SSA Project

Name – Amin Mithil Paragbhai

CWID – A20386345

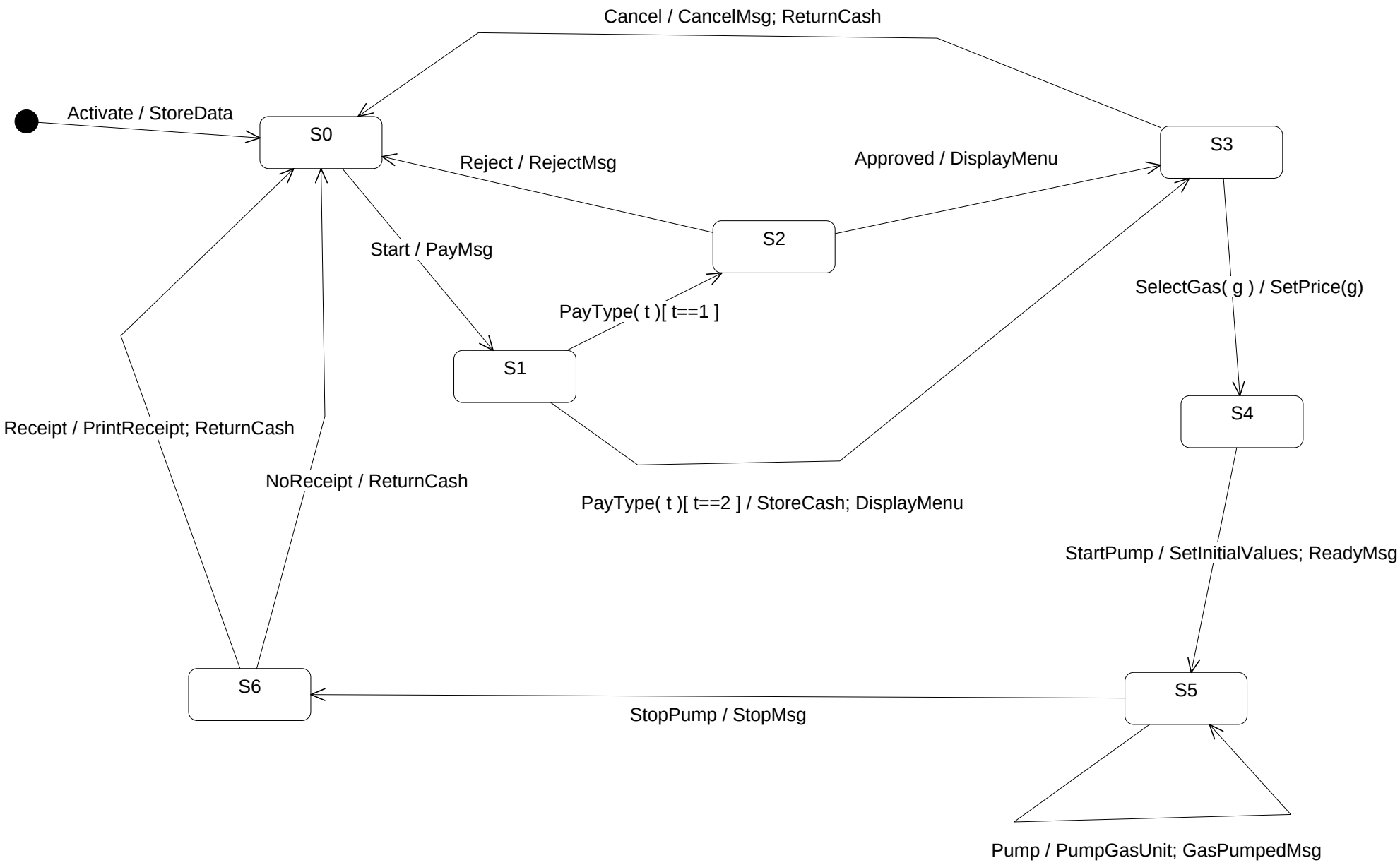
MDA-EFSM Gas Pump

MDA-EFSM Events:

Activate()
Start()
PayType(int t) //credit: t=1; cash: t=2
Reject()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int g)
Receipt()
NoReceipt()

MDA-EFSM Actions:

StoreData	// 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
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



MDA-EFSM for Gas Pumps

Operations of the Input Processor
(GasPump-1)

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

Start() {
        m->Start();
}

PayCredit() {
        m->PayType(1);
}

Reject() {
        m->Reject();
}

Cancel() {
        m->Cancel();
}

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

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

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

StartPump() {
        m->StartPump();
}

PumpGallon() {
        m->Pump();

StopPump() {
        m->StopPump();
        m->Receipt();
}
```

Notice:
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)

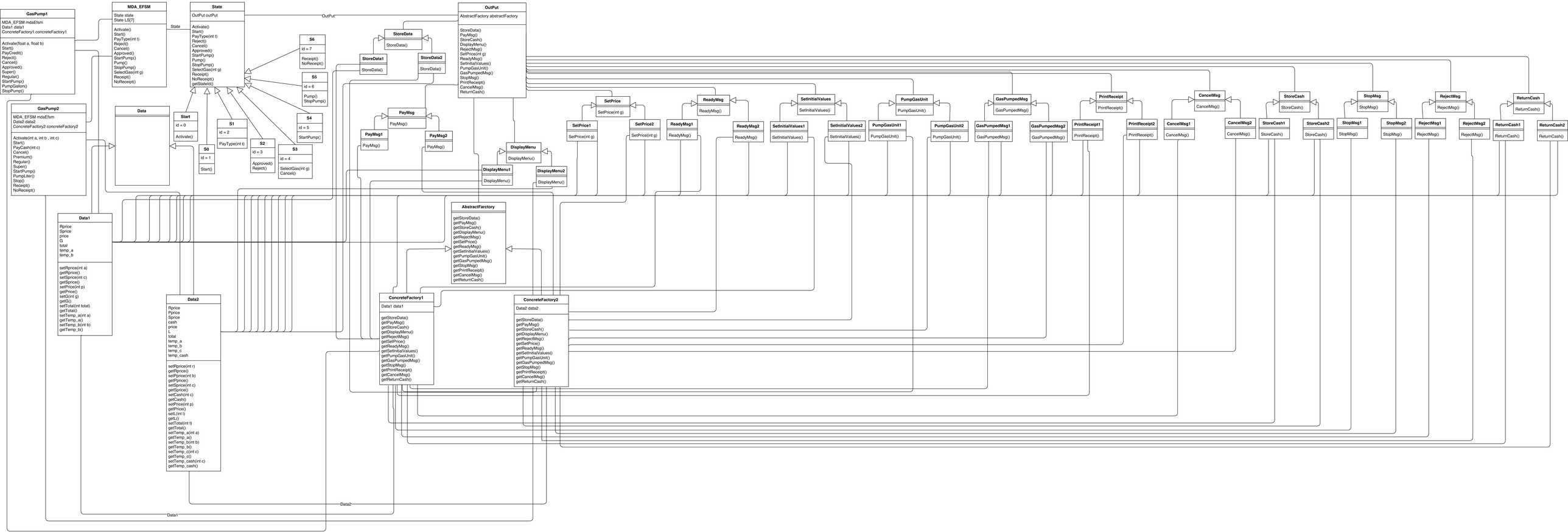
```
Activate(int a, int b, int c) {
        if ((a>0)&&(b>0)&&(c>0)) {
            d->temp_a=a;
            d->temp_b=b;
            d->temp_c=c
            m->Activate()
        }
}

Start() {
        m->Start();
}

PayCash(float c) {
        if (c>0) {
            d->temp_cash=c;
            m->PayType(2)
        }
}

