2018

CS586 – GAS PUMP

PROJECT

MOUNA GIRI – A20376551

Outline of the Report & Deliverables

1. MDA-EFSM model for the GasPump components

i. A list of meta events for the MDA-EFSM

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

iii. A state diagram of the MDA-EFSM

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

MDA-EFSM Events:

Activate()
Start()
PayType(int t) **//credit: t=1; cash: t=2; debit: t=3**
Reject()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int g) **// Regular: g=1; Super: g=2; Premium: g=3; Diesel: g=4**
Receipt()
NoReceipt()
CorrectPin()
IncorrectPin()
Continue()

MDA-EFSM Actions:

StorePrices	// stores price(s) for the gas from the temporary data store
PayMsg	// displays a type of payment method
StoreCash	// stores cash from the temporary data store
DisplayMenu	// display a menu with a list of selections
RejectMsg	// displays credit card not approved message
SetPrice(int g, int M)	// set the price for the gas identified by g identifier as in SelectGas(int g);
ReadyMsg	// displays the ready for pumping message
SetInitialValues	// set G (or L) and total to 0;
PumpGasUnit	// disposes unit of gas and counts # of units disposed
GasPumpedMsg	// displays the amount of disposed gas
StopMsg	// stop pump message and receipt? msg (optionally)
PrintReceipt	// print a receipt
CancelMsg	// displays a cancellation message
ReturnCash	// returns the remaining cash
WrongPinMsg	// displays incorrect pin message
StorePin	// stores the pin from the temporary data store
EnterPinMsg	// displays a message to enter pin
InitializeData	// set the value of price and cash to 0

Operations of the Input Processor
(GasPump-1)

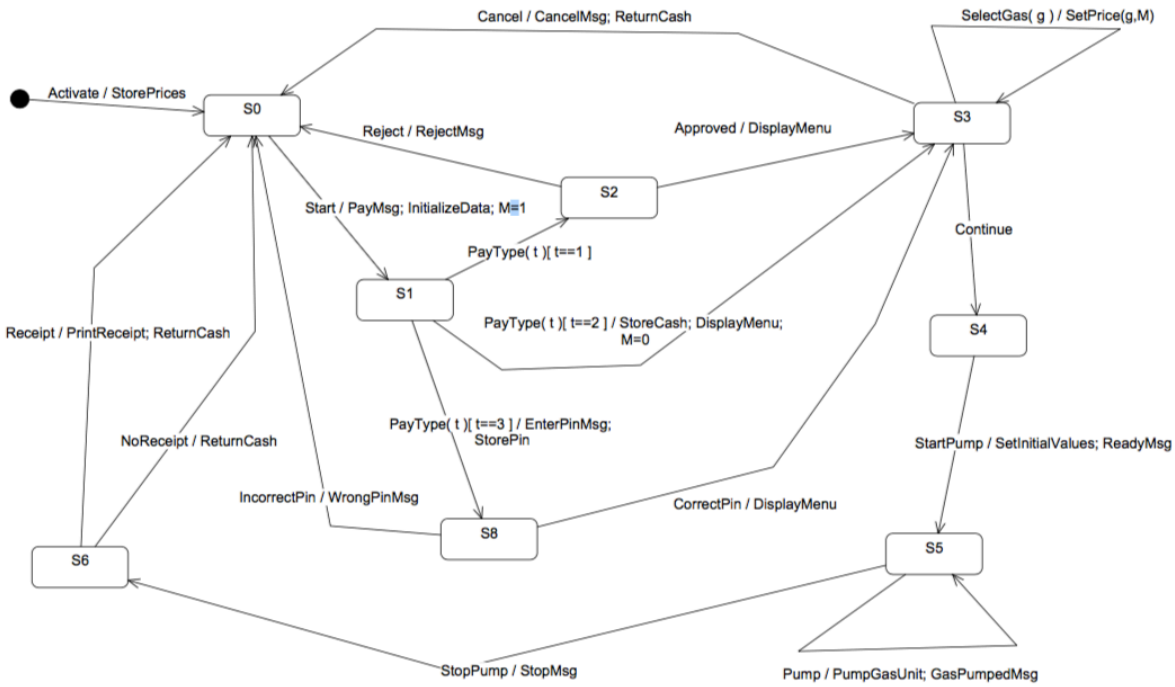
```
Activate(float a, float b) {
        if ((a>0)&&(b>0)) {
            d->temp_a=a;
            d->temp_b=b;
            m->Activate()
        }
}

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

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

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

PayDebit(string p) {
        d->temp_p=p;
        m->PayType(3);
}

Pin(string x) {
        if (d->pin==x) m->CorrectPin()
        else m->InCorrectPin();
}

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

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

Diesel() {
        m->SelectGas(4)
}

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

StartPump() {
        if (d->price>0) {
            m->Continue();
            m->StartPump();
        }
}

PumpGallon() {
        m->Pump();

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

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

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

```
Operations of the Input Processor
        (GasPump-2)
Activate(int a, int b, int c) {
        if ((a>0)&&(b>0)&&(c>0)) {
                d->temp_a=a;
                d->temp_b=b;
                d->temp_c=c
                m->Activate()
        }
}

