Sai Krishna Akula - A20396253

Gas Pump

Report

Table of Contents

[1. MDA-EFSM Model: 2](#_Toc57199353)

[1.1. Meta Events: 2](#_Toc57199354)

[1.2. Meta Actions: 3](#_Toc57199355)

[1.3. State Diagram: 4](#_Toc57199356)

[1.4. Pseudo Code: 5](#_Toc57199357)

[2. Class Diagrams: 10](#_Toc57199358)

[2.1. Overall MDA Class Diagram: 10](#_Toc57199359)

[2.2. State Pattern: 11](#_Toc57199360)

[2.3. Strategy Pattern: 12](#_Toc57199361)

[2.4. Abstract Factory Pattern: 13](#_Toc57199362)

[3. Sequence Diagrams: 14](#_Toc57199363)

[3.1. Scenario-I: 14](#_Toc57199364)

[3.1.1. Activate: 14](#_Toc57199365)

[3.1.2. Start: 14](#_Toc57199366)

[3.1.3. PayCash: 15](#_Toc57199367)

[3.1.4. StartPump: 15](#_Toc57199368)

[3.1.5. PumpLiter: 16](#_Toc57199369)

[3.1.6. PumpLiter: 16](#_Toc57199370)

[3.2. Scenario-II: 17](#_Toc57199371)

[3.2.1. Activate: 17](#_Toc57199372)

[3.2.2. Start: 17](#_Toc57199373)

[3.2.3. PayDebit: 18](#_Toc57199374)

[3.2.4. Pin(CBA): 18](#_Toc57199375)

[3.2.5. Pin(abc): 18](#_Toc57199376)

[3.2.6. Super: 19](#_Toc57199377)

[3.2.7. StartPump: 19](#_Toc57199378)

[3.2.8. PumpGallon: 20](#_Toc57199379)

[3.2.9. FullTank: 20](#_Toc57199380)

Gas Pump Report

Sai Krishna Akula | A20396253 | Section 02

# 1. MDA-EFSM Model:

## 1.1. Meta Events:

* Activate()
* Start()
* PayCredit()
* PayCash()
* PayDebit()
* Reject()
* Cancel()
* Approved()
* StartPump()
* Pump()
* StopPump()
* SelectGas(int g) // Regular: g=1; Super: g=2; Diesel: g=3
* CorrectPin()
* IncorrectPin(int max)

## 1.2. Meta 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
* GasPumpedMsg // displays the amount of disposed gas
* PrintReceipt // print a receipt
* CancelMsg // displays a cancellation message
* ReturnCash // returns the remaining cash
* WrongPinMsg // displays incorrect pin message
* StorePin // stores the pin from the temporary data store
* EnterPinMsg // displays a message to enter pin
* InitializeData // set the value of price to 0 for GP-2; do nothing for GP-1
* EjectCard() // card is ejected
* SetW(int w) // set value for cash flag

## Diagram, schematic Description automatically generated1.3. State Diagram:

## 1.4. Pseudo Code:

**Operations of the Input Processor (GasPump-1)**

Activate(int a) {

if (a>0) {

d->temp\_a=a;

m->Activate()

}

}

Start() {

m->Start();

}

PayCash(float c) {

if (c>0) {

d->temp\_c=c;

m->PayCash()

}

}

PayCredit() {

m->PayCredit();

}

Reject() {

m->Reject();

}

Approved() {

m-> Approved();

}

Cancel() {

m->Cancel();

}

StartPump() {

m->StartPump();

}

PumpLiter() {

if (d->w==1) m->Pump()

else if (d->cash>0)&&(d->cash < d->price\*(d->L+1))

m->StopPump();

else m->Pump()

}

StopPump() {

m->StopPump();

}

**Notice:**

cash: contains the value of cash deposited

price: contains the price of the selected gas

L: contains the number of litters already pumped

w: cash flag (cash: w=0; otherwise: 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) {

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\_p=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();