Cancel() {
        m->Cancel();
}

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

Premium() {
        m->SelectGas(3);
}

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

```
StartPump() {
        m->StartPump();
}

PumpLiter() {
        if (d->cash<(d->L+1)*d->price)
            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
L: contains the number of liters already pumped

cash , *L*, *price* are in the data store
m: is a pointer to the MDA-EFSM object
d: is a pointer to the Data Store object

Part -2 Class Diagram



Part -3 Description of class and its responsibilities

1. Class GasPump1
   - This class represents a specific method to implement gas pump
   - This class has an implementation of payment method credit card
     - Variable/Pointers
       i. MDA_EFSM mdaEfsm
       ii. Data1 data1
     - Methods
       1. Activate() – this method takes an input of Regular an Super gas price and call Activate method of MDA_EFSM class
       2. Start() – this method start the pump and call the Start method of MDA_EFSM class
       3. PayCredit() – this method call PayType method of MDA_EFSM class with the parameter 1
       4. Reject() – this method call Reject method of MDA_EFSM class
       5. Cancel() – this method call Cancel method of MDA_EFSM class
       6. Approved() – this method call Approved method of MDA_EFSM class
       7. Super() – this method call SelectGas method of MDA_EFSM class with the parameter 2
       8. Regular() – this method call SelectGas method of MDA_EFSM class with the parameter 1
       9. StartPump() – this method call StartPump method of MDA_EFSM class
       10. PumpGallon() – this method call Pump method of MDA_EFSM class
       11. StopPump() – this method call StopPump and Receipt method of MDA_EFSM class

2. Class GasPump2
   - This class represent a specific method to implement gas pump
   - This class has an implementation of payment method cash
     - Variables/Pointers
       i. MDA_EFSM mdaEfsm
       ii. Data2 data2
     - Methods
       1. Activate() – this method takes an input of regular, super and premium gas price and call Activate method of MDA_EFSM class
       2. Start() – this method start the pump and call Start method of MDA_EFSM class
       3. PayCash(int c) – this method call PayType method of MDA_EFSM class with the parameter 2
       4. Cancel() – this method call Cancel method of MDA_EFSM class

5. Premium() – this method call SelectGas method of MDA_EFSM class with parameter 3
6. Regular() – this method call SelectGas method of MDA_EFSM class with parameter 1
7. Super() – this method call SelectGas method of MDA_EFSM class with the parameter 2
8. StartPump() – this method call StartPump method of MDA_EFSM class
9. PumpLiter() – this method will check that is there enough cash to fill 1 more liter of gas. If has enough cash then call Pump otherwise call StopPump method of MDA_EFSM class
10. Stop() – this method call StopPump method of MDA_EFSM class
11. Receipt() – this method call Receipt method of MDA_EFSM class
12. NoReceipt() – this method call NoReceipt method of MDA_EFSM class

3. Class MDA_EFSM
   - This class is responsible for the change of state. In this example I use centralize design pattern so this class will get the state id from the state class and then change the state according the current state and operation.
   - This class calls all the operation of platform independent module.
   - Calls each operations in their respective states.
     - Variables/Pointers
       i. State state – this pointer points to the current state
       ii. State[] ls – this is the list of states
