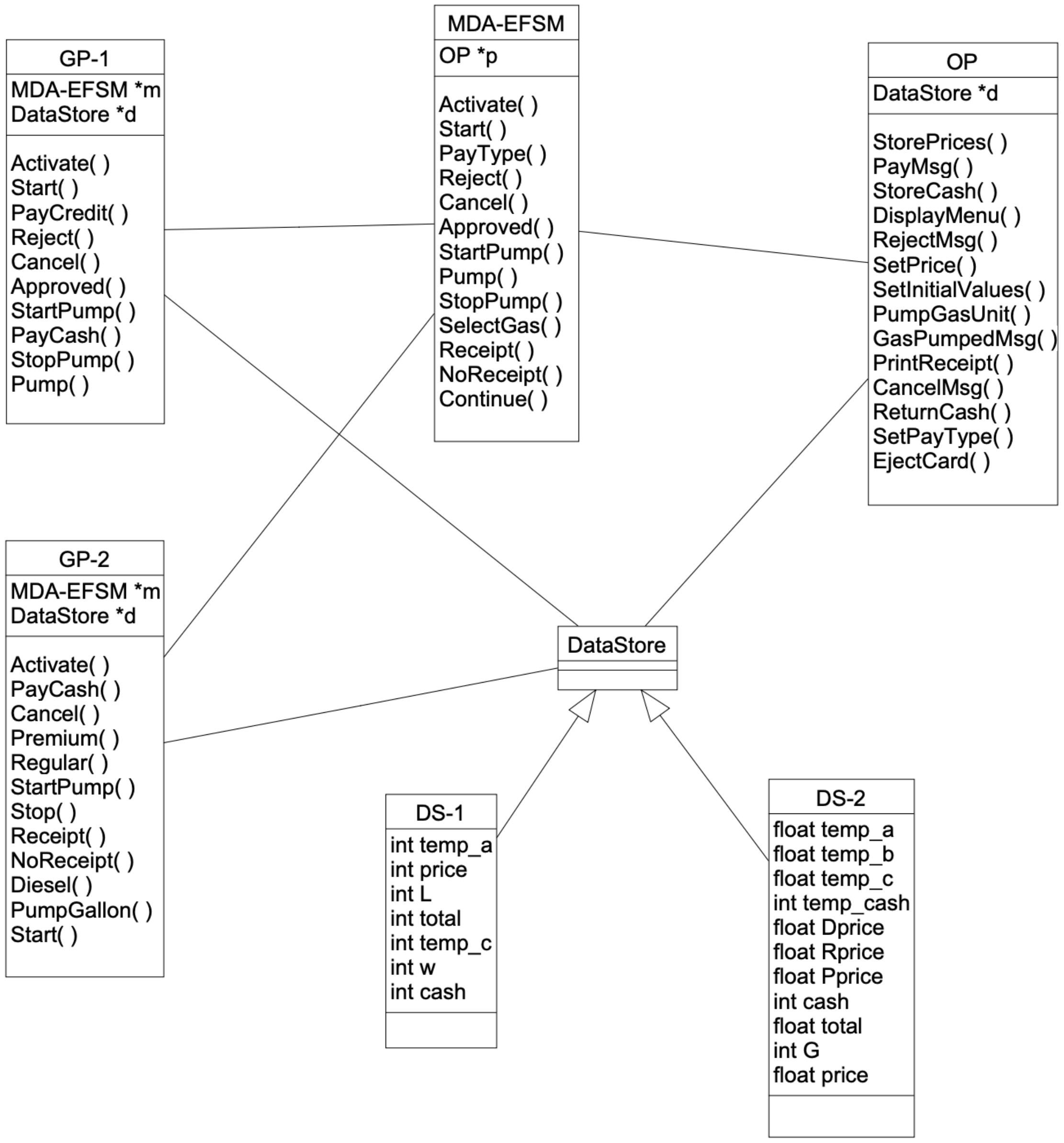
Project Part 2

Project Part 2.....1
1. MDA-EFSM model for the GP components (I used the sample)..... 3
2. Class diagram of the MDA of the GP components.....6
3. For each class in the class diagram:..... 7
OP Class:.....7
GP1 Class:.....7
GP2 Class:.....7
Classes about State Pattern:.....8
MDAEFSM Class:.....8
State Class:.....8
Start Class:..... 9
S0 Class:.....9
S1 Class:.....9
S2 Class:.....9
S3 Class:.....9
S4 Class:.....9
S5 Class:.....9
S6 Class:.....10
Classes about Strategy Pattern:..... 10
StorePrices Class:..... 10
StorePrices1 Class:..... 10
StorePrices2 Class:..... 10
PayMsg Class:.....10
PayMsg1 Class:.....10
PayMsg2 Class:.....10
StoreCash Class:..... 11
StoreCash1 Class:..... 11
StoreCash2 Class:..... 11
DisplayMenu Class:..... 11
DisplayMenu1 Class:..... 11
DisplayMenu2 Class:..... 11
RejectMsg Class:..... 11
RejectMsg1 Class:..... 11

RejectMsg2 Class:........ 12
SetPrice Class:........ 12
SetPrice1 Class:........ 12
SetPrice2 Class:........ 12
SetInitValues Class:........ 12
SetInitValues1 Class:........ 12
SetInitValues2 Class:........ 12
PumpGasUnit Class:........12
PumpGasUnit1 Class:........12
PumpGasUnit2 Class:........13
GasPumpedMsg Class:........ 13
GasPumpedMsg1 Class:........ 13
GasPumpedMsg2 Class:........ 13
PrintReceipt Class:........ 13
PrintReceipt1 Class:........ 13
PrintReceipt2 Class:........ 13
CancelMsg Class:........ 14
CancelMsg1 Class:........ 14
CancelMsg2 Class:........ 14
ReturnCash Class:........14
ReturnCash1 Class:........14
ReturnCash2 Class:........14
SetPayType Class:........14
SetPayType1 Class:........14
SetPayType2 Class:........15
EjectCard Class:........ 15
EjectCard1 Class:........ 15
EjectCard2 Class:........ 15
Classes about Abstract Factory Pattern:........ 15
AbstractFactory Class:........15
CF1 Class:........ 16
CF2 Class:........ 16
4. Dynamics. Provide two sequence diagrams for two Scenarios:........ 17
a. Scenario-I:........ 17
b. Scenario-II:........ 18

1. MDA-EFSM model for the GP components *(I used the sample)*



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

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

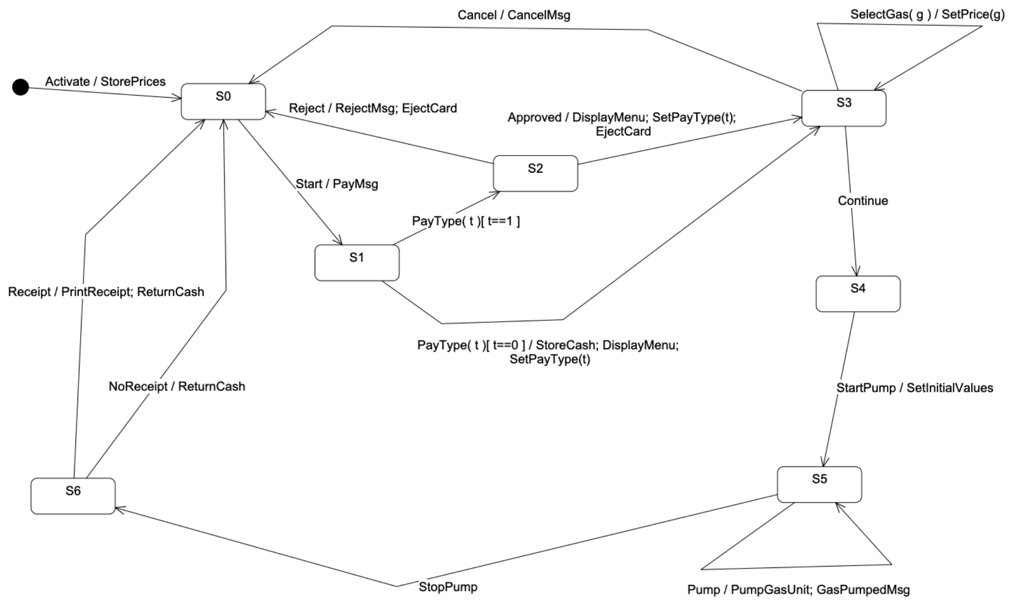
MDA-EFSM Events:

Activate()
Start()
PayType(int t) //credit: t=1; cash: t=0;
Reject()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int g) // Regular: g=1; Diesel: g=2; Premium: g=3
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

c. A state diagram of the MDA-EFSM



Cancel / CancelMsg
SelectGas(g) / SetPrice(g)
Activate / StorePrices
S0
Reject / RejectMsg; EjectCard
Approved / DisplayMenu; SetPayType(t); EjectCard
S3
S2
Start / PayMsg
Continue
PayType(t)[t==1]
S1
S4
Receipt / PrintReceipt; ReturnCash
PayType(t)[t==0] / StoreCash; DisplayMenu; SetPayType(t)
NoReceipt / ReturnCash
StartPump / SetInitialValues
S6
S5
StopPump
Pump / PumpGasUnit; GasPumpedMsg

d. Pseudo-code of all operations of Gas Pump: GP-1 andGP-2

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(0)
        }
}

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)

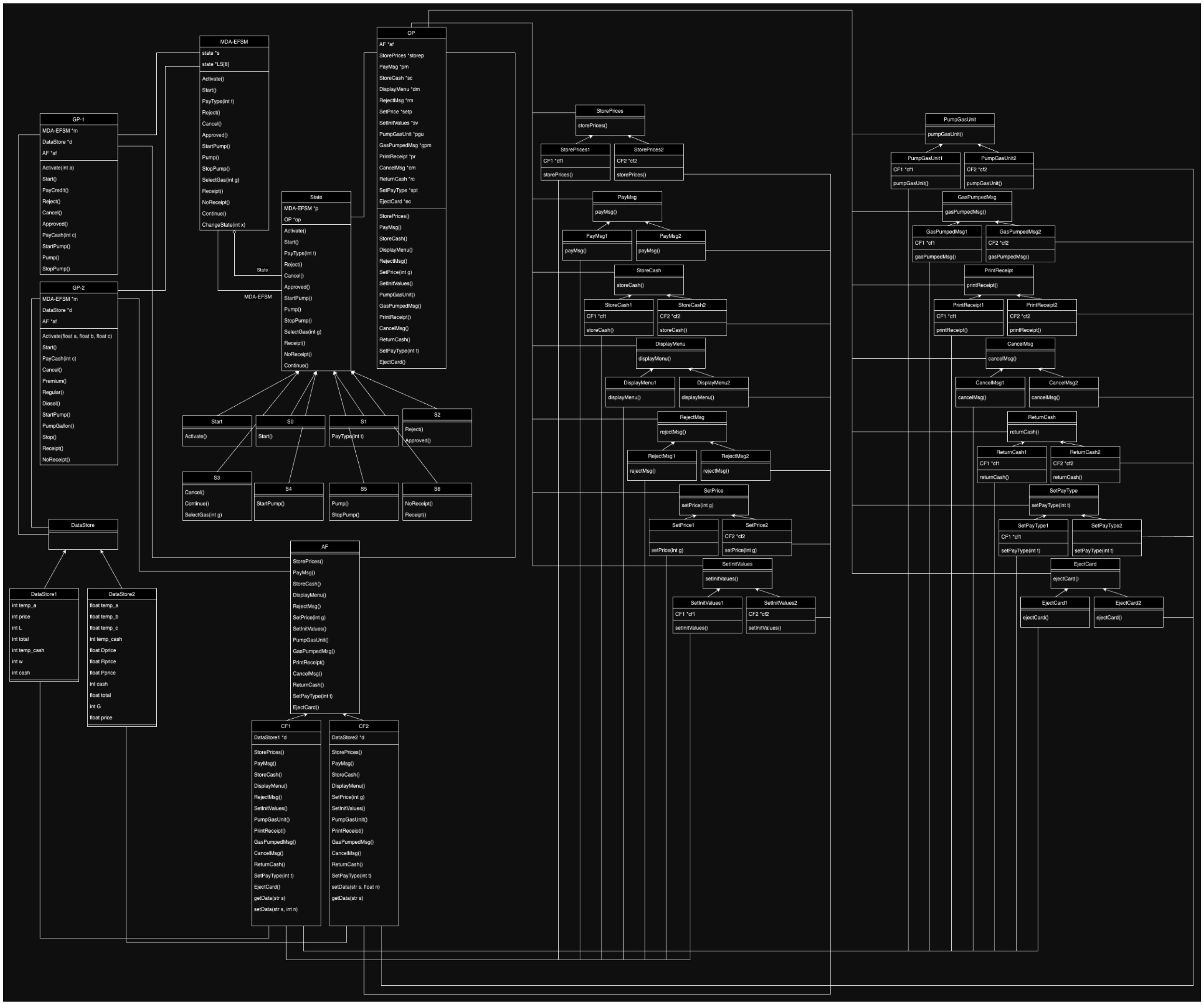
```
Activate(float a, float b, float c) {
        if ((a>0)&&(b>0)&&(c>0)) {
             d->temp_a=a;
             d->temp_b=b;
             d->temp_c=c
             m->Activate()
        }
}

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

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

2. Class diagram of the MDA of the GP components

Here is a link to this diagram.



3. For each class in the class diagram:

- **a. Describe the purpose of the class**
- **b. Describe the responsibility of each operation supported by each class**

OP Class:

Purpose: Represents the Output Processor responsible for processing outputs in the Gas Pump.
Responsibilities:
OP::StorePrices(): Initiates the store prices process.
OP::PayMsg(): Initiates the display payment message process.
OP::StoreCash(): Initiates the store cash process.
OP::DisplayMenu(): Initiates the display menu process.
OP::RejectMsg(): Initiates the display rejection message process.
OP::SetPrice(int g): Initiates the set price process.
OP::SetInitValues(): Initiates the set initial values process.
OP::PumpGasUnit(): Initiates the pump gas unit process.
OP::GasPumpedMsg(): Initiates the display gas pumped message process.
OP::PrintReceipt(): Initiates the print receipt process.
OP::CancelMsg(): Initiates the display cancellation message process.
OP::ReturnCash(): Initiates the return cash process.
OP::SetPayType(int t): Initiates the set payment type process.
OP::EjectCard(): Initiates the eject card process.

GP1 Class:

Purpose: Represents Gas Pump 1.
Responsibilities:
GP1::Activate(int a): Activates Gas Pump 1.
GP1::Start(): Starts Gas Pump 1.
GP1::PayCredit(): Handles credit payment in Gas Pump 1.
GP1::Reject(): Handles rejection in Gas Pump 1.
GP1::Cancel(): Cancels the transaction in Gas Pump 1.
GP1::Approved(): Handles approval in Gas Pump 1.
GP1::PayCash(int c): Handles cash payment in Gas Pump 1.
GP1::StartPump(): Starts pumping in Gas Pump 1.
GP1::Pump(): Performs pumping operation in Gas Pump 1.
GP1::StopPump(): Stops pumping in Gas Pump 1.

GP2 Class:

Purpose: Represents Gas Pump 2.
Responsibilities:
GP2::Activate(float a, float b, float c): Activates Gas Pump 2.
GP2::Start(): Starts Gas Pump 2.
GP2::PayCash(int c): Handles cash payment in Gas Pump 2.
GP2::Cancel(): Cancels the transaction in Gas Pump 2.

GP2::Premium(): Handles selection of premium gas in Gas Pump 2.
GP2::Regular(): Handles selection of regular gas in Gas Pump 2.
GP2::Diesel(): Handles selection of diesel gas in Gas Pump 2.
GP2::StartPump(): Starts pumping in Gas Pump 2.
GP2::PumpGallon(): Performs pumping operation in Gas Pump 2.
GP2::Stop(): Stops pumping in Gas Pump 2.
GP2::Receipt(): Generates receipt in Gas Pump 2.
GP2::NoReceipt(): Handles scenario when no receipt is needed in Gas Pump 2.

Classes about State Pattern:

Purpose: Represents a set of classes implementing the state pattern, which is used for managing state transitions and operations in the Gas Pump system.

MDAEFSM Class:
Purpose: Represents the MDAEFSM responsible for managing the state transitions and operations of a Gas Pump.
Responsibilities:
MDAEFSM::Activate(): Activates the current state.
MDAEFSM::Start(): Initiates the start operation.
MDAEFSM::PayType(int t): Handles the selection of payment type.
MDAEFSM::Reject(): Handles the rejection of credit card payment.
MDAEFSM::Cancel(): Cancels the transaction.
MDAEFSM::Approved(): Handles the approval of credit card payment.
MDAEFSM::StartPump(): Initiates the start pump operation.
MDAEFSM::Pump(): Performs pumping operation.
MDAEFSM::StopPump(): Stops the pumping operation.
MDAEFSM::SelectGas(int g): Selects the type of gas.
MDAEFSM::Receipt(): Generates a receipt.
MDAEFSM::NoReceipt(): Handles the scenario when no receipt is needed.
MDAEFSM::Continue(): Continues the operation.
MDAEFSM::ChangeState(int x): Changes the current state based on the index provided.

State Class:
Purpose: Represents the abstract base class for various states in the Gas Pump's state machine.
Responsibilities:
State::Activate(): Virtual function for activating the state.
State::Start(): Virtual function for initiating the start operation.
State::PayType(int t): Virtual function for handling payment type selection.
State::Reject(): Virtual function for handling credit card payment rejection.
State::Cancel(): Virtual function for canceling the transaction.
State::Approved(): Virtual function for handling credit card payment approval.
State::StartPump(): Virtual function for initiating the start pump operation.
State::Pump(): Virtual function for performing pumping operation.

State::StopPump(): Virtual function for stopping the pumping operation.
State::SelectGas(int g): Virtual function for selecting the type of gas.
State::Receipt(): Virtual function for generating a receipt.
State::NoReceipt(): Virtual function for handling scenarios when no receipt is needed.
State::Continue(): Virtual function for continuing the operation.

Start Class:
Purpose: Represents the Start state of the Gas Pump's state machine.
Responsibilities:
Start::Activate(): Overrides the Activate method to initiate the activation of the Gas Pump.

S0 Class:
Purpose: Represents the S0 state of the Gas Pump's state machine.
Responsibilities:
S0::Start(): Overrides the Start method to initiate the start operation.

S1 Class:
Purpose: Represents the S1 state of the Gas Pump's state machine.
Responsibilities:
S1::PayType(int t): Overrides the PayType method to handle payment type selection.

S2 Class:
Purpose: Represents the S2 state of the Gas Pump's state machine.
Responsibilities:
S2::Reject(): Overrides the Reject method to handle credit card payment rejection.
S2::Approved(): Overrides the Approved method to handle credit card payment approval.

S3 Class:
Purpose: Represents the S3 state of the Gas Pump's state machine.
Responsibilities:
S3::Cancel(): Overrides the Cancel method to cancel the transaction.
S3::Continue(): Overrides the Continue method to continue the operation.
S3::SelectGas(int g): Overrides the SelectGas method to select the type of gas.

S4 Class:
Purpose: Represents the S4 state of the Gas Pump's state machine.
Responsibilities:
S4::StartPump(): Overrides the StartPump method to initiate the start pump operation.

S5 Class:
Purpose: Represents the S5 state of the Gas Pump's state machine.
Responsibilities:
S5::Pump(): Overrides the Pump method to perform pumping operation.

S5::StopPump(): Overrides the StopPump method to stop the pumping operation.

S6 Class:
Purpose: Represents the S6 state of the Gas Pump's state machine.
Responsibilities:
S6::NoReceipt(): Overrides the NoReceipt method to handle scenarios when no receipt is needed.
S6::Receipt(): Overrides the Receipt method to generate a receipt.

Classes about Strategy Pattern:

Purpose: Represents a set of classes implementing the Strategy design pattern.

StorePrices Class:
Purpose: Represents the abstract base class for storing prices in the Gas Pump.
Responsibilities:
StorePrices::storePrices(): Virtual function for storing prices.

StorePrices1 Class:
Purpose: Represents the implementation of storing prices for Gas Pump 1.
Responsibilities:
StorePrices1::storePrices(): Implements the storage of prices for Gas Pump 1.

StorePrices2 Class:
Purpose: Represents the implementation of storing prices for Gas Pump 2.
Responsibilities:
StorePrices2::storePrices(): Implements the storage of prices for Gas Pump 2.

PayMsg Class:
Purpose: Represents the abstract base class for displaying payment messages.
Responsibilities:
PayMsg::payMsg(): Virtual function for displaying payment messages.

PayMsg1 Class:
Purpose: Represents the implementation of displaying payment messages for Gas Pump 1.
Responsibilities:
PayMsg1::payMsg(): Implements the display of payment messages for Gas Pump 1.

PayMsg2 Class:
Purpose: Represents the implementation of displaying payment messages for Gas Pump 2.
Responsibilities:
PayMsg2::payMsg(): Implements the display of payment messages for Gas Pump 2.

