CS-586

SOFTWARE SYSTEMS ARCHITECTURE

Project-Report: MDA-EFSM GAS PUMP

NAME: Vivek Sumanth Chintakula

CWID: A20457282

1. MDA-EFSM model for the Gas Pump components

A list of meta events for the MDA-EFSM

Activate()

Start()

PayCredit()

PayCash()

PayDebit()

Reject()

Cancel()

Approved()

StartPump()

Pump()

StopPump()

SelectGas(int g)

CorrectPin()

IncorrectPin(int max)

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

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

SetInitialValues // set G (or L) and total to 0;

PumpGasUnit // disposes unit of gas and counts # of units disposed

GasAckMsg // displays the amount of disposed gas

PrintReceipt // print a receipt

CancelMsg // displays a cancellation message

ReturnCash // returns the remaining cash

IncorrectPinMsg // displays incorrect pin message

StorePin // stores the pin from the temporary data store

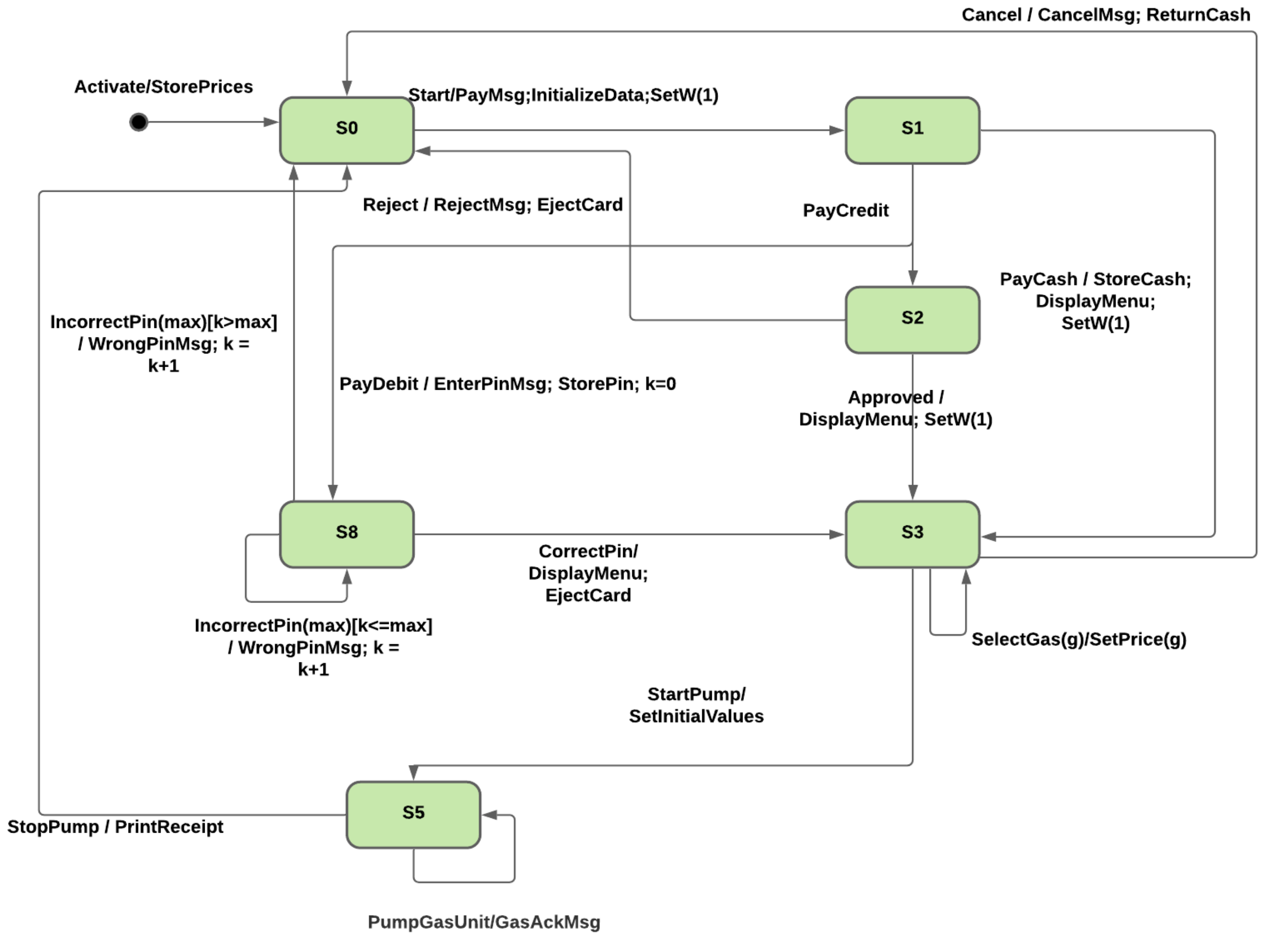
EnterPinMsg // displays a message to enter pin

InitializeData // set the value of price to 0 for GP-2; do nothing for GP-1

EjectCard() // card is ejected SetW(int w) // set value for cash flag

SetW(int W) // set value for cash flag

state diagram/model of the MDA-EFSM


Cancel / CancelMsg; ReturnCash
Activate/StorePrices
Start/PayMsg;InitializeData;SetW(1)
S0
S1
Reject / RejectMsg; EjectCard
PayCredit
S2
PayCash / StoreCash; DisplayMenu; SetW(1)
IncorrectPin(max)[k>max] / WrongPinMsg; k = k+1
PayDebit / EnterPinMsg; StorePin; k=0
Approved / DisplayMenu; SetW(1)
S8
S3
CorrectPin/ DisplayMenu; EjectCard
IncorrectPin(max)[k<=max] / WrongPinMsg; k = k+1
SelectGas(g)/SetPrice(g)
StartPump/ SetInitialValues
S5
StopPump / PrintReceipt
PumpGasUnit/GasAckMsg

Pseudo-code of all operations of Input Processors of GP-1 and GP-2

Pseudo Code of GP-1

```
Activate(int a) {

        if (a > 0) {
                d -> temp_a == a   // int a is stored in temporary variable in data store
                m -> Activate()
                }
        }

Start(){
        m -> Start()
        }

PayCredit(){
        m -> PayCredit()
        }

PayCash(float c){
        if (c > 0){
                d -> temp_c = c              // temp_c is a temporary variable in data store
                m -> PayCash()
                }
        }

Reject(){
        m -> Reject()
        }

Cancel(){
        m -> Cancel()
        }

Approved(){
        m -> Approved()
        }

StartPump(){

        m -> StartPump()
        }
```

```
PumpLiter(){

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

StopPump(){

          m -> StopPump()
          }
```

- m is pointer for MDA-EFSM
- d is pointer for Data Store
- cash contains price of selected gas
- L is number of liters pumped
- Cash flag (Cash: w = 0 or else w = 1)
- Cash, L ,Price are in Data

Pseudo Code of GP-2

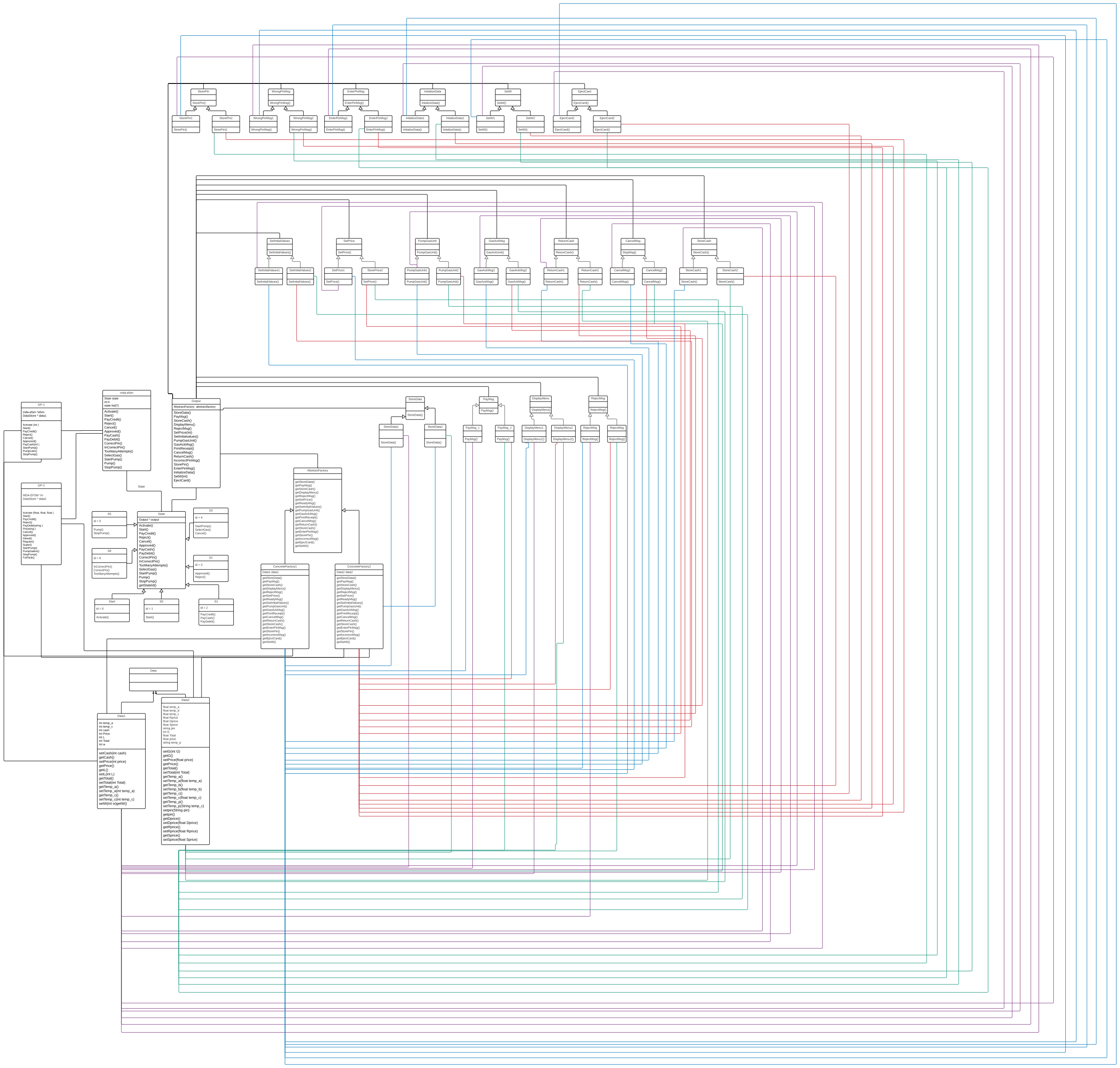
```
Activate (float a, float b,float c) {
        if ((a > 0) && (b > 0) && (c > 0)){
                d -> temp_a = a
                d -> temp_b = b
                d -> temp_c = c
                m -> Activate()
                }
        }

Start(){
        m -> start()
        }

PayCredit(){
        m -> PayCredit()
        }

Reject(){
        m -> Reject()
        }

PayDebit(string p){
        d -> temp_pin = p;
        m -> PayDebit()
        }

Pin(string x){
        if ( d -> pin == x){
                m -> CorrectPin()
        }
        else {
                m -> InCorrectPin(1)
        }

Cancel(){
        m -> cancel()
        }

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

```
Diesel(){
        m -> SelectGas(3)
        }

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

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

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

PumpGallon(){
        m -> Pump()
        }

StopPump(){
        m -> StopPump()
        }

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

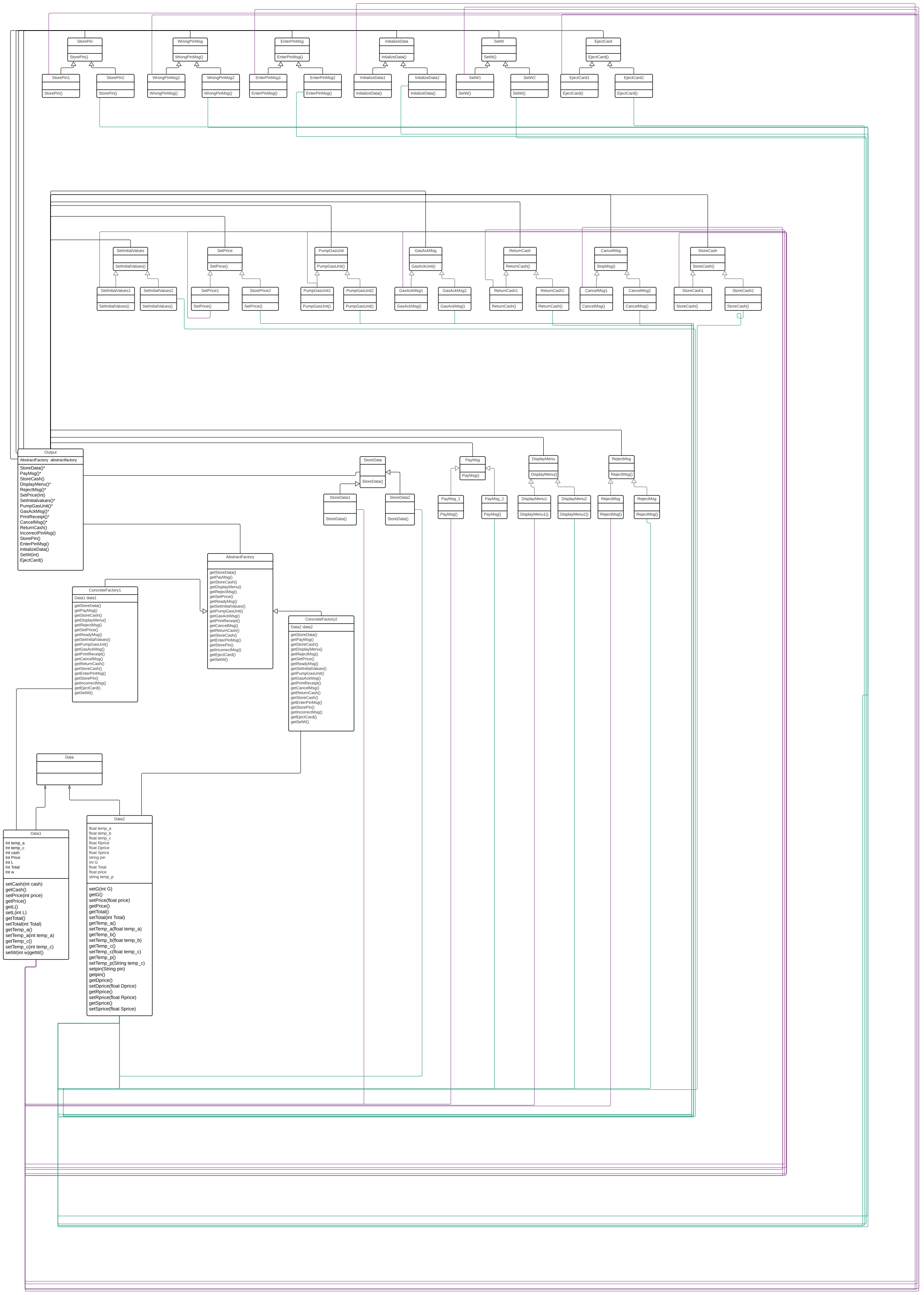
- d is pointer for Data Store
- m is pointer for MDA-EFSM
- pin contains pin in Data Store Object
- SelectGas(g) : Regular(1), Super(2), Diesel(3)

