Final REPORT

1. **MDA-EFSM model for the Vending Machine components**

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

MDA-EFSM Events:

1. create()
2. insert_cups(int n) // n represents # of cups
3. coin(int f)
4. card()
5. cancel()
6. set_price()
7. dispose_drink(int d)
8. additive(int a)

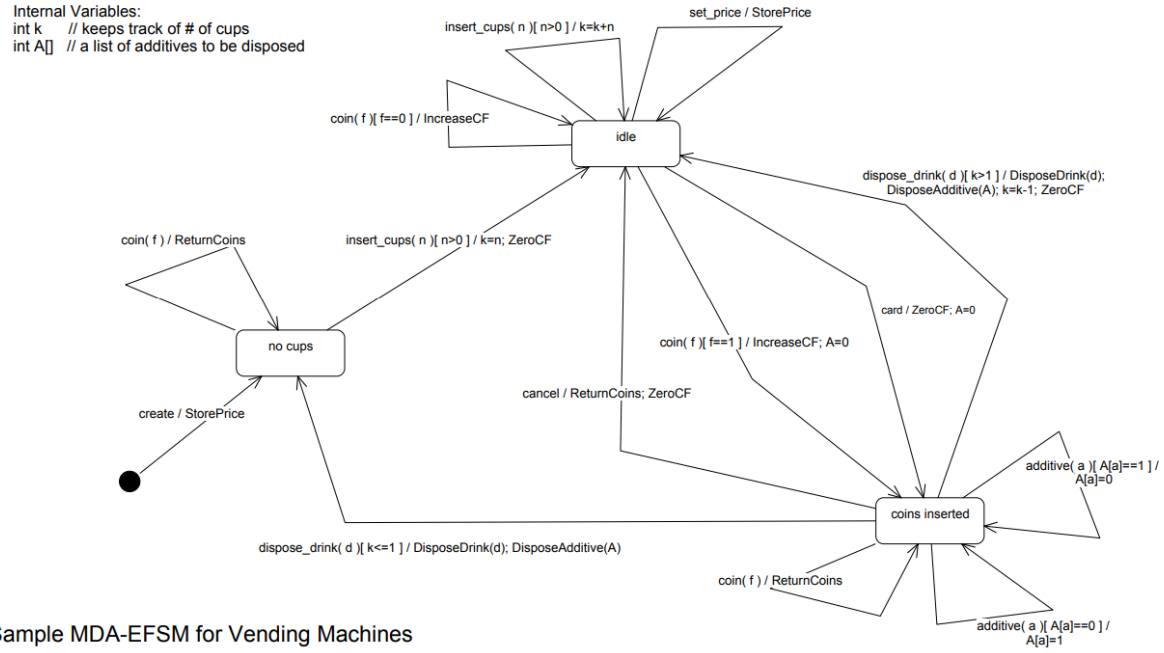
b. A list of meta actions for the MDA-EFSM with their descriptions

MDA-EFSM Actions:

1. StorePrice()
2. ZeroTotal() // zero total funds
3. IncreaseTotal() // increase Cumulative Fund cf
4. ReturnChange() // return coins inserted for a drink
5. DisposeDrink(int d) // dispose a drink with d id
6. DisposeAdditive(int A[])//dispose marked additives in A list,

// where additive with i id is disposed when A[i]=1

c. A state diagram of the MDA-EFSM



Sample MDA-EFSM for Vending Machines

d. Pseudo-code of all operations of Input Processors of Vending Machines: VM-1 and VM-2

Vending-Machine-1

Vending-Machine-1	where, *m:* pointer to the MDA-EFSM d: pointer to the data store DS-1 In the data store: *cf*: represents a cumulative fund *price*: represents the price for a drink