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

PayCredit() {

        m->start();
        m->PayType(1);
}

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

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

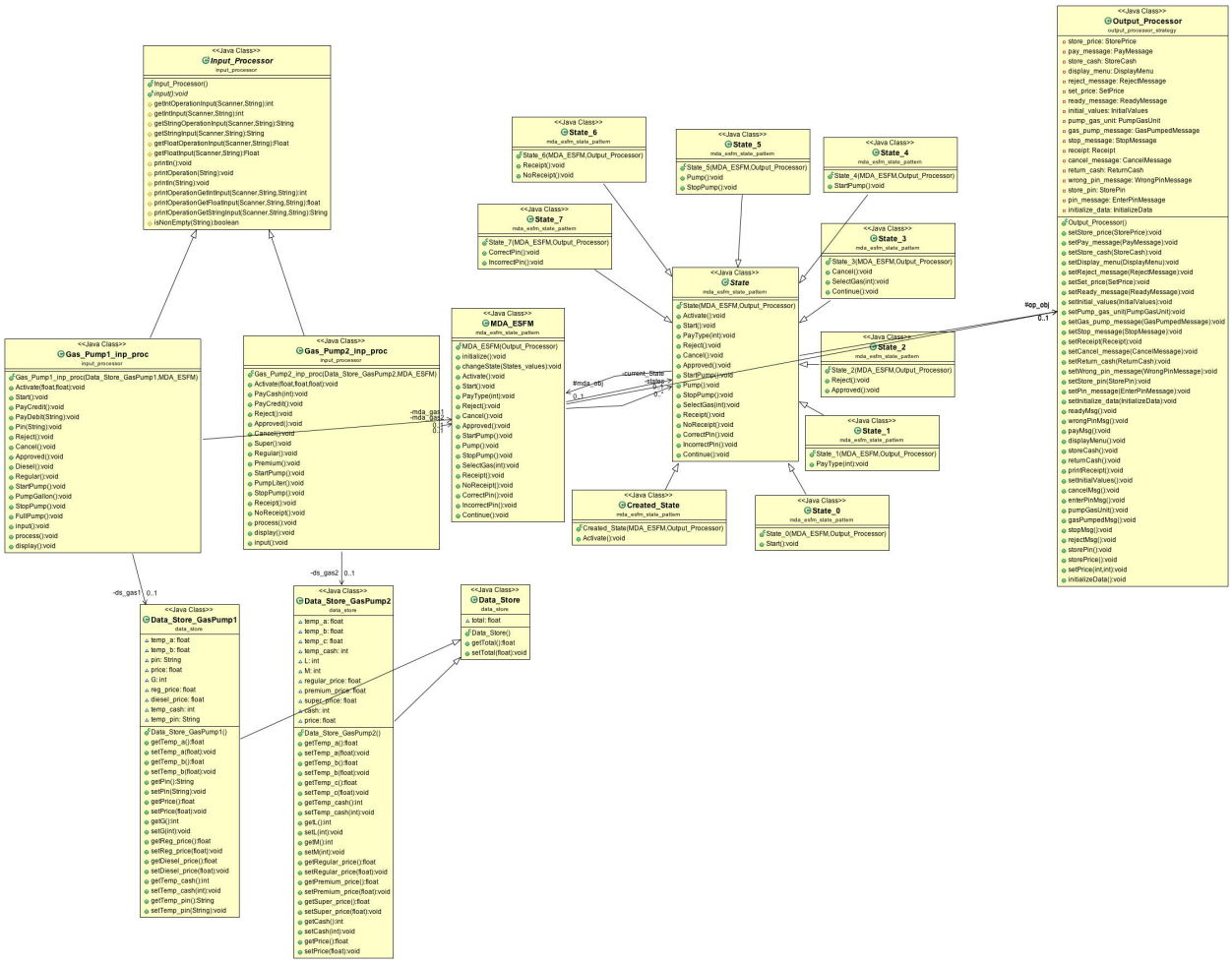
```
Super() {
        m->SelectGas(2);
        m->Continue();
}

Premium() {
        m->SelectGas(3);
        m->Continue();
}

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

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

PumpLiter() {
if (d->cash>0)&&(d->cash < d->price*(d-
>L+1))
                m->StopPump();
else m->Pump()
}

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

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

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



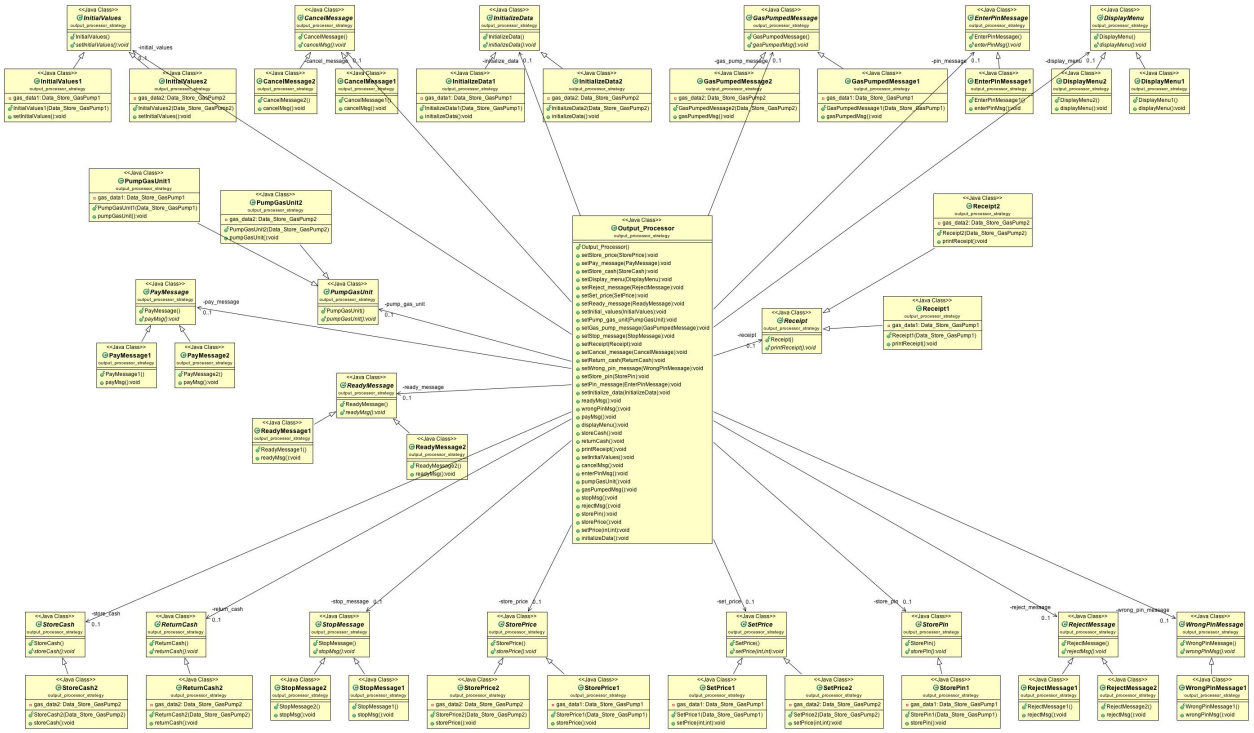
MDA-EFSM for Gas Pumps

2. Class diagram(s) of the MDA of the GasPump components. In your design, you MUST use the following OO design patterns:

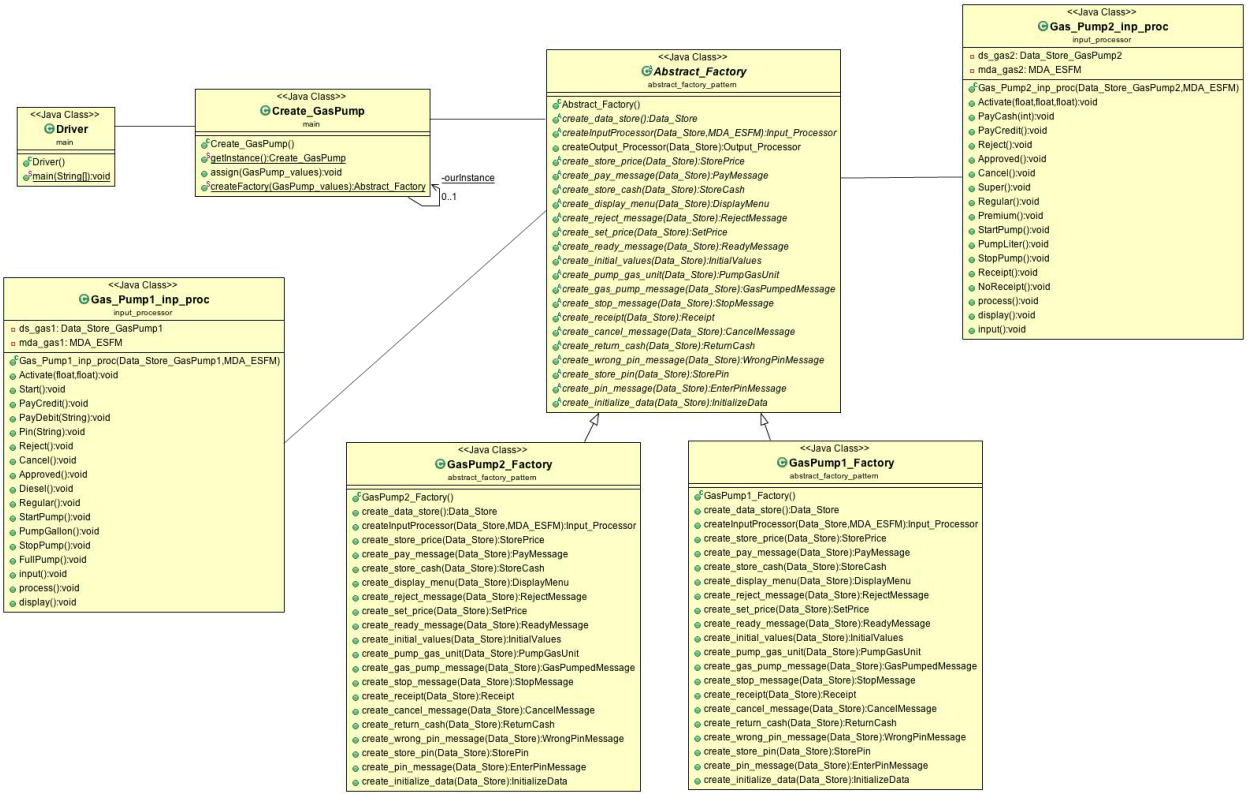
i. State pattern



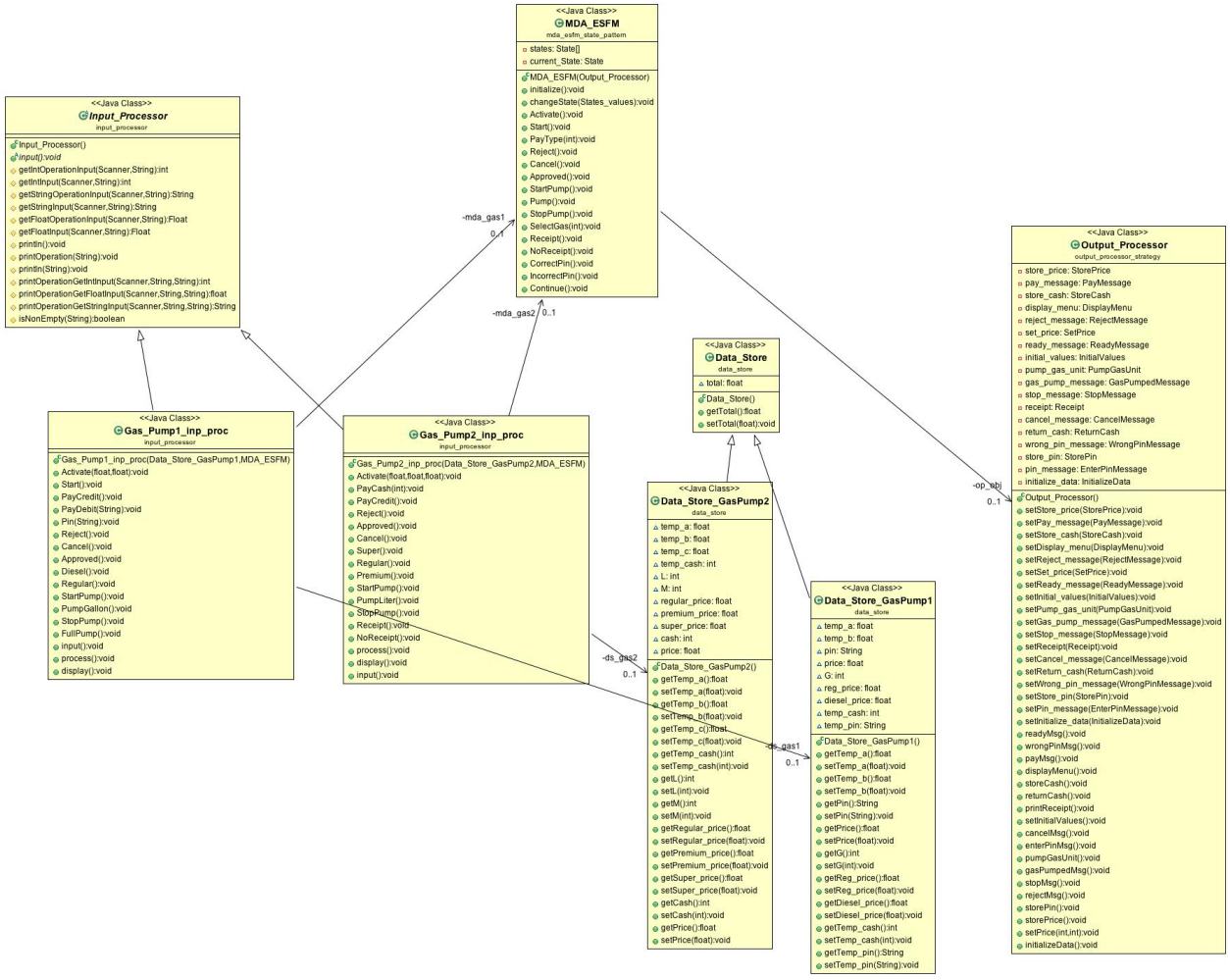
ii. Strategy pattern



iii. Abstract factory pattern

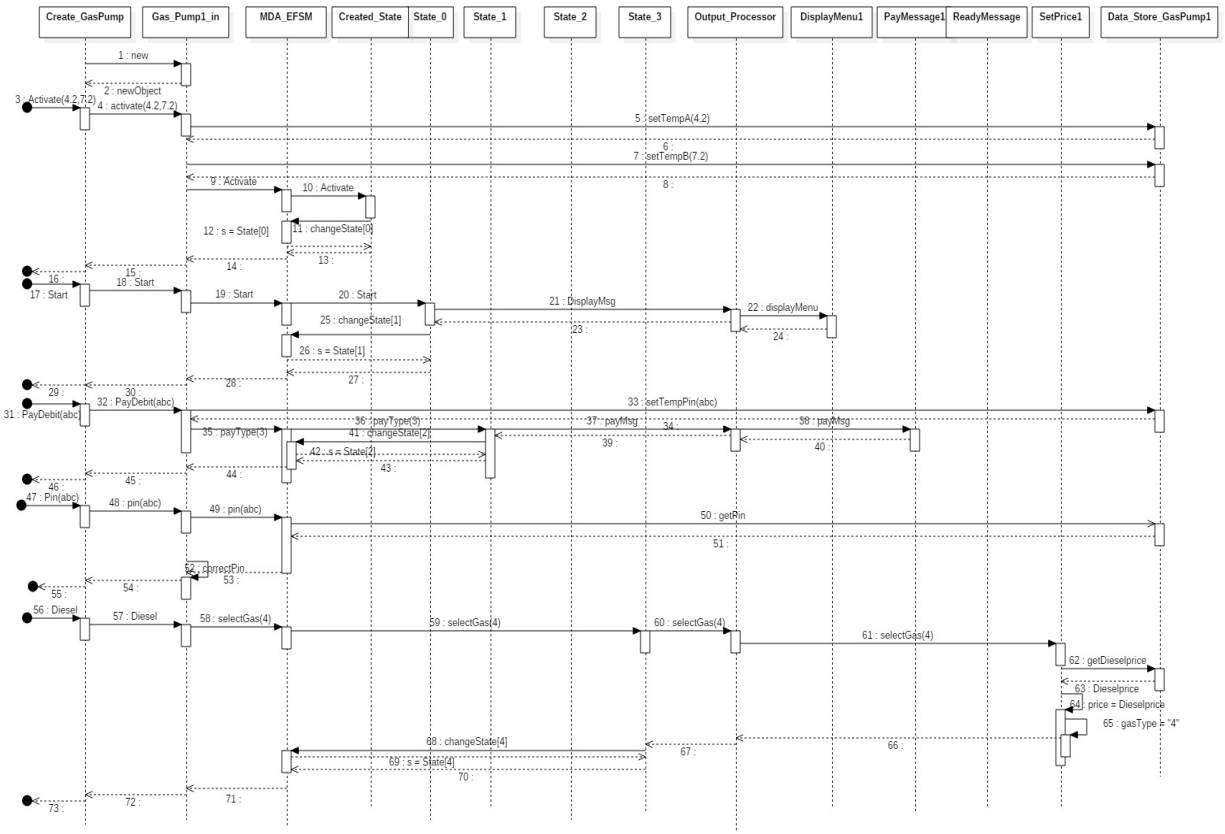


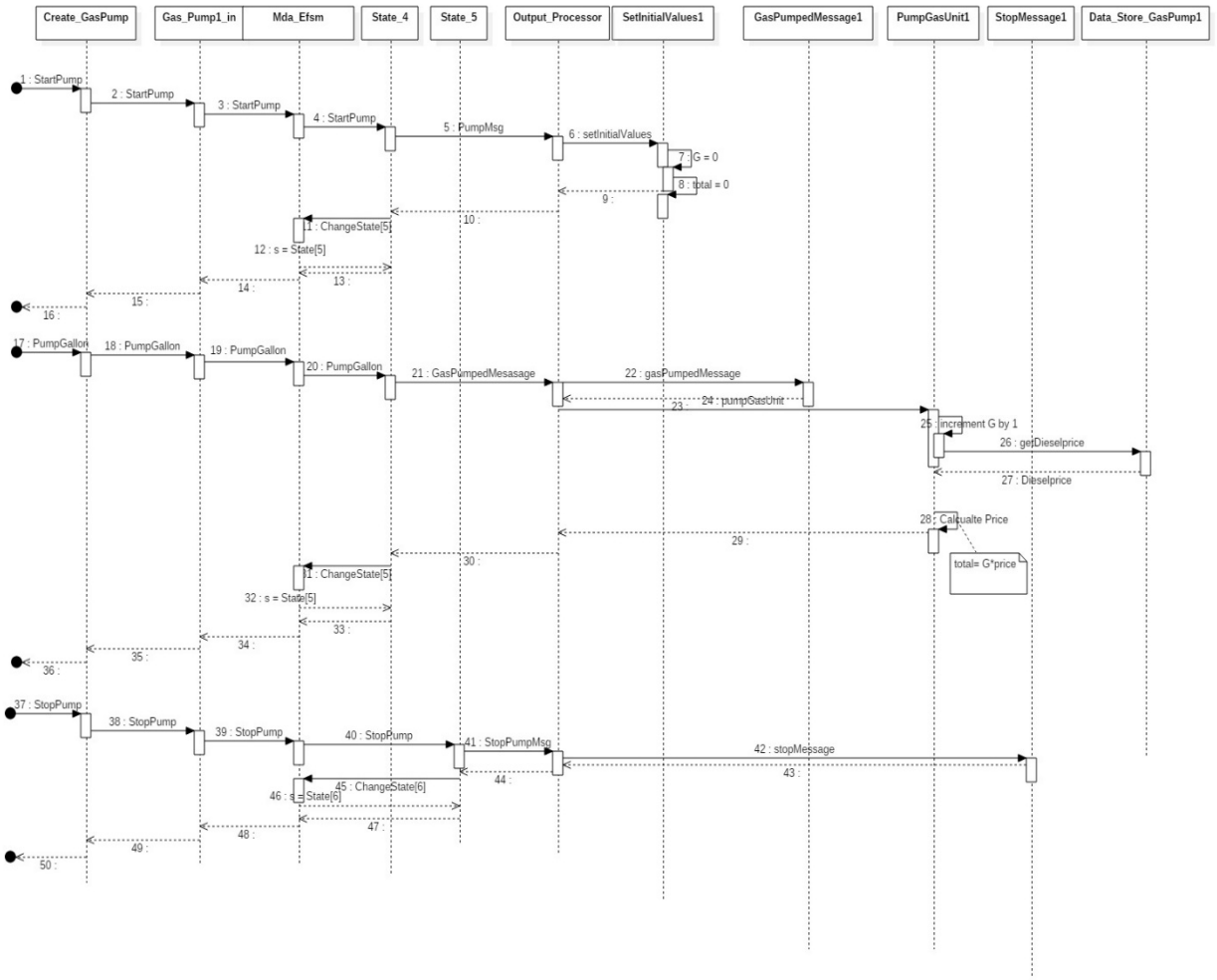
iv. Overview


<<Java Class>>
MDA_ESFM
mda_esfm_state_pattern
states: State[]
current_State: State
MDA_ESFM(Output_Processor)
initialize():void
changeState(States_values):void
Activate():void
Start():void
PayType(int):void
Reject():void
Cancel():void
Approved():void
StartPump():void
Pump():void
StopPump():void
SelectGas(int):void
Receipt():void
NoReceipt():void
CorrectPin():void
IncorrectPin():void
Continue():void
<<Java Class>>
Input_Processor
input_processor
Input_Processor()
input():void
getIntOperationInput(Scanner,String):int
getIntInput(Scanner,String):int
getStringOperationInput(Scanner,String):String
getStringInput(Scanner,String):String
getFloatOperationInput(Scanner,String):Float
getFloatInput(Scanner,String):Float
println():void
printOperation(String):void
println(String):void
printOperationGetIntInput(Scanner,String,String):int
printOperationGetFloatInput(Scanner,String,String):float
printOperationGetStringInput(Scanner,String,String):String
isNonEmpty(String):boolean
-mda_gas1
0..1
-mda_gas2
0..1
<<Java Class>>
Data_Store
data_store
total: float
Data_Store()
getTotal():float
setTotal(float):void
<<Java Class>>
Output_Processor
output_processor_strategy
store_price: StorePrice
pay_message: PayMessage
store_cash: StoreCash
display_menu: DisplayMenu
reject_message: RejectMessage
set_price: SetPrice
ready_message: ReadyMessage
initial_values: InitialValues
pump_gas_unit: PumpGasUnit
gas_pump_message: GasPumpedMessage
stop_message: StopMessage
receipt: Receipt
cancel_message: CancelMessage
return_cash: ReturnCash
wrong_pin_message: WrongPinMessage
store_pin: StorePin
pin_message: EnterPinMessage
initialize_data: InitializeData
Output_Processor()
setStore_price(StorePrice):void
setPay_message(PayMessage):void
setStore_cash(StoreCash):void
setDisplay_menu(DisplayMenu):void
setReject_message(RejectMessage):void
setSet_price(SetPrice):void
setReady_message(ReadyMessage):void
setInitial_values(InitialValues):void
setPump_gas_unit(PumpGasUnit):void
setGas_pump_message(GasPumpedMessage):void
setStop_message(StopMessage):void
setReceipt(Receipt):void
setCancel_message(CancelMessage):void
setReturn_cash(ReturnCash):void
setWrong_pin_message(WrongPinMessage):void
setStore_pin(StorePin):void
setPin_message(EnterPinMessage):void
setInitialize_data(InitializeData):void
readyMsg():void
wrongPinMsg():void
payMsg():void
displayMenu():void
storeCash():void
returnCash():void
printReceipt():void
setInitialValues():void
cancelMsg():void
enterPinMsg():void