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

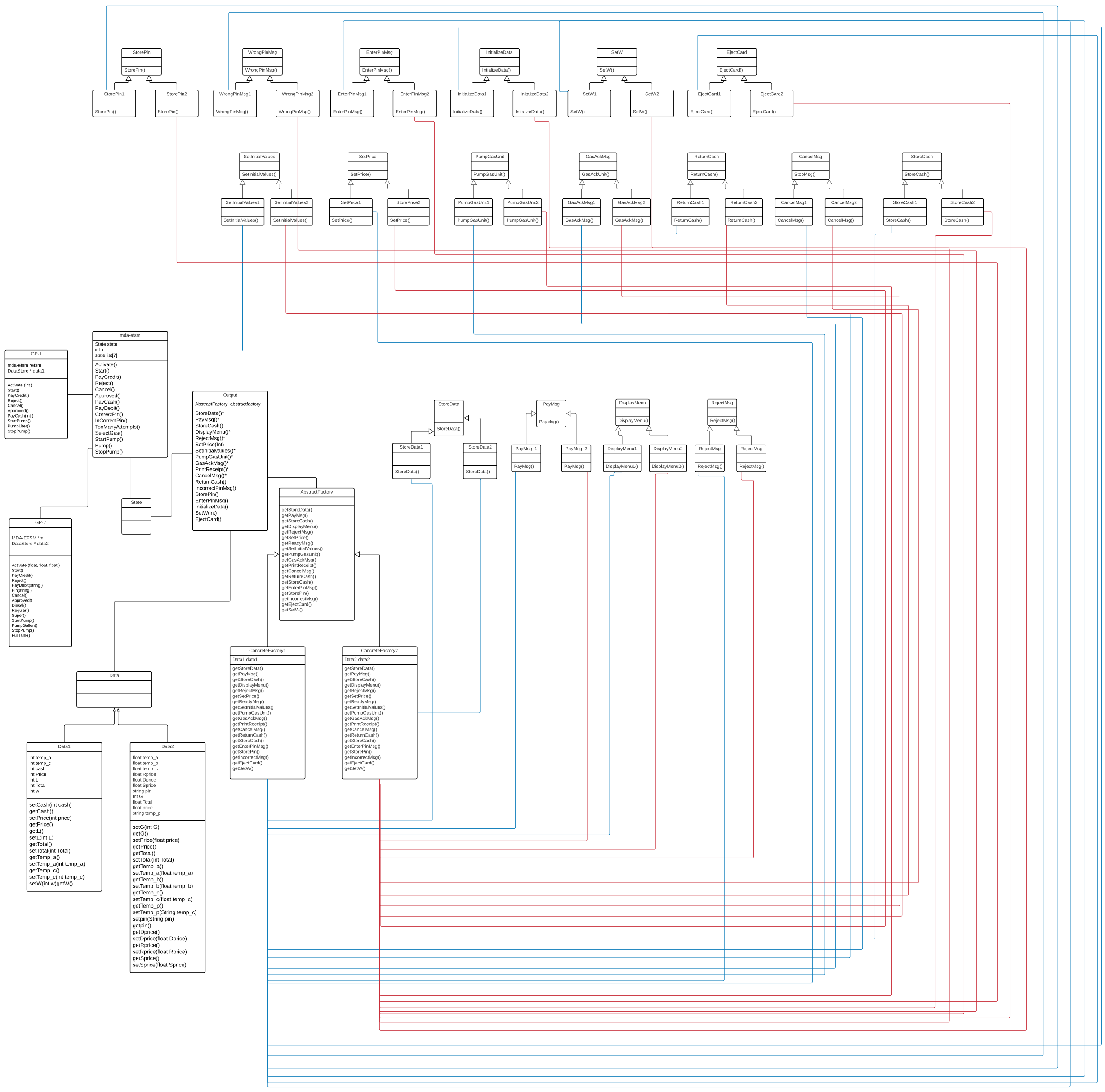
Class Diagram



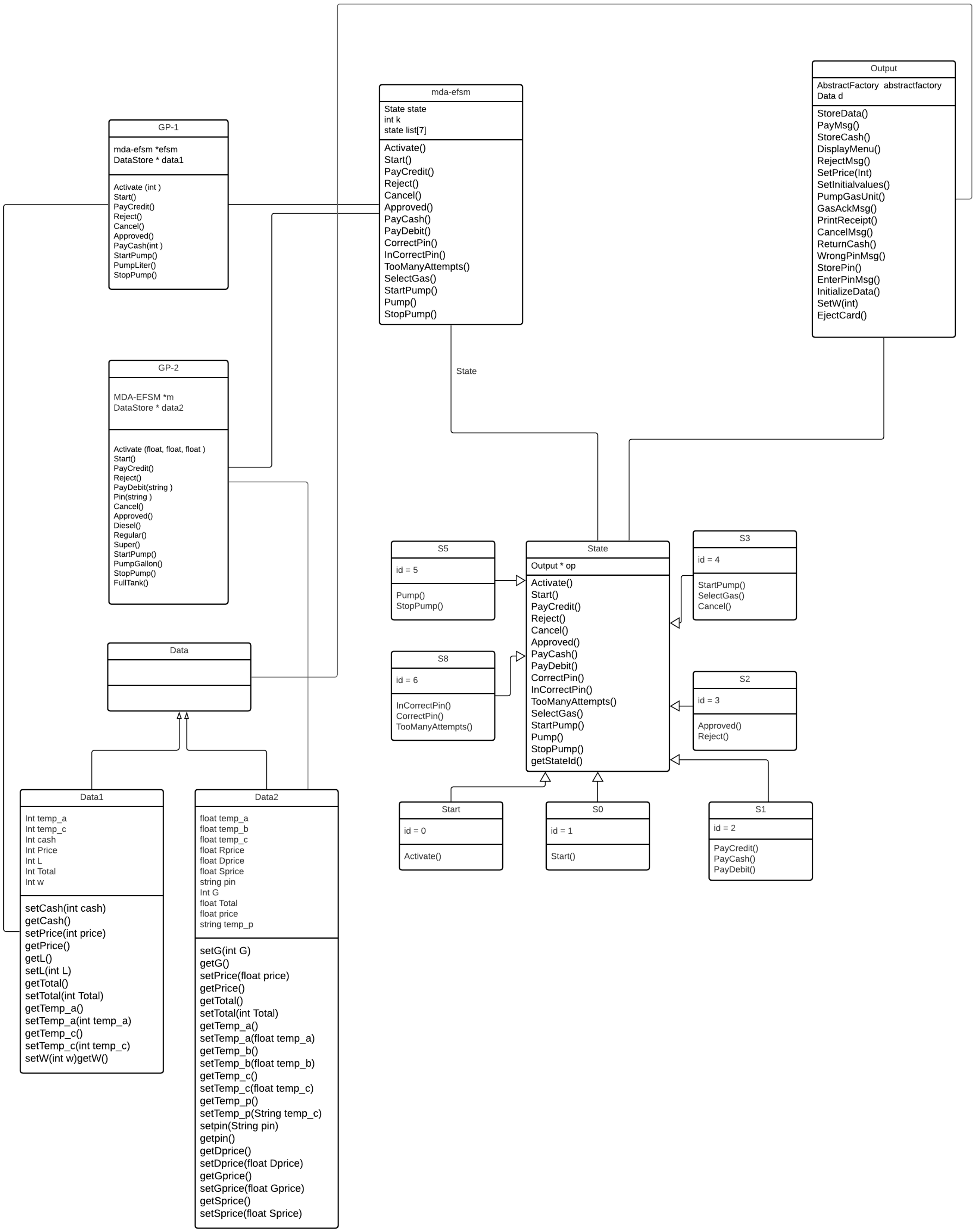
Stratergy Pattern



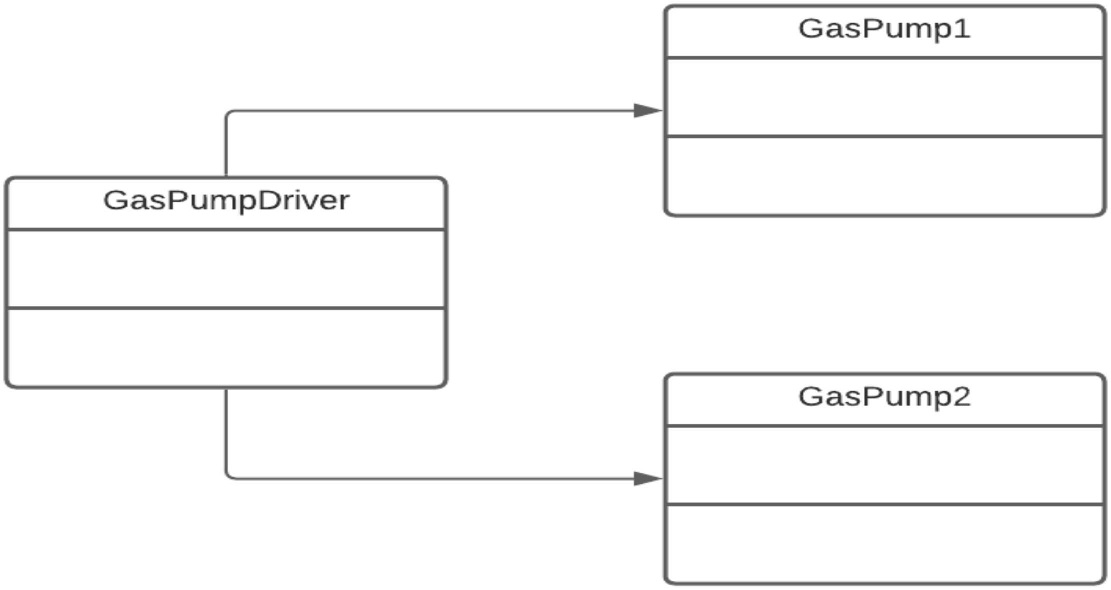
Abstract Factory Design Pattern



State Design Pattern



3. Purpose of the class and responsibility of the each operation supported by each class



GasPumpDriver:

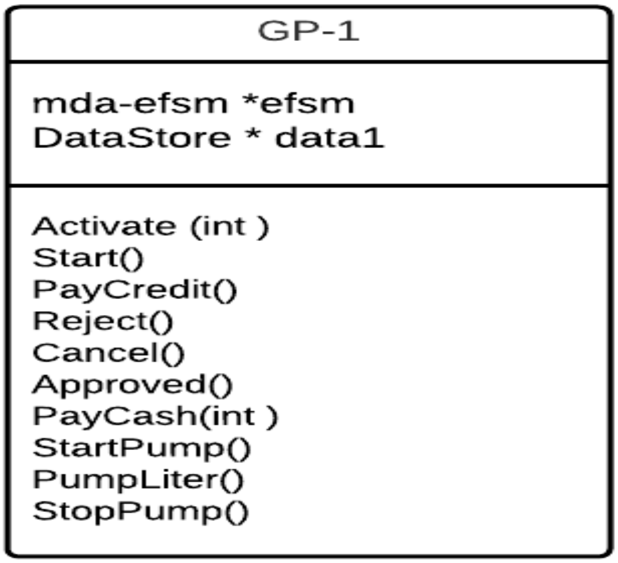
Through the GasPumpDriver Class User can select either of GasPump1 or GasPump2 where GasPump1 has methods related to GasPump1, Similarly GasPump2 has methods related to GasPump2.

When a GasPump is selected then Concrete Factory Class is created along with the objects.

User can pass input data specific to that GasPump which stores in Data Class

Here are two Gas Pumps, user can select one of the gas pump

GasPump1



Gas Pump 1 class has methods to specific to that gas pump.

Pointers and Variables:

mda-efsm * efsm
DataStore * data1

Methods:

Activate(int): Activate method takes input of type Integer and sets the price of the gas by invoking Activate() in mda-efsm.

Start(): Start method should be invoked to do operations on the pump, This method invokes Start() in mda-efsm.

PayCredit(): PayCredit is a payment method which should be selected, and this method invokes PayCredit() in mda-efsm.

Reject(): Reject method invokes Reject() in mda-efsm.

Cancel(): Cancel method invokes Cancel() in mda-efsm.

Approved(): Approved method should be invoked before the Startpump() method, this invokes Approced() in mda-efsm

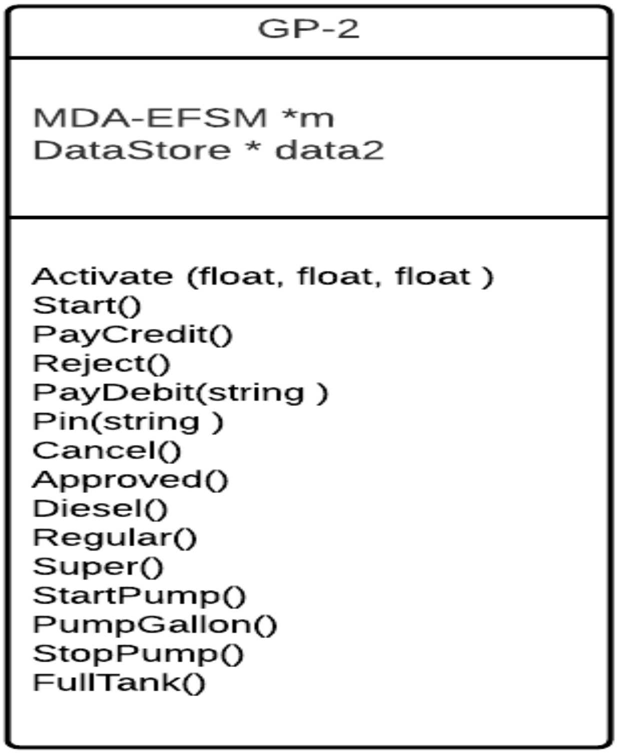
PayCash(int): PayCash is another mode of payment available to GasPump1, which takes Integer as input. This method invokes PayCash() in mda-efsm.

StartPump(): StartPump method is invoked to start pump and this method invokes StartPump in mda-efsm.

PumpLiter(): PumpLiter method invokes pumpliter() in mda-efsm.

StopPump(): StopPump() method is invoked to stop pump, this method invokes StopPump() in mda-efsm.

Gas Pump 2



GP-2

MDA-EFSM *m
DataStore * data2

Activate (float, float, float)
Start()
PayCredit()
Reject()
PayDebit(string)
Pin(string)
Cancel()
Approved()
Diesel()
Regular()
Super()
StartPump()
PumpGallon()
StopPump()
FullTank()

Gas Pump 2 class has methods to specific to that gas pump.

Pointers and Variables:

mda-efsm * efsm
DataStore * data2

Methods:

Activate(): Activate method takes input of type Integer and sets the price of the gas by invoking Activate() in mda-efsm.

Start(): Start method should be invoked to do operations on the pump, This method invokes Start() in mda-efsm.

PayCredit(): this is a payment method this method invokes PayCredit() in mda-efsm.

Reject(): Reject method invokes Reject() in mda-efsm.

PayDebit(String): this a Payment method, this method takes input as String and this method invokes PayCredit() in mda-efsm.

Pin(String): this method validates Pin, takes input as String and this method invokes PayCredit() in mda-efsm.

Cancel(): Cancel method invokes Cancel() in mda-efsm.