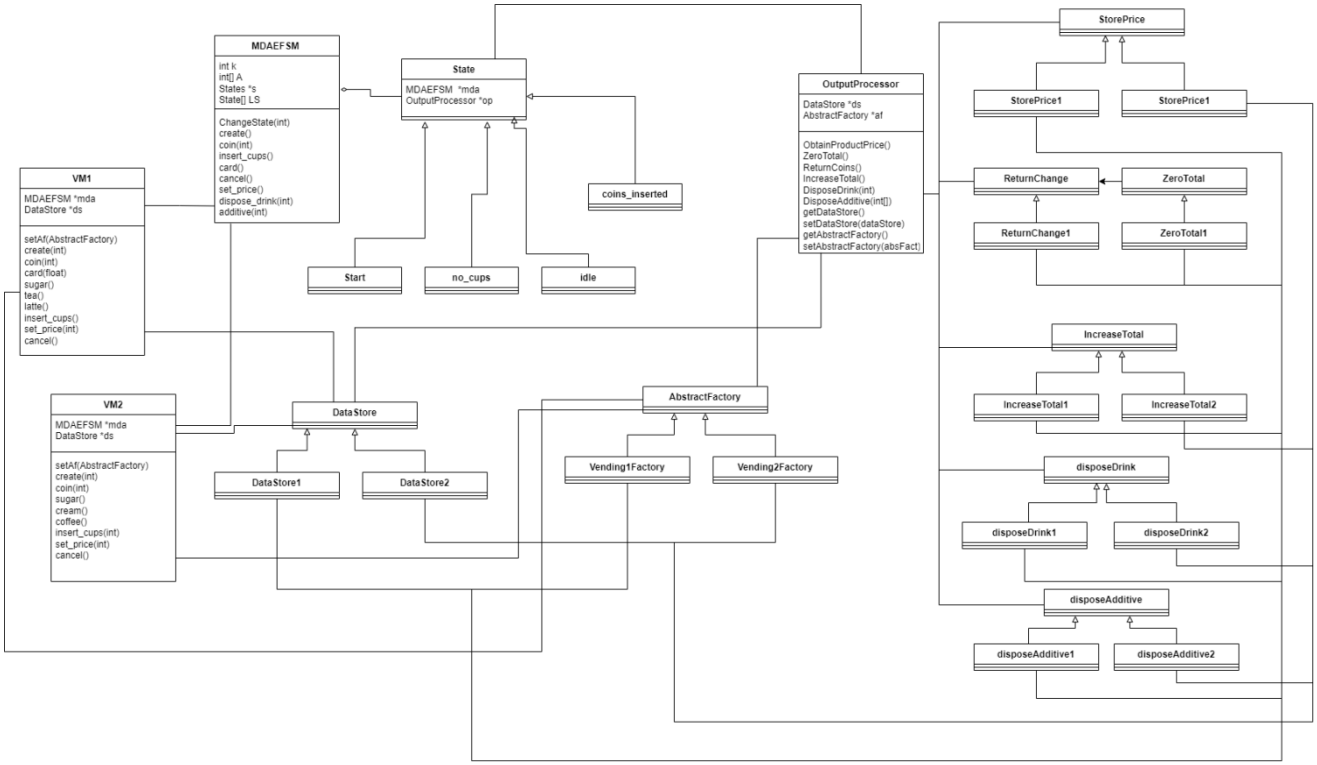
```
create(int p) {
        d->temp_p=p;
        m->create();
}

coin(float v) {
        d->temp_v=v;
        if (d->cf+v>=d->price) m->coin(1);
        else m->coin(0);
}

sugar() {
        m->additive(1);
}

tea() {
        m->dispose_drink(1);
}

latte() {
        m->dispose_drink(2);
}

insert_cups(int n) {
        m->insert_cups(n);
