paymsg2;
   GpMsg1 gmMsg1;
    SetW1 sw1;
      StoreCash2 sc2;
      SetPrice2 sp2;
      SetInitialValue2 sv2;
      DisplayMenu2 dm2;
      PumpGasUnit2 pu2;
      StopMsg1 sm1;
      Receipt2 rcp2;
```

```
  /*
   * Concrete Factory for GasPump2
   */
  public CFactory2() {
        db2 =new DataStore2();
      ds2=new StoreData2();
      paymsg2=new PayMsg2();
      gmMsg1 = new GpMsg1();
      sw1=new SetW1();
      sc2=new StoreCash2();
      sp2=new SetPrice2();
      sv2=new SetInitialValue2();
      dm2=new DisplayMenu2();
      pu2=new PumpGasUnit2();
      sm1=new StopMsg1();
      rcp2=new Receipt2();
}

  @Override
  public DataStore getdata() {
        // TODO Auto-generated method stub
        return db2;
  }

  @Override
  public StoreData getdatastore() {
        // TODO Auto-generated method stub
        return ds2;
  }

  @Override
  public PayMsg getpaymsg() {
        // TODO Auto-generated method stub
        return paymsg2;
  }

  @Override
  public SetW get_w() {
        // TODO Auto-generated method stub
        return sw1;
  }


  @Override
  public StoreCash getsc() {
        // TODO Auto-generated method stub
        return sc2;
  }

  @Override
  public SetPrice getpsp() {
        // TODO Auto-generated method stub
        return sp2;
  }
```

```
	@Override
	public SetInitialValue getsv() {
		// TODO Auto-generated method stub
		return sv2;
	}
	@Override
	public DisplayMenu getdm() {
		// TODO Auto-generated method stub
		return dm2;
	}


	@Override
	public PumpGasUnit getpu() {
		// TODO Auto-generated method stub
		return pu2;
	}

	@Override
	public StopMsg getsm() {
		// TODO Auto-generated method stub
		return sm1;
	}

	@Override
	public Receipt getrcp() {
		// TODO Auto-generated method stub
		return rcp2;
	}

	@Override
	public GasPumpedMsg getGpMsg() {
		// TODO Auto-generated method stub
		return gmMsg1;
	}

}
```

Class : CFactory-1

```
/*
 * Concrete factory class for GasPump-3
 */
package Factory;

import Datastore.DataStore3;
import Datastore.DataStore;
import OutputProcesser.DisplayMenu;
import OutputProcesser.DisplayMenu3;
import OutputProcesser.SetW;
import OutputProcesser.SetW1;
import OutputProcesser.StopMsg1;
```

```
import OutputProcesser.StoreData;
import OutputProcesser.StoreData2;
import OutputProcesser.GasPumpedMsg;
import OutputProcesser.GpMsg2;
import OutputProcesser.PumpGasUnit;
import OutputProcesser.PumpGasUnit3;
import OutputProcesser.PayMsg;
import OutputProcesser.Receipt;
import OutputProcesser.Receipt2;
import OutputProcesser.StoreCash;
import OutputProcesser.StoreCash2;
import OutputProcesser.StopMsg;
import OutputProcesser.SetPrice;
import OutputProcesser.SetPrice2;
import OutputProcesser.SetInitialValue;
import OutputProcesser.SetInitialValue2;
import OutputProcesser.PayMsg3;

public class CFactory3 extends AbstractFactory{

            DataStore3 db3;
          StoreData2 ds2;
          PayMsg3 paymsg3;
          GpMsg2 gmMsg2;
            StoreCash2 sc2;
            SetPrice2 sp2;
            SetInitialValue2 sv2;
          SetW1 sw1;
            DisplayMenu3 dm3;
            PumpGasUnit3 pu3;
            StopMsg1 sm1;
            Receipt2 rcp2;
            /*
             * Concrete Factory for GasPump1
             */
            public CFactory3() {
                  db3 =new DataStore3();
                ds2=new StoreData2();
                paymsg3=new PayMsg3();
                gmMsg2 = new GpMsg2();
                sw1=new SetW1();
                sc2=new StoreCash2();
                sp2=new SetPrice2();
                sv2=new SetInitialValue2();
                dm3=new DisplayMenu3();
                pu3=new PumpGasUnit3();
                sm1=new StopMsg1();
                rcp2=new Receipt2();
          }

            @Override
            public DataStore getdata() {
                  // TODO Auto-generated method stub
                  return db3;
            }
```

```
@Override
public StoreData getdatastore() {
	// TODO Auto-generated method stub
	return ds2;
}

@Override
public PayMsg getpaymsg() {
	// TODO Auto-generated method stub
	return paymsg3;
}