pumpGasUnit():void
gasPumpedMsg():void
stopMsg():void
rejectMsg():void
storePin():void
storePrice():void
setPrice(int,int):void
initializeData():void
-op_obj
0..1
<<Java Class>>
Gas_Pump1_inp_proc
input_processor
Gas_Pump1_inp_proc(Data_Store_GasPump1,MDA_ESFM)
Activate(float,float):void
Start():void
PayCredit():void
PayDebit(String):void
Pin(String):void
Reject():void
Cancel():void
Approved():void
Diesel():void
Regular():void
StartPump():void
PumpGallon():void
StopPump():void
FullPump():void
input():void
process():void
display():void
<<Java Class>>
Gas_Pump2_inp_proc
input_processor
Gas_Pump2_inp_proc(Data_Store_GasPump2,MDA_ESFM)
Activate(float,float,float):void
PayCash(int):void
PayCredit():void
Reject():void
Approved():void
Cancel():void
Super():void
Regular():void
Premium():void
StartPump():void
PumpLiter():void
StopPump():void
Receipt():void
NoReceipt():void
process():void
display():void
input():void
-ds_gas2
0..1
<<Java Class>>
Data_Store_GasPump2
data_store
temp_a: float
temp_b: float
temp_c: float
temp_cash: int
L: int
M: int
regular_price: float
premium_price: float
super_price: float
cash: int
price: float
Data_Store_GasPump2()
getTemp_a():float
setTemp_a(float):void
getTemp_b():float
setTemp_b(float):void
getTemp_c():float
setTemp_c(float):void
getTemp_cash():int
setTemp_cash(int):void
getL():int
setL(int):void
getM():int
setM(int):void
getRegular_price():float
setRegular_price(float):void
getPremium_price():float
setPremium_price(float):void
getSuper_price():float
setSuper_price(float):void
getCash():int
setCash(int):void
getPrice():float
setPrice(float):void
-ds_gas1
0..1
<<Java Class>>
Data_Store_GasPump1
data_store
temp_a: float
temp_b: float
pin: String
price: float
G: int
reg_price: float
diesel_price: float
temp_cash: int
temp_pin: String
Data_Store_GasPump1()
getTemp_a():float
setTemp_a(float):void
getTemp_b():float
setTemp_b(float):void
getPin():String
setPin(String):void
getPrice():float
setPrice(float):void
getG():int
setG(int):void
getReg_price():float
setReg_price(float):void
getDiesel_price():float
setDiesel_price(float):void
getTemp_cash():int
setTemp_cash(int):void
getTemp_pin():String
setTemp_pin(String):void

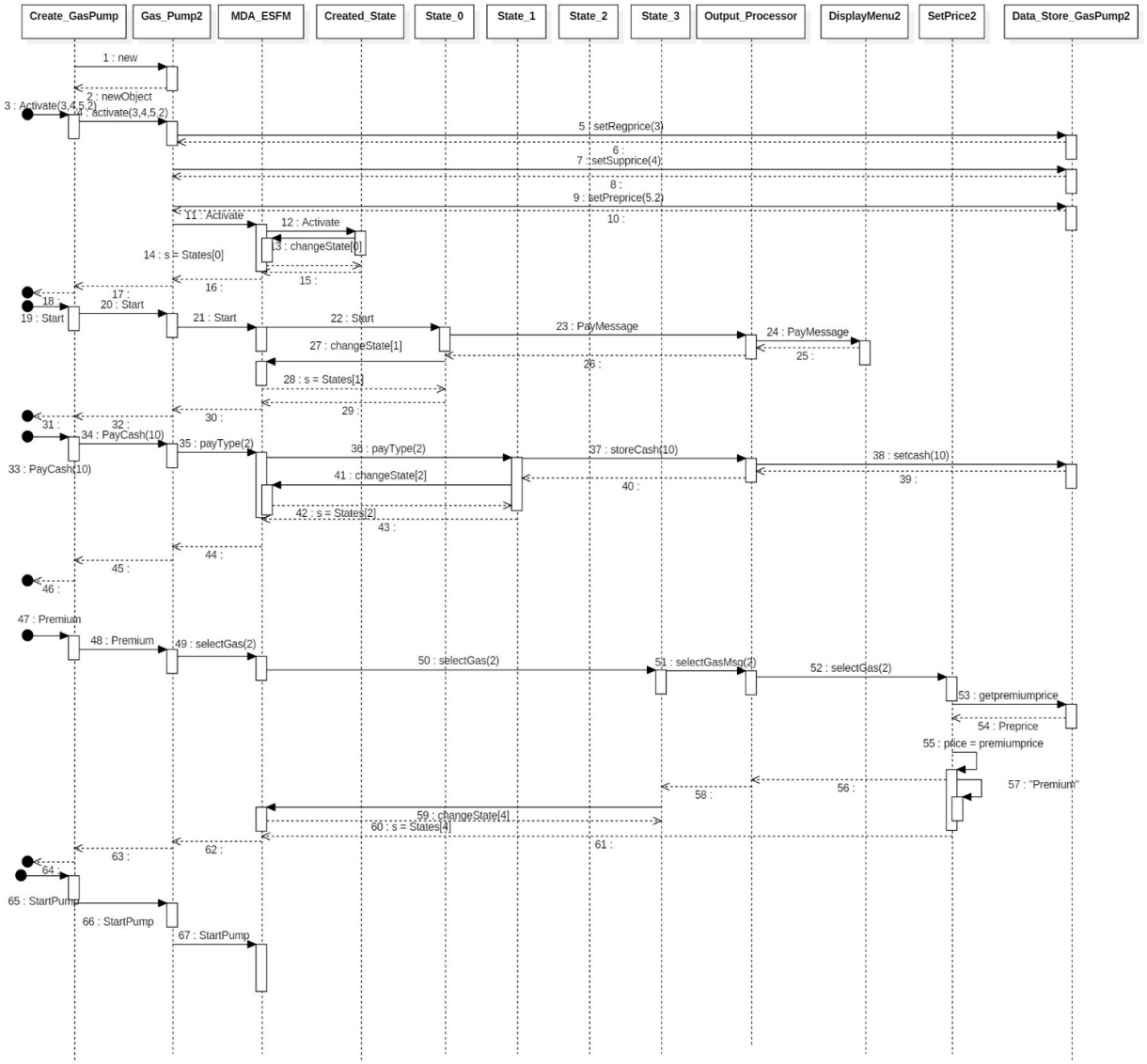
4. Dynamics. Provide two sequence diagrams for two Scenarios:

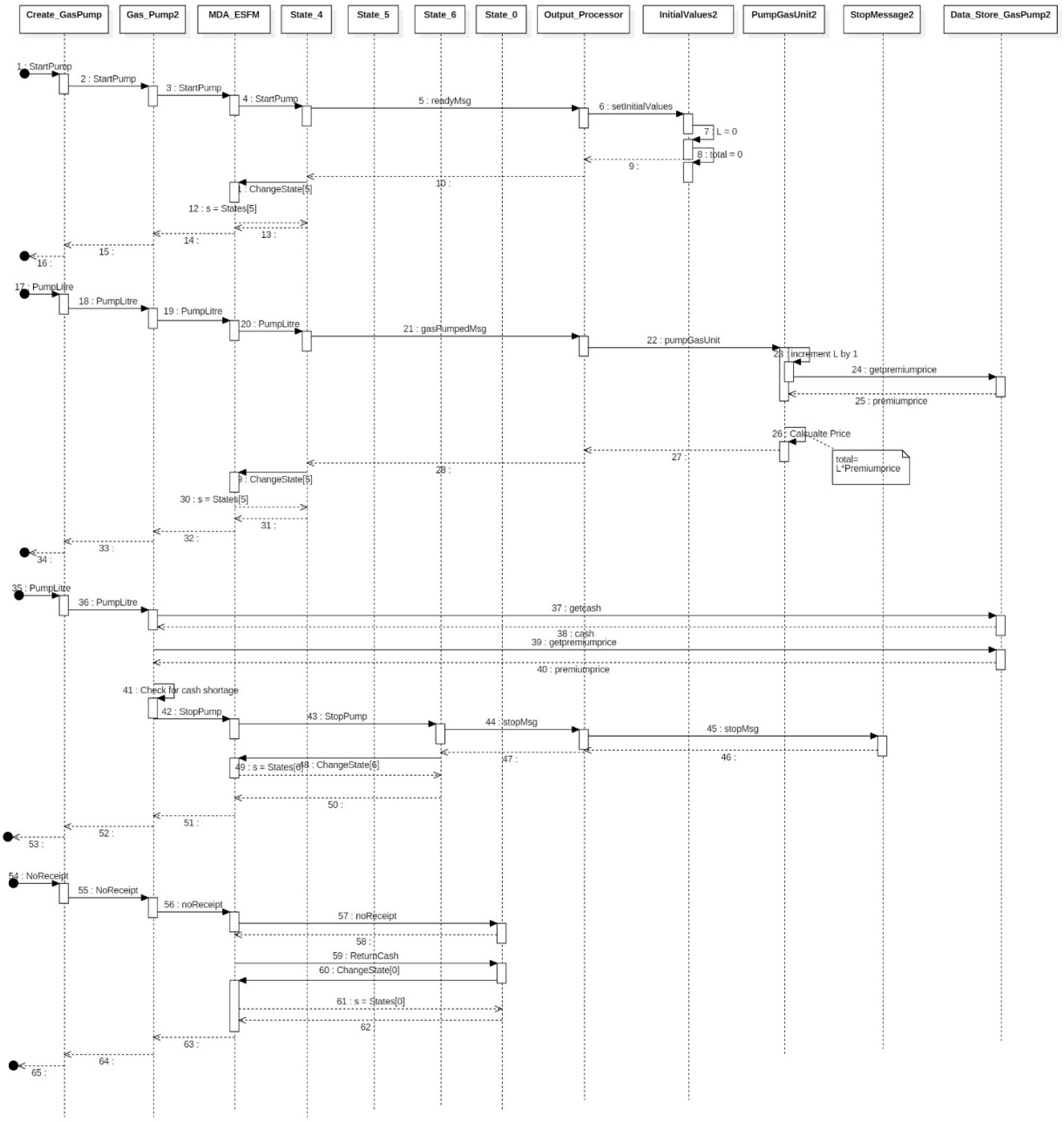
a. Scenario-I should show how one gallon of Diesel gas is disposed in GasPump-1, i.e., the following sequence of operations is issued: Activate(4.2, 7.2), Start(), PayDebit("abc"), Pin("abc"), Diesel(), StartPump(), PumpGallon(), FullTank()




Create_GasPump
Gas_Pump1_in
Mda_Efsm
State_4
State_5
Output_Processor
SetInitialValues1
GasPumpedMessage1
PumpGasUnit1
StopMessage1
Data_Store_GasPump1
1 : StartPump
2 : StartPump
3 : StartPump
4 : StartPump
5 : PumpMsg
6 : setInitialValues
7 : G = 0
8 : total = 0
9 :
10 :
11 : ChangeState[5]
12 : s = State[5]
13 :
14 :
15 :
16 :
17 : PumpGallon
18 : PumpGallon
19 : PumpGallon
20 : PumpGallon
21 : GasPumpedMesasage
22 : gasPumpedMessage
23 :
24 : pumpGasUnit
25 : increment G by 1
26 : getDieselprice
27 : Dieselprice
28 : Calcualte Price
29 :
30 :
total= G*price
31 : ChangeState[5]
32 : s = State[5]
33 :
34 :
35 :
36 :
37 : StopPump
38 : StopPump
39 : StopPump
40 : StopPump
41 : StopPumpMsg
42 : stopMessage
43 :
44 :
45 : ChangeState[6]
46 : s = State[6]
47 :
48 :
49 :
50 :

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





5. Well documented (commented) source code.

In the source-code you should clearly indicate/highlight which parts of the source code are responsible for the implementation of the three required design patterns

1. package abstract_factory_pattern;

1.a Class Abstract_Factory

