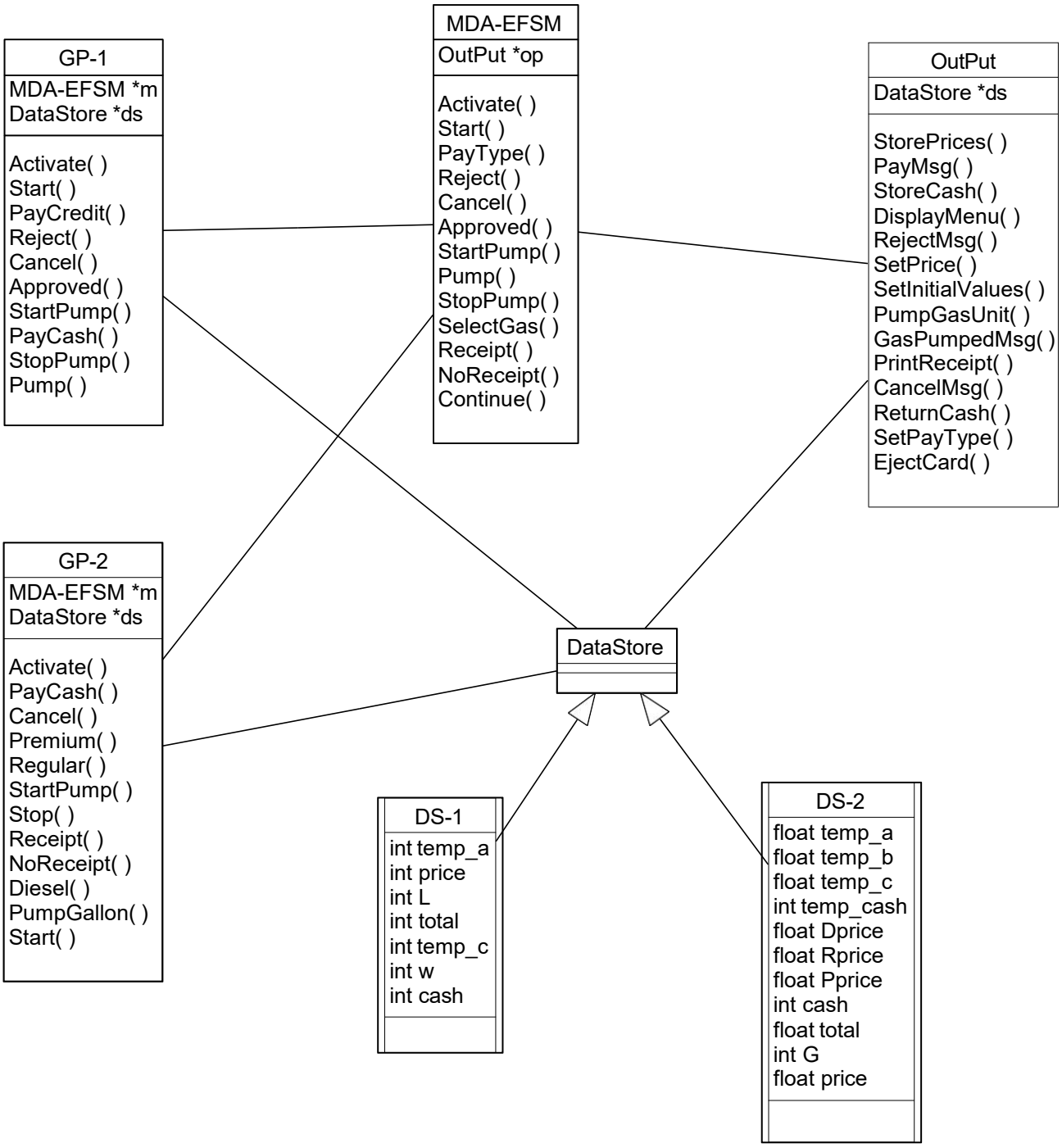
Final Project Report

Software System Architecture (CS586)
(GasPump System)

Belthangady Akash Vittaldas Narayana Pai
(A20560317)

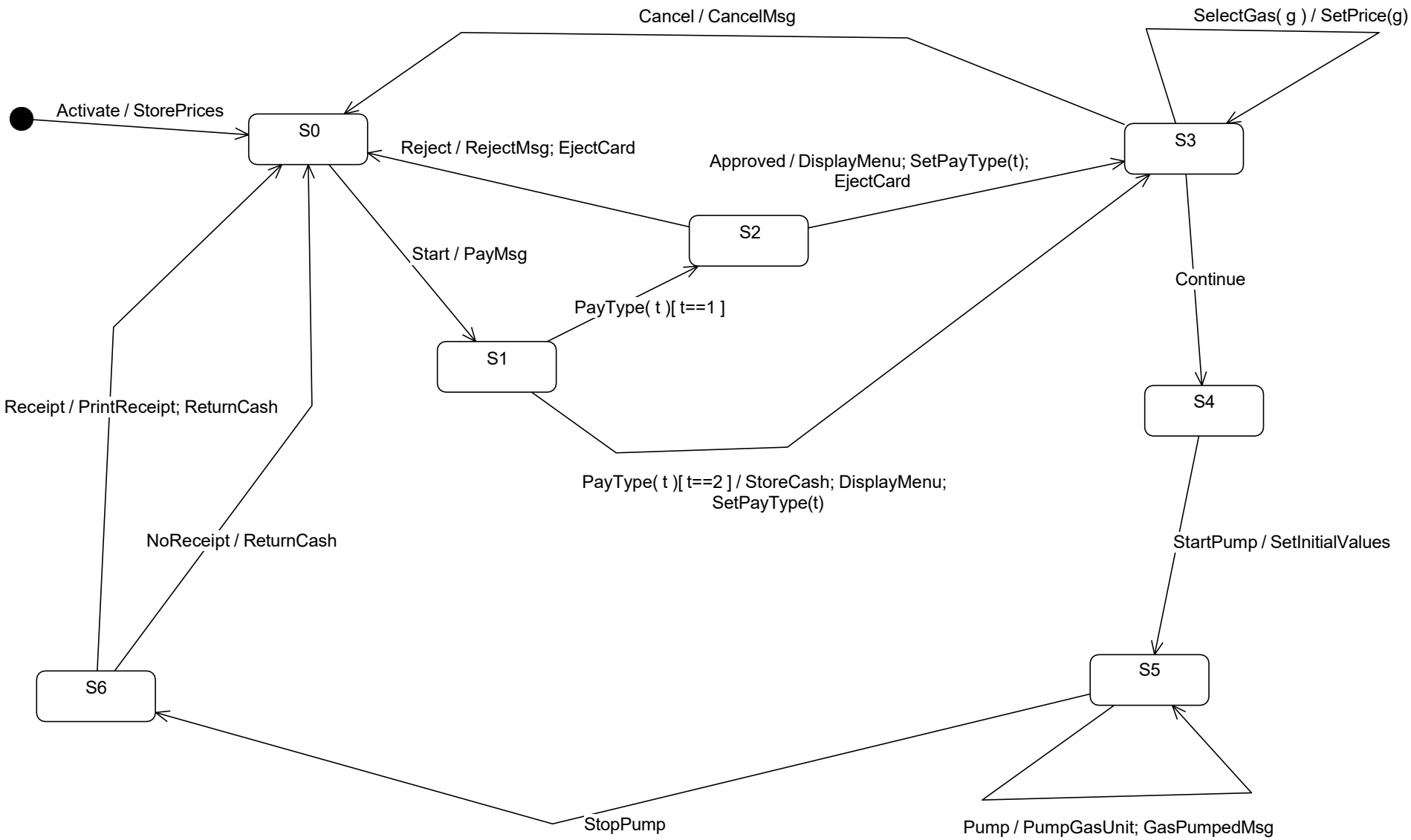

GP-1
MDA-EFSM *m
DataStore *ds
Activate( )
Start( )
PayCredit( )
Reject( )
Cancel( )
Approved( )
StartPump( )
PayCash( )
StopPump( )
Pump( )
MDA-EFSM
OutPut *op
Activate( )
Start( )
PayType( )
Reject( )
Cancel( )
Approved( )
StartPump( )
Pump( )
StopPump( )
SelectGas( )
Receipt( )
NoReceipt( )
Continue( )
OutPut
DataStore *ds
StorePrices( )
PayMsg( )
StoreCash( )
DisplayMenu( )
RejectMsg( )
SetPrice( )
SetInitialValues( )
PumpGasUnit( )
GasPumpedMsg( )
PrintReceipt( )
CancelMsg( )
ReturnCash( )
SetPayType( )
EjectCard( )
GP-2
MDA-EFSM *m
DataStore *ds
Activate( )
PayCash( )
Cancel( )
Premium( )
Regular( )
StartPump( )
Stop( )
Receipt( )
NoReceipt( )
Diesel( )
PumpGallon( )
Start( )
DataStore
DS-1
int temp_a
int price
int L
int total
int temp_c
int w
int cash
DS-2
float temp_a
float temp_b
float temp_c
int temp_cash
float Dprice
float Rprice
float Pprice
int cash
float total
int G
float price

MDA-EFSM Events:

Activate() Start()
PayType(int t) //credit: t=1; cash: t=2;
Reject()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int g) // Regular: g=1; Diesel: g=3; Premium: g=2
Receipt()
NoReceipt()
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) // set the price for the gas identified by *g* identifier as in SelectGas(int g);
SetInitialValues() // set *G* (or *L*) and *total* to 0;
PumpGasUnit() // disposes unit of gas and counts # of units disposed and computes Total
GasPumpedMsg() // displays the amount of disposed gas
PrintReceipt() // print a receipt
CancelMsg() // displays a cancellation message
ReturnCash() // returns the remaining cash
SetPayType(t) // Stores pay type *t* to variable *w* in the data store
EjectCard() // Card is ejected