}

set_price(float p) {
        d->temp_p=p;
        m->set_price()
}

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

Vending-Machine-2

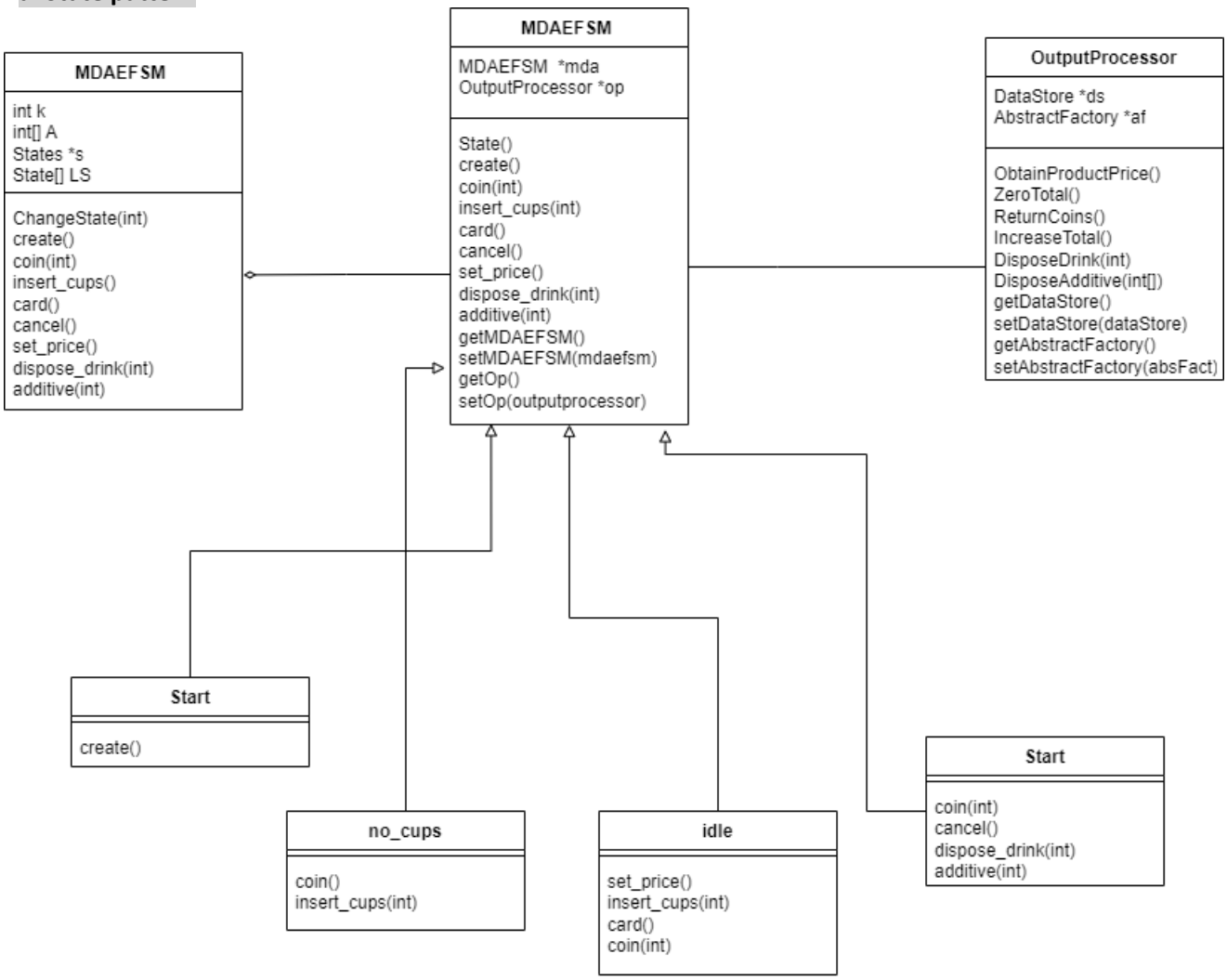
Vending-Machine-2	where, *m:* pointer to the MDA-EFSM d: pointer to the data store DS-2 In the data store: *cf*: represents a cumulative fund *price*: represents the price for a drink