     - Methods
       1. Activate() – calls Activate method of Start state and change state to S0
       2. Start() – calls Start method of S0 state and change state to S1
       3. PayType(int t) – calls PayType of method S1 state with the parameter of t
          If t==1 then change state to S2
          If t==2 then change state to S3
       4. Reject() – calls Reject method of S2 state and change state to S0
       5. Cancel() – calls Cancel method of S3 state and change state to S0
       6. Approved() – calls Approved method of S2 state and change state to S3
       7. SelectGas(int g) – calls SelectGas method of S3 state with parameter g and change state to S4
       8. StartPump() – calls StartPump method of S4 state ad change state to S5
       9. Pump() – calls Pump method of S5 state and no change of state
       10. StopPump() – calls StopPump method of S5 state and change state to S6
       11. Receipt() – calls Receipt method of S6 state and change state to S0
       12. NoReceipt() – calls NoReceipt method of S6 and change state to S0

4. Class State
   - This is an abstract class of the state design pattern
   - This is the parent class of all state classes
     - Variables/Pointers
       i. OutPut outPut
     - Methods
       1. Activate() – this is an abstract method
       2. Start() – this is an abstract method
       3. PayType(int t) – this is an abstract method
       4. Reject() – this is an abstract method
       5. Cancel() – this is an abstract method
       6. Approved() – this is an abstract method
       7. StartPump() – this is an abstract method
       8. Pump() – this is an abstract method
       9. StopPump() – this is an abstract method
       10. SelectGas() – this is an abstract method
       11. Receipt() – this is an abstract method
       12. NoReceipt() – this is an abstract method
       13. getStateId() – this is an abstract method

5. Class Start
   - This is a class of start state and child class of State class
   - In start class only Activate method can be perform
     - Variables/Pointers
       i. id = 0 – represent state id
     - Methods
       1. Activate() – this method call StoreData() of OutPut class
       2. getStateId() – return value of id

6. Class S0
   - This is a class of S0 state and child class of State class
   - In S0 class only Start() method can be perform
     - Variables/Pointers
       i. id = 1 – represent state id
     - Methods
       1. Start() – this method call PayMsg() of OutPut class
       2. getStateId() – return value of id

7. Class S1
   - This is a class of S1 state and child class of State class
   - In S1 class only PayType(int t) method can be perform

- Variables/Pointers
    i. id = 2 – represent state id
- Methods
    1. PayType(int t) – if payment type is credit then no event is called and if payment type is cash then call StoreCash and DisplayMenu of OutPut class
    2. getStateId() – return value of id

8. Class S2
- This is a class of S2 state and child class of State class
- In S2 class only Reject() and Approved() method can be perform
    - Variables/Pointers
        i. id = 3 – represent state id
    - Methods
        1. Reject() – this method call RejectMsg() of OutPut class
        2. Approved() – this method call DisplayMenu() of OutPut class
        3. getStateId() – return value of id

9. Class S3
- This is a class of S3 state and child class of State class
- In S3 class only Cancel() and SelectGas(int g) method can be performed
    - Variables/Pointers
        i. id = 4 – represent state id
    - Methods
        1. Cancel() – this method call CancelMsg() and ReturnCash() method of OutPut class