MDA-EFSM for Gas Pumps

Operations of the Input Processor
(GasPump-1)

```
Activate(int a) {
        if (a>0) {
            d->temp_a=a;
            m->Activate()
        }
}

Start() {
        m->Start();
}
PayCash(int c) {
        if (c>0) {
            d->temp_c=c;
            m->PayType(2)
        }
}

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

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

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

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

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

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

Notice:
cash: contains the value of cash deposited
price: contains the price of the gas
L: contains the number of liters already pumped
w: pay type flag (cash: w=0; credit: w=1)
cash, *L*, *price, w:* are in the data store
m: is a pointer to the MDA-EFSM object
d: is a pointer to the Data Store object

Operations of the Input Processor (GasPump-2)

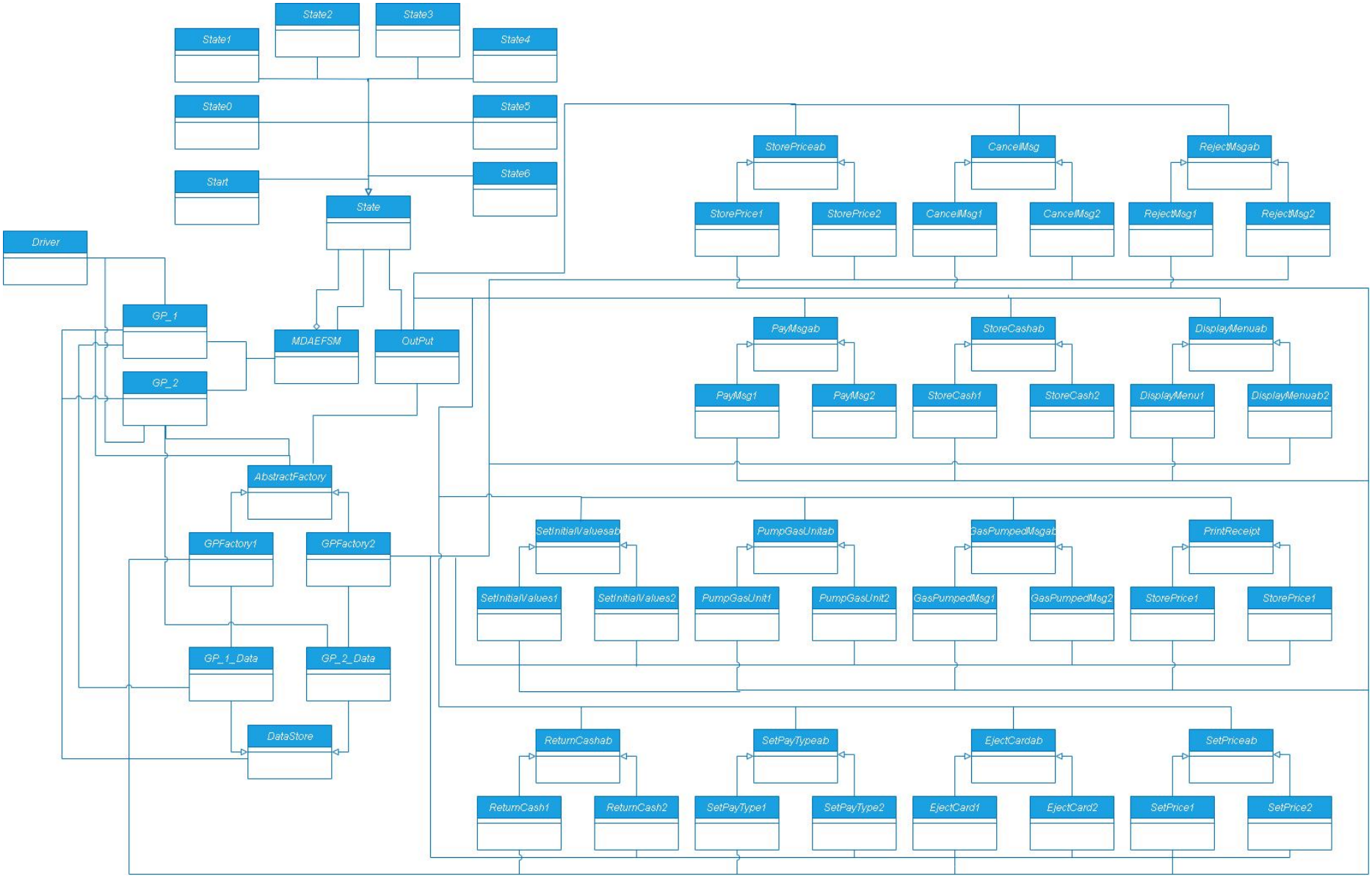
```
Activate(float a, float b, float c) {
        if ((a>0)&&(b>0)&&(c>0))
            { d->temp_a=a;
            d->temp_b=b;
            d->temp_c=c
            m->Activate()
        }
}

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


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

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

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

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

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

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

PumpGallon() {
if (d->cash < d->price*(d->G+1))
            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
G: contains the number of Gallons already pumped

cash, *G*, *price* are in the data store
m: is a pointer to the MDA-EFSM object
d: is a pointer to the Data Store object

Class Diagram : (this is to show the State pattern Abstract Factory Pattern and Strategy Pattern used. Detailed description of the Classes are further down in this Project Report)



Detailed Description of each class in the Project

Package- MainClass

Class GP_1

This class has functions that are allowed in GasPump 1 which intern calls the reqired meta events of MDEFSM class in the mean time loads and retirives data from class GP_1_Data which is the Data store class of GasPump 1

Refferences/variables:

MDAEFSM m
DataStore ds
GP_1_Data d
GPFactory1 af

Methods:

- GP_1() - constructor to initialize the above refferences
- Activate(int) -stores the the price of fuel in temp_a in GP_1_Data.and calls the activate meta-event in MDAEFSM class.
- Start()-Call the start() meta-event of the MDAEFSM class
- PayCredit()-Calsl the payType(1) meta-event of the MDAEFSM class
- Reject()-Calsl the reject() meta-event of the MDAEFSM class
- Cancel()- Calls the cancel() meta-event of the MDAEFSM class
- Approved()- Calls the approve() meta-event of the MDAEFSM class
- PayCash()-saves the cash enterted in temp_ca at GP_1_Data then Call the payType(2) meta-event of the MDAEFSM class

- StartPump()-first calls Continue() meta-event to change the state to 4 then calls the meta event startPump()

- Pump()-Based on the if condition either calls stopPump() and recipt()
- Of MDAEFSM class or pump() in MDAEFSM class.

- StopPump()- Calls the stopPump() meta-event of the MDAEFSM class

Class GP_2

This class has functions that are allowed in GasPump 2 which intern calls the reqired meta events of MDEFSM class in the mean time loads and retirives data from class GP_2_Data which is the Data store class of GasPump 2

Refferences/variables

MDAEFSM m
DataStore ds
GP_2_Data d
GPFactory2 af

Methods:

- GP_2() - constructor to initialize the above refferences

- Activate(float,float,float)-stores the the price of fuel in temp_a,temp_b,_temp_c in GP_2_Data.and calls the activate meta-event in MDAEFSM class.

- Start()-Call the start() meta-event of the MDAEFSM class

- Cancel()- Calls the cancel() meta-event of the MDAEFSM class

- PayCash()-saves the cash enterted in temp_ca at GP_2_Data then Call the payType(2) meta-event of the MDAEFSM class

- StartPump()- calls the meta event startPump() of MDAEFSM class

- Regular()-calls the selectGas(1) and then calls Continue() meta event of the MDAEFSM class

- Premium()-calls the selectGas(2) and then calls Continue() meta event of the MDAEFSM class

- Diesel()-calls the selectGas(3) and then calls Continue() meta event of the MDAEFSM class

- PumpGallon()-Based on the if condition either calls stopPump() Of MDAEFSM class or pump() in MDAEFSM class.

- Stop()- Calls the stopPump() meta-event of the MDAEFSM class

- Receipt()-Calls the receipt() meta-event of the MDAEFSM class

- NoReceipt()-Calls the noreceipt() meta-event of the MDAEFSM class

Class MDAEFSM

De centralised State pattern is used
Holds the meta events that are called by either of the GasPumps this intern calls the nessary functions using State pattern keeping Class state as main state class

Refferences/variables

State s
State LS[8]

Methods:

- ChangeState()- called by functions in each of the state classes to update the change of state after a particular operation.

- Activate()- calls Activate() function with the current State class refference

- Start()- calls Start() function with the current State class refference

- PayType(int t)- calls PayType(int t) function with the current State class refference (t=1 Credit Card t=2 Cash)

- Approved()- calls Approved() function with the current State class refference

- Reject()- calls Reject() function with the current State class refference

- Cancel()- calls Cancel() function with the current State class refference

- SelectGas(int g)- calls SelectGas(int g) function with the current State class refference (g=1 Regular , g=2 Preminum, g= 3 Diesel)

- StartPump()- calls StartPump() function with the current State class refference

- Pump()- calls Pump() function with the current State class refference

- StopPump()- calls StopPump() function with the current State class refference

- Receipt()- calls Receipt() function with the current State class refference

- NoReceipt()- calls NoReceipt() function with the current State class refference

- Continue()- calls Continue() function with the current State class refference

- getOP()- called bt the state class to get the current Output class instance to acess the required strategies

- setOP() - creates the instance of OutPut class and loads to op variable

Class OutPut

This class is the general output processor for the gas pump system. Abstract Factory design pattern is used to link the required meta Action. Strategy Pattern is used to implement the same Meta Action This works By combining Strategy and Abstract Factory Design Pattern.

Refferences/variables

AbstractFactory af

Methods:

- OutPut()-constructor funtion that sets thefactory reffrerence to af
- StorePrice()- calls StorePrice() function from strategy StorePriceab based on the refference in af
- PayMsg()- calls PayMsg() function from strategy PayMsgab based on the refference in af
- StoreCash()- calls StoreCash() function from strategy StoreCashab
- based on the refference in af
- DisplayMenu()- calls DisplayMenu() function from strategy DisplayMenuab based on the refference in af
- RejectMsg()- calls RejectMsg() function from strategy RejectMsgab based on the refference in af
- PumpGasUnit()- calls PumpGasUnit() function from strategy PumpGasUnitab based on the refference in af
- GasPumpedMsg()- calls GasPumpedMsg() function from strategy GasPumpedMsgab based on the refference in af
- PrintRecipt()- calls PrintRecipt() function from strategy PrintReciptab based on the refference in af

- CancelMsg()- calls CancelMsg() function from strategy CancelMsgab based on the refference in af

- ReturnCash()- calls ReturnCash() function from strategy ReturnCashab based on the refference in af
- SetPrice()- calls SetPrice() function from strategy SetPriceab based on the refference in af

- SetPatType(intt)- calls SetPatType() function from strategy SetPatTypeab based on the refference in af

- EjectCard()- calls EjectCard() function from strategy EjectCardab based on the refference in af

Package State

Class State
(State Pattern)
Abstract class which is inherited by each of the state classes the functions in this listed below are generally abstract but for our convinience so as to make the user know that the particular operation is not allowed in the current state we use function Incorrect this is in all classes in State Class which will only be executed if that partcular function is not overiden by a inherited state class .

Refferences/variables:

MDAEFSM m

Methods: explanation of methods in given in State class Description

- Activate()
- Start()
- PayType(int t)
- Reject()
- Cancle()
- Approved()
- StartPump()
- Pump()
- StopPump()

- SelectGas(int g)
- Recipt()
- NoRecipt()
- Continue()
- Incorrect()-called when ever the above functions are not overidden by the inhereted state classes

- State()-constructor to set m as a reffrence to MDAEFSM class

Class Start

This is the first class where the state system lies before the the activation of gaspump system this inherits State class

Refferences/variables

MDAEFSM m
OutPut Op

Methods :

Activate()- calls StorePrice() function of OutPut class and then calls the ChangeState() to update the state to State0

Class State

This is the 2nd class where the state system lies after the the activation of gaspump system and activation . this inherits State class

Refferences/variables

MDAEFSM m
OutPut Op

Methods :

Start()- calls PayMsg() function of OutPut class and then calls the ChangeState() to update the state to State1

Class State1

This is the 3rd class where the state system lies after the the activation of gaspump system and activation . this inherits State class

Refferences/variables

MDAEFSM m
OutPut Op

Methods :

PayType()- depending of the payment type it either calls StoreCash() , DisplayMenu() SetPayType() function of OutPut class and then calls the ChangeState() to update the state to State3(cash)
Or
Just calls ChangeState() to update the state to State2(credit card)

Class State2

This is the 4rd class where the state system lies after the the activation of gaspump system and activation if the payment is by Credit Card . this inherits State class

Refferences/variables

MDAEFSM m
OutPut Op

Methods :

Approved()- calls StoreCash() , DisplayMenu() SetPayType() function of OutPut class and then calls the ChangeState() to update the state to State3(cash)

Class State3

this inherits State class and has the below methods

Refferences/variables

MDAEFSM m

OutPut Op

Methods :

Continue() - changes the the state to state4 by calling ChangeState() of MDAEFSM class

SelectGas(int g)- calls the SetPrice(g) function of the OutPut class

Cancel()-calls the CancleMsg() function of the OutPut class ans then changes the state to State0 using ChangeState()

Class State4

this inherits State class and has the below methods

Refferences/variables

MDAEFSM m
OutPut Op

Methods :

StartPump()- calls the SetInitialValues() function of the OutPut class then changes the state to State5 using ChangeState()

Class State5

this inherits State class and has the below methods

Refferences/variables

MDAEFSM m
OutPut Op

Methods :

StoptPump()- changes the state to State6 using ChangeState()

Pump()- calls the PumpGasUnit() and GasPumpedMsg() function of the OutPut class

Class State6

this inherits State class and has the below methods

Refferences/variables

MDAEFSM m
OutPut Op

Methods :

Receipt()- calls the PrintRecipt(), ReturnCash() and changes the state to State0 using ChangeState()

NoReceipt()- calls the ReturnCash() and changes the state to State0 using ChangeState()

Package AbstractFactory

Class AbstractFactory
(Abstract Factory Pattern)
Abstract class that is inherietd by the Concret Factory classes GPFactory1 and GPFactory2 all the methods present in this class are abstract and are defined in the inherited classes

Methods :
- getData()
- getStorePrices()
- getPayMsg()
- getStoreCas()
- getDisplayMenu()
- getRejectMsg()
- getSetPrice()
- getSetInitiaValues()
- getPumpGasUnit()
- getGasPumpedMsg()
- getPrintReceipt()
- getCancelMsg()
- getReturnCash()
- getSetPayType()
- getEjectCard()

Class GPFactory1

This class picks the instances related to Gas Pump 1 of the required strategy class and returns it to Output class and pics the required DataStore class ans returns where ever needed.