}

**Notice:**

pin: contains the pin in the data store

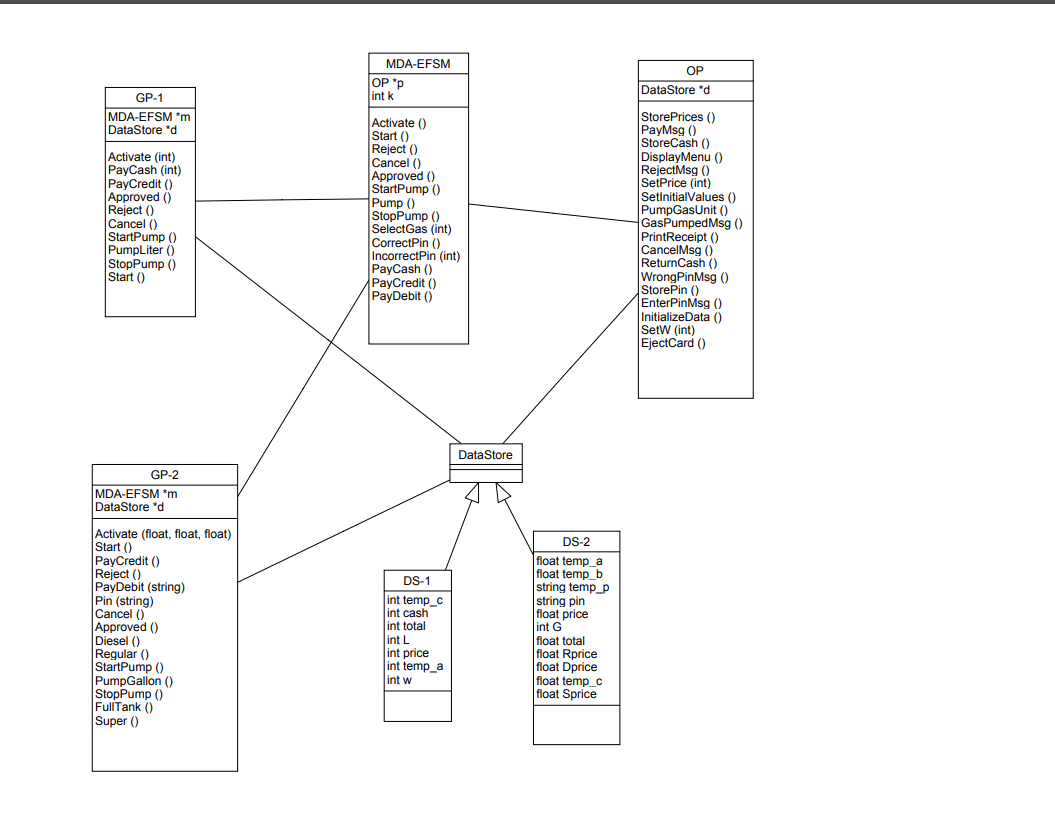
m: is a pointer to the MDA-EFSM object

d: is a pointer to the Data Store object

SelectGas(g): Regular: g=1; Super: g=2; Diesel: g=3