```
CREATE(float p) {
        d->temp_p=p;
        m->create();
}

COIN(int v) {
        d->temp_v=v;
        if (d->cf+v>=d->price) m->coin(1);
        else m->coin(0);
}

CARD(int x) {
        if (x>=d->price) m->card();
}

SUGAR() {
        m->additive(2);
}

CREAM() {
        m->additive(1);
}

COFFEE() {
        m->dispose_drink(1);
}

InsertCups(int n) {
        m->insert_cups(n);
}

SetPrice(int p) {
        d->temp_p=p;
        m->set_price()
}

CANCEL() {
        m->cancel();
}
```

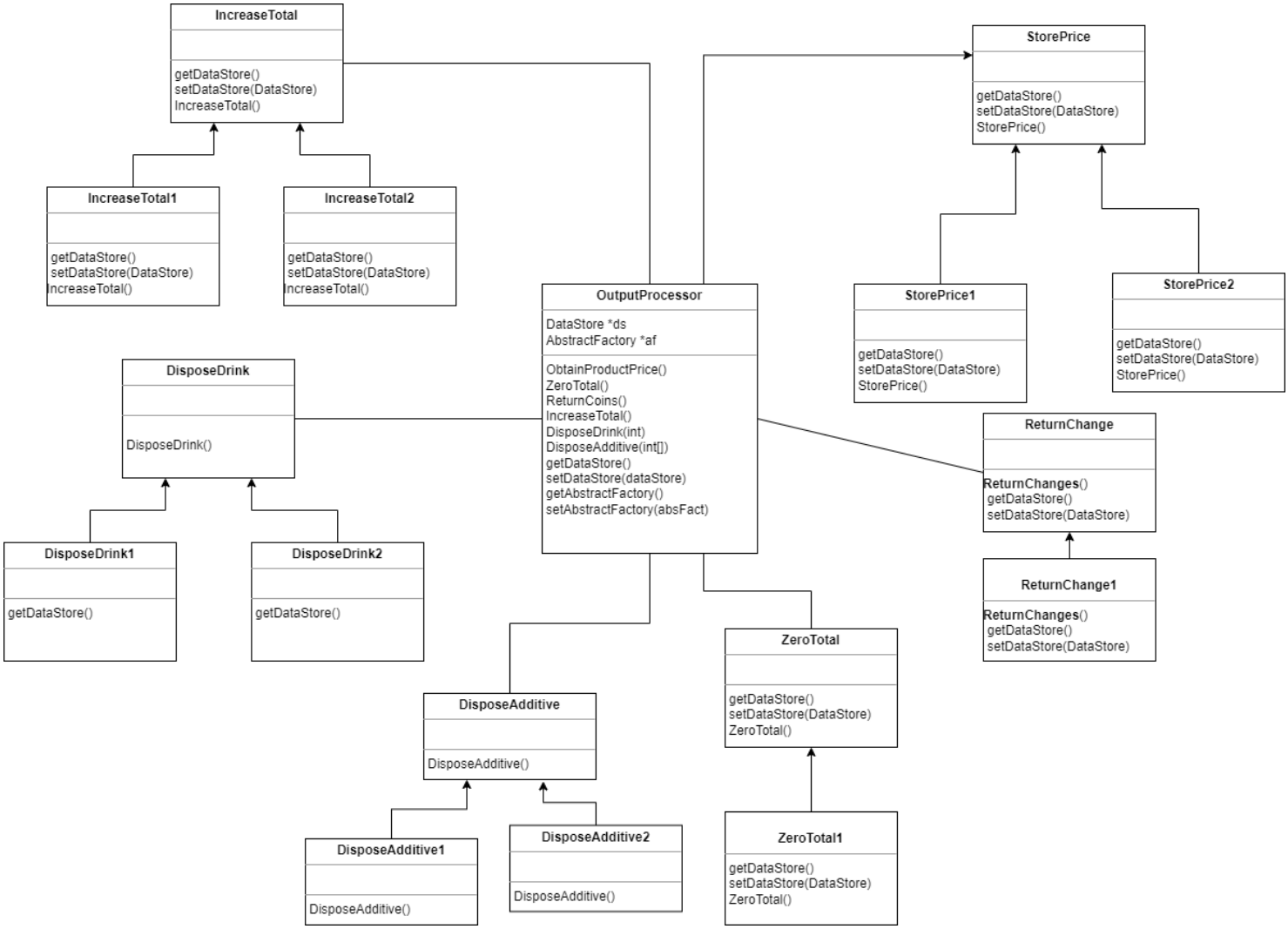
2. Class diagram(s) of the MDA of the Vending Machine components. In your design, you MUST use the following OO design patterns:


MDAEFSM
int k
int[] A
States *s
State[] LS
ChangeState(int)
create()
coin(int)
insert_cups()
card()
cancel()
set_price()
dispose_drink(int)
additive(int)
State
MDAEFSM *mda
OutputProcessor *op
OutputProcessor
DataStore *ds
AbstractFactory *af
ObtainProductPrice()
ZeroTotal()
ReturnCoins()
IncreaseTotal()
DisposeDrink(int)
DisposeAdditive(int[])
getDataStore()
setDataStore(dataStore)
getAbstractFactory()
setAbstractFactory(absFact)
StorePrice
StorePrice1
StorePrice1
ReturnChange
ZeroTotal
ReturnChange1
ZeroTotal1
VM1
MDAEFSM *mda
DataStore *ds
setAf(AbstractFactory)
create(int)
coin(int)
card(float)
sugar()
tea()
latte()
insert_cups()
set_price(int)
cancel()
coins_inserted
Start
no_cups
idle
IncreaseTotal
IncreaseTotal1
IncreaseTotal2
VM2
MDAEFSM *mda
DataStore *ds
setAf(AbstractFactory)
create(int)
coin(int)
sugar()
cream()
coffee()
insert_cups(int)
set_price(int)
cancel()
DataStore
DataStore1
DataStore2
AbstractFactory
Vending1Factory
Vending2Factory
disposeDrink
disposeDrink1
disposeDrink2
disposeAdditive
disposeAdditive1
disposeAdditive2

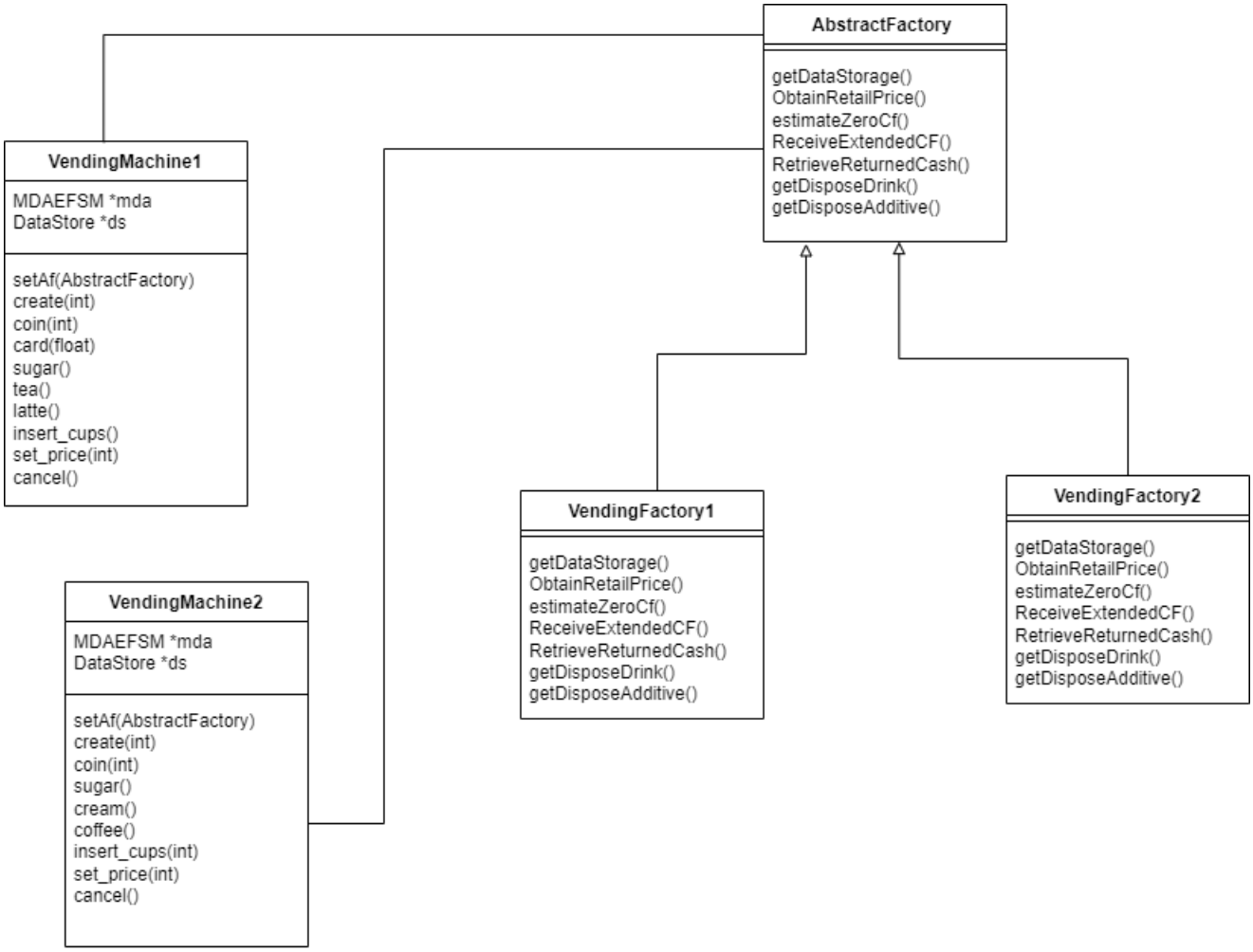
a. State pattern



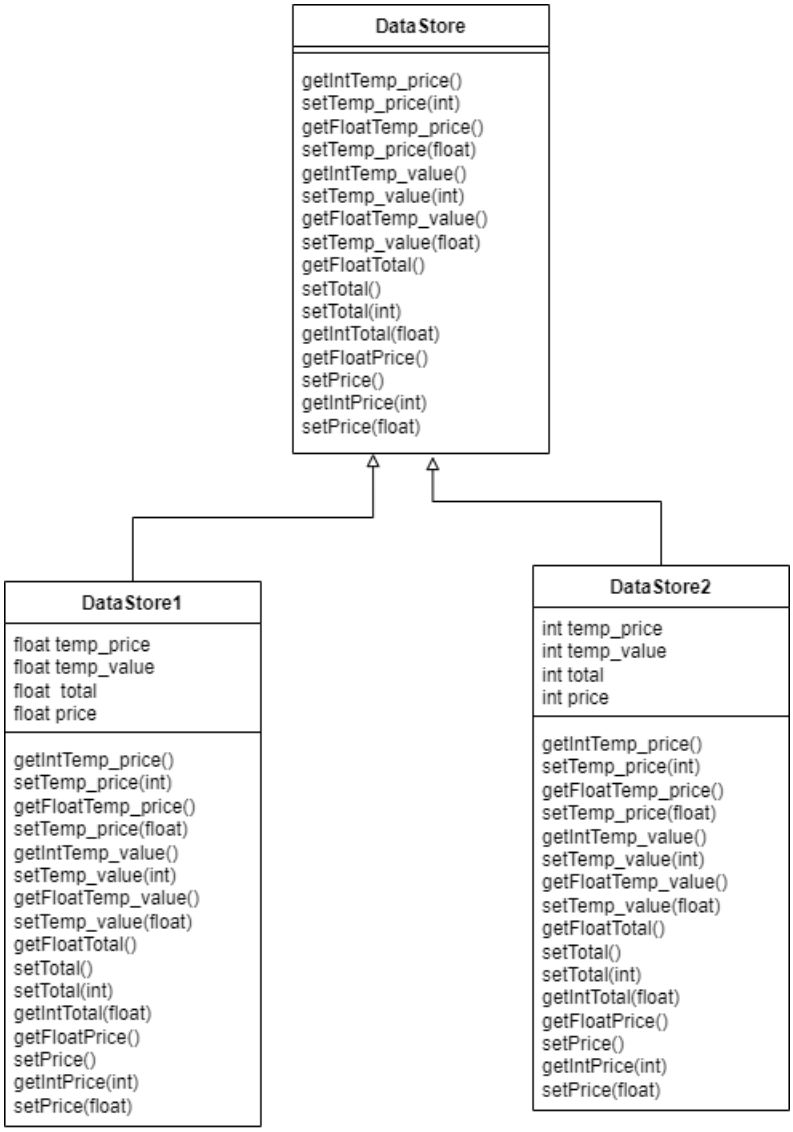
b. Strategy pattern



c. Abstract factory pattern



DataStore


DataStore
getIntTemp_price()
setTemp_price(int)
getFloatTemp_price()
setTemp_price(float)
getIntTemp_value()
setTemp_value(int)
getFloatTemp_value()
setTemp_value(float)
getFloatTotal()
setTotal()
setTotal(int)
getIntTotal(float)
getFloatPrice()
setPrice()
getIntPrice(int)
setPrice(float)
DataStore1
float temp_price
float temp_value
float total