Refference/Variables:

DataStore ds;

Methods :

- getData()-returns instance of GP_1_Data
- getStorePrices()-returns instance of StorePrice1
- getPayMsg()-returns instance of PayMsg1
- getStoreCas()-returns instance of StoreCash1
- getDisplayMenu()-returns instance of DisplauMenu1
- getRejectMsg()-returns instance of RejectMsg1
- getSetPrice()-returns instance of SetPrice1
- getSetInitiaValues()-returns instance of SetInitialValues1
- getPumpGasUnit()-returns instance of PumpGasUnit1
- getGasPumpedMsg()-returns instance of GasPumpedMsg1
- getPrintReceipt()-returns instance of Receipt1
- getCancelMsg()-returns instance of CancelMsg1
- getReturnCash()-returns instance of ReturnCash1
- getSetPayType()-returns instance of SetPayType1
- getEjectCard()-returns instance of EjectCard1

Class GPFactory2

This class picks the instances related to Gas Pump 2 of the required strategy class and returns it to Output class and pics the required DataStore class ans returns where ever needed.

Refference/Variables:

DataStore ds;

Methods :

- getData()-returns instance of GP_2_Data
- getStorePrices()-returns instance of StorePrice2
- getPayMsg()-returns instance of PayMsg2
- getStoreCas()-returns instance of StoreCash2
- getDisplayMenu()-returns instance of DisplauMenu2
- getRejectMsg()-returns instance of RejectMsg2
- getSetPrice()-returns instance of SetPrice2
- getSetInitiaValues()-returns instance of SetInitialValues2
- getPumpGasUnit()-returns instance of PumpGasUnit2
- getGasPumpedMsg()-returns instance of GasPumpedMsg2
- getPrintReceipt()-returns instance of Receipt2
- getCancelMsg()-returns instance of CancelMsg2
- getReturnCash()-returns instance of ReturnCash2
- getSetPayType()-returns instance of SetPayType2
- getEjectCard()-returns instance of EjectCard2

Class DataStore

This is a abstract class which is inherited by GP_1_Data and GP_2_Data the class is left empty as there are very less commom methods hence all the Data getters setters and related variables are in the inherited class

Class GP_1_Data

Variables

int temp_a
int price
int L
int total
int temp_ca
int w
int cash

Methods :

There are 2 type of functions in this class get_X and set_X where X represents each of the variable given above

get_X()- this returns the required varable X where ever it is called

set_X(x)-sets the value of 'x' to the required varable X

Class GP_2_Data

Variables

float temp_a
float temp_b
float temp_c
float price
float Rprice
float Dprice
float Pprice
int G
float total

int temp_ca
int cash

Methods :

There are 2 type of functions in this class get_X and set_X where X represents each of the variable given above

get_X()- this returns the required varable X where ever it is called

set_X(x)-sets the value of 'x' to the required varable X

Package ActionStrategies
(Strategy Patterns)

Package CancelMsg

Class CancelMsgab

This is a abstract class inherited by concret strategy classes CancelMsg1 and CancelMsg2

Method

CancleMsg() - abstract function no implimentation

Class CancelMsg1

This is a concret strategy class of GasPump 1 for function CancelMsg()

Method

CancleMsg() - print cancel Msg

Class CancelMsg2

This is a concret strategy class of GasPump 2 for function CancelMsg()

Method

CancleMsg() - print cancel Msg

Package DisplayMenu

Class DisplayMenuab

This is a abstract class inherited by concret strategy classes DisplayMenu1 and DisplayMenu2

Method

DisplayMenu() - abstract function no implimentation

Class DisplayMenu1

This is a concret strategy class of GasPump 1 for function DisplayMenu()

Method

DisplayMenu() - Displays the Menu with price of fuel

Class DisplayMenu2

This is a concret strategy class of GasPump 2 for function DisplayMenu()

Method

DisplayMenu() - Displays the Menu with price of fuel

Package EjectCard

Class EjectCardab

This is a abstract class inherited by concret strategy classes EjectCard1 and EjectCard2

Method

EjectCard() - abstract function no implimentation

Class EjectCard1

This is a concret strategy class of GasPump 1 for function EjectCard()

Method

EjectCard() - Eject the Credit card after authentication

Class EjectCard2

This is a concret strategy class of GasPump 2 for function EjectCard()

Method

EjectCard() - does nothing as GasPump2 has no EjectCard action

Package GasPumpedMsg

Class GasPumpedMsgab

This is a abstract class inherited by concret strategy classes GasPumpedMsg1 and GasPumpedMsg2

Method

GasPumpedMsg() - abstract function no implimentation

Class GasPumpedMsg1

This is a concret strategy class of GasPump 1 for function GasPumpedMsg()

Method

GasPumpedMsg() - print GasPumped Msg

Class GasPumpedMsg2

This is a concret strategy class of GasPump 2 for function GasPumpedMsg()

Method

GasPumpedMsg() - print GasPumpedMsg

Package PayMsg

Class PayMsgab

This is a abstract class inherited by concret strategy classes PayMsg1 and PayMsg2

Method

PayMsg() - abstract function no implimentation

Class PayMsg1

This is a concret strategy class of GasPump 1 for function PayMsg()

Method

PayMsg() - print PayMsg

Class PayMsg2

This is a concret strategy class of GasPump 2 for function PayMsg()

Method

PayMsg() - print PayMsg

Package PrintRecipt

Class PrintReciptab

This is a abstract class inherited by concret strategy classes PrintRecipt1 and PrintRecipt2

Method

PrintRecipt() - abstract function no implimentation

Class PrintRecipt1

This is a concret strategy class of GasPump 1 for function PrintRecipt()

Method

PrintRecipt() - Print the Receipt

Class PrintRecipt2

This is a concret strategy class of GasPump 2 for function PrintRecipt()

Method

PrintRecipt() - Print the Receipt

Package PumpGasUnit

Class PumpGasUnitab

This is a abstract class inherited by concret strategy classes PumpGasUnit1 and PumpGasUnit2

Method

PumpGasUnit() - abstract function no implimentation

Class PumpGasUnit1

This is a concret strategy class of GasPump 1 for function PumpGasUnit()

Method

PumpGasUnit() - get the vales of price and fuel quantity and update total and fuel quantity

Class PumpGasUnit2

This is a concret strategy class of GasPump 2 for function PumpGasUnit()

Method

PumpGasUnit() - get the vales of price and fuel quantity and update total and fuel quantity

Package RejectMsg

Class RejectMsgab

This is a abstract class inherited by concret strategy classes RejectMsg1 and RejectMsg2

Method

RejectMsg() - abstract function no implimentation

Class RejectMsg1

This is a concret strategy class of GasPump 1 for function RejectMsg()

Method

RejectMsg() - print RejectMsg

Class RejectMsg2

This is a concret strategy class of GasPump 2 for function RejectMsg()

Method

RejectMsg() - dosent do anything because GasPump2 dosent have this action

Package ReturnCash

Class ReturnCashab

This is a abstract class inherited by concret strategy classes ReturnCash1 and ReturnCash2

Method

ReturnCash() - abstract function no implimentation

Class ReturnCash1

This is a concret strategy class of GasPump 1 for function ReturnCash()

Method

ReturnCash() - does nothing as GasPump1 dosent have ReturnCash action

Class ReturnCash2

This is a concret strategy class of GasPump 2 for function ReturnCash()

Method

ReturnCash() - calculates the total and return the balance for Gaspump 2

Package SetInitialValues

Class SetInitialValuesab

This is a abstract class inherited by concret strategy classes SetInitialValues1 and SetInitialValues2

Method

SetInitialValues() - abstract function no implimentation

Class SetInitialValues1

This is a concret strategy class of GasPump 1 for function SetInitialValues()

Method

SetInitialValues() - sets L and total to zero in GP_1_Data

Class SetInitialValues2

This is a concret strategy class of GasPump 2 for function SetInitialValues()

Method

SetInitialValues() - sets G and total to zero in GP_2_Data

Package SetPayType

Class SetPayTypeab

This is a abstract class inherited by concret strategy classes SetPayType1 and SetPayType2

Method

SetPayType(int w) - abstract function no implimentation

Class SetPayType1

This is a concret strategy class of GasPump 1 for function SetPayType(int w)

Method

SetPayType(int w) - sets the value of W in GP_1_Data

Class SetPayType2

This is a concret strategy class of GasPump 2 for function SetPayType(int w)

Method

SetPayType(int w) - dose nothing as SetPayType action is not present in GasPump 2

Package SetPrice

Class SetPriceab

This is a abstract class inherited by concret strategy classes SetPrice1 and SetPrice2

Method

SetPrice(int g) - abstract function no implimentation

Class SetPrice1

This is a concret strategy class of GasPump 1 for function SetPrice(int g)

Method

SetPrice(int g) - dose nothing as SetPayType action is not present in GasPump 2

Class SetPrice2

This is a concret strategy class of GasPump 2 for function SetPrice(int g)

Method

SetPrice(int g) - Based on the value of g load the value of selected fuel price to price in GP_2_Data

Package StoreCash

Class StoreCashab

This is a abstract class inherited by concret strategy classes StoreCash1 and StoreCash2

Method

StoreCash() - abstract function no implimentation

Class StoreCash1

This is a concret strategy class of GasPump 1 for function StoreCash()

Method

StoreCash() - sets the cash value in GP_1_Data

Class StoreCash2

This is a concret strategy class of GasPump 2 for function StoreCash()

Method

StoreCash() - sets the cash value in GP_2_Data

Package StorePrice

Class StorePriceab

This is a abstract class inherited by concret strategy classes StorePrice1 and StorePrice2

Method

StorePrice() - abstract function no implimentation

Class StorePrice1

This is a concret strategy class of GasPump 1 for function StorePrice()