        2. SelectGas(int g) – this method call SetPrice(g) method of OutPut class
        3. getStateId() – return value of id

10. Class S4
- This is a class of S4 state and child class of State class
- In S4 class only StartPump() method can be performed
    - Variables/Pointers
        i. id = 5 – represent state id
    - Methods
        1. StartPump() – this method call SetInitialValues() and ReadyMsg() method of OutPut class
        2. getStateId() – return value of id

11. Class S5
- This is a class of S5 state and child class of State class
- In S5 class only Pump() and StopPump() method can be performed

- Variables/Pointers
    i. id = 6 – represent state id
- Methods
    1. Pump() – this method call PumpGasUnit() and GasPumpedMsg() method of OutPut class
    2. StopPump() – this method call StopMsg() method of OutPut class
    3. getStateId() – return value of id

12. Class S6
- This is a class of S6 state and child class of state class
- In S6 class only Receipt() and NoReceipt() method can be performed
    - Variables/Pointers
        i. id = 7 – represent state id
    - Methods
        1. Receipt() – this method call PrintReceipt() and ReturnCash() method of OutPut class
        2. NoReceipt() – this method call ReturnCash() method of OutPut class
        3. getStateId() – return value of id

13. Class OutPut
- This class has implementation of all events
- This class has an object of all strategy classes. The events are going to call by the strategy class objects.
    - Methods
        1. StoreData() – call StoreData() method of StoreData class
        2. PayMsg() – call PayMsg() method of PayMsg class
        3. StoreCash() – call StoreCash() method of StoreCash class
        4. DisplayMenu() – call DisplayMenu() method of DisplayMenu class
        5. RejectMsg() – call RejectMsg() method of RejectMsg class
        6. SetPrice() – call SetPrice() method of SetPrice class
        7. ReadyMsg() – call ReadyMsg() method of ReadyMsg class
        8. SetInitialValues () – call SetInitialValues() method of SetInitialValues class
        9. PumpGasUnit() – call PumpGasUnit() method of PumpGasUnit class
        10. GasPumpedMsg() – call GasPumpedMsg() method of GasPumpedMsg class
        11. StopMsg () – call StopMsg() method of StopMsg class
        12. PrintReceipt() – call PrintReceipt() method of PrintReceipt class
        13. CancelMsg() – call CancelMsg() method of CancelMsg class
        14. ReturnCash() – call ReturnCash() method of ReturnCash class

14. Class AbstractFactory

- This is an abstract class of AbstractFactory Design pattern
- This call declares all the methods which are implemented in the abstract factory design pattern
- All these methods return object based on their Factory
  - Methods
    1. getStoreData() – this is an abstract method
    2. getPayMsg() – this is an abstract method
    3. getStoreCash() – this is an abstract method
    4. getDisplayMenu() – this is an abstract method
    5. getRejectMsg() – this is an abstract method
    6. getSetPrice() – this is an abstract method
    7. getReadyMsg() – this is an abstract method
    8. getSetInitialValues () – this is an abstract method
    9. getPumpGasUnit() – this is an abstract method
    10. getGasPumpedMsg() – this is an abstract method
    11. getStopMsg() – this is an abstract method
    12. getPrintReceipt() – this is an abstract method
    13. getCancelMsg() – this is an abstract method
    14. getReturnCash() – this is an abstract method

15. Class ConcreteFactory1

- This class is a child class of the AbstractFactory design pattern
- This class is responsible to create objects of all the events which are related to GasPump1
  - Variables/Pointers
    i. Data1 data1
  - Methods
    1. getStoreData() – return StoreData1 instance
    2. getPayMsg() – return PayMsg1 instance
    3. getStoreCash() – return StoreCash1 instance
    4. getDisplayMenu() – return DisplayMenu1 instance
    5. getRejectMsg() – return RejectMsg1 instance
    6. getSetPrice() – return SetPrice1 instance
    7. getReadyMsg() – return ReadyMsg1 instance
    8. getSetInitialValues () – return SetInitialValues1 instance
    9. getPumpGasUnit() – return PumpGasUnit1 instance
    10. getGasPumpedMsg() – return GasPumpedMsg1 instance
    11. getStopMsg() – return StopMsg1 instance
    12. getPrintReceipt() – return PrintReceipt1 instance
    13. getCancelMsg() – return CancelMSg1 instance
    14. getReturnCash() – return ReturnCash1 instance

16. Class ConcreteFactory2
    - This class is a child class of the AbstractFactory design pattern
    - This class is responsible to create objects of all the events which are related to GasPump2
        - Variables/Pointers
            i. Data2 data2
        - Methods
            1. getStoreData() – return StoreData2 instance
            2. getPayMsg() – return PayMsg2 instance
            3. getStoreCash() – return StoreCash2 instance
            4. getDisplayMenu() – return DisplayMenu2 instance
            5. getRejectMsg() – return RejectMsg2 instance
            6. getSetPrice() – return SetPrice2 instance
            7. getReadyMsg() – return ReadyMsg2 instance
            8. getSetInitialValues () – return SetInitialValues2 instance
            9. getPumpGasUnit() – return PumpGasUnit2 instance
            10. getGasPumpedMsg() – return GasPumpedMsg2 instance
            11. getStopMsg() – return StopMsg2 instance
            12. getPrintReceipt() – return PrintReceipt2 instance
            13. getCancelMsg() – return CancelMSg2 instance
            14. getReturnCash() – return ReturnCash2 instance

17. Class Data
    - This is an abstract class for the Data store

18. Class Data1
    - This class is used to store and get the data related to GasPump1
        - Variables/Pointer
            i. Float Rprice
            ii. Float Sprice
            iii. Float price
            iv. Float total
            v. int G
            vi. float temp_a
            vii. float temp_b
        - Methods
            1. getRprice() – return Rprice
            2. setRprice(float rprice) – set Rprice
            3. getSprice() – return Sprice
            4. setSprice(float sprice) – set Sprice
            5. getPrice() – return price
            6. setPrice(float price) – set price
            7. getTotal() – return total

8. setTotal(float total) – set total
9. getG() – return G
10. setG(int g) – set G
11. getTemp_a() – return temp_a
12. setTemp_a(float tempa) – set temp_a
13. getTemp_b() – return temp_b
14. setTemp_b(float tempa) – set temp_b

19. Class Data2

- This class is used to store and get the data related to GasPump2
  - Variables/Pointers
    i. Int Rprice
    ii. Int Pprice
    iii. Int Sprice
    iv. Int cash
    v. Int price
    vi. Int L
    vii. Int total
    viii. Int temp_a
    ix. Int temp_b
    x. Int temp_c
    xi. Int temp_cash
  - Methods
    1. getRprice() – return Rprice
    2. setRprice(int rprice) – set Rprice
    3. getPprice() – return Pprice
    4. setPprice(int pprice) – set Pprice
    5. getSprice() – return Sprice
    6. setSprice(int sprice) – set Sprice
    7. getcash() – return cash
    8. setCash(int cash) – set cash
    9. getprice() – return price
    10. setprice(int price) – set price
    11. getL() – return L
    12. setL(int l) – set L
    13. getTotal() – return total
    14. setTotal(int total) – set total
    15. getTemp_a() – return temp_a
    16. setTemp_a(int temp_a) – set temp_a
    17. getTemp_b() – return temp_b
    18. setTemp_b(int temp_b) – set temp_b
    19. getTemp_c() – return temp_c
    20. setTemp_c(int temp_c) – set temp_c

21. getTemp_cash() – return temp_cash
22. setTemp_cash(int temp_cash) – set temp_cash

20. Class StoreData
- This is an abstract class
- It is a part of StrategyPattern
  - Methods
  1. StoreData() – this is an abstract method

21. Class StoreData1
- This is a child class of StoreData
  - Methods
  1. StoreData() – get value of temp_a and temp_b and set it to a Rprice and Sprice

22. Class StoreData2
- This is a child class of StoreData
  - Methods
  1. StoreData() – get value of temp_a, temp_b, temp_c and set it to a Rprice, Pprice and Sprice

23. Class PayMsg
- This is an abstract class
- It is a part of StrategyPattern
  - Methods
  1. PayMsg() – this is an abstract method

24. Class PayMsg1
- This is a child class of PayMsg
  - Methods
  1. PayMsg() – display Pay Msg

25. Class PayMsg2
- This is a child class of PayMsg
  - Methods
  1. PayMsg() – display Pay Msg

26. Class StoreCash
- This is an abstract class
- It is a part of StrategyPattern
  - Methods
  1. StoreCash() – this is an abstract method

27. Class StoreCash1
    - This is a child class of StoreCash
        - Methods
        1. StoreCash() – no event for this method

28. Class StoreCash2
    - This is a child class of StoreCash2
        - Methods
        1. StoreCash() – get temp_cash value and store it into cash value from Data2 class

29. Class DisplayMenu
    - This is an abstract class
    - It is a part of StrategyPattern
        - Methods
        1. DisplayMenu() – this is an abstract method

30. Class DisplayMenu1
    - This is a child class of DisplayMenu
        - Methods
        1. DisplayMenu() – display available gas type

31. Class DisplayMenu2
    - This is a child class of DisplayMenu
        - Methods
        1. DisplayMenu() – display available gas type

32. Class RejectMsg
    - This is an abstract class
    - It is a part of StrategyPattern
        - Methods
        1. RejectMsg() – this is an abstract method

33. Class RejectMsg1
    - This is a child class of RejectMsg
        - Methods
        1. RejectMsg() – display Reject Message for credit card

34. Class RejectMsg2
    - This is a child class of RejectMsg
        - Methods
        1. RejectMsg() – No Events in this method

35. Class SetPrice
    - This is an abstract class
    - It is a part of StrategyPattern
        - Methods
        1. SetPrice() – this is an abstract method

36. Class SetPrice1
    - This is a child class of SetPrice
        - Methods
        1. SetPrice() – get selected gas price and set it to a price in Data1

37. Class SetPrice2
    - This is a child class of SetPrice
        - Methods
        1. SetPrice() – get selected gas price and set it to a price in Data2

38. Class ReadyMsg
    - This is an abstract class
    - It is a part of StrategyPattern
        - Methods
        1. ReadyMsg() – this is an abstract method

39. Class ReadyMsg1
    - This is a child class of ReadyMsg
        - Methods
        1. ReadyMsg() – display ready pumping message and display selected gas price

40. Class ReadyMsg2
    - This is a child class of ReadyMsg
        - Methods
        1. ReadyMsg() – display ready for pumping message and display selected gas price

41. Class SetInitialValues
    - This is an abstract class
    - It is a part of StrategyPattern
        - Methods

1. SetInitialValues() – this is an abstract method

42. Class SetInitialValues1
- This is a child class of SetInitialValues
  - Methods
  1. SetInitialValues() – set G and total to 0

43. Class SetInitialValues2
- This is a child class of SetInitialValues
  - Methods
  1. SetInitialValues() – set L and total to 0

44. Class PumpGasUnit
- This is an abstract class
- It is a part of StrategyPattern
  - Methods
  1. PumpGasUnit() – this is an abstract method

45. Class PUmpGasUnit1
- This is a child class of PumpGasUnit
  - Methods
  1. PumpGasUnit() – increase the value of G by 1 and calculate total

46. Class PumpGasUnit2
- This is a child class of PumpGasUnit
  - Methods
  1. PumpGasUnit() – increase the value of L by 1 and calculate total

47. Class GasPumpedMsg
- This is an abstract class
- It is a part of StrategyPattern
  - Methods
  1. GasPumpedMsg() – this is an abstract method

48. Class GasPumpedMsg1
- This is a child class of GasPumpedMsg
  - Methods
  1. GasPumpedMsg() – Display value of G

49. Class GasPumpedMsg2
    - This is a child class of GasPumpedMsg
        - Methods
        1. GasPumpedMsg() – Display value of L

50. Class StopMsg
    - This is an abstract class
    - It is a part of StrategyPattern
        - Methods
        1. StopMsg() – this is an abstract method

51. Class StopMsg1
    - This is a child class of StopMsg
        - Methods
        1. StopMsg() – display stop pump message

52. Class StopMsg2
    - This is a child class of StopMsg
        - Methods
        1. StopMsg() – display stop pump message

53. Class PrintReceipt
    - This is an abstract class
    - It is a part of StrategyPattern
        - Methods
        1. PrintReceipt() – this is an abstract method

54. Class PrintReceipt1
    - This is a child class of PrintReceipt
        - Methods
        1. PrintReceipt() – display total poured gas in gallons and display the total price

55. Class PrintReceipt2
    - This is a child class of PrintReceipt
        - Methods
        1. PrintReceipt() – display total poured gas in liters and display the total price and display entered cash

56. Class CancelMsg
    - This is an abstract class
    - It is a part of StrategyPattern

- Methods
1. CancelMsg() – this is an abstract method

57. Class CancelMsg1
- This is a child class of CancelMsg
  - Methods
  1. CancelMsg1() – display a cancellation message

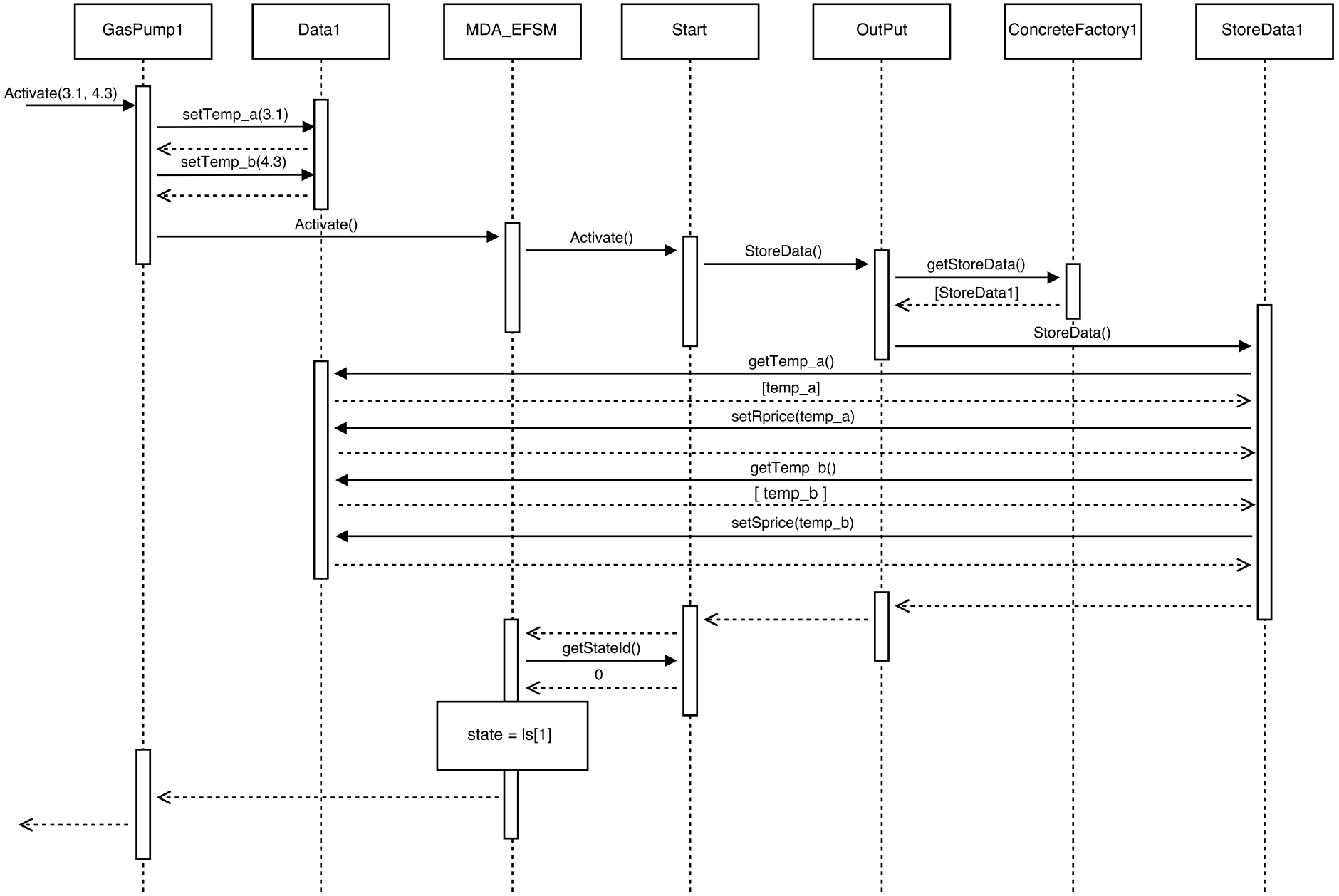
58. Class CancelMsg2
- This is a child class of CancelMsg
  - Methods
  1. CancelMsg() – display a cancellation message

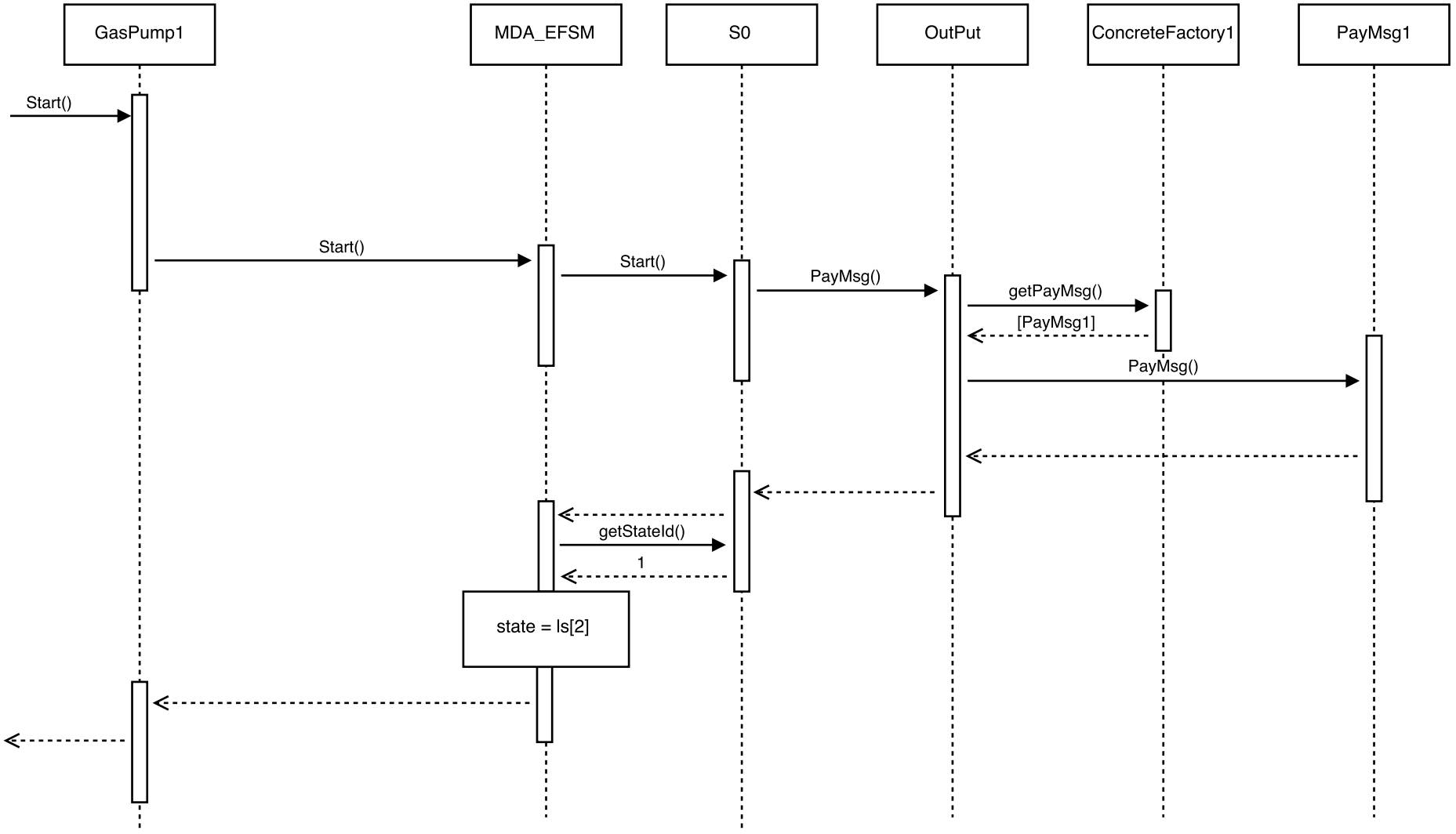
59. Class ReturnCash
- This is an abstract class
- It is a part of StrategyPattern
  - Methods
  1. ReturnCash() – this is an abstract method

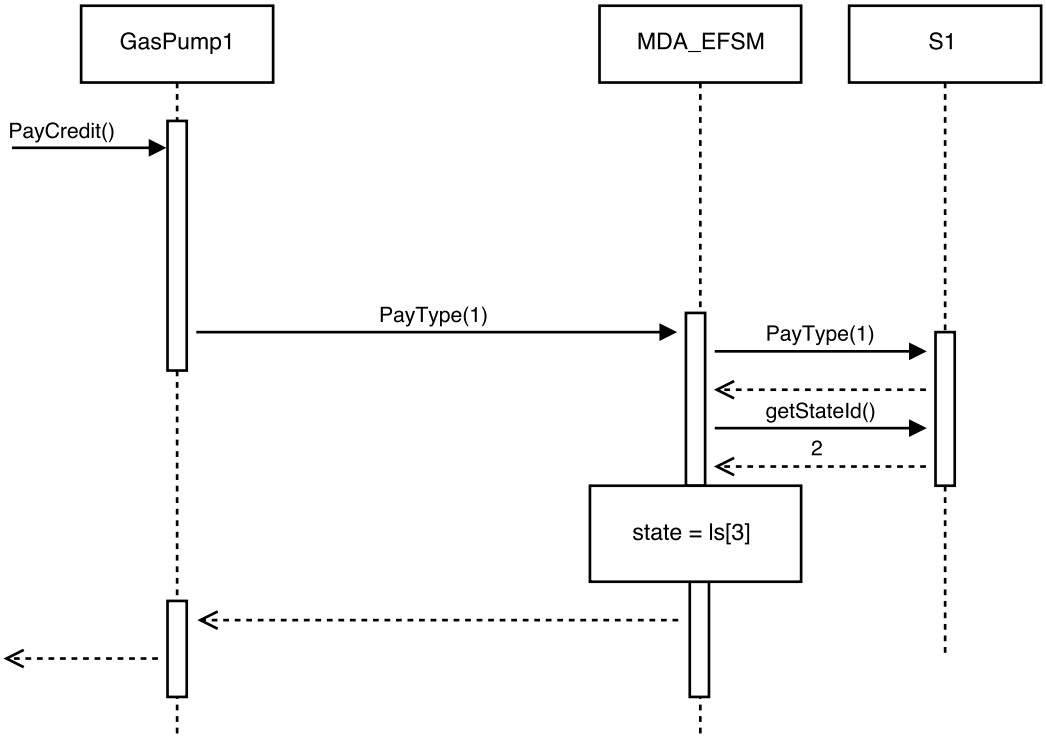
60. Class ReturnCash1
- This is a child class of ReturnCash
  - Methods
  1. ReturnCash() – no event for gas pump 1

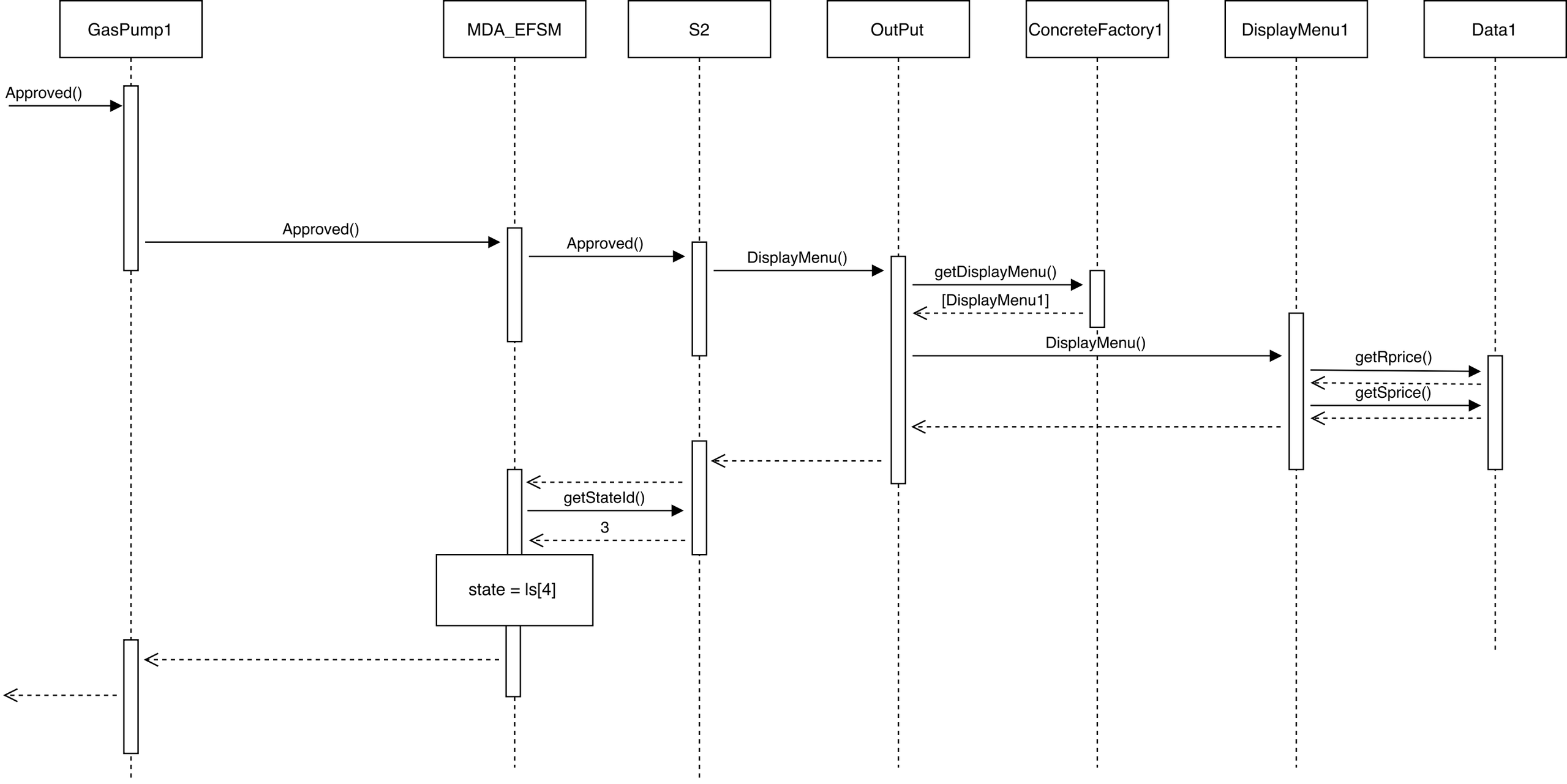
61. Class ReturnCash2
- This is a child class of ReturnCash
  - Methods
  1. ReturnCash() – display remaining cash

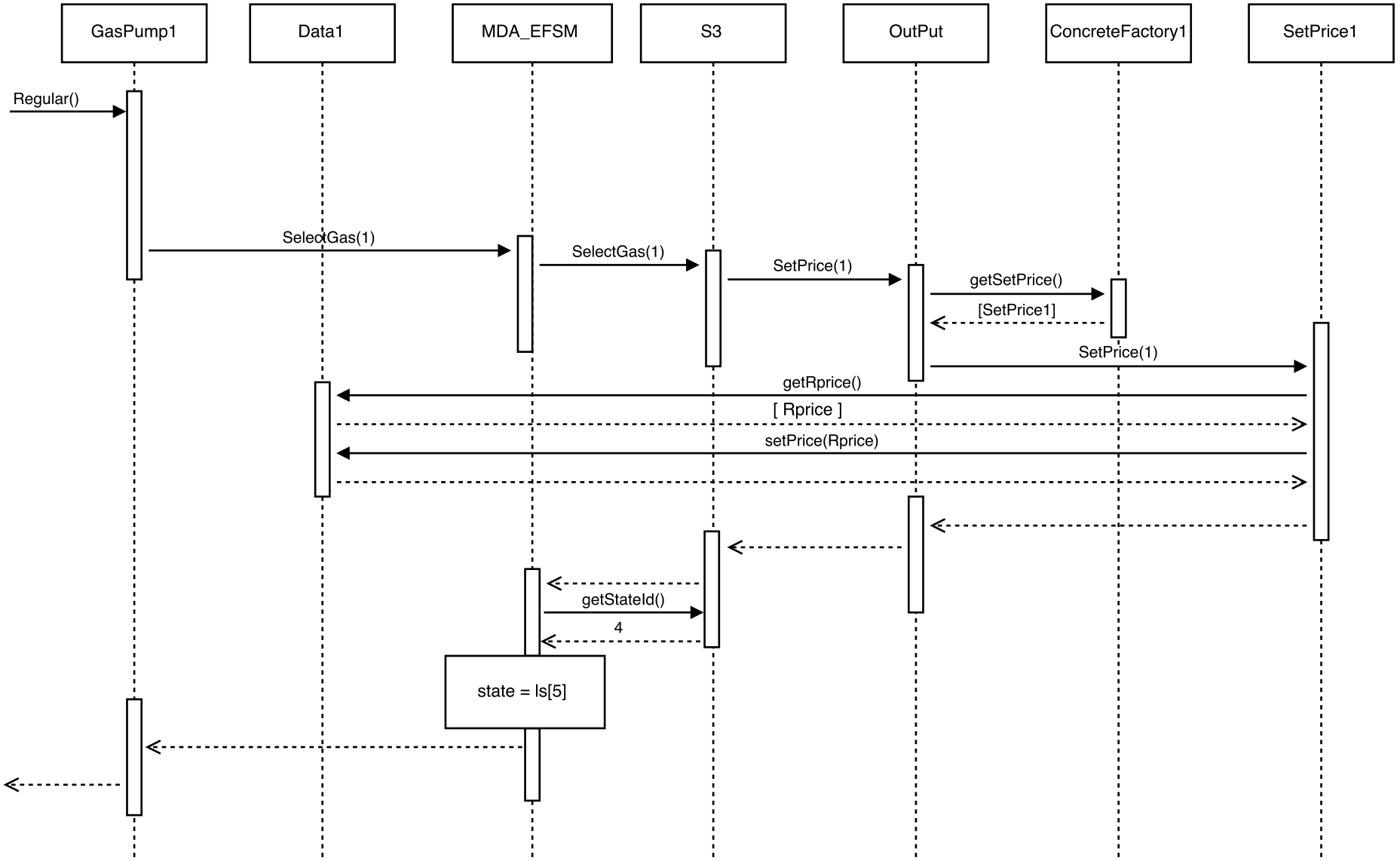
Part - 4 Sequence Diagram [A]

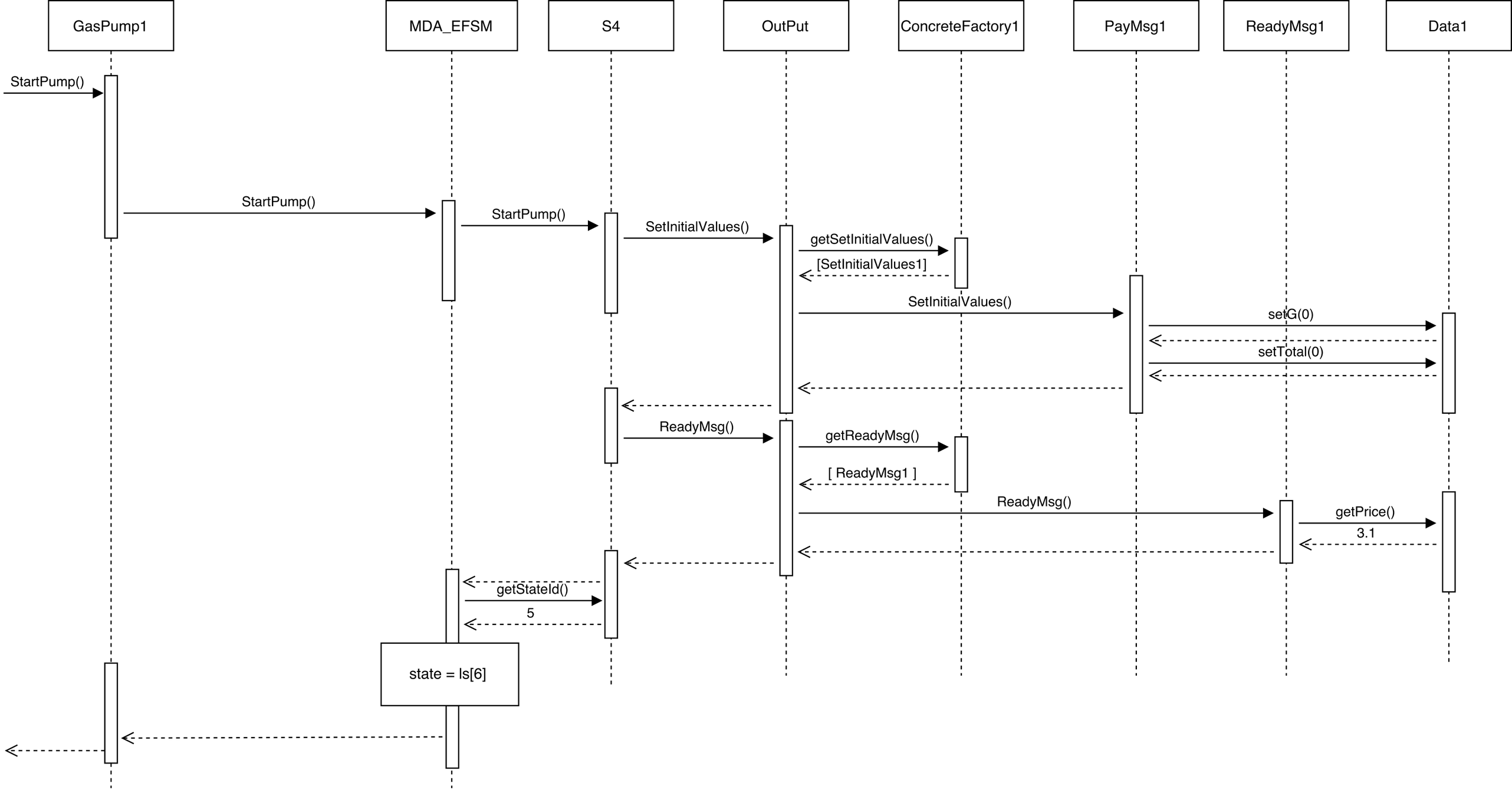


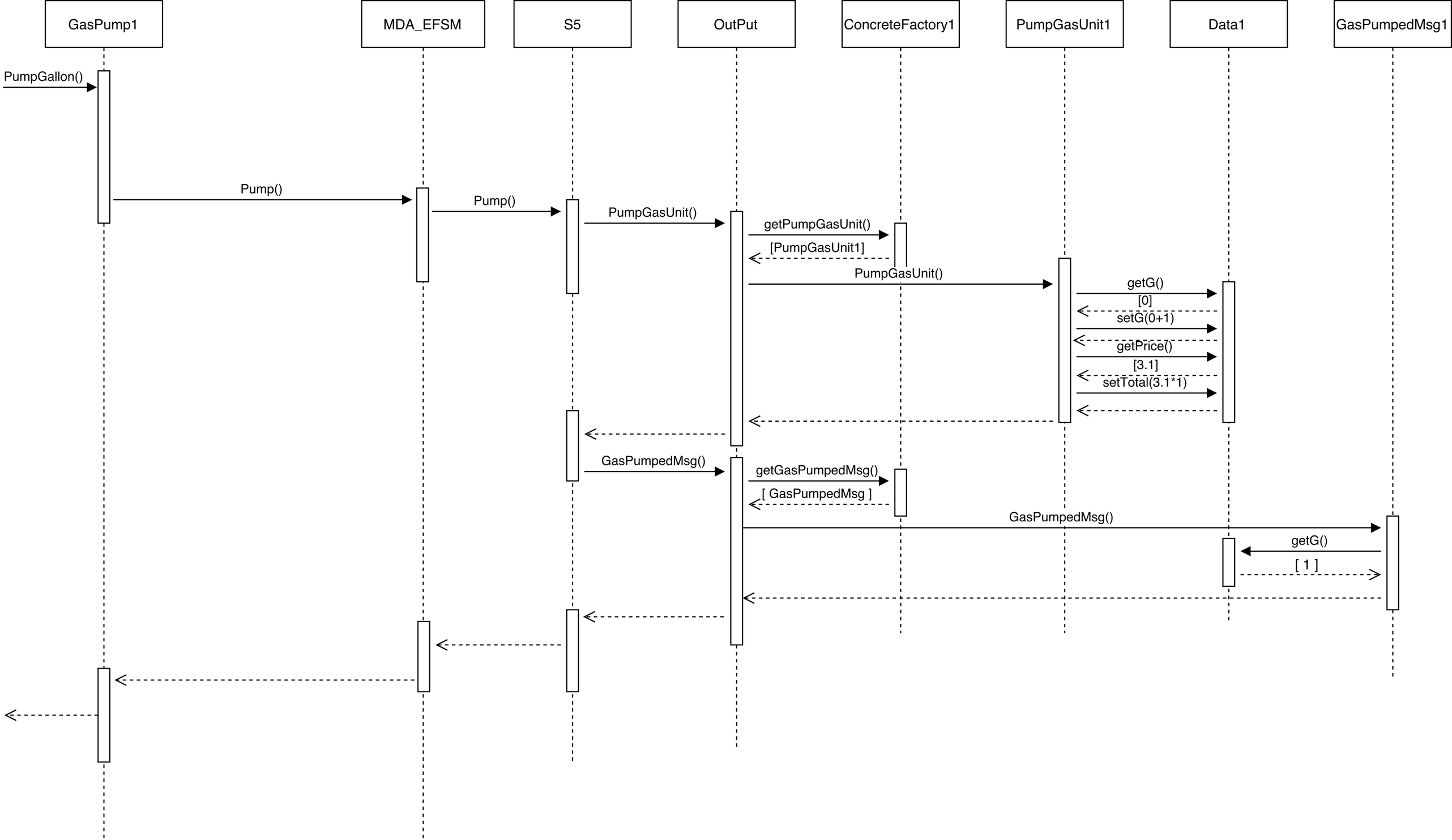

GasPump1
MDA_EFSM
S0
OutPut
ConcreteFactory1
PayMsg1
Start()
Start()
Start()
PayMsg()
getPayMsg()
[PayMsg1]
PayMsg()
getStateId()
1
state = ls[2]

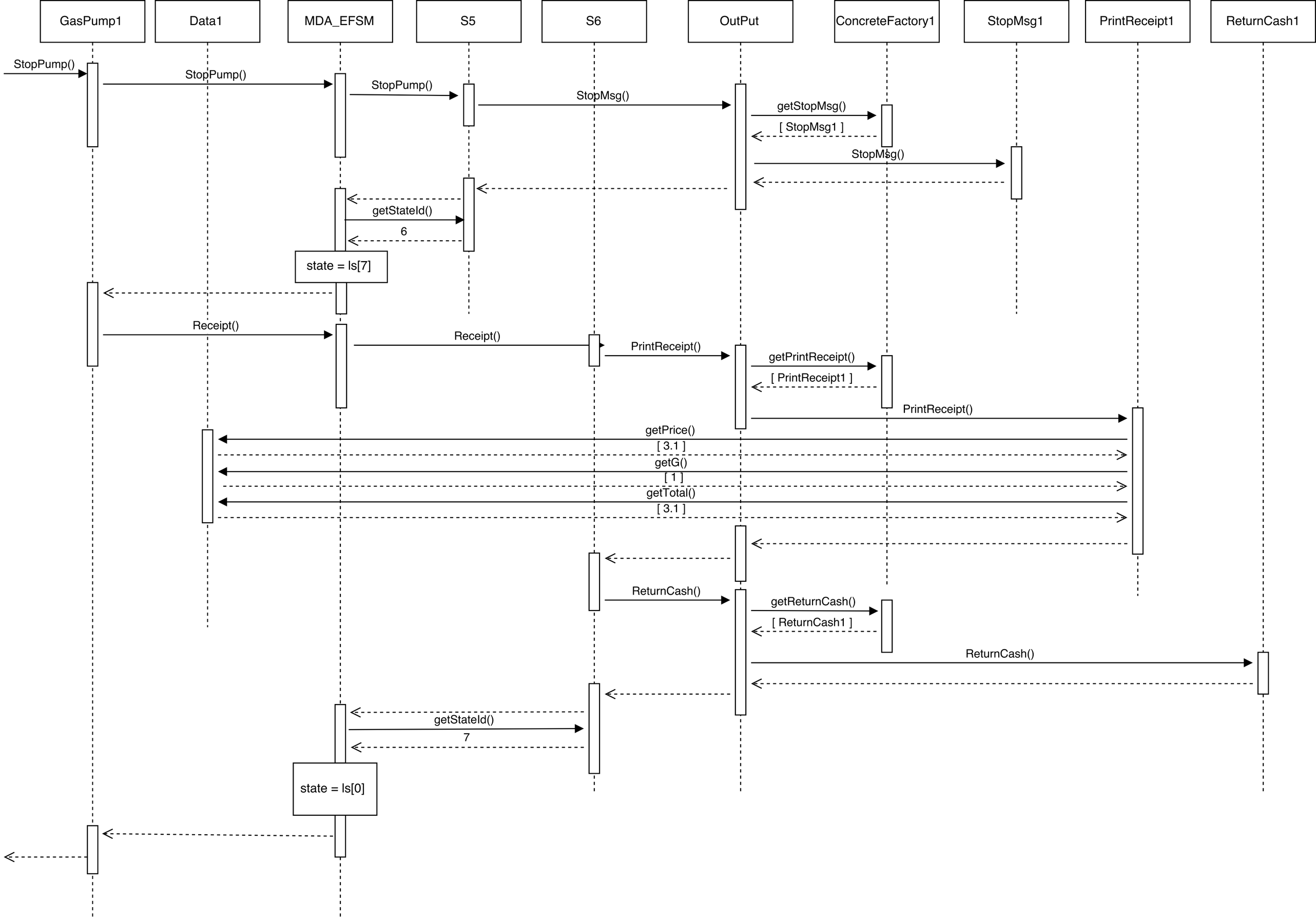

GasPump1
MDA_EFSM
S1
PayCredit()
PayType(1)
PayType(1)
getStateId()
2
state = ls[3]


GasPump1
MDA_EFSM
S2
OutPut
ConcreteFactory1
DisplayMenu1
Data1
Approved()
Approved()
Approved()
DisplayMenu()
getDisplayMenu()
[DisplayMenu1]
DisplayMenu()
getRprice()