```
package abstract_factory_pattern;

import input_processor.Input_Processor;
import mda_esfm_state_pattern.MDA_ESFM;
import output_processor_strategy.CancelMessage;
import output_processor_strategy.DisplayMenu;
import output_processor_strategy.EnterPinMessage;
import output_processor_strategy.GasPumpedMessage;
import output_processor_strategy.InitialValues;
import output_processor_strategy.InitializeData;
import output_processor_strategy.Output_Processor;
import output_processor_strategy.PayMessage;
import output_processor_strategy.PumpGasUnit;
import output_processor_strategy.ReadyMessage;
import output_processor_strategy.Receipt;
import output_processor_strategy.RejectMessage;
import output_processor_strategy.ReturnCash;
import output_processor_strategy.SetPrice;
import output_processor_strategy.StopMessage;
import output_processor_strategy.StoreCash;
import output_processor_strategy.StorePin;
import output_processor_strategy.StorePrice;
import output_processor_strategy.WrongPinMessage;
import data_store.Data_Store;

public abstract class Abstract_Factory {

	public abstract Data_Store create_data_store();

	public abstract Input_Processor createInputProcessor(Data_Store data,MDA_ESFM
mda);

	public Output_Processor createOutput_Processor(Data_Store data){

	final Output_Processor op_obj = new Output_Processor();

	op_obj.setStore_price(create_store_price(data));

	op_obj.setPay_message(create_pay_message(data));

	op_obj.setStore_cash(create_store_cash(data));

	op_obj.setDisplay_menu(create_display_menu(data));
```

```
        op_obj.setReject_message(create_reject_message(data));

        op_obj.setSet_price(create_set_price(data));

        op_obj.setReady_message(create_ready_message(data));

        op_obj.setInitial_values(create_initial_values(data));

        op_obj.setPump_gas_unit(create_pump_gas_unit(data));

        op_obj.setGas_pump_message(create_gas_pump_message(data));

        op_obj.setStop_message(create_stop_message(data));

        op_obj.setReceipt(create_receipt(data));

        op_obj.setCancel_message(create_cancel_message(data));

        op_obj.setReturn_cash(create_return_cash(data));

        op_obj.setWrong_pin_message(create_wrong_pin_message(data));

        op_obj.setStore_pin(create_store_pin(data));

        op_obj.setPin_message(create_pin_message(data));

        op_obj.setInitialize_data(create_initialize_data(data));

        return op_obj;
}


    public abstract StorePrice create_store_price(Data_Store data);
    public abstract PayMessage create_pay_message(Data_Store data);
    public abstract StoreCash create_store_cash(Data_Store data);
    public abstract DisplayMenu create_display_menu(Data_Store data);
    public abstract RejectMessage create_reject_message(Data_Store data);
    public abstract SetPrice create_set_price(Data_Store data);
    public abstract ReadyMessage create_ready_message(Data_Store data);
    public abstract InitialValues create_initial_values(Data_Store data);
    public abstract PumpGasUnit create_pump_gas_unit(Data_Store data);
    public abstract GasPumpedMessage create_gas_pump_message(Data_Store data);
    public abstract StopMessage create_stop_message(Data_Store data);
    public abstract Receipt create_receipt(Data_Store data);
```

```
    public abstract CancelMessage create_cancel_message(Data_Store data);
    public abstract ReturnCash create_return_cash(Data_Store data);
    public abstract WrongPinMessage create_wrong_pin_message(Data_Store data);
    public abstract StorePin create_store_pin(Data_Store data);
    public abstract EnterPinMessage create_pin_message(Data_Store data);
    public abstract InitializeData create_initialize_data(Data_Store data);

}
```

1 b. GasPump1_Factory

```
package abstract_factory_pattern;

import input_processor.Gas_Pump1_inp_proc;
import input_processor.Input_Processor;
import mda_esfm_state_pattern.MDA_ESFM;
import output_processor_strategy.CancelMessage;
import output_processor_strategy.CancelMessage1;
import output_processor_strategy.DisplayMenu;
import output_processor_strategy.DisplayMenu1;
import output_processor_strategy.EnterPinMessage;
import output_processor_strategy.EnterPinMessage1;
import output_processor_strategy.GasPumpedMessage;
import output_processor_strategy.GasPumpedMessage1;
import output_processor_strategy.InitialValues;
import output_processor_strategy.InitialValues1;
import output_processor_strategy.InitializeData;
import output_processor_strategy.InitializeData1;
import output_processor_strategy.Output_Processor;
import output_processor_strategy.PayMessage;
import output_processor_strategy.PayMessage1;
import output_processor_strategy.PumpGasUnit;
import output_processor_strategy.PumpGasUnit1;
import output_processor_strategy.ReadyMessage;
import output_processor_strategy.ReadyMessage1;
import output_processor_strategy.Receipt;
import output_processor_strategy.Receipt1;
import output_processor_strategy.RejectMessage;
import output_processor_strategy.RejectMessage1;
import output_processor_strategy.ReturnCash;
import output_processor_strategy.SetPrice;
import output_processor_strategy.SetPrice1;
import output_processor_strategy.StopMessage;
import output_processor_strategy.StopMessage1;
```

```
import output_processor_strategy.StoreCash;
import output_processor_strategy.StorePin;
import output_processor_strategy.StorePin1;
import output_processor_strategy.StorePrice;
import output_processor_strategy.WrongPinMessage1;

import output_processor_strategy.StorePrice1;
import output_processor_strategy.WrongPinMessage;
import data_store.Data_Store;
import data_store.Data_Store_GasPump1;

public class GasPump1_Factory extends Abstract_Factory{

	@Override
	public Data_Store create_data_store() {

		return new Data_Store_GasPump1();
	}

	@Override
	public Input_Processor createInputProcessor(Data_Store data, MDA_ESFM mda) {

		return new Gas_Pump1_inp_proc((Data_Store_GasPump1)data, mda);
	}

	@Override
	public StorePrice create_store_price(Data_Store data) {

		return new StorePrice1((Data_Store_GasPump1)data);
	}

	@Override
	public PayMessage create_pay_message(Data_Store data) {

		return new PayMessage1();
	}

	@Override
	public StoreCash create_store_cash(Data_Store data) {

		return null;
	}

	@Override
```

```
public DisplayMenu create_display_menu(Data_Store data) {

        return new DisplayMenu1();
}

@Override
public RejectMessage create_reject_message(Data_Store data) {

        return new RejectMessage1();
}

@Override
public SetPrice create_set_price(Data_Store data) {

        return new SetPrice1((Data_Store_GasPump1)data);
}

@Override
public ReadyMessage create_ready_message(Data_Store data) {

        return new ReadyMessage1();
}

@Override
public InitialValues create_initial_values(Data_Store data) {

        return new InitialValues1((Data_Store_GasPump1)data);
}

@Override
public PumpGasUnit create_pump_gas_unit(Data_Store data) {

        return new PumpGasUnit1((Data_Store_GasPump1)data);
}

@Override
public GasPumpedMessage create_gas_pump_message(Data_Store data) {

        return new GasPumpedMessage1((Data_Store_GasPump1)data);
}

@Override
public StopMessage create_stop_message(Data_Store data) {
```

```
        return new StopMessage1();
    }

    @Override
    public Receipt create_receipt(Data_Store data) {

        return new Receipt1((Data_Store_GasPump1)data);
    }

    @Override
    public CancelMessage create_cancel_message(Data_Store data) {

        return new CancelMessage1();
    }

    @Override
    public ReturnCash create_return_cash(Data_Store data) {

        return null;
    }

    @Override
    public WrongPinMessage create_wrong_pin_message(Data_Store data) {

        return new WrongPinMessage1();
    }

    @Override
    public StorePin create_store_pin(Data_Store data) {

        return new StorePin1((Data_Store_GasPump1)data);
    }

    @Override
    public EnterPinMessage create_pin_message(Data_Store data) {

        return new EnterPinMessage1();
    }

    @Override
    public InitializeData create_initialize_data(Data_Store data) {

        return new InitializeData1((Data_Store_GasPump1)data);
    }
```

}

1.c GasPump2_Factory