Approved(): Approved method should be invoked before the Startpump() method, this invokes Approced() in mda-efsm

Diesel(): This Method invokes SelectGas method passing 1 as parameter

Regular(): This Method invokes SelectGas method passing 2 as parameter

Super(): This Method invokes SelectGas method passing 3 as parameter

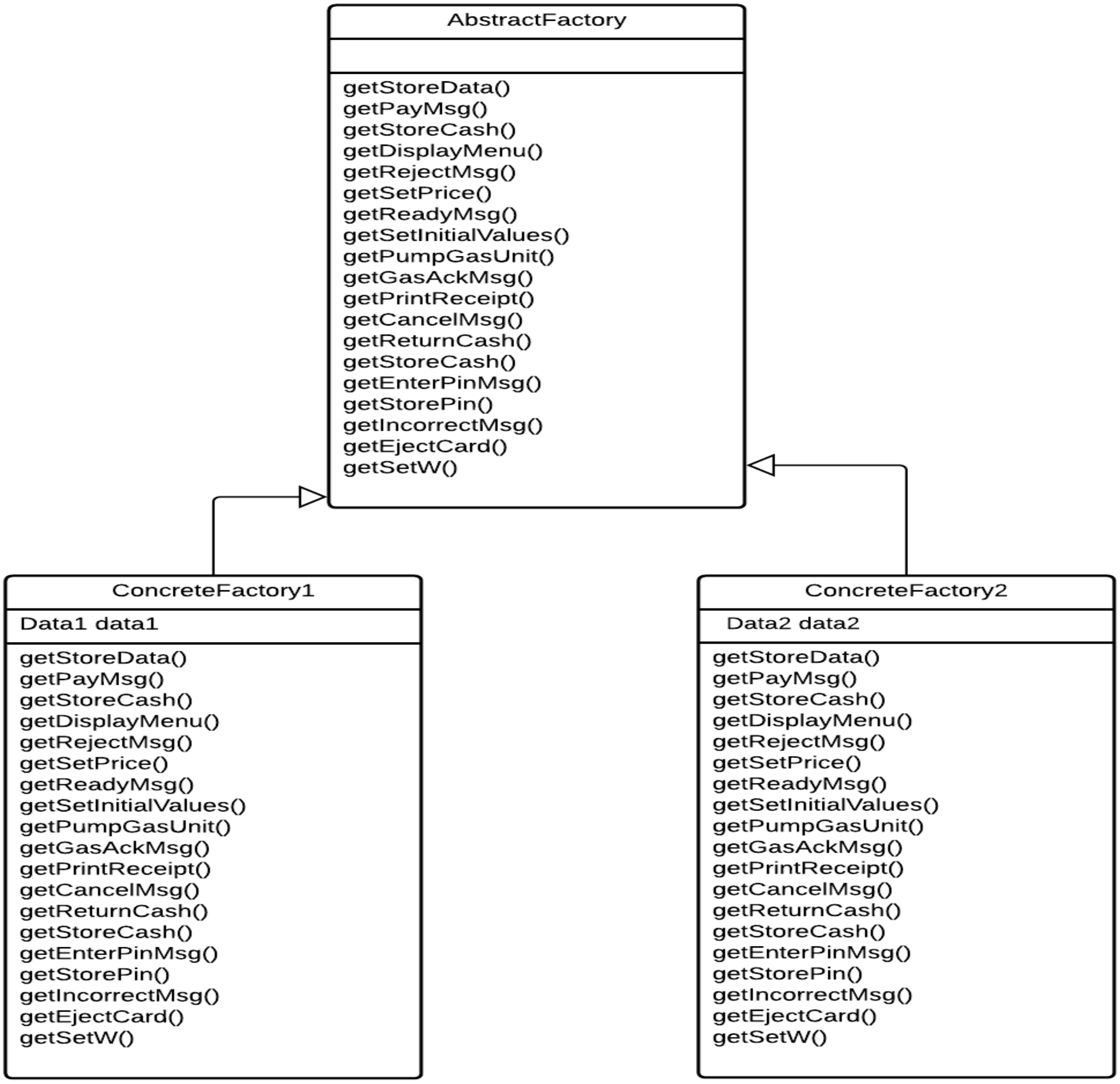
StartPump(): StartPump method is invoked to start pump and this method invokes StartPump in mda-efsm.

PumpGallon(): PumpLiter method invokes pumpliter() in mda-efsm.

StopPump(): StopPump() method is invoked to stop pump, this method invokes StopPump() in mda-efsm.

FullTank(): This method invokes FullTank() in mda-efsm.

Abstract Factory Pattern



Class Abstract Factory:

This is AbstractFactory class is abstract class of Abstract Factory Design Pattern.

Methods:

getStoreData()
getPayMsg()
getStoreCash()

getDisplayMenu()
getRejectMsg()
getSetPrice()
getReadyMsg()
getSetInitialValues()
getPumpGasUnit()
getGasAckMsg()
getPrintReceipt()
getCancelMsg()
getReturnCash()
getStoreCash()
getEnterPinMsg()
getStorePin()
getIncorrectMsg()
getEjectCard()
getSetW()

All these methods are abstract methods

Class Concrete Factory1

This creates the objects of Stratergy classes DataStore of GasPump1,This is a child class of abstract factory design pattern.

Pointers and variables

Data1 data1 **-** Pointer to Data1 class

Methods

getStoreData()	: Returns instance of StoreData1
getPayMsg()	: Returns instance of PayMsg1
getDisplayMenu()	: Returns instance of DisplayMenu1
getRejectMsg()	: Returns instance of RejectMsg1
getSetPrice()	: Returns instance of SetPrice1
getReadyMsg()	: Returns instance of ReadyMsg1
getSetInitialValues()	: Returns instance of SetInitialValues1
getPumpGasUnit()	: Returns instance of PumpGasUnit1
getGasAckMsg()	: Returns instance of GasAckMsg1
getPrintReceipt()	: Returns instance of PrintReceipt1
getCancelMsg()	: Returns instance of CancelMsg1
getReturnCash()	: Returns instance of ReturnCash1

getStoreCash()	: Returns instance of StoreCash1
getEnterPinMsg()	: Returns instance of EnterPinMsg1
getStorePin()	: Returns instance of StorePin1
getIncorrectMsg()	: Returns instance of InCorrectMsg1
getEjectCard()	: Returns instance of EjectCard1
getSetW()	: Returns instance of SetW1

Class Concrete Factory2

This creates the objects of Stratergy classes DataStore of GasPump2,This is a child class of abstract factory design pattern.

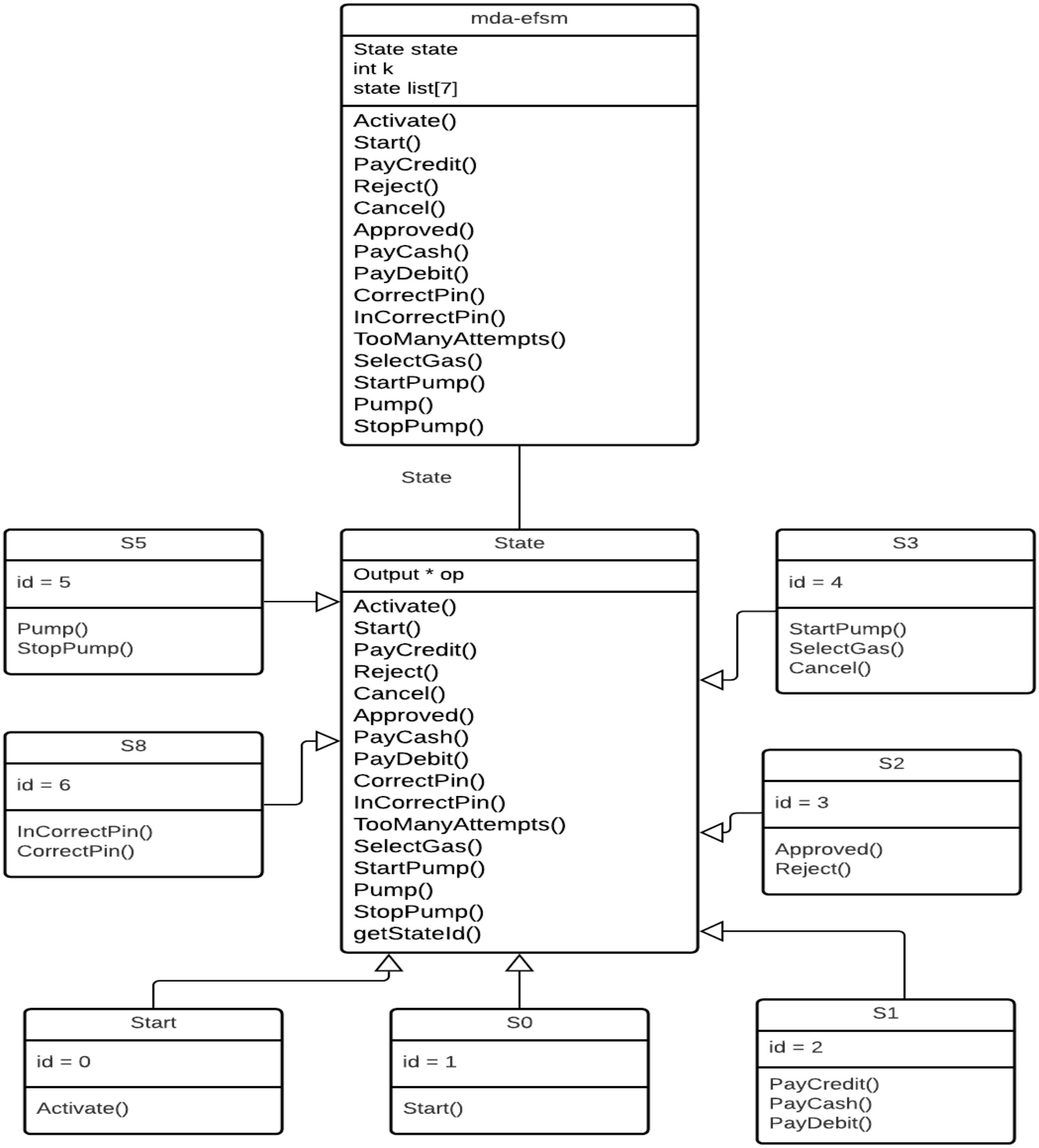
Pointers and variables

Data2 data2 - Pointer to Data2 class

Methods

getStoreData()	: Returns instance of StoreData2
getPayMsg()	: Returns instance of PayMsg2
getDisplayMenu()	: Returns instance of DisplayMenu2
getRejectMsg()	: Returns instance of RejectMsg2
getSetPrice()	: Returns instance of SetPrice2
getReadyMsg()	: Returns instance of ReadyMsg2
getSetInitialValues()	: Returns instance of SetInitialValues2
getPumpGasUnit()	: Returns instance of PumpGasUnit2
getGasAckMsg()	: Returns instance of GasAckMsg2
getPrintReceipt()	: Returns instance of PrintReceipt2
getCancelMsg()	: Returns instance of CancelMsg2
getReturnCash()	: Returns instance of ReturnCash2
getStoreCash()	: Returns instance of StoreCash2
getEnterPinMsg()	: Returns instance of EnterPinMsg2
getStorePin()	: Returns instance of StorePin2
getIncorrectMsg()	: Returns instance of InCorrectMsg2
getEjectCard()	: Returns instance of EjectCard2
getSetW()	: Returns instance of SetW2

mda-efsm Class



GasPump 1 and GasPump 2 methods invokes the methods in class mda-efsm.All the state changes happens here, Intial state is set to start which is S0.

Pointers and Variables:

State state
int k
state list[7]

list[0] = Start
list[1] = S0
list[2] = S1
list[3] = S2
list[4] = S3
list[5] = S5
list[6] = S8

Methods:

Activate() **:** Activate method invokes state class changes state from S0 -> S1

Start() **:** Start method invokes state class changes state from S1 -> S2

PayCredit() **:** PayCredit method invokes state class changes state from S2 -> S3

Reject() **:** Reject method invokes state class changes state from S3 -> S1

Cancel() **:** Cancel method invokes state class changes state from S4 -> S1

Approved() **:** Approved method invokes state class changes state from S3 -> S4

PayCash() **:** PayCash method invokes state class changes state from S2 -> S4

PayDebit() **:** PayDebit method invokes state class changes state from S2 -> S6

CorrectPin() : CorrectPin method invokes state class
changes state from S6 -> S4

InCorrectPin() : InCorrectPin method invokes state class
if maximum attempts are reached TooManyAttempts() is
Called

TooManyAttempts(): TooManyAttempts method invokes state
class changes state from S0 -> S1

SelectGas() : Activate method invokes state

StartPump() : Activate method invokes state
class changes state from S4 -> S5

Pump() : Activate method invokes state class

StopPump() : Activate method invokes state class
changes state from S5 -> S1

State Class

State Design Pattern is Inplemented here,This is a Abstract Class.

Pointers and Variables

Output output

Methods:

Activate()
Start()
PayCredit()
Reject()
Cancel()
Approved()
PayCash()
PayDebit()
CorrectPin()
InCorrectPin()
TooManyAttempts()
SelectGas()

StartPump()
Pump()
StopPump()
getStateId()

All these methods are Abstract Methods

Start Class

Pointers and Variables

id = 0

Methods:

Activate(): Invokes StoreData() in Output Class

getStateId(): Returns state id of class

S0 Class

Pointers and Variables

id = 1

Methods:

Start(): Invokes PayMsg() in Output Class

getStateId(): Returns state id of class

S1 Class

Pointers and Variables

id = 2

Methods:

PayCredit(): has No Action

PayDebit(): Invokes StorePin(), EnterPinMsg(), SetW(0) in Output Class

PayCash(): Invokes StoreCash() and DisplayMenu(),SetW(1) in Output Class

getStateId(): Returns state id of class

S2 Class

Pointers and Variables

id = 3

Methods:

Approved(): Invokes DisplayMenu() and EjectCard() in Output Class

Reject(): Invokes RejectMsg() in Output Class

getStateId(): Returns state id of class

S3 Class

Pointers and Variables

id = 4

Methods:

StartPump(): Invokes SetInitialValues() and ReadyMsg() in Output Class

SelectGas(): Invokes SetPrice() in Output Class

Cancel(): Invokes CancelMsg() and ReturnCash() in Output Class

getStateId(): Returns state id of class

S5 Class

Pointers and Variables

id = 5

Methods:

Pump(): Invokes PumpGasUnit() and GasAckMsg() in Output Class

StopPump(): Invokes StopMsg() and PrintRecipt() in Output Class

getStateId(): Returns state id of class

S8 Class

Pointers and Variables

id = 6

Methods:

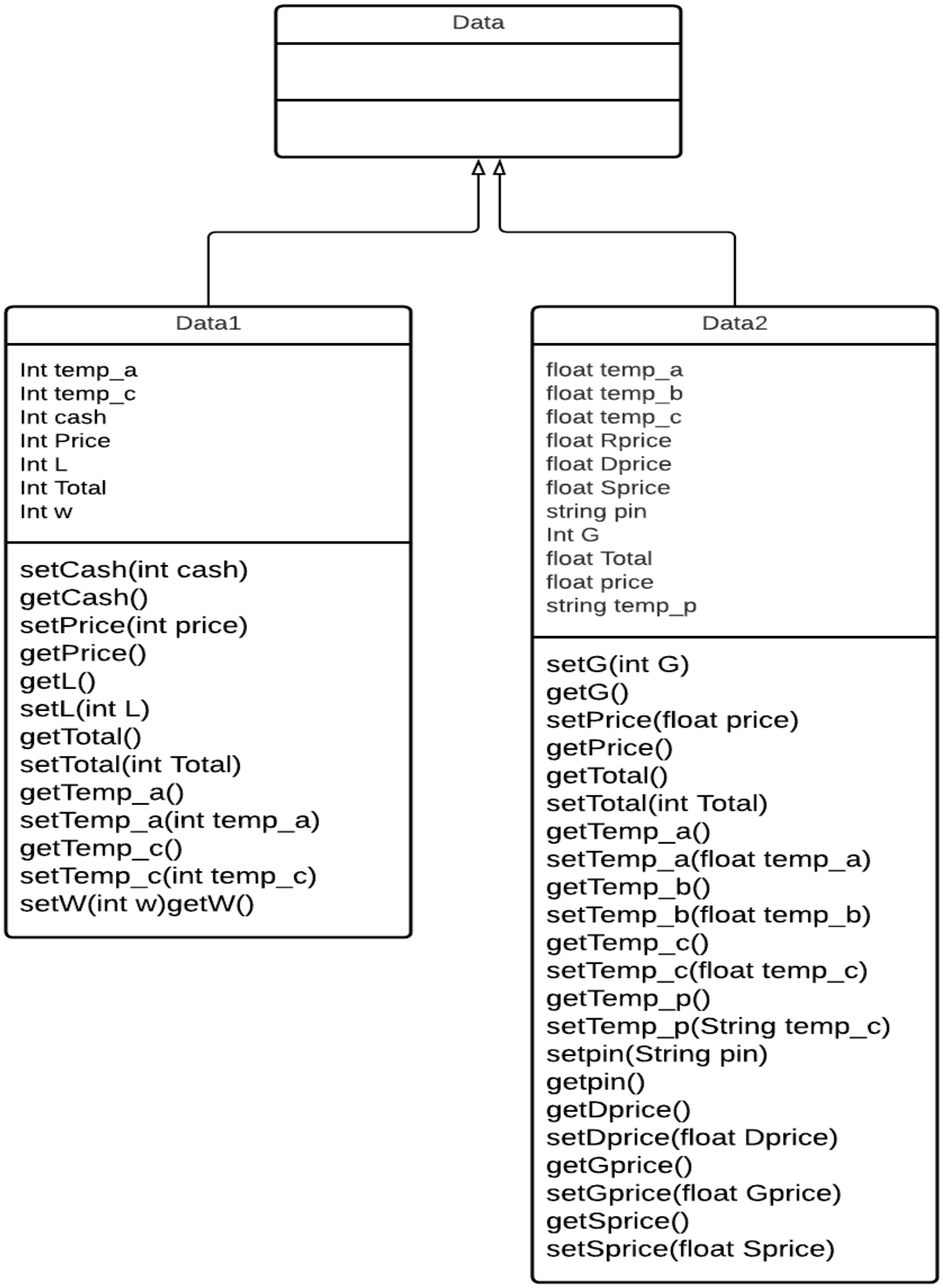
InCorrectPin(): Invokes InCorrectPinMsg() in Output Class

CorrectPin(): Invokes EjectCard() and DisplayMenu() in Output Class

TooManyAttempts(): Invokes InCorrectPinMsg(), EjectCard() in Output Class

getStateId(): Returns state id of class

Data Class


Data
Data1
Int temp_a
Int temp_c
Int cash
Int Price
Int L
Int Total
Int w
setCash(int cash)
getCash()
setPrice(int price)
getPrice()
getL()
setL(int L)
getTotal()
setTotal(int Total)
getTemp_a()
setTemp_a(int temp_a)
getTemp_c()
setTemp_c(int temp_c)
setW(int w)getW()
Data2
float temp_a
float temp_b
float temp_c
float Rprice
float Dprice
float Sprice
string pin
Int G
float Total
float price
string temp_p
setG(int G)
getG()
setPrice(float price)
getPrice()
getTotal()
setTotal(int Total)
getTemp_a()
setTemp_a(float temp_a)
getTemp_b()
setTemp_b(float temp_b)
getTemp_c()
setTemp_c(float temp_c)
getTemp_p()
setTemp_p(String temp_c)
setpin(String pin)
getpin()
getDprice()
setDprice(float Dprice)
getGprice()
setGprice(float Gprice)
getSprice()
setSprice(float Sprice)

Data Class

This is a Abstract Class

Data1 Class

This Class is used to Store Data for GasPump1, Here we have getters and setter methods where we store into a temporary variable in GasPump's and set into a permanent variable though Output Class

Variables and Pointers

Integer temp_a
Integer temp_c
Integer Cash
Integer Price
Integer Total
Integer L
Integer W

Methods

setCash(int cash)	set cash value to variable Cash
getCash()	returns cash
setPrice(int price)	set price value to variable Price
getPrice()	returns Price
getL()	returns L
setL(int L)	set L value to variable L
getTotal()	returns Total
setTotal(int Total)	set Total value to variable Total
getTemp_a()	returns temp_a
setTemp_a(int temp_a)	set temp_a value to variable temp_a
getTemp_c()	returns temp_c
setTemp_c(int temp_c)	set temp_c value to variable temp_c
setW(int w)	set w value to variable temp_c
getW()	returns w

Data2 Class

This Class is used to Store Data for GasPump2

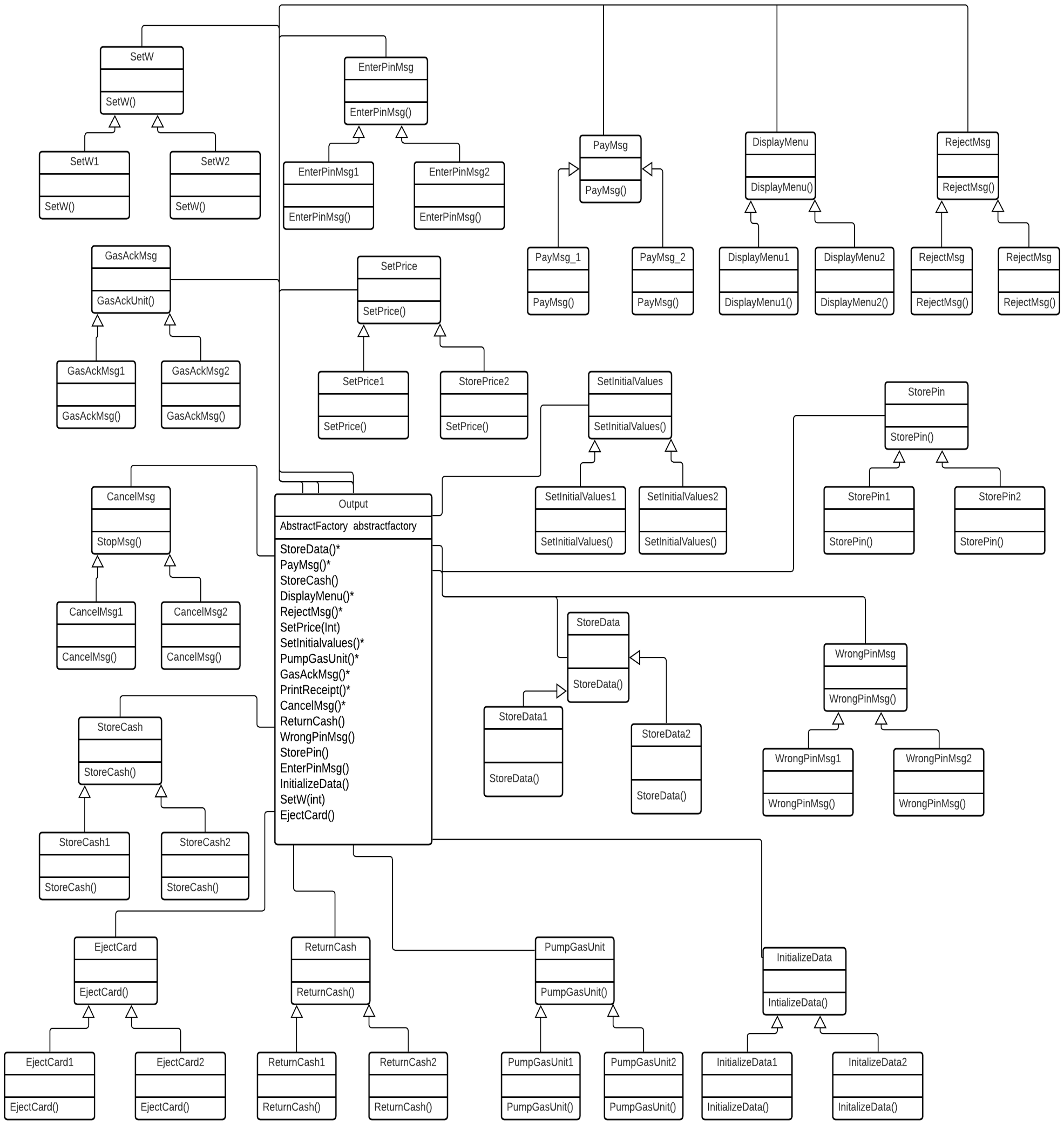
Variables and Pointers

Float temp_a
Float temp_b
Float temp_c
String temp_p
String pin
Float Price
Integer G
Float Total
Float RPrice
Float DPrice
Float SPrice

Methods

setG(int G)	:	set G to variable G
getG()	:	returns G
setPrice(float price)	:	set price to variable price
getPrice()	:	returns price
getTotal()	:	returns Total
setTotal(int Total)	:	set Total to variable Total
getTemp_a()	:	returns temp_a
setTemp_a(float temp_a)	:	set temp_a to variable temp_a
getTemp_b()	:	returns temp_b
setTemp_b(float temp_b)	:	set temp_b to variable temp_b
getTemp_c()	:	returns temp_c
setTemp_c(float temp_c)	:	set temp_c to variable temp_c
getTemp_p()	:	returns temp_p
setTemp_p(String temp_p)	:	set temp_p to variable temp_p
setpin(String pin)	:	set pin to variable Pin
getpin()	:	returns pin
getDprice()	:	returns Dprice
setDprice(float Dprice)	:	set Dprice to variable Dprice (Diesel)
getRprice()	:	returns RPrice()
setRprice(float Gprice)	:	set Rprice to variable RPrice (Regular)
getSprice()	:	returns Sprice
setSprice(float Sprice)	:	set Sprice to variable SPrice (Super)

Output Class



Output Class

Output Class Implements all the methods, Output Class gets objects from Concrete Factory 1 and Concrete Factory 2 and Performs the operations in Statergy Pattern, these methods are called by stratergy class objects according to GasPump1 and GasPump2.

Variables and Pointers

AbstractFactory abstractfactory
StoreData storeData
PayMsg payMsg
StoreCash storeCash
DisplayMenu displayMenu;
RejectMsg rejectMsg;
SetPrice setPrice;
SetInitialValues setInitialValues;
PumpGasUnit pumpGasUnit;
GasAckMsg gasAckMsg; // gas pumped message
PrintReceipt printReceipt;
CancelMsg cancelMsg;
ReturnCash returnCash;
IncorrectPinMsg incorrectPinMsg; // wrong pin message
StorePin storePin;
EnterPinMsg enterPinMsg;
EjectCard ejectcard;
SetW setw;

Methods

StoreData() : Calls StoreData() of StoreData Class
PayMsg() : Calls PayMsg() of StoreData Class
StoreCash() : Calls StoreCash() of StoreCash Class
DisplayMenu() : Calls DisplayMenu() of StoreCash Class
RejectMsg() : Calls RejectMsg() of RejectMsg Class
SetPrice(Int) : Calls SetPrice() of SetPrice Class
SetInitialvalues() : Calls SetInitialValues() of SetIntialValues Class
PumpGasUnit() : Calls PumpGasUnit() of PumpGasUnit Class
GasAckMsg() : Calls GasAckMsg() of GasAckMsg Class
PrintReceipt() : Calls PrintReceipt() of PrintReceipt Class
CancelMsg() : Calls CancelMsg() of CancelMsg Class
ReturnCash() : Calls ReturnCash() of ReturnCash Class
IncorrectPinMsg() : Calls InCorrectPinMsg() of InCorrectPinMsg Class
StorePin() : Calls StorePin() of StorePin Class

EnterPinMsg() : Calls EnterPinMsg() of EnterPinMsg Class
SetW(int) : Calls SetW() of SetW Class
EjectCard() : Calls EjectCard() of EjectCard Class

Store Data Class

Methods:

StoreData(): This is a abstract method

StoreData1 Class

This Class extends StoreData() abstract class

Methods:

StoreData(): Sets Price by getting temp_a

StoreData2 Class

This Class extends StoreData() abstract class

Methods:

StoreData(): Sets Rprice, Dprice, Sprice by getting temp_a, temp_b,temp_c

PayMsg

Methods:

PayMsg(): This is a abstract method

PayMsg1

This Class extends PayMsg() abstract class

Methods:

PayMsg1(): Displays PayMsg1

PayMsg2

This Class extends PayMsg() abstract class

Methods:

PayMsg2(): Displays PayMsg2

StoreCash

Methods:

StoreCash(): This is a abstract method

storecash1

This Class extends storecash() abstract class

Methods:

StoreCash1(): get temp_c and set cash

storeCash2

This Class extends storecash() abstract class

Methods:

StoreCash2(): // No action

DisplayMenu

Methods:

DisplayMenu(): This is a abstract method

DisplayMenu1

This Class extends DisplayMenu() abstract class

Methods:

DisplayMenu1(): Display Available Options

DisplayMenu2

This Class extends DisplayMenu() abstract class

Methods:

DisplayMenu2(): Displays Available Options and options of gas

RejectMsg

Methods:

RejectMsg(): This is a abstract method

RejectMsg1

This Class extends RejectMsg() abstract class

Methods:

RejectMsg1(): Displays RejectMsg 1

RejectMsg2

This Class extends RejectMsg() abstract class

Methods:

RejectMsg2(): Displays RejectMsg 2

SetPrice(Int)

Methods:

SetPrice(): This is a abstract method

SetPrice1(Int)

This Class extends SetPrice() abstract class

Methods:

SetPrice1(): No Action

SetPrice2(Int)

This Class extends SetPrice() abstract class

Methods:

SetPrice2(): sets Price by type of Gas

SetInitialvalues

Methods:

SetInitialValues(): This is a abstract method

SetInitialvalues1

This Class extends SetInitialvalues () abstract class

Methods:

SetInitialValues1(): Sets L to 0 and set total to 0

SetInitialvalues

This Class extends SetInitialvalues () abstract class

Methods:

SetInitialValues2(): Sets G to 0 and set total to 0

PumpGasUnit

Methods:

PumpGasUnit(): This is a abstract method

PumpGasUnit1

This Class extends PumpGasUnit () abstract class

Methods:

PumpGasUnit1():Sets L by getting L and Calculats Total and Sets Total

PumpGasUnit2

This Class extends PumpGasUnit () abstract class

Methods:

PumpGasUnit2(): Sets G by getting G and Calculats Total and Sets Total

GasAckMsg

Methods:

GasAckMsg(): This is a abstract method

GasAckMsg1

This Class extends GasAckMsg1() abstract class

Methods:

GasAckMsg1 (): Displays Number of Liters pumped and options available

GasAckMsg2

This Class extends GasAckMsg2() abstract class

Methods:

GasAckMsg2(): Displays Number of Gallons pumped and options available

PrintReceipt

Methods:

PrintReceipt(): This is a abstract method

PrintReceipt1

This Class extends PrintReceipt() abstract class

Methods:

PrintReceipt1(): Displays Receipt

PrintReceipt2

This Class extends PrintReceipt() abstract class

Methods:

PrintReceipt2(): Displays Receipt

CancelMsg

Methods:

CancelMsg(): This is a abstract method

CancelMsg1

This Class extends CancelMsg() abstract class

Methods:

CancelMsg1(): Displays Transaction Cancelled Message and available options

CancelMsg2

This Class extends CancelMsg() abstract class

Methods:

CancelMsg2(): Displays Transaction Cancelled Message and available options

ReturnCash

Methods:

ReturnCash(): This is a abstract method

ReturnCash1

This Class extends ReturnCash() abstract class

Methods:

ReturnCash1(): Return Action by getting getCash and getTotal

ReturnCash2

This Class extends ReturnCash() abstract class

Methods:

ReturnCash2(): No Action

IncorrectPinMsg

Methods:

InCorrectPinMsg(): This is a abstract method

IncorrectPinMsg1

This Class extends IncorrectPinMsg() abstract class

Methods:

InCorrectPinMsg1(): No Action

IncorrectPinMsg2

This Class extends IncorrectPinMsg() abstract class

Methods:

InCorrectPinMsg2(): Prints InCorrectPin Msg and displays enter pin option.

StorePin

Methods:

StorePin(): This is a abstract method

StorePin1

This Class extends StorePin() abstract class

Methods:

StorePin1(): No Action

StorePin2

This Class extends StorePin() abstract class

Methods:

StorePin2(): get Temp_p value and SetPin

EnterPinMsg

Methods:

EnterPinMsg(): This is a abstract method

EnterPinMsg1

This Class extends EnterPinMsg() abstract class

Methods:

EnterPinMsg1(): No Action

EnterPinMsg2

This Class extends EnterPinMsg() abstract class

Methods:

EnterPinMsg2(): Prints Pin message

SetW(int)

Methods:

SetW(): This is a abstract method

SetW1(int)

This Class extends SetW1() abstract class

Methods:

SetW1(): Changes Flag according to Payment Type

SetW2(int)

This Class extends SetW2() abstract class

Methods:

SetW2(): No Action

EjectCard

Methods:

EjectCard(): This is a abstract method

EjectCard1

This Class extends EjectCard1() abstract class

Methods:

EjectCard1(): Prints Ejected Card Message

EjectCard2

This Class extends EjectCard2() abstract class

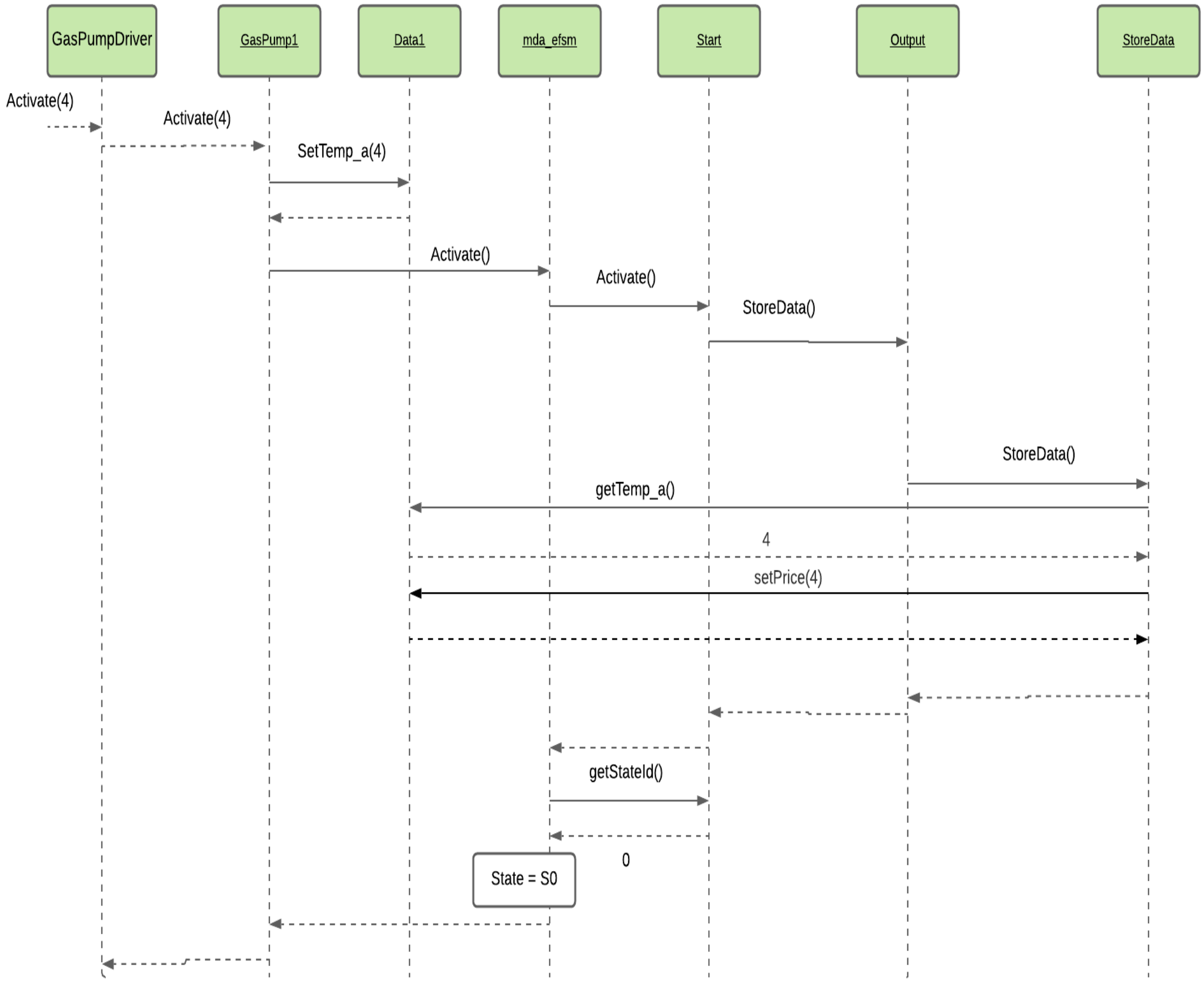
Methods:

EjectCard2(): Prints Ejected Card Message

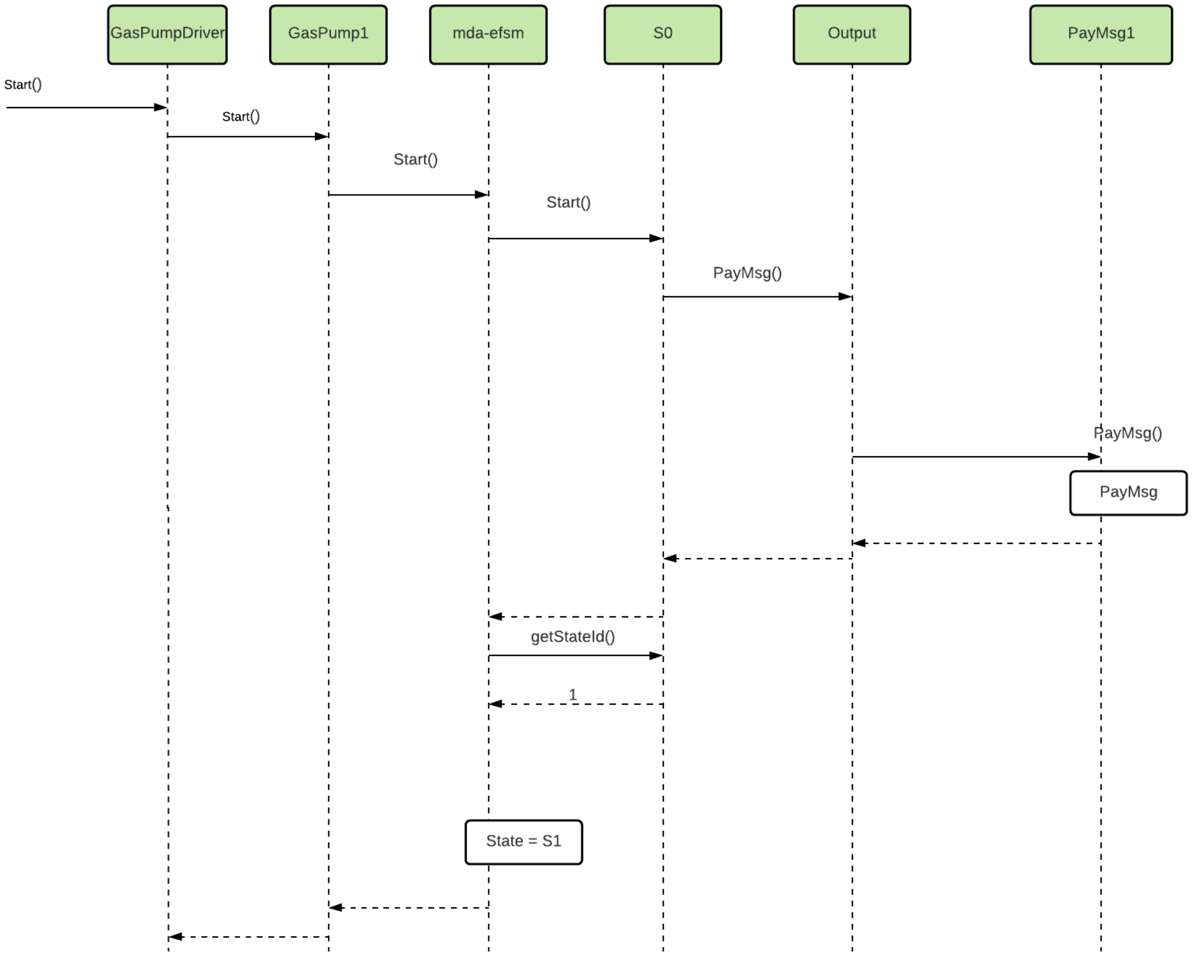
Dynamics

Scenario-I should show how one liter of gas is disposed in GasPump-1, i.e., the following sequence of operations is issued: Activate(4), Start(), PayCash(5), StartPump(), PumpLiter(), PumpLiter

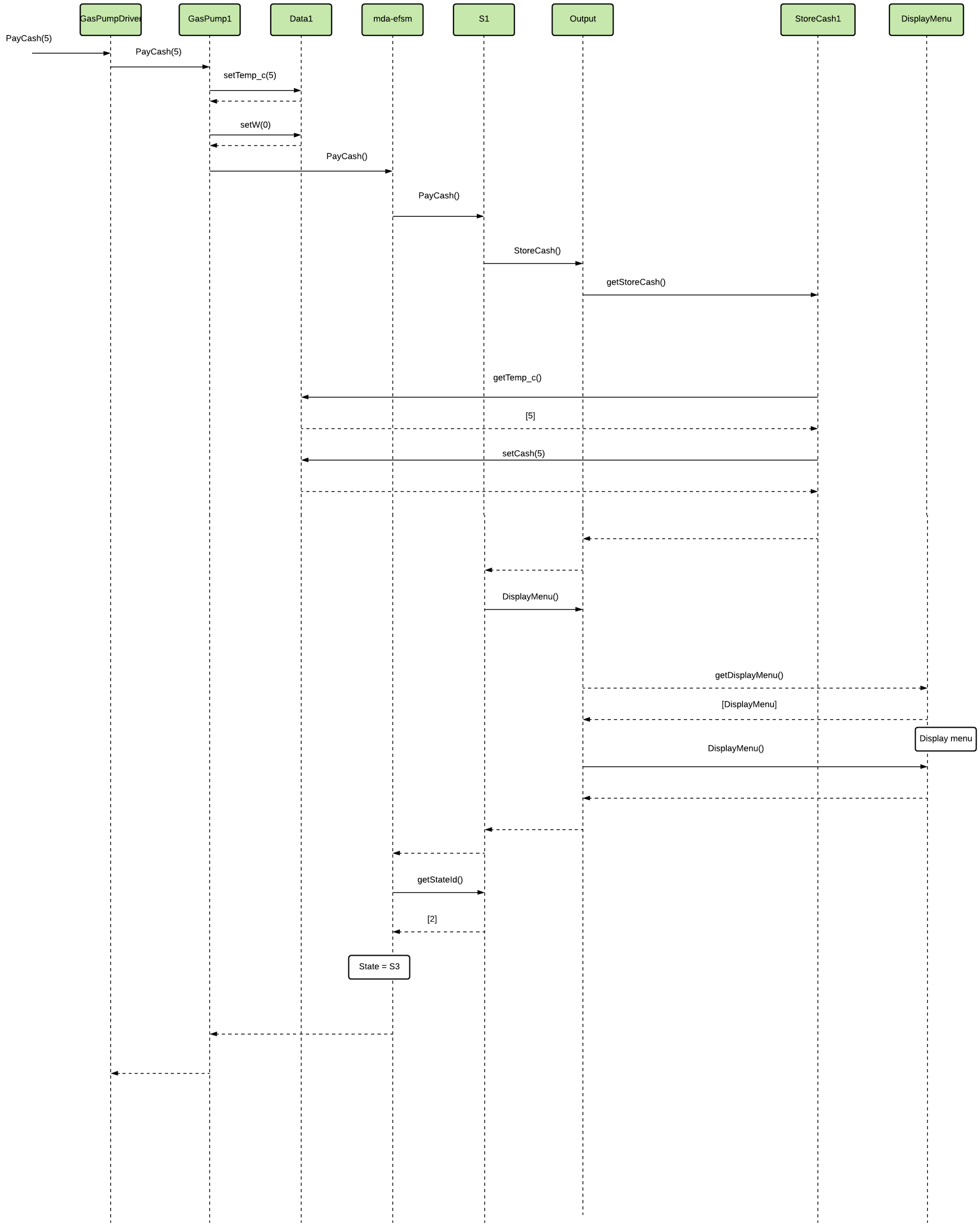
Activate(4)



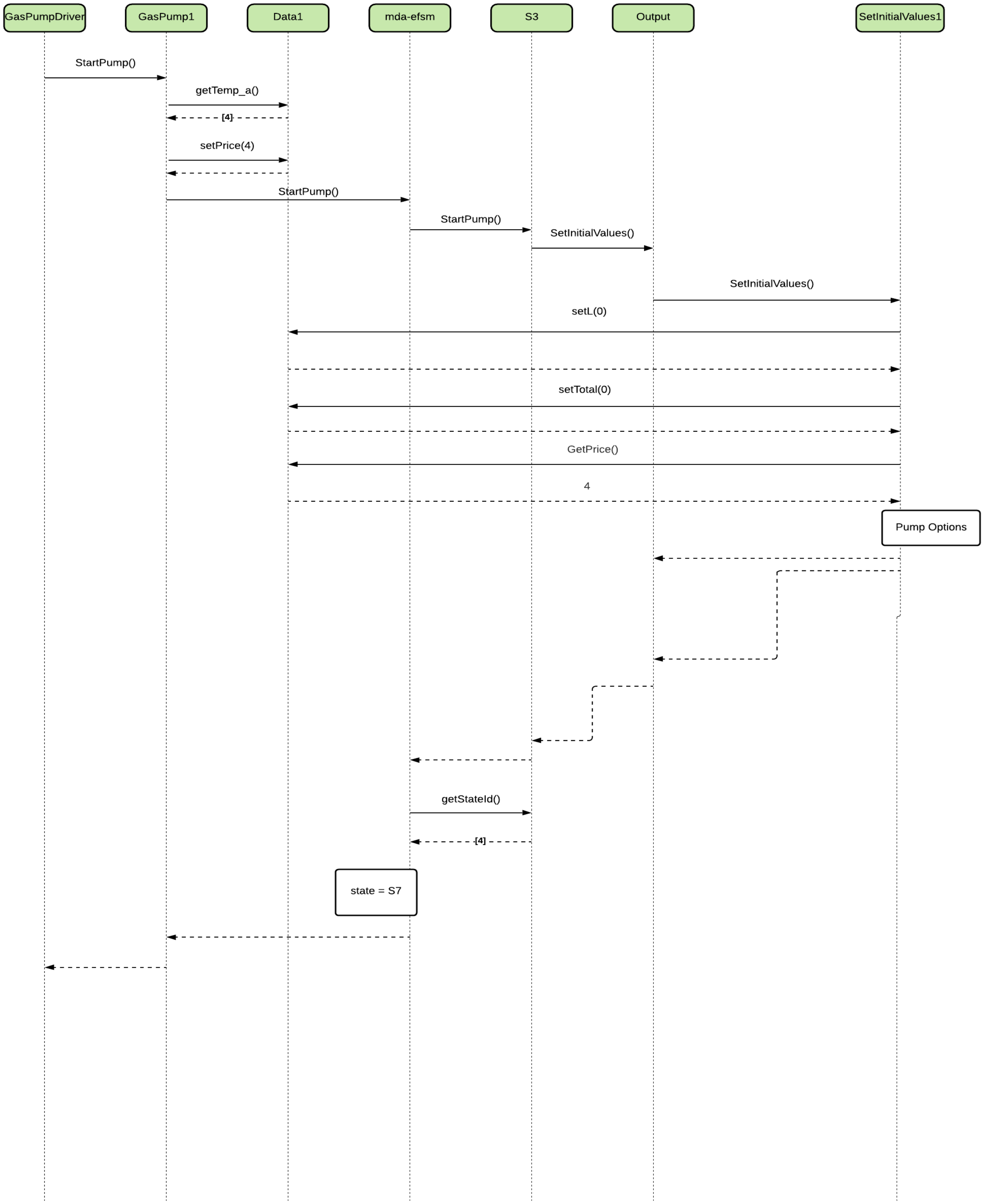
Start()


GasPumpDriver
GasPump1
mda-efsm
S0
Output
PayMsg1
Start()
Start()
Start()
Start()
PayMsg()
PayMsg()
PayMsg
getStateId()
1
State = S1

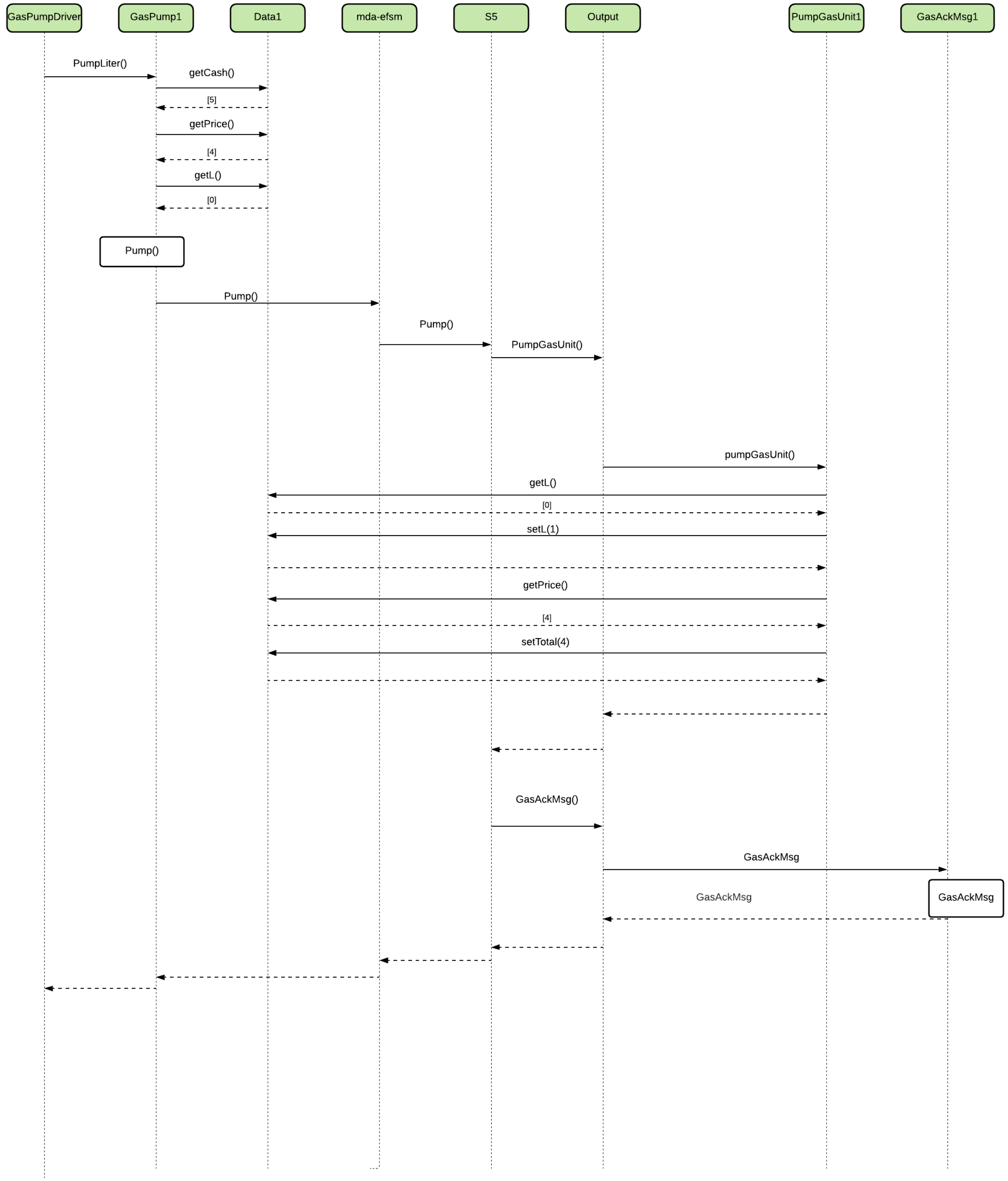
PayCash()


GasPumpDriver
GasPump1
Data1
mda-efsm
S1
Output
StoreCash1
DisplayMenu
PayCash(5)
PayCash(5)
setTemp_c(5)
setW(0)
PayCash()
PayCash()
StoreCash()
getStoreCash()
getTemp_c()
[5]
setCash(5)
DisplayMenu()
getDisplayMenu()
[DisplayMenu]
Display menu
DisplayMenu()
getStateId()
[2]
State = S3

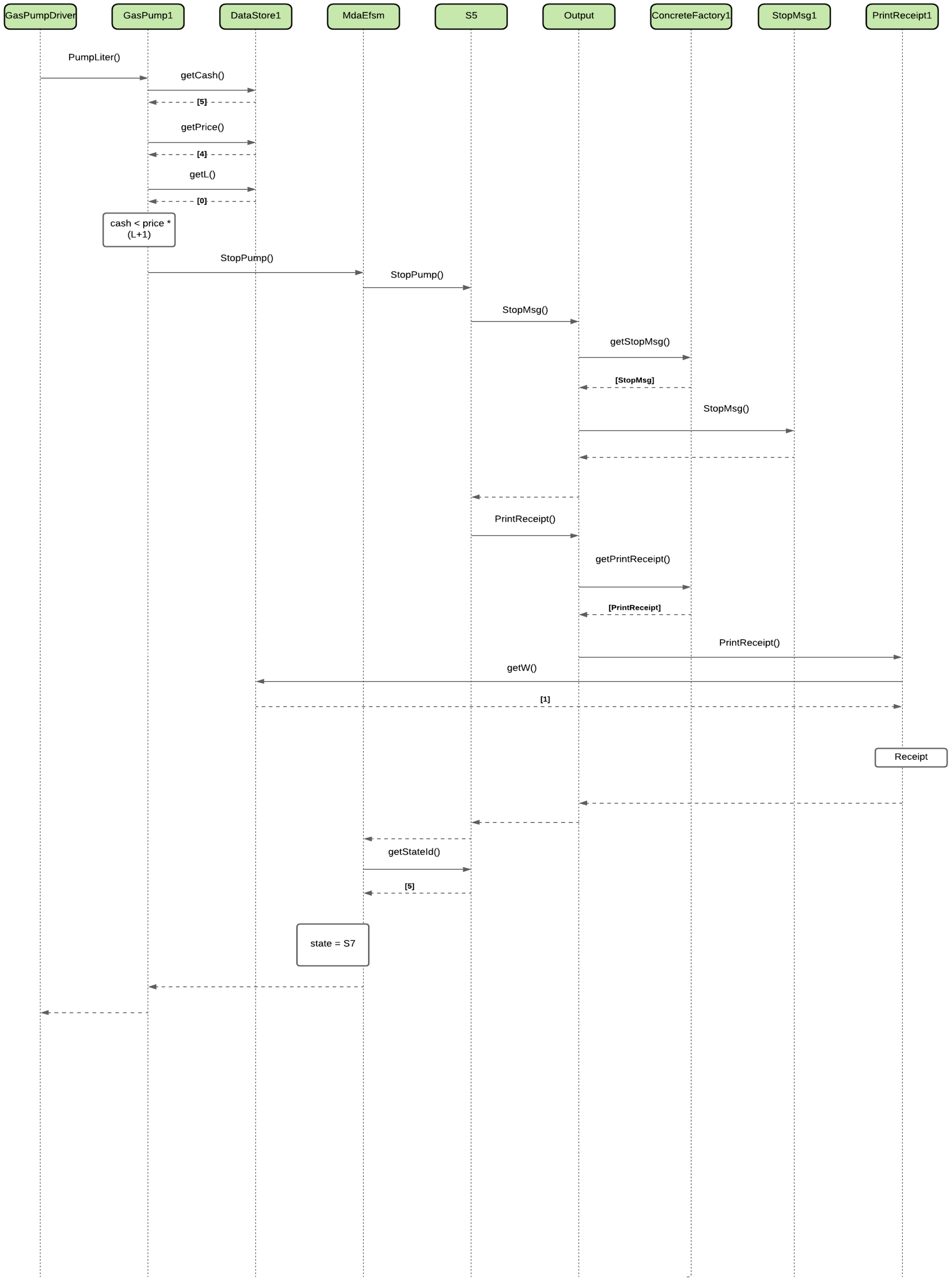
StartPump()


GasPumpDriver
GasPump1
Data1
mda-efsm
S3
Output
SetInitialValues1
StartPump()
getTemp_a()
[4]
setPrice(4)
StartPump()
StartPump()
SetInitialValues()
SetInitialValues()
setL(0)
setTotal(0)
GetPrice()
4
Pump Options
getStateId()
[4]
state = S7

Pumpliter()

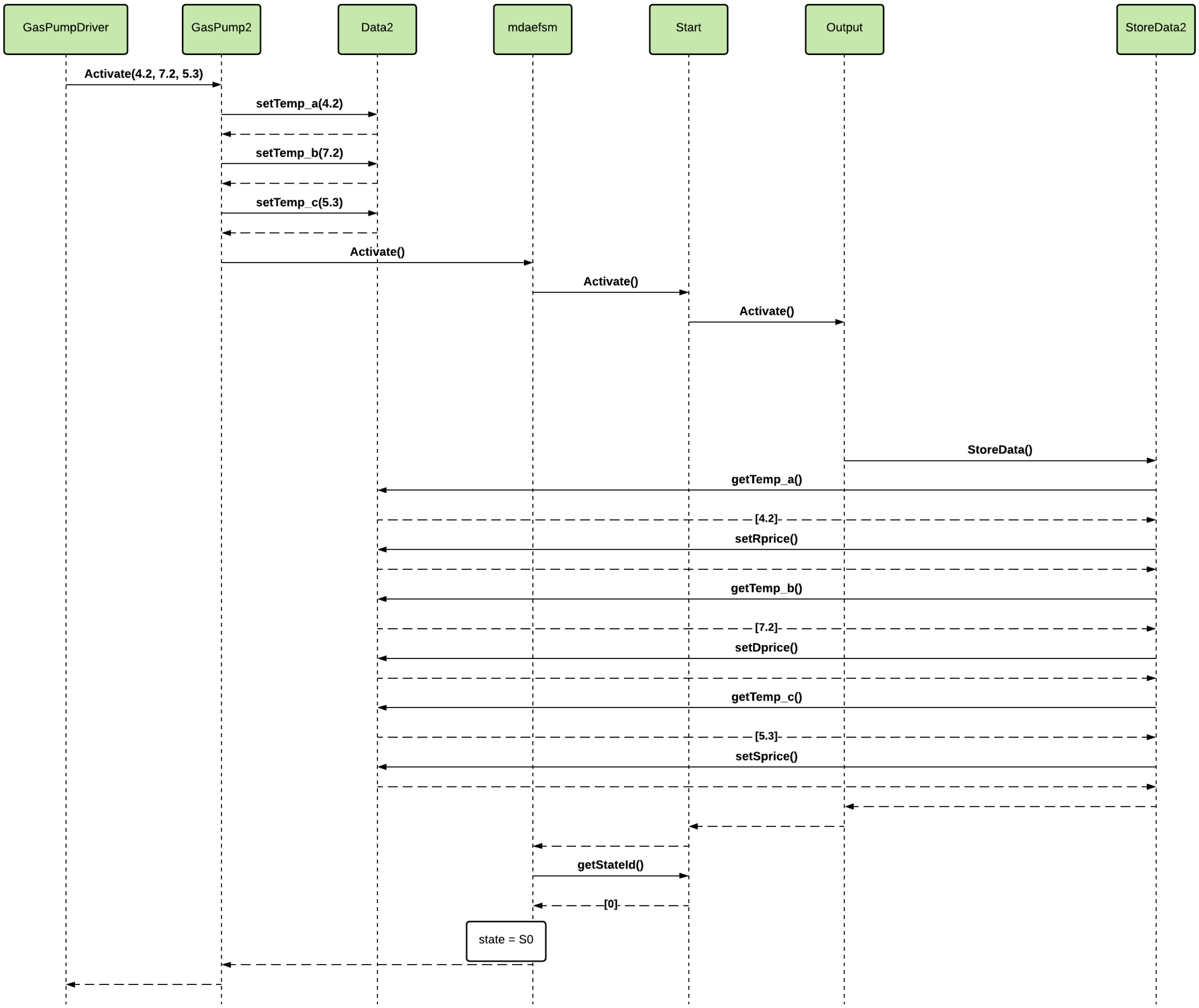

GasPumpDriver
GasPump1
Data1
mda-efsm
S5
Output
PumpGasUnit1
GasAckMsg1
PumpLiter()
getCash()
[5]
getPrice()
[4]
getL()
[0]
Pump()
Pump()
Pump()
PumpGasUnit()
pumpGasUnit()
getL()
[0]
setL(1)
getPrice()
[4]
setTotal(4)
GasAckMsg()
GasAckMsg
GasAckMsg
GasAckMsg

PumpLiter()

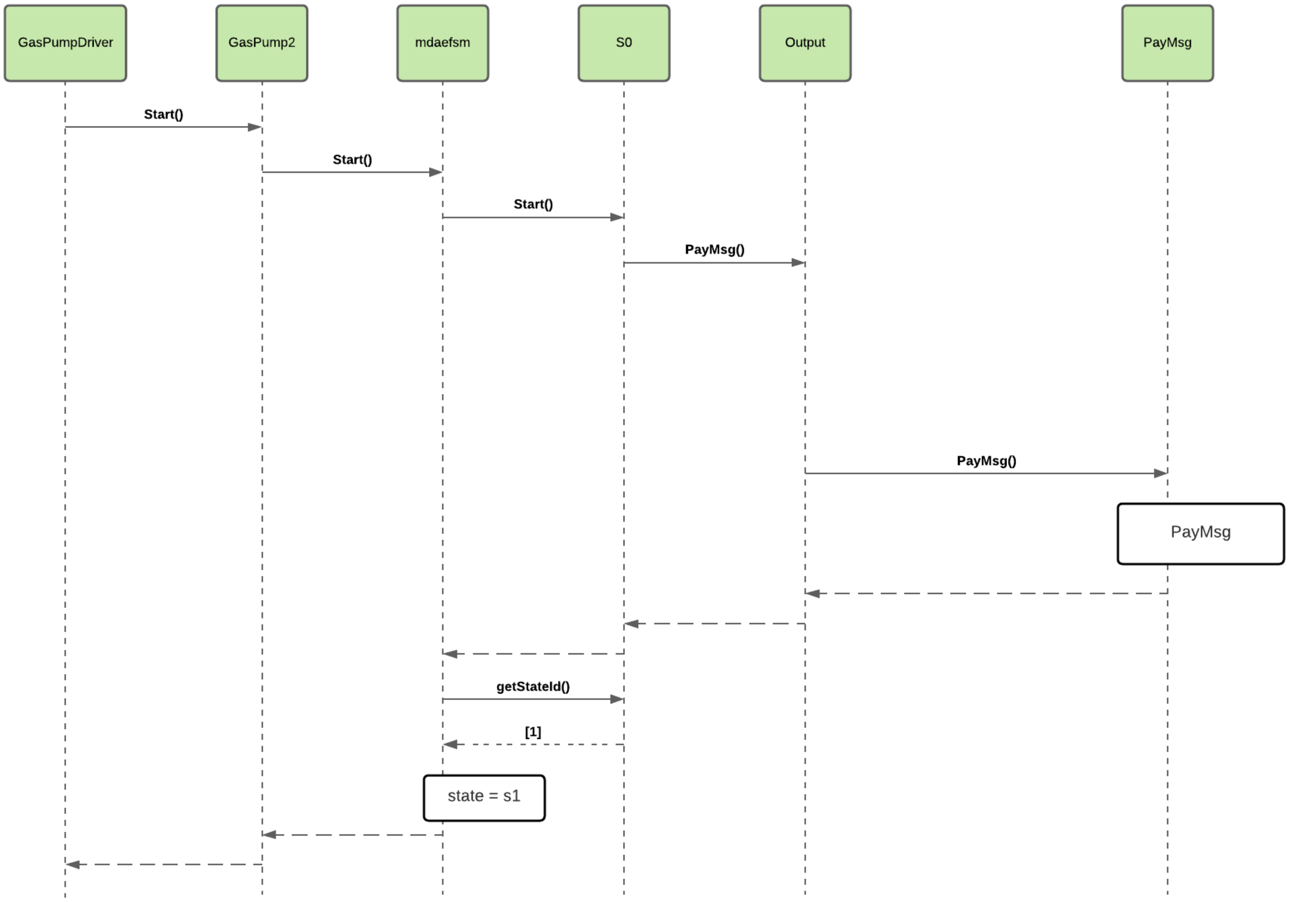

GasPumpDriver
GasPump1
DataStore1
MdaEfsm
S5
Output
ConcreteFactory1
StopMsg1
PrintReceipt1
PumpLiter()
getCash()
[5]
getPrice()
[4]
getL()
[0]
cash < price * (L+1)
StopPump()
StopPump()
StopMsg()
getStopMsg()
[StopMsg]
StopMsg()
PrintReceipt()
getPrintReceipt()
[PrintReceipt]
PrintReceipt()
getW()
[1]
Receipt
getStateId()
[5]
state = S7

Scenario-II should show how one gallon of Super gas is disposed in GasPump-2, i.e., the following sequence of operations is issued: Activate(4.2, 7.2, 5.3), Start(), PayDebit(“abc”), Pin(“cba”), Pin(“abc”), Super(), StartPump(), PumpGallon(), FullTank()

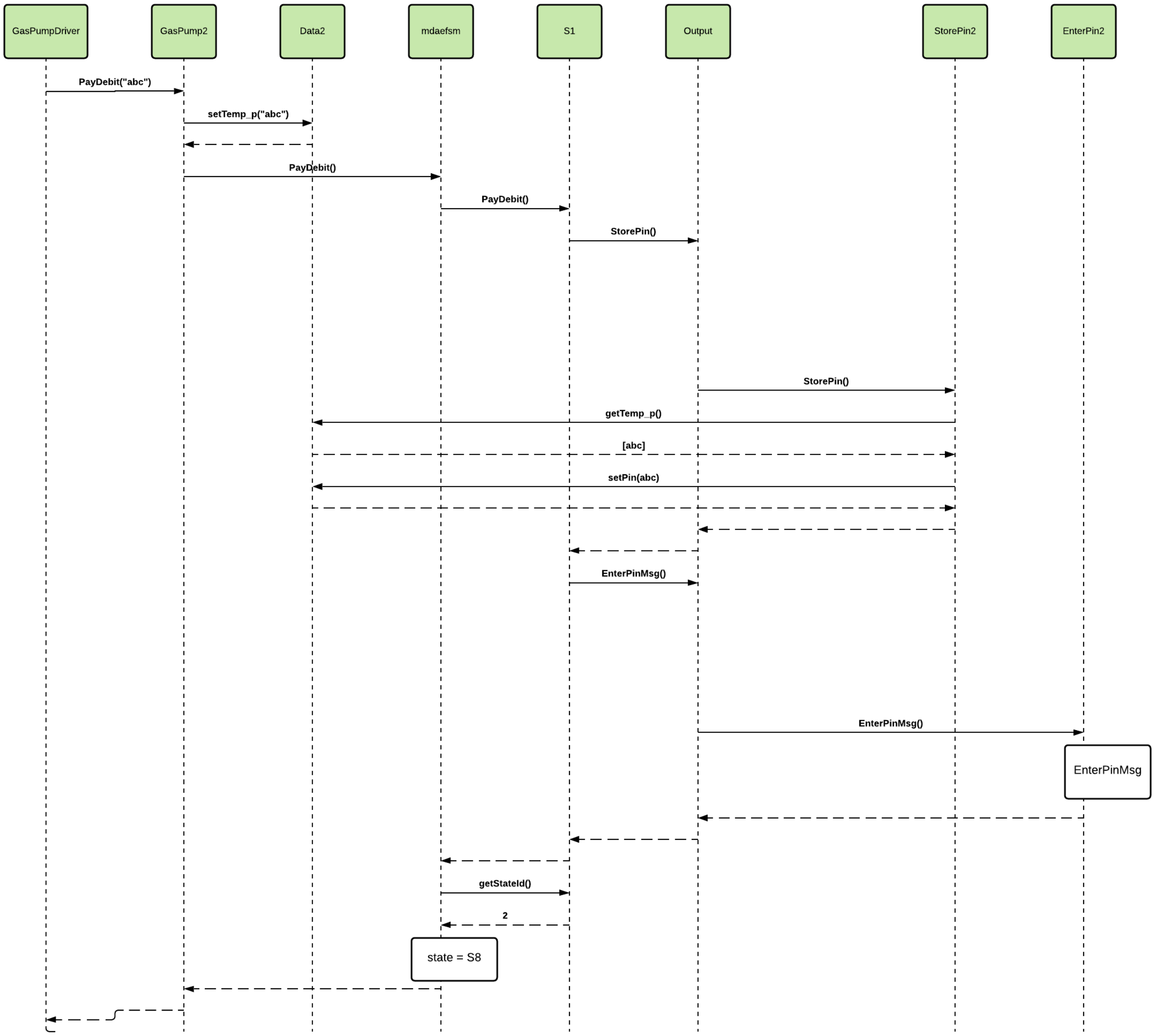
Activate(4.2, 7.2, 5.3)



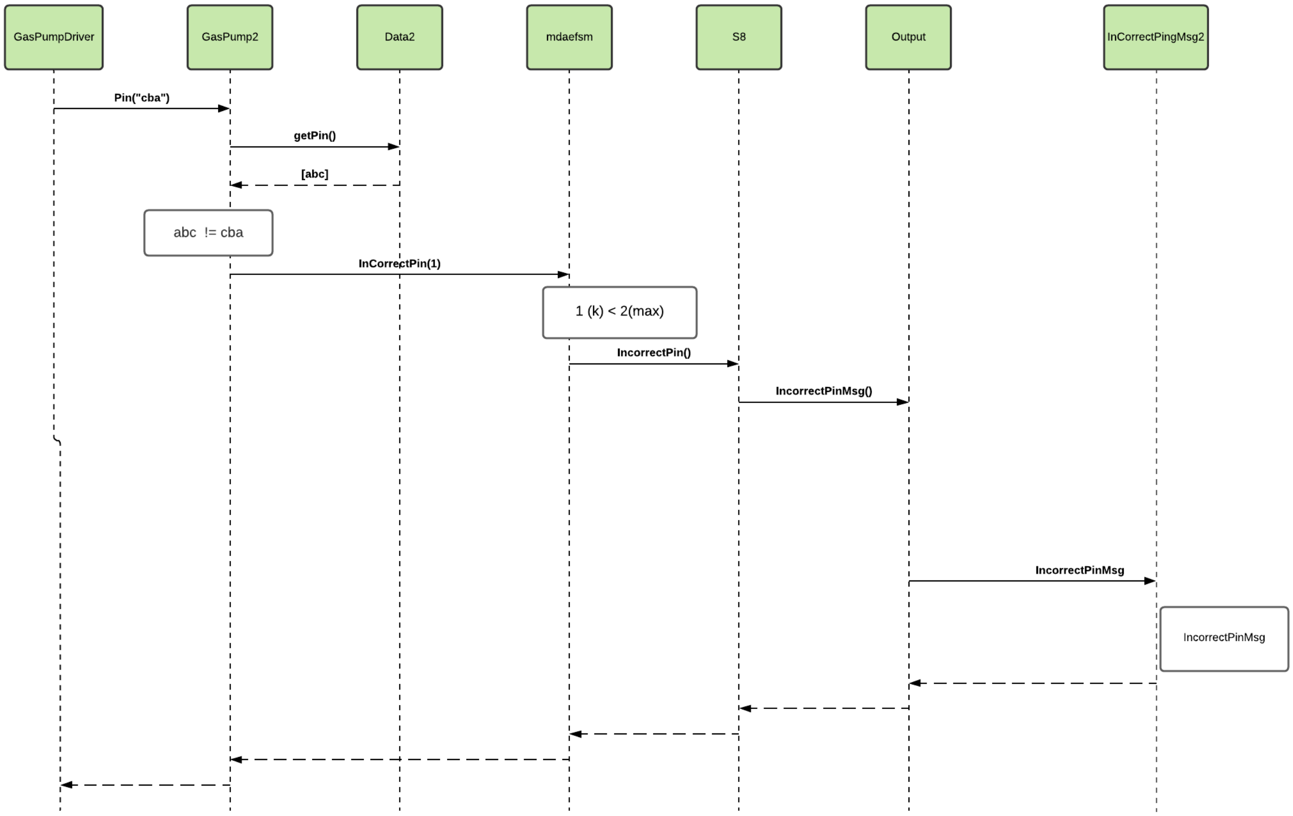
Start()


GasPumpDriver
GasPump2
mdaefsm
S0
Output
PayMsg
Start()
Start()
Start()
PayMsg()
PayMsg()
PayMsg
getStateId()
[1]
state = s1

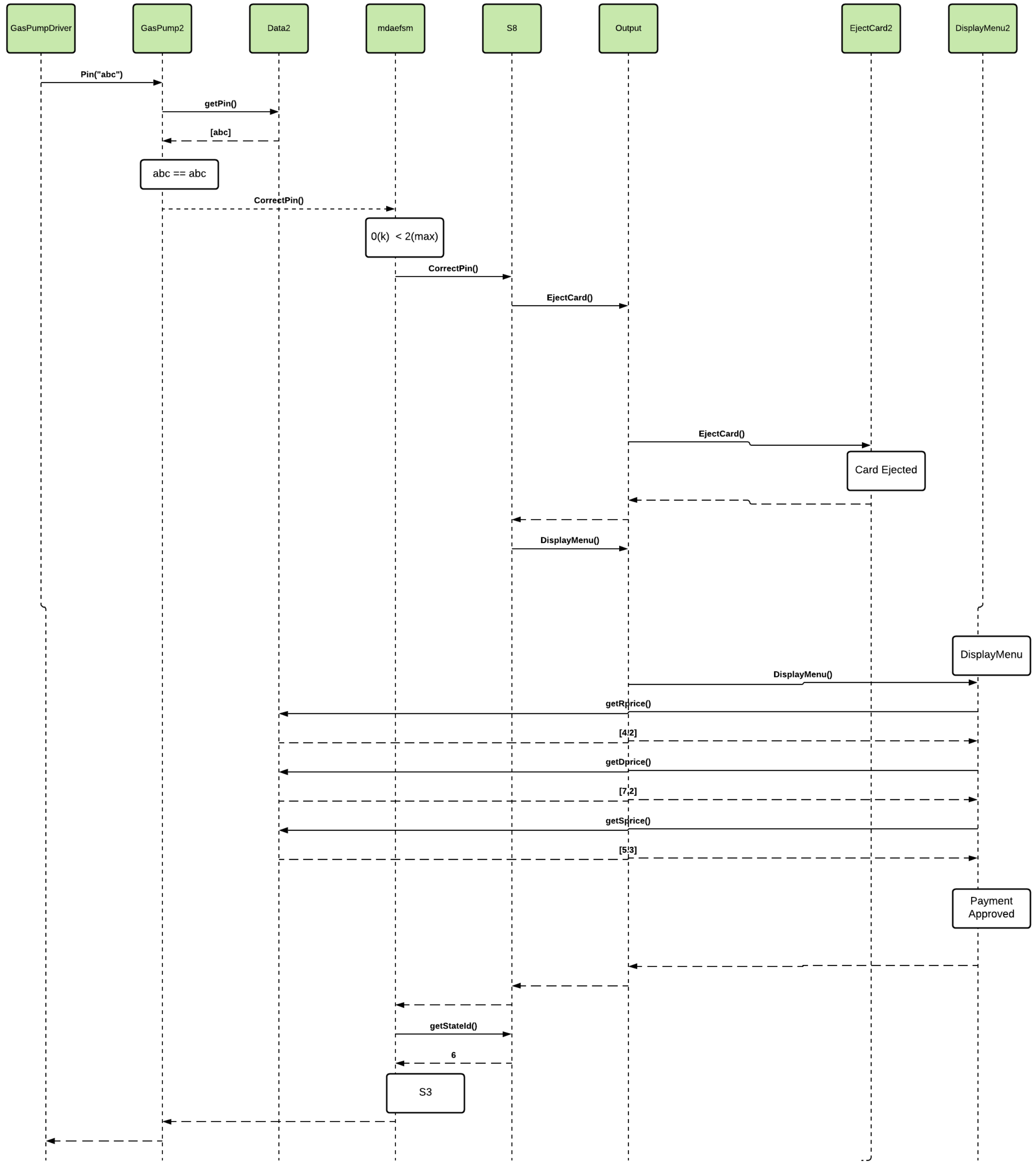
PayDebit(“abc”)



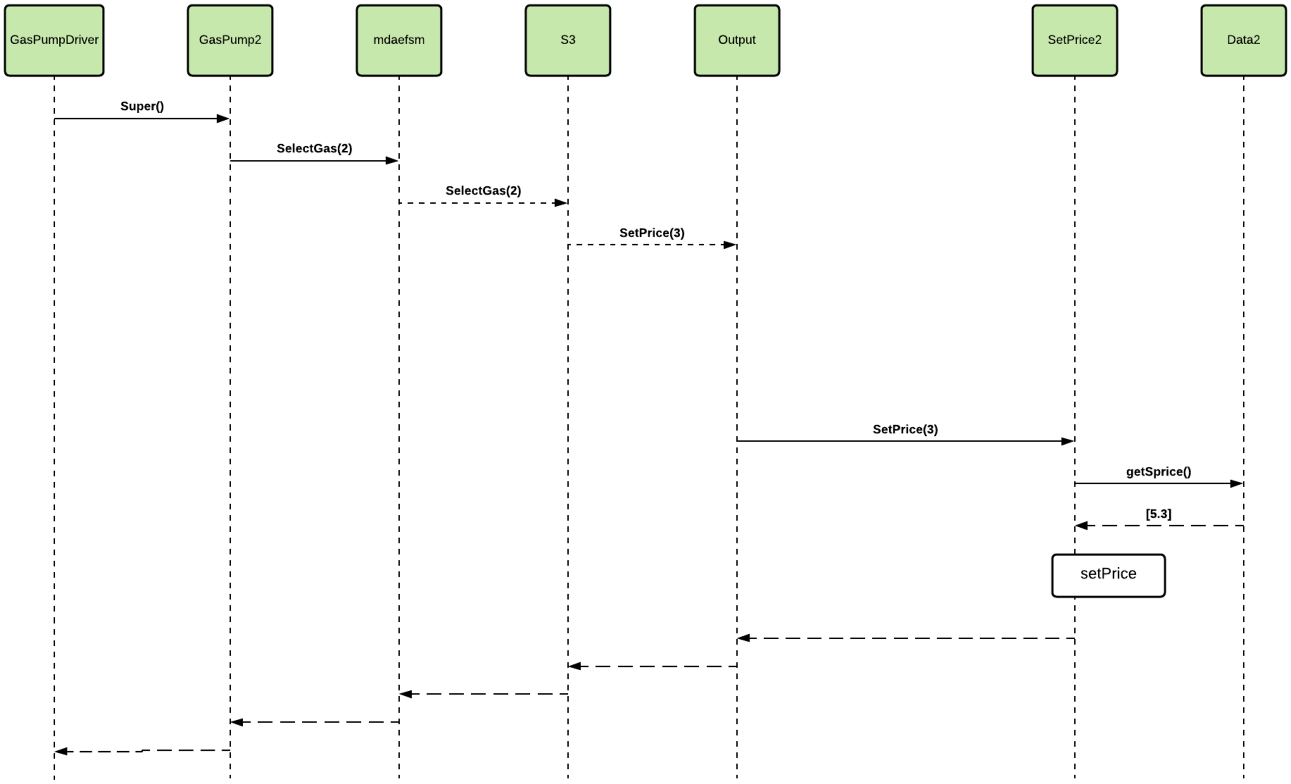
Pin(“cba”)


GasPumpDriver
GasPump2
Data2
mdaefsm
S8
Output
InCorrectPingMsg2
Pin("cba")
getPin()
[abc]
abc != cba
InCorrectPin(1)
1 (k) < 2(max)
IncorrectPin()
IncorrectPinMsg()
IncorrectPinMsg
IncorrectPinMsg

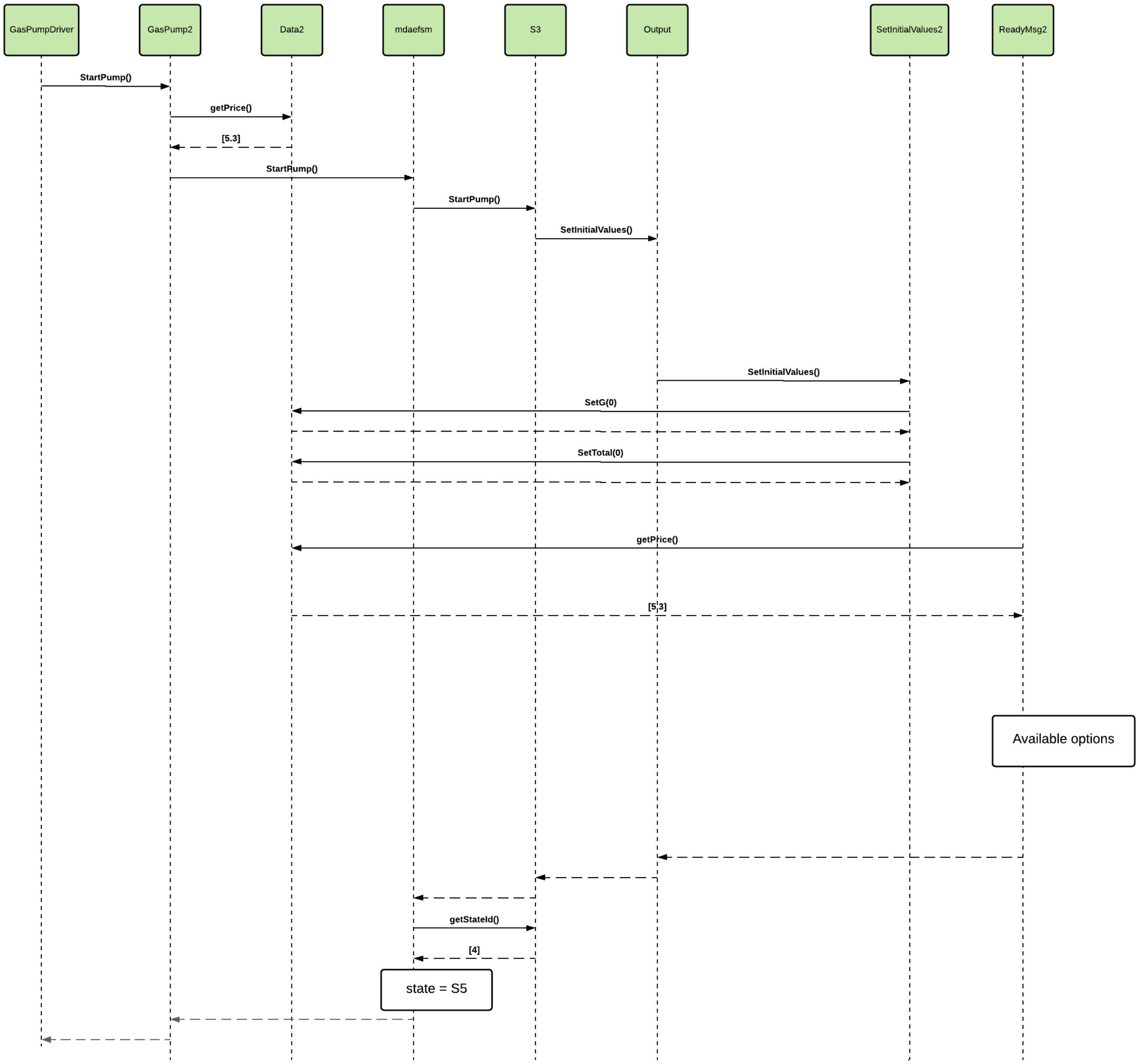
Pin(“abc”)


GasPumpDriver
GasPump2
Data2
mdaefsm
S8
Output
EjectCard2
DisplayMenu2
Pin("abc")
getPin()
[abc]
abc == abc
CorrectPin()
0(k) < 2(max)
CorrectPin()
EjectCard()
EjectCard()
Card Ejected
DisplayMenu()
DisplayMenu
DisplayMenu()
getRprice()
[4.2]
getDprice()
[7.2]
getSprice()
[5.3]
Payment Approved
getStateId()
6
S3

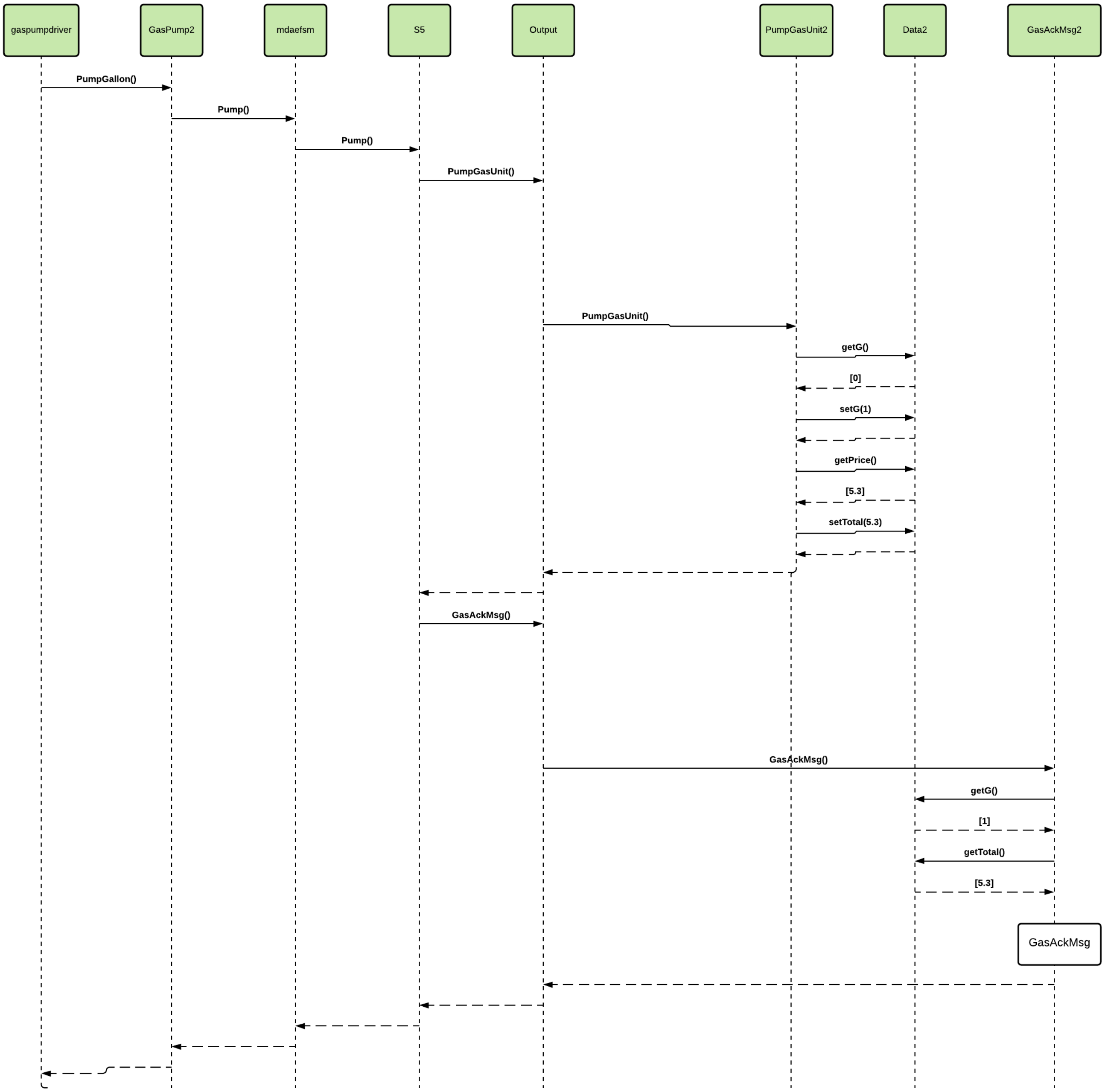
Super()


GasPumpDriver
GasPump2
mdaefsm
S3
Output
SetPrice2
Data2
Super()
SelectGas(2)
SelectGas(2)
SetPrice(3)
SetPrice(3)
getSprice()
[5.3]
setPrice

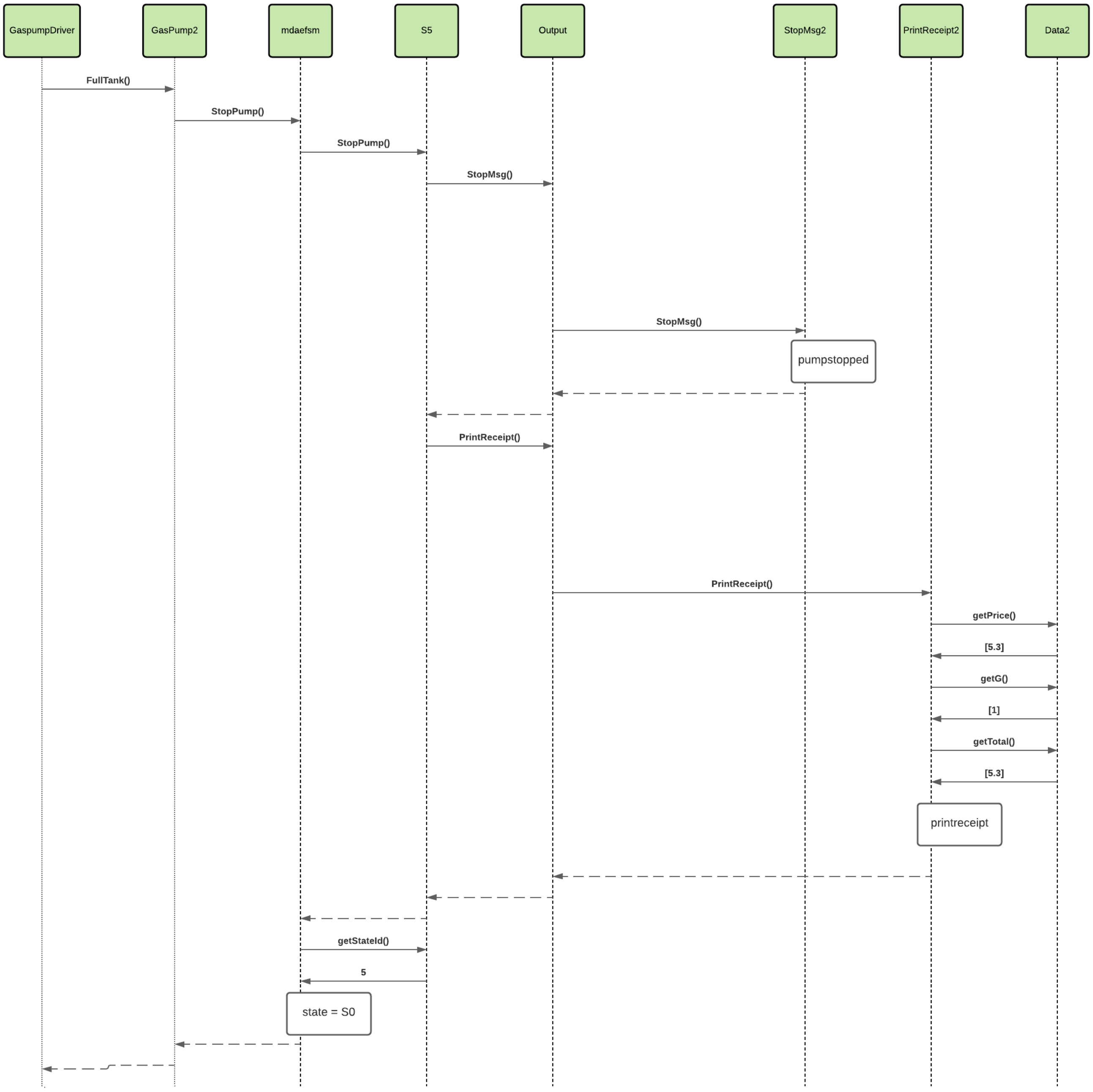
StartPump()


GasPumpDriver
GasPump2
Data2
mdaefsm
S3
Output
SetInitialValues2
ReadyMsg2
StartPump()
getPrice()
[5.3]
StartPump()
StartPump()
SetInitialValues()
SetInitialValues()
SetG(0)
SetTotal(0)
getPrice()
[5.3]
Available options
getStateId()
[4]
state = S5

PumpGallon()


gaspumpdriver
GasPump2
mdaefsm
S5
Output
PumpGasUnit2
Data2
GasAckMsg2
PumpGallon()
Pump()
Pump()
PumpGasUnit()
PumpGasUnit()
getG()
[0]
setG(1)
getPrice()
[5.3]
setTotal(5.3)
GasAckMsg()
GasAckMsg()
getG()
[1]
getTotal()
[5.3]
GasAckMsg

FullTank()


GaspumpDriver
GasPump2
mdaefsm
S5
Output
StopMsg2
PrintReceipt2
Data2
FullTank()
StopPump()
StopPump()
StopMsg()
StopMsg()
pumpstopped
PrintReceipt()
PrintReceipt()
getPrice()
[5.3]
getG()
[1]
getTotal()
[5.3]
printreceipt
getStateId()
5
state = S0