# 2. Class Diagrams:

## 2.1. Overall MDA Class Diagram:

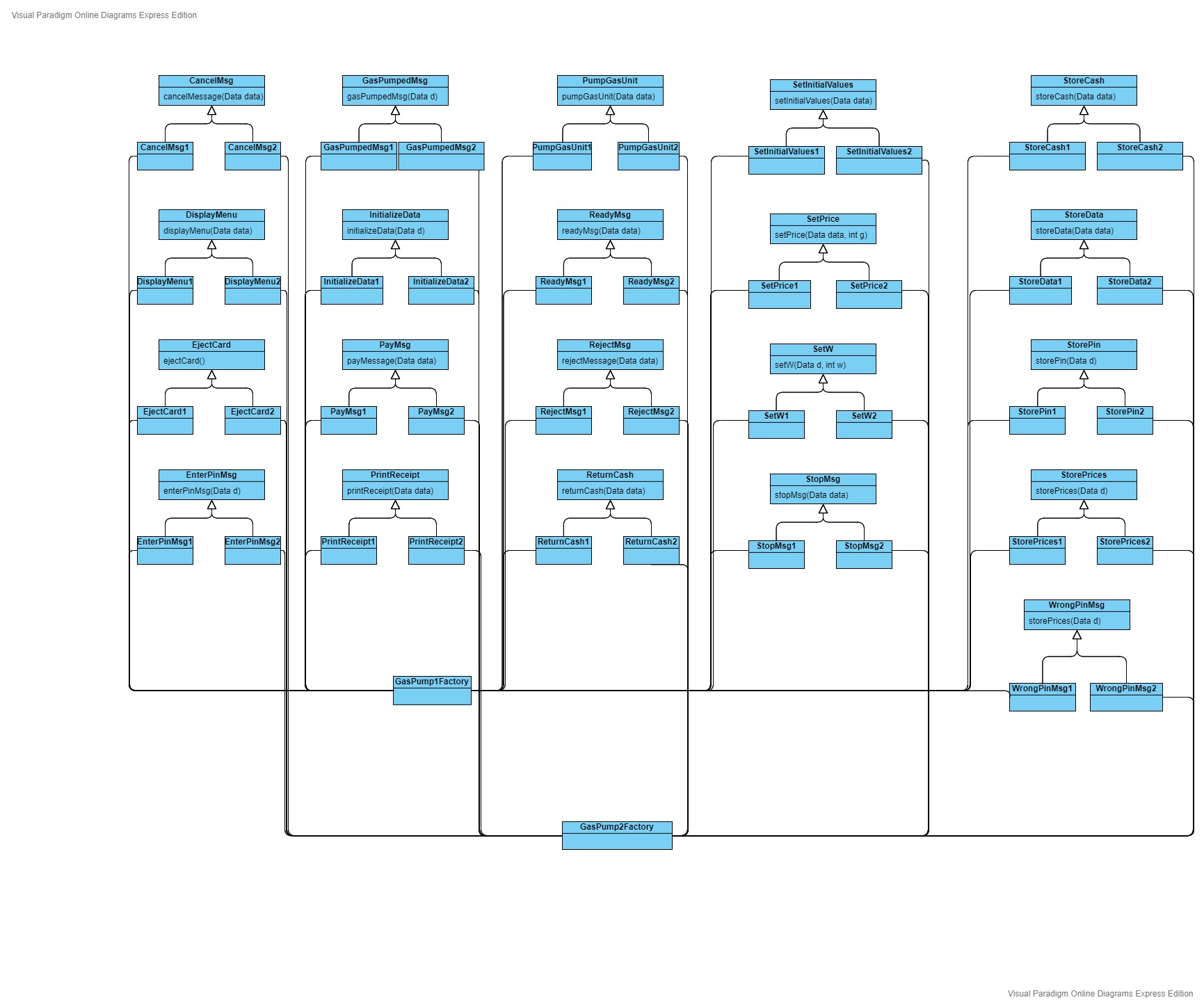


All the three patterns will be shown in the following sections, as class diagram with all patterns will be hard to read, I have divided them as below.