```
package abstract_factory_pattern;

import input_processor.Gas_Pump2_inp_proc;
import input_processor.Input_Processor;
import mda_esfm_state_pattern.MDA_ESFM;
import output_processor_strategy.CancelMessage;
import output_processor_strategy.CancelMessage2;
import output_processor_strategy.DisplayMenu;
import output_processor_strategy.DisplayMenu2;
import output_processor_strategy.EnterPinMessage;

import output_processor_strategy.GasPumpedMessage;
import output_processor_strategy.GasPumpedMessage2;
import output_processor_strategy.InitialValues;
import output_processor_strategy.InitialValues2;
import output_processor_strategy.InitializeData;
import output_processor_strategy.InitializeData2;
import output_processor_strategy.PayMessage;
import output_processor_strategy.PayMessage2;
import output_processor_strategy.PumpGasUnit;
import output_processor_strategy.PumpGasUnit2;
import output_processor_strategy.ReadyMessage;
import output_processor_strategy.ReadyMessage2;
import output_processor_strategy.Receipt;
import output_processor_strategy.Receipt2;
import output_processor_strategy.RejectMessage;
import output_processor_strategy.RejectMessage2;
import output_processor_strategy.ReturnCash;
import output_processor_strategy.ReturnCash2;
import output_processor_strategy.SetPrice;
import output_processor_strategy.SetPrice2;
import output_processor_strategy.StopMessage;
import output_processor_strategy.StopMessage2;
import output_processor_strategy.StoreCash;
import output_processor_strategy.StoreCash2;
import output_processor_strategy.StorePin;

import output_processor_strategy.StorePrice;
```

```
import output_processor_strategy.StorePrice2;
import output_processor_strategy.WrongPinMessage;

import data_store.Data_Store;
import data_store.Data_Store_GasPump2;


public class GasPump2_Factory extends Abstract_Factory {

	@Override
	public Data_Store create_data_store() {

		return new Data_Store_GasPump2();
	}

	@Override
	public Input_Processor createInputProcessor(Data_Store data, MDA_ESFM mda) {

		return new Gas_Pump2_inp_proc((Data_Store_GasPump2)data, mda);
	}

	@Override
	public StorePrice create_store_price(Data_Store data) {

		return new StorePrice2((Data_Store_GasPump2)data);
	}

	@Override
	public PayMessage create_pay_message(Data_Store data) {

		return new PayMessage2();
	}

	@Override
	public StoreCash create_store_cash(Data_Store data) {

		return new StoreCash2((Data_Store_GasPump2)data);
	}

	@Override
	public DisplayMenu create_display_menu(Data_Store data) {

		return new DisplayMenu2();
	}
```

```
@Override
public RejectMessage create_reject_message(Data_Store data) {

	return new RejectMessage2();
}

@Override
public SetPrice create_set_price(Data_Store data) {

	return new SetPrice2((Data_Store_GasPump2)data);
}

@Override
public ReadyMessage create_ready_message(Data_Store data) {

	return new ReadyMessage2();
}

@Override
public InitialValues create_initial_values(Data_Store data) {

	return new InitialValues2((Data_Store_GasPump2)data);
}

@Override
public PumpGasUnit create_pump_gas_unit(Data_Store data) {

	return new PumpGasUnit2((Data_Store_GasPump2)data);
}

@Override
public GasPumpedMessage create_gas_pump_message(Data_Store data) {

	return new GasPumpedMessage2((Data_Store_GasPump2)data);
}

@Override
public StopMessage create_stop_message(Data_Store data) {

	return new StopMessage2();
}

@Override
```

```
public Receipt create_receipt(Data_Store data) {

        return new Receipt2((Data_Store_GasPump2)data);
}

@Override
public CancelMessage create_cancel_message(Data_Store data) {

        return new CancelMessage2();
}

@Override
public ReturnCash create_return_cash(Data_Store data) {

        return new ReturnCash2((Data_Store_GasPump2)data);
}

@Override
public WrongPinMessage create_wrong_pin_message(Data_Store data) {

        return null;
}

@Override
public StorePin create_store_pin(Data_Store data) {

        return null;
}

@Override
public EnterPinMessage create_pin_message(Data_Store data) {

        return null;
}

@Override
public InitializeData create_initialize_data(Data_Store data) {

        return new InitializeData2((Data_Store_GasPump2)data);
}
```

2. Package Data_Store

2.a . Data_Store_GasPump1

```
package data_store;

public class Data_Store_GasPump1 extends Data_Store {

	float temp_a;
	float temp_b;
	String pin;
	float price;
	int G;
	float reg_price;
	float diesel_price;
	int temp_cash;
	String temp_pin;

	public Data_Store_GasPump1() {
		super();
		total=0;
	}

	public float getTemp_a() {
		return temp_a;
	}

	public void setTemp_a(float temp_a) {
		this.temp_a = temp_a;
	}

	public float getTemp_b() {
		return temp_b;
	}

	public void setTemp_b(float temp_b) {
		this.temp_b = temp_b;
	}

	public String getPin() {
		return pin;
	}

	public void setPin(String pin) {
		this.pin = pin;
	}
```

```
public float getPrice() {
	return price;
}

public void setPrice(float price) {
	this.price = price;
}

public int getG() {
	return G;
}

public void setG(int g) {
	G = g;
}

public float getReg_price() {
	return reg_price;
}

public void setReg_price(float reg_price) {
	this.reg_price = reg_price;
}

public float getDiesel_price() {
	return diesel_price;
}

public void setDiesel_price(float diesel_price) {
	this.diesel_price = diesel_price;
}

public int getTemp_cash() {
	return temp_cash;
}

public void setTemp_cash(int temp_cash) {
	this.temp_cash = temp_cash;
}

public String getTemp_pin() {
	return temp_pin;
}
```

```
	public void setTemp_pin(String temp_pin) {
		this.temp_pin = temp_pin;
	}



}
```

2.b Data_Store_GasPump2

```
package data_store;

public class Data_Store_GasPump2 extends Data_Store {

	float temp_a;
	float temp_b;
	float temp_c;
	int temp_cash;
	int L;
	int M;
	float regular_price;
	float premium_price;
	float super_price;
	int cash;
	float price;

	public Data_Store_GasPump2() {
		super();
		total=0;
	}

	public float getTemp_a() {
		return temp_a;
	}

	public void setTemp_a(float temp_a) {
		this.temp_a = temp_a;
	}

	public float getTemp_b() {
		return temp_b;
	}

	public void setTemp_b(float temp_b) {
```

```
        this.temp_b = temp_b;
    }

    public float getTemp_c() {
        return temp_c;
    }

    public void setTemp_c(float temp_c) {
        this.temp_c = temp_c;
    }

    public int getTemp_cash() {
        return temp_cash;
    }

    public void setTemp_cash(int temp_cash) {
        this.temp_cash = temp_cash;
    }

    public int getL() {
        return L;
    }

    public void setL(int l) {
        L = l;
    }

    public int getM() {
        return M;
    }

    public void setM(int m) {
        M = m;
    }

    public float getRegular_price() {
        return regular_price;
    }

    public void setRegular_price(float regular_price) {
        this.regular_price = regular_price;
    }

    public float getPremium_price() {
```

```
		return premium_price;
	}

	public void setPremium_price(float premium_price) {
		this.premium_price = premium_price;
	}

	public float getSuper_price() {
		return super_price;
	}

	public void setSuper_price(float super_price) {
		this.super_price = super_price;
	}

	public int getCash() {
		return cash;
	}

	public void setCash(int cash) {
		this.cash = cash;
	}

	public float getPrice() {
		return price;
	}

	public void setPrice(float price) {
		this.price = price;
	}



}
```

2.c Data_Store

```
package data_store;

public class Data_Store {

  float total;
```

```
public float getTotal() {
	return total;
}

public void setTotal(float total) {
	this.total = total;
}



}
```

3. Package fixed_values

3.a GasPump_values

```
package fixed_values;

public enum GasPump_values {

	GasPump1(1),
	GasPump2(2);

	private int num;

	private GasPump_values(int num){
		this.num=num;
	}

	public int getNum(){
		return num;
	}

}
```

3.b Operation_values

```
package fixed_values;

public enum Operation_values {
```

```
        Activate(0),
        Start(1),
        PayCredit(2),
        PayDebit(3),
        PayCash(4),
        Reject(5),
        Cancel(6),
        Approved(7),
        StartPump(8),
        Pump(9),
        StopPump(10),
        Regular(11),
        Super(12),
        Premium(13),
        Diesel(14),
        Receipt(15),
        NoReceipt(16),
        CorrectPin(17),
        IncorrectPin(18),
        Continue(19),
        Pin(20),
        Exit(21);

        private int event;

        private Operation_values(int event){
                this.event=event;
        }

        public static Operation_values value(int code){
                final Operation_values[] val = Operation_values.class.getEnumConstants();
                for(Operation_values op_val:val){
                        if(op_val.event==code)
                                return op_val;
                }
                return null;
        }

}
```

3 C States_values

```
package fixed_values;
```

```
public enum States_values {
	Created_State(8),
	S0(0),
	S1(1),
	S2(2),
	S3(3),
	S4(4),
	S5(5),
	S6(6),
	S7(7);

	private int state_val;

	private States_values(int state_val){
		this.state_val=state_val;
	}
	public int getState_val(){
		return state_val;
	}

	public static int total(){
		return States_values.class.getEnumConstants().length;
	}
}
```

4. Input_processor

4a Gas_Pump1_inp_proc

```
	package input_processor;

	import java.util.Scanner;

	import abstract_factory_pattern.GasPump1_Factory;
	import mda_esfm_state_pattern.MDA_ESFM;
	import data_store.Data_Store_GasPump1;
	import fixed_values.Operation_values;

	public class Gas_Pump1_inp_proc extends Input_Processor {

		private Data_Store_GasPump1 ds_gas1;
```

```
        private MDA_ESFM mda_gas1;

         public Gas_Pump1_inp_proc(Data_Store_GasPump1 ds_gas1,  MDA_ESFM
mda_gas1) {
              this.mda_gas1 = mda_gas1;
              this.ds_gas1 = ds_gas1;
           }

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

        ds_gas1.setTemp_a(a);
        ds_gas1.setTemp_b(b);
        mda_gas1.Activate();
     } else {
        System.out.println("Invalid Input");
     }
  }

 public void Start()
 {
        mda_gas1.Start();

 }

 public void PayCredit(){

                mda_gas1.PayType(1);


 }

 public void PayDebit(String p){

                ds_gas1.setTemp_pin(p);
                mda_gas1.PayType(3);

}


 public void Pin(String x) {
        String pin = ds_gas1.getPin();

 if (pin.equals(x))
```

```
            mda_gas1.CorrectPin();
    else
            mda_gas1.IncorrectPin();
    }

    public void Reject(){
            mda_gas1.Reject();
    }

    public void Cancel(){
            mda_gas1.Cancel();
    }

    public void Approved(){
            mda_gas1.Approved();
    }

    public void Diesel(){
            mda_gas1.SelectGas(4);
    }

    public void Regular(){
                    mda_gas1.SelectGas(1);

    }

    public void StartPump(){
            float price = ds_gas1.getPrice();

            if (price > 0)
            {

                    mda_gas1.Continue();
                    mda_gas1.StartPump();
            }
    }

    public void PumpGallon(){
            mda_gas1.Pump();
    }

    public void StopPump(){
            mda_gas1.StopPump();
            mda_gas1.Receipt();
```

```
}

public void FullPump(){
        mda_gas1.StopPump();
        mda_gas1.Receipt();
}

@Override
public void input() {
        process();
}

public void process(){
        boolean flag = true;
         Operation_values opCode = null;



         while(flag){

                display();
                Scanner in = new Scanner(System.in);


                try{
                        int option = getIntInput(in, null);
                        opCode = Operation_values.value(option);

                        switch(opCode){

                        case Activate:
                          float a = printOperationGetFloatInput(in, "Activate(a,b)",
"a(RegularGas Price)):");
         float b = getFloatOperationInput(in, "b(DieselPrice Gas Price):");
         Activate(a,b);
         break;

                        case Start:
                                printOperation("Start()");
                                mda_gas1.Start();
            break;

                        case PayCredit:
                                printOperation("PayCredit()");
```

```
            PayCredit();
            break;


                        case PayDebit:

                                String pinx = printOperationGetStringInput(in,
"PayDebit(String p)", "pin");

                    PayDebit(pinx);
            break;

                        case Pin:
                                // printOperation("Pin()");
                                String pin_check = printOperationGetStringInput(in,
"Pin(String x)", "Pin:");

                    Pin(pin_check);
            break;

                        case Reject:
            printOperation("Rejected()");

            Reject();
            break;

                        case Cancel:
                                flag=false;
                                printOperation("Cancel()");

            Cancel();

            break;

                        case Approved:
            printOperation("Approved()");
            Approved();
            break;

                        case Diesel:
                                printOperation("Diesel Selected()");

            Diesel();
```

```
            break;

                        case Regular:
                                printOperation("Regular Selected()");

            Regular();
            break;

                        case StartPump:
            printOperation("Start Pump");
            StartPump();
            break;

                        case Pump:
                                printOperation("Pump Gas");
            PumpGallon();
                                break;

                        case StopPump:
                                printOperation("Stop pump");
            StopPump();
                                break;



                        case Exit:
                                flag = false;
             break;


                        default:
            println("Invalid option");
            break;
                        }
                        System.out.println();
                    }catch(Exception e){

                    }
                }
                System.out.println("GasPump 1 Exit");
    }

    public void display() {
```

```
            System.out.println("WELCOME TO GASPUMP1");
            System.out.println("Activate(float,float) - TYPE 0");
            System.out.println("Start() - TYPE 1");
            System.out.println("PayCredit() - TYPE 2");
            System.out.println("Reject() - TYPE 5");
            System.out.println("PayDebit(String p)  - TYPE 3");
            System.out.println("Pin(String x)  - TYPE 20");
            System.out.println("Cancel()  - TYPE 6");
            System.out.println("Approved()  - TYPE 7");
            System.out.println("Diesel() - TYPE 14");
            System.out.println("Regular() - TYPE 11");
            System.out.println("StartPump() - TYPE 8");
            System.out.println("PumpGallon() - TYPE 9");
            System.out.println("StopPump() - TYPE  10");
            System.out.println("FullTank() - TYPE 10");
            System.out.println("Exit  - TYPE 21");

      }


    }
```

4 b Gas_Pump2_inp_proc

```
    package input_processor;

    import java.util.Scanner;

    import mda_esfm_state_pattern.MDA_ESFM;
    import data_store.Data_Store_GasPump1;
    import data_store.Data_Store_GasPump2;
    import fixed_values.Operation_values;

    public class Gas_Pump2_inp_proc extends Input_Processor {

            private Data_Store_GasPump2 ds_gas2;
            private MDA_ESFM mda_gas2;

             public Gas_Pump2_inp_proc(Data_Store_GasPump2 ds_gas2,  MDA_ESFM
    mda_gas2) {
                    this.mda_gas2 = mda_gas2;
                    this.ds_gas2 = ds_gas2;
```

```
            }


         public void Activate(float a,float  b,float c) {
     if ( (a > 0) && (b>0) && (c>0) ){
        ds_gas2.setTemp_a(a);
        ds_gas2.setTemp_b(b);
        ds_gas2.setTemp_c(c);
        mda_gas2.Activate();
     } else {
        System.out.println("Invalid Input");
     }
  }


public void PayCash(int d){

          if(d>0)
          {
                    ds_gas2.setTemp_cash(d);
                    mda_gas2.Start();
                    mda_gas2.PayType(2);
          }
          else{
        System.out.println("Invalid Input");
     }
}

public void  PayCredit()
{
          mda_gas2.Start();
          mda_gas2.PayType(1);
          }

public void Reject(){

          mda_gas2.Reject();
}

public void Approved() {
          mda_gas2.Approved();
}

public void Cancel(){
```

```
            mda_gas2.Cancel();
    }


    public void Super(){
            mda_gas2.SelectGas(3);
            mda_gas2.Continue();
    }

    public void Regular(){
                    mda_gas2.SelectGas(1);
                    mda_gas2.Continue();
    }

    public void Premium(){
            mda_gas2.SelectGas(2);
            mda_gas2.Continue();
   }

    public void StartPump(){

                    mda_gas2.StartPump();

    }

    public void PumpLiter(){

                    float cash = ds_gas2.getCash();
            float price =ds_gas2.getPrice();
            float L = ds_gas2.getL();

            if((cash >0) && (cash < (L+1) *price))
            {
                    System.out.println("Cash isnt sufficient !");
                    mda_gas2.StopPump();
            }
            else
            {
                    mda_gas2.Pump();
            }

    }
```

```
public void StopPump(){
        mda_gas2.StopPump();

}

public void Receipt(){
        mda_gas2.Receipt();
}

public void NoReceipt(){

        mda_gas2.NoReceipt();
}

public void process(){

        boolean flag = true;
        Operation_values opCode = null;


        while(flag){


                display();
                Scanner in = new Scanner(System.in);

                try{
                        int option = getIntInput(in, null);
                        opCode = Operation_values.value(option);

                        switch(opCode){

                        case Activate:
                                float a = printOperationGetFloatInput(in, "Activate(a,b,c)",
"a(RegularGas Price)):");
            float b = getFloatOperationInput(in, "b(PremiumGas Price):");
            float c = getFloatOperationInput(in, "c(SuperGas Price):");


        Activate(a,b,c);
        break;
```

```
                case PayCash:
                        int g = getIntOperationInput(in, "g(Enter cash):");
                        printOperation("PaybyCash()");

PayCash(g);
break;

                case PayCredit:
                        printOperation("PayCredit()");

PayCredit();
break;

                case Reject:
        printOperation("Rejected()");

        Reject();
        break;

                case Cancel:
                        flag=false;
                        printOperation("Cancel()");

Cancel();
break;


                case Super:
                        printOperation("SuperGas Selected()");
Super();
break;

                case Regular:
                        printOperation("RegularGas Selected()");
Regular();
break;

                case Premium:
                        printOperation("PremiumGas Selected()");

Premium();
break;

                case Approved:
```

```
            printOperation("Approved()");
            Approved();
            break;

                    case StartPump:
printOperation("Start Pump");
StartPump();
break;

                    case Pump:
                            printOperation("Pump Gas");
PumpLiter();
                            break;

                    case StopPump:
                            printOperation("Stop pump");
StopPump();
                            break;

                    case Receipt:
                            printOperation("Receipt Requested");
Receipt();
                            break;

                    case NoReceipt:
                            printOperation("No Receipt");
NoReceipt();
                            break;

                    case Exit:
                            flag = false;
 break;


                    default:
System.out.println("Invalid option");
break;
                    }
                    System.out.println();
            }catch(Exception e){

            }
}
System.out.println("GasPump 2 exiting.");
```

```
    }

    public void display() {
        System.out.println();
        System.out.println(" WELCOME TO GASPUMP 2");
        System.out.println("0. Activate(int,int,int) - TYPE 0");
        System.out.println("1. PayCash()  - TYPE 4");
        System.out.println("2. PayCredit() - TYPE  2");
        System.out.println("3. Approved() - TYPE 7");
        System.out.println("4. Reject()  - TYPE 5");
        System.out.println("5. Cancel()  - TYPE 6");
        System.out.println("6. Premium() - TYPE 13");
        System.out.println("7. Regular()  - TYPE 11");
        System.out.println("8. Super()  - TYPE 12");
        System.out.println("9. StartPump() - TYPE 8");
        System.out.println("10.PumpLiter()  - TYPE 9");
        System.out.println("11.Stop()  - TYPE 10");
        System.out.println("12.Receipt()  - TYPE 15");
        System.out.println("13.NoReceipt()  - TYPE 16");
        System.out.println("14. Exit  - TYPE 21");
      }



    @Override
    public void input() {
            process();
    }
    }
```

4 c . Input_Processor

```
    package input_processor;

    import java.util.InputMismatchException;
    import java.util.Scanner;

    public abstract class Input_Processor {

            public abstract void input();
```

```
protected int getIntOperationInput(Scanner in, String msg) {
   return getIntInput(in, "Enter parameter " + msg);
}

protected int getIntInput(Scanner in, String msg) {
   int p = -1;
   if (msg != null) {
      println(msg);
   }
   try {
      p = in.nextInt();
   } catch (InputMismatchException e) {
      println("Invalid data entered");
   }
   return p;
}

protected String getStringOperationInput(Scanner in, String msg) {
   return getStringInput(in, "Enter parameter " + msg);
}

protected String getStringInput(Scanner in, String msg) {
   String p = null;
   println(msg);
   try {
      p = in.next();
   } catch (InputMismatchException e) {
      println("Invalid data entered");
   }
   return p;
}

protected Float getFloatOperationInput(Scanner in, String msg) {
   return getFloatInput(in, "Enter parameter " + msg);
}

protected Float getFloatInput(Scanner in, String msg) {
   Float p = null;
   println(msg);
   try {
      p = in.nextFloat();
   } catch (InputMismatchException e) {
      println("Invalid data entered");
   }
```

```
        return p;
    }

    protected void println() {
        System.out.println();
    }

    protected void printOperation(String msg) {
        System.out.println("Operation performed " + msg);
    }

    protected void println(String msg) {
        System.out.println(msg);
    }

    protected int printOperationGetIntInput(Scanner in, String operationMsg, String
inputMsg) {
        printOperation(operationMsg);
        return getIntOperationInput(in, inputMsg);
    }

    protected float printOperationGetFloatInput(Scanner in, String operationMsg, String
inputMsg) {
        printOperation(operationMsg);
        return getFloatOperationInput(in, inputMsg);
    }

    protected String printOperationGetStringInput(Scanner in, String operationMsg, String
inputMsg) {
        printOperation(operationMsg);
        return getStringOperationInput(in, inputMsg);
    }

    protected boolean isNonEmpty(String p) {
        return p != null && !p.isEmpty();
    }

}
```

5. main package

   5a Create_GasPump

   ```
   package main;
   ```

```
import input_processor.Input_Processor;
import output_processor_strategy.Output_Processor;
import mda_esfm_state_pattern.MDA_ESFM;
import data_store.Data_Store;
import fixed_values.GasPump_values;
import abstract_factory_pattern.Abstract_Factory;
import abstract_factory_pattern.GasPump1_Factory;
import abstract_factory_pattern.GasPump2_Factory;

public class Create_GasPump {

        private static Create_GasPump ourInstance = new Create_GasPump();

   public static Create_GasPump getInstance() {
      return ourInstance;
   }

   public void assign(GasPump_values account) {

      final Abstract_Factory accountFactory = createFactory(account);
      final Data_Store data_store = accountFactory.create_data_store();
      Output_Processor op = accountFactory.createOutput_Processor(data_store);
      MDA_ESFM model = new MDA_ESFM(op);
      Input_Processor inputProcessor = accountFactory.createInputProcessor(data_store,
model);
      inputProcessor.input();
   }

   public static Abstract_Factory createFactory(GasPump_values account) {
      if (GasPump_values.GasPump1.equals(account)) {
         return new GasPump1_Factory();
      } else if (GasPump_values.GasPump2.equals(account)) {
         return new GasPump2_Factory();
      } else {
         System.out.println("Invalid parameter");
      }
                  return null;
   }
}
```

5b Driver

```
package main;

import java.util.Scanner;
import fixed_values.GasPump_values;
public class Driver {
	public static void main(String[] args) {

	Create_GasPump create = Create_GasPump.getInstance();

	boolean flag = true;
	while(flag){
		Scanner input = new Scanner(System.in);
		int option= 0;

		System.out.println("WELCOME TO GAS PUMP PROJECT:");
		System.out.println("1. GasPump1");
		System.out.println("2. GasPump2");
		System.out.println("ENTER THE PARAMETER [1 or 2] :");

		try{
			option=input.nextInt();
		}catch(Exception e){
			System.out.println("Invalid parameter");
			continue;
		}
		switch(option){
		case 1:
			create.assign(GasPump_values.GasPump1);
			break;

		case 2:
			create.assign(GasPump_values.GasPump2);
			break;

		default:
			System.out.println("Invalid");
			break;
		}

	}

}}
```

6. package mda_esfm_state_pattern;

6 a Created_State

```
package mda_esfm_state_pattern;

import output_processor_strategy.Output_Processor;
import static fixed_values.States_values.*;

public class Created_State extends State{

	public  Created_State(MDA_ESFM mda_obj, Output_Processor op_obj) {
      super(mda_obj, op_obj);
   }

	public void Activate() {

		op_obj.storePrice();

		mda_obj.changeState(S0);
   }
}
```

6 b MDA_ESFM

```
package mda_esfm_state_pattern;

import fixed_values.States_values;
import static fixed_values.States_values.*;

import output_processor_strategy.Output_Processor;

public class MDA_ESFM {

	private State states[];
	private State current_State;
	private Output_Processor op_obj ;


	public MDA_ESFM(Output_Processor op){
		this.op_obj =  op;
		initialize();
	}
```