@Override
public SetW get_w() {
	// TODO Auto-generated method stub
	return sw1;
}

@Override
public DisplayMenu getdm() {
	// TODO Auto-generated method stub
	return dm3;
}

@Override
public StoreCash getsc() {
	// TODO Auto-generated method stub
	return sc2;
}

@Override
public SetPrice getpsp() {
	// TODO Auto-generated method stub
	return sp2;
}

@Override
public SetInitialValue getsv() {
	// TODO Auto-generated method stub
	return sv2;
}

@Override
public PumpGasUnit getpu() {
	// TODO Auto-generated method stub
	return pu3;
}

@Override
public StopMsg getsm() {
	// TODO Auto-generated method stub
	return sm1;
}
```

```
	@Override
	public Receipt getrcp() {
		// TODO Auto-generated method stub
		return rcp2;
	}

	@Override
	public GasPumpedMsg getGpMsg() {
		// TODO Auto-generated method stub
		return gmMsg2;
	}

}
```

Class : DataStore

```
package Datastore;

import Factory.AbstractFactory;

public abstract class DataStore {

	public static AbstractFactory af;

	public int gettemp_a() {
		return 0;
	};

	public float gettemp_2a() {
		return 0;
	};

	public float gettemp_2b() {
		return 0;
	};

	public void setprice(int price) {
	}

	public void setprice_r(float price) {
	}

	public void setprice_s(float price) {
	}

	public int getprice() {
		return 0;
	}

	public float getprice_r() {
		return 0;
	}

	public float getprice_s() {
		return 0;
	}
```

```
public int gettw() {
	return 0;
}

public int getw() {
	return 0;
}

public void setw(int w) {
}

public int gettemp_c() {
	return 0;
}

public float gettemp_d() {
	return 0;
}

public int getc() {
	return 0;
}

public void setc(int c) {
}

public float getd() {
	return 0;
}

public void setc(float c) {
}

public int getpc() {
	return 0;
}

public float getpcf() {
	return 0;
}

public void setpc(int pc) {
}

public void setpc(float pc) {
}

public int gettemp_G() {
	return 0;
}

public int gettemp_L() {
	return 0;
}
```

```
	public int getG() {
		return 0;
	}

	public int getL() {
		return 0;
	}

	public void setG(int c) {
	}

	public void setL(int c) {
	}

	public int gettotal() {
		return 0;
	}

	public float gettotalf() {
		return 0;
	}

	public void settotal(int t) {
	}

	public void settotal(float t) {
	}

}
```

Class : DataStore-1

```
/**
 * DataStorage for GasPump-1
 */
package Datastore;

public class DataStore1 extends DataStore {

	public int priced;
	public int tempa;
	public int tempw;
	public int wd;
	public int tempc;
	public int cd;
	public int tempG;
	public int Gd;
    public int total;
    public int pricec;
	public int gettemp_a() {
		return tempa;
	}

	public void setprice(int price) {
```

```
            priced=price;
      }
    public int gettw() {
    return tempw;
    }
    public int getw()
    {return wd;}
    public void setw(int w) {
      wd=w;
    }
    public int gettempc(){return tempc;}
    public int getc(){return cd;}
    public void setc(int c){cd=c;}
    public int getpc(){return pricec;}
    public void setpc(int pc){pricec=pc;}
    public int gettemp_G(){return tempG;}
    public int getG(){return Gd;}
    public void setG(int c){Gd=c;}
    public int getprice(){return priced;}
    public int gettotal(){return total;}
    public void settotal(int t){total=t;}

}
```

Class : DataStore-2

```
/**
 * DataStorage for GasPump-2
 */
package Datastore;

public class DataStore2 extends DataStore {


      public float rpriced;
      public float spriced;
      public float tempa;
      public float tempb;
      public int tempw;
      public int wd;
      public int tempc;
      public int cd;
      public int tempG;
      public int Gd;
    public float total;
    public float pricec;
      public float gettemp_2a() {
            return tempa;
      }
      public float gettemp_2b() {
            return tempb;
      }

      public void setprice_r(float price) {
            // TODO Auto-generated method stub
```

```
            rpriced=price;
      }
      public void setprice_s(float price) {
            // TODO Auto-generated method stub
            spriced=price;
      }
    public int gettw()
    {
    return tempw;
    }
    public int getw()
    {return wd;}
    public void setw(int w)
    {
      wd=w;    }
    public int gettempc(){return tempc;}
    public int getc(){return cd;}
    public void setc(int c){cd=c;}
    public float getpcf(){return pricec;}
    public void setpc(float pc){pricec=pc;}
    public int gettemp_G(){return tempG;}
    public int getG(){return Gd;}
    public void setG(int d){Gd=d;}
    public float getprice_r(){return rpriced;}
    public float getprice_s(){return spriced;}
    public float gettotalf(){return total;}
    public void settotal(float t){total=t;}