Method

StorePrice() - sets the price value in GP_1_Data from the temperory variable

Class StorePrice2

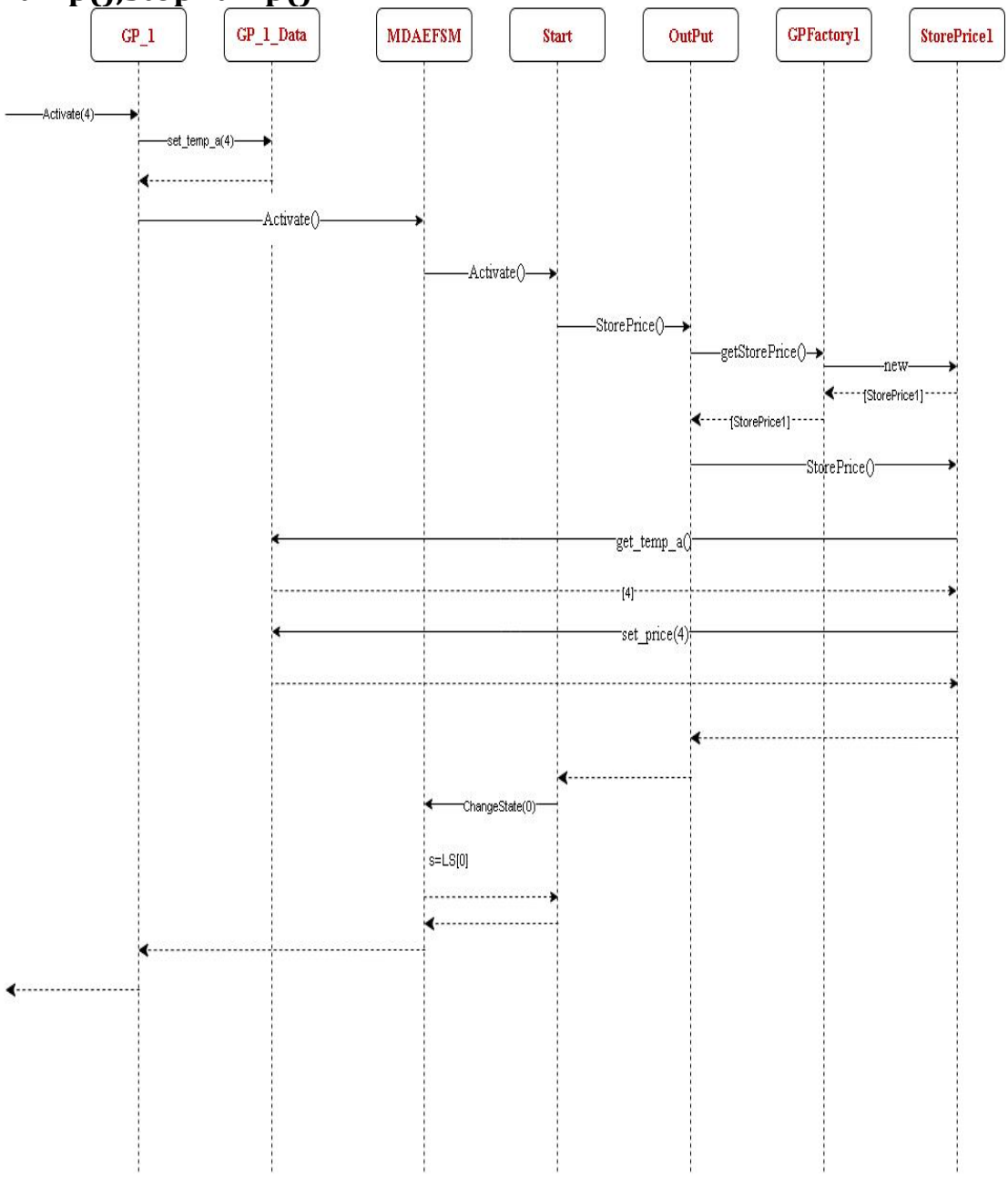
This is a concret strategy class of GasPump 2 for function StorePrice()

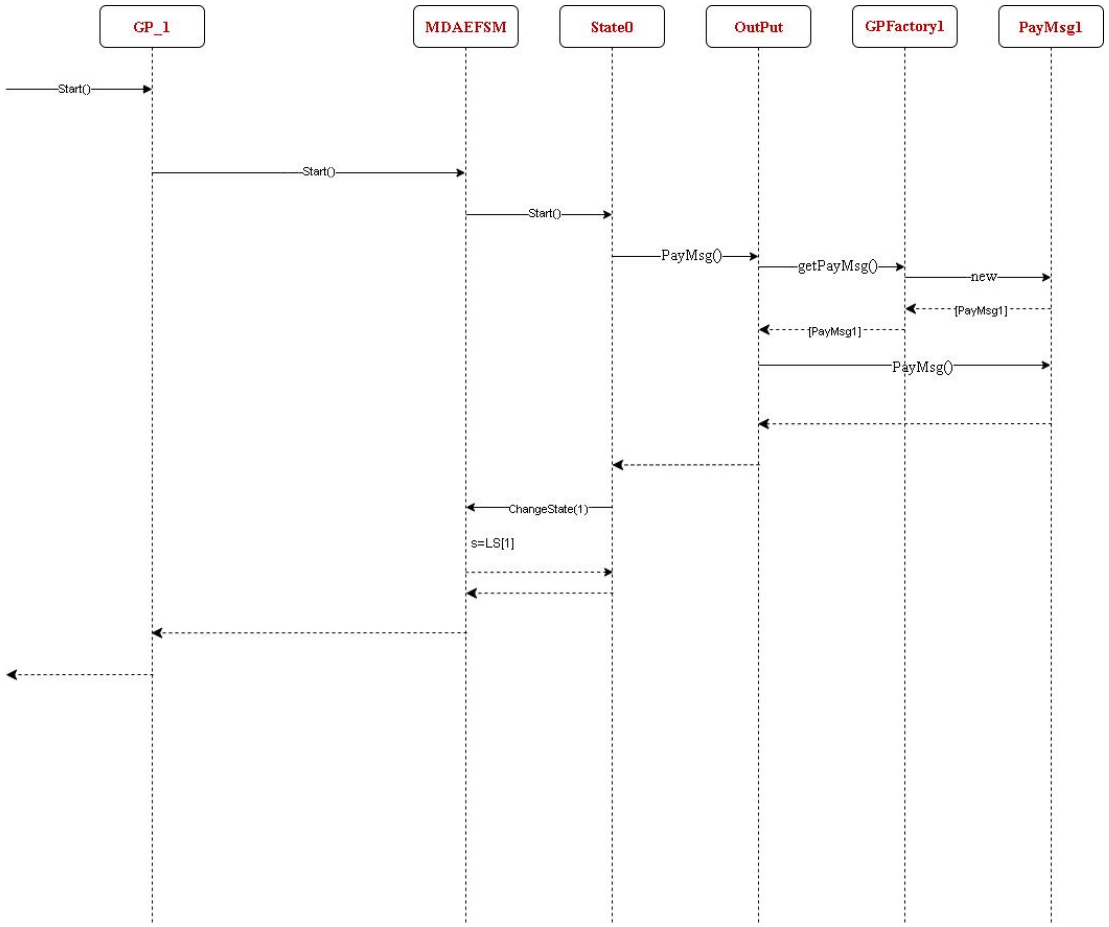
Method

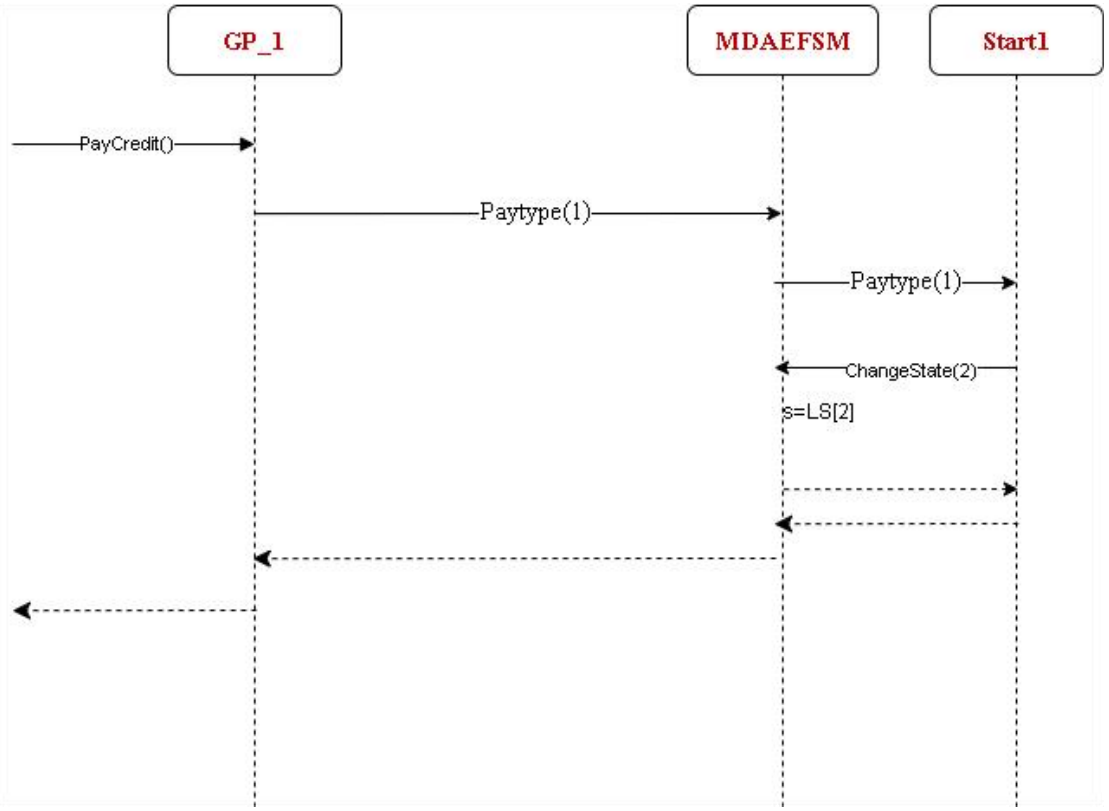
StorePrice() - sets the Rprice,Pprice,Dprice value in GP_2_Data from the temperory variable