getSprice()
getStateId()
3
state = ls[4]

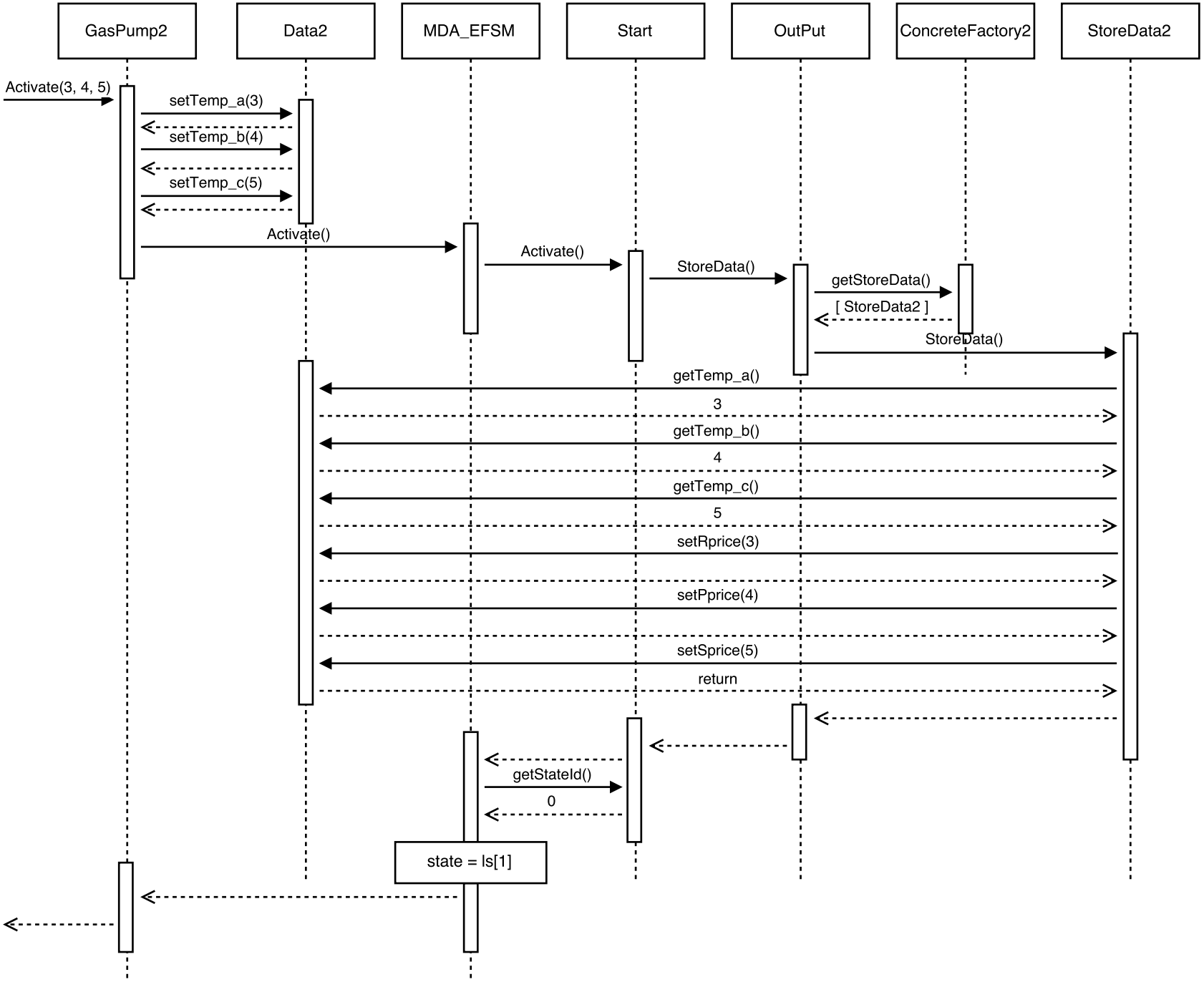

GasPump1
Data1
MDA_EFSM
S3
OutPut
ConcreteFactory1
SetPrice1
Regular()
SelectGas(1)
SelectGas(1)
SetPrice(1)
getSetPrice()
[SetPrice1]
SetPrice(1)
getRprice()
[Rprice]
setPrice(Rprice)
getStateId()
4
state = ls[5]

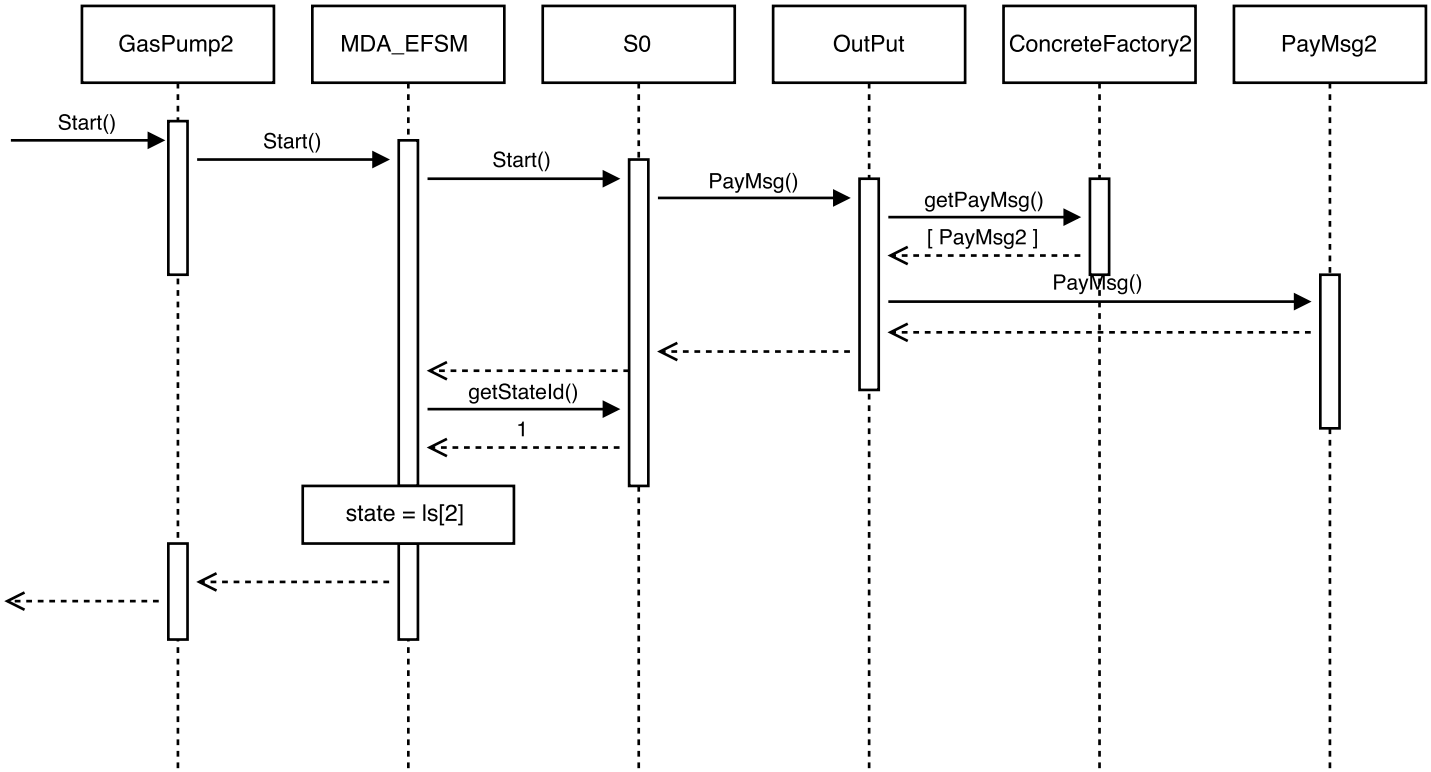

GasPump1
MDA_EFSM
S4
OutPut
ConcreteFactory1
PayMsg1
ReadyMsg1
Data1
StartPump()
StartPump()
StartPump()
SetInitialValues()
getSetInitialValues()
[SetInitialValues1]
SetInitialValues()
setG(0)
setTotal(0)
ReadyMsg()
getReadyMsg()
[ReadyMsg1]
ReadyMsg()
getPrice()
3.1
getStateId()
5
state = ls[6]

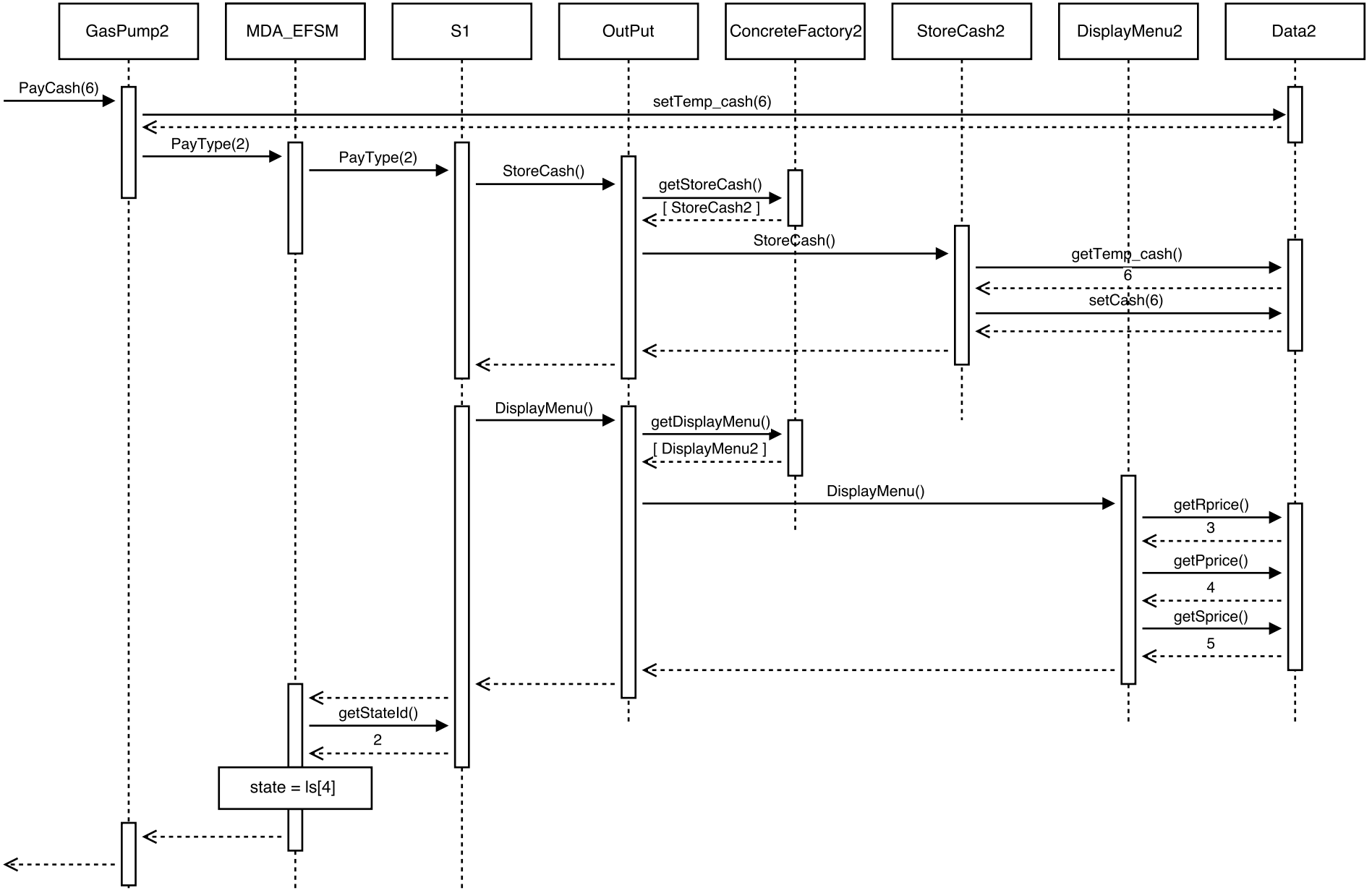

GasPump1
MDA_EFSM
S5
OutPut
ConcreteFactory1
PumpGasUnit1
Data1
GasPumpedMsg1
PumpGallon()
Pump()
Pump()
PumpGasUnit()
getPumpGasUnit()
[PumpGasUnit1]
PumpGasUnit()
getG()
[0]
setG(0+1)
getPrice()
[3.1]
setTotal(3.1*1)
GasPumpedMsg()
GasPumpedMsg()
getGasPumpedMsg()
[GasPumpedMsg]
GasPumpedMsg()
getG()
[1]

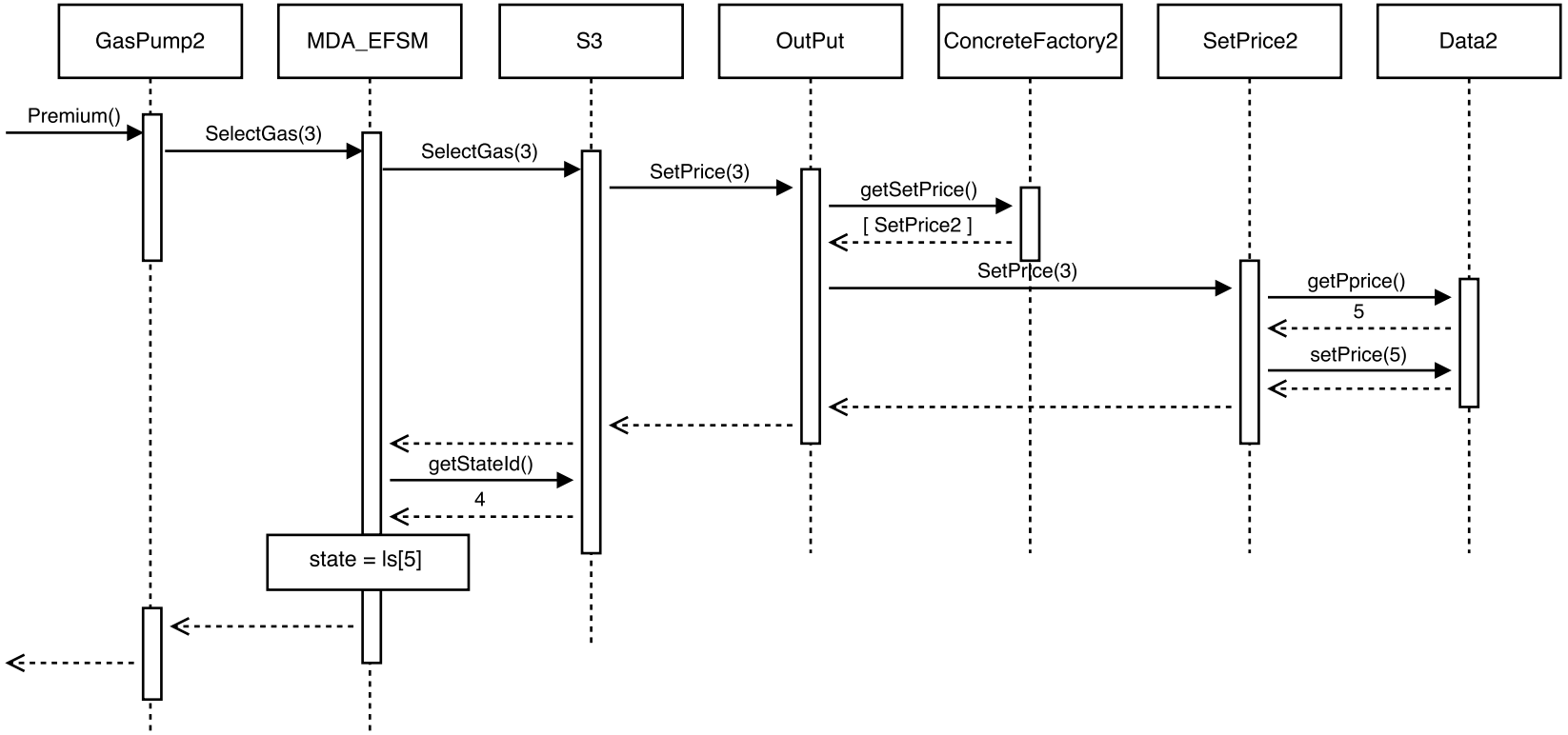

GasPump1
Data1
MDA_EFSM
S5
S6
OutPut
ConcreteFactory1
StopMsg1
PrintReceipt1
ReturnCash1
StopPump()
StopPump()
StopPump()
StopMsg()
getStopMsg()
[StopMsg1]
StopMsg()
getStateId()
6
state = ls[7]
Receipt()
Receipt()
PrintReceipt()
getPrintReceipt()
[PrintReceipt1]
PrintReceipt()
getPrice()
[3.1]
getG()
[1]
getTotal()
[3.1]
ReturnCash()
getReturnCash()
[ReturnCash1]
ReturnCash()
getStateId()
7
state = ls[0]

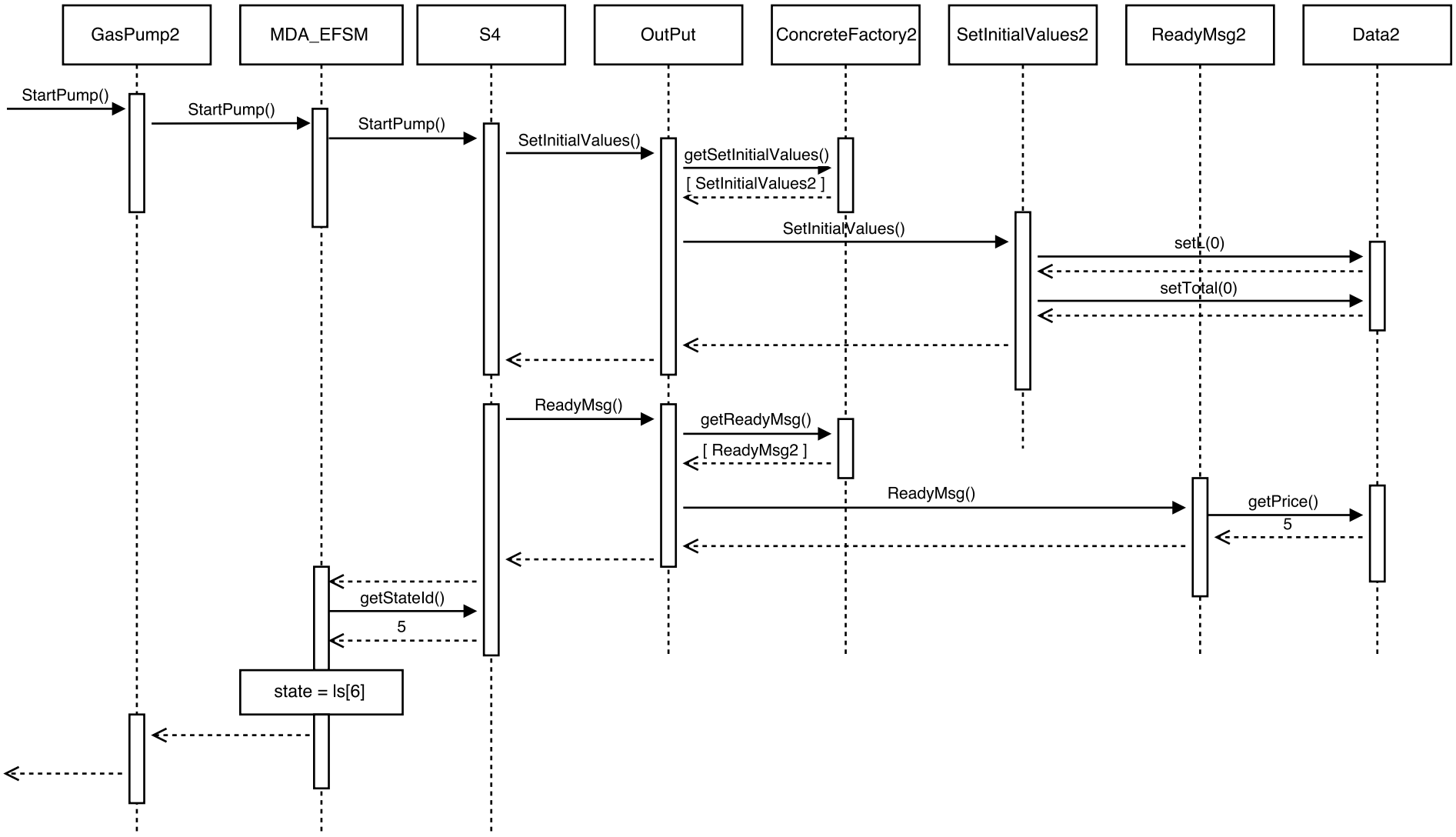
[B]

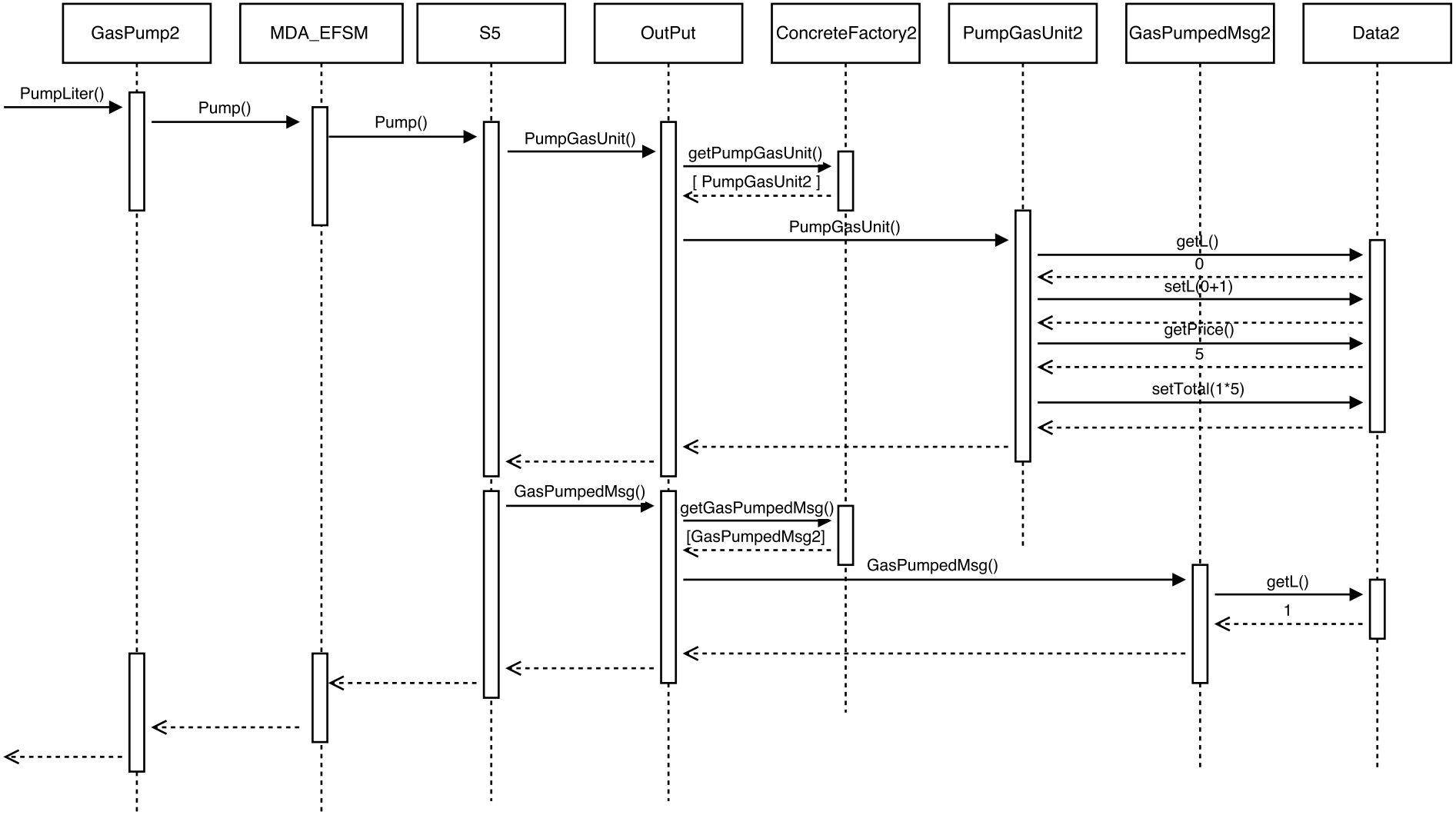

GasPump2
Data2
MDA_EFSM
Start
OutPut
ConcreteFactory2
StoreData2
Activate(3, 4, 5)
setTemp_a(3)
setTemp_b(4)
setTemp_c(5)
Activate()
Activate()
StoreData()
getStoreData()
[StoreData2]
StoreData()
getTemp_a()
3
getTemp_b()
4
getTemp_c()
5
setRprice(3)
setPprice(4)
setSprice(5)
return
getStateId()
0
state = ls[1]

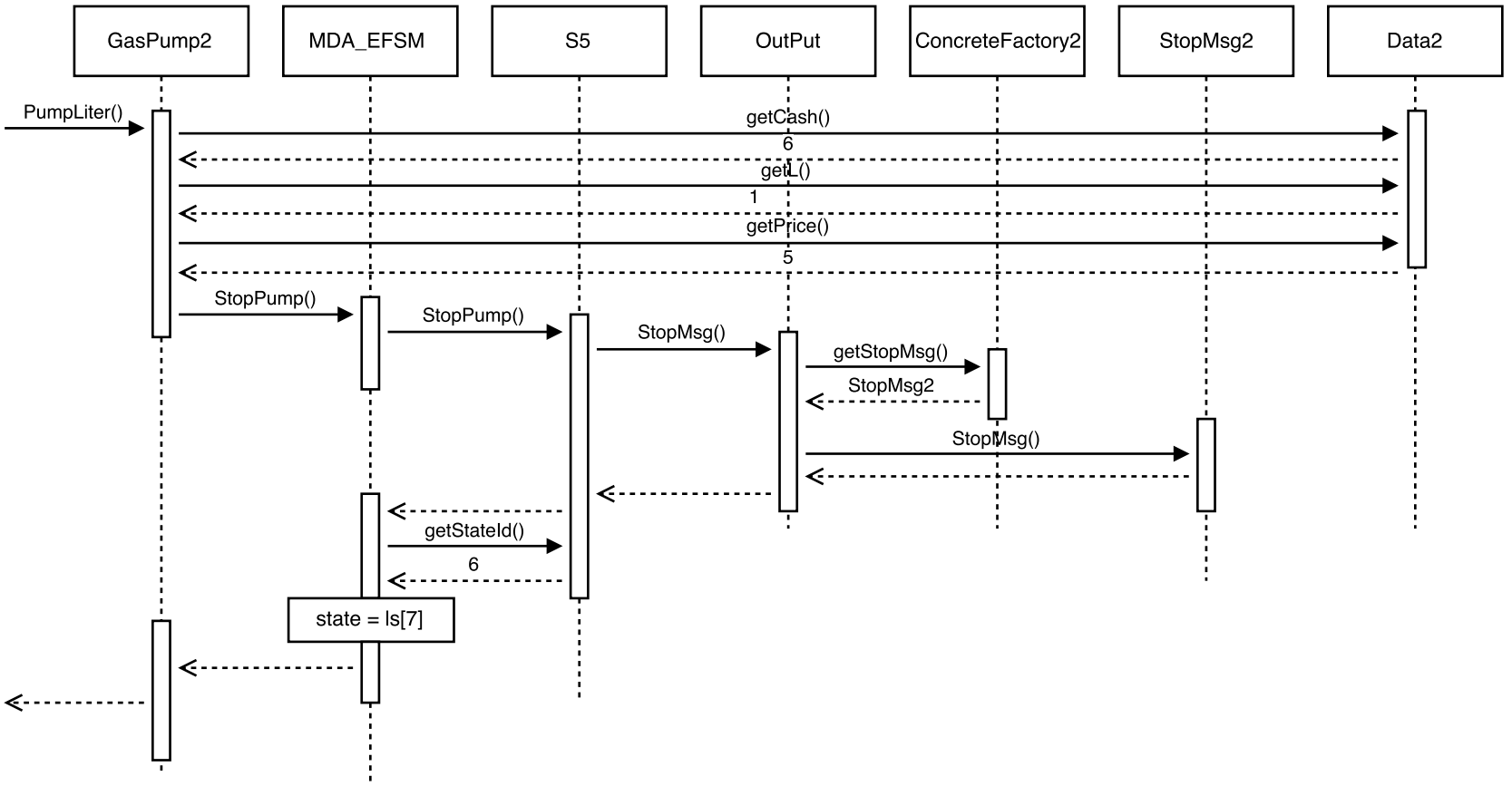

GasPump2
MDA_EFSM
S0
OutPut
ConcreteFactory2
PayMsg2
Start()
Start()
Start()
PayMsg()
getPayMsg()
[PayMsg2]
PayMsg()
getStateId()
1
state = ls[2]

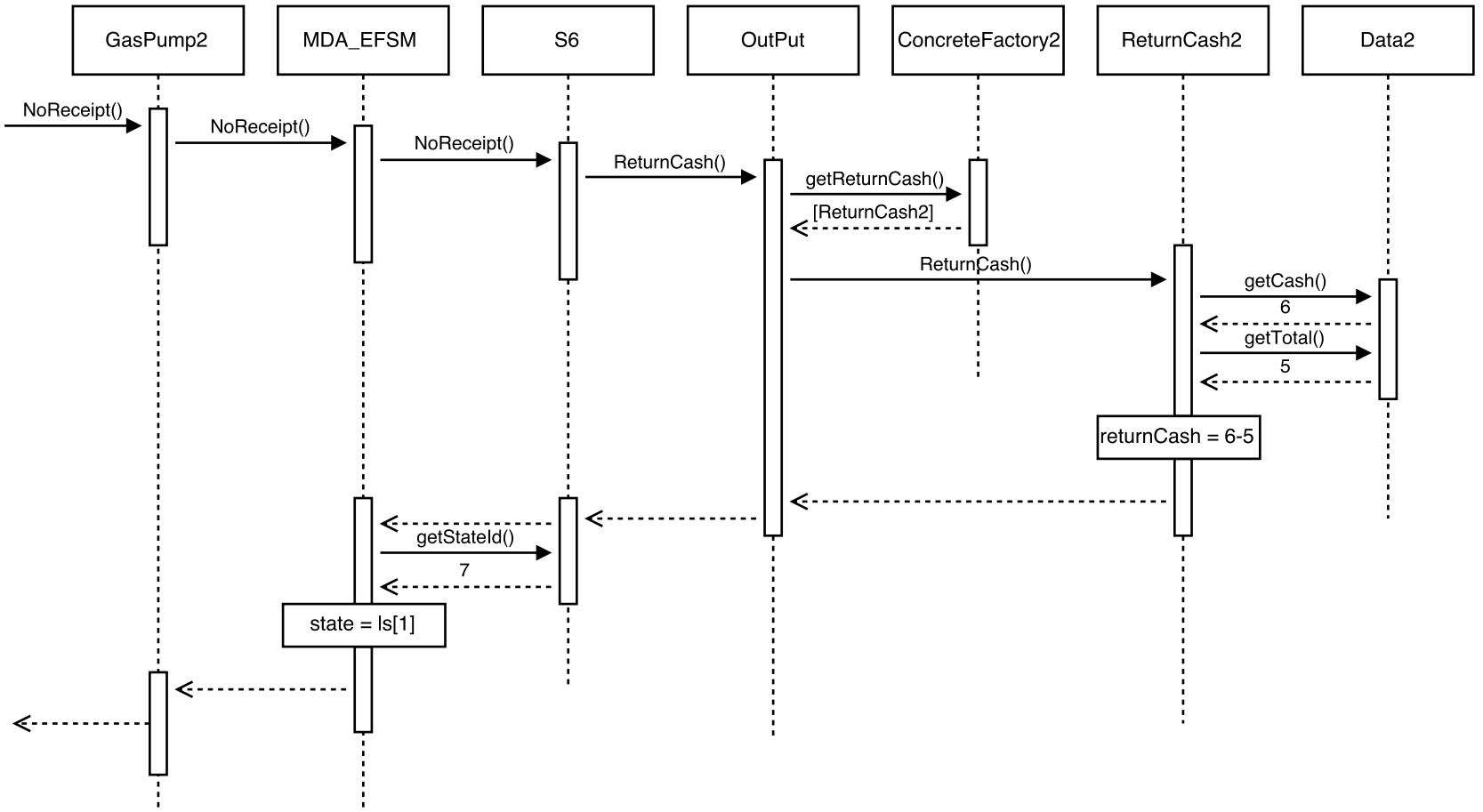

GasPump2
MDA_EFSM
S1
OutPut
ConcreteFactory2
StoreCash2
DisplayMenu2
Data2
PayCash(6)
setTemp_cash(6)
PayType(2)
PayType(2)
StoreCash()
getStoreCash()
[StoreCash2]
StoreCash()
getTemp_cash()
6
setCash(6)
DisplayMenu()
getDisplayMenu()
[DisplayMenu2]
DisplayMenu()
getRprice()
3
getPprice()
4
getSprice()
5
getStateId()
2
state = ls[4]


GasPump2
MDA_EFSM
S3
OutPut
ConcreteFactory2
SetPrice2
Data2
Premium()
SelectGas(3)
SelectGas(3)
SetPrice(3)
getSetPrice()
[SetPrice2]
SetPrice(3)
getPprice()
5
setPrice(5)
getStateId()
4
state = ls[5]


GasPump2
MDA_EFSM
S4
OutPut
ConcreteFactory2
SetInitialValues2
ReadyMsg2
Data2
StartPump()
StartPump()
StartPump()
SetInitialValues()
getSetInitialValues()
[SetInitialValues2]
SetInitialValues()
setL(0)
setTotal(0)
ReadyMsg()
getReadyMsg()
[ReadyMsg2]
ReadyMsg()
getPrice()
5
getStateId()
5
state = ls[6]


GasPump2
MDA_EFSM
S5
OutPut
ConcreteFactory2
PumpGasUnit2
GasPumpedMsg2
Data2
PumpLiter()
Pump()
Pump()
PumpGasUnit()
getPumpGasUnit()
[PumpGasUnit2]
PumpGasUnit()
getL()
0
setL(0+1)
getPrice()
5
setTotal(1*5)
GasPumpedMsg()
getGasPumpedMsg()
[GasPumpedMsg2]
GasPumpedMsg()
getL()
1


GasPump2
MDA_EFSM
S5
OutPut
ConcreteFactory2
StopMsg2
Data2
PumpLiter()
getCash()
6
getL()
1
getPrice()
5
StopPump()
StopPump()
StopMsg()
getStopMsg()
StopMsg2
StopMsg()
getStateId()
6
state = ls[7]


GasPump2
MDA_EFSM
S6
OutPut
ConcreteFactory2
ReturnCash2
Data2
NoReceipt()
NoReceipt()
NoReceipt()
ReturnCash()
getReturnCash()
[ReturnCash2]
ReturnCash()
getCash()
6
getTotal()
5
returnCash = 6-5
getStateId()
7
state = ls[1]