}
```

Class : DataStore-3

```
/**
 * DataStorage for GasPump-3
 */
package Datastore;

public class DataStore3 extends DataStore {


      public float rpriced;
      public float spriced;
      public float tempa;
      public float tempb;
      public int tempw;
      public int wd;
      public int tempL;
      public int Ld;
    public float total;
      public float tempd;
      public float cd;
    public float pricec;

      public float gettemp_2a() {
            return tempa;
```

```
      }
      public float gettemp_2b() {
            return tempb;
      }
      public float getprice_r(){
      return rpriced;
    }
    public float getprice_s(){
      return spriced;
    }
    public float gettotalf(){
      return total;
    }
    public void settotal(float t){
      total=t;
    }
      public void setprice_r(float price) {
            rpriced=price;
      }
      public void setprice_s(float price) {
            spriced=price;
      }

    public float gettemp_d(){
      return tempd;
    }
    public float getd(){
      return cd;
    }
    public void setc(float c){
      cd=c;
    }
    public float getpcf(){
      return pricec;
    }
    public void setpc(float pc){
      pricec=pc;
    }
    public int gettemp_L(){
      return tempL;
    }
    public int getL(){
      return Ld;
    }
    public void setL(int d){
      Ld=d;
    }


}
```

Class : States

```
/*
 * Abstract Class for all States
```

```
 */
package States;

import OutputProcesser.OP;

public abstract class States {
	public abstract void activate();
	public abstract int get_ID();
	public abstract void start();
	public abstract void payCredit();
	public abstract void reject();
	public abstract void approved();
	public abstract void selectGas(int n);
	public abstract void startPump();
	public abstract void pumpUnit();
	public abstract void stopPump();
	public abstract void receipt();
	public abstract void noReceipt();
	public abstract void cancel();
	public abstract void payCash();
	public OP op=new OP();
}
```

Class: S0

```
/*
 * S0 state class
 */
package States;


public class S0 extends States {
public final int ID=1;
	@Override
	public void activate() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public int get_ID() {
		// TODO Auto-generated method stub
		return ID;
	}

	@Override
	public void start() {
		// TODO Auto-generated method stub
		op.pay_Msg();
	}

	@Override
	public void payCredit() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
```

```
                +"\n Please Select Appropiate operation... ");
    }

    @Override
    public void reject() {
          // TODO Auto-generated method stub
          System.out.println("\n  Invalid State !"
                      +"\n Please Select Appropiate operation... ");
    }

    @Override
    public void approved() {
          // TODO Auto-generated method stub
          System.out.println("\n  Invalid State !"
                      +"\n Please Select Appropiate operation... ");
    }

    @Override
    public void cancel() {
          // TODO Auto-generated method stub
          System.out.println("\n  Invalid State !"
                      +"\n Please Select Appropiate operation... ");
    }

    @Override
    public void payCash() {
          // TODO Auto-generated method stub
          System.out.println("\n  Invalid State !"
                      +"\n Please Select Appropiate operation... ");
    }


    @Override
    public void startPump() {
          // TODO Auto-generated method stub
          System.out.println("\n  Invalid State !"
                      +"\n Please Select Appropiate operation... ");
    }

    @Override
    public void selectGas(int n) {
          // TODO Auto-generated method stub
          System.out.println("\n  Invalid State !"
                      +"\n Please Select Appropiate operation... ");
    }

    @Override
    public void pumpUnit() {
          // TODO Auto-generated method stub
          System.out.println("\n  Invalid State !"
                      +"\n Please Select Appropiate operation... ");
    }

    @Override
    public void stopPump() {
```

```
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void receipt() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void noReceipt() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
}
```

Class:S1

```
/*
 * S1 state class
 */
package States;


public class S1 extends States{
public final int ID=2;
	@Override
	public void activate() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public int get_ID() {
		// TODO Auto-generated method stub
		return ID;
	}

	@Override
	public void start() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void payCredit() {
		// TODO Auto-generated method stub
```

```
		op.pay_credit();

	}

	@Override
	public void reject() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void approved() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void cancel() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void payCash() {
		// TODO Auto-generated method stub
		op.store_cash();
		op.set_w();
		op.display_menu();

	}

	@Override
	public void startPump() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void selectGas(int n) {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void pumpUnit() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
```

```
	@Override
	public void stopPump() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void receipt() {
		// TODO Auto-generated method stub

	}

	@Override
	public void noReceipt() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
}
```

Class : S2