float price
getIntTemp_price()
setTemp_price(int)
getFloatTemp_price()
setTemp_price(float)
getIntTemp_value()
setTemp_value(int)
getFloatTemp_value()
setTemp_value(float)
getFloatTotal()
setTotal()
setTotal(int)
getIntTotal(float)
getFloatPrice()
setPrice()
getIntPrice(int)
setPrice(float)
DataStore2
int temp_price
int temp_value
int total
int price
getIntTemp_price()
setTemp_price(int)
getFloatTemp_price()
setTemp_price(float)
getIntTemp_value()
setTemp_value(int)
getFloatTemp_value()
setTemp_value(float)
getFloatTotal()
setTotal()
setTotal(int)
getIntTotal(float)
getFloatPrice()
setPrice()
getIntPrice(int)
setPrice(float)

3. For each class in the class diagram(s) you should:

Class Driver: This class allows the user to select VM and perform operations on them.
Main(String[] args): This method allows to user to input different operations that can be performed by the VM.

INPUT PROCESSOR
MDAFSM *mda- Pointer to MDAEFSM object.
DataStore *ds- Pointer to DataStore object

create(int)- This method creates a vending machine and sets the price for the items
coin(int)- This method takes parameter indicating the coins inserted and compares it with the price based on which 2 paths are taken.
card(float)- This method selects card as the method of payment.
sugar()-This method is used to add Sugar as a Additive.
tea()-This method is used to dispose tea.
latte()-This method is used to dispose lattee
insert_cups()-This method is used to insert cups
set_price()-This price is used to override the previously set price value during create
cancel()-This methods used to end any transcations like revoking command after inserting coin.

Class VendingMachine2

MDAFSM *mda Pointer to MDAEFSM object.
DataStore *ds Pointer to DataStore object
Methods
CREATE(float)- This method creates a vending machine and sets the price for the items
COIN(float)-This method takes parameter indicating the coins inserted and compares it with the price based on which 2 paths are taken.
CREAM()-This method is used to add cream as an Additive.
SUGAR()-This method is used to add Sugar as an Additive.
COFFEE()-This method is used to dispose coffee.
InsertCups(int)-This method is used to insert cups
SetPrice(flaot)-This price is used to override the previously set price value during create
