Software System Architecture Project

Fujie Bao

A20348799

1. MDA-EFSM

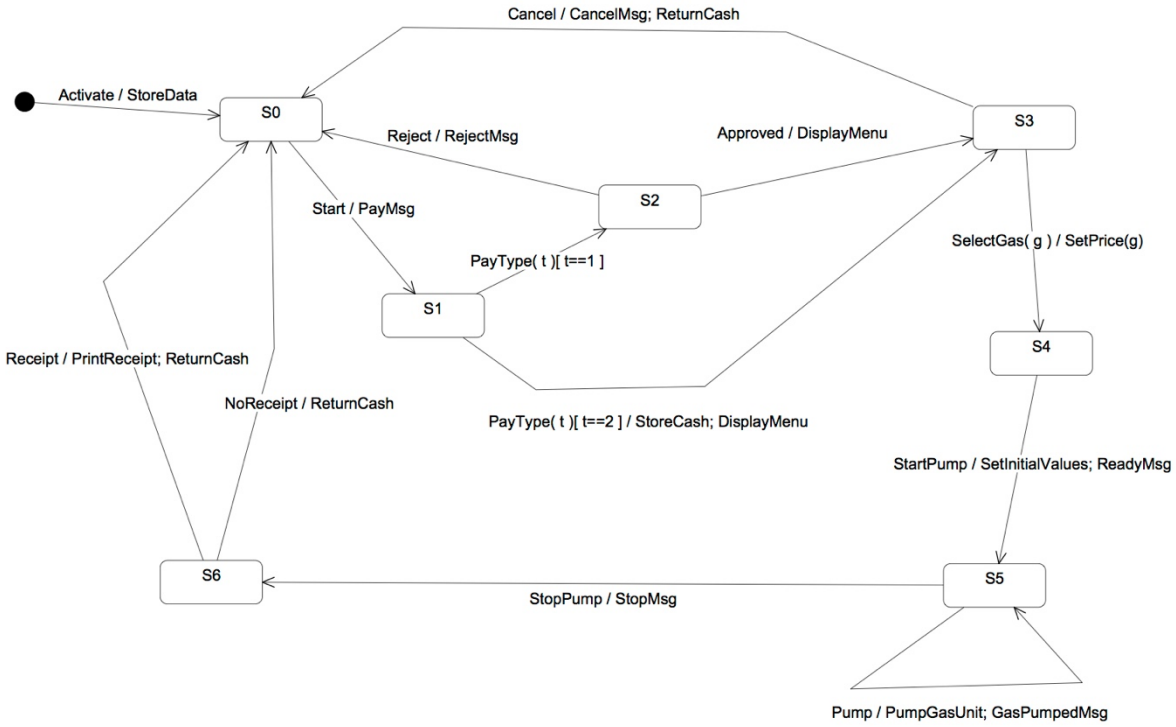
a. Meta events for MDA-EFSM

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

b. Meta actions for MDA-EFSM

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

c. State diagram



MDA-EFSM for Gas Pumps

d. Input processors of GasPump-1

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

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

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

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

Cancel() {
```

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

Super() {
    m->SelectGas(2) }

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

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

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

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

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

e. Input processors of GasPump-2

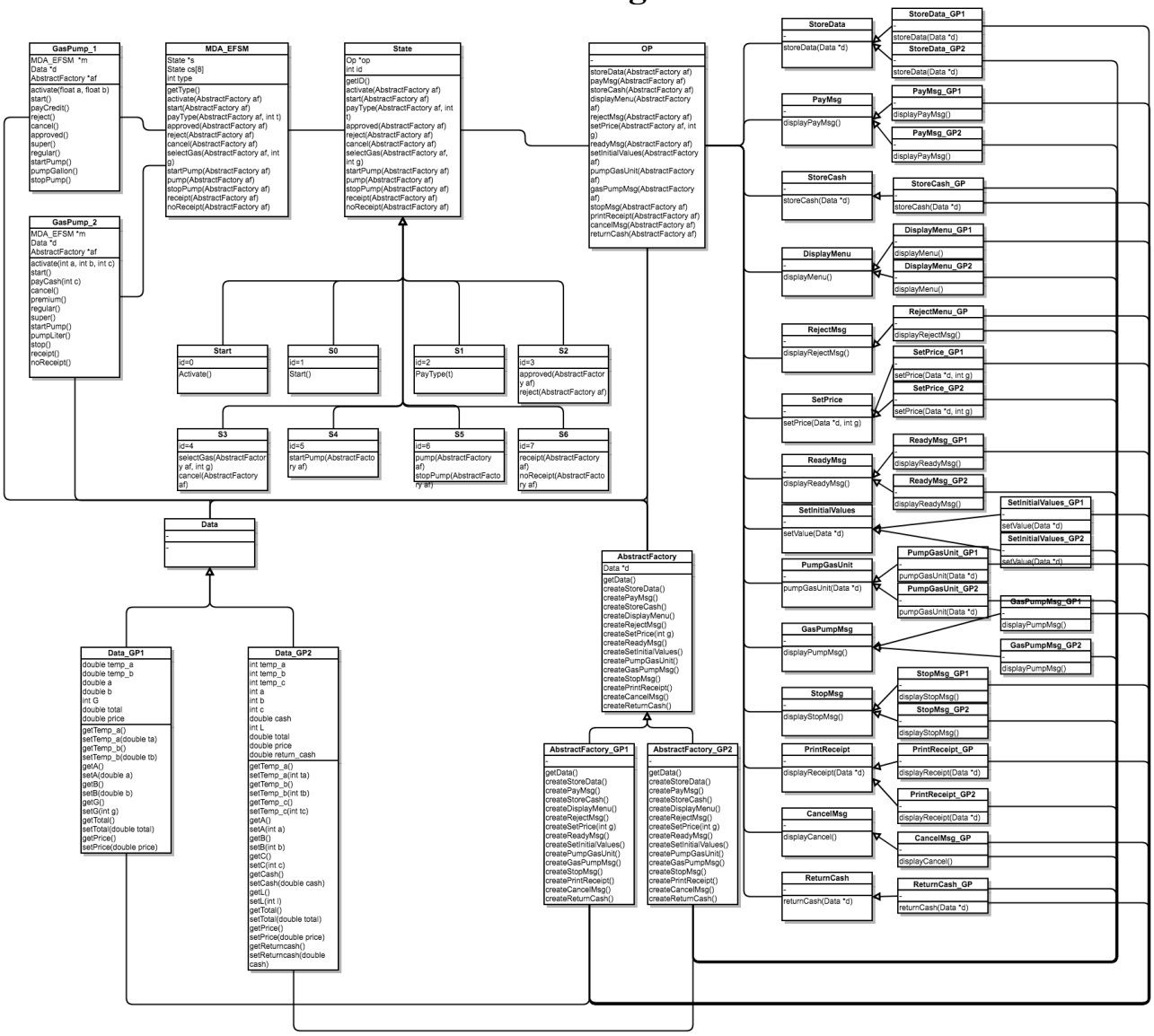
```
Activate(int a, int b, int c) {
        if ((a>0)&&(b>0)&&(c>0)) {
                d->temp_a=a;
                d->temp_b=b;
                d->temp_c=c;
                m->Activate() }
        }

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

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

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

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

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

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

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

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

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

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

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

Notice:

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

2. Class Diagram



3. The purpose and responsibility of each class

- **Class GasPump_1 //provide operations of GasPump-1**

Void activate(float a, float b) //activate the GasPump-1 and store the price of regular gas and super gas
Void start() //start the system to choose the pay type
Void payCredit() //pay the gas by credit card
Void reject() //credit card is rejected
Void cancel() //cancel the payment and return to start
Void approved() //credit card is approved
Void super() //select the super gas
Void regular() //select the regular gas
Void startPump() //start pump and show ready message
Void pumpGallon() //pump gas in a unit of gallon
Void stopPump() //stop pump and print receipt

- **Class GasPump_2 //provide operations of GasPump-2**

Void activate(int a, int b, int c) //activate the GasPump-2 and store the price of regular gas, premium gas and super gas
Void start() //start the system to choose the pay type
Void payCash(int c) //pay the gas by cash c
Void cancel() //cancel the payment and return to start
Void premium() //select the premium gas
Void regular() //select the regular gas
Void super() //select the super gas
Void startPump() //start pump and show ready message
Void pumpLiter() //pump gas in a unit of liter
Void stop() //stop pump and choose whether to print receipt
Void Receipt() //print receipt and return cash
Void noReceipt() //do not print receipt and return cash

- **Class Data //abstract class**

- **Class Data_GP1 //store data for GasPump-1**

Double getTemp_a() //return the value of temp_a
Void setTemp_a(double a) //set the value of temp_a
Double getTemp_b() return the value of temp_b
Void setTemp_b(double b) //set the value of temp_b
Double getA() //return the value of a
Void setA(double a) //set the value of a
Double getB() //return the value of b
Void setB(double b) //set the value of b
Int getG() //return the value of G
Void setG(int g) //set the value of G
Double getTotal() //return the value of total
Void setTotal(double total) set the value of total

Double getPrice() //return the value of price
Void setPrice(double price) //set the value of price

- **Class Data_GP2**

Int getTemp_a() //return the value of temp_a
Void setTemp_a(int a) //set the value of temp_a
Int getTemp_b() return the value of temp_b
Void setTemp_b(int b) //set the value of temp_b
Int getTemp_c() //return the value of temp_c
Void setTemp_c(int c) //set the value of temp_c
Int getA() //return the value of a
Void setA(int a) //set the value of a
Int getB() //return the value of b
Void setB(int b) //set the value of b
Int getC() //return the value of c
Void setC(int c) //set the value of c
Int getCash() //return the value of cash
Void setCash(int cash) //set the value of cash
Int getL() //return the value of L
Void setL(int l) //set the value of L
Double getTotal() //return the value of total
Void setTotal(double total) set the value of total
Int getPrice() //return the value of price
Void setPrice(int price) //set the value of price

- **Class MDA-EFSM //input processor incorporated with states**

Public MDA-EFSM() //allocate each states and initialize start state and ID=0
Int getType() //return the value of type
Void activate(AbstractFactory af) //if the current state is correct transfer state to S0
Void start(AbstractFactory af) // if the current state is correct transfer state to S1
Void payType(AbstractFactory af, int t) //if the current state is correct transfer state to S2 or S3
Void approved(AbstractFactory af) //if the current state is correct transfer state to S3
Void reject(AbstractFactory af) //if the current state is correct transfer state to S0
Void cancel(AbstractFactory af) //if the current state is correct transfer state to S0
Void selectGas(AbstractFactory af, int g) //if the current state is correct transfer state to S4
Void startPump(AbstractFactory af) //if the current state is correct transfer state to S5
Void pump(AbstractFactory af) //no state change
Void stopPump(AbstractFactory af) //if the current state is correct transfer state to S6
Void receipt(AbstractFactory af) //if the current state is correct transfer state to S0
Void noReceipt(AbstractFactory af) //if the current state is correct transfer state to S0

- **Class State //abstract class provide access to different state subclass**

Int getid() //return the value of id

- **Class Start //start state**

Int getid() //return id=0
Void activate(AbstractFactory af) //call according op events

- **Class S0 //S0 state**

Int getid() //return id=1
Void start(AbstractFactory af) //call according op events

- **Class S1 //state S1**

Int getid() //return id=2
Void payType(AbstractFactory af, int t) //call according op events

- **Class S2 //state S2**

Int getid() //return id=3
Void approved(AbstractFactory af) //call according op events
Void reject(AbstractFactory af) //call according op events

- **Class S3 //State S3**

Int getid() //return id=4
Void selectGas(AbstractFactory af, int g) //call according op events
Void cancel(AbstractFactory af) //call according op events

- **Class S4 //state S4**

Int getid() //return id=5
Void startPump(AbstractFactory af) //call according op events

- **Class S5 //state S5**

Int getid() //return id=6
Void pump(AbstractFactory af) //call according op events
Void stopPump(AbstractFctory af) //call according op events

- **Class S6 //state S6**

Int getid() //return id=7
Void receipt(AbstractFactory af) //call according op events
Void noReceipt(AbstractFactory af) //call according op events

- **Class Op //output processor execute actions**

Void storeData(AbstractFactory af) //call storeData by proper objects
Void payMsg(AbstractFactory af) //call displayPayMsg by proper objects

Void storeCash(AbstractFactory af) //call storeCash by proper objects
Void displayMenu(AbstractFactory af) //call displayMenu by proper objects
Void rejectMsg(AbstractFactory af) //call displayRejectMsg by proper objects
Void setPrice(AbstractFactory af, int g) //call setPrice by proper objects
Void ReadyMsg(AbstractFactory af) //call displayReadyMsg by proper objects
Void setInitialValue(AbstractFactory af) //call setValue by proper objects
Void pumpGasUnit(AbstractFactory af) //call pumpGasUnit by proper objects
Void gasPumpMsg(AbstractFactory af) //call displayPumpMsg by proper objects
Void stopMsg(AbstractFactory af) //call displayStopMsg by proper objects
Void printReceipt(AbstractFactory af) //call displayReceipt by proper objects
Void cancelMsg(AbstractFactory af) //call displayCancel by proper objects
Void returnCash(AbstractFactory af) //call returnCash by proper objects

- **Class AbstractFactory //abstract class provide access to different abstractfactory subclass**

- **Class AbstractFactory_GP1 //create objects of actions class for GasPump-1**

Data getData() //return current data object
StoreData createStoreData() //return GP1 StoreData object
PayMsg createPayMsg() //return GP1 PayMsg object
StoreCash createStoreCash() //return GP1 StoreCash object
DisplayMenu createDisplayMenu() //return GP1 DisplayMenu object
RejectMsg createRejectMsg() //return GP1 RejectMsg object
SetPrice createSetPrice() //return GP1 SetPrice object
ReadyMsg createReadyMsg() //return GP1 ReadyMsg object
SetInitialValue createSetInitialValue() //return GP1 SetInitialValue object
PumpGasUnit createPumpGasUnit() //return GP1 PumpGasUnit object
GasPumpMsg createGasPumpMsg() //return GP1 GasPumpMsg object
StopMsg createStopMsg() //return GP1 StopMsg object
PrintReceipt createPrintReceipt() //return GP1 PrintReceipt object
CancelMsg createCancelMsg() //return GP1 CancelMsg object
ReturnCash createReturnCash() //return GP1 ReturnCash object

- **Class AbstractFactory_GP2 //create objects of actions class for GasPump-2**

Data getData() //return current data object
StoreData createStoreData() //return GP2 StoreData object
PayMsg createPayMsg() //return GP2 PayMsg object
StoreCash createStoreCash() //return GP2 StoreCash object
DisplayMenu createDisplayMenu() //return GP2 DisplayMenu object
RejectMsg createRejectMsg() //return GP2 RejectMsg object
SetPrice createSetPrice() //return GP2 SetPrice object
ReadyMsg createReadyMsg() //return GP2 ReadyMsg object
SetInitialValue createSetInitialValue() //return GP2 SetInitialValue object
PumpGasUnit createPumpGasUnit() //return GP2 PumpGasUnit object
GasPumpMsg createGasPumpMsg() //return GP2 GasPumpMsg object

StopMsg createStopMsg() //return GP2 StopMsg object
PrintReceipt createPrintReceipt() //return GP2 PrintReceipt object
CancelMsg createCancelMsg() //return GP2 CancelMsg object
ReturnCash createReturnCash() //return GP2 ReturnCash object

- **Class StoreData //abstract class provide access to StoreData subclass**

- **Class StoreData_GP1 //concrete class of StoreData for GasPump-1**

Void storeData(Data d) //store the gas price by temp data

- **Class StroeData_GP2 //concrete class of StoreData for GasPump-2**

Void storeData(Data d) //store the gas price by temp data

- **Class PayMsg //abstract class provide access to PayMsg subclass**

- **Class PayMsg_GP1 //concrete class of PayMsg for GasPump-1**

Void displayPayMsg() //display pay message to choose pay type

- **Class PayMsg_GP2 //concrete class of PayMsg for GasPump-2**

Void displayPayMsg() //display pay message to choose pay type

- **Class StoreCash //abstract class provide access to StoreCash subclass**

- **Class StoreCash_GP //concrete class of StoreCash for GasPump**

Void storeCash(Data d) //store the cash by temp cash data

- **Class DisplayMenu //abstract class provide access to DisplayMenu subclass**

- **Class DisplayMenu_GP1 //concrete class of DisplayMenu for GasPump-1**

Void displayMenu() //display the menu to choose the gas type

- **Class DisplayMenu_GP2 //concrete class of DisplayMenu for GasPump-2**

Void displayMenu() //display the menu to choose the gas type

- **Class RejectMsg //abstract class provide access to RejectMsg subclass**

- **Class RejectMsg_GP //concrete class of RejectMsg for GasPump**

Void rejectMsg() //display message to show credit card has been rejected

- **Class SetPrice //abstract class provide access to SetPrice subclass**

- **Class SetPrice_GP1 //concrete class of SetPrice for GasPump-1**

Void setPrice(Data d, int g) //set the price of selected gas type by stored gas price

- **Class SetPrice_GP2 //concrete class of SetPrice for GasPump-2**

Void setPrice(Data d, int g) //set the price of selected gas type by stored gas price

- **Class ReadyMsg //abstract class provide access to ReadyMsg subclass**

- **Class ReadyMsg_GP //concrete class of ReadyMsg for GasPump**

Void readyMsg() //display message to show ready

- **Class SetInitialValue //abstract class provide access to SetInitialValue subclass**

- **Class SetInitialValue_GP1 //concrete class of SetInitialValue for GasPump-1**

Void setValue(Data d) //initial G=0 and total=0

- **Class SetInitialValue_GP2 //concrete class of SetInitialValue for GasPump-2**

Void setValue(Data d) //initial L=0 and total=0

- **Class PumpGasUnit //abstract class provide access to PumpGasUnit class**

- **Class PumpGasUnit_GP1 //concrete class of PumpGasUnit for GasPump-1**

Void pumpGasUnit(Data d) //increment gallon by 1 and calculate total

- **Class PumpGasUnit_GP2 //concrete class of PumpGasUnit for GasPump-2**

Void pumpGasUnit(Data d) //increment liter by 1 and calculate total

- **Class GasPumpMsg //abstract class provide access to GasPumpMsg subclass**

- **Class GasPumpMsg_GP1 //concrete class of GasPumpMsg for GasPump-1**

Void displayPumpMsg() //display a gallon has been pumped

- **Class GasPumpMsg_GP2 //concrete class of GasPumpMsg for GasPump-2**

Void displayPumpMsg() //display a liter has been pumped

- **Class StopMsg //abstract class provide access to StopMsg subclass**

- **Class StopMsg_GP1 //concrete class of StopMsg for GasPump-1**

Void displayStopMsg() //display stop message

- **Class StopMsg_GP2 //concrete class of StopMsg for GasPump-2**

Void displayStopMsg() //display stop message

- **Class PrintReceipt //abstract class provide access to PrintReceipt subclass**

- **Class PrintReceipt_GP1 //concrete class of PrintReceipt for GasPump-1**

Void displayReceipt(Data d) //display total

- **Class PrintReceipt_GP2 //concrete class of PrintReceipt for GasPump-2**

Void displayReceipt(Data d) //display total liter and total price

- **Class CancelMsg //abstract class provide access to CancelMsg subclass**

- **Class CancelMsg_GP //concrete class of CancelMsg for GasPump**

Void dispayCancel() //display cancel message

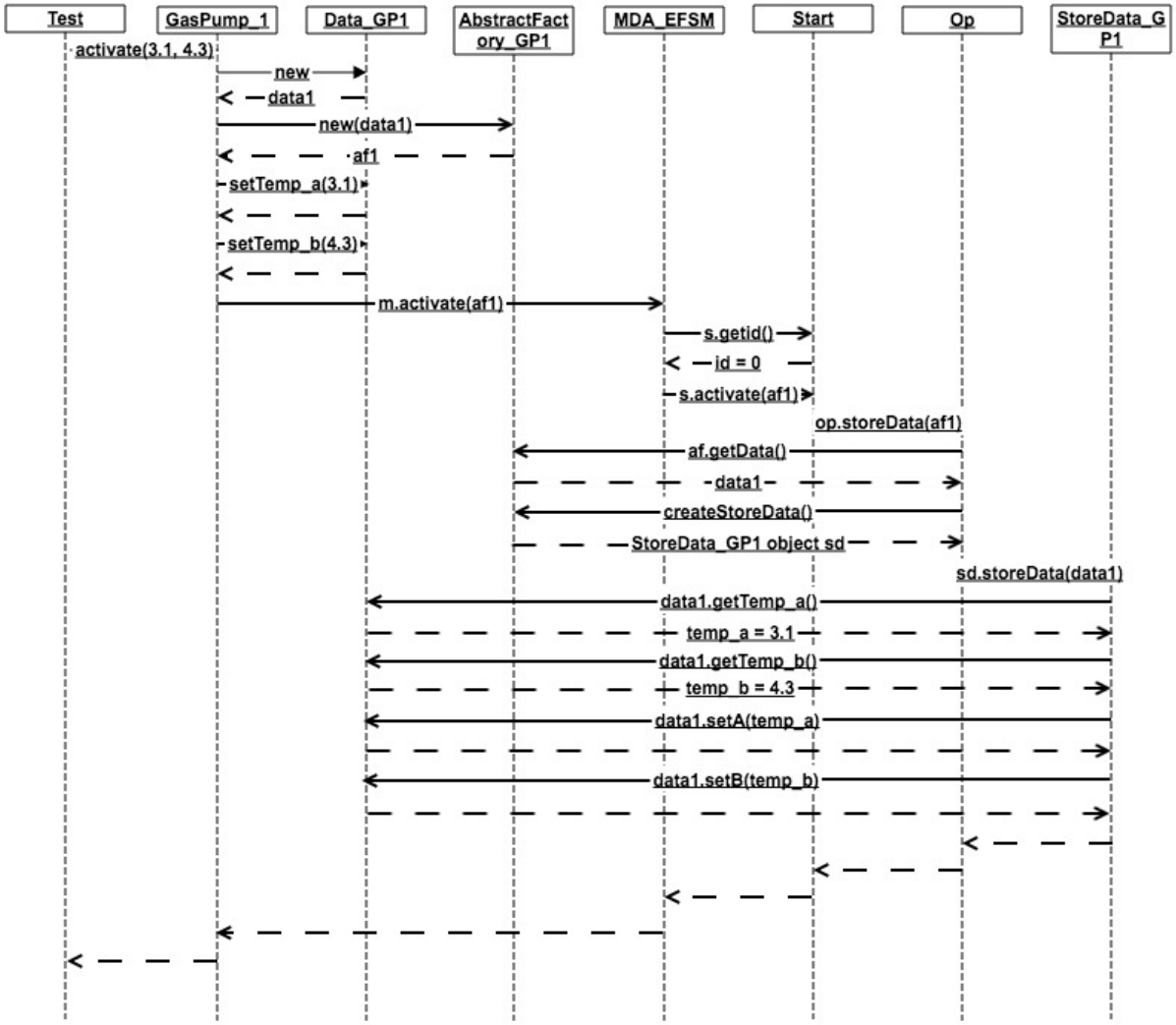
- **Class ReturnCash //abstract class provide access to ReturnCash subclass**

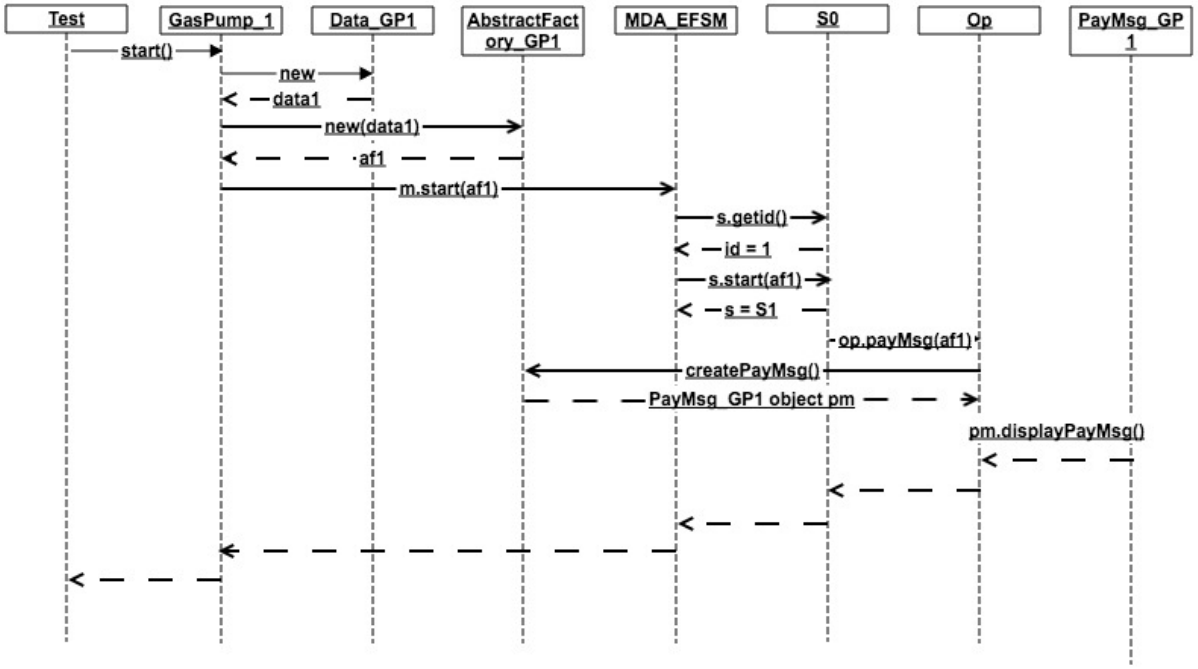
- **Class ReturnCash_GP //concrete class of ReturnCash for GasPump**

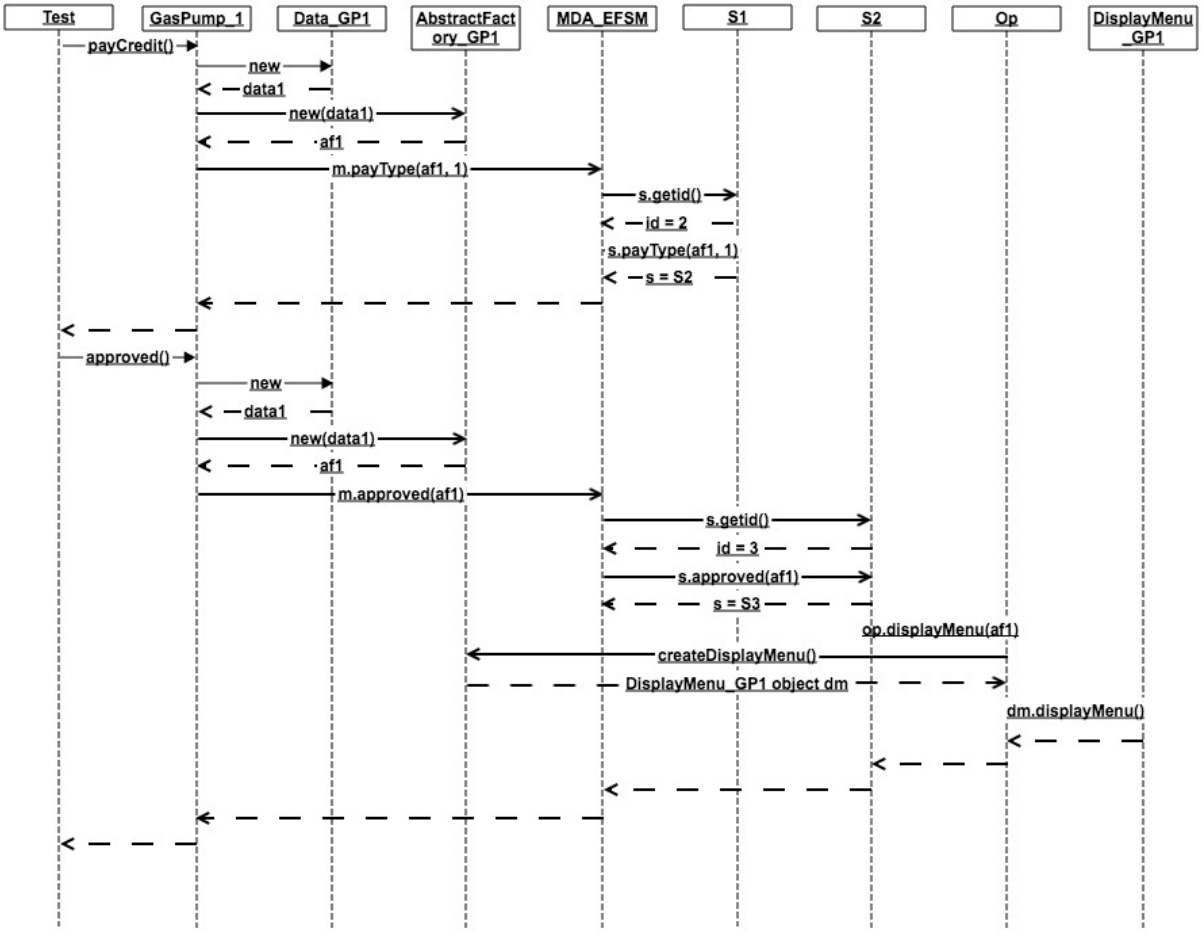
Void returnCash(Data d) //calculate the return cash and display it

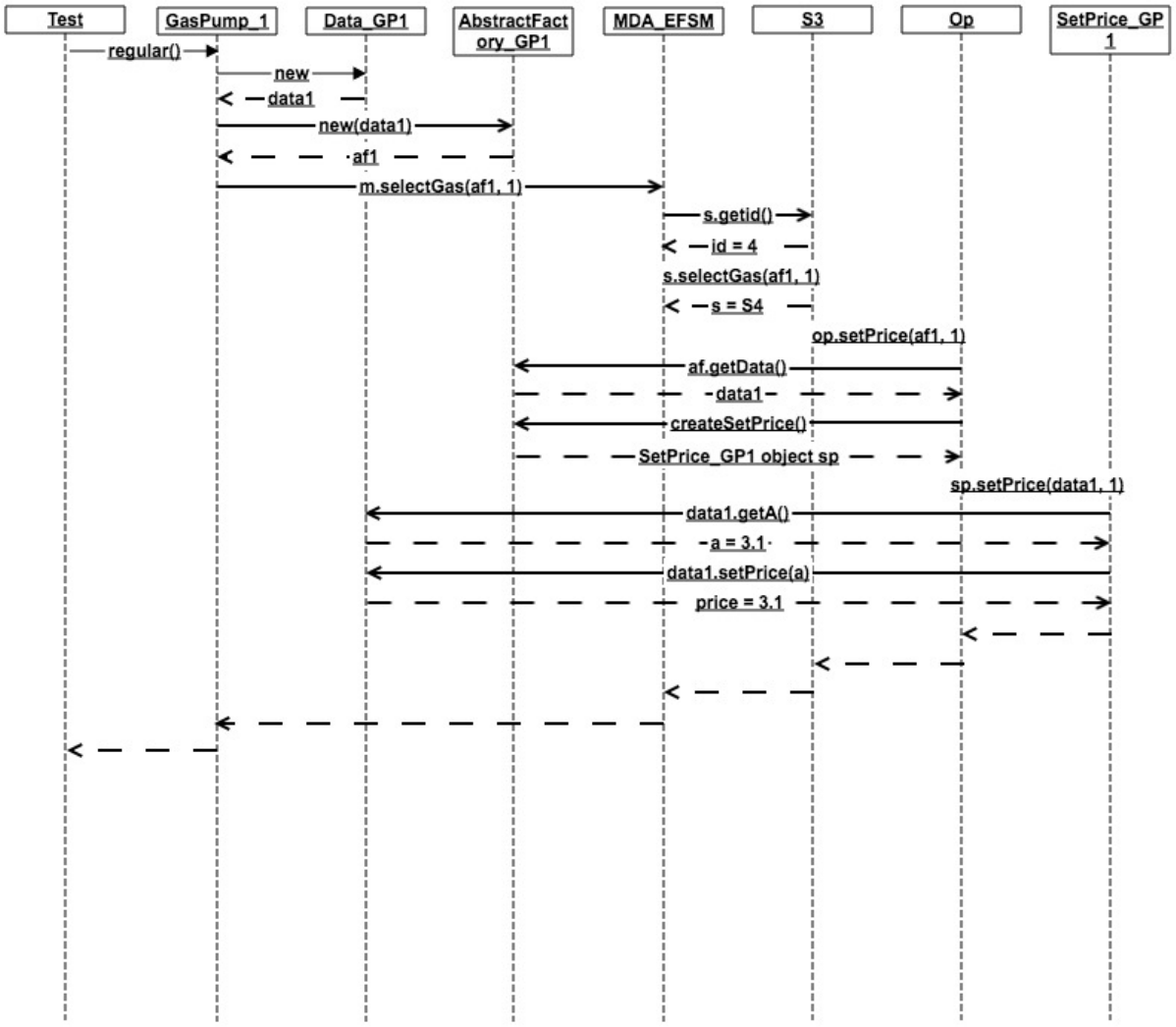
4. Sequence Diagram

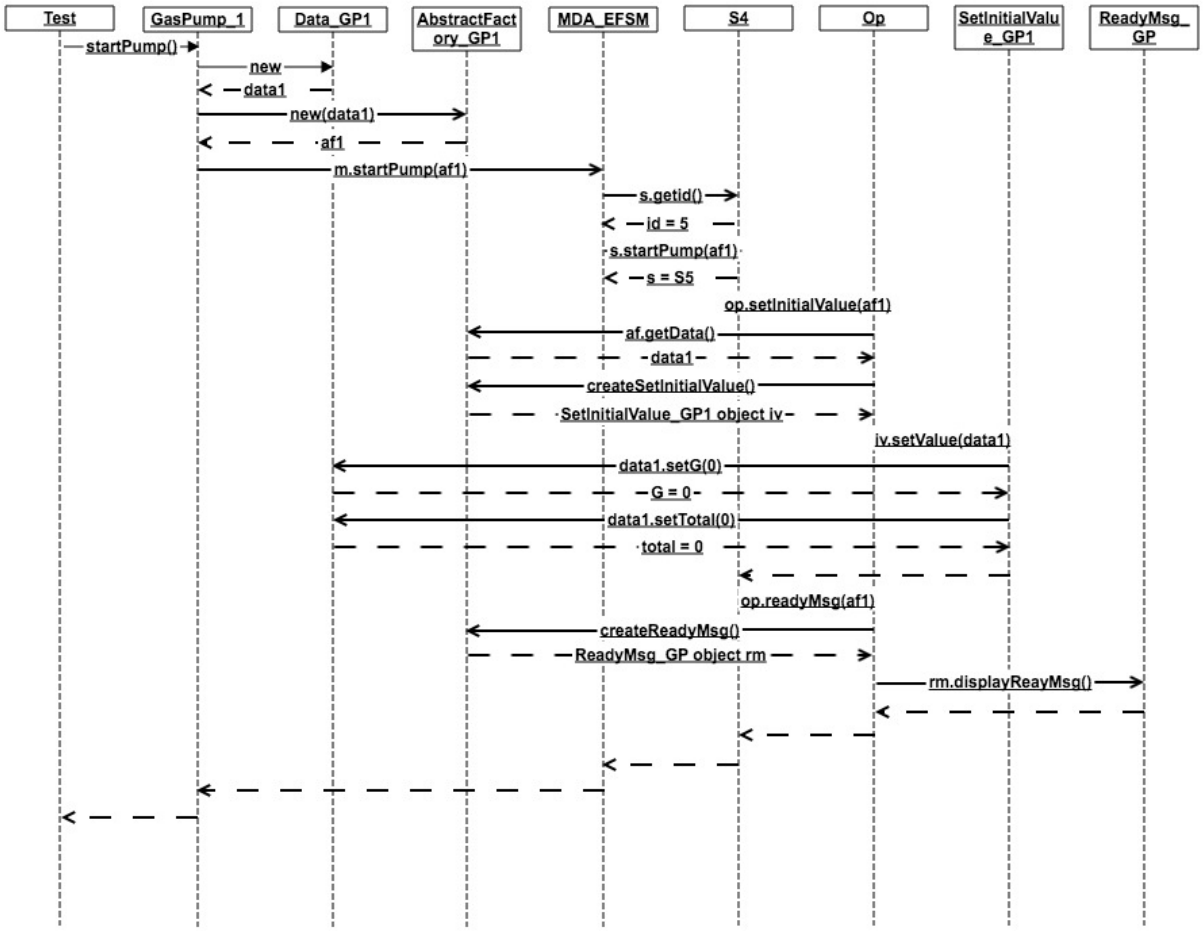
a. GasPump-1: Activate(3.1, 4.3), Start(), PayCredit(), Approved(), Regular(), StartPump(), PumpGallon(), StopPump()

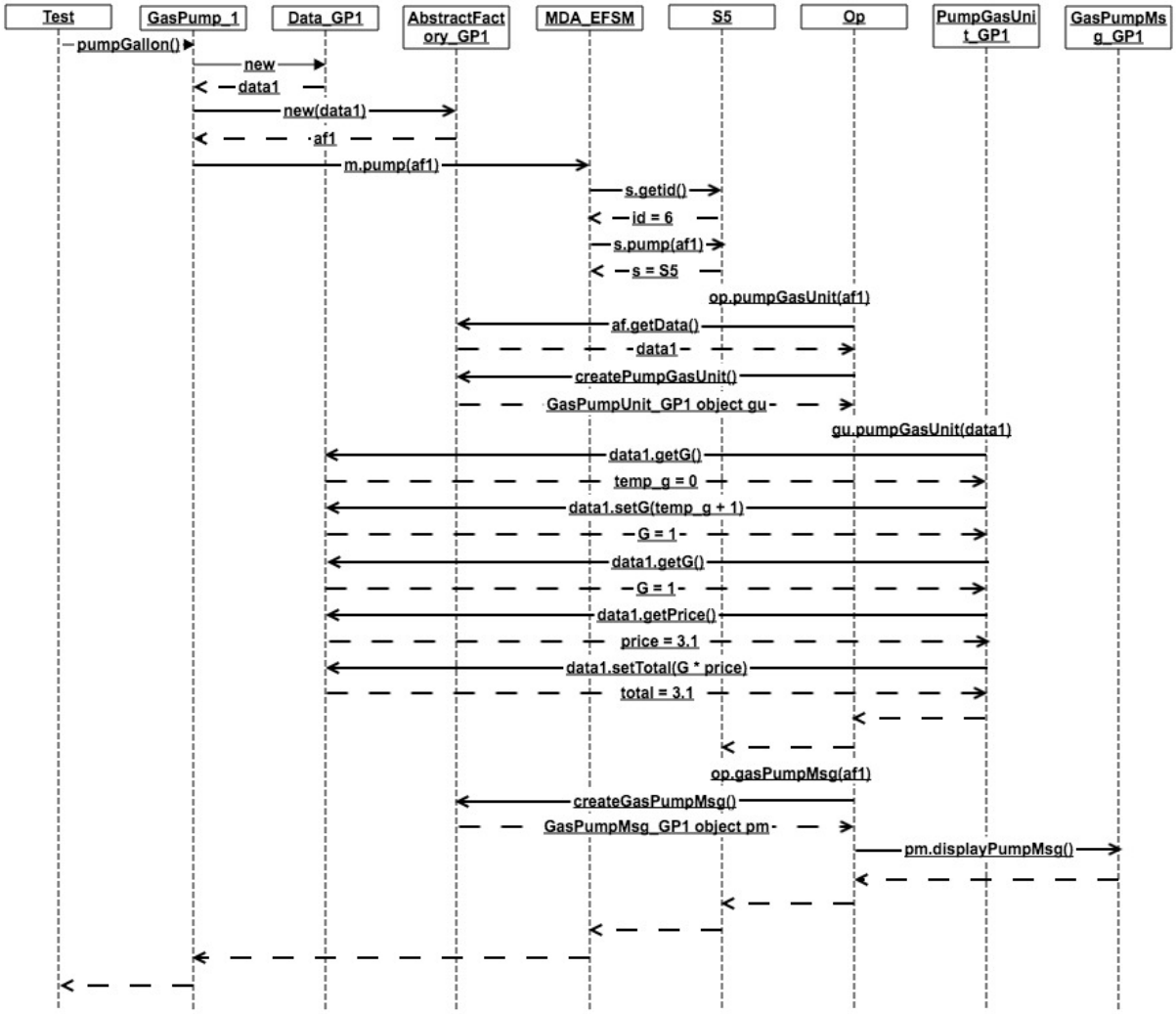


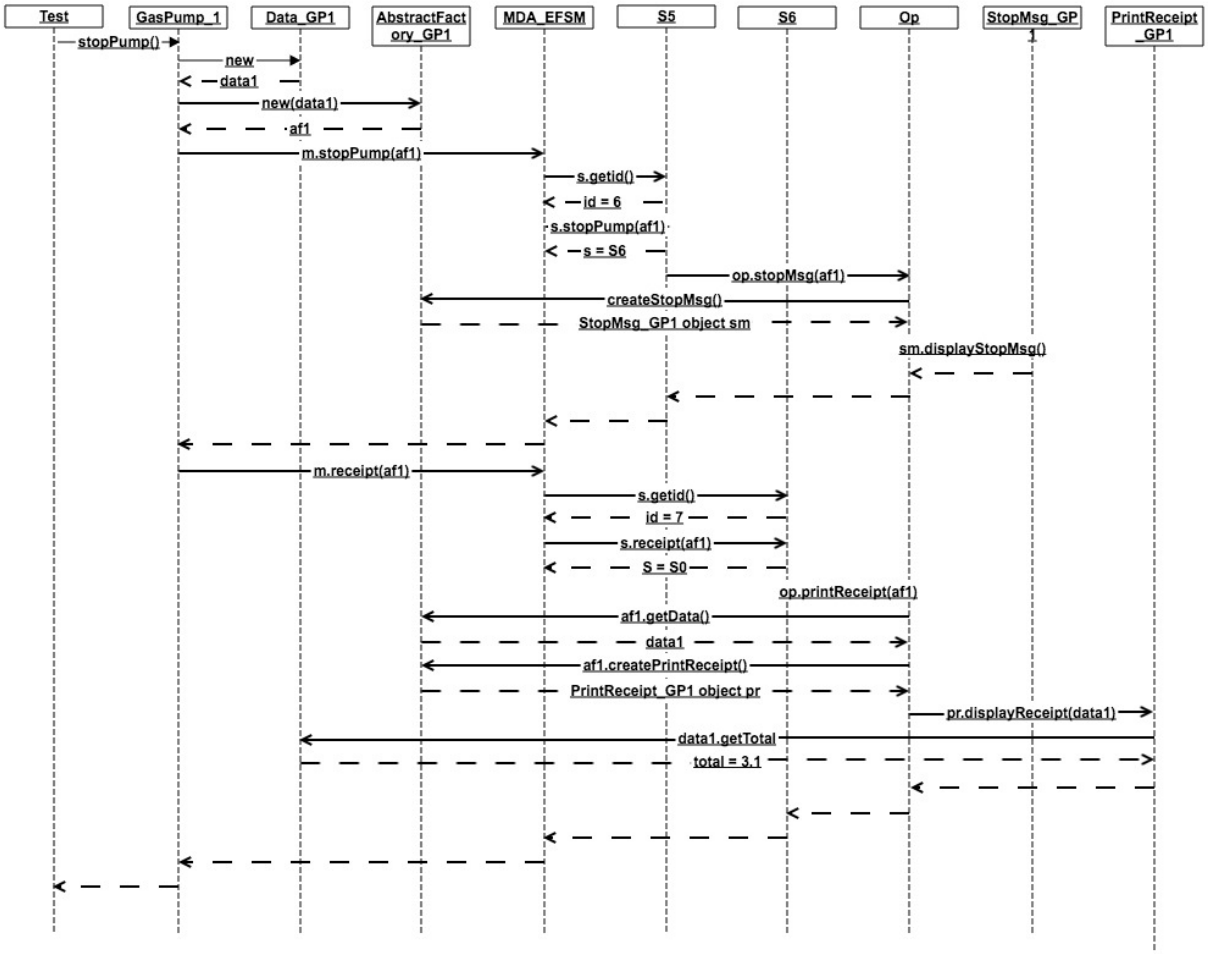

Test
GasPump_1
Data_GP1
AbstractFactory_GP1
MDA_EFSM
S0
Op
PayMsg_GP1
start()
new
data1
new(data1)
af1
m.start(af1)
s.getid()
id = 1
s.start(af1)
s = S1
op.payMsg(af1)
createPayMsg()
PayMsg_GP1 object pm
pm.displayPayMsg()


Test
GasPump_1
Data_GP1
AbstractFactory_GP1
MDA_EFSM
S1
S2
Op
DisplayMenu_GP1
payCredit()
new
data1
new(data1)
af1
m.payType(af1, 1)
s.getid()
id = 2
s.payType(af1, 1)
s = S2
approved()
new
data1
new(data1)
af1
m.approved(af1)
s.getid()
id = 3
s.approved(af1)
s = S3
op.displayMenu(af1)
createDisplayMenu()
DisplayMenu_GP1 object dm
dm.displayMenu()

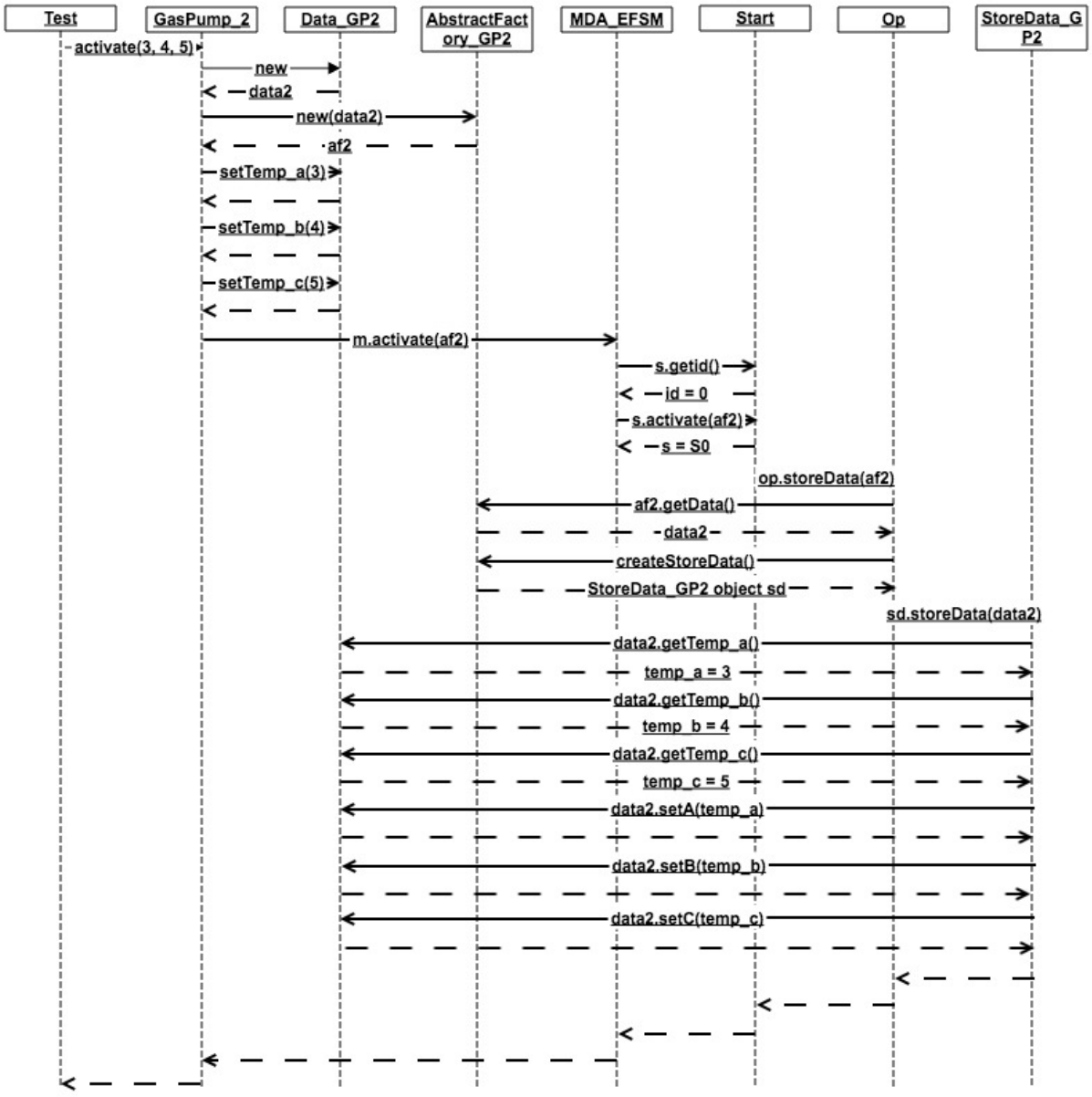

Test
GasPump_1
Data_GP1
AbstractFactory_GP1
MDA_EFSM
S3
Op
SetPrice_GP1
regular()
new
data1
new(data1)
af1
m.selectGas(af1, 1)
s.getid()
id = 4
s.selectGas(af1, 1)
s = S4
op.setPrice(af1, 1)
af.getData()
data1
createSetPrice()
SetPrice_GP1 object sp
sp.setPrice(data1, 1)
data1.getA()
a = 3.1
data1.setPrice(a)
price = 3.1

