MDA-EFSM Events:

Activate()
Start()
PayType(int t) //credit: t=1; cash: t=2; debit: t=3
Reject()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int g) // Regular: g=1; Super: g=2; Premium: g=3; Diesel: g=4
Receipt()
NoReceipt()
CorrectPin()
IncorrectPin()
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
ReadyMsg	// displays the ready for pumping message
SetInitialValues	// set G (or L) and total to 0;
PumpGasUnit	// disposes unit of gas and counts # of units disposed
GasPumpedMsg	// displays the amount of disposed gas
StopMsg	// stop pump message and receipt? msg (optionally)
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 and cash to 0

Operations of the Input Processor (GasPump-1)

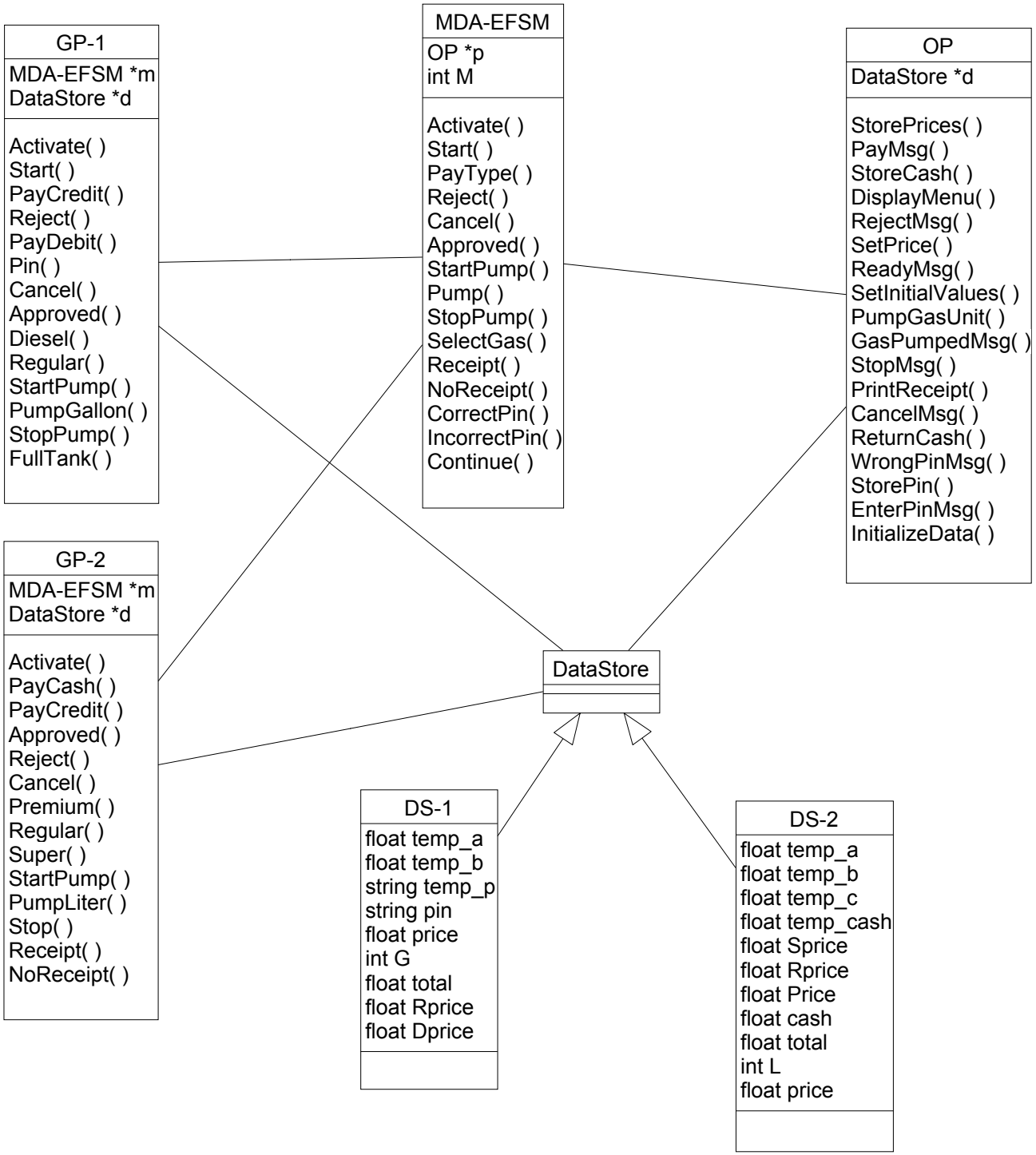
```
public void Activate(float a, float b) {
	if ((a > 0) && (b > 0)) {
		DS.setTemp_a(a);
		DS.setTemp_b(b);
		MD.Activate();
	}
}

public void Start() {
	MD.Start();
}

public void PayCredit() {
	MD.PayType(1);
}

public void Reject() {
	MD.Reject();
}

public void PayDebit(String p) {
	DS.setTemp_p(p);
	MD.PayType(3);
}

public void Pin(String x) {
	if (DS.getPin().equals(x))
		MD.CorrectPin();
	else
		MD.IncorrectPin();
}

public void Cancel() {
	MD.Cancel();
}

public void Approved() {
	MD.Approved();
}

public void Diesel() {
	MD.SelectGas(4);
}
```

```
public void Regular() {
	MD.SelectGas(1);
}

public void StartPump() {
	if (DS.getPrice() > 0) {
		MD.Continue();
		MD.StartPump();
	}
}

public void PumpGallon() {
	MD.Pump();
}

public void StopPump() {
	MD.StopPump();
	MD.Receipt();
}

public void FullTank() {
	MD.StopPump();
	MD.Receipt();
}
```

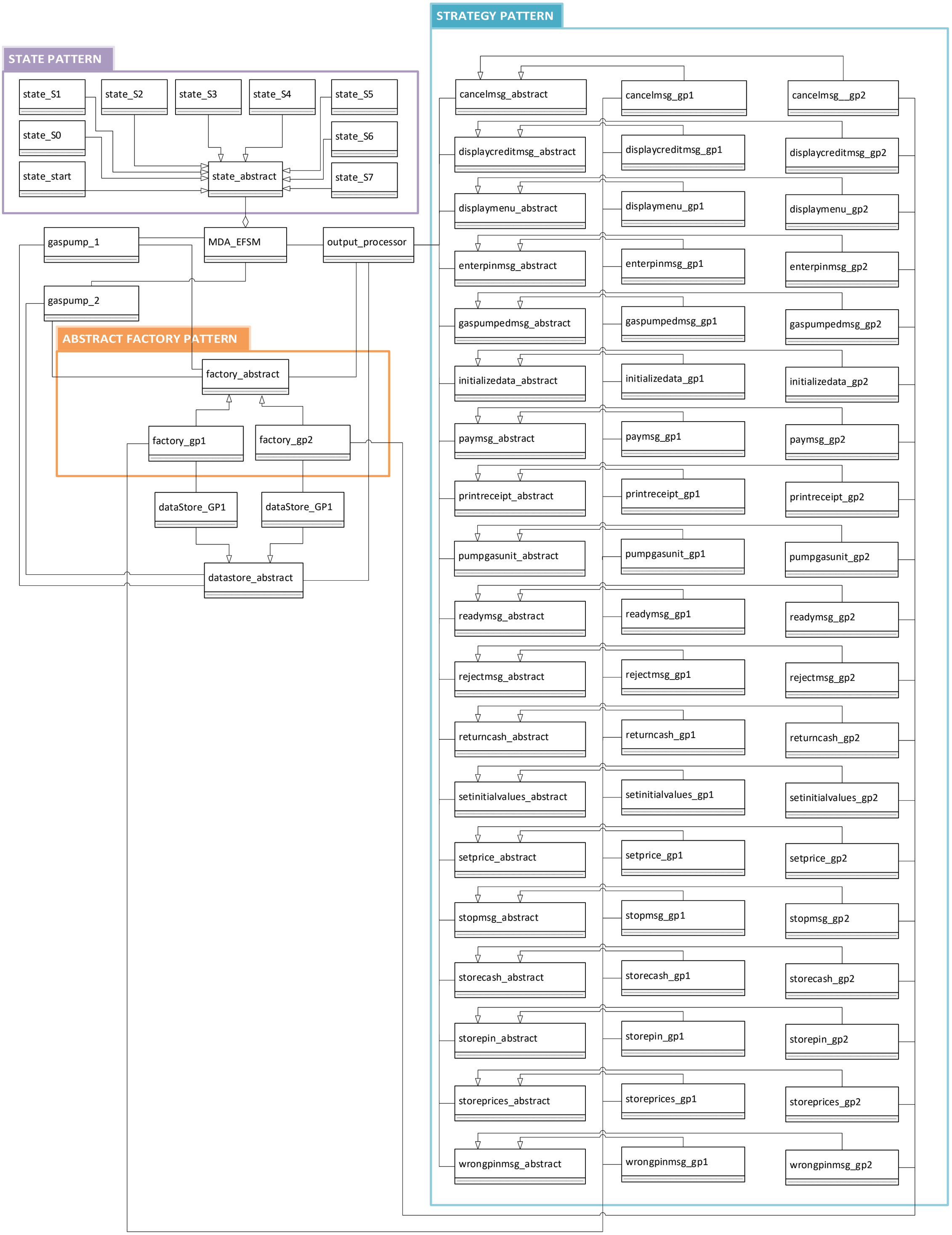
Operations of the Input Processor (GasPump-2)

```
public void Activate(float a, float b, float c) {
	if ((a > 0) && (b > 0) && (c > 0)) {
		DS.setTemp_a(a);
		DS.setTemp_b(b);
		DS.setTemp_c(c);
		MD.Activate();
		MD.Start();
	}
}

public void PayCash(float c) {
	if (c > 0) {
		DS.setTemp_cash(c);
		//MD.Start();
		MD.PayType(2);
	}
}

public void PayCredit() {
	//MD.Start();
	MD.PayType(1);
}

public void Reject() {
	MD.Reject();
}

public void Approved() {
	MD.Approved();
}

public void Cancel() {
	MD.Cancel();
}

public void Super() {
	MD.SelectGas(2);
	MD.Continue();
}

public void Premium() {
	MD.SelectGas(3);
	MD.Continue();
```

```
}

public void Regular() {
	MD.SelectGas(1);
	MD.Continue();
}

public void StartPump() {
	MD.StartPump();
}

public void PumpLiter() {
	if (DS.getCash() > 0 && (DS.getCash() < DS.getPrice() * (DS.getL() + 1))) // TODO MD.M == 0
		MD.StopPump();
	else
		MD.Pump();
}

public void Stop() {
	MD.StopPump();
}

public void Receipt() {
	MD.Receipt();
}

public void NoReceipt() {
	MD.NoReceipt();
}
```


GP-1
MDA-EFSM *m
DataStore *d
Activate( )
Start( )
PayCredit( )
Reject( )
PayDebit( )
Pin( )
Cancel( )
Approved( )
Diesel( )
Regular( )
StartPump( )
PumpGallon( )
StopPump( )
FullTank( )
MDA-EFSM
OP *p
int M
Activate( )
Start( )
PayType( )
Reject( )
Cancel( )
Approved( )
StartPump( )
Pump( )
StopPump( )
SelectGas( )
Receipt( )
NoReceipt( )
CorrectPin( )
IncorrectPin( )
Continue( )
OP
DataStore *d
StorePrices( )
PayMsg( )
StoreCash( )
DisplayMenu( )
RejectMsg( )
SetPrice( )
ReadyMsg( )
SetInitialValues( )
PumpGasUnit( )
GasPumpedMsg( )
StopMsg( )
PrintReceipt( )
CancelMsg( )
ReturnCash( )
WrongPinMsg( )
StorePin( )
EnterPinMsg( )
InitializeData( )
GP-2
MDA-EFSM *m
DataStore *d
Activate( )
PayCash( )
PayCredit( )
Approved( )
Reject( )
Cancel( )
Premium( )
Regular( )
Super( )
StartPump( )
PumpLiter( )
Stop( )
Receipt( )
NoReceipt( )
DataStore
DS-1
float temp_a
float temp_b
string temp_p
string pin
float price
int G
float total
float Rprice
float Dprice
DS-2
float temp_a
float temp_b
float temp_c
float temp_cash
float Sprice
float Rprice
float Price
float cash
float total
int L
float price

High level class diagram



Class description:

gaspump_1
factory_abstract AF; datastore_abstract DS; MDAEFSM MD;
void set_MDAEFSM(MDAEFSM md) void set_factory(factory_abstract abs_fact) void set_datestore() void Activate(float a, float b) void Start() void PayCredit() void Reject() void PayDebit(String p) void Pin(String x) void Cancel() void Approved() void Diesel() void Regular() void StartPump() void PumpGallon() void StopPump() void FullTank()

gaspump_1

factory_abstract AF - store pointer to abstract factory for GP1
datastore_abstract DS - store pointer to datastore for GP1
MDAEFSM MD - store pointer to MDA-EFSM

void set_MDAEFSM(MDAEFSM md) - sets MDAEFSM pointer to object created in driver
void set_factory(factory_abstract abs_fact) - set GP1 factory pointer to object created in driver
void set_datestore() - get datastore for GP1 from abstract factory
void Activate(float a, float b) – sets temp_a and temp_b values and invokes Activate in MDAEFSM
void Start() - invokes Start in MDAEFSM
void PayCredit() - invokes PayType in MDAEFSM with value 1 as argument
void Reject() – invokes Reject in MDAEFSM
void PayDebit(String p) - invokes PayType in MDAEFSM with value 3 as argument
void Pin(String x) – accepts pin x from user and compares with saved vlalue. If PIN matches invokes CorrectPin on MDAEFSM or else invokes IncorrectPin on MDAEFSM.
void Cancel() – Invokes Cancel on MDAEFSM
void Approved() – Invokes Approved on MDAEFSM
void Diesel() – Invokes SelectGas on MDAEFSM with value 4
void Regular() – Invokes SelectGas on MDAEFSM with value 1
void StartPump() – checks if price is more than 0, then invokes Continue and StartPump on MDAEFSM
void PumpGallon() – Invokes Pump on MDAEFSM
void StopPump() - Invokes StopPump and Receipt on MDAEFSM
void FullTank() - Invokes StopPump and Receipt on MDAEFSM

gaspump_2
factory_abstract AF; datastore_abstract DS; MDAEFSM MD;
void set_MDAEFSM(MDAEFSM md) void set_factory(factory_abstract abs_fact) void set_datestore() void Activate(float a, float b, float c) void PayCash(float c) void PayCredit() void Reject() void Cancel() void Approved() void Super() void Premium() void Regular() void Start() void PumpLiter() void Receipt() void NoReceipt()

gaspump_2

factory_abstract AF - store pointer to abstract factory for GP2
datastore_abstract DS - store pointer to datastore for GP2
MDAEFSM MD - store pointer to MDA-EFSM

void set_MDAEFSM(MDAEFSM md) - sets MDAEFSM pointer to object created in driver
void set_factory(factory_abstract abs_fact) - set GP2 factory pointer to object created in driver
void set_datestore() - get datastore for GP2 from abstract factory
void void Activate(float a, float b, float c) – sets temp_a , temp_b and temp_c values and invokes Activate followed by Start in MDAEFSM
void PayCash(float c) – if cash amount is greater than 0, set temp_cash and invoke PayType on MDAEFSM with value 2
void PayCredit() - invokes PayType in MDAEFSM with value 1 as argument
void Reject() – invokes Reject in MDAEFSM
void Cancel() – Invokes Cancel on MDAEFSM
void Approved() – Invokes Approved on MDAEFSM
void Super() – Invokes SelectGas on MDAEFSM with value 2
void Premium() – Invokes SelectGas on MDAEFSM with value 1
void Regular() – Invokes SelectGas on MDAEFSM with value 1
void StartPump() – Invokes StartPump on MDAEFSM
void PumpLiter() – checks if cash available to pump more gas, then invoke Pump on MDAEFSM else invoke Stop on MDAEFSM
void Stop() - Invokes StopPump and Receipt on MDAEFSM
void Receipt() - Invokes Receipt on MDAEFSM
void NoReceipt() - Invokes NoReceipt on MDAEFSM

datastore_abstract

abstract float getTemp_a();
abstract void setTemp_a(float temp_a);
abstract float getTemp_b();
abstract void setTemp_b(float temp_b);
abstract float getPrice();
abstract void setPrice(float price);
abstract float getTotal();
abstract void setTotal(float total);
abstract float getRprice();
abstract void setRprice(float rprice);
 	
abstract String getTemp_p();
abstract void setTemp_p(String temp_p);
abstract String getPin();
abstract void setPin(String pin);
abstract int getG();
abstract void setG(int g);
abstract float getDprice();
abstract void setDprice(float dprice);
 	
abstract float getTemp_c();
abstract void setTemp_c(float temp_c);
abstract float getTemp_cash();
abstract void setTemp_cash(float temp_cash);
abstract float getSprice();
abstract void setSprice(float sprice);
abstract float getPprice();
abstract void setPprice(float pprice);
abstract float getCash();
abstract void setCash(float cash);
abstract int getL();
abstract void setL(int l);

datastore_abstract

This is the data abstract class which the datastore for gas pump 1 and 2 inherit from. It contains abstract methods of both the gas pumps. The function are defined in datastore_gp1 and datastore_gp2.

datastore_gp1
float temp_a; float temp_b; String temp_p; String pin; float price; int G; float total; float Rprice; float Dprice;

float getTemp_a();
void setTemp_a(float temp_a);
float getTemp_b();
void setTemp_b(float temp_b);
float getPrice();
void setPrice(float price);
float getTotal();
void setTotal(float total);
float getRprice();
void setRprice(float rprice);
String getTemp_p();
void setTemp_p(String temp_p);
String getPin();
void setPin(String pin);
int getG();
void setG(int g);
float getDprice();
void setDprice(float dprice);

datastore_gp1

temp_a, temp_b, temp_p – temporary variables used by input processor
pin – store PIN for debit card payment
price – price of the gas selected
G – store quantity(gallons)
total – total amount of the gas filled
Rprice – regular gas price
Dprice – diesel gas price

This class contains getter, setter function for all variables of GP1 since the variables are private, setter methods are used to store the value in the variables getter methods are used to read the saved values.

datastore_gp1
float temp_a; float temp_b; float temp_c; float temp_cash; float Sprice; float Rprice; float PPrice; float cash; float total; int L; float price;

float getTemp_a();
void setTemp_a(float temp_a);
float getTemp_b();
void setTemp_b(float temp_b);
float getPrice();
void setPrice(float price);
float getTotal();
void setTotal(float total);
float getRprice();
void setRprice(float rprice);
String getTemp_p();
void setTemp_p(String temp_p);
String getPin();
void setPin(String pin);
int getG();
void setG(int g);
float getDprice();
void setDprice(float dprice);

datastore_gp2

temp_a, temp_b, temp_c, temp_cash – temporary variables used by input processor
cash – cash paid by user
price – price of the gas selected
L – store quantity(liter)
total – total amount of the gas filled
Rprice – regular gas price
Sprice – super gas price
Pprice – premium gas price

This class contains getter, setter function for all variables of GP2 since the variables are private, setter methods are used to store the value in the variables getter methods are used to read the saved values.

ABSTRACT FACTORY PATTERN

factory_abstract

abstract datastore_abstract getDataStore();
abstract cancelmsg_abstract getCancelMsg_obj();
abstract displaycreditmsg_abstract getDisplayCreditMsg_obj();
abstract displaymenu_abstract getDisplayMenu_obj();
abstract enterpinmsg_abstract getEnterPinMsg_obj();
abstract gaspumpedmsg_abstract getGasPumpedMsg_obj();
abstract initializedata_abstract getInitializeData_obj();
abstract paymsg_abstract getPayMsg_obj();
abstract printreceipt_abstract getPrintReceipt_obj();
abstract pumpgasunit_abstract getPumpGasUnit_obj();
abstract readymsg_abstract getReadyMsg_obj();
abstract rejectmsg_abstract getRejectMsg_obj();
abstract returncash_abstract getReturnCash_obj();
abstract setinitialvalues_abstract getSetInitialValues_obj();
abstract setprice_abstract getSetPrice_obj();
abstract stopmsg_abstract getStopMsg_obj();
abstract storecash_abstract getStoreCash_obj();
abstract storepin_abstract getStorePin_obj();
abstract storeprices_abstract getStorePrices_obj();
abstract wrongpinmsg_abstract getWrongPinMsg_obj();

factory_abstract

This class contains abstract getter methods for which the concrete methods are specified in the inherited classes, factory_gp1 and factory_gp2.

factory_gp1
datastore_gp1 DS; cancelmsg_gp1 CM; displaycreditmsg_gp1 DCCM; displaymenu_gp1 DM; enterpinmsg_gp1 EPM; gaspumpedmsg_gp1 GPM; initializedata_gp1 ID; paymsg_gp1 PM; printreceipt_gp1 PR; pumpgasunit_gp1 PGU; readymsg_gp1 RM; rejectmsg_gp1 RJM; returncash_gp1 RC; setinitialvalues_gp1 SIV; setprice_gp1 SP; stopmsg_gp1 SM; storecash_gp1 SC; storepin_gp1 SPN; storeprices_gp1 SPR; wrongpinmsg_gp1 WPM;

datastore_abstract getDataStore();
cancelmsg_abstract getCancelMsg_obj();
displaycreditmsg_abstract getDisplayCreditMsg_obj();
displaymenu_abstract getDisplayMenu_obj();
enterpinmsg_abstract getEnterPinMsg_obj();
gaspumpedmsg_abstract getGasPumpedMsg_obj();
initializedata_abstract getInitializeData_obj();
paymsg_abstract getPayMsg_obj();
printreceipt_abstract getPrintReceipt_obj();
pumpgasunit_abstract getPumpGasUnit_obj();
readymsg_abstract getReadyMsg_obj();
rejectmsg_abstract getRejectMsg_obj();
returncash_abstract getReturnCash_obj();
setinitialvalues_abstract getSetInitialValues_obj();
setprice_abstract getSetPrice_obj();
stopmsg_abstract getStopMsg_obj();
storecash_abstract getStoreCash_obj();
storepin_abstract getStorePin_obj();
storeprices_abstract getStorePrices_obj();
wrongpinmsg_abstract getWrongPinMsg_obj();

factory_gp1

DS – pointer to the datastore for GP1
CM, DCCM, DM, EPM, GPM, ID, PM, PR, PGU, RM, RJM, RC, SIV, SP, SM, SC, SPN, SPR, WPM – pointers to various action classes for GP1.

This class contains concrete getter methods that return the action class object for GP1 when requested for by the output processor. If no object exists, it creates a new one and saves the pointer in the factory and returns the pointer to that which is then used the next time the OP requests for the same object.

factory_gp2
datastore_gp2 DS; cancelmsg_gp2 CM; displaycreditmsg_gp2 DCCM; displaymenu_gp2 DM; enterpinmsg_gp2 EPM; gaspumpedmsg_gp2 GPM; initializedata_gp2 ID; paymsg_gp2 PM; printreceipt_gp2 PR; pumpgasunit_gp2 PGU; readymsg_gp2 RM; rejectmsg_gp2 RJM; returncash_gp2 RC; setinitialvalues_gp2 SIV; setprice_gp2 SP; stopmsg_gp2 SM; storecash_gp2 SC; storepin_gp2 SPN; storeprices_gp2 SPR; wrongpinmsg_gp2 WPM;

datastore_abstract getDataStore();
cancelmsg_abstract getCancelMsg_obj();
displaycreditmsg_abstract getDisplayCreditMsg_obj();
displaymenu_abstract getDisplayMenu_obj();
enterpinmsg_abstract getEnterPinMsg_obj();
gaspumpedmsg_abstract getGasPumpedMsg_obj();
initializedata_abstract getInitializeData_obj();
paymsg_abstract getPayMsg_obj();
printreceipt_abstract getPrintReceipt_obj();
pumpgasunit_abstract getPumpGasUnit_obj();
readymsg_abstract getReadyMsg_obj();
rejectmsg_abstract getRejectMsg_obj();
returncash_abstract getReturnCash_obj();
setinitialvalues_abstract getSetInitialValues_obj();
setprice_abstract getSetPrice_obj();
stopmsg_abstract getStopMsg_obj();
storecash_abstract getStoreCash_obj();
storepin_abstract getStorePin_obj();
storeprices_abstract getStorePrices_obj();
wrongpinmsg_abstract getWrongPinMsg_obj();

factory_gp2

DS – pointer to the datastore for GP2
CM, DCCM, DM, EPM, GPM, ID, PM, PR, PGU, RM, RJM, RC, SIV, SP, SM, SC, SPN, SPR, WPM – pointers to various action classes for GP2.

This class contains concrete getter methods that return the action class object for GP2 when requested for by the output processor. If no object exists, it creates a new one and saves the pointer in the factory and returns the pointer to that which is then used the next time the OP requests for the same object.

MDAEFSM
private int M; state_abstract ST; state_abstract state_list[];
void setStateList(state_abstract s_list[]) void changestate(int index) void Activate() void Start() void PayType(int t) void Reject() void Cancel() void Approved() void StartPump() void Pump() void StopPump() void SelectGas(int g) void Receipt() void NoReceipt() void CorrectPin() void IncorrectPin() void Continue()

MDA-EFSM

M – stores 0 if cash payment or else stores 1
ST – pointer to current state
State_list – stores the list of states

void setStateList(state_abstract s_list[]) – sets the list of states and set the pointer ST to the first state
void changestate(int index) – changes state to index value(decentralized)
void Activate() – invokes Activate in the current state
void Start() – sets value of M to 1 and invokes Start in the current state
void PayType(int t) – invokes PayType in the current state and sets M to 0 if payment method is cash
void Reject() – invokes Reject in the current state
void Cancel() – invokes Cancel in the current state
void Approved() – invokes Approved in the current state
void StartPump() – invokes StartPump in the current state
void Pump() – invokes Pump in the current state
void StopPump() – invokes StopPump in the current state
void SelectGas(int g) – invokes SelectGas in the current state
void Receipt() – invokes Receipt in the current state
void NoReceipt() – invokes NoReceipt in the current state
void CorrectPin() – invokes CorrectPin in the current state
void IncorrectPin() – invokes IncorrectPin in the current state
void Continue() – invokes Continue in the current state

output_processor
factory_abstract AF; datastore_abstract DS; cancelmsg_abstract CM; displaycreditmsg_abstract DCM; displaymenu_abstract DM; enterpinmsg_abstract EPM; gaspumpedmsg_abstract GPM; initializedata_abstract ID; paymsg_abstract PM; printreceipt_abstract PR; pumpgasunit_abstract PGU; readymsg_abstract RM; rejectmsg_abstract RJM; returncash_abstract RC; setinitialvalues_abstract SIV; setprice_abstract SP; stopmsg_abstract SM; storecash_abstract SC; storepin_abstract SPN; storeprices_abstract SPR; wrongpinmsg_abstract WPM;
void setFactory(factory_abstract abs_fact) void setDataStore(datastore_abstract ds) void StorePrices() void PayMsg() void StoreCash() void DisplayCreditMsg() void DisplayMenu() void RejectMsg() void SetPrice(int g) void ReadyMsg() void SetInitialValues(int M) void PumpGasUnit(int M) void GasPumpedMsg(int M) void StopMsg() void PrintReceipt() void CancelMsg() void ReturnCash() void WrongPinMsg() void StorePin() void EnterPinMsg() void InitializeData()

output_processor

AF – pointer to abstract factory 1 or 2 depending on the pump selection
DS – pointer to the datastore 1 or 2 depending on the pump selection
CM, DCCM, DM, EPM, GPM, ID, PM, PR, PGU, RM, RJM, RC, SIV, SP, SM, SC, SPN, SPR, WPM – pointers to various action classes for GP1 or GP2.

void setFactory(factory_abstract abs_fact) – set pointer to the factory object of GP1 or GP2 from the driver
void setDataStore(datastore_abstract ds) – set pointer to the datastore object of GP1 or GP2 from the driver

The methods get the pointers to the respective the action classes from the abstract factory and call the method to perform some specific action.

STATE PATTERN

state_abstract
output_processor OP; MDAEFSM MD;

abstract void Activate( );
abstract void Start( );
abstract void PayType(int t);
abstract void Reject( );
abstract void Cancel( );
abstract void Approved( );
abstract void StartPump(int M);
abstract void Pump(int M);
abstract void StopPump( );
abstract void SelectGas(int g);
abstract void Receipt( );
abstract void NoReceipt( );
abstract void CorrectPin( );
abstract void IncorrectPin( );
abstract void Continue( );

state_abstract

OP – pointer to the output processor which is set during the creation of the instance of the state
MD – pointer to MDAEFSM object (decentralized version)

The abstract state class contains abstract methods for which the concrete definition is specified in the respective state classes for which the action is valid. The invalid functions for any state is empty.

STATE LIST

state_start
void Activate( )

state_start

Activate( ) - invokes StorePrices on output processor and changes state to 0.

state_S0
void Start()

state_S0

Start() – invokes PayMsg on the output processor and changes state to 1 in MDAEFSM.

state_S1
void PayType(int t)

state_S1

PayType(int t) – if credit card payment, invokes DisplayCreditMsg on the output processor and changes state to 2 else if cash payment, invokes StoreCash and DisplayMenu on output processor and changes state to 3, else if debit card payment, invokes StorePin and EnterPinMsg on output processor and changes state to 7.

state_S2
void Approved() void Reject()

state_S2

Approved() - invokes DisplayMenu on output processor and changes state to 3.
Reject() - invokes RejectMsg on output processor and changes state to 0.

state_S3
void SelectGas(int g) void Continue() void Cancel()

state_S3

SelectGas(int g) – invokes SetPrice on output processor
Continue() - changes state to 4
Cancel() - invokes CancelMsg and ReturnCah on output processor

state_S4
void StartPump(int M)

state_S4

StartPump(int M) - invokes SetInitialValues and ReadyMsg on output processor and changes state to 5

state_S5
void Pump(int M) void StopPump()

state_S5

Pump(int M) - invokes PumpGasUnit and GasPumpedMsg on output processor.
StopPump() - invokes StopMsg on output processor and changes state to 6.

state_S6
void Receipt( ) void NoReceipt( )

state_S6

Receipt() - invokes PrintReceipt and ReturnCash on output processor and changes state to 0.
NoReceipt() - invokes ReturnCash on output processor and changes state to 0.

state_S7
void CorrectPin( ) void IncorrectPin( )

state_S7

CorrectPin( ) - invokes DisplayMenu on output processor and changes state to 3.
InorrectPin( ) - invokes WrongPinMsg on output processor and changes state to 0.

STRATEGY PATTERN

cancelmsg_abstract
abstract void CancelMsg()

This class stores the abstract method CancelMsg for which the concrete methods are defined in inherited classes, cancelmsg_gp1 and cancelmsg_gp2.

cancelmsg_gp1
void CancelMsg()

CancelMsg() - This method displays the message “Transaction Cancelled” for GP1.

cancelmsg_gp2
void CancelMsg()

CancelMsg() - This method displays the message “Transaction Cancelled” for GP2.

displaycreditmsg_abstract
abstract void DisplayCreditMsg()

This class stores the abstract method DisplayCreditMsg for which the concrete methods are defined in inherited classes, displaycreditmsg_gp1 and displaycreditmsg_gp2.

displaycreditmsg_gp1
void DisplayCreditMsg()

DisplayCreditMsg() - This method displays the message “Waiting for approval...” after providing credit card as mode of payment for GP1.

displaycreditmsg_gp2
void DisplayCreditMsg()

DisplayCreditMsg() - This method displays the message “Waiting for approval...” after providing credit card as mode of payment for GP2.

displaymenu_abstract
abstract void DisplayMenu()

This class stores the abstract method DisplayMenu for which the concrete methods are defined in inherited classes, displaymenu_gp1 and displaymenu_gp2.

displaymenu_gp1
void DisplayMenu()

DisplayMenu() - This method displays the menu for gas selection in GP1(Regular or Diesel).

displaymenu_gp2
void DisplayMenu()

DisplayMenu() - This method displays the menu for gas selection in GP2(Regular, Super or Premium).

enterpinmsg_abstract
abstract void EnterPinMsg()

This class stores the abstract method EnterPinMsg for which the concrete methods are defined in inherited classes, enterpinmsg_gp1 and enterpinmsg_gp2.

enterpinmsg_gp1
void EnterPinMsg()

EnterPinMsg() - This method displays message to notify user to provide PIN for debit card for GP1.

enterpinmsg_gp2
void EnterPinMsg()

EnterPinMsg() - This method is empty for GP2 since there is no option for debit card in this pump.

gaspumpedmsg_abstract
abstract void GasPumpedMsg(int M, datastore_abstract DS)

This class stores the abstract method GasPumpedMsg for which the concrete methods are defined in inherited classes, gaspumpedmsg_gp1 and gaspumpedmsg_gp2.

gaspumpedmsg_gp1
void GasPumpedMsg(int M, datastore_abstract DS)

GasPumpedMsg() - This class displays quantity of fuel pumped(gallons) and the total amount for GP1.

gaspumpedmsg_gp2
void GasPumpedMsg(int M, datastore_abstract DS)

GasPumpedMsg() - This class displays quantity of fuel pumped(liters) and the total amount for GP2.

initializedata_abstract
abstract void InitializeData(datastore_abstract DS)

This class stores the abstract method InitializeData for which the concrete methods are defined in inherited classes, initializedata_gp1 and initializedata_gp2.

initializedata_gp1
void InitializeData(datastore_abstract DS)

InitializeData() - This method initializes the price to 0 for GP1.

initializedata_gp2
void InitializeData(datastore_abstract DS)

InitializeData() - This method initializes the price to 0 and cash to 0 for GP2.

paymsg_abstract
abstract void PayMsg()

This class stores the abstract method PayMsg for which the concrete methods are defined in inherited classes, paymsg_gp1 and paymsg_gp2.

paymsg_gp1
void PayMsg()

PayMsg() - This method displays the list of payment options GP1(Credit or Debit).

paymsg_gp2
void PayMsg()

PayMsg() - This method displays the list of payment options GP2(Credit or Cash).

printreceipt_abstract
abstract void PrintReceipt(datastore_abstract DS)

This class stores the abstract method PrintReceipt for which the concrete methods are defined in inherited classes, printreceipt_gp1 and printreceipt_gp2.

printreceipt_gp1
void PrintReceipt(datastore_abstract DS)

PrintReceipt() - This method displays the receipt with the quantity in gallons and the total amount for GP1.

printreceipt_gp2
void PrintReceipt(datastore_abstract DS)

PrintReceipt() - This method displays the receipt with the quantity in liters and the total amount for GP2.

pumpgasunit_abstract
Abstract void PumpGasUnit(int M, datastore_abstract DS)

This class stores the abstract method PumpGasUnit for which the concrete methods are defined in inherited classes, pumpgasunit_gp1 and pumpgasunit_gp2.

pumpgasunit_gp1
void PumpGasUnit(int M, datastore_abstract DS)

PumpGasUnit() - This method adds 1 to the total quantity pumped and price of 1 unit of gas to the total for GP1.

pumpgasunit_gp2
void PumpGasUnit(int M, datastore_abstract DS)

PumpGasUnit() - This method adds 1 to the total quantity pumped and price of 1 unit of gas to the total for GP2.

readymsg_abstract
Abstract void ReadyMsg()

This class stores the abstract method ReadyMsg for which the concrete methods are defined in inherited classes, readymsg_gp1 and readymsg_gp2.

readymsg_gp1
void ReadyMsg()

ReadyMsg() - This method displays message saying that the pump is ready for GP1.

readymsg_gp2
void ReadyMsg()

ReadyMsg() - This method displays message saying that the pump is ready for GP2.

rejectmsg_abstract
Abstract void RejectMsg()

This class stores the abstract method RejectMsg for which the concrete methods are defined in inherited classes, rejectmsg_gp1 and rejectmsg_gp2.

rejectmsg_gp1
void RejectMsg()

RejectMsg() - This method displays credit card rejection message for GP1.

rejectmsg_gp2
void RejectMsg()

RejectMsg() - This method displays credit card rejection message for GP2.

returncash_abstract
Abstract void ReturnCash(datastore_abstract DS)

This class stores the abstract method ReturnCash for which the concrete methods are defined in inherited classes, returncash_gp1 and returncash_gp2.

returncash_gp1
void ReturnCash(datastore_abstract DS)

ReturnCash() - This method is empty for GP1 since this pump does not have cash payment option.

returncash_gp2
void ReturnCash(datastore_abstract DS)

ReturnCash() - This method displays the cash to be returned for GP2.

setinitialvalues_abstract
abstract void SetInitialValues(int M, datastore_abstract DS)

This class stores the abstract method SetInitialValues for which the concrete methods are defined in inherited classes, setinitialvalues_gp1 and setinitialvalues_gp2.

setinitialvalues _gp1
void SetInitialValues(int M, datastore_abstract DS)

SetInitialValues() - This method sets G to 0 to start measure the quantity of gas filled in GP1.

setinitialvalues _gp2
void SetInitialValues(int M, datastore_abstract DS)

SetInitialValues() - This method sets L to 0 to start measure the quantity of gas filled and price to price * 1.1 for cash payments in GP2.

setprice_abstract
abstract void SetPrice(int g, datastore_abstract DS);

This class stores the abstract method SetPrice for which the concrete methods are defined in inherited classes, setprice_gp1 and setprice_gp2.

setprice_gp1
void SetPrice(int g, datastore_abstract DS);

SetPrice() - This method sets the price based on gas selection in GP1.

setprice_gp2
void SetPrice(int g, datastore_abstract DS);

SetPrice() - This method sets the price based on gas selection in GP2.

stopmsg_abstract
abstract void StopMsg();

This class stores the abstract method StopMsg for which the concrete methods are defined in inherited classes, stopmsg_gp1 and stopmsg_gp2.

stopmsg_gp1
void StopMsg();

StopMsg() - This method notifies user when the pump has stopped in GP1.

stopmsg_gp2
void StopMsg();

StopMsg() - This method notifies user when the pump has stopped in GP2.

storecash_abstract
abstract void StoreCash(datastore_abstract DS)

This class stores the abstract method StoreCash for which the concrete methods are defined in inherited classes, storecash_gp1 and storecash_gp2.

storecash_gp1
void StoreCash(datastore_abstract DS)

StoreCash() - This method is empty since there no option for cash payment in GP1.

storecash_gp2
void StoreCash(datastore_abstract DS)

StoreCash() - This method stores the cash provided by user in GP2.

storepin_abstract
abstract void StorePin(datastore_abstract DS)

This class stores the abstract method StorePin for which the concrete methods are defined in inherited classes, storepin_gp1 and storepin_gp2.

storepin_gp1
void StorePin(datastore_abstract DS)

StorePin() - This method stores the debit card PIN in GP1.

storepin_gp2
void StorePin(datastore_abstract DS)

StorePin() - This method is empty since there no option for debit card payment in GP2.

storeprices_abstract
abstract void StorePrices(datastore_abstract DS)

This class stores the abstract method StorePrices for which the concrete methods are defined in inherited classes, storeprices_gp1 and storeprices_gp2.

storeprices_gp1
void StorePrices(datastore_abstract DS)

StorePrices() - This method stores the prices of gas during activation in GP1.

storeprices_gp2
void StorePrices(datastore_abstract DS)

StorePrices() - This method stores the prices of gas during activation in GP2.

wrongpinmsg_abstract
abstract void WrongPinMsg()

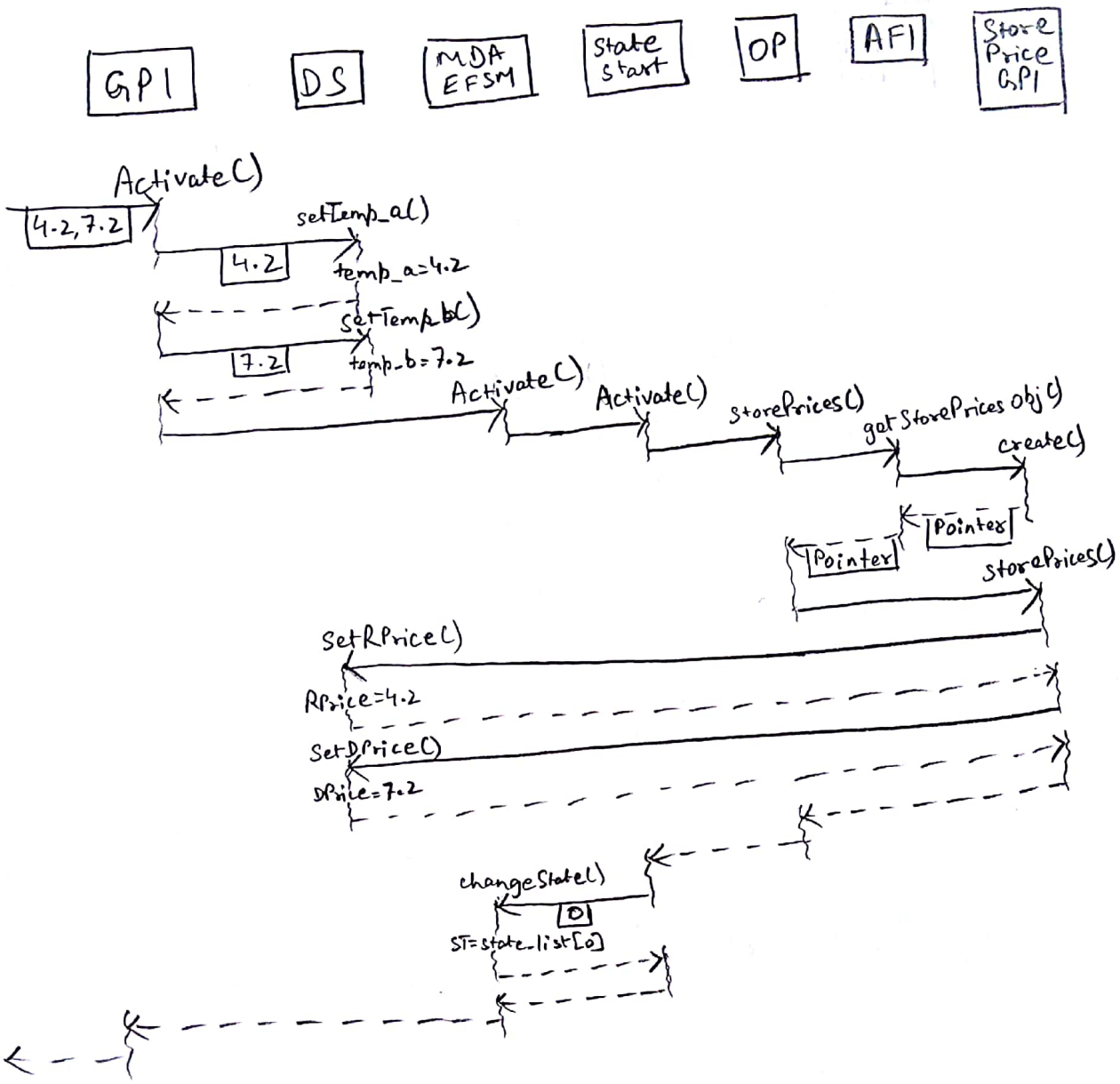
This class stores the abstract method WrongPinMsg for which the concrete methods are defined in inherited classes, wrongpinmsg_gp1 and wrongpinmsg_gp2.

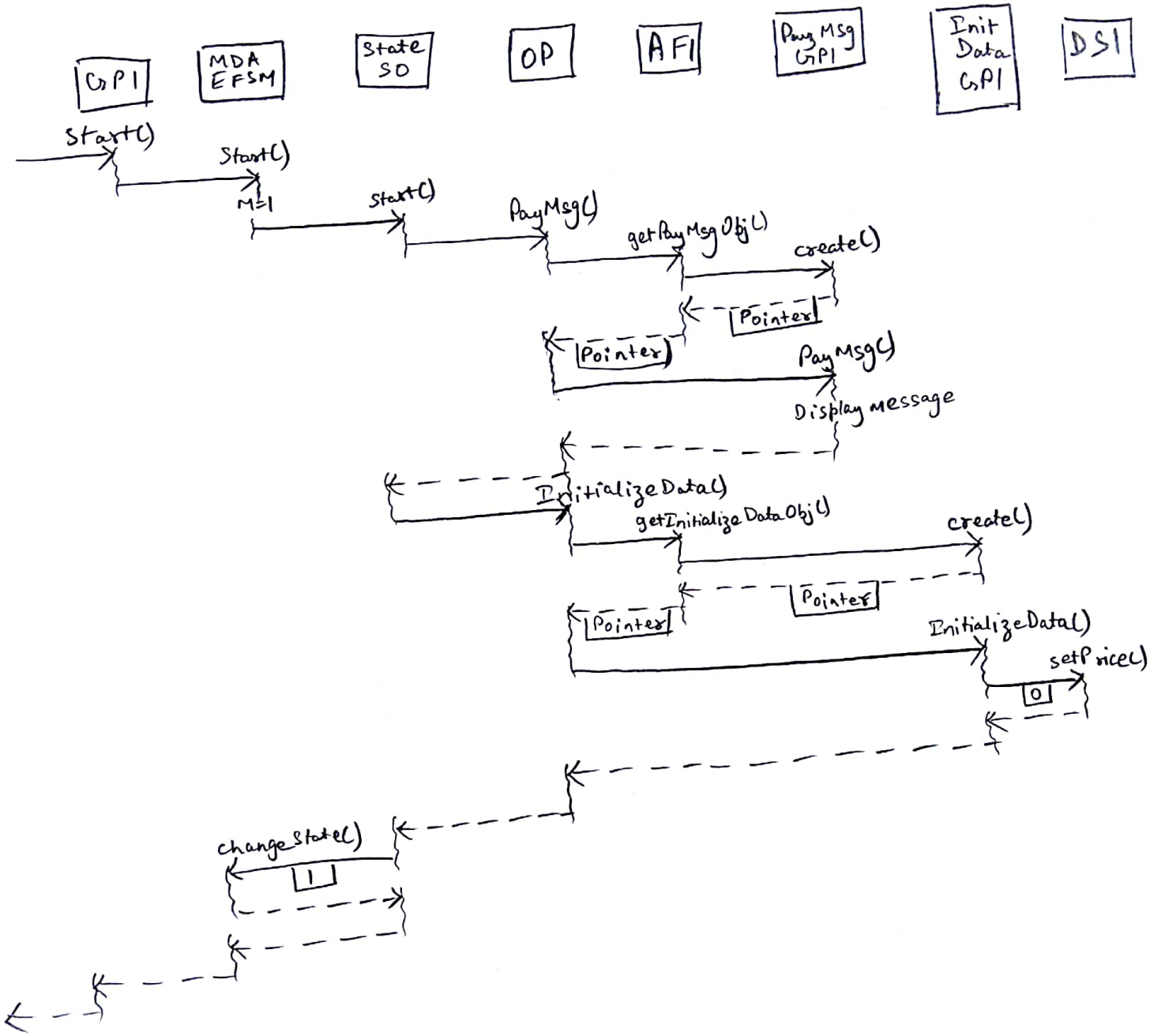
wrongpinmsg_gp1
void WrongPinMsg()

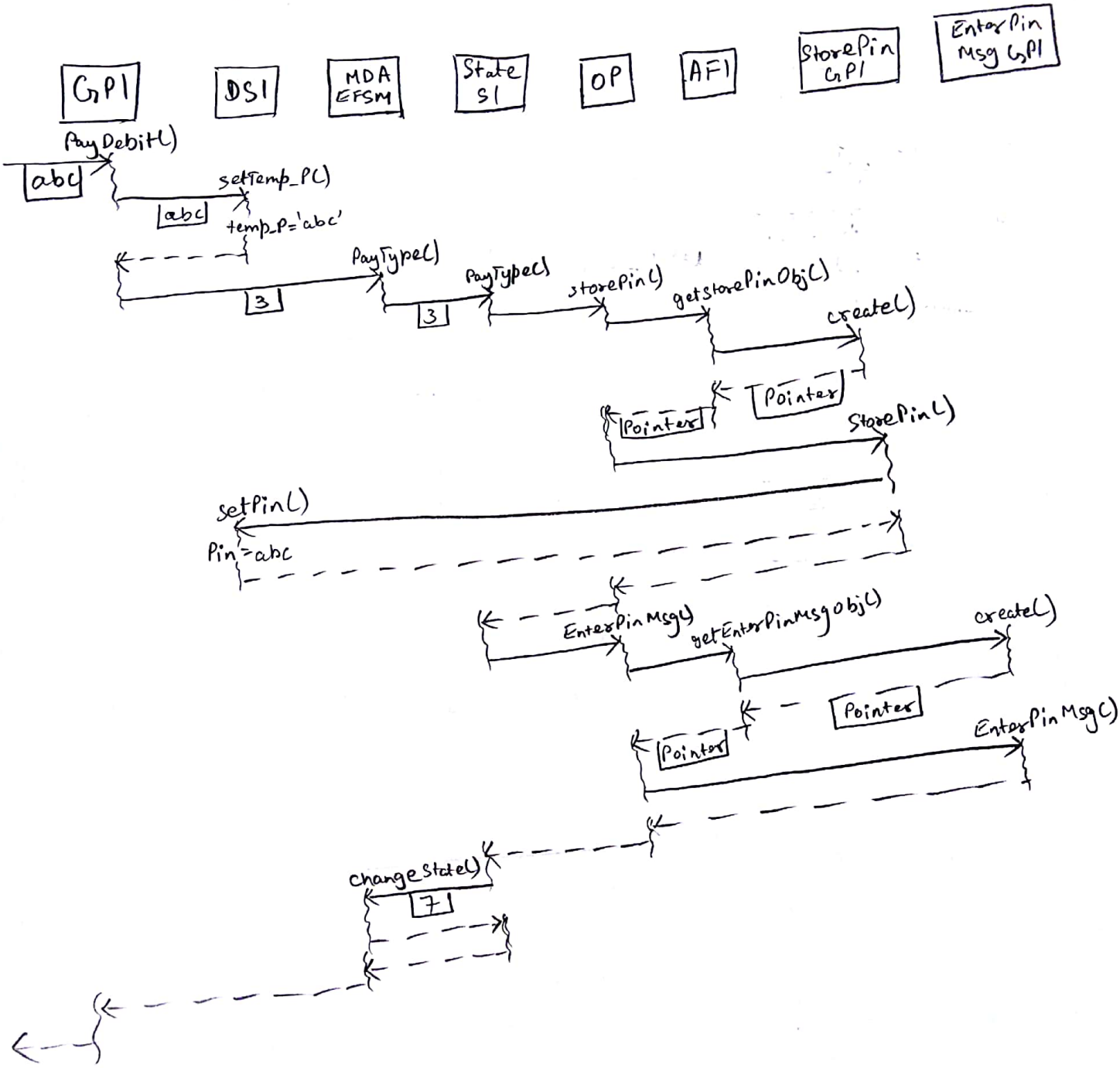
WrongPinMsg() - This method displays message if the PIN provided by user is wrong in GP1.

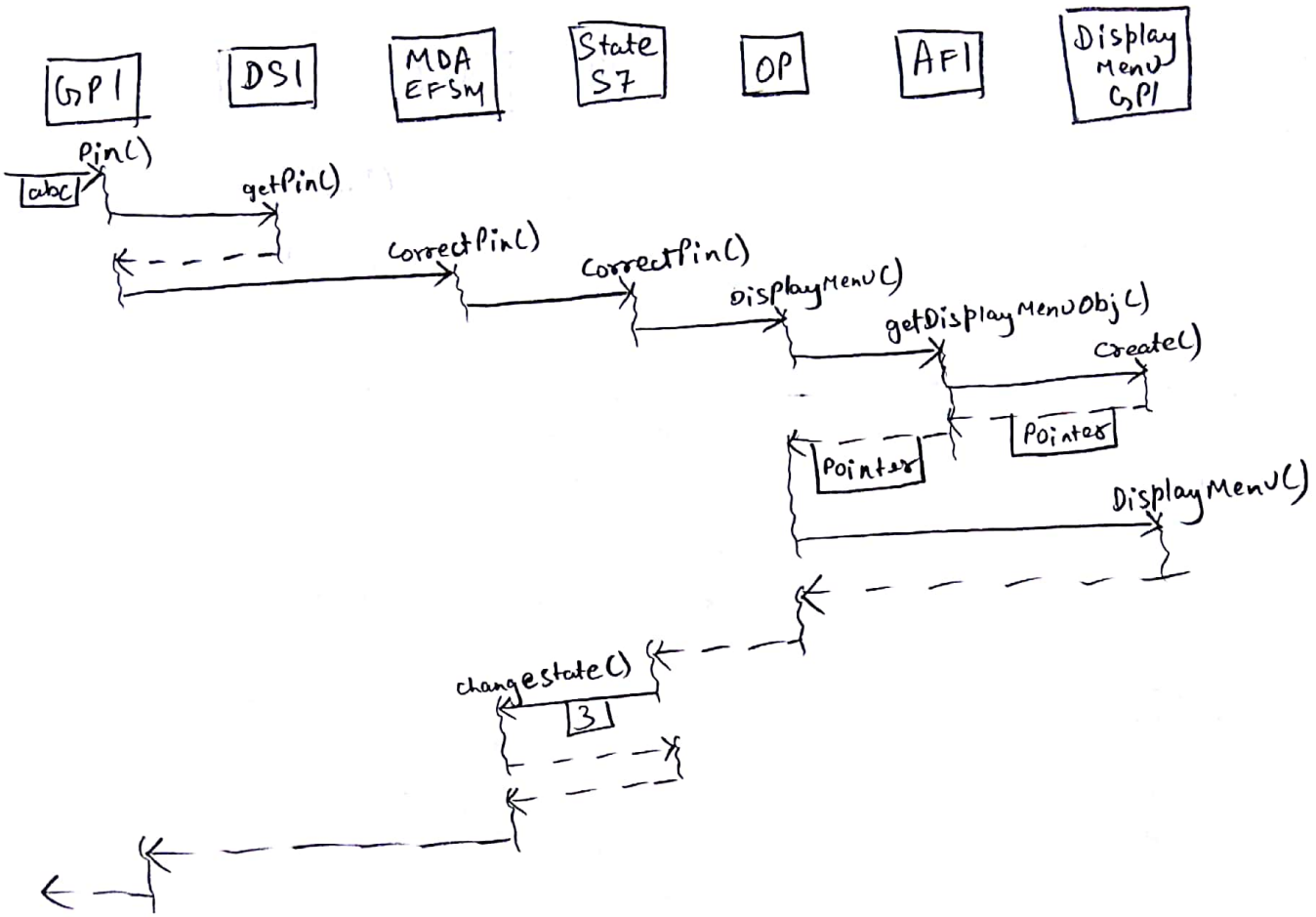
wrongpinmsg_gp1
void WrongPinMsg()

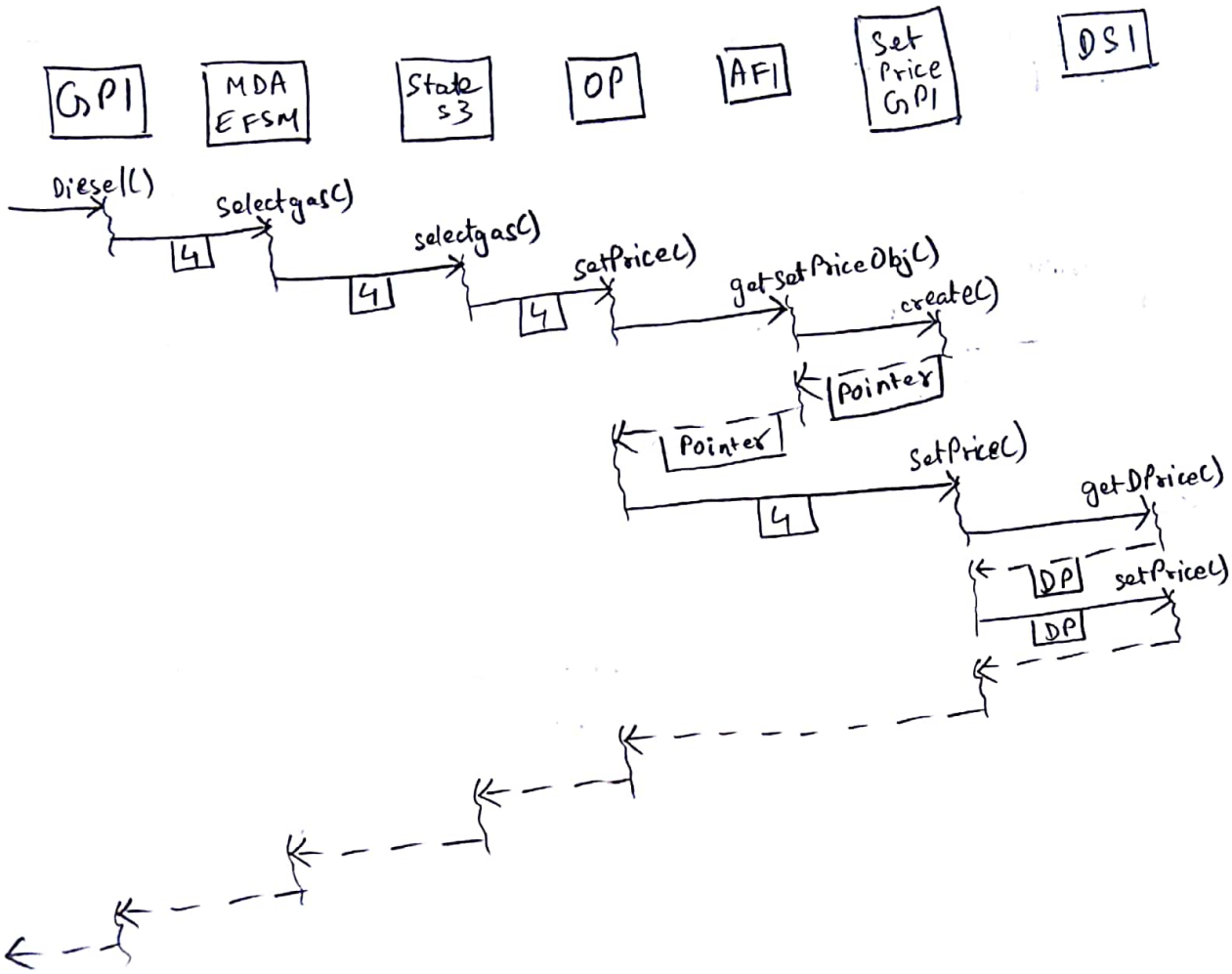
WrongPinMsg() - This method displays message if the PIN provided by user is wrong in GP2.

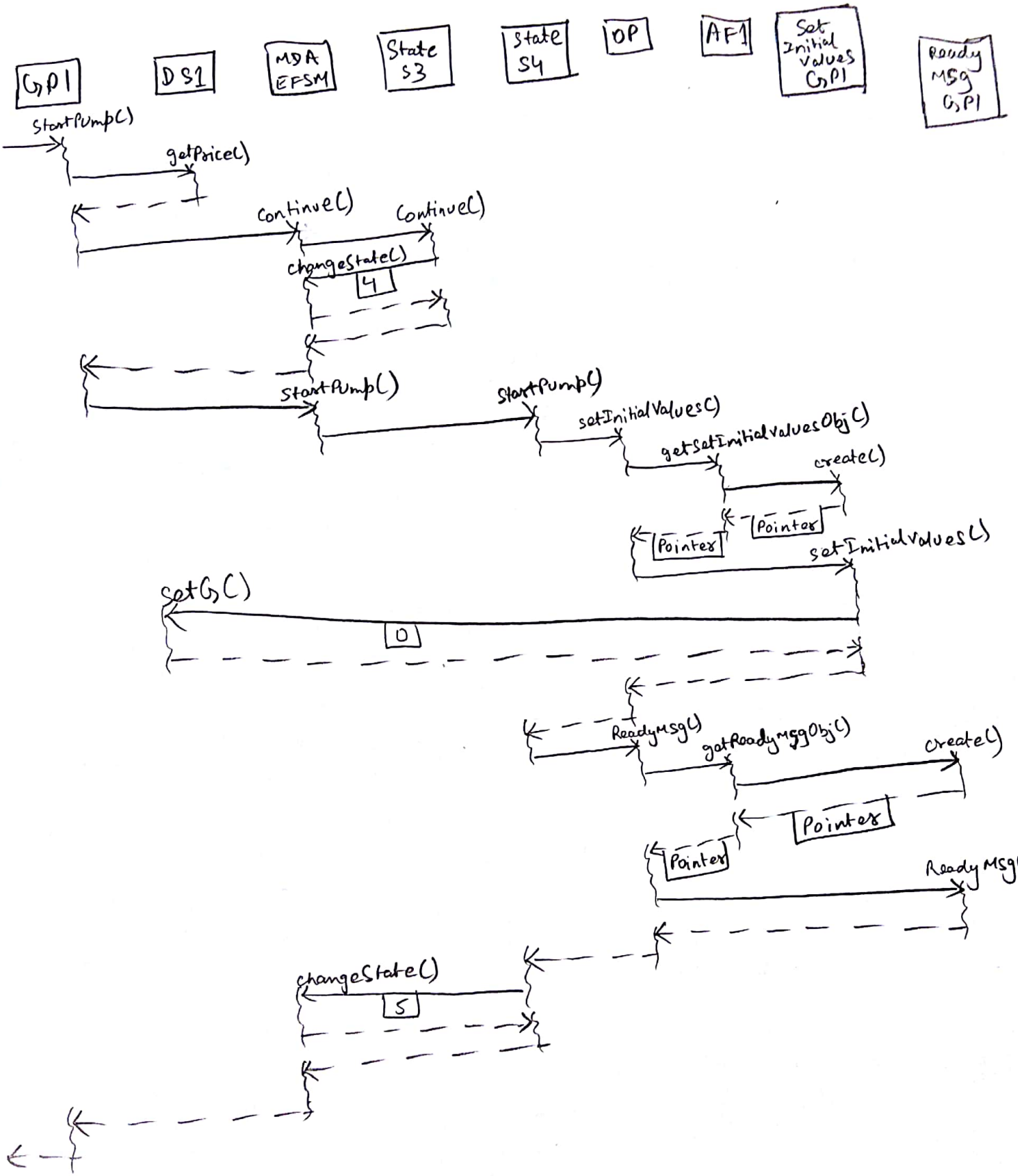

GPI
DS
MDA EFSM
State start
OP
AFI
Store Price GPI
Activate()
4.2,7.2
setTemp_a()
4.2
temp_a=4.2
setTemp_b()
7.2
temp_b=7.2
Activate()
Activate()
storePrices()
get StorePrices obj()
create()
Pointer
Pointer
storePrices()
SetRPrice()
RPrice=4.2
SetDPrice()
DPrice=7.2
changeState()
0
ST=state_list[0]

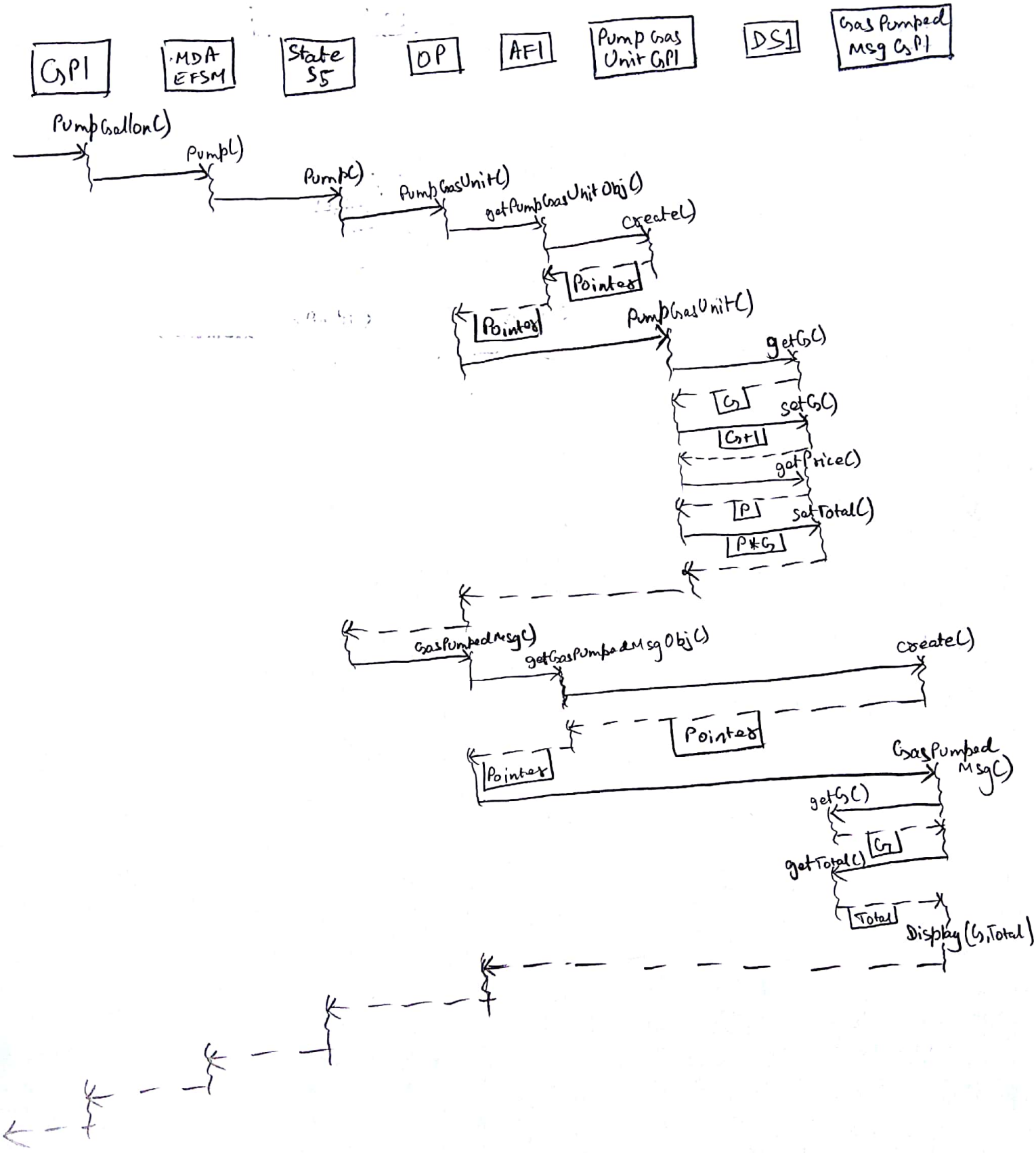

CsPI
MDA EFSM
State SO
OP
AFI
Pay Msg CsPI
Init Data CsPI
DSI
start()
start()
M=1
start()
PayMsg()
getPayMsgObj()
create()
Pointer
Pointer
PayMsg()
Display message
InitializeData()
getInitializeDataObj()
create()
Pointer
Pointer
InitializeData()
setPrice()
0
changeState()
1

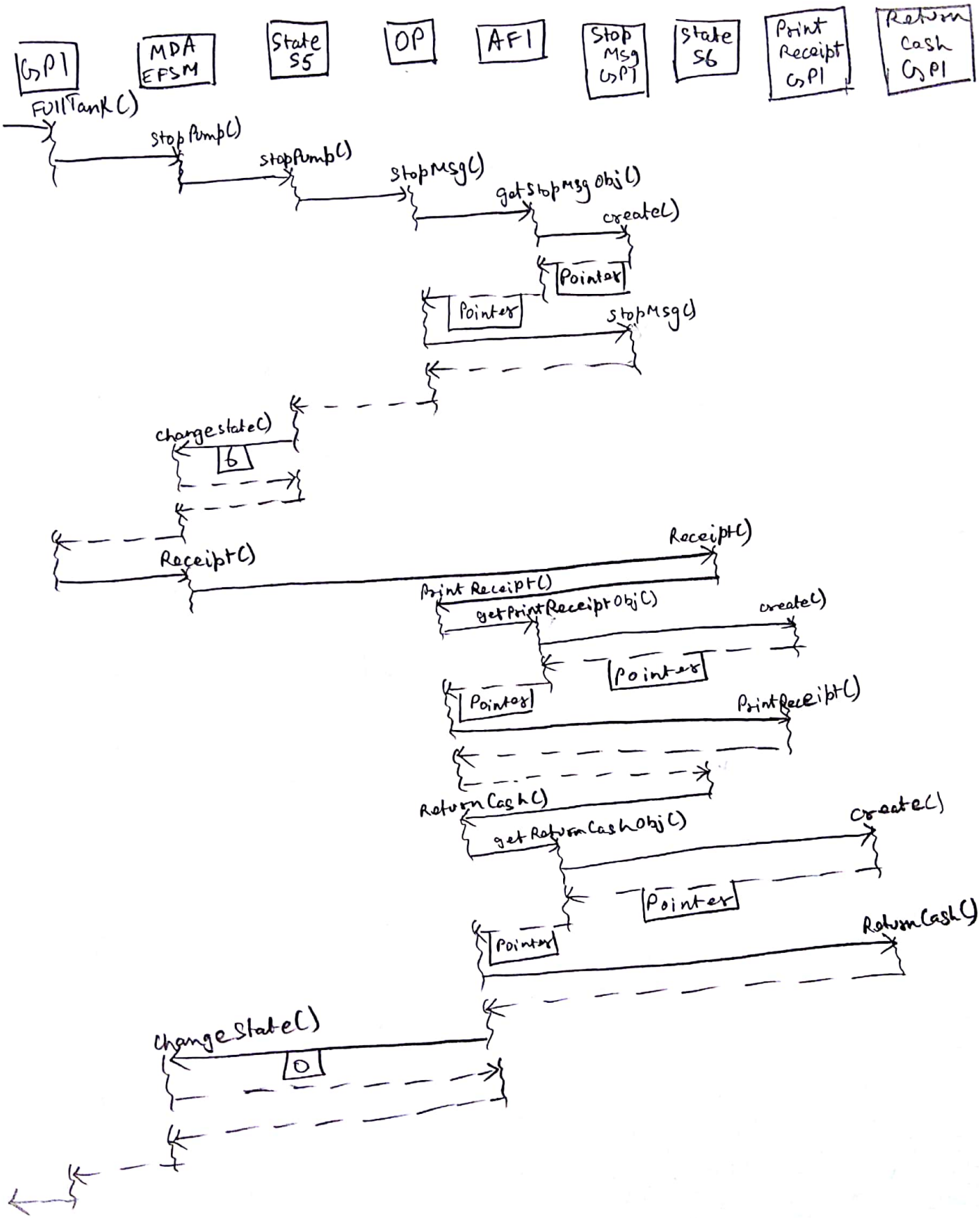

GPI
DSI
MDA EFSM
State S1
OP
AFI
StorePin GPI
EnterPin Msg GPI
PayDebit()
abc
setTemp_P()
abc
temp_P='abc'
PayType()
3
PayType()
3
storePin()
getstorePinObj()
create()
Pointer
Pointer
StorePin()
setPin()
Pin=abc
EnterPinMsg()
getEnterPinMsgObj()
create()
Pointer
Pointer
EnterPinMsg()
changeState()
7


GPI
DSI
MDA EFSM
State S7
OP
AFI
Display Menu GPI
Pin()
abc
getPin()
correctPin()
correctPin()
displayMenu()
getDisplayMenuObj()
create()
Pointer
Pointer
displayMenu()
changestate()
3

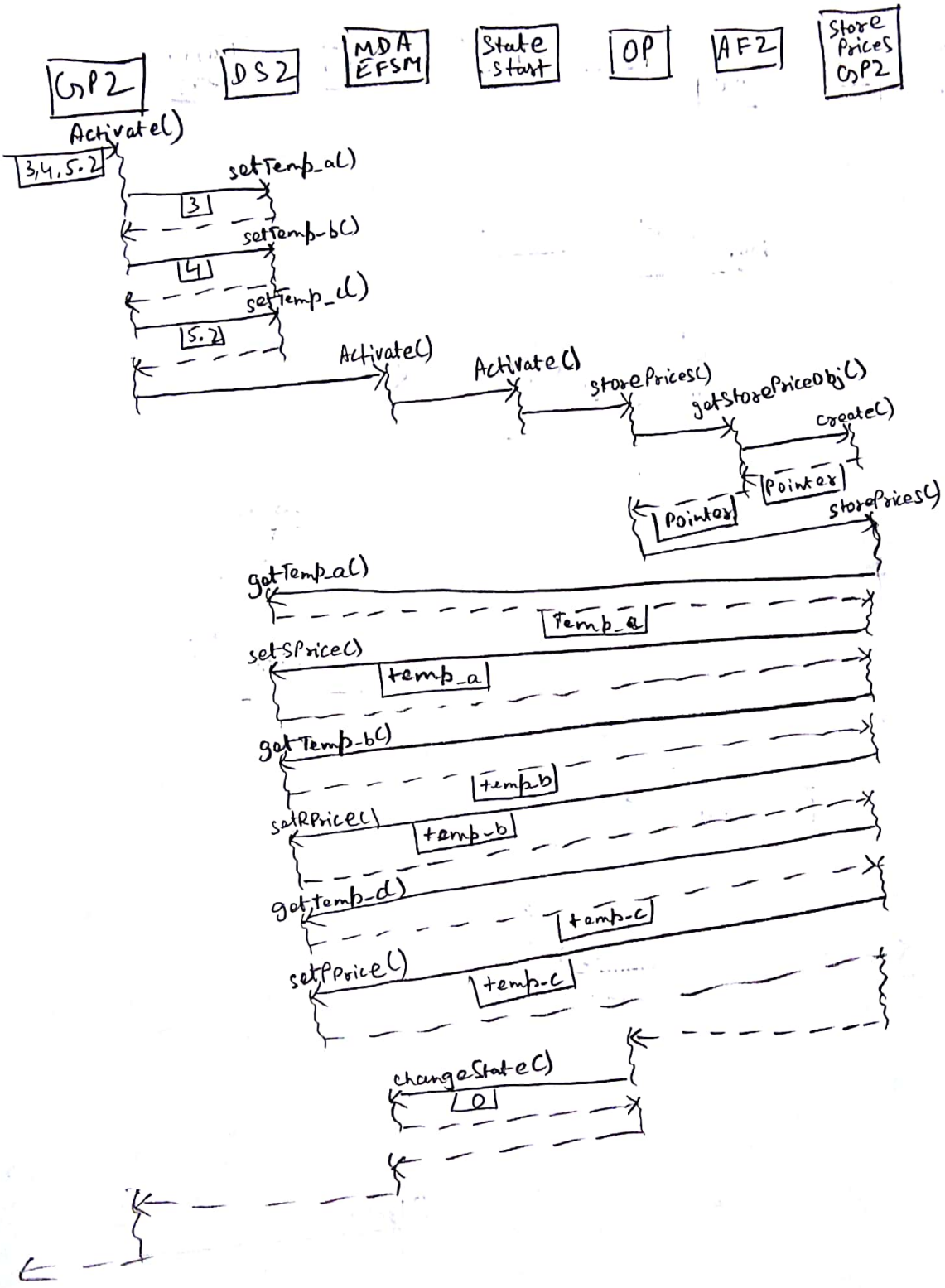

GPI
MDA EFSM
State S3
OP
AFI
Set Price GPI
DSI
Diesel()
selectgas()
selectgas()
setPrice()
getSetPriceObj()
create()
Pointer
Pointer
setPrice()
getDPrice()
DP
setPrice()
DP