CANCEL()-This methods used to end any transcations like revoking command after inserting coin.

MDAEFSM

Class MDAEFSM

and vm2
Attributes
State *S Pointer to current state of MDAEFSM.
State[] LS-Stores the objects of different state classes.
Int k-Internal data variable contains number of cups
Int[] A-Contains a array of additives based on which we performs actions later on.
Methods
ChangeState(int) -This method is used to change state.
create()-This method is used to create and set price.
coin(int)-This method is used to add coins.
insert_cups(int)-This method is used to insert cups
card()-This method id used to pay via card.
cancel()- This method is used to cancel after addition of money
set_price()-This method is used to update the price.
dispose_drink(int)-This method select and dispose particular drink
additive(int)-This method is used to select additive.

Class State

Attributes
MDAEFSM *mda Pointer to MDAEFSM object.
OutputProcessor *op Pointer to OutputProcessor class object
Abstract Methods
create()-This method is used to create and set price.
coin(int)-This method is used to add coins.
insert_cups(int)- This method is used to insert cups
card()- This method id used to pay via card.
cancel()- This method is used to cancel after addition of money
set_price()-This method is used to update the price.
dispose_drink(int)-This method select and dispose particular drink
additive(int)-This method is used to select additive.
Methods
getMDAEFSM()-This method is used to get MDAEFSM object.
setMDAEFSM(MDAEFSM)-This method is used to set MDAEFSM object
getOp()- This method is used to get OutputProcessor object.
setOp(OutputProcessor)- This method is used to set OutputProcessor object.

Class start(extends of State class and represents start state.**)**
create()-Stores the price and changes the state to no_cups

Class no_cups(extends of State class and represents no_cups state.)
coin(int)- Returns any coins inserted
insert_cup(int)- If parameter is > 0 store the number of cups set cf to 0 and change state to idle.

Class idle(extends of State class and represents idle state**)**

set_price()-Stores the price value
Insert_cups(int)-If the parameter is positive we add it to the no of cups stored before