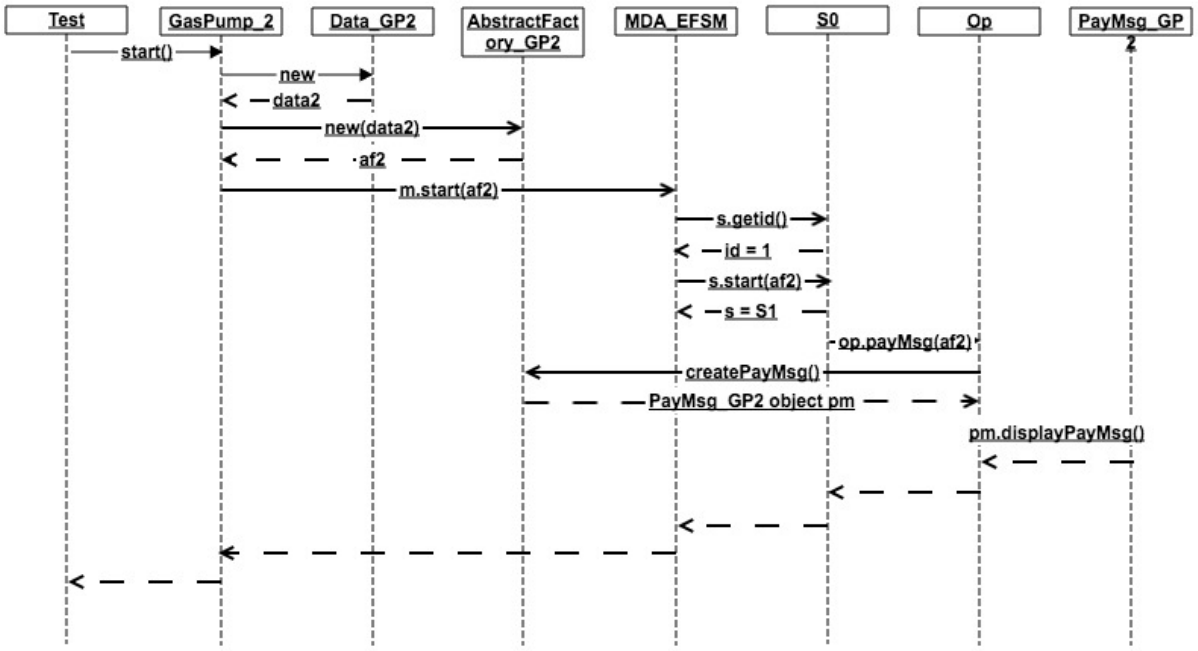

Test
GasPump_1
Data_GP1
AbstractFactory_GP1
MDA_EFSM
S4
Op
SetInitialValue_GP1
ReadyMsg_GP
startPump()
new
data1
new(data1)
af1
m.startPump(af1)
s.getid()
id = 5
s.startPump(af1)
s = S5
op.setInitialValue(af1)
af.getData()
data1
createSetInitialValue()
SetInitialValue_GP1 object iv
iv.setValue(data1)
data1.setG(0)
G = 0
data1.setTotal(0)
total = 0
op.readyMsg(af1)
createReadyMsg()
ReadyMsg_GP object rm
rm.displayReayMsg()

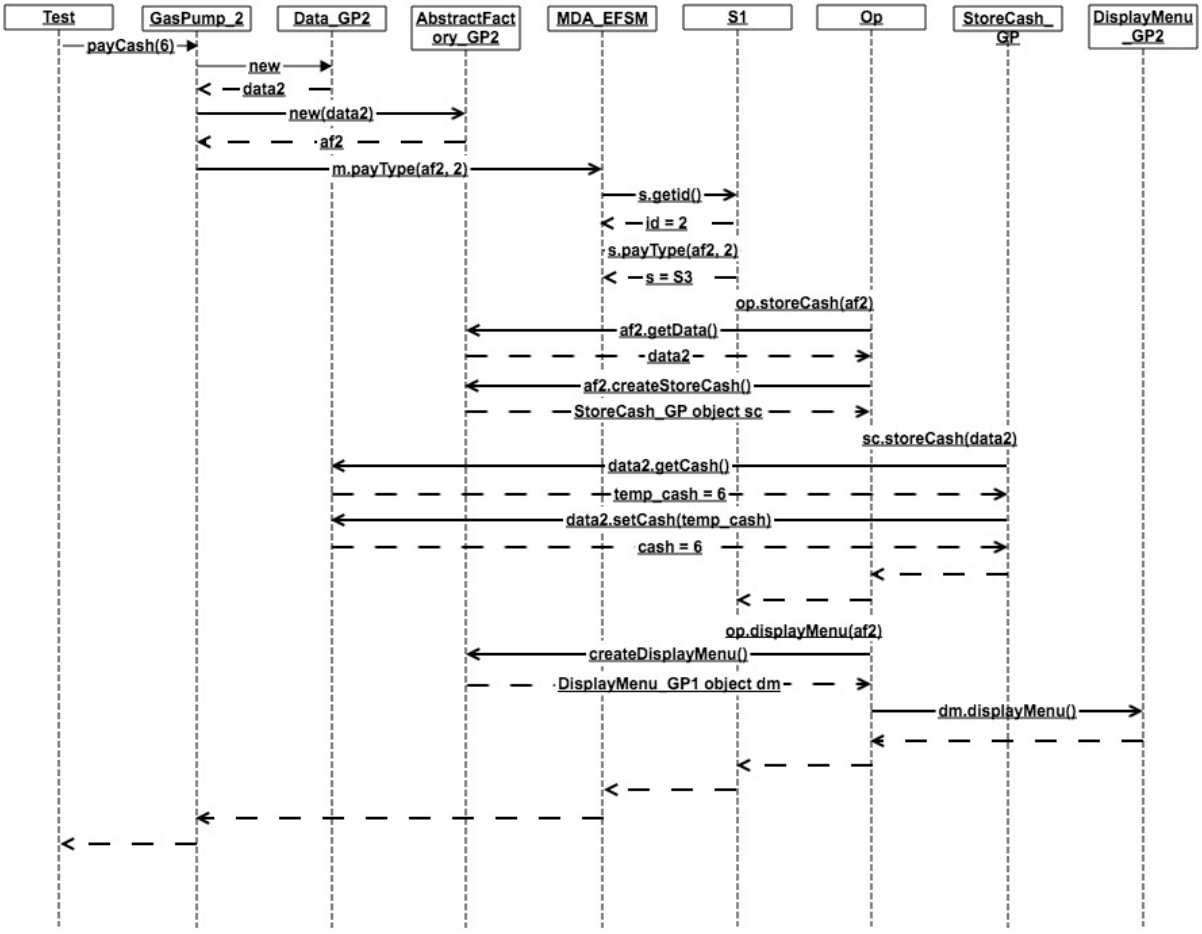

Test
GasPump_1
Data_GP1
AbstractFactory_GP1
MDA_EFSM
S5
Op
PumpGasUnit_GP1
GasPumpMsg_GP1
pumpGallon()
new
data1
new(data1)
af1
m.pump(af1)
s.getid()
id = 6
s.pump(af1)
s = S5
op.pumpGasUnit(af1)
af.getData()
data1
createPumpGasUnit()
GasPumpUnit_GP1 object gu
gu.pumpGasUnit(data1)
data1.getG()
temp_g = 0
data1.setG(temp_g + 1)
G = 1
data1.getG()
G = 1
data1.getPrice()
price = 3.1
data1.setTotal(G * price)
total = 3.1
op.gasPumpMsg(af1)
createGasPumpMsg()
GasPumpMsg_GP1 object pm
pm.displayPumpMsg()

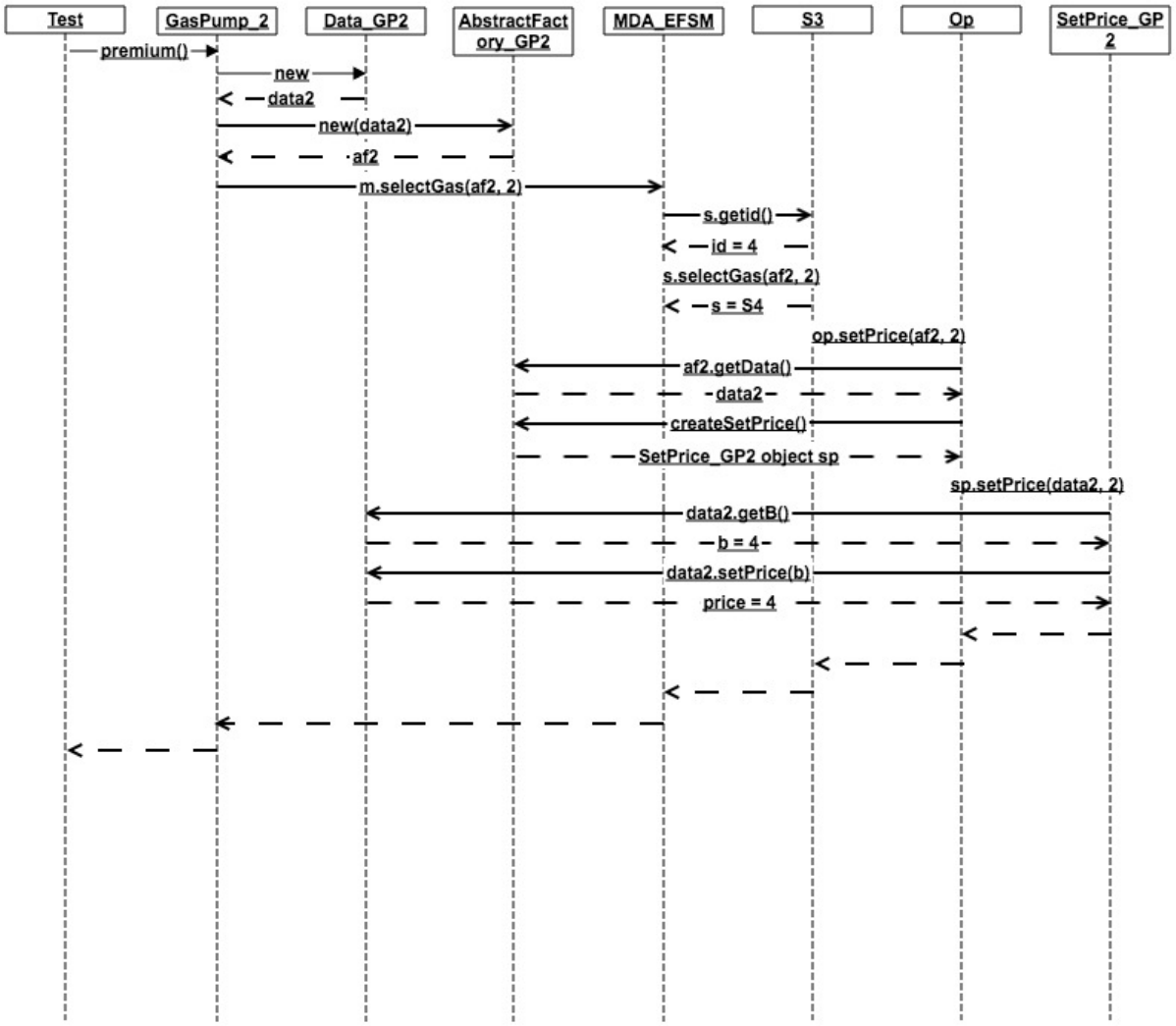

Test
GasPump_1
Data_GP1
AbstractFactory_GP1
MDA_EFSM
S5
S6
Op
StopMsg_GP1
PrintReceipt_GP1
stopPump()
new
data1
new(data1)
af1
m.stopPump(af1)
s.getid()
id = 6
s.stopPump(af1)
s = S6
op.stopMsg(af1)
createStopMsg()
StopMsg_GP1 object sm
sm.displayStopMsg()
m.receipt(af1)
s.getid()
id = 7
s.receipt(af1)
S = S0
op.printReceipt(af1)
af1.getData()
data1
af1.createPrintReceipt()
PrintReceipt_GP1 object pr
pr.displayReceipt(data1)
data1.getTotal
total = 3.1