```
/*
 * S2 state class
 */
package States;


public class S2 extends States {
   final public int ID=3;
	@Override
	public void activate() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public int get_ID() {
		// TODO Auto-generated method stub
		return ID;
	}

	@Override
	public void start() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void payCredit() {
		// TODO Auto-generated method stub
```

```
      System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
}

@Override
public void reject() {
      // TODO Auto-generated method stub
   op.reject_msg();
}

@Override
public void approved() {
      // TODO Auto-generated method stub
op.set_w();
op.display_menu();
}

@Override
public void cancel() {
      // TODO Auto-generated method stub
      System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
}

@Override
public void payCash() {
      // TODO Auto-generated method stub
      System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
}



@Override
public void startPump() {
      // TODO Auto-generated method stub
      System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
}

@Override
public void selectGas(int n) {
      // TODO Auto-generated method stub
      System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
}

@Override
public void pumpUnit() {
      // TODO Auto-generated method stub
      System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
}

@Override
```

```
	public void stopPump() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void receipt() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void noReceipt() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
}
```

Class : S3

```
/*
 * S3 state class
 */
package States;


public class S3 extends States {
public final int ID=4;
	@Override
	public void activate() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public int get_ID() {
		// TODO Auto-generated method stub
		return ID;
	}

	@Override
	public void start() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void payCredit() {
		// TODO Auto-generated method stub
```

```
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void reject() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void approved() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void cancel() {
		// TODO Auto-generated method stub
	 op.calcel_msg();
	}

	@Override
	public void payCash() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void startPump() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void selectGas(int n) {
		// TODO Auto-generated method stub
		op.set_price(n);
	}

	@Override
	public void pumpUnit() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void stopPump() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
```

```
                  +"\n Please Select Appropiate operation... ");
      }

      @Override
      public void receipt() {
            // TODO Auto-generated method stub
            System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
      }

      @Override
      public void noReceipt() {
            // TODO Auto-generated method stub
            System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
      }
}
```

Class: S4

```
/*
 * S4 state class
 */
package States;


public class S4 extends States{
public final int ID=5;
      @Override
      public void activate() {
            // TODO Auto-generated method stub
            System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
      }

      @Override
      public int get_ID() {
            // TODO Auto-generated method stub
            return ID;
      }

      @Override
      public void start() {
            // TODO Auto-generated method stub
            System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
      }

      @Override
      public void payCredit() {
            // TODO Auto-generated method stub
            System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
      }
```

```
@Override
public void reject() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void approved() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void cancel() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void payCash() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void selectGas(int n) {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void startPump() {
	// TODO Auto-generated method stub
	op.setini_val();
	op.ready_msg();

}

@Override
public void pumpUnit() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void stopPump() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
```

```
                  +"\n Please Select Appropiate operation... ");
      }

      @Override
      public void receipt() {
            // TODO Auto-generated method stub
            System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
      }

      @Override
      public void noReceipt() {
            // TODO Auto-generated method stub
            System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
      }
}
```

Class : S5

```
/*
 * S5 state class
 */
package States;


public class S5 extends States{
public final int ID=6;
      @Override
      public void activate() {
            // TODO Auto-generated method stub
            System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
      }

      @Override
      public int get_ID() {
            // TODO Auto-generated method stub
            return ID;
      }

      @Override
      public void start() {
            // TODO Auto-generated method stub
            System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
      }

      @Override
      public void payCredit() {
            // TODO Auto-generated method stub
            System.out.println("\n  Invalid State !"
                  +"\n Please Select Appropiate operation... ");
      }
```

```
@Override
public void reject() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void approved() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void cancel() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void payCash() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void selectGas(int n) {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void startPump() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void pumpUnit() {
	// TODO Auto-generated method stub
op.pump_unit();
op.gp_msg();
}

@Override
public void stopPump() {
	// TODO Auto-generated method stub
op.stop_msg();
}
```

```
	@Override
	public void receipt() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void noReceipt() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

}
```

Class : S6

```
/*
 * S6 state class
 */
package States;


public class S6 extends States {

	public final int ID=7;

	@Override
	public void activate() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public int get_ID() {
		// TODO Auto-generated method stub
		return ID;
	}

	@Override
	public void start() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void payCredit() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
```

```
@Override
public void reject() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void approved() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void cancel() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void payCash() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void selectGas(int n) {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void startPump() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void pumpUnit() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void stopPump() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}
```

```
	@Override
	public void receipt() {
		// TODO Auto-generated method stub
	op.print_receipt();

	}

	@Override
	public void noReceipt() {
		// TODO Auto-generated method stub
	 System.out.println("No receipt");

	}

}
```

Class: S6

```
/*
 * S6 state class
 */
package States;


public class S6 extends States {

	public final int ID=7;

	@Override
	public void activate() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public int get_ID() {
		// TODO Auto-generated method stub
		return ID;
	}

	@Override
	public void start() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void payCredit() {
		// TODO Auto-generated method stub
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
```

```
}

@Override
public void reject() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void approved() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void cancel() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void payCash() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void selectGas(int n) {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void startPump() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void pumpUnit() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
			+"\n Please Select Appropiate operation... ");
}

@Override
public void stopPump() {
	// TODO Auto-generated method stub
	System.out.println("\n  Invalid State !"
```

```
                        +"\n Please Select Appropiate operation... ");
      }

      @Override
      public void receipt() {
            // TODO Auto-generated method stub
      op.print_receipt();

      }

      @Override
      public void noReceipt() {
            // TODO Auto-generated method stub
       System.out.println("No receipt");

      }

}
```

Class : Start

```
/*
 * Start state class
 */
package States;


public class start extends States {
 public final int ID=0;
      @Override
      public void activate() {
         //System.out.println("In state start");
            op.storeData();
      }
      @Override
      public int get_ID() {
            return ID;
      }
      @Override
      public void start() {
            System.out.println("\n  Invalid State !"
                        +"\n Please Select Appropiate operation... ");
      }
      @Override
      public void payCredit() {
            System.out.println("\n  Invalid State !"
                                    +"\n Please Select Appropiate operation... ");
      }
      @Override
      public void reject() {
            System.out.println("\n  Invalid State !"
                        +"\n Please Select Appropiate operation... ");
      }
      @Override
```

```
	public void approved() {
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
	@Override
	public void cancel() {
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
	@Override
	public void payCash() {
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}

	@Override
	public void startPump() {
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
	@Override
	public void selectGas(int n) {
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
	@Override
	public void pumpUnit() {
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
	@Override
	public void stopPump() {
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
	@Override
	public void receipt() {
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
	@Override
	public void noReceipt() {
		System.out.println("\n  Invalid State !"
				+"\n Please Select Appropiate operation... ");
	}
}
```

Class : State

```
/*
 * Abstract Class for all States
 */
package States;
```

```
import OutputProcesser.OP;

public abstract class States {
	public abstract void activate();
	public abstract int get_ID();
	public abstract void start();
	public abstract void payCredit();
	public abstract void reject();
	public abstract void approved();
	public abstract void selectGas(int n);
	public abstract void startPump();
	public abstract void pumpUnit();
	public abstract void stopPump();
	public abstract void receipt();
	public abstract void noReceipt();
	public abstract void cancel();
	public abstract void payCash();
	public OP op=new OP();
}
```

Class: OP

```
/*
 * OutputProcessor class to handle all operations
 */
package OutputProcesser;
import Datastore.DataStore;
import Factory.AbstractFactory;

public class OP {
	AbstractFactory af;
	StoreData ds1;
	DataStore db1;
	PayMsg pm;
	GasPumpedMsg gm;
	SetW sw;
	DisplayMenu dm;
	StoreCash sc;
	SetPrice sp;
	SetInitialValue sv;
	PumpGasUnit pu;
	StopMsg sm;
	Receipt rcp;

	public OP() {
		af=DataStore.af;
	}
	public void storeData() {
		af=DataStore.af;
		db1=af.getdata();
		ds1=af.getdatastore();
		ds1.datastore(db1);
	}
	public void set_w() {
		af=DataStore.af;
```

```
      db1=af.getdata();
      sw=af.get_w();
      sw.setW(db1);
}
public void display_menu() {
      af=DataStore.af;
      dm=af.getdm();
      dm.displayMenu();
}
public void store_cash() {
      af=DataStore.af;
  sc=af.getsc();
  db1=af.getdata();
  sc.storeCash(db1);
}
public void set_price(int n) {
      af=DataStore.af;
      sp=af.getpsp();
      db1=af.getdata();
      sp.setPrice(n, db1);
}
public void setini_val() {
      af=DataStore.af;
      sv=af.getsv();
    db1=af.getdata();
    sv.setIniVal(db1);
}
public void pump_unit() {
      af=DataStore.af;
      pu=af.getpu();
      db1=af.getdata();
      pu.pumpGasUnit(db1);
}
/*
 * Print Receipt
 */
public void print_receipt() {
      af=DataStore.af;
      rcp=af.getrcp();
      db1=af.getdata();
      rcp.printReceipt(db1);
}
public void pay_Msg() {
      af=DataStore.af;
    pm=af.getpaymsg();
    pm.payMsg();
}
public void pay_credit() {
      System.out.println("Waiting for approvel.");
}
public void reject_msg() {
      System.out.println("Your Card is Rejected");
}
public void ready_msg() {
      System.out.println("\n Start Pumping..");
```

```
	}
	public void gp_msg() {
	    af=DataStore.af;
	    gm=af.getGpMsg();
	    db1=af.getdata();
	    gm.gasPumpedMsg(db1);;
	}
	public void stop_msg() {
	    af=DataStore.af;
	    sm=af.getsm();
	    sm.stopMsg();
	}
	public void calcel_msg() {
	      System.out.println("\n Cancel Transaction");
	}
}
```

Class: StoreData

```
/*
 * StoreData Abstract Class
 */
package OutputProcesser;



import Datastore.DataStore;

public abstract class StoreData {

	public abstract void datastore(DataStore db1);

}
```

Class: StoreData1

```
/*
 * StoreData for GasPump-1 in DataStorage
 */
package OutputProcesser;


import Datastore.DataStore;

public class StoreData1 extends StoreData {

	/*Store data for GasPump1
	 * (non-Javadoc)
	 * @see OutputProcesser.StoreData#datastore(Datastore.DataStore)
	 */
	public void datastore(DataStore db1) {
	      int temp=db1.gettemp_a();
	      db1.setprice(temp);
	}
}
```

Class:StoreData2

```
/*
 * StoreData for GasPump-2 & 3 in DataStorage
 */
package OutputProcesser;


import Datastore.DataStore;

public class StoreData2 extends StoreData {

	/* Store data for GasPump-1 and 2 for Gas Prices
	 * (non-Javadoc)
	 * @see OutputProcesser.StoreData#datastore(Datastore.DataStore)
	 */
	public void datastore(DataStore db1) {
		float temp1=db1.gettemp_2a();
		float temp2=db1.gettemp_2b();
		db1.setprice_r(temp1);
		db1.setprice_s(temp2);;
	}
}
```

Class: DisplayMenu

```
 /*
 * DisplayMenu Abstract Class
 */
package OutputProcesser;

public abstract class DisplayMenu {
public abstract void displayMenu();
}
```

Class: DisplayMenu1

```
/*
 * DisplayMenu class for GasPump-1
 */
package OutputProcesser;

public class DisplayMenu1 extends DisplayMenu{

	@Override
	/* Display menu for GasPump1
	 * (non-Javadoc)
	 * @see OutputProcesser.DisplayMenu#display()
	 */
	public void displayMenu() {
		System.out.println(" \n Menu : \n 5. Cancel \n 8. StartPump");
	}
```

```
}
```

Class: DisplayMenu2

```
/*
 * DisplayMenu class for GasPump-2
 */
package OutputProcesser;

public class DisplayMenu2 extends DisplayMenu
{

	@Override
	/*Display menu for GasPump2
	 * (non-Javadoc)
	 * @see OutputProcesser.DisplayMenu#display()
	 */
	public void displayMenu() {
		// TODO Auto-generated method stub
		System.out.println(" \n Menu : \n 5. Super \n 6. Regular");
	}


}
```

Class : DisplayMenu3

```
/*
 * DisplayMenu class for GasPump-3
 */
package OutputProcesser;

public class DisplayMenu3 extends DisplayMenu{

	@Override
	/*Display menu for GasPump1
	 * (non-Javadoc)
	 * @see OutputProcesser.DisplayMenu#display()
	 */
	public void displayMenu() {
		// TODO Auto-generated method stub
		System.out.println(" \n Menu : \n 6. Premium \n 5. Regular");
	}


}
```

Class : GasPumpedMsg

```
/*
 * GasPumpedMessage Abstract Class
 */
package OutputProcesser;
```

```
import Datastore.DataStore;

public abstract class GasPumpedMsg {
	public abstract void gasPumpedMsg(DataStore db);
}
```

Class: GpMsg1

```
/*
 * GasPumpedMessage for GasPump-1 & 2 to show total GasPumped in Unites
 */
package OutputProcesser;

import Datastore.DataStore;

public class GpMsg1 extends GasPumpedMsg {

	@Override
	public void gasPumpedMsg(DataStore db) {
		System.out.println("\n"+db.getG()+" Gallon Gas Pumped  "  );
	}
}
```

Class: GpMsg2

```
/*
 * GasPumpedMessage for GasPump-3 to show total GasPumped in Unites
 */
package OutputProcesser;

import Datastore.DataStore;

public class GpMsg2 extends GasPumpedMsg {

	@Override
	public void gasPumpedMsg(DataStore db) {
		System.out.println("\n"+db.getL()+" Liter Gas Pumped  " );
	}
}
```

Class : PayMsg

```
/*
 * Abstract class for Pay Message
 */
package OutputProcesser;

public abstract class PayMsg {
public abstract void payMsg();
}
```

Class : PayMsg1