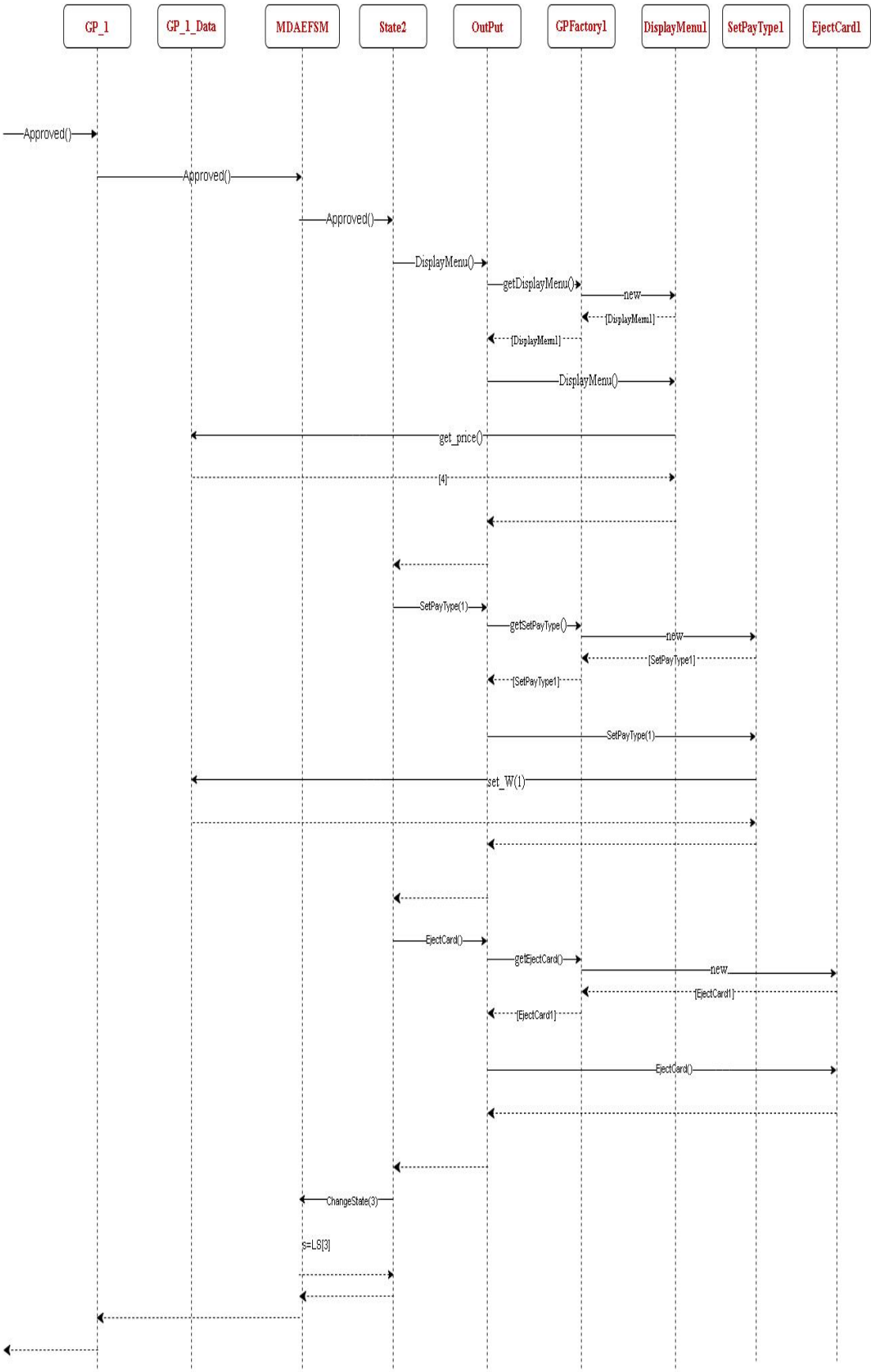
Sequence Diagram 1

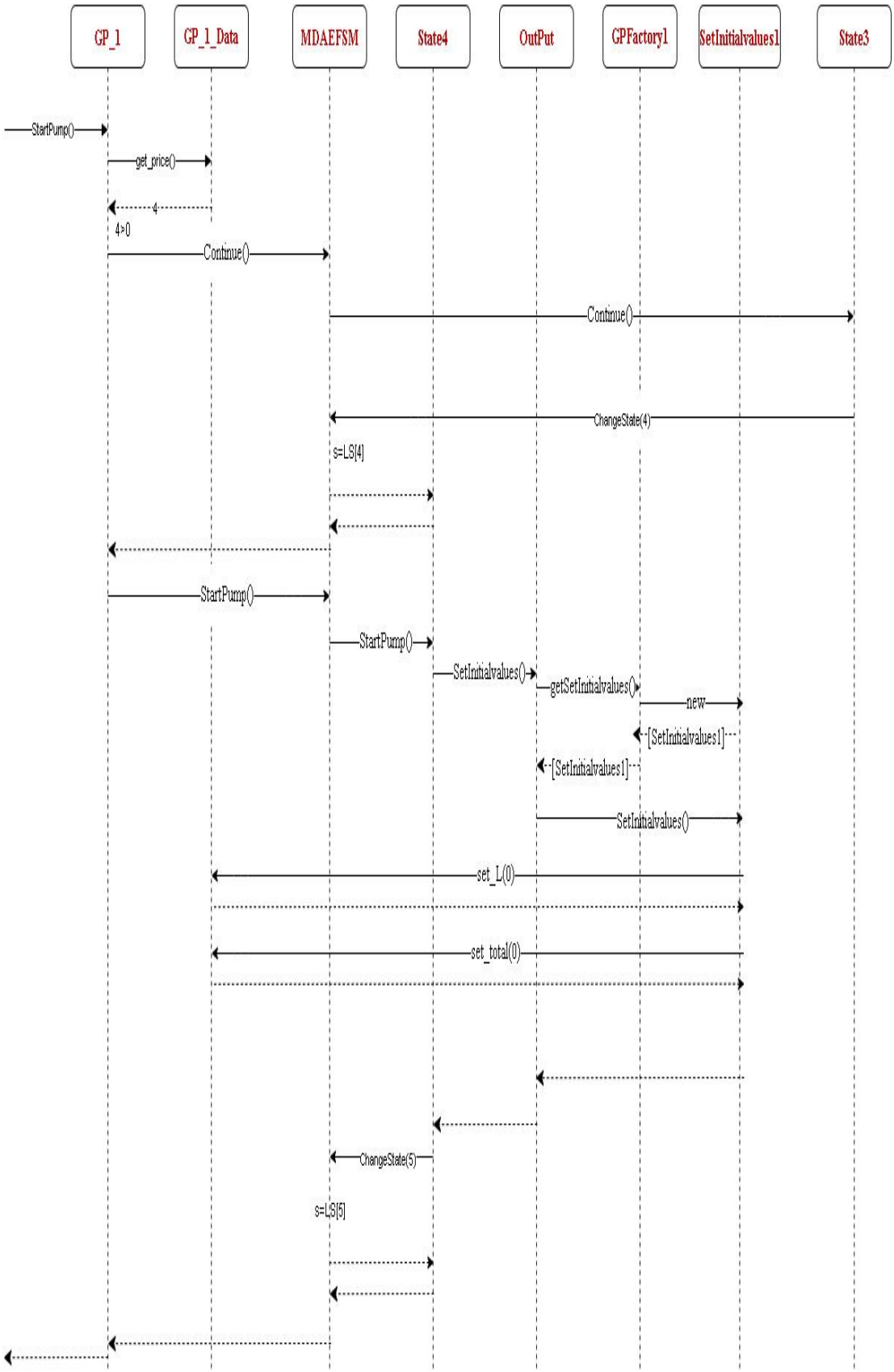
a. Scenario-I should show how one liter of gas is disposed in the Gas Pump GP-1 component, i.e., the following sequence of operations is issued:
Activate(4), Start(), PayCredit(), Approved(), StartPump(), Pump(),StopPump()

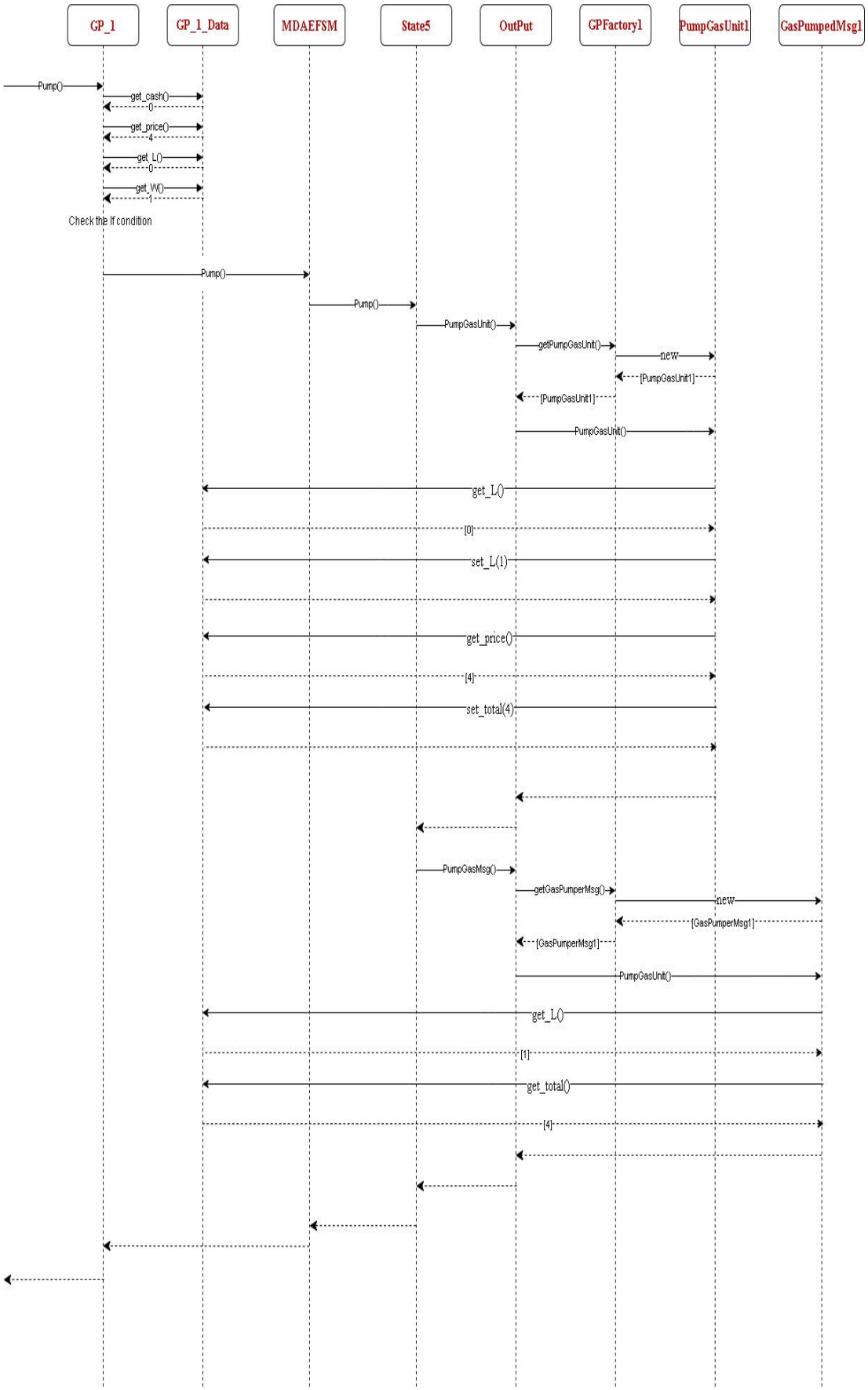


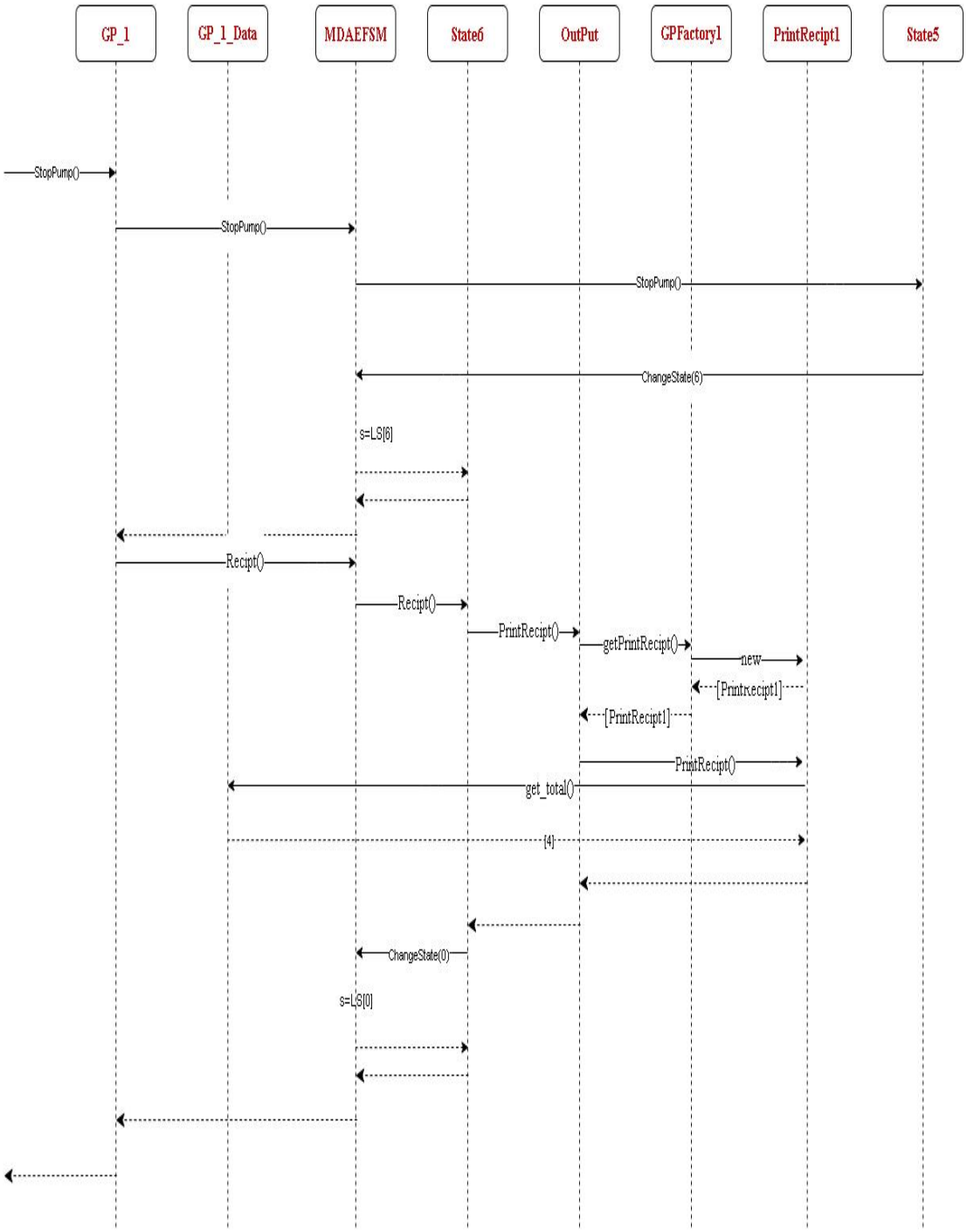

GP_1
MDAEFSM
State0
OutPut
GPFactory1
PayMsg1
Start()
Start()
Start()
PayMsg()
getPayMsg()
new
[PayMsg1]
[PayMsg1]
PayMsg()
ChangeState(1)
s=LS[1]


GP_1
MDAEFSM
Start1
PayCredit()
Paytype(1)
Paytype(1)
ChangeState(2)
s=LS[2]


GP_1
GP_1_Data
MDAEFSM
State2
OutPut
GPFactory1
DisplayMenu1
SetPayType1
EjectCard1
Approved()
Approved()
Approved()
DisplayMenu()
getDisplayMenu()
new
[DisplayMem1]
[DisplayMem1]
DisplayMenu()
get_price()
[4]
SetPayType(1)
getSetPayType()
new
[SetPayType1]
[SetPayType1]
SetPayType(1)
set_W(1)
EjectCard()
getEjectCard()
new
[EjectCard1]
[EjectCard1]
EjectCard()
ChangeState(3)
s=LS[3]

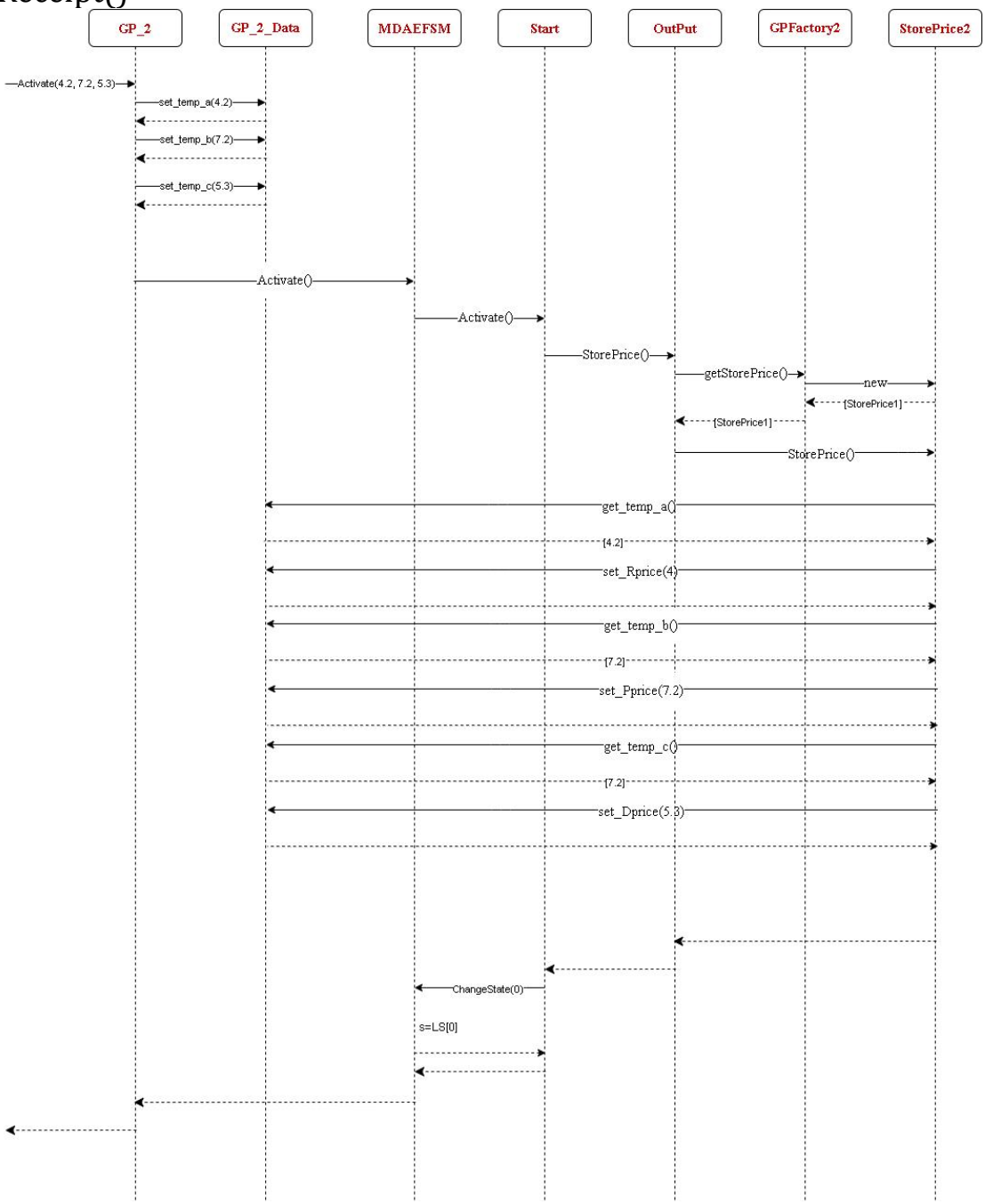

GP_1
GP_1_Data
MDAEFSM
State4
OutPut
GPFactory1
SetInitialvalues1
State3
StartPump()
get_price()
4
4>0
Continue()
Continue()
ChangeState(4)
s=LS[4]
StartPump()
StartPump()
SetInitialvalues()
getSetInitialvalues()
new
[SetInitialvalues1]
[SetInitialvalues1]
SetInitialvalues()
set_L(0)
set_total(0)
ChangeState(5)
s=LS[5]

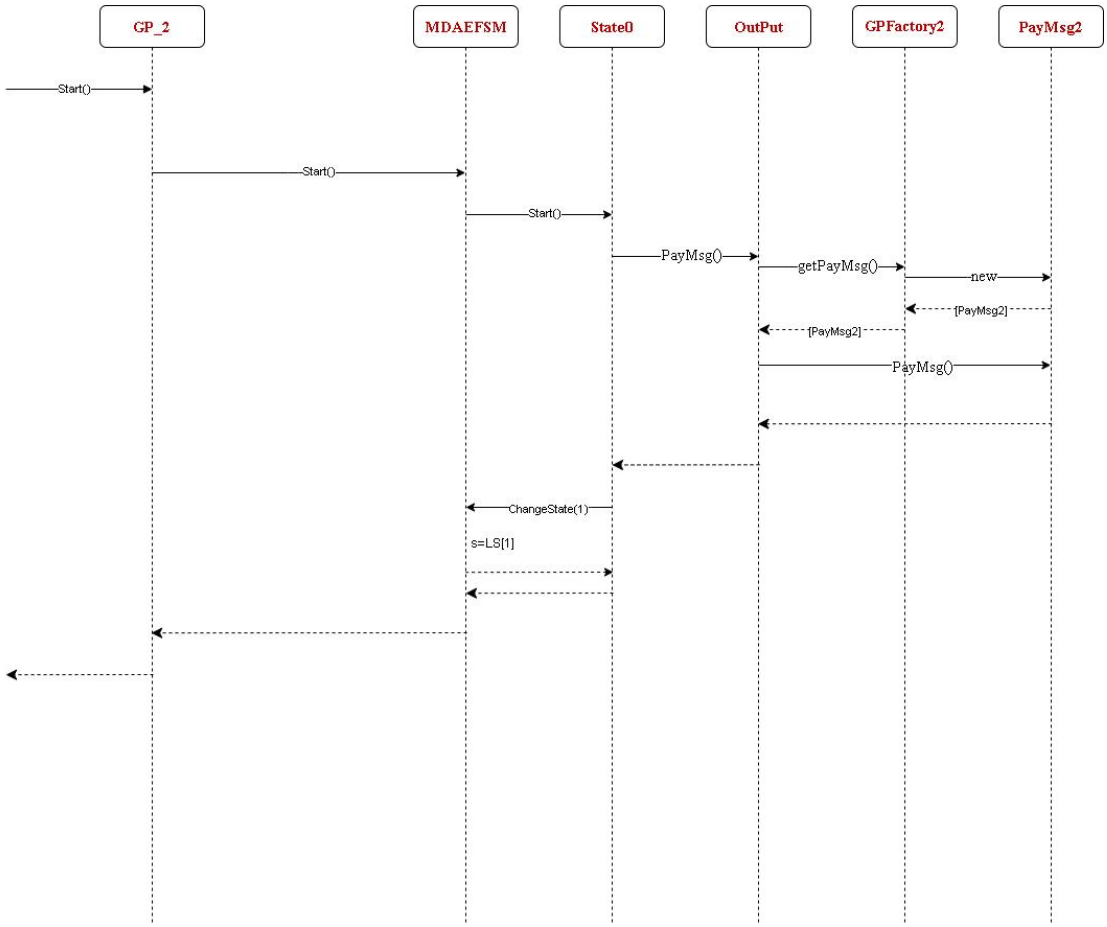

GP_1
GP_1_Data
MDAEFSM
State5
OutPut
GPFactory1
PumpGasUnit1
GasPumpedMsg1
Pump()
get_cash()
0
get_price()
4
get_L()
0
get_W()
1
Check the If condition
Pump()
Pump()
PumpGasUnit()
getPumpGasUnit()
new
[PumpGasUnit1]
[PumpGasUnit1]
PumpGasUnit()
get_L()
[0]
set_L(1)
get_price()
[4]
set_total(4)
PumpGasMsg()
getGasPumperMsg()
new
[GasPumperMsg1]
[GasPumperMsg1]
PumpGasUnit()
get_L()
[1]
get_total()
[4]

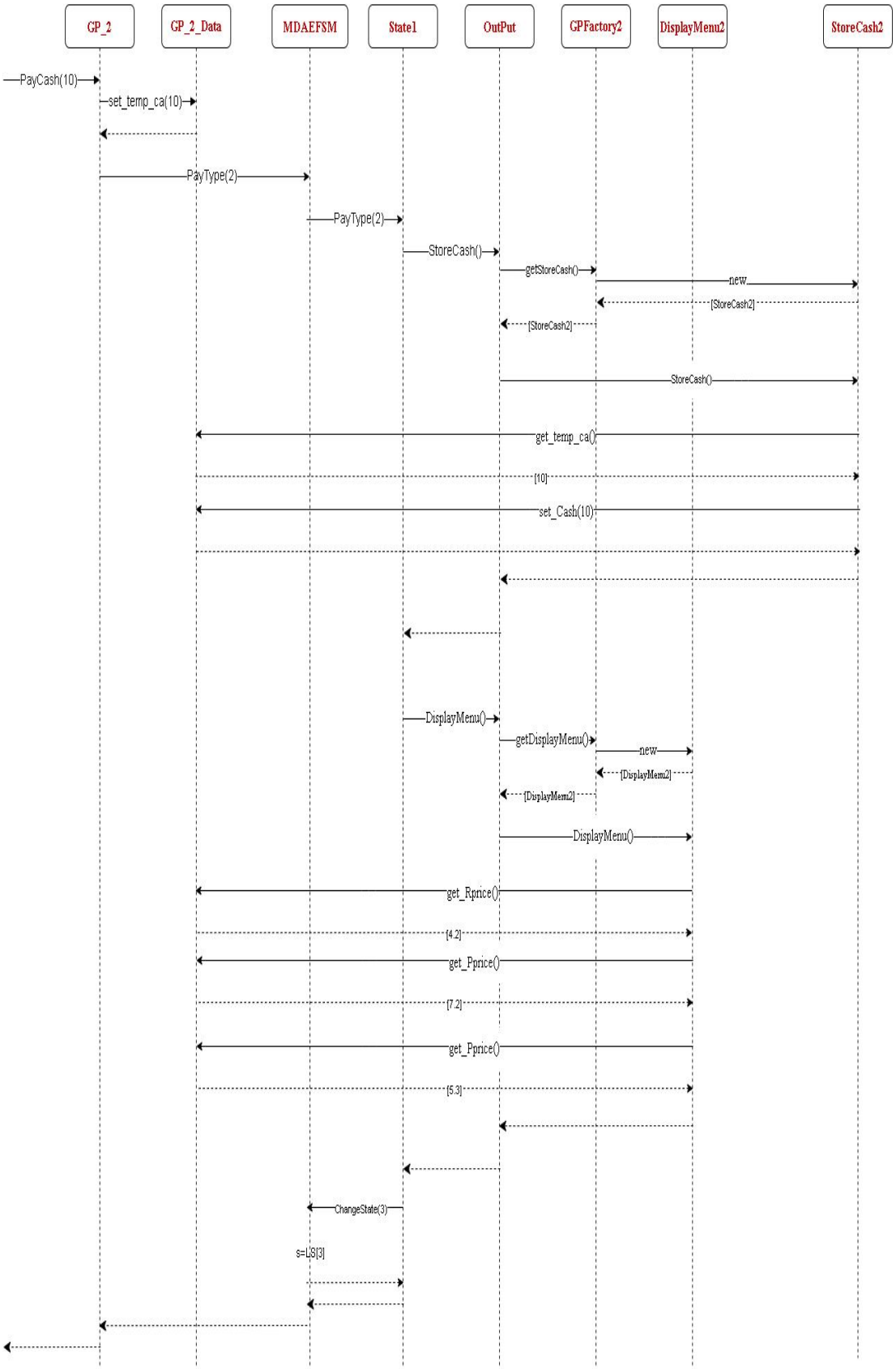

GP_1
GP_1_Data
MDAEFSM
State6
OutPut
GPFactory1
PrintReciptl
State5
StopPump()
StopPump()
StopPump()
ChangeState(6)
s=LS[6]
Recipt()
Recipt()
PrintRecipt()
getPrintRecipt()
new
[Printrecipt1]
[PrintReciptl]
PrintRecipt()
get_total()
[4]
ChangeState(0)
s=LS[0]

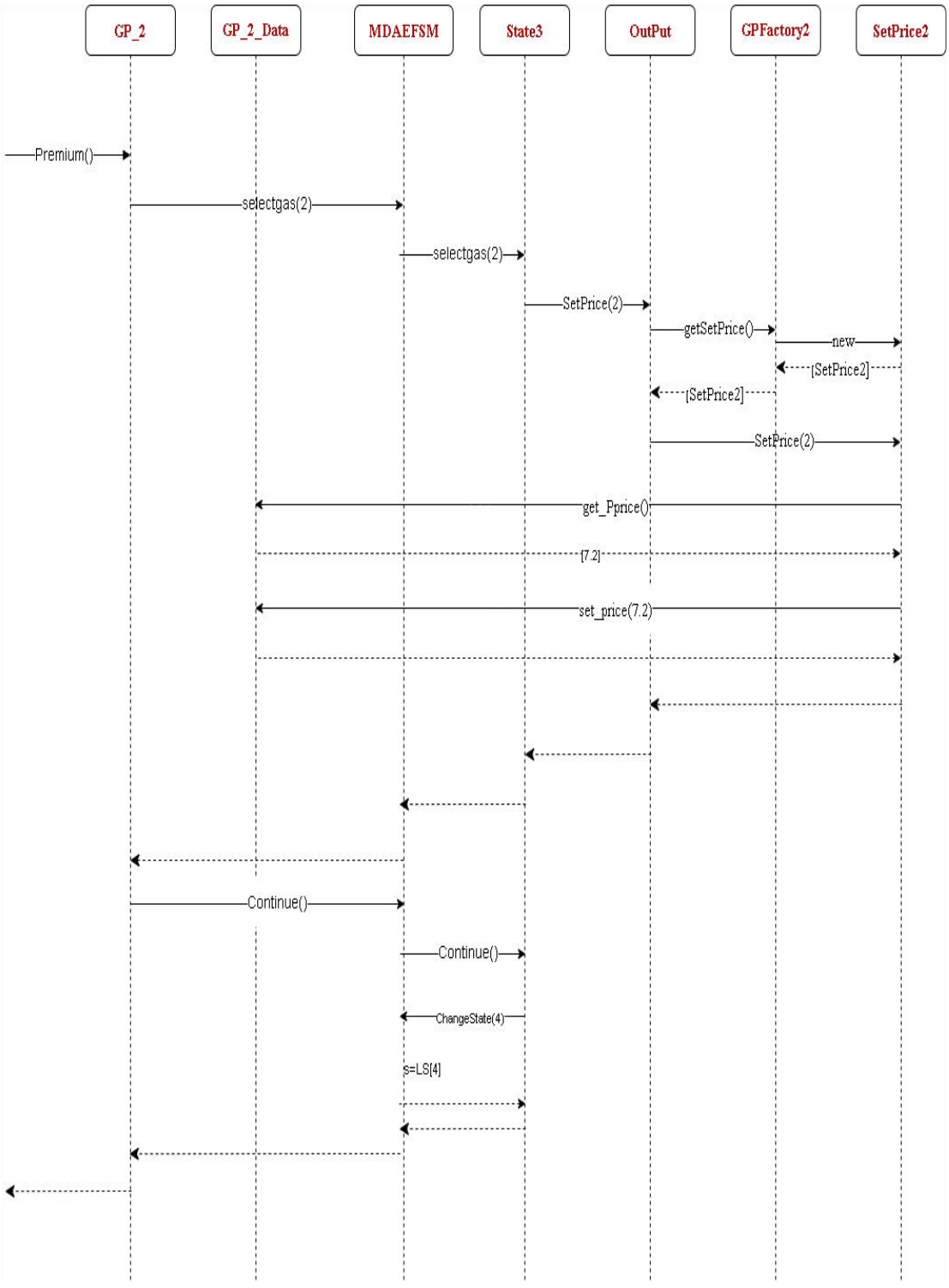
Sequence Diagram 2

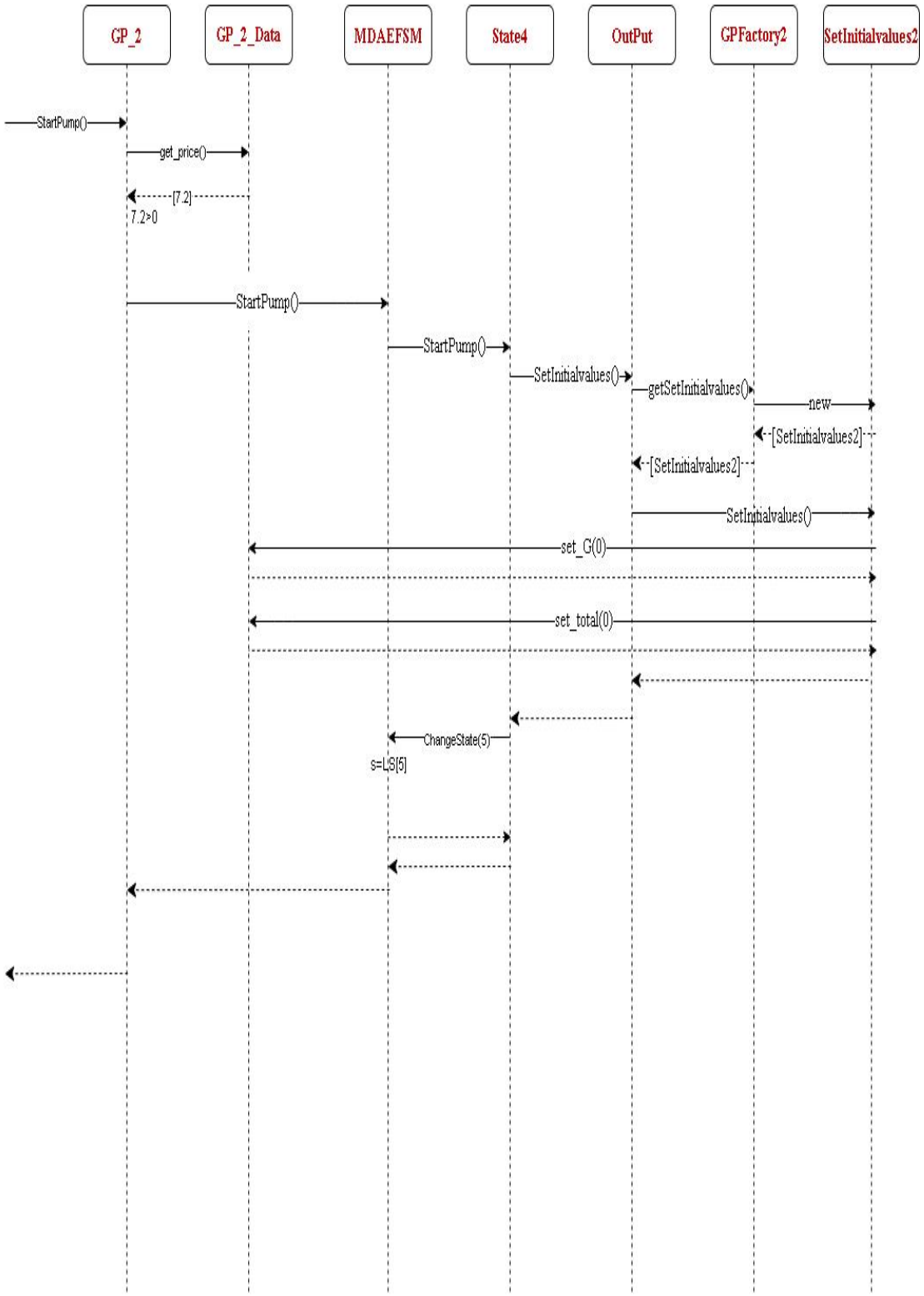
b. Scenario-II should show how one gallon of Premium gas is disposed in the Gas Pump GP-2 component, i.e., the following sequence of operations is issued: Activate(4.2, 7.2, 5.3), Start(), PayCash(10), Premium(), StartPump(), PumpGallon(), PumpGallon(), Receipt()

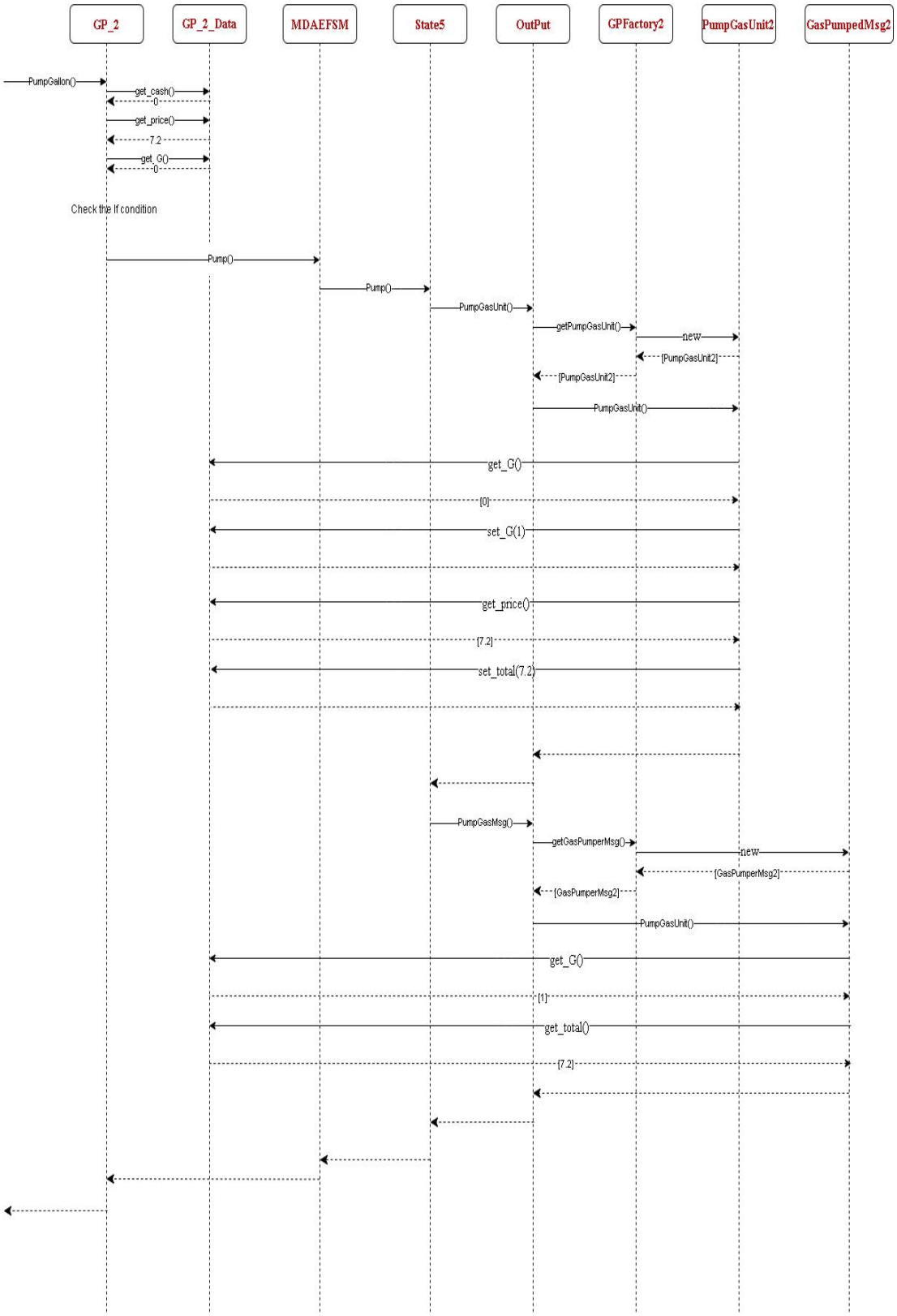


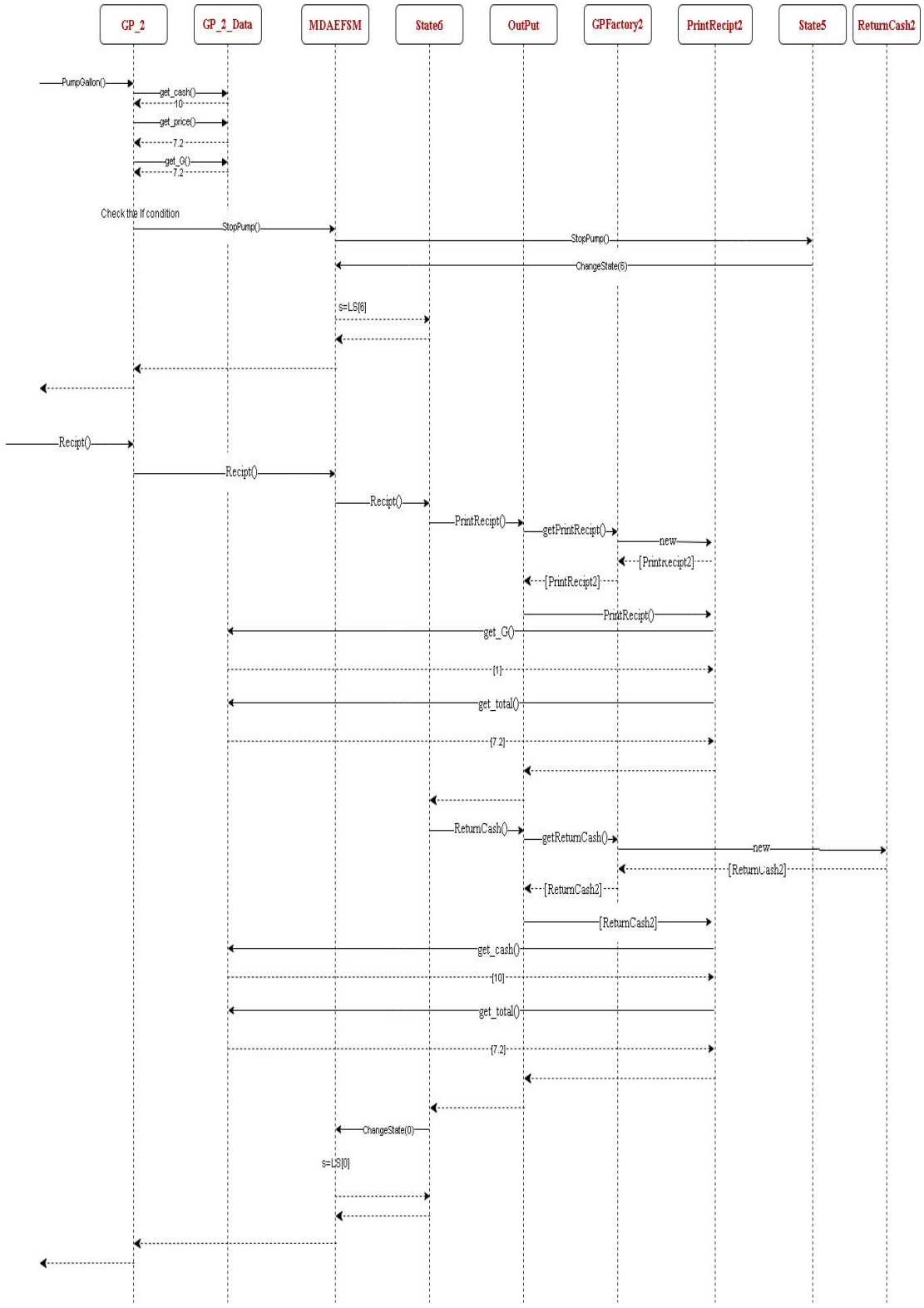

GP_2
MDAEFSM
State0
OutPut
GPFactory2
PayMsg2
Start()
Start()
Start()
PayMsg()
getPayMsg()
new
[PayMsg2]
[PayMsg2]
PayMsg()
ChangeState(1)
s=LS[1]


GP_2
GP_2_Data
MDAEFSM
State1
OutPut
GPFactory2
DisplayMenu2
StoreCash2
PayCash(10)
set_temp_ca(10)
PayType(2)
PayType(2)
StoreCash()
getStoreCash()
new
[StoreCash2]
[StoreCash2]
StoreCash()
get_temp_ca()
[10]
set_Cash(10)
DisplayMenu()
getDisplayMenu()
new
[DisplayMem2]
[DisplayMem2]
DisplayMenu()
get_Rprice()
[4.2]
get_Pprice()
[7.2]
get_Pprice()
[5.3]
ChangeState(3)
s=LS[3]


GP_2
GP_2_Data
MDAEFSM
State3
OutPut
GPFactory2
SetPrice2
Premium()
selectgas(2)
selectgas(2)
SetPrice(2)
getSetPrice()
new
[SetPrice2]
[SetPrice2]
SetPrice(2)
get_Pprice()
[7.2]
set_price(7.2)
Continue()
Continue()
ChangeState(4)
s=LS[4]


GP_2
GP_2_Data
MDAEFSM
State4
OutPut
GPFactory2
SetInitialvalues2
StartPump()
get_price()
[7.2]
7.2>0
StartPump()
StartPump()
SetInitialvalues()
getSetInitialvalues()
new
[SetInitialvalues2]
[SetInitialvalues2]
SetInitialvalues()
set_G(0)
set_total(0)
ChangeState(5)
s=LS[5]


GP_2
GP_2_Data
MDAEFSM
State5
OutPut
GPFactory2
PumpGasUnit2
GasPumpedMsg2
PumpGallon()
get_cash()
0
get_price()
7.2
get_G()
0
Check the If condition
Pump()
Pump()
PumpGasUnit()
getPumpGasUnit()
new
[PumpGasUnit2]
[PumpGasUnit2]
PumpGasUnit()
get_G()
[0]
set_G(1)
get_price()
[7.2]
set_total(7.2)
PumpGasMsg()
getGasPumperMsg()
new
[GasPumperMsg2]
[GasPumperMsg2]
PumpGasUnit()
get_G()
[1]
get_total()
[7.2]


GP_2
GP_2_Data
MDAEFSM
State6
OutPut
GPFactory2
PrintRecipt2
State5
ReturnCash2
PumpGallon()
get_cash()
10
get_price()
7.2
get_G()
7.2
Check the If condition
StopPump()
StopPump()
ChangeState(6)
s=LS[6]
Recipt()
Recipt()
Recipt()
PrintRecipt()
getPrintRecipt()
new
[PrintRecipt2]
[PrintRecipt2]
PrintRecipt()
get_G()
[1]
get_total()
[7.2]
ReturnCash()
getReturnCash()
new
[ReturnCash2]
[ReturnCash2]
[ReturnCash2]
get_cash()
[10]
get_total()
[7.2]
ChangeState(0)
s=LS[0]