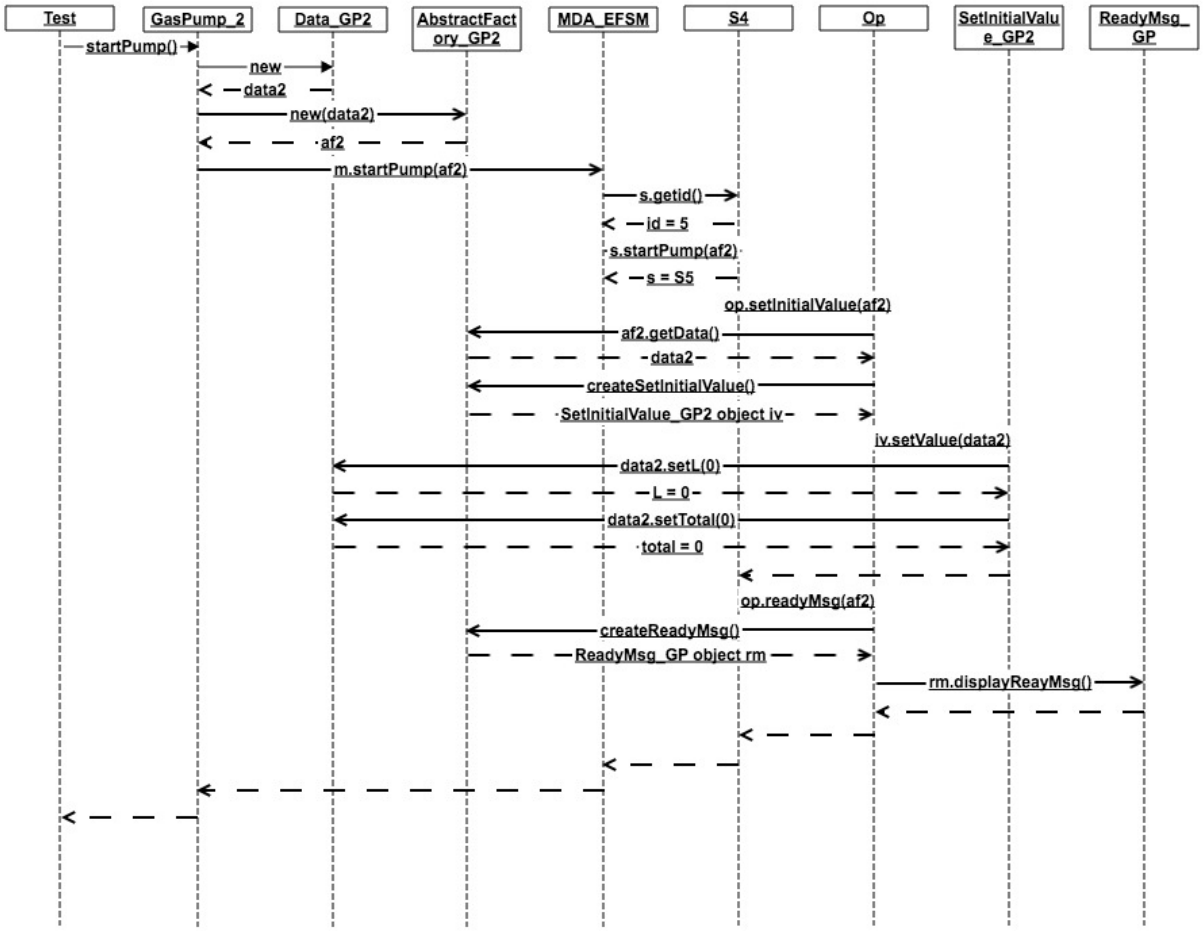
b. GasPump-2: Activate(3, 4, 5), Start(), PayCash(), Premium(), StartPump(), PumpLiter(), PumpLiter(), Noreceipt()

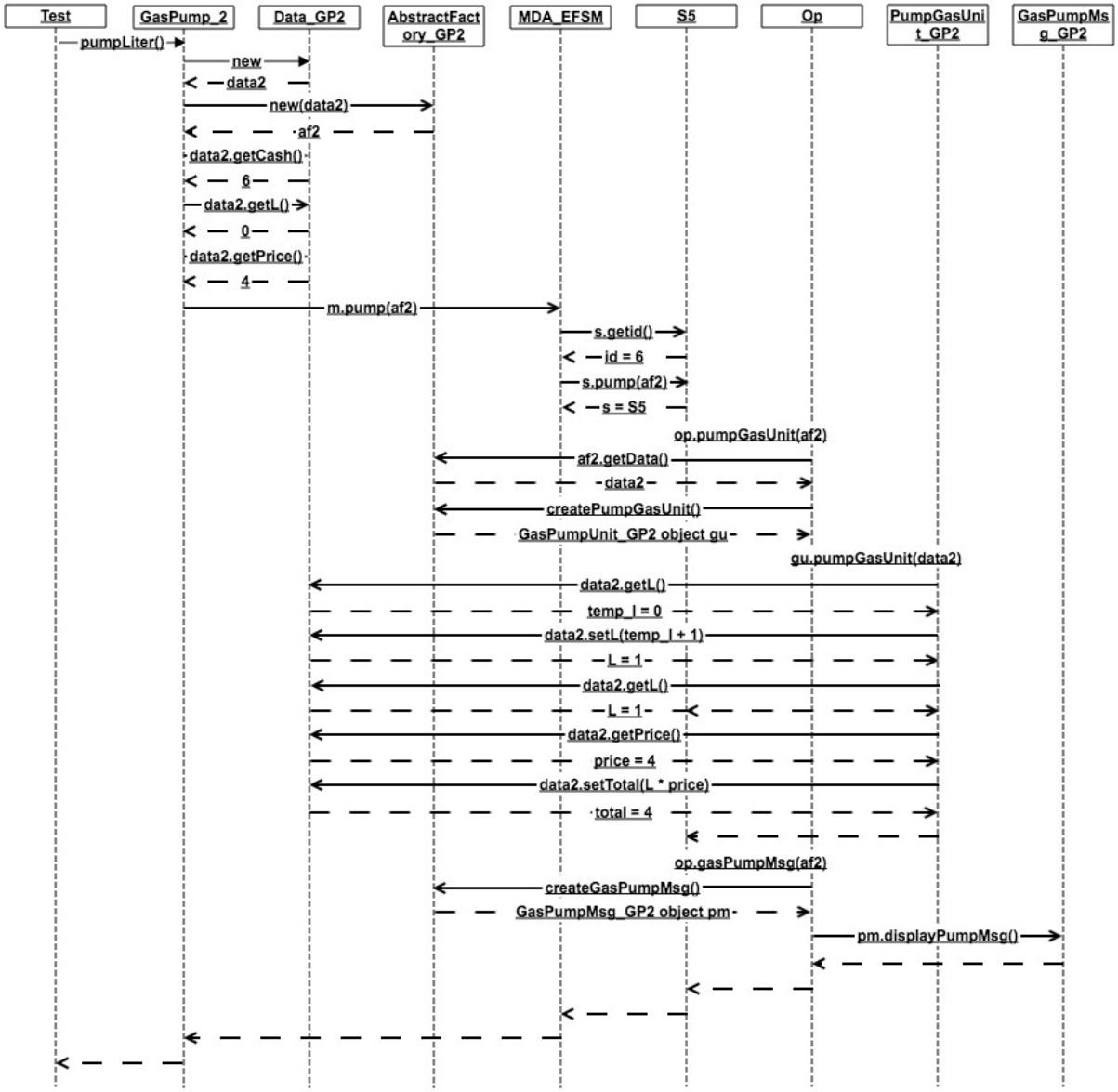


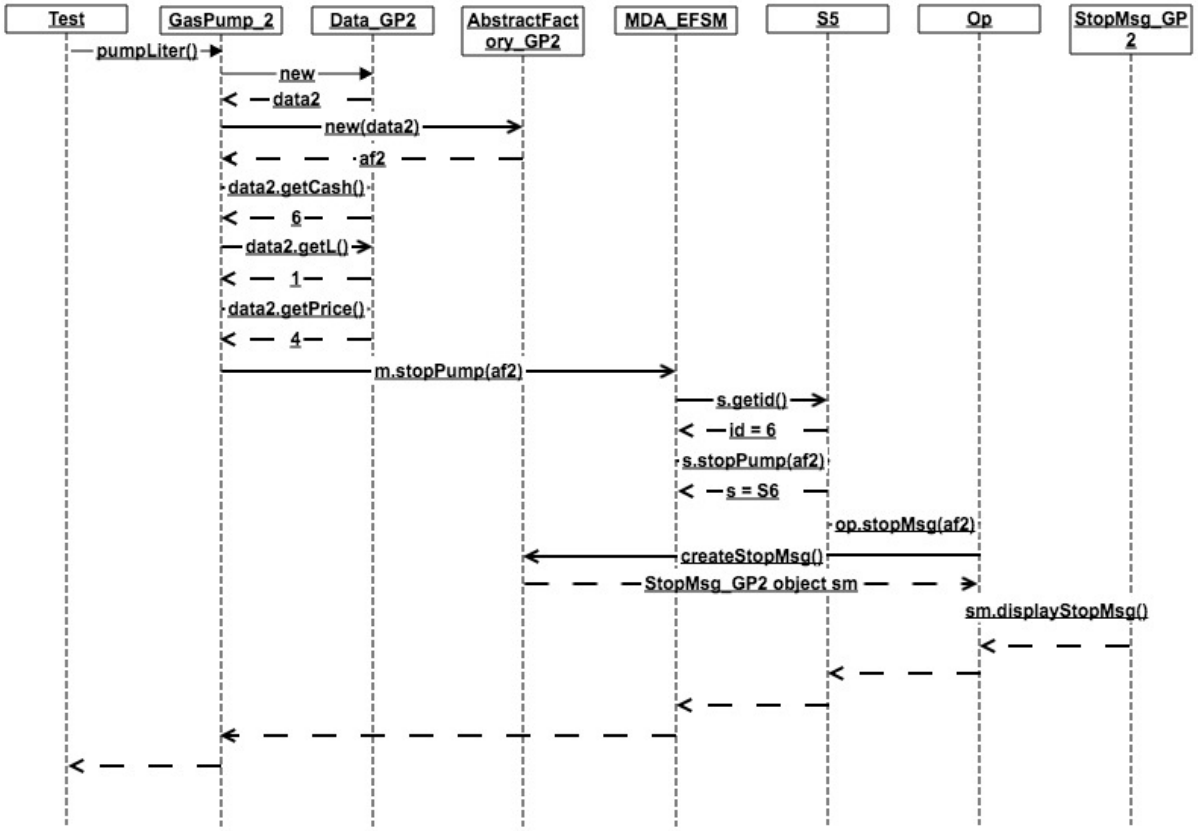

Test
GasPump_2
Data_GP2
AbstractFactory_GP2
MDA_EFSM
S0
Op
PayMsg_GP2
start()
new
data2
new(data2)
af2
m.start(af2)
s.getid()
id = 1
s.start(af2)
s = S1
op.payMsg(af2)
createPayMsg()
PayMsg_GP2 object pm
pm.displayPayMsg()

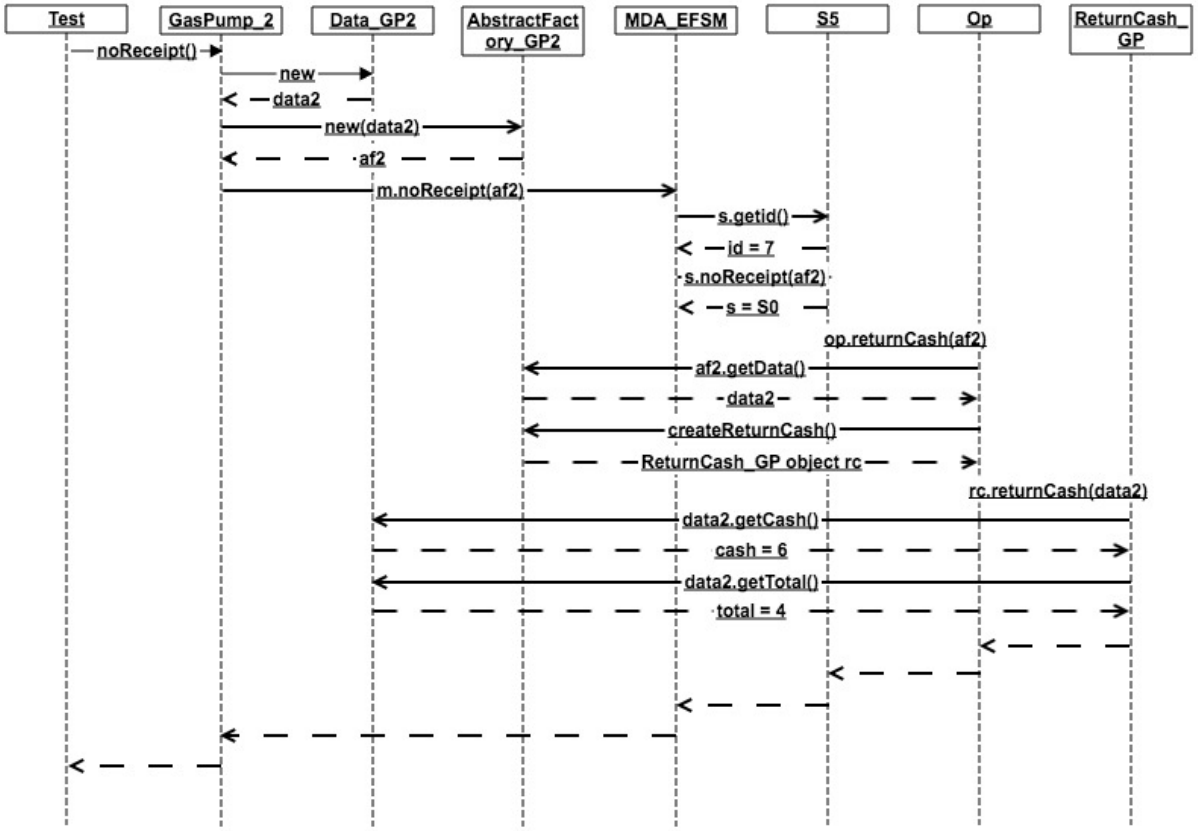

Test
GasPump_2
Data_GP2
AbstractFactory_GP2
MDA_EFSM
S1
Op
StoreCash_GP
DisplayMenu_GP2
payCash(6)
new
data2
new(data2)
af2
m.payType(af2, 2)
s.getid()
id = 2
s.payType(af2, 2)
s = S3
op.storeCash(af2)
af2.getData()
data2
af2.createStoreCash()
StoreCash_GP object sc
sc.storeCash(data2)
data2.getCash()
temp_cash = 6
data2.setCash(temp_cash)
cash = 6
op.displayMenu(af2)
createDisplayMenu()
DisplayMenu_GP1 object dm
dm.displayMenu()