```
        public void initialize() {

    states = new State[States_values.total()];

    states[Created_State.getState_val()] = new Created_State(this, op_obj);
    states[S0.getState_val()] = new State_0(this, op_obj);
    states[S1.getState_val()] = new State_1(this, op_obj);
    states[S2.getState_val()] = new State_2(this, op_obj);
    states[S3.getState_val()] = new State_3(this, op_obj);
    states[S4.getState_val()] = new State_4(this, op_obj);
    states[S5.getState_val()] = new State_5(this, op_obj);
    states[S6.getState_val()] = new State_6(this, op_obj);
    states[S7.getState_val()] = new State_7(this, op_obj);
    changeState(Created_State);
}

        public void changeState(States_values state_val) {
    if (state_val != null) {
       current_State = states[state_val.getState_val()];
    }
}

        public void Activate() {

    current_State.Activate();

}

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

public void PayType(int t) {
    current_State.PayType(t);
}

public void Reject() {
    current_State.Reject();
}

public void Cancel() {
    current_State.Cancel();
}
```

```
public void Approved() {
   current_State.Approved();
}

public void StartPump() {
   current_State.StartPump();

}

public void Pump() {
   current_State.Pump();

}

public void StopPump() {
   current_State.StopPump();

}

public void SelectGas(int g) {
   current_State.SelectGas(g);

}

public void Receipt() {
   current_State.Receipt();

}

public void NoReceipt() {
   current_State.NoReceipt();

}

public void CorrectPin() {
   current_State.CorrectPin();
}

public void IncorrectPin() {
   current_State.IncorrectPin();
}

public void Continue() {
```

```
        current_State.Continue();
    }


}
```

6 c State_0

```
package mda_esfm_state_pattern;

import output_processor_strategy.Output_Processor;
import static fixed_values.States_values.*;

public class State_0 extends State{


        public  State_0 (MDA_ESFM mda_obj, Output_Processor op_obj) {
        super(mda_obj, op_obj);
    }

        public void Start() {
                op_obj.payMsg();
        op_obj.initializeData();
                mda_obj.changeState(S1);
    }
}
```

6 d State_1

```
package mda_esfm_state_pattern;

import output_processor_strategy.Output_Processor;
import static fixed_values.States_values.*;

public class State_1 extends State {

        public  State_1 (MDA_ESFM mda_obj, Output_Processor op_obj) {
        super(mda_obj, op_obj);
    }

public void PayType(int t) {

                if(t==1)
```

```
                {
                        mda_obj.changeState(S2);
                }
                else if(t==2)
                {
                        op_obj.displayMenu();
                        op_obj.storeCash();
                        mda_obj.changeState(S3);
                }
                else if(t==3)
                {
                        op_obj.enterPinMsg();
                        op_obj.storePin();
                        mda_obj.changeState(S7);
                }

    }
}
```

6 e State_2

```
package mda_esfm_state_pattern;

import output_processor_strategy.Output_Processor;
import static fixed_values.States_values.*;

public class State_2 extends State{

        public  State_2 (MDA_ESFM mda_obj, Output_Processor op_obj) {
      super(mda_obj, op_obj);
    }

        public void Reject() {
                op_obj.rejectMsg();
            mda_obj.changeState(S0);
    }

        public void Approved() {
                op_obj.displayMenu();
            mda_obj.changeState(S3);
    }

}
```

6 f State_3

```
package mda_esfm_state_pattern;

import output_processor_strategy.Output_Processor;
import static fixed_values.States_values.*;

public class State_3 extends State{

        public  State_3 (MDA_ESFM mda_obj, Output_Processor op_obj) {
      super(mda_obj, op_obj);
   }

        public void Cancel() {
                op_obj.cancelMsg();
                op_obj.returnCash();
           mda_obj.changeState(S0);
   }
         public void SelectGas(int g) {

                 op_obj.setPrice(g,1);
                 //mda_obj.changeState(S3);
           }

         public void Continue() {

                 mda_obj.changeState(S4);
           }
}
```

6g State_4

```
package mda_esfm_state_pattern;

import output_processor_strategy.Output_Processor;
import static fixed_values.States_values.*;

public class State_4 extends State  {

        public  State_4 (MDA_ESFM mda_obj, Output_Processor op_obj) {
      super(mda_obj, op_obj);
   }
```

```
        public void StartPump() {
                op_obj.setInitialValues();
                op_obj.readyMsg();
             mda_obj.changeState(S5);
   }
}
```

6h State_5

```
package mda_esfm_state_pattern;

import output_processor_strategy.Output_Processor;
import static fixed_values.States_values.*;

public class State_5 extends State{
        public  State_5 (MDA_ESFM mda_obj, Output_Processor op_obj) {
      super(mda_obj, op_obj);

   }

        public void Pump() {
                op_obj.pumpGasUnit();
                op_obj.gasPumpedMsg();
      //mda_obj.changeState(S5);

   }

        public void StopPump() {
                op_obj.stopMsg();
      mda_obj.changeState(S6);
   }

}
```

6h State_6

```
package mda_esfm_state_pattern;

import output_processor_strategy.Output_Processor;
import static fixed_values.States_values.*;
```

```
public class State_6 extends State {
        public  State_6 (MDA_ESFM mda_obj, Output_Processor op_obj) {
      super(mda_obj, op_obj);
    }

        public void Receipt() {
                op_obj.printReceipt();
                op_obj.returnCash();
      mda_obj.changeState(S0);
    }

        public void NoReceipt() {
                op_obj.returnCash();
      mda_obj.changeState(S0);
    }
  }
```

6i State_7

```
package mda_esfm_state_pattern;

import output_processor_strategy.Output_Processor;
import static fixed_values.States_values.*;

public class State_7 extends State{

        public  State_7 (MDA_ESFM mda_obj, Output_Processor op_obj) {
    super(mda_obj, op_obj);
  }

        public void CorrectPin() {
                op_obj.displayMenu();
    mda_obj.changeState(S3);
  }

        public void IncorrectPin() {
                op_obj.wrongPinMsg();
    mda_obj.changeState(S0);
  }

}
```

6 j State

```
package mda_esfm_state_pattern;

import output_processor_strategy.Output_Processor;

public abstract class State {

        protected MDA_ESFM mda_obj;

   protected Output_Processor op_obj = new Output_Processor();

   public State(MDA_ESFM mda_obj, Output_Processor op_obj) {
      this.mda_obj = mda_obj;
      this.op_obj = op_obj;
   }

        public void Activate() {

   }

   public void Start() {

   }

   public void PayType(int t) {

   }

   public void Reject() {

   }

   public void Cancel() {

   }

   public void Approved() {

   }

   public void StartPump() {

   }

   public void Pump() {
```

```
    }
    public void StopPump() {
    }
    public void SelectGas(int g) {
    }
    public void Receipt() {
    }
    public void NoReceipt() {
    }
    public void CorrectPin() {
    }
    public void IncorrectPin() {
    }
    public void Continue() {
    }
}
```

7. package output_processor_strategy;

7 a CancelMessage

```
public abstract class CancelMessage {
        public abstract void cancelMsg();
}
```

7 b CancelMessage1

```
package output_processor_strategy;

public class CancelMessage1 extends CancelMessage{

	@Override
	public void cancelMsg() {
		System.out.println("\n Operation is cancelled \n");
	}

}
```

7 c CancelMessage2

```
package output_processor_strategy;

public class CancelMessage2 extends CancelMessage {

	@Override
	public void cancelMsg() {
		System.out.println("\n Operation is cancelled \n");
		}

}
```

7 d DisplayMenu

```
package output_processor_strategy;

public abstract class DisplayMenu {

	public abstract void displayMenu();

}
```

7 e DisplayMenu1

```
package output_processor_strategy;

public class DisplayMenu1 extends DisplayMenu {
```

```
	@Override
	public void displayMenu() {
		System.out.println("\n DISPLAY MENU ");
		 System.out.println("\n REGULAR FUEL ");

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

	}

}
```

7 f DisplayMenu2

```
package output_processor_strategy;

public class DisplayMenu2 extends DisplayMenu{

	@Override
	public void displayMenu() {
		System.out.println("\n DISPLAY MENU");
		 System.out.println("\n SUPER FUEL ");

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

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


	}

}
```

```
package output_processor_strategy;

import data_store.Data_Store_GasPump1;

public class PumpGasUnit1 extends PumpGasUnit{
```

```
private  Data_Store_GasPump1 gas_data1;

        public PumpGasUnit1(Data_Store_GasPump1 gas_data1) {
      this.gas_data1 = gas_data1;
   }

        @Override
        public void pumpGasUnit() {

                int g=gas_data1.getG();
            g=g+1;
            float total;
            float price = gas_data1.getPrice();
            total = price * g;
            gas_data1.setG(g);
            gas_data1.setTotal(total);

        }

}

package output_processor_strategy;

public abstract class PayMessage {

        public abstract void payMsg();

}

package output_processor_strategy;

import mda_esfm_state_pattern.MDA_ESFM;
import abstract_factory_pattern.GasPump1_Factory;
import data_store.Data_Store_GasPump1;

public class Output_Processor {

   private StorePrice store_price;
   private PayMessage pay_message;
   private StoreCash store_cash;
   private DisplayMenu display_menu;
   private RejectMessage reject_message;
   private SetPrice set_price;
   private ReadyMessage ready_message;
```

```
private InitialValues initial_values;
private PumpGasUnit pump_gas_unit;
private GasPumpedMessage gas_pump_message;
private StopMessage stop_message;
private Receipt receipt;
private CancelMessage cancel_message;
private ReturnCash return_cash;
private WrongPinMessage wrong_pin_message;
private StorePin store_pin;
private EnterPinMessage pin_message;
private InitializeData initialize_data;

	public void setStore_price(StorePrice store_price) {
		this.store_price = store_price;
	}
	public void setPay_message(PayMessage pay_message) {
		this.pay_message = pay_message;
	}
	public void setStore_cash(StoreCash store_cash) {
		this.store_cash = store_cash;
	}
	public void setDisplay_menu(DisplayMenu display_menu) {
		this.display_menu = display_menu;
	}
	public void setReject_message(RejectMessage reject_message) {
		this.reject_message = reject_message;
	}
	public void setSet_price(SetPrice set_price) {
		this.set_price = set_price;
	}
	public void setReady_message(ReadyMessage ready_message) {
		this.ready_message = ready_message;
	}
	public void setInitial_values(InitialValues initial_values) {
		this.initial_values = initial_values;
	}
	public void setPump_gas_unit(PumpGasUnit pump_gas_unit) {
		this.pump_gas_unit = pump_gas_unit;
	}
	public void setGas_pump_message(GasPumpedMessage gas_pump_message) {
		this.gas_pump_message = gas_pump_message;
	}
	public void setStop_message(StopMessage stop_message) {
		this.stop_message = stop_message;
```

```
}
public void setReceipt(Receipt receipt) {
	this.receipt = receipt;
}
public void setCancel_message(CancelMessage cancel_message) {
	this.cancel_message = cancel_message;
}
public void setReturn_cash(ReturnCash return_cash) {
	this.return_cash = return_cash;
}
public void setWrong_pin_message(WrongPinMessage wrong_pin_message) {
	this.wrong_pin_message = wrong_pin_message;
}
public void setStore_pin(StorePin store_pin) {
	this.store_pin = store_pin;
}
public void setPin_message(EnterPinMessage pin_message) {
	this.pin_message = pin_message;
}
public void setInitialize_data(InitializeData initialize_data) {
	this.initialize_data = initialize_data;
}


public void readyMsg() {
	ready_message.readyMsg();

}

public void wrongPinMsg() {
	wrong_pin_message.wrongPinMsg();

}
public void payMsg() {
	pay_message.payMsg() ;

}
public void displayMenu() {
	display_menu.displayMenu();

}
public void storeCash() {
	store_cash.storeCash();
```

```
}
public void returnCash() {
	return_cash.returnCash();

}
public void printReceipt() {
	receipt.printReceipt();

}
public void setInitialValues() {
	initial_values.setInitialValues();