## 2.2. State Pattern:

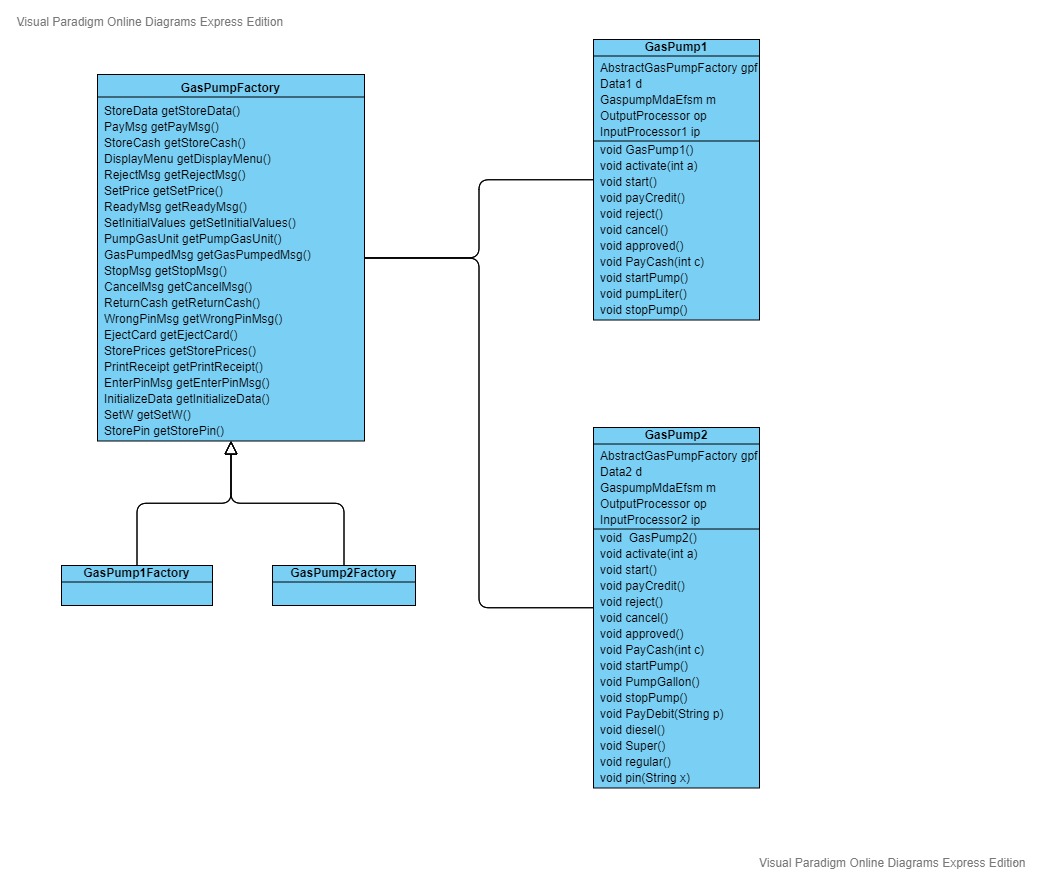
* All the state changes are done in the class “GasPumpMdaEfsm” by the function changeState centrally.
* State is a abstract class for all the states available in the system.
* Initial State implements the active function which is used to set the gas price and it has the stated as 0.
* S0 is the state has start which is used to call the initialize data function for the gas pump like the L value and payment types message is displayed to the user. State id is 1.
* S1 is state calls the appropriate implementation for the payments related to gas pump. State id is 2.
* S2 is special state for credit card payment, where either credit card will be approved or rejected. State id is 3.
* S3 provides options for the user after payment options are selected, here user will be prompted for selecting the type of gas they want to pump depending upon the gas pump selected. State id is 4.
* S5 to control the gas pump and S6 is used for the debit card payment

## 2.3. Strategy Pattern:

All the actions are performed in the strategy classes, every gas pump have its own strategy class and every action will have an interface which will be extended by the class implementing the function. I have followed a pattern where every class ending with a number signifies gas pump for which it is implemented. GasPump1Factory and GasPump2Factory are responsible for selecting the appropriate strategy class.