Test
GasPump_2
Data_GP2
AbstractFactory_GP2
MDA_EFSM
S3
Op
SetPrice_GP2
premium()
new
data2
new(data2)
af2
m.selectGas(af2, 2)
s.getid()
id = 4
s.selectGas(af2, 2)
s = S4
op.setPrice(af2, 2)
af2.getData()
data2
createSetPrice()
SetPrice_GP2 object sp
sp.setPrice(data2, 2)
data2.getB()
b = 4
data2.setPrice(b)
price = 4


Test
GasPump_2
Data_GP2
AbstractFactory_GP2
MDA_EFSM
S4
Op
SetInitialValue_GP2
ReadyMsg_GP
startPump()
new
data2
new(data2)
af2
m.startPump(af2)
s.getid()
id = 5
s.startPump(af2)
s = S5
op.setInitialValue(af2)
af2.getData()
data2
createSetInitialValue()
SetInitialValue_GP2 object iv
iv.setValue(data2)
data2.setL(0)
L = 0
data2.setTotal(0)
total = 0
op.readyMsg(af2)
createReadyMsg()
ReadyMsg_GP object rm
rm.displayReayMsg()


Test
GasPump_2
Data_GP2
AbstractFactory_GP2
MDA_EFSM
S5
Op
PumpGasUnit_GP2
GasPumpMsg_GP2
pumpLiter()
new
data2
new(data2)
af2
data2.getCash()
6
data2.getL()
0
data2.getPrice()
4
m.pump(af2)
s.getid()
id = 6
s.pump(af2)
s = S5
op.pumpGasUnit(af2)
af2.getData()
data2
createPumpGasUnit()
GasPumpUnit_GP2 object gu
gu.pumpGasUnit(data2)
data2.getL()
temp_l = 0
data2.setL(temp_l + 1)
L = 1
data2.getL()
L = 1
data2.getPrice()
price = 4
data2.setTotal(L * price)
total = 4
op.gasPumpMsg(af2)
createGasPumpMsg()
GasPumpMsg_GP2 object pm
pm.displayPumpMsg()


Test
GasPump_2
Data_GP2
AbstractFactory_GP2
MDA_EFSM
S5
Op
StopMsg_GP2
pumpLiter()
new
data2
new(data2)
af2
data2.getCash()
6
data2.getL()
1
data2.getPrice()
4
m.stopPump(af2)
s.getid()
id = 6
s.stopPump(af2)
s = S6
op.stopMsg(af2)
createStopMsg()
StopMsg_GP2 object sm
sm.displayStopMsg()


Test
GasPump_2
Data_GP2
AbstractFactory_GP2
MDA_EFSM
S5
Op
ReturnCash_GP
noReceipt()
new
data2
new(data2)
af2
m.noReceipt(af2)
s.getid()
id = 7
s.noReceipt(af2)
s = S0
op.returnCash(af2)
af2.getData()
data2
createReturnCash()
ReturnCash_GP object rc
rc.returnCash(data2)
data2.getCash()
cash = 6
data2.getTotal()
total = 4

5. Source code and patterns

a. State pattern

State pattern consists of MDA-EFSM class, state class and its subclasses which are start, S0, S1, S2, S3, S4, S5, S6 classes. I use centralize version to implement the state pattern. MDA-EFSM controls the transition of states and state classes will call the corresponding actions based on the MDA-EFSM

MDA-EFSM class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public class MDA_EFSM {
    State s;
    State[] cs;
    int type = 0;
    int ID;

    public MDA_EFSM(){
        cs = new State[8];
        cs[0] = new Start();
        cs[1] = new S0();
        cs[2] = new S1();
        cs[3] = new S2();
        cs[4] = new S3();
        cs[5] = new S4();
        cs[6] = new S5();
        cs[7] = new S6();

        s = cs[0];    //set the initial state to start
        ID = 0;
    }

    public int getType() {
        return type;
    }

    //transfer states
    public void activate(AbstractFactory af){
        ID = s.getid();
        if(ID == 0){
            s.activate(af);
            s = cs[1];
        }
    }
```

```
public void start(AbstractFactory af){
   ID = s.getid();
   if(ID == 1){
      s.start(af);
      s = cs[2];
   }
}

public void payType(AbstractFactory af, int t){
   ID = s.getid();
   if(ID == 2){
      if(t == 1){
         s.payType(af, 1);
         s = cs[3];
      }else if(t == 2){
         s.payType(af, 2);
         s = cs[4];
      }
   }
}

public void approved(AbstractFactory af){
   ID = s.getid();
   if(ID == 3){
      s.approved(af);
      s = cs[4];
   }
}

public void reject(AbstractFactory af){
   ID = s.getid();
   if(ID == 3){
      s.reject(af);
      s = cs[1];
   }
}

public void cancel(AbstractFactory af){
   ID = s.getid();
   if(ID == 4){
      s.cancel(af);
      s = cs[1];
   }
}
```

```
    public void selectGas(AbstractFactory af, int g){
        ID = s.getid();
        if(ID == 4){
            s.selectGas(af, g);
            s = cs[5];
        }
    }

    public void startPump(AbstractFactory af){
        ID = s.getid();
        if(ID == 5){
            s.startPump(af);
            s = cs[6];
        }
    }

    public void pump(AbstractFactory af){
        s.pump(af);
    }

    public void stopPump(AbstractFactory af){
        ID = s.getid();
        if(ID == 6){
            s.stopPump(af);
            s = cs[7];
        }
    }

    public void receipt(AbstractFactory af){
        ID = s.getid();
        if(ID == 7){
            s.receipt(af);
            s = cs[1];
        }
    }

    public void noReceipt(AbstractFactory af){
        ID = s.getid();
        if(ID == 7){
            s.noReceipt(af);
            s = cs[1];
        }
    }
}
```

State class

```
package com.fujiebao;

/**
 * Created by vincent on 4/3/17.
 */
public abstract class State {
    Op op;
    int id;

    public State(){
        op = new Op();
        id = 0;
    }

    public int getid(){
        return id;
    }

    public void activate(AbstractFactory af){

    }

    public void start(AbstractFactory af){

    }

    public void payType(AbstractFactory af, int t){

    }

    public void approved(AbstractFactory af){

    }

    public void reject(AbstractFactory af){

    }

    public void cancel(AbstractFactory af){

    }

    public void selectGas(AbstractFactory af, int g){

    }
```

```
    public void startPump(AbstractFactory af){

    }

    public void pump(AbstractFactory af){

    }

    public void stopPump(AbstractFactory af){

    }

    public void receipt(AbstractFactory af){

    }

    public void noReceipt(AbstractFactory af){

    }

}
```

Start class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public class Start extends State{

    @Override
    public int getid(){
        id = 0;
        return id;
    }

    @Override
    public void activate(AbstractFactory af){
        op.storeData(af);
    }
}
```

S0 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public class S0 extends State {

    @Override
    public int getid(){
        id = 1;
        return id;
    }

    @Override
    public void start(AbstractFactory af){
        op.payMsg(af);
    }
}
```

S1 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public class S1 extends State {

    @Override
    public int getid(){
        id = 2;
        return id;
    }

    @Override
    public void payType(AbstractFactory af, int t){
        if(t == 2){
            op.storeCash(af);
            op.displayMenu(af);
        }
    }
}
```

S2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public class S2 extends State {

    @Override
    public int getid(){
        id = 3;
        return id;
    }

    @Override
    public void approved(AbstractFactory af){
        op.displayMenu(af);
    }

    @Override
    public  void reject(AbstractFactory af){
        op.rejectMsg(af);
    }
}
```

S3 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public class S3 extends State {

    @Override
    public int getid(){
        id = 4;
        return id;
    }

    @Override
    public void selectGas(AbstractFactory af, int g){
        op.setPrice(af, g);
    }

    @Override
```

```
    public void cancel(AbstractFactory af){
        op.cancelMsg(af);
        op.returnCash(af);
    }
}
```

S4 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public class S4 extends State {

    @Override
    public int getid(){
        id = 5;
        return id;
    }

    @Override
    public void startPump(AbstractFactory af) {
        op.setInitialValue(af);
        op.readyMsg(af);
    }
}
```

S5 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public class S5 extends State {

    @Override
    public int getid(){
        id = 6;
        return id;
    }

    @Override
    public void pump(AbstractFactory af){
        op.pumpGasUnit(af);
```

```
        op.gasPumpMsg(af);
    }

    @Override
    public void stopPump(AbstractFactory af){
        op.stopMsg(af);
    }
}
```

S6 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public class S6 extends State {

    @Override
    public int getid(){
        id = 7;
        return id;
    }

    @Override
    public void receipt(AbstractFactory af){
        op.printReceipt(af);
        //op.returnCash(af);
    }

    @Override
    public void noReceipt(AbstractFactory af){
        op.returnCash(af);
    }
}
```

b. Strategy pattern

Strategy pattern consists of Op class and actions called by Op classes. Each class has a abstract class and concrete classes

Op class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public class Op {

    public void storeData(AbstractFactory af){
        Data data = af.getData();
        StoreData sd = af.createStoreData();
        sd.storeData(data);
    }

    public void payMsg(AbstractFactory af){
        PayMsg pm = af.createPayMsg();
        pm.displayPayMsg();
    }

    public void storeCash(AbstractFactory af){
        Data data = af.getData();
        StoreCash sc = af.createStoreCash();
        sc.storeCash(data);
    }

    public void displayMenu(AbstractFactory af){
        DisplayMenu dm = af.createDisplayMenu();
        dm.displayMenu();
    }

    public void rejectMsg(AbstractFactory af){
        RejectMsg rm = af.createRejectMsg();
        rm.displayRejectMsg();
    }

    public void setPrice(AbstractFactory af, int g){
        Data data = af.getData();
        SetPrice sp = af.createSetPrice();
        sp.setPrice(data, g);
    }

    public void readyMsg(AbstractFactory af){
```

```
        ReadyMsg rm = af.createReadyMsg();
        rm.displayReadyMsg();
    }

    public void setInitialValue(AbstractFactory af){
        Data data = af.getData();
        SetInitialValue iv = af.createSetInitialValue();
        iv.setValue(data);
    }

    public void pumpGasUnit(AbstractFactory af){
        Data data = af.getData();
        PumpGasUnit gu = af.createPumpGasUnit();
        gu.pumpGasUnit(data);
    }

    public void gasPumpMsg(AbstractFactory af){
        GasPumpMsg pm = af.createGasPumpMsg();
        pm.displayPumpMsg();
    }

    public void stopMsg(AbstractFactory af){
        StopMsg sm = af.createStopMsg();
        sm.displayStopMsg();
    }

    public void printReceipt(AbstractFactory af){
        Data data = af.getData();
        PrintReceipt pr = af.createPrintReceipt();
        pr.displayReceipt(data);
    }

    public void cancelMsg(AbstractFactory af){
        CancelMsg cm = af.createCancelMsg();
        cm.displayCancel();
    }

    public void returnCash(AbstractFactory af){
        Data data = af.getData();
        ReturnCash rc = af.createReturnCash();
        rc.returnCash(data);
    }
}
```

StoreData class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class StoreData {

    public void storeData(Data d){

    }
}
```

StoreData_GP1 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class StoreData_GP1 extends StoreData {

    @Override
    public void storeData(Data d){
        double temp_a = ((Data_GP1) d).getTemp_a();
        double temp_b = ((Data_GP1) d).getTemp_b();

        ((Data_GP1) d).setA(temp_a);
        ((Data_GP1) d).setB(temp_b);

        System.out.println("Gas Pump Activated");
//        System.out.println("the price of regular gas is " + ((Data_GP1) d).getA());
//        System.out.println("the price of regular gas is " + ((Data_GP1) d).getB());
    }
}
```

StoreData_GP2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class StoreData_GP2 extends StoreData {

    @Override
    public void storeData(Data d){
        int temp_a = ((Data_GP2) d).getTemp_a();
```

```
        int temp_b = ((Data_GP2) d).getTemp_b();
        int temp_c = ((Data_GP2) d).getTemp_c();

        ((Data_GP2) d).setA(temp_a);
        ((Data_GP2) d).setB(temp_b);
        ((Data_GP2) d).setC(temp_c);

        System.out.println("Gas Pump Activated");
//        System.out.println("the price of regular gas is " + ((Data_GP2) d).getA());
//        System.out.println("the price of premium gas is " + ((Data_GP2) d).getB());
//        System.out.println("the price of super gas is " + ((Data_GP2) d).getC());
    }
}
```

PayMsg class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class PayMsg {

    public void displayPayMsg(){

    }
}
```

PayMsg_GP1 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class PayMsg_GP1 extends PayMsg {

    @Override
    public void displayPayMsg(){
        System.out.println("Choose the pay type: Credit or Cash!");
        System.out.println("Choose credit press 2");
    }
}
```

PayMsg_GP2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class PayMsg_GP2 extends PayMsg {

    @Override
    public void displayPayMsg(){
        System.out.println("Choose the pay type: Credit or Cash!");
        System.out.println("Choose cash press 2");
    }
}
```

StoreCash class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class StoreCash {

    public void storeCash(Data d){

    }
}
```

StoreCash_GP class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class StoreCash_GP extends StoreCash {

    @Override
    public void storeCash(Data d){
        int temp_cash = ((Data_GP2) d).getCash();

        ((Data_GP2) d).setCash(temp_cash);

        System.out.println("Paycash Succeed!");
    }
}
```

DisplayMenu class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class DisplayMenu {

   public void displayMenu(){

   }
}
```

DisplayMenu_GP1 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class DisplayMenu_GP1 extends DisplayMenu {

   @Override
   public void displayMenu(){
      System.out.println("Select a gas type: Regular or Super\n" +
            "choose regular press 7\n" +
            "choose super press 6");
   }
}
```

DisplayMenu_GP2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class DisplayMenu_GP2 extends DisplayMenu {

   @Override
   public void displayMenu(){
      System.out.println("Select a gas type: Regular or Premium or Super\n" +
            "choose regular press 5\n" +
            "choose premium press 4\n" +
            "choose super press 6");
   }
}
```

RejectMsg class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class RejectMsg {

    public void displayRejectMsg(){

    }
}
```

RejectMsg_GP class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class RejectMsg_GP extends RejectMsg {

    @Override
    public void displayRejectMsg(){
        System.out.println("Credit card is rejected!");
    }
}
```

SetPrice class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class SetPrice {

    public void setPrice(Data d, int g){

    }
}
```

SetPrice_GP1 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
```

```
public class SetPrice_GP1 extends SetPrice {

    @Override
    public void setPrice(Data d, int g){
        double temp_price;
        if(g == 1) {
            temp_price = ((Data_GP1) d).getA();
            ((Data_GP1) d).setPrice(temp_price);
        }else if(g == 3){
            temp_price = ((Data_GP1) d).getB();
            ((Data_GP1) d).setPrice(temp_price);
        }

        System.out.println("Gas type is selected!");
        System.out.println("the price of current gas is " + ((Data_GP1) d).getPrice() + "$ !");
    }
}
```

SetPrice_GP2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class SetPrice_GP2 extends SetPrice {

    @Override
    public void setPrice(Data d, int g) {
        int temp_price;
        if(g == 1){
            temp_price = ((Data_GP2) d).getA();
            ((Data_GP2) d).setPrice(temp_price);
        }else if(g == 2){
            temp_price = ((Data_GP2) d).getB();
            ((Data_GP2) d).setPrice(temp_price);
        }else if(g == 3){
            temp_price = ((Data_GP2) d).getC();
            ((Data_GP2) d).setPrice(temp_price);
        }

        System.out.println("Gas type is selected");
        System.out.println("the price of current gas is " + ((Data_GP2) d).getPrice() + "$ !");
    }
}
```

ReadyMsg class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class ReadyMsg {

    public void displayReadyMsg(){

    }
}
```

ReadyMsg_GP class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class ReadyMsg_GP extends ReadyMsg {

    @Override
    public void displayReadyMsg(){
        System.out.println("Ready for pump");
    }
}
```

SetInitialValue class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class SetInitialValue {

    public void setValue(Data d){

    }
}
```

SetInitialValue_GP1 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
```

```
public class SetInitialValue_GP1 extends SetInitialValue {

    @Override
    public void setValue(Data d){
        ((Data_GP1) d).setG(0);
        ((Data_GP1) d).setTotal(0);
    }
}
```

SetInitialValue_GP2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class SetInitialValue_GP2 extends SetInitialValue {

    @Override
    public void setValue(Data d){
        ((Data_GP2) d).setL(0);
        ((Data_GP2) d).setTotal(0);
    }
}
```

PumpGasUnit class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class PumpGasUnit {

    public void pumpGasUnit(Data d){

    }
}
```

PumpGasUnit_GP1 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class PumpGasUnit_GP1 extends PumpGasUnit {

    @Override
```

```
    public void pumpGasUnit(Data d){
        int temp_g = ((Data_GP1) d).getG();

        ((Data_GP1) d).setG(temp_g + 1);
        ((Data_GP1) d).setTotal(((Data_GP1) d).getPrice() * ((Data_GP1) d).getG());

        System.out.println("Pump gas unit succeed!");
        System.out.println("The current gallon is: " + ((Data_GP1) d).getG() + " G!");
    }
}
```

PumpGasUnit_GP2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class PumpGasUnit_GP2 extends PumpGasUnit {

    @Override
    public void pumpGasUnit(Data d){
        int temp_L = ((Data_GP2) d).getL();

        ((Data_GP2) d).setL(temp_L + 1);
        ((Data_GP2) d).setTotal(((Data_GP2) d).getPrice() * ((Data_GP2) d).getL());

        System.out.println("Pump gas unit succeed!");
        System.out.println("The current liter is: " + ((Data_GP2) d).getL() + " L!");
    }
}
```

GasPumpMsg class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class GasPumpMsg {

    public void displayPumpMsg(){

    }
}
```

GasPumpMsg_GP1 class

```
package com.fujiebao;
```

```
/**
 * Created by vincent on 4/17/17.
 */
public class GasPumpMsg_GP1 extends GasPumpMsg {

    @Override
    public void displayPumpMsg(){
        System.out.println("1 Gallon has been pumped!");
    }
}
```

GasPumpMsg_GP2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class GasPumpMsg_GP2 extends GasPumpMsg {

    @Override
    public void displayPumpMsg(){
        System.out.println("1 Liter has been pumped!");
    }
}
```

StopMsg class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class StopMsg {

    public void displayStopMsg(){

    }
}
```

StopMsg_GP1 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class StopMsg_GP1 extends StopMsg {
```

```
    @Override
    public void displayStopMsg(){
        System.out.println("Stop pump and print receipt!");
    }
}
```

StopMsg_GP2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class StopMsg_GP2 extends StopMsg {

    @Override
    public void displayStopMsg(){
        System.out.println("Stop pump!");
        System.out.println("Do you want to print receipt?");
    }
}
```

PrintReceipt class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class PrintReceipt {

    public void displayReceipt(Data d){

    }
}
```

PrintReceipt_GP1 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class PrintReceipt_GP1 extends PrintReceipt {

    @Override
    public void displayReceipt(Data d){
        double total = ((Data_GP1) d).getTotal();
```

```
        System.out.println("Total price is: " + total + "$ !");
    }
}
```

PrintReceipt_GP2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class PrintReceipt_GP2 extends PrintReceipt {

    @Override
    public void displayReceipt(Data d){
        int l = ((Data_GP2) d).getL();
        int cash = ((Data_GP2) d).getCash();
        double total = l * ((Data_GP2) d).getPrice();
        double r_cash = (double)cash - total;

        System.out.println("Total liter is: " + l + "L!");
        System.out.println("Total price is: " + total + "$ !");
        System.out.println("Return cash is: " + r_cash + "$ !");
    }
}
```

CancelMsg class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class CancelMsg {

    public void displayCancel(){

    }
}
```

CancelMsg_GP class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class CancelMsg_GP extends CancelMsg {
```

```
    @Override
    public void displayCancel(){
        System.out.println("Cancel pump!");
    }
}
```

ReturnCash class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public abstract class ReturnCash {

    public void returnCash(Data d){

    }

}
```

ReturnCash_GP class

```
package com.fujiebao;

/**
 * Created by vincent on 4/17/17.
 */
public class ReturnCash_GP extends ReturnCash {

    @Override
    public void returnCash(Data d){
        double cash = ((Data_GP2) d).getCash();
        double total = ((Data_GP2) d).getTotal();
        double r_cash = cash - total;

        System.out.println("Return cash is: " + r_cash + "$ !");
    }
}
```

c. Abstract factory pattern

Abstract factory pattern consists of AbstractFactory class and concrete classes AbstractFactory_GP1 class and AbstractFactory_GP2 class which executed to create the object of corresponding action classes.

AbstractFactory class

```
package com.fujiebao;

/**
 * Created by vincent on 4/9/17.
 */
public abstract class AbstractFactory {  //abstract class
    Data d;

    //abstract method
    public abstract Data getData();

    public abstract StoreData createStoreData();

    public abstract PayMsg createPayMsg();

    public abstract StoreCash createStoreCash();

    public abstract DisplayMenu createDisplayMenu();

    public abstract RejectMsg createRejectMsg();

    public abstract SetPrice createSetPrice();

    public abstract ReadyMsg createReadyMsg();

    public abstract SetInitialValue createSetInitialValue();

    public abstract PumpGasUnit createPumpGasUnit();

    public abstract GasPumpMsg createGasPumpMsg();

    public abstract StopMsg createStopMsg();

    public abstract PrintReceipt createPrintReceipt();

    public abstract CancelMsg createCancelMsg();

    public abstract ReturnCash createReturnCash();
}
```

AbstractFactory_GP1 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public class AbstractFactory_GP1 extends AbstractFactory {

    public AbstractFactory_GP1(Data data){
        d = data;
    }

    @Override
    public Data getData(){
        return d;
    }

    @Override
    public StoreData createStoreData(){
        return new StoreData_GP1();
    }

    public PayMsg createPayMsg(){
        return new PayMsg_GP1();
    }

    public StoreCash createStoreCash(){
        return new StoreCash_GP();
    }

    public DisplayMenu createDisplayMenu(){
        return new DisplayMenu_GP1();
    }

    public RejectMsg createRejectMsg(){
        return new RejectMsg_GP();
    }

    public SetPrice createSetPrice(){
        return new SetPrice_GP1();
    }

    public ReadyMsg createReadyMsg(){
        return new ReadyMsg_GP();
    }
```

```
    public SetInitialValue createSetInitialValue(){
        return new SetInitialValue_GP1();
    }

    public PumpGasUnit createPumpGasUnit(){
        return new PumpGasUnit_GP1();
    }

    public GasPumpMsg createGasPumpMsg(){
        return new GasPumpMsg_GP1();
    }

    public StopMsg createStopMsg(){
        return new StopMsg_GP1();
    }

    public PrintReceipt createPrintReceipt(){
        return new PrintReceipt_GP1();
    }

    public CancelMsg createCancelMsg(){
        return new CancelMsg_GP();
    }

    public ReturnCash createReturnCash(){
        return new ReturnCash_GP();
    }
}
```

AbstractFactory_GP2 class

```
package com.fujiebao;

/**
 * Created by vincent on 4/14/17.
 */
public class AbstractFactory_GP2 extends AbstractFactory{

    public AbstractFactory_GP2(Data data){
        d = data;
    }

    public Data getData(){
        return d;
    }

    public StoreData createStoreData(){
        return new StoreData_GP2();
    }

    public PayMsg createPayMsg(){
        return new PayMsg_GP2();
    }

    public StoreCash createStoreCash(){
        return new StoreCash_GP();
    }

    public DisplayMenu createDisplayMenu(){
        return new DisplayMenu_GP2();
    }

    public RejectMsg createRejectMsg(){
        return new RejectMsg_GP();
    }

    public SetPrice createSetPrice(){
        return new SetPrice_GP2();
    }

    public ReadyMsg createReadyMsg(){
        return new ReadyMsg_GP();
    }

    public SetInitialValue createSetInitialValue(){
        return new SetInitialValue_GP2();
```

```
    }

    public PumpGasUnit createPumpGasUnit(){
        return new PumpGasUnit_GP2();
    }

    public GasPumpMsg createGasPumpMsg(){
        return new GasPumpMsg_GP2();
    }

    public StopMsg createStopMsg(){
        return new StopMsg_GP2();
    }

    public PrintReceipt createPrintReceipt(){
        return new PrintReceipt_GP2();
    }

    public CancelMsg createCancelMsg(){
        return new CancelMsg_GP();
    }

    public ReturnCash createReturnCash(){
        return new ReturnCash_GP();
    }
}
```