StoreCash Class:
Purpose: Represents the abstract base class for storing cash in the Gas Pump.
Responsibilities:
StoreCash::storeCash(): Virtual function for storing cash.

StoreCash1 Class:
Purpose: Represents the implementation of storing cash for Gas Pump 1.
Responsibilities:
StoreCash1::storeCash(): Implements the storage of cash for Gas Pump 1.

StoreCash2 Class:
Purpose: Represents the implementation of storing cash for Gas Pump 2.
Responsibilities:
StoreCash2::storeCash(): Implements the storage of cash for Gas Pump 2.

DisplayMenu Class:
Purpose: Represents the abstract base class for displaying menus in the Gas Pump.
Responsibilities:
DisplayMenu::displayMenu(): Virtual function for displaying menus.

DisplayMenu1 Class:
Purpose: Represents the implementation of displaying menus for Gas Pump 1.
Responsibilities:
DisplayMenu1::displayMenu(): Implements the display of menus for Gas Pump 1.

DisplayMenu2 Class:
Purpose: Represents the implementation of displaying menus for Gas Pump 2.
Responsibilities:
DisplayMenu2::displayMenu(): Implements the display of menus for Gas Pump 2.

RejectMsg Class:
Purpose: Represents the abstract base class for displaying rejection messages.
Responsibilities:
RejectMsg::rejectMsg(): Virtual function for displaying rejection messages.

RejectMsg1 Class:
Purpose: Represents the implementation of displaying rejection messages for Gas Pump 1.
Responsibilities:
RejectMsg1::rejectMsg(): Implements the display of rejection messages for Gas Pump 1.

RejectMsg2 Class:
Purpose: Represents the implementation of displaying rejection messages for Gas Pump 2.

Responsibilities:
RejectMsg2::rejectMsg(): Implements the display of rejection messages for Gas Pump 2 (not used).

SetPrice Class:
Purpose: Represents the abstract base class for setting prices in the Gas Pump.
Responsibilities:
SetPrice::setPrice(int g): Virtual function for setting prices.

SetPrice1 Class:
Purpose: Represents the implementation of setting prices for Gas Pump 1.
Responsibilities:
SetPrice1::setPrice(int g): Implements the setting of prices for Gas Pump 1 (not used).

SetPrice2 Class:
Purpose: Represents the implementation of setting prices for Gas Pump 2.
Responsibilities:
SetPrice2::setPrice(int g): Implements the setting of prices for Gas Pump 2.

SetInitValues Class:
Purpose: Represents the abstract base class for setting initial values in the Gas Pump.
Responsibilities:
SetInitValues::setInitValues(): Virtual function for setting initial values.

SetInitValues1 Class:
Purpose: Represents the implementation of setting initial values for Gas Pump 1.
Responsibilities:
SetInitValues1::setInitValues(): Implements the setting of initial values for Gas Pump 1.

SetInitValues2 Class:
Purpose: Represents the implementation of setting initial values for Gas Pump 2.
Responsibilities:
SetInitValues2::setInitValues(): Implements the setting of initial values for Gas Pump 2.

PumpGasUnit Class:
Purpose: Represents the abstract base class for pumping gas units in the Gas Pump.
Responsibilities:
PumpGasUnit::pumpGasUnit(): Virtual function for pumping gas units.

PumpGasUnit1 Class:
Purpose: Represents the implementation of pumping gas units for Gas Pump 1.
Responsibilities:
PumpGasUnit1::pumpGasUnit(): Implements the pumping of gas units for Gas Pump 1.

PumpGasUnit2 Class:
Purpose: Represents the implementation of pumping gas units for Gas Pump 2.
Responsibilities:
PumpGasUnit2::pumpGasUnit(): Implements the pumping of gas units for Gas Pump 2.

GasPumpedMsg Class:
Purpose: Represents the abstract base class for displaying gas pumped messages.
Responsibilities:
GasPumpedMsg::gasPumpedMsg(): Virtual function for displaying gas pumped messages.

GasPumpedMsg1 Class:
Purpose: Represents the implementation of displaying gas pumped messages for Gas Pump 1.
Responsibilities:
GasPumpedMsg1::gasPumpedMsg(): Implements the display of gas pumped messages for Gas Pump 1.

GasPumpedMsg2 Class:
Purpose: Represents the implementation of displaying gas pumped messages for Gas Pump 2.
Responsibilities:
GasPumpedMsg2::gasPumpedMsg(): Implements the display of gas pumped messages for Gas Pump 2.

PrintReceipt Class:
Purpose: Represents the abstract base class for printing receipts in the Gas Pump.
Responsibilities:
PrintReceipt::printReceipt(): Virtual function for printing receipts.

PrintReceipt1 Class:
Purpose: Represents the implementation of printing receipts for Gas Pump 1.
Responsibilities:
PrintReceipt1::printReceipt(): Implements the printing of receipts for Gas Pump 1.

PrintReceipt2 Class:
Purpose: Represents the implementation of printing receipts for Gas Pump 2.
Responsibilities:
PrintReceipt2::printReceipt(): Implements the printing of receipts for Gas Pump 2.

CancelMsg Class:
Purpose: Represents the abstract base class for displaying cancellation messages.
Responsibilities:
CancelMsg::cancelMsg(): Virtual function for displaying cancellation messages.

CancelMsg1 Class:
Purpose: Represents the implementation of displaying cancellation messages for Gas Pump 1.
Responsibilities:
CancelMsg1::cancelMsg(): Implements the display of cancellation messages for Gas Pump 1.

CancelMsg2 Class:
Purpose: Represents the implementation of displaying cancellation messages for Gas Pump 2.
Responsibilities:
CancelMsg2::cancelMsg(): Implements the display of cancellation messages for Gas Pump 2.

ReturnCash Class:
Purpose: Represents the abstract base class for returning cash in the Gas Pump.
Responsibilities:
ReturnCash::returnCash(): Virtual function for returning cash.

ReturnCash1 Class:
Purpose: Represents the implementation of returning cash for Gas Pump 1.
Responsibilities:
ReturnCash1::returnCash(): Implements the returning of cash for Gas Pump 1.

ReturnCash2 Class:
Purpose: Represents the implementation of returning cash for Gas Pump 2.
Responsibilities:
ReturnCash2::returnCash(): Implements the returning of cash for Gas Pump 2.

SetPayType Class:
Purpose: Represents the abstract base class for setting payment types in the Gas Pump.
Responsibilities:
SetPayType::setPayType(int t): Virtual function for setting payment types.

SetPayType1 Class:
Purpose: Represents the implementation of setting payment types for Gas Pump 1.
Responsibilities:
SetPayType1::setPayType(int t): Implements the setting of payment types for Gas Pump 1.

SetPayType2 Class:
Purpose: Represents the implementation of setting payment types for Gas Pump 2.

Responsibilities:
SetPayType2::setPayType(int t): Implements the setting of payment types for Gas Pump 2.

EjectCard Class:
Purpose: Represents the abstract base class for ejecting cards in the Gas Pump.
Responsibilities:
EjectCard::ejectCard(): Virtual function for ejecting cards.

EjectCard1 Class:
Purpose: Represents the implementation of ejecting cards for Gas Pump 1.
Responsibilities:
EjectCard1::ejectCard(): Implements the ejecting of cards for Gas Pump 1.

EjectCard2 Class:
Purpose: Represents the implementation of ejecting cards for Gas Pump 2.
Responsibilities:
EjectCard2::ejectCard(): Implements the ejecting of cards for Gas Pump 2 (not used).

Classes about Abstract Factory Pattern:

Purpose: Represents the abstract base class for the Abstract Factory pattern, which provides an interface for creating families of related or dependent objects without specifying their concrete classes.

AbstractFactory Class:
Purpose: Represents the abstract base class for creating families of related objects (concrete factories).
Responsibilities:
AbstractFactory::StorePrices(): Virtual function for creating StorePrices objects.
AbstractFactory::PayMsg(): Virtual function for creating PayMsg objects.
AbstractFactory::StoreCash(): Virtual function for creating StoreCash objects.
AbstractFactory::DisplayMenu(): Virtual function for creating DisplayMenu objects.
AbstractFactory::RejectMsg(): Virtual function for creating RejectMsg objects.
AbstractFactory::SetPrice(int g): Virtual function for creating SetPrice objects.
AbstractFactory::SetInitValues(): Virtual function for creating SetInitValues objects.
AbstractFactory::PumpGasUnit(): Virtual function for creating PumpGasUnit objects.
AbstractFactory::GasPumpedMsg(): Virtual function for creating GasPumpedMsg objects.
AbstractFactory::PrintReceipt(): Virtual function for creating PrintReceipt objects.
AbstractFactory::CancelMsg(): Virtual function for creating CancelMsg objects.
AbstractFactory::ReturnCash(): Virtual function for creating ReturnCash objects.
AbstractFactory::SetPayType(int t): Virtual function for creating SetPayType objects.
AbstractFactory::EjectCard(): Virtual function for creating EjectCard objects.

<u>CF1 Class:</u>
Purpose: Represents a concrete factory responsible for creating algorithm objects specific to Gas Pump 1.
Responsibilities:
CF1::StorePrices(): Overrides the StorePrices method to create StorePrices1 objects.
CF1::PayMsg(): Overrides the PayMsg method to create PayMsg1 objects.
CF1::StoreCash(): Overrides the StoreCash method to create StoreCash1 objects.
CF1::DisplayMenu(): Overrides the DisplayMenu to create DisplayMenu1 objects.
CF1::RejectMsg(): Overrides the RejectMsg method to create RejectMsg1 objects.
CF1::SetPrice(int g): Overrides the SetPrice method to create SetPrice1 objects.
CF1::SetInitValues(): Overrides the SetInitValues to create SetInitValues1 objects.
CF1::PumpGasUnit(): Overrides the PumpGasUnit to create PumpGasUnit1 objects.
CF1::GasPumpedMsg(): Overrides the GasPumpedMsg to create GasPumpedMsg1.
CF1::PrintReceipt(): Overrides the PrintReceipt method to create PrintReceipt1 objects.
CF1::CancelMsg(): Overrides the CancelMsg method to create CancelMsg1 objects.
CF1::ReturnCash(): Overrides the ReturnCash method to create ReturnCash1 objects.
CF1::SetPayType(int t): Overrides the SetPayType to create SetPayType1 objects.
CF1::EjectCard(): Overrides the EjectCard method to create EjectCard1 objects.

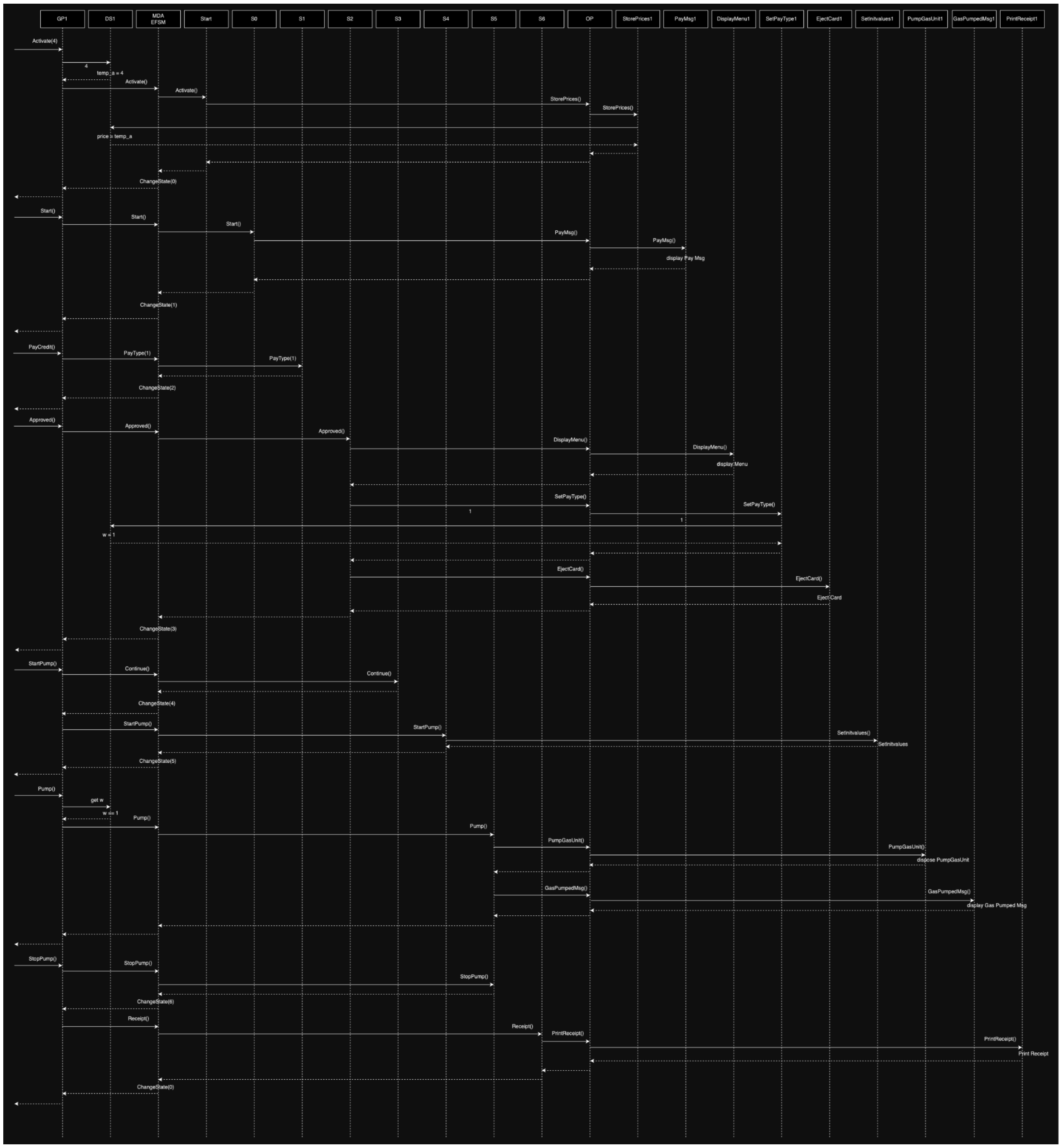
<u>CF2 Class:</u>
Purpose: Represents a concrete factory responsible for creating algorithm objects specific to Gas Pump 2.
Responsibilities:
CF2::StorePrices(): Overrides the StorePrices method to create StorePrices2 objects.
CF2::PayMsg(): Overrides the PayMsg method to create PayMsg2 objects.
CF2::StoreCash(): Overrides the StoreCash method to create StoreCash2 objects.
CF2::DisplayMenu(): Overrides the DisplayMenu to create DisplayMenu2 objects.
CF2::RejectMsg(): Overrides the RejectMsg method to create RejectMsg2 objects.
CF2::SetPrice(int g): Overrides the SetPrice method to create SetPrice2 objects.
CF2::SetInitValues(): Overrides the SetInitValues to create SetInitValues2 objects.
CF2::PumpGasUnit(): Overrides the PumpGasUnit to create PumpGasUnit2 objects.
CF2::GasPumpedMsg(): Overrides the GasPumpedMsg to create GasPumpedMsg2.
CF2::PrintReceipt(): Overrides the PrintReceipt method to create PrintReceipt2 objects.
CF2::CancelMsg(): Overrides the CancelMsg method to create CancelMsg2 objects.
CF2::ReturnCash(): Overrides the ReturnCash method to create ReturnCash2 objects.
CF2::SetPayType(int t): Overrides the SetPayType to create SetPayType2 objects.
CF2::EjectCard(): Overrides the EjectCard method to create EjectCard2 objects.

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

a. ***Scenario-I:***

should show how one liter of gas is disposed in the Gas Pump GP- 1 component: Activate(4), Start(), PayCredit(), Approved(), StartPump(), Pump(), StopPump()

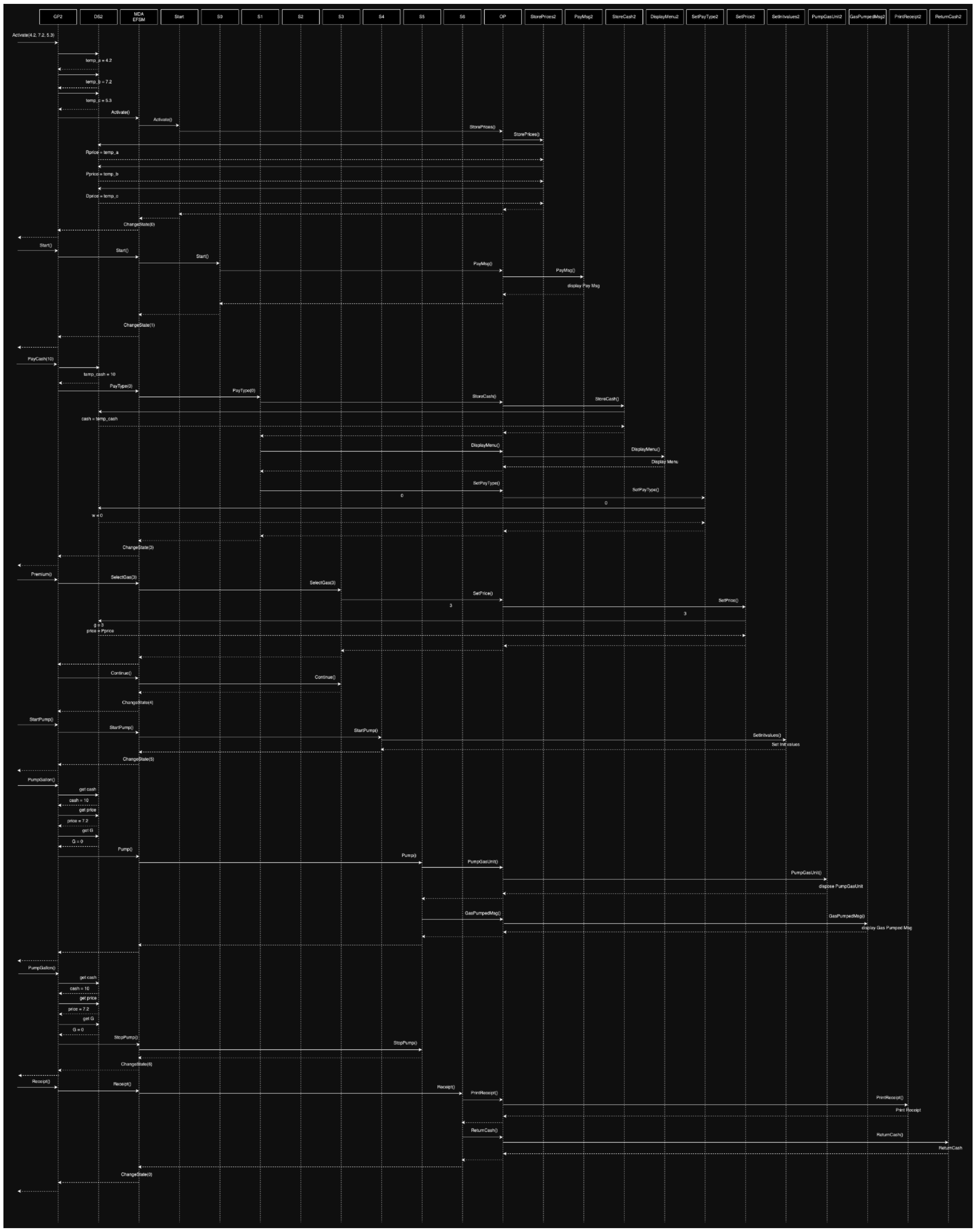
Here is a link to this diagram.



b. *Scenario-II:*

should show how one gallon of Premium gas is disposed in the Gas Pump GP-2 component:
Activate(4.2, 7.2, 5.3), Start(), PayCash(10), Premium(), StartPump(), PumpGallon(), PumpGallon(), Receipt()

Here is a link to this diagram.


GP2
DS2
MDA EFSM
Start
S0
S1
S2
S3
S4
S5
S6
OP
StorePrices2
PayMsg2
StoreCash2
DisplayMenu2
SetPayType2
SetPrice2
SetInitvalues2
PumpGasUnit2
GasPumpedMsg2
PrintReceipt2
ReturnCash2
Activate(4.2, 7.2, 5.3)
Activate()
StorePrices()
ChangeState(0)
Start()
PayMsg()
display Pay Msg
ChangeState(1)
PayCash(10)
temp_cash = 10
PayType(0)
StoreCash()
cash = temp_cash
DisplayMenu()
Display Menu
SetPayType()
ChangeState(3)
Premium()
SelectGas(3)
SetPrice()
g = 3
price = Pprice
Continue()
ChangeState(4)
StartPump()
SetInitvalues()
Set Init values
ChangeState(5)
PumpGallon()
get cash
cash = 10
get price
price = 7.2
get G
G = 0
Pump()
PumpGasUnit()
dispose PumpGasUnit
GasPumpedMsg()
display Gas Pumped Msg
StopPump()
ChangeState(6)
Receipt()
PrintReceipt()
Print Receipt
ReturnCash()
ChangeState(0)