Card()-Set cf to zero and changes state to coins_inserted
Coin(int)-If argv is 1 increase cf create an array for additives and change states to coins_inserted

Class coins_inserted(extends of State class and represents coins_inserted state.**)**

coin(int)-Returns any coins inserted
cancel()-Changes state to idle
dispose_drink(int)-Disposes drink with additive and changes the state based on number of cups.
additive(int)-Sets the particular additive to 1 if 0 or otherwise

Class AbstractFactory(This is a abstract class is used to create DataStore and actions objects. Abstract Factory design pattern.**)**

getDataStore()-This is an abstract method to create and return DataStore object
getStorePrice()-This is an abstract method to create and return StorePrice object (OutputProcessor)
getZerototal()-This is an abstract method to create and return ZeroCF object (OutputProcessor)
getIncreasetotal()-This is an abstract method to create and return IncreaseCF object (OutputProcessor)
getReturnChange()-This is an abstract method to create and return ReturnCoins object (OutputProcessor)
getDisposeDrink()-This is an abstract method to create and return DisposeDrink object (OutputProcessor)
getDisposeAdditive()-This is an abstract method to create and return DisposeAdditive object (OutputProcessor)

Class Vending1Factory= This class is used to create the data store and actions objects for VendingMachine1

Class Vending2Factory= This class is used to create the data store and actions objects for VendingMachine2

getDataStore()-This is an method to create and return DataStore object
getStorePrice()-This is an method to create and return StorePrice object (OutputProcessor)
getZerototal()-This is an method to create and return ZeroCF object (OutputProcessor)
getIncreasetotal()-This is an method to create and return IncreaseCF object (OutputProcessor)
getReturnChange()-This is an method to create and return ReturnCoins object (OutputProcessor)
getDisposeDrink()-This is an method to create and return DisposeDrink object (OutputProcessor)
getDisposeAdditive()-This is an method to create and return DisposeAdditive object (OutputProcessor)

Class DataStore(This is an abstract class and is used to store platform dependent data.**)**