## 2.4. Abstract Factory Pattern:

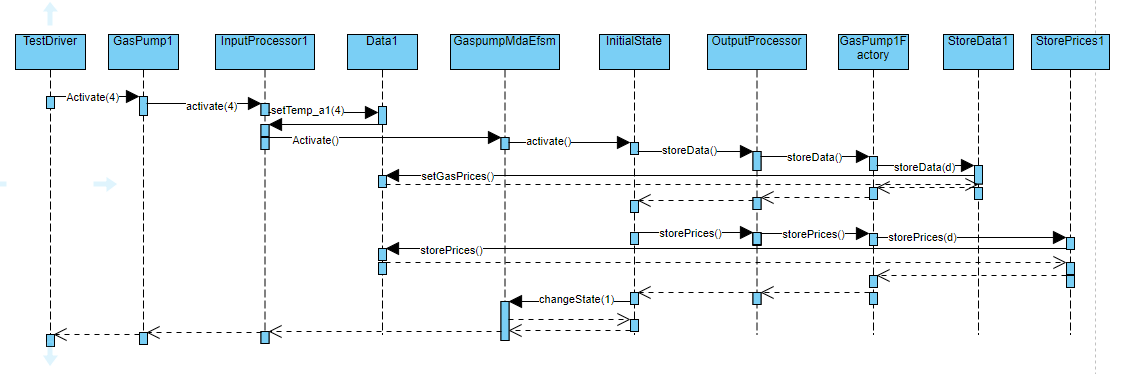
Initialization of the factory is done in the construction of the gaspump1 and gaspump2 class. From previous diagram we can see that gasPump1Factory and gasPump2Factory will be initializing the strategies.



# 3. Sequence Diagrams:

## 3.1. Scenario-I:

### 3.1.1. Activate:



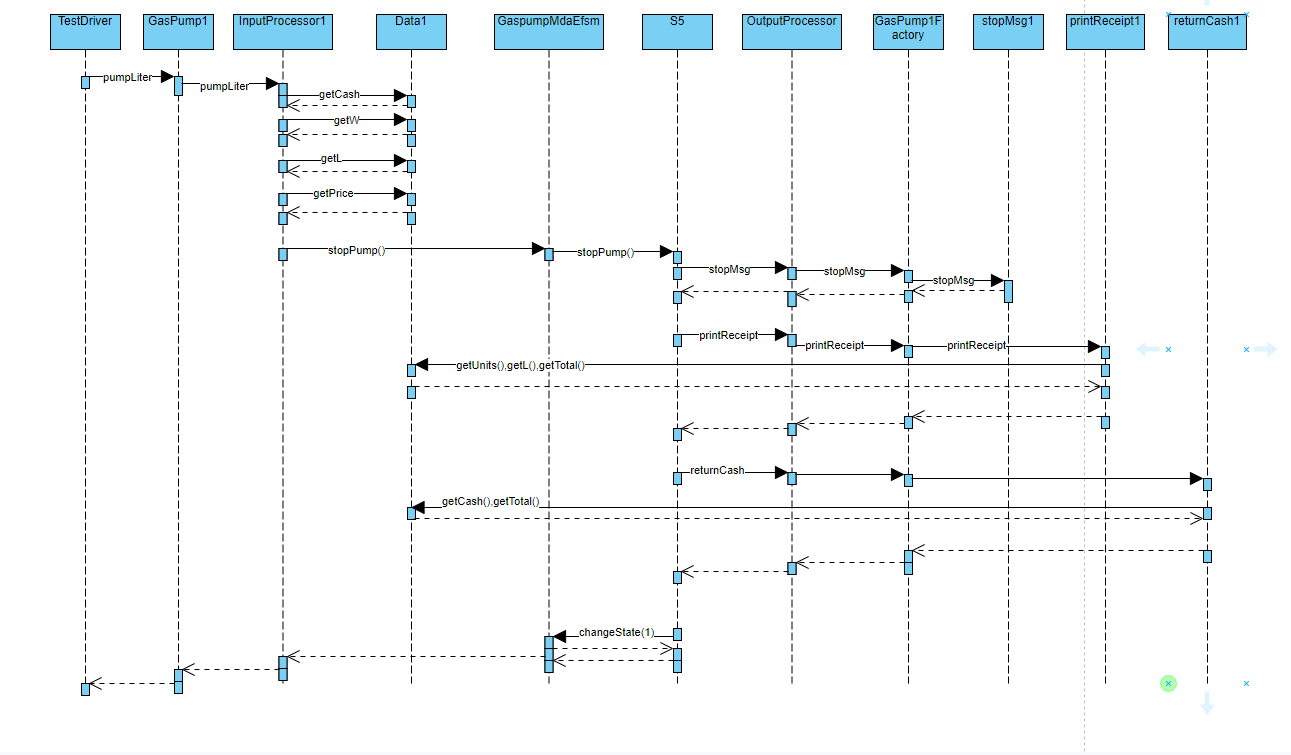
### 3.1.2. Start:

### 3.1.3. PayCash:

### 3.1.4. StartPump:

### 3.1.5. PumpLiter:

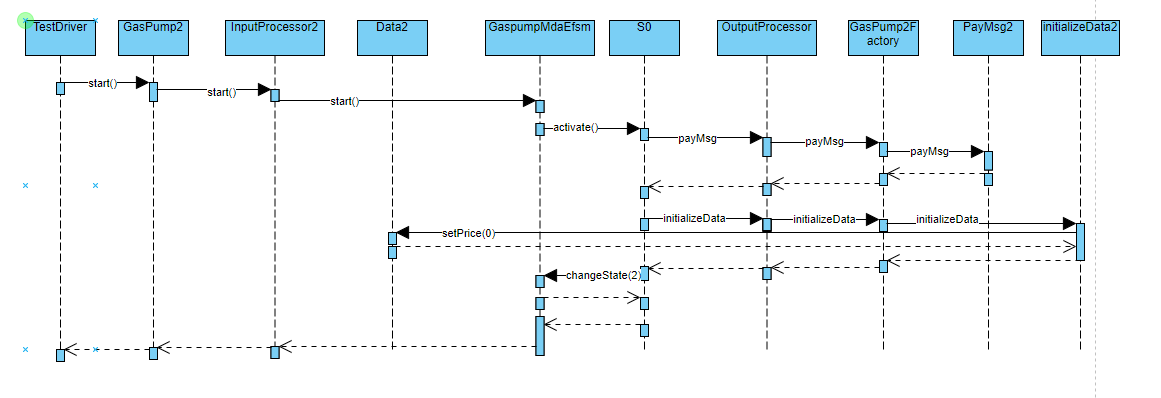
### 3.1.6. PumpLiter:



## 3.2. Scenario-II:

### 3.2.1. Activate:

### 3.2.2. Start:

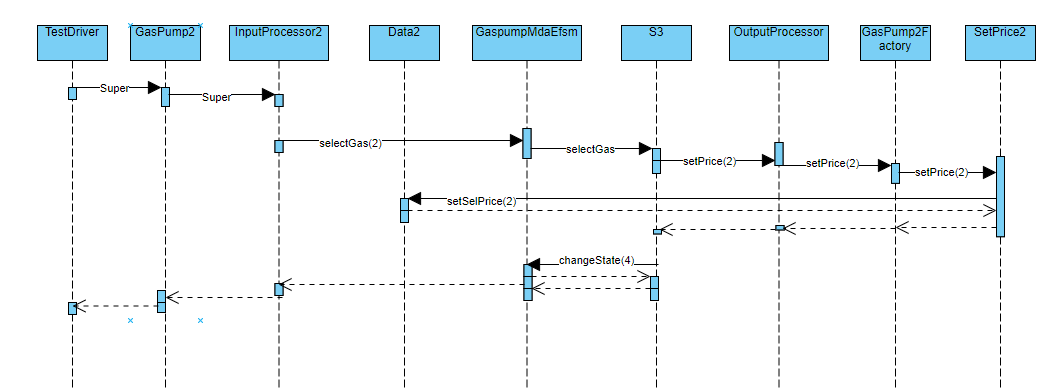


### 3.2.3. PayDebit:

### 3.2.4. Pin(CBA):

### 3.2.5. Pin(abc):

### 3.2.6. Super:



### 3.2.7. StartPump:

### 3.2.8. PumpGallon:

Typo in the above image changeState(6) not changeState(1)

### 3.2.9. FullTank: