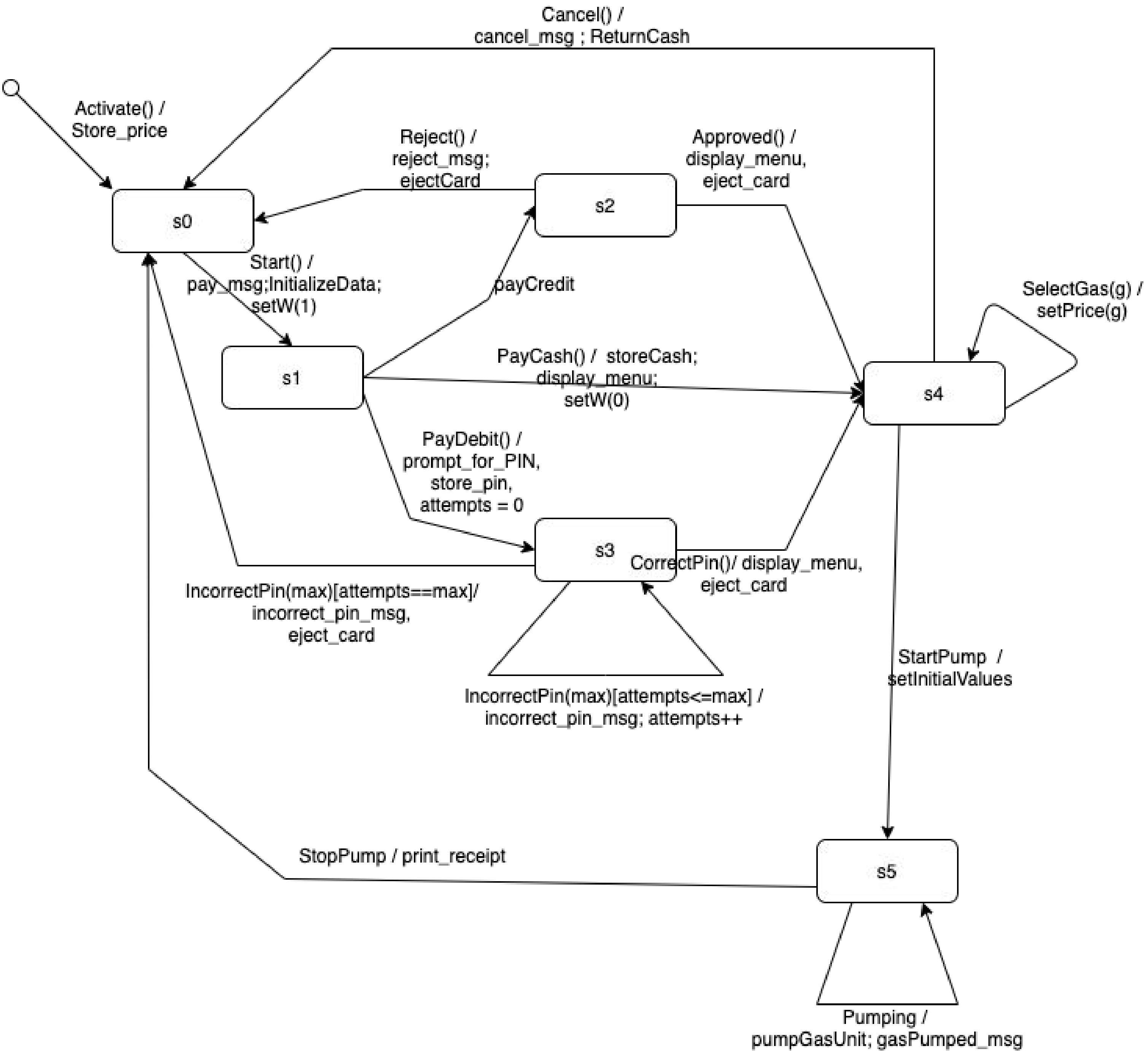
1. MDA-EFSM model for the Gas Pump components

MDA-EFSM Events:
Activate()
Start()
PayCredit()
PayCash()
PayDebit()
IncorrectPin(int max)
CorrectPin()
Reject()
Cancel()
Approved()
StartPump()
StopPump()
Pumping()
SelectGas(int g)

MDA-EFSM Actions:
store_pin //store the pin from the temporary data store
store_price //store the each price of the gas from the temporary data store
store_cash //stores cash from the temporary data store
pay_msg //display a type of payment method
reject_msg //display credit card not approved message
cancel_msg //display cancel message
gasPumped_msg //displays the amount of disposed gas
setPrice(int g) //set the price for the gas identified by g indentifier as in SelectGas(int g)
setInitialValues //set G/L and total to 0
setW(int w) //set value for cash flag
initializeData //set the value of price to 0 for GP-2; do nothing for GP-1
pumpGasUnit //disposes unit of gas and counts # of units disposed
prompt_for_PIN //prompts to enter Pin
incorrect_pin_msg //dispaly incorrect pin message
display_menu //display a menu with a list of selections
print_receipt //print the receipt
returnCash //returns the remaining cash
eject_card //eject the card


Cancel() /
cancel_msg ; ReturnCash
Activate() /
Store_price
Reject() /
reject_msg;
ejectCard
Approved() /
display_menu,
eject_card
s0
s2
Start() /
pay_msg;InitializeData;
setW(1)
payCredit
SelectGas(g) /
setPrice(g)
s1
PayCash() / storeCash;
display_menu;
setW(0)
s4
PayDebit() /
prompt_for_PIN,
store_pin,
attempts = 0
s3
CorrectPin()/ display_menu,
eject_card
IncorrectPin(max)[attempts==max]/
incorrect_pin_msg,
eject_card
StartPump /
setInitialValues
IncorrectPin(max)[attempts<=max] /
incorrect_pin_msg; attempts++
StopPump / print_receipt
s5
Pumping /
pumpGasUnit; gasPumped_msg

Operations of IP

GP-1:

data structure:

m: pointer to the MDA-EFSM

d: pointer to the data store

af: pointer to the AbstractFactory

in the data store:

price: the price of the gas per liter

cash: contains the value of cash deposited

L: contains the number of liters already pumped

w: cash flag(cash:w=0; otherwise: w=1)

```
Activate(int a){
    if(a>0){
        d->temp_a = a
        m->Activate()   }
}
Start(){
    m->Start()
}
PayCash(int c){
    if(c>0){
        d->tmp_c= c
        m->payCash()   }
}
PayCredit(){
    m->PayCredit()
}
Reject(){
    m->Reject()
}
Cancel(){
    m->Cancel()
}
Approved(){
    m->Approved()
}
StartPump(){
    m->StartPump()
}
PumpLiter(){
    if(d->w == 1)
        m->Pumping()
    else if( d->cash < d->price *(d->L + 1) && d->cash > 0){
        m->StopPump()   }
    else {   m->Pumping()     }
}
StopPump(){
    m->StopPump()      }
```

GP-2:

data structure:
m: pointer to the MDA-EFSM
d: pointer to the data store
af: AbstractFactory
SelectGas(g): Regular: g=1; Super: g=2; Diesel: g=3

```
Activate(float a,float b,float c){
    if((a>0)&&(b>0)&&(c>0)){
        d->tmp_a = a
        d->tmp_b = b
        d->tmp_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(x == d->pin){
        m->CorrectPin()
    }
    else
        m->IncorrectPin(3)
}
Cancel(){
    m->Cancel()
}
Approved(){
    m->Approved()
}
Diesel(){
    m->SelectGas(3)
}
Regular(){
    m->SelectGas(1)
}
```

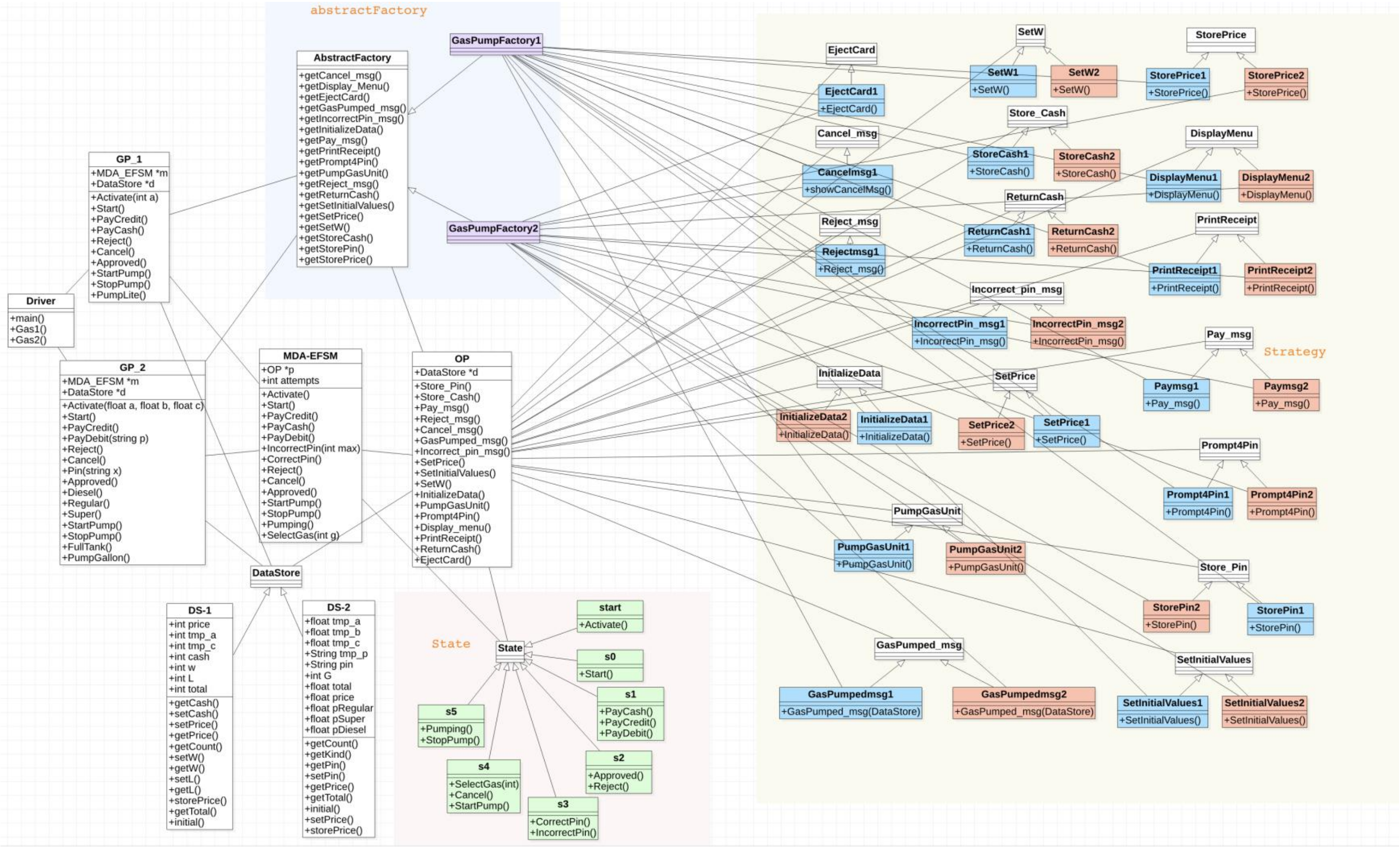
```
Super(){
	m->SelectGas(2)
}
StartPump(){
	m->StartPump()
}
PumpGallon(){
	m->Pumping()
}
StopPump(){
	m->StopPump()
}
FullTank(){
	m->StopPump()
}
```

2. Class diagram(s) of the MDA of the Gas Pump components.

a. State pattern

b. Strategy pattern

c. Abstract factory pattern



3. For each class in the class diagram(s)

a. Describe the purpose of the class, i.e., responsibilities.

b. Describe the responsibility of each operation supported by each class.

Driver: the entrance of the application. To get the operations and input data.

GasPump: two gas pump with different operations. Make MDFAEFSM to do associated operations. store data into temporary data.

MDAEFSM: control the current state to do operations. Each operations matches one MDAEFSM Events.

State: 6 different states, each state has their associated operations. Also they can change to the next state.

Start : to activate the system and store data from the temporary data. Then change to S0.

S0 : to start the system, show the pay message and initialize the data for GP2, setW for GP1, then change to S1

S1 : it can choose to PayCredit, PayDebit and PayCash. For Credit, just change to S2. For Debit, it stores the correct pin and prompts for pin, then change to S3. For Cash, stores the cash amount, set W as 0, display Menu and then change to S4.

S2 : It has reject or approved. Reject: show the message, eject the credit card and return to S0. Approved: display the menu, eject the card and change to S4.

S3 : It has IncorrectPin and CorrectPin. Incorrect: It will check the attempt times and do it again or eject card with backing to S0. Correct: display the menu, eject the card and change to S4.

S4 : Cancel:show the cancel message, return the cash and return to S0. StartPump: set total and gas count to 0. change to S5. SelectGas: choose one kind of gas and set the price as that gas’s price.

S5: StopPump : print receipt and return back to S0. or Pumping: show the gas unit and gas kind.

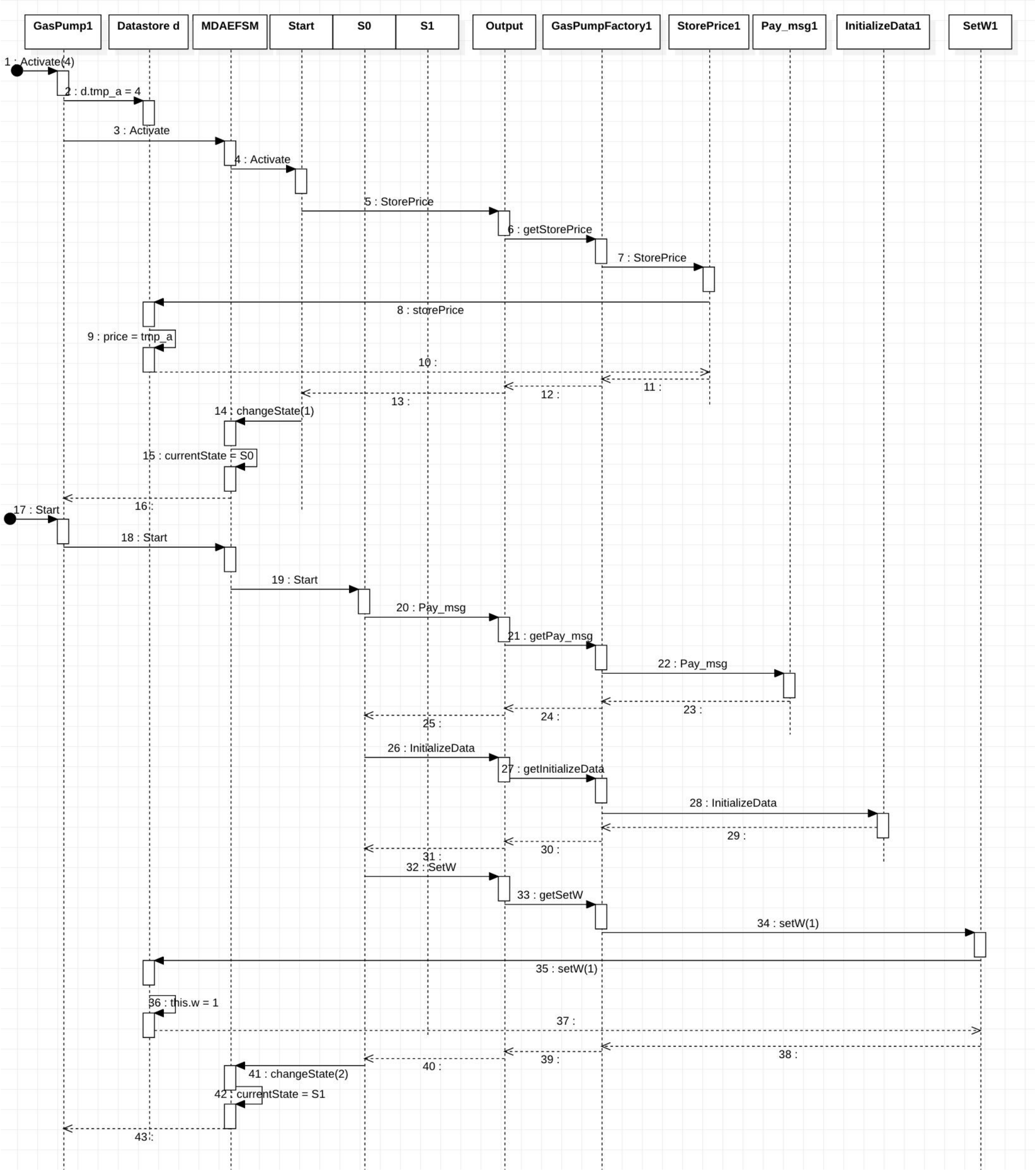
Output: make abstarctFactory to do the output operations such as: store data and show the message.

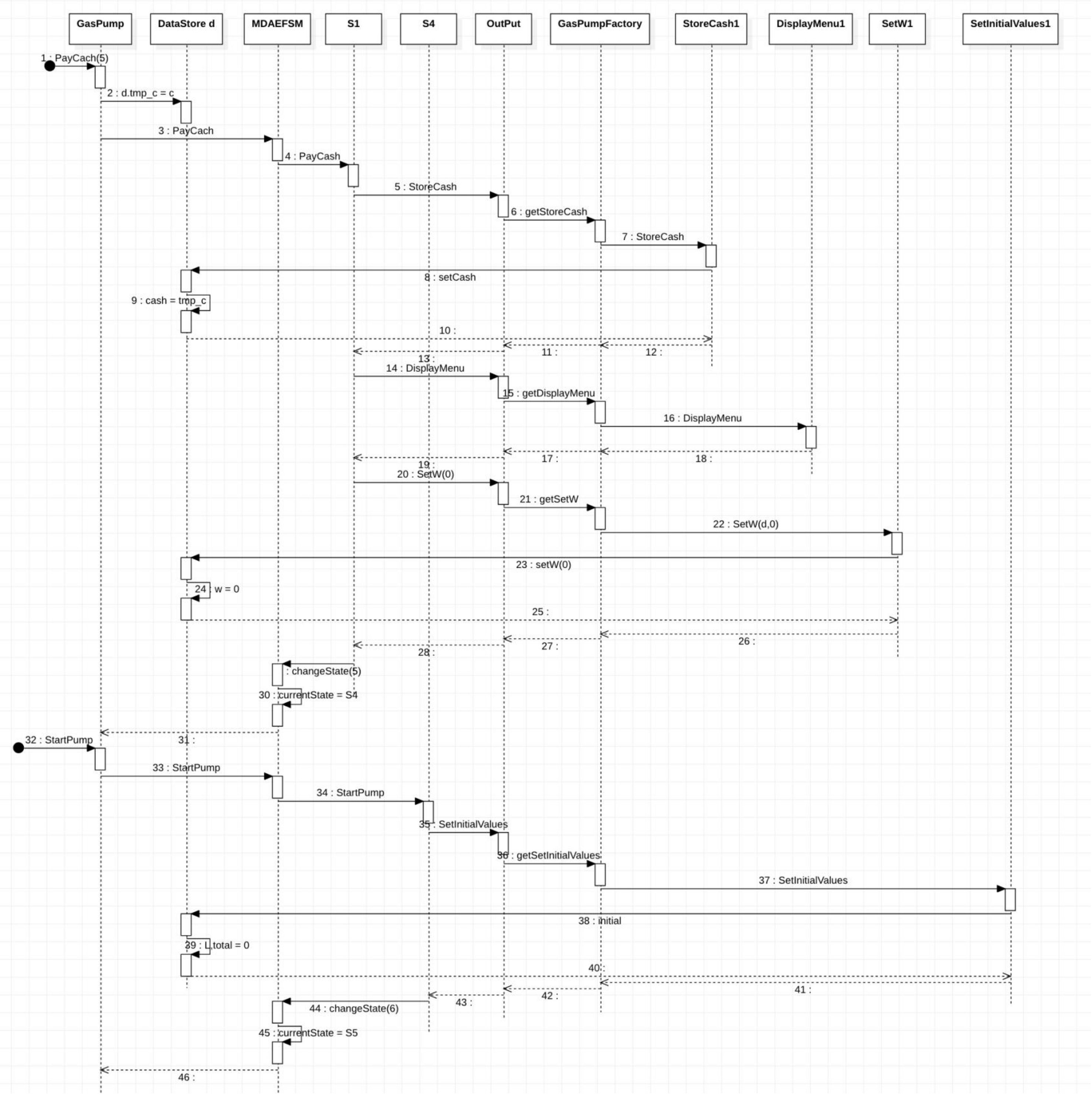
AbstractFactory: 2 different factory to get different strategy method.

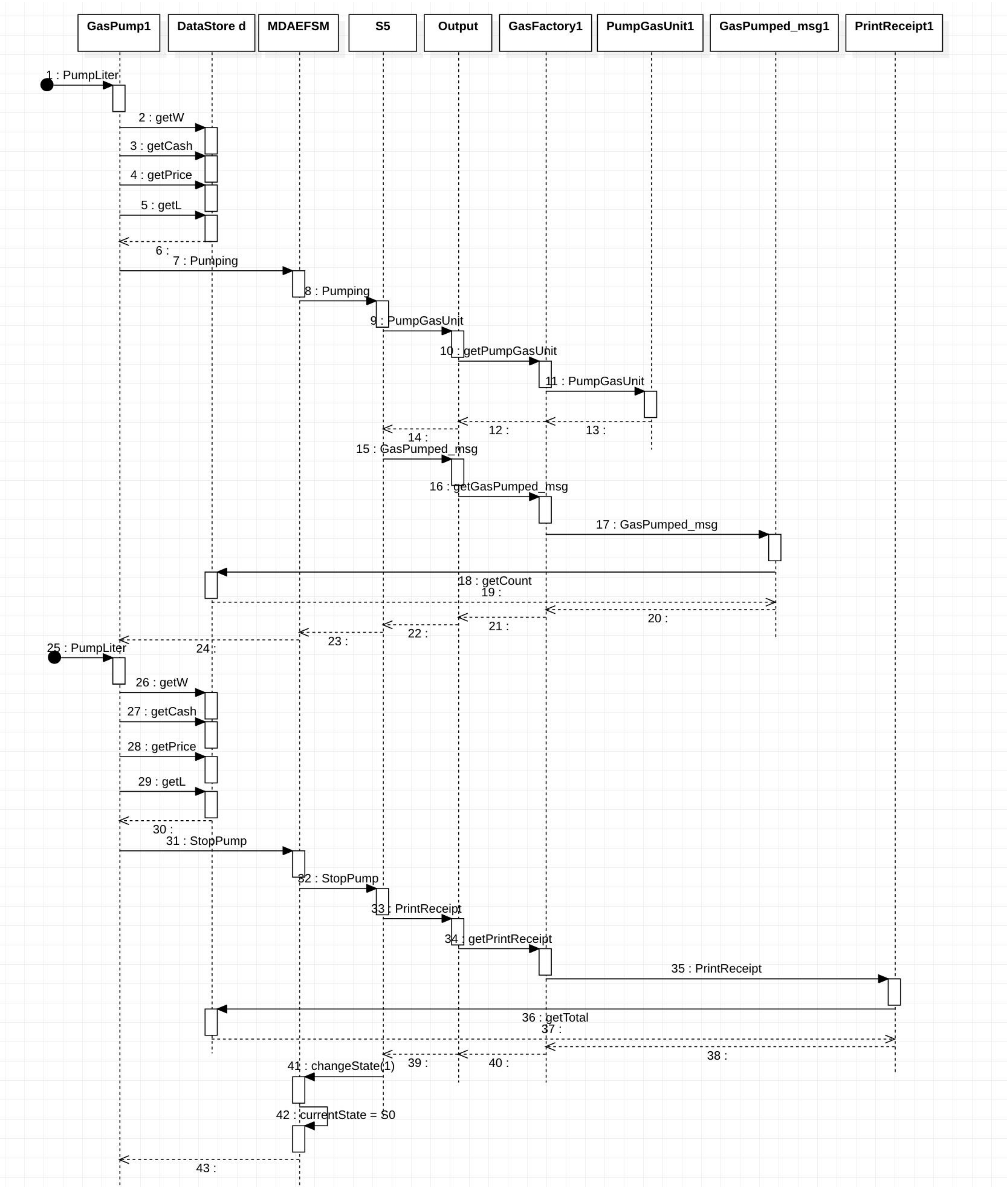
Strategy: each strategy do one operation and has 2 version of operations to fit different needs. Each strategy matches one MDAEFSM actions.

4.Provide sequence diagrams for two Scenarios:

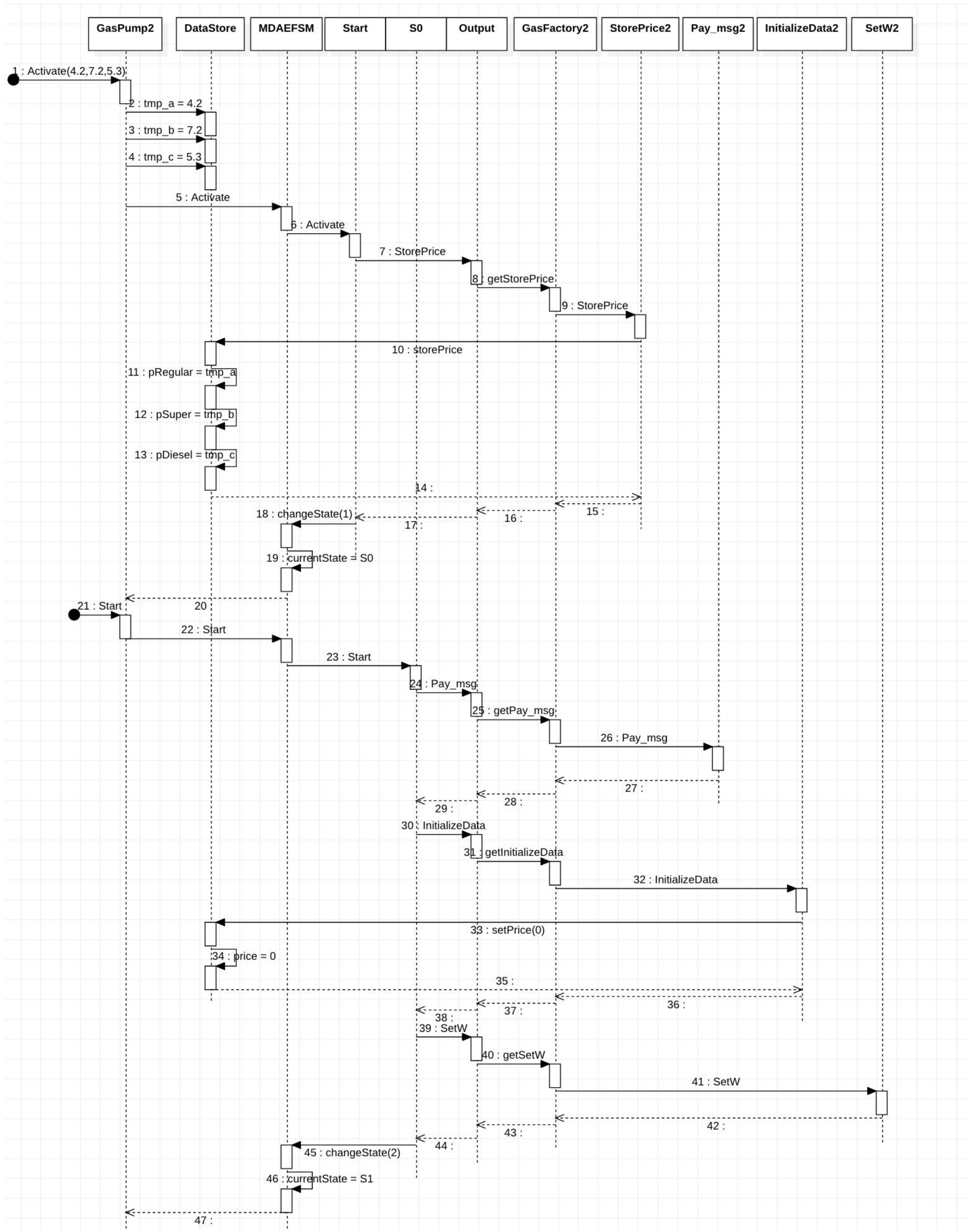
a. Scenario-I : should show how one liter of gas is disposed in *GasPump-1*, i.e., the following sequence of operations is issued: *Activate(4), Start(), PayCash(5), StartPump(), PumpLiter(), PumpLiter()*

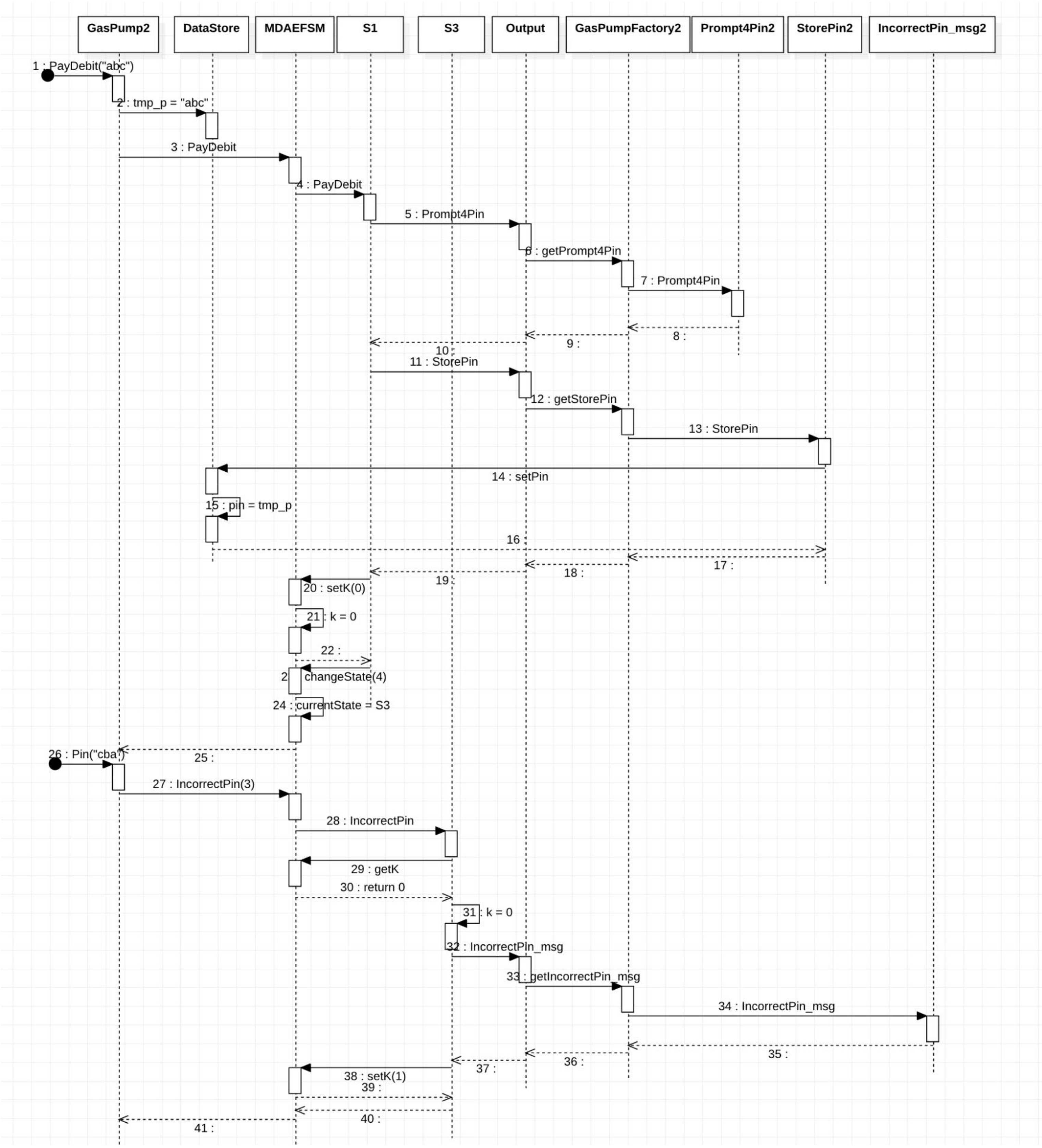


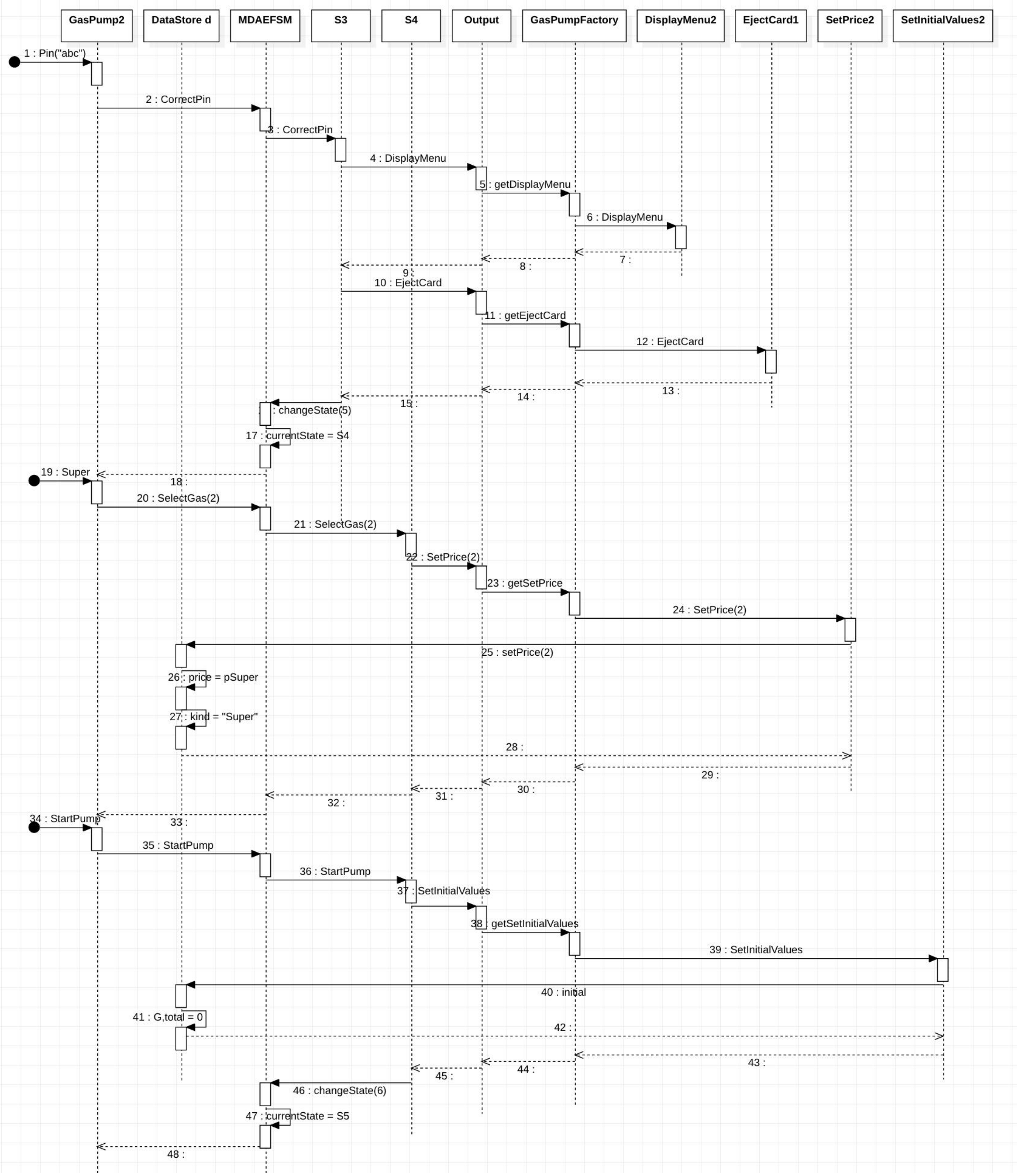

GasPump
DataStore d
MDAEFSM
S1
S4
OutPut
GasPumpFactory
StoreCash1
DisplayMenu1
SetW1
SetInitialValues1
1 : PayCach(5)
2 : d.tmp_c = c
3 : PayCach
4 : PayCash
5 : StoreCash
6 : getStoreCash
7 : StoreCash
8 : setCash
9 : cash = tmp_c
10 :
11 :
12 :
13 :
14 : DisplayMenu
15 : getDisplayMenu
16 : DisplayMenu
17 :
18 :
19 :
20 : SetW(0)
21 : getSetW
22 : SetW(d,0)
23 : setW(0)
24 : w = 0
25 :
26 :
27 :
28 :
: changeState(5)
30 : currentState = S4
31 :
32 : StartPump
33 : StartPump
34 : StartPump
35 : SetInitialValues
36 : getSetInitialValues
37 : SetInitialValues
38 : initial
39 : L.total = 0
40 :
41 :
42 :
43 :
44 : changeState(6)
45 : currentState = S5
46 :

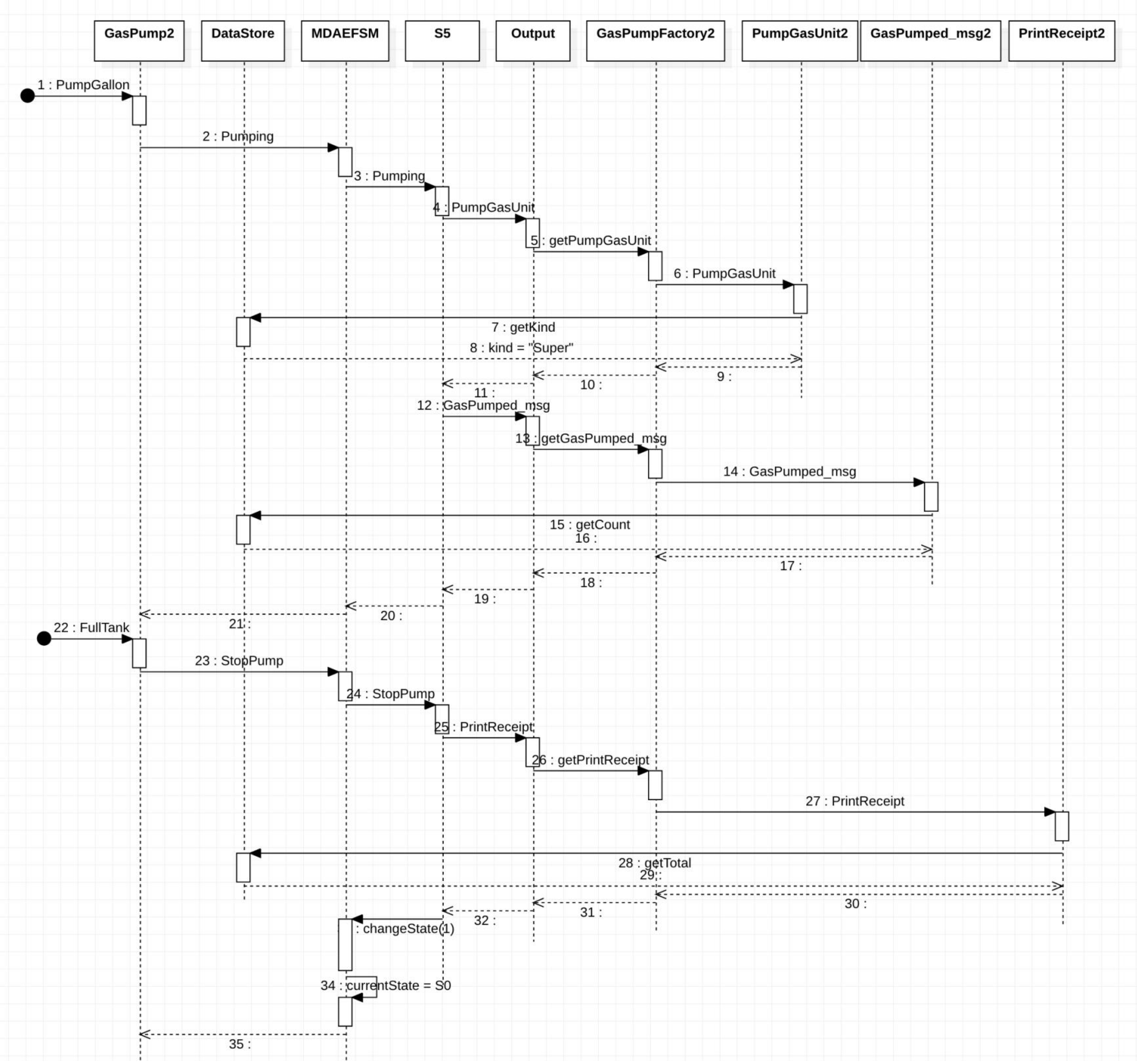

GasPump1
DataStore d
MDAEFSM
S5
Output
GasFactory1
PumpGasUnit1
GasPumped_msg1
PrintReceipt1
1 : PumpLiter
2 : getW
3 : getCash
4 : getPrice
5 : getL
6 :
7 : Pumping
8 : Pumping
9 : PumpGasUnit
10 : getPumpGasUnit
11 : PumpGasUnit
12 :
13 :
14 :
15 : GasPumped_msg
16 : getGasPumped_msg
17 : GasPumped_msg
18 : getCount
19 :
20 :
21 :
22 :
23 :
24 :
25 : PumpLiter
26 : getW
27 : getCash
28 : getPrice
29 : getL
30 :
31 : StopPump
32 : StopPump
33 : PrintReceipt
34 : getPrintReceipt
35 : PrintReceipt
36 : getTotal
37 :
38 :
39 :
40 :
41 : changeState(1)
42 : currentState = S0
43 :

b. 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()*




GasPump2
DataStore
MDAEFSM
S1
S3
Output
GasPumpFactory2
Prompt4Pin2
StorePin2
IncorrectPin_msg2
1 : PayDebit("abc")
2 : tmp_p = "abc"
3 : PayDebit
4 : PayDebit
5 : Prompt4Pin
6 : getPrompt4Pin
7 : Prompt4Pin
8 :
9 :
10 :
11 : StorePin
12 : getStorePin
13 : StorePin
14 : setPin
15 : pin = tmp_p
16 :
17 :
18 :
19 :
20 : setK(0)
21 : k = 0
22 :
2 changeState(4)
24 : currentState = S3
25 :
26 : Pin("cba")
27 : IncorrectPin(3)
28 : IncorrectPin
29 : getK
30 : return 0
31 : k = 0
32 : IncorrectPin_msg
33 : getIncorrectPin_msg
34 : IncorrectPin_msg
35 :
36 :
37 :
38 : setK(1)
39 :
40 :
41 :


GasPump2
DataStore d
MDAEFSM
S3
S4
Output
GasPumpFactory
DisplayMenu2
EjectCard1
SetPrice2
SetInitialValues2
1 : Pin("abc")
2 : CorrectPin
3 : CorrectPin
4 : DisplayMenu
5 : getDisplayMenu
6 : DisplayMenu
7 :
8 :
9 :
10 : EjectCard
11 : getEjectCard
12 : EjectCard
13 :
14 :
15 :
: changeState(5)
17 : currentState = S4
18 :
19 : Super
20 : SelectGas(2)
21 : SelectGas(2)
22 : SetPrice(2)
23 : getSetPrice
24 : SetPrice(2)
25 : setPrice(2)
26 : price = pSuper
27 : kind = "Super"
28 :
29 :
30 :
31 :
32 :
33 :
34 : StartPump
35 : StartPump
36 : StartPump
37 : SetInitialValues
38 : getSetInitialValues
39 : SetInitialValues
40 : initial
41 : G,total = 0
42 :
43 :
44 :
45 :
46 : changeState(6)
47 : currentState = S5
48 :


GasPump2
DataStore
MDAEFSM
S5
Output
GasPumpFactory2
PumpGasUnit2
GasPumped_msg2
PrintReceipt2
1 : PumpGallon
2 : Pumping
3 : Pumping
4 : PumpGasUnit
5 : getPumpGasUnit
6 : PumpGasUnit
7 : getKind
8 : kind = "Super"
9 :
10 :
11 :
12 : GasPumped_msg
13 : getGasPumped_msg
14 : GasPumped_msg
15 : getCount
16 :
17 :
18 :
19 :
20 :
21 :
22 : FullTank
23 : StopPump
24 : StopPump
25 : PrintReceipt
26 : getPrintReceipt
27 : PrintReceipt
28 : getTotal
29 :
30 :
31 :
32 :
: changeState(1)
34 : currentState = S0
35 :