}
public void cancelMsg() {
	cancel_message.cancelMsg();

}
public void enterPinMsg() {
	pin_message.enterPinMsg();

}
public void pumpGasUnit() {
	pump_gas_unit.pumpGasUnit();

}
public void gasPumpedMsg() {
	gas_pump_message.gasPumpedMsg();

}
public void stopMsg() {
	stop_message.stopMsg();
}
public void rejectMsg() {
	reject_message.rejectMsg();

}
public void storePin() {
	store_pin.storePin();

}

public void storePrice() {
	store_price.storePrice();
```

```
	}

	public void setPrice(int g, int M) {
		set_price.setPrice(g,M);

	}
	public void initializeData() {
		initialize_data.initializeData();

	}

}

package output_processor_strategy;

public abstract class StorePrice {

	public abstract void storePrice();

}

package output_processor_strategy;

public abstract class SetPrice {

	public abstract void setPrice(int g, int m);

}

package output_processor_strategy;

public abstract class EnterPinMessage {

	public abstract void enterPinMsg();

}

package output_processor_strategy;

import data_store.Data_Store_GasPump1;

public class StorePrice1 extends StorePrice {

private  Data_Store_GasPump1 gas_data1;
```

```
        public StorePrice1(Data_Store_GasPump1 gas_data1) {
      this.gas_data1 = gas_data1;
   }
        @Override
        public void storePrice() {
                float a, b;
                System.out.println("enterrrvasjcvjhvshkn");
            a=gas_data1.getTemp_a();
            gas_data1.setReg_price(a);
            b=gas_data1.getTemp_b();
            gas_data1.setDiesel_price(b);
            System.out.println(a +" values in store price "+ b );

        }

}

package output_processor_strategy;

public abstract class InitialValues {

        public abstract void setInitialValues();

}

package output_processor_strategy;

public abstract class RejectMessage {

        public abstract void rejectMsg();

}

package output_processor_strategy;

import data_store.Data_Store_GasPump2;

public class InitializeData2 extends InitializeData {

private  Data_Store_GasPump2 gas_data2;

        public InitializeData2(Data_Store_GasPump2 gas_data2) {
      this.gas_data2 = gas_data2;
```

```
    }

        @Override
        public void initializeData() {
                gas_data2.setPrice(0.0f);
                gas_data2.setCash(0);
        }

}

package output_processor_strategy;

public abstract class StoreCash {

        public abstract void storeCash();

}

package output_processor_strategy;

public class WrongPinMessage1 extends WrongPinMessage{

        @Override
        public void wrongPinMsg() {
                 System.out.println("\n Wrong Pin entered  ");

        }

}

package output_processor_strategy;

import data_store.Data_Store_GasPump2;

public class StorePrice2 extends StorePrice{
private  Data_Store_GasPump2 gas_data2;

        public StorePrice2(Data_Store_GasPump2 gas_data2) {
      this.gas_data2 = gas_data2;
    }
        @Override
        public void storePrice() {
                float a,b,c;
            a=gas_data2.getTemp_a();
```

```
            gas_data2.setRegular_price(a);
            b=gas_data2.getTemp_b();
            gas_data2.setPremium_price(b);
            c=gas_data2.getTemp_c();
            gas_data2.setSuper_price(c);

        }

}

package output_processor_strategy;

import data_store.Data_Store_GasPump1;

public class InitialValues1 extends InitialValues{

private  Data_Store_GasPump1 gas_data1;

        public InitialValues1(Data_Store_GasPump1 gas_data1) {
    this.gas_data1 = gas_data1;
  }

        @Override
        public void setInitialValues() {
                gas_data1.setTotal(0.0f);
                gas_data1.setG(0);

        }

}

package output_processor_strategy;

import data_store.Data_Store_GasPump1;

public class GasPumpedMessage1 extends GasPumpedMessage {

private  Data_Store_GasPump1 gas_data1;

        public GasPumpedMessage1(Data_Store_GasPump1 gas_data1) {
    this.gas_data1 = gas_data1;
  }
        @Override
        public void gasPumpedMsg() {
```

```
                float g = gas_data1.getG();
        System.out.printf("Gas is pumped sucessfully and " +g +"gallon is filled \n");

        }

}

package output_processor_strategy;

public class PayMessage1 extends PayMessage{

        @Override
        public void payMsg() {
                System.out.println("\nPlease insert credit or debit card");

        }

}

package output_processor_strategy;

public class StopMessage1 extends StopMessage{

        @Override
        public void stopMsg() {

                System.out.println("\n Stopping the pump and receipt ?");
        }

}

package output_processor_strategy;

public abstract class GasPumpedMessage {

        public abstract void gasPumpedMsg();

}

package output_processor_strategy;

public class ReadyMessage1 extends ReadyMessage{

        @Override
```

```
	public void readyMsg() {

		System.out.println("\n Ready for pumping ");

	}

}

package output_processor_strategy;

import data_store.Data_Store_GasPump1;

public class SetPrice1 extends SetPrice{

private  Data_Store_GasPump1 gas_data1;

	public SetPrice1(Data_Store_GasPump1 gas_data1) {
      this.gas_data1 = gas_data1;
   }

	@Override
	public void setPrice(int g, int m) {

		float a=gas_data1.getReg_price();
	   float b=gas_data1.getDiesel_price();
	   System.out.println("reg pri"+a);
	   System.out.println("dies pri"+b);

	     if( g == 1){
			gas_data1.setPrice(a);
		}
		else if (g == 4)
			gas_data1.setPrice(b);

	}
}

package output_processor_strategy;

public class RejectMessage2 extends RejectMessage {

	@Override
	public void rejectMsg() {
		System.out.println("\n Credit Card Rejected  ");
```

```
	}

}

package output_processor_strategy;

public class PayMessage2 extends PayMessage {

	@Override
	public void payMsg() {
		System.out.println("\nPlease insert credit or debit card or by cash");

	}

}

package output_processor_strategy;

import data_store.Data_Store_GasPump2;

public class GasPumpedMessage2 extends GasPumpedMessage{

	private  Data_Store_GasPump2 gas_data2;

	public GasPumpedMessage2(Data_Store_GasPump2 gas_data2) {
      this.gas_data2 = gas_data2;
   }
	@Override
	public void gasPumpedMsg() {
		float l = gas_data2.getL();
	System.out.printf("Gas is pumped sucessfully and " + l +"litres is filled ");

	}

}

package output_processor_strategy;

public abstract class StopMessage {

	public abstract void stopMsg();

}
```

```
package output_processor_strategy;

import data_store.Data_Store_GasPump1;

public class StorePin1 extends StorePin {
private  Data_Store_GasPump1 gas_data1;

        public StorePin1(Data_Store_GasPump1 gas_data1) {
      this.gas_data1 = gas_data1;
   }
        @Override
        public void storePin() {
                String pin;
                pin=gas_data1.getTemp_pin();

                gas_data1.setPin(pin);

        }

}

package output_processor_strategy;

import data_store.Data_Store_GasPump1;

public class Receipt1 extends Receipt{

private  Data_Store_GasPump1 gas_data1;

        public Receipt1(Data_Store_GasPump1 gas_data1) {
      this.gas_data1 = gas_data1;
   }

        @Override
        public void printReceipt() {
                int G=gas_data1.getG();
        System.out.printf(" \n Total Gallon's of Gas pumped is "+G);

                float total = gas_data1.getTotal();
        System.out.printf("\n Amount is "+total);

        }
```

```
}

package output_processor_strategy;

import data_store.Data_Store_GasPump2;

public class SetPrice2 extends SetPrice{

private  Data_Store_GasPump2 gas_data2;

        public SetPrice2(Data_Store_GasPump2 gas_data2) {
      this.gas_data2 = gas_data2;
   }
        @Override
        public void setPrice(int g, int m) {

                float a=gas_data2.getSuper_price();
            float c=gas_data2.getRegular_price();
            float b=gas_data2.getPremium_price();

            if( g== 1)
                gas_data2.setPrice(c);
            else if (g == 2)
                gas_data2.setPrice(b);
            else if (g == 3)
                gas_data2.setPrice(a);
        }

}

package output_processor_strategy;

import data_store.Data_Store_GasPump2;

public class PumpGasUnit2 extends PumpGasUnit {

private  Data_Store_GasPump2 gas_data2;

        public PumpGasUnit2(Data_Store_GasPump2 gas_data2) {
      this.gas_data2 = gas_data2;
   }

        @Override
        public void pumpGasUnit() {
```

```
                int l=gas_data2.getL();
            l=l+1;
            float total;
            float price = gas_data2.getPrice();
            total = price * l;
            gas_data2.setL(l);
            gas_data2.setTotal(total);

        }

}

package output_processor_strategy;

import data_store.Data_Store_GasPump2;

public class StoreCash2 extends StoreCash{

private  Data_Store_GasPump2 gas_data2;

        public StoreCash2(Data_Store_GasPump2 gas_data2) {
     this.gas_data2 = gas_data2;
   }
        @Override
        public void storeCash() {
                int d;
                d=gas_data2.getTemp_cash();

                gas_data2.setCash(d);

        }

}

package output_processor_strategy;

public class ReadyMessage2 extends ReadyMessage{

        @Override
        public void readyMsg() {

                 System.out.println("\n Ready for pumping ");
```

```
	}

}

package output_processor_strategy;

import data_store.Data_Store_GasPump2;

public class ReturnCash2 extends ReturnCash{

private  Data_Store_GasPump2 gas_data2;

	public ReturnCash2(Data_Store_GasPump2 gas_data2) {
      this.gas_data2 = gas_data2;
   }
	@Override
	public void returnCash() {

		float return_cash=gas_data2.getCash()-gas_data2.getTotal();
		if(return_cash >0 && return_cash!=0)
		{
			System.out.println("Remaining Cash is " + return_cash);
		}

	}

}

package output_processor_strategy;

public abstract class InitializeData {

	public abstract void initializeData();

}

package output_processor_strategy;

import data_store.Data_Store_GasPump2;

public class Receipt2 extends Receipt{
private  Data_Store_GasPump2 gas_data2;

	public Receipt2(Data_Store_GasPump2 gas_data2) {
```

```
        this.gas_data2 = gas_data2;
    }

        @Override
        public void printReceipt() {
                int L=gas_data2.getL();
        System.out.printf(" \n Total Litres of Gas pumped is "+L);

                float total = gas_data2.getTotal();
        System.out.printf("\n Total amount is "+total);

        }

}

package output_processor_strategy;

public abstract class StorePin {

        public abstract void storePin();

}

package output_processor_strategy;

public abstract class Receipt {

        public abstract void printReceipt();

}

package output_processor_strategy;

public class RejectMessage1 extends RejectMessage {

        @Override
        public void rejectMsg() {
                System.out.println("\n Credit Card Rejected  ");

        }

}

package output_processor_strategy;
```

```
public abstract class ReturnCash {

	public abstract void returnCash();

}

package output_processor_strategy;

public abstract class WrongPinMessage {

	public abstract void wrongPinMsg();

}

package output_processor_strategy;

public abstract class PumpGasUnit {

	public abstract void pumpGasUnit();

}

package output_processor_strategy;

public abstract class ReadyMessage {

	public abstract void readyMsg();

}

package output_processor_strategy;

public class StopMessage2 extends StopMessage {

	@Override
	public void stopMsg() {
		System.out.println("\n Stopping the pump and receipt ?");

	}

}

package output_processor_strategy;
```

```
import data_store.Data_Store_GasPump1;

public class InitializeData1 extends InitializeData {

private  Data_Store_GasPump1 gas_data1;

	public InitializeData1(Data_Store_GasPump1 gas_data1) {
      this.gas_data1 = gas_data1;
   }

	@Override
	public void initializeData() {
		gas_data1.setPrice(0.0f);

	}

}

package output_processor_strategy;

import data_store.Data_Store_GasPump2;

public class InitialValues2 extends InitialValues{

private  Data_Store_GasPump2 gas_data2;

	public InitialValues2(Data_Store_GasPump2 gas_data2) {
      this.gas_data2 = gas_data2;
   }

	@Override
	public void setInitialValues() {
		gas_data2.setL(0);
		gas_data2.setTotal(0.0f);

	}

}

package output_processor_strategy;

public class EnterPinMessage1 extends EnterPinMessage{
```

```
	@Override
	public void enterPinMsg() {
		System.out.println("\n PLease enter your pin ");

	}

}
```