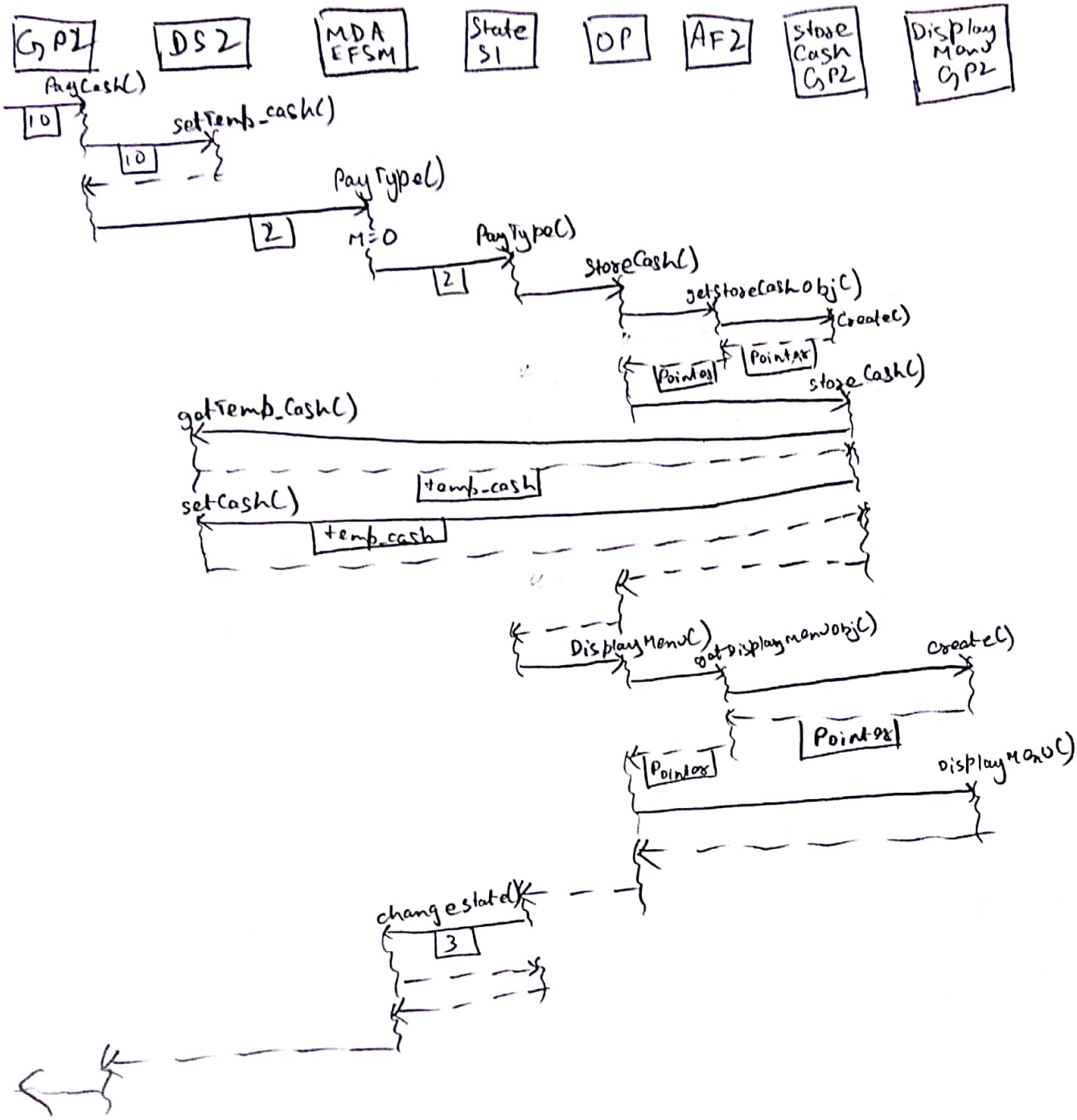

G,P1
DS1
MDA EFSM
State S3
State S4
OP
AF1
Set Initial Values G,P1
Ready Msg G,P1
StartPump()
getPrice()
Continue()
Continue()
changeState()
4
StartPump()
StartPump()
setInitialValues()
getSetInitialValuesObj()
create()
Pointer
Pointer
setInitialValues()
setG,()
0
ReadyMsg()
getReadyMsgObj()
create()
Pointer
Pointer
ReadyMsg()
changeState()
5

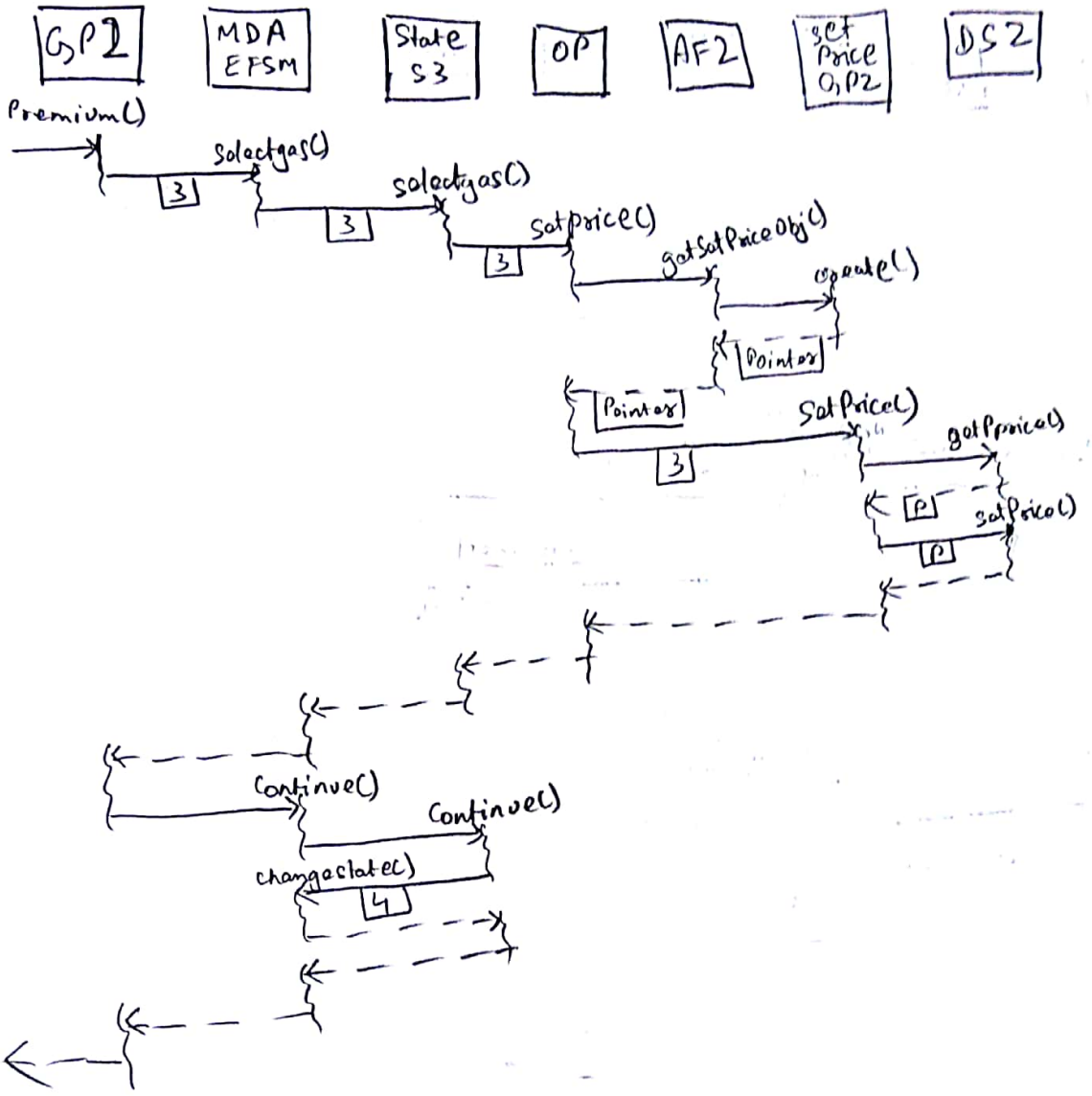

GPI
MDA EFSM
State S5
DP
AFI
Pump Gas Unit GPI
DS1
Gas Pumped Msg GPI
PumpGallon()
Pump()
Pump()
PumpGasUnit()
getPumpGasUnitObj()
create()
Pointer
Pointer
PumpGasUnit()
getG()
G
setG()
G+1
getPrice()
P
setTotal()
P*G
GasPumpedMsg()
getGasPumpedMsgObj()
create()
Pointer
Pointer
GasPumped Msg()
getG()
G
getTotal()
Total
Display(G,Total)

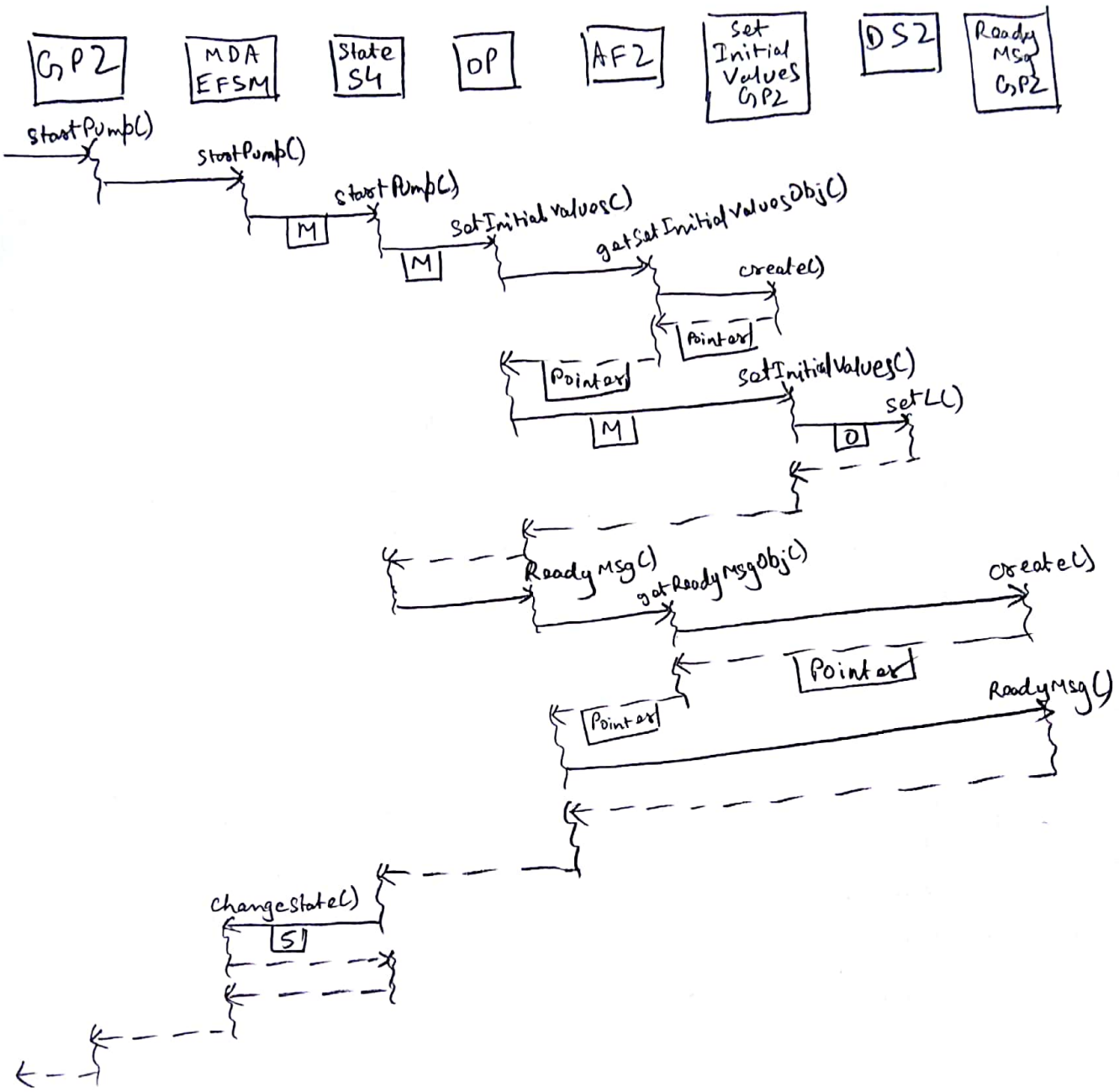

GPI
MDA EFSM
State S5
OP
AFI
Stop Msg GPI
State S6
Print Receipt GPI
Return Cash GPI
FullTank()
stopPump()
stopPump()
stopMsg()
getStopMsgObj()
create()
Pointer
Pointer
stopMsg()
changeState()
6
Receipt()
Receipt()
PrintReceipt()
getPrintReceiptObj()
create()
Pointer
Pointer
PrintReceipt()
ReturnCash()
getReturnCashObj()
create()
Pointer
Pointer
ReturnCash()
changeState()
0

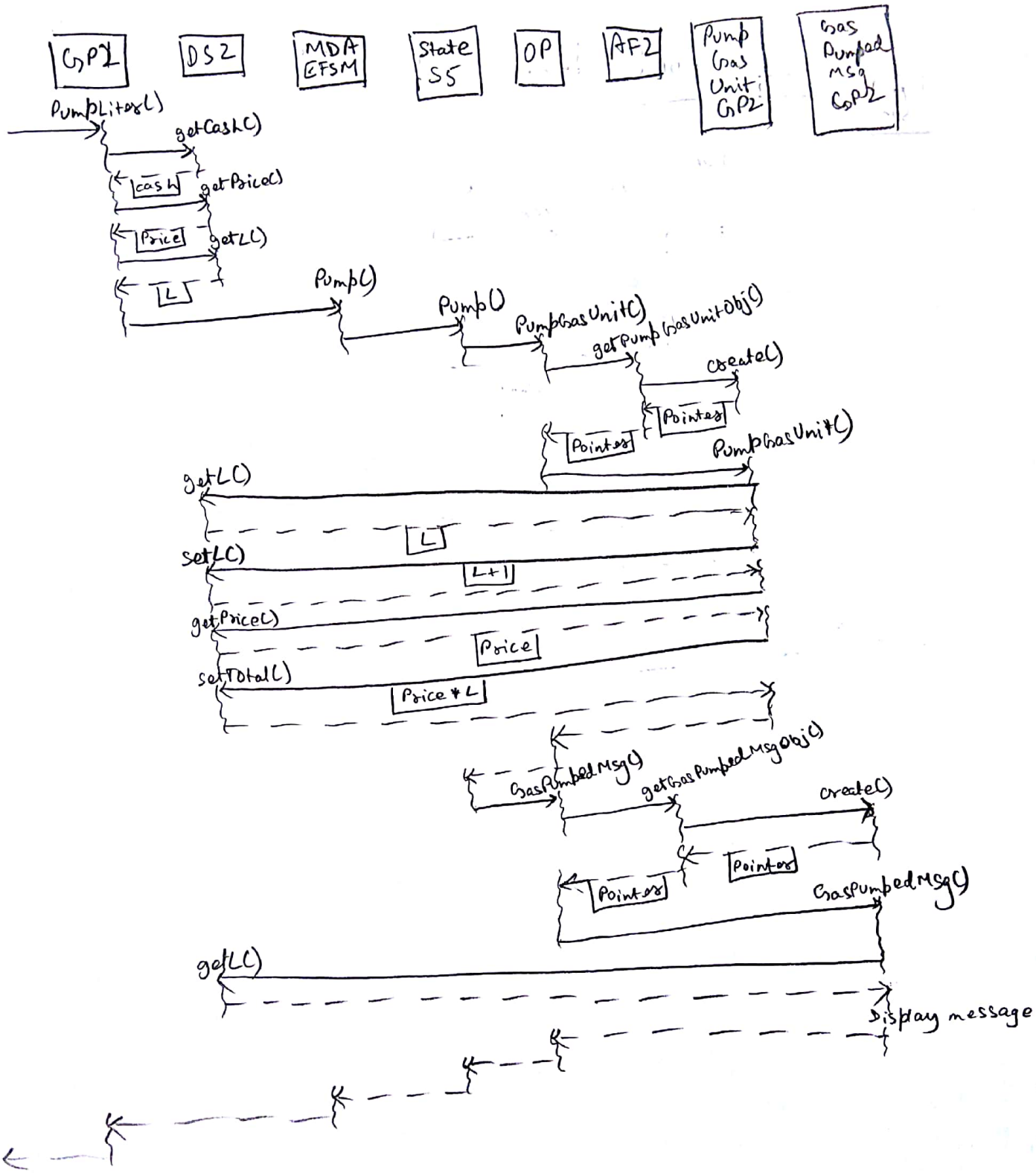
9b

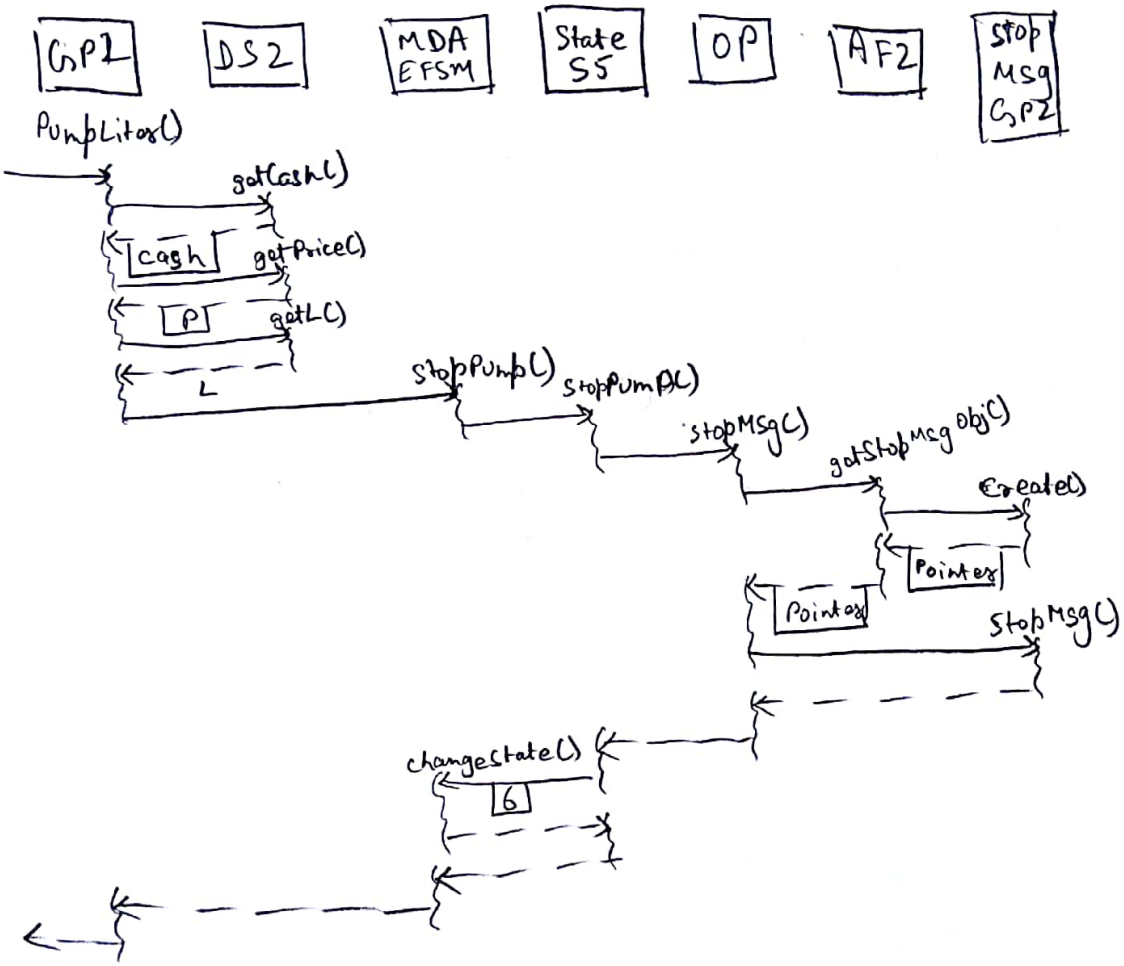

CSP2
DS2
MDA EFSM
State start
OP
AF2
Store Prices CSP2
Activate()
3,4,5.2
setTemp_a()
3
setTemp_b()
4
setTemp_c()
5.2
Activate()
Activate()
storePrices()
getStorePriceObj()
create()
Pointer
Pointer
storePrices()
getTemp_a()
Temp_a
setSPrice()
temp_a
getTemp_b()
temp_b
setRPrice()
temp_b
getTemp_d()
temp_c
setPPrice()
temp_c
changeState()
0

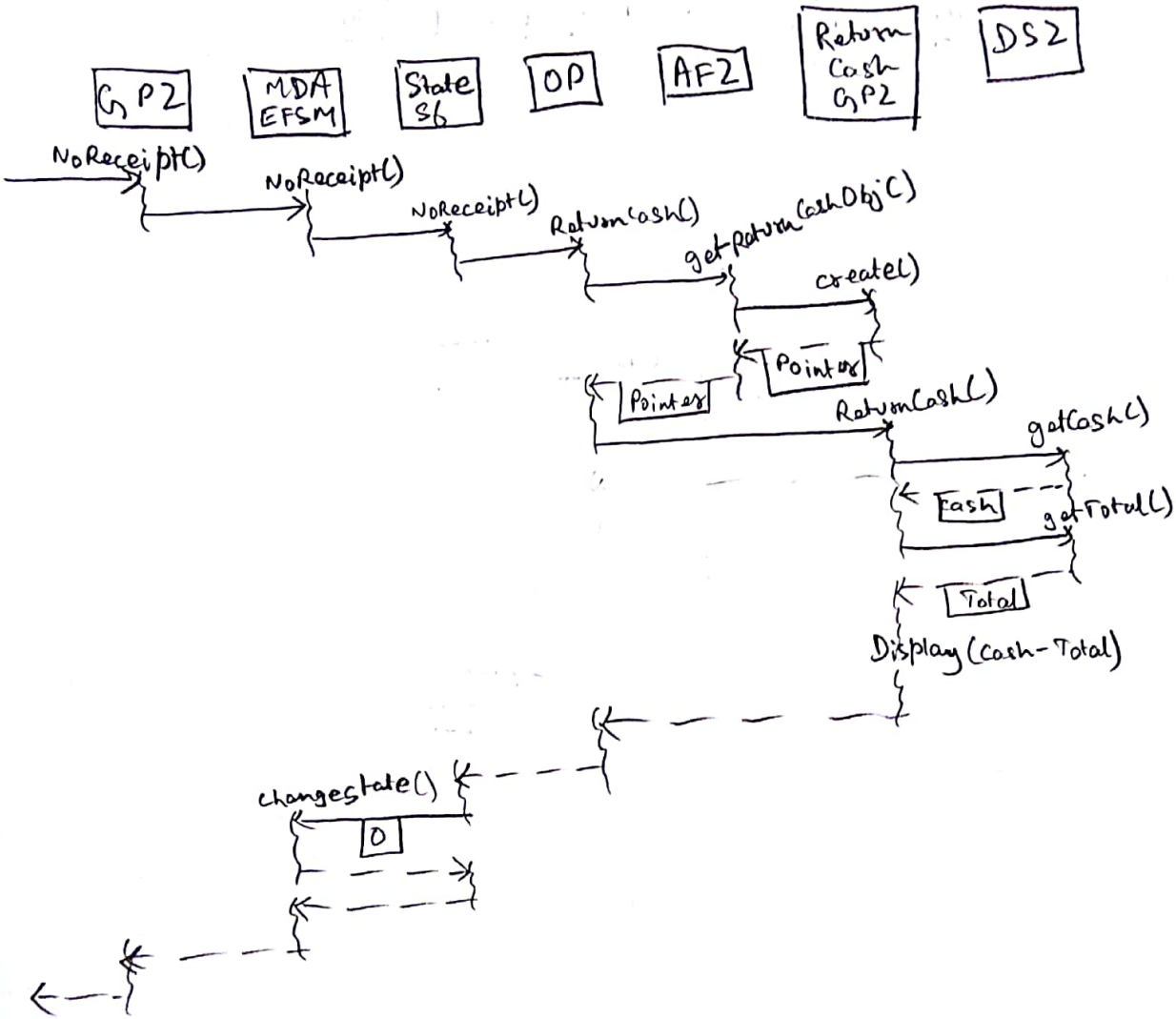

GP2
DS2
MDA EFSM
State S1
OP
AF2
store Cash GP2
Display Manu GP2
PayCash()
10
setTemp_cash()
10
PayTypo()
2
M=0
PayTypo()
2
StoreCash()
getstoreCashObjC)
Create()
Pointer
PointAt
store_Cash()
getTemp_Cash()
temp_cash
setCash()
temp_cash
DisplayMenu()
getDisplayManuObj()
Create()
Pointer
Pointer
displayManu()
changestate()
3


GP2
MDA EFSM
State S3
OP
AF2
set Price O,P2
DS2
Premium()
Selectgas()
3
selectgas()
3
setprice()
3
getSetPriceObj()
operate()
Pointer
Pointer
SetPrice()
3
getPrice()
P
setPrice()
P
Continue()
Continue()
changeState()
4


GP2
MDA EFSM
State S4
OP
AF2
Set Initial Values GP2
DS2
Ready Msg GP2
startPump()
startPump()
startPump()
M
setInitialValues()
getSetInitialValuesObj()
create()
Pointer
Pointer
setInitialValues()
M
setL()
0
ReadyMsg()
getReadyMsgObj()
create()
Pointer
Pointer
ReadyMsg()
changeState()
S


GP2
DS2
MDA EFSM
State SS
OP
AF2
Pump Gas Unit GP2
Gas Pumped Msg GP2
PumpLiter()
getCash()
cash
getPrice()
Price
getL()
L
Pump()
Pump()
PumpGasUnit()
getPumpGasUnitObj()
create()
Pointer
Pointer
PumpGasUnit()
getL()
L
setL()
L+1
getPrice()
Price
setTotal()
Price * L
GasPumpedMsg()
getGasPumpedMsgObj()
create()
Pointer
Pointer
GasPumpedMsg()
getL()
Display message


GSP2
DS2
MDA EFSM
State S5
OP
AF2
Stop Msg GSP2
PumpLiters()
getCash()
cash
getPrice()
P
getL()
L
stopPump()
stopPump()
stopMsg()
getStopMsg Obj()
Create()
Pointer
Pointer
StopMsg()
changeState()
6


GP2
MDA EFSM
State S6
OP
AF2
Return Cash GP2
DS2
NoReceipt()
NoReceipt()
NoReceipt()
ReturnCash()
getReturnCashObj()
create()
Pointer
Pointer
ReturnCash()
getCash()
Cash
getTotal()
Total
Display(Cash-Total)
changestate()
0