```
/*
 * PayMessage class for GasPump-1
 */
package OutputProcesser;

public class PayMsg1 extends PayMsg {

	@Override
	/*Give Payment method available for Gaspump1
	 * (non-Javadoc)
	 * @see OutputProcesser.PayMsg#paymsg()
	 */
	public void payMsg() {
		System.out.println("\n Select Payment :\n 7.Cash \n 3.Credit:");
	}
}
```

Class: PayMsg3

```
/*
 * PayMessage class for GasPump-3
 */
package OutputProcesser;

public class PayMsg3 extends PayMsg {

	@Override
	/* Give Payment method available for Gaspump3
	 * (non-Javadoc)
	 * @see OutputProcesser.PayMsg#paymsg()
	 */
	public void payMsg() {
		System.out.println("Please Pay by Cash");
	}
}
```

Class : PayMsg2

```
/*
 * PayMessage class for GasPump-2
 */
package OutputProcesser;

public class PayMsg2 extends PayMsg {

	@Override
	/*Give Payment method available for Gaspump2
	 * (non-Javadoc)
	 * @see OutputProcesser.PayMsg#paymsg()
	 */
	public void payMsg() {
		System.out.println("Please Pay by Credit");
	}
}
```

Class: PumpGasUnit

```
/*
 * Abstract class for PumpGasUnit
 */
package OutputProcesser;

import Datastore.DataStore;

public abstract class PumpGasUnit {

public abstract void pumpGasUnit(DataStore db);

}
```

Class: PumpGasUnit1

```
/*
 * PumpGasUnit class to calculate pumped gas amount for GasPump-1
 */
package OutputProcesser;

import Datastore.DataStore;

public class PumpGasUnit1 extends PumpGasUnit{

	@Override
	/* Calculate G and total for GasPump-1 in integer
	 * (non-Javadoc)
	 * @see OutputProcesser.PumpGasUnit#pumpgasunit(Datastore.DataStore)
	 */
	public void pumpGasUnit(DataStore db) {
	  db.setG(db.getG()+1);
	  db.settotal(db.getpc()*db.getG());
	}
}
```

Class: PumpGasUnit2

```
/*
 * PumpGasUnit class to calculate pumped gas amount for GasPump-2
 */
package OutputProcesser;

import Datastore.DataStore;

public class PumpGasUnit2 extends PumpGasUnit {

	@Override
	/*Calculate total and G for GasPump2 in float
	 * (non-Javadoc)
	 * @see OutputProcesser.PumpGasUnit#pumpgasunit(Datastore.DataStore)
	 */
```

```
	public void pumpGasUnit(DataStore db) {
		db.setG(db.getG()+1);
		db.settotal(db.getpcf()*db.getG());
	}
}
```

Class : PumpGasUNit3

```
/*
 * PumpGasUnit class to calculate pumped gas amount for GasPump-3
 */
package OutputProcesser;

import Datastore.DataStore;

public class PumpGasUnit3 extends PumpGasUnit{

	@Override
	/* Calculate total and L for GasPump3
	 * (non-Javadoc)
	 * @see OutputProcesser.PumpGasUnit#pumpgasunit(Datastore.DataStore)
	 */
	public void pumpGasUnit(DataStore db) {
	  db.setL(db.getL()+1);
	  db.settotal(db.getpcf()*db.getL());
	}
}
```

Class: Receipt

```
/*
 * Abstract Class for Receipt
 */
package OutputProcesser;

import Datastore.DataStore;

public abstract class Receipt {

public abstract void printReceipt(DataStore db);
}
```

Class : Receipt1

```
/*
 * Receipt Class for GasPump-1
 */
package OutputProcesser;

import Datastore.DataStore;

public class Receipt1 extends Receipt {
```

```
	@Override
	/*Print receipt for total value of gas pumped in integer for GasPump1
	 * (non-Javadoc)
	 * @see OutputProcesser.Receipt#printreceipt(Datastore.DataStore)
	 */
	public void printReceipt(DataStore db) {
		// TODO Auto-generated method stub
		System.out.println("\n Receipt : \n Total:  "+db.gettotal());
	}

}
```

Class : Receipt2

```
/*
 * Receipt Class for GasPump-2 & 3
 */
package OutputProcesser;

import Datastore.DataStore;

public class Receipt2 extends Receipt {

	@Override
	/*Print receipt for total value of gas pumped in float for gaspump-2 and 3
	 * (non-Javadoc)
	 * @see OutputProcesser.Receipt#printreceipt(Datastore.DataStore)
	 */
	public void printReceipt(DataStore db) {
		System.out.println("\n Receipt : \n Total:  "+db.gettotalf());
	}

}
```

Class: SetInitValue

```
/*
 * Abstract Class for SetInitialValue
 */
package OutputProcesser;

import Datastore.DataStore;

public abstract class SetInitialValue {

public abstract void setIniVal(DataStore db);
}
```

Class : SetInitialValue1

```
/*
 * SetInitialValue class for GasPUmp-1 & 2 to set G
 */package OutputProcesser;
```

```
import Datastore.DataStore;

public class SetInitialValue1 extends SetInitialValue {

	@Override
	/*Set initial value is equal to 0 for G
	 * (non-Javadoc)
	 * @see OutputProcesser.SetInitialValue#setinvalue(Datastore.DataStore)
	 */
	public void setIniVal(DataStore db) {
		// TODO Auto-generated method stub
		db.setG(db.gettemp_G());
	}
}
```

Class : SetInitialValue2

```
/*
 * SetInitialValue class for GasPUmp-3 to set L
 */
package OutputProcesser;

import Datastore.DataStore;

public class SetInitialValue2 extends SetInitialValue {

	@Override
	/*set initial value equals to 0 for L
	 * (non-Javadoc)
	 * @see OutputProcesser.SetInitialValue#setinvalue(Datastore.DataStore)
	 */
	public void setIniVal(DataStore db) {
		db.setL(db.gettemp_L());
	}

}
```

Class : SetPrice

```
/*
 * SetPrice Abstract Class
 */
package OutputProcesser;

import Datastore.DataStore;

public abstract class SetPrice {
	public abstract void setPrice(int n,DataStore db);
}
```

Class: SetPrice1

```
/*
 * SetPrice Class For GasPump-1
 */
package OutputProcesser;

import Datastore.DataStore;

public class SetPrice1 extends SetPrice{

	@Override
	/*Set price for gas in Gaspump1
	 * (non-Javadoc)
	 * @see OutputProcesser.SetPrice#setprice(int, Datastore.DataStore)
	 */
	public void setPrice(int n, DataStore db) {
		if(n==1) {
			db.setpc(db.getprice());
		}
	}
}
```

Class : SetPrice2

```
/*
 * SetPrice Class For GasPump-2 & 3
 */
package OutputProcesser;

import Datastore.DataStore;

/*
 * Set price of selected gas type for GasPump2 and 3
 */
public class SetPrice2 extends SetPrice {

	@Override
	public void setPrice(int n, DataStore db) {
		if(n==1) {
			db.setpc(db.getprice_r());
		}
		else if(n==2) {
			db.setpc(db.getprice_s());
		}
	}
}
```

Class : SetW

```
/*
 * Abstract Class Set W
 */
package OutputProcesser;
```

```
import Datastore.DataStore;

public abstract class SetW {
	public abstract void setW(DataStore db);
}
```

Class: SetW1

```
package OutputProcesser;

import Datastore.DataStore;

public class SetW1 extends SetW {

	@Override
	public void setW(DataStore db) {
		db.setw(db.gettw());
	}

}
```

Class : StopMsg

```
/*
 * StopMessage Abstract Class
 */
package OutputProcesser;

public abstract class StopMsg {

	public abstract void stopMsg();

}
```

Class: StopMsg1

```
/*
 * StopPump Message class for All GasPumps
 */
package OutputProcesser;

public class StopMsg1 extends StopMsg {

	@Override
	/* Stop message given during StopPump
	 * (non-Javadoc)
	 * @see OutputProcesser.StopMsg#stopmsg()
	 */
	public void stopMsg() {
		System.out.printf("\n Pump is Stopped ");
	}
}
```

Class :StoreCash

```
/*
 * StoeCash Abstract Class
 */
package OutputProcesser;

import Datastore.DataStore;

public abstract class StoreCash {

	public abstract void storeCash(DataStore db);

}
```

Class : StoreCash1

```
/*
 * StoreCash for GasPump-1
 */
package OutputProcesser;

import Datastore.DataStore;


public class StoreCash1 extends StoreCash {

	@Override
	/*Store cash given during cash payment in integer for GasPump-1
	 * (non-Javadoc)
	 * @see OutputProcesser.StoreCash#storecash(Datastore.DataStore)
	 */
	public void storeCash(DataStore db) {
		db.setc(db.gettemp_c());
	}


}
```

Class : StoreCash2

```
/*
 * StoreCash for GasPump-3
 */
package OutputProcesser;

import Datastore.DataStore;


public class StoreCash2 extends StoreCash {

	@Override
	/*Store cash given during cash payment in float for GasPump-3
	 * (non-Javadoc)
```

```
	 * @see OutputProcesser.StoreCash#storecash(Datastore.DataStore)
	 */
	public void storeCash(DataStore db) {
		db.setc(db.gettemp_d());
	}
}
```