getIntTemp_price()-This is abstract method to get the value of temporary variable int temp_price.
setTemp_price(int)-This is abstract method to set the value of temporary variable int temp_ price.
getFloatTemp_ price ()-This is abstract method to get the value of temporary variable float temp_ price.
setTemp_ price (float)-This is abstract method to set the value of temporary variable float temp_ price.
getIntTemp_value()-This is abstract method to get the value of temporary variable int temp_value
setTemp_ value (int)-This is abstract method to set the value of temporary variable int temp_ value.
getFloatTemp_ value ()-This is abstract method to get the value of temporary variable float temp_ value.

setTemp_ value (float)-This is abstract method to set the value of temporary variable float temp_ value.
getFloatTotal()-This is abstract method to get the value of float total.
getIntTotal()-This is abstract method to get the value of int total.
setTotal(int)-This is abstract method to set the value of int total.
setTotal(float)-This is abstract method to set the value of float total.
getFloatPrice()-This is abstract method to get the value of float price
getIntPrice()-This is abstract method to get the value of int price.
setPrice(int)-This is abstract method to set the value of int Price
setPrice(float)-This is abstract method to set the value of float Price

Class DataStore1(This is an abstract class and is used to store platform dependent data.)

getIntTemp_price()-This is abstract method to get the value of temporary variable int temp_price.
setTemp_price(int)-This is abstract method to set the value of temporary variable int temp_ price.
getFloatTemp_ price ()-This is abstract method to get the value of temporary variable float temp_ price.
setTemp_ price (float)-This is abstract method to set the value of temporary variable float temp_ price.
getIntTemp_value()-This is abstract method to get the value of temporary variable int temp_value
setTemp_ value (int)-This is abstract method to set the value of temporary variable int temp_ value.
getFloatTemp_ value ()-This is abstract method to get the value of temporary variable float temp_ value.
setTemp_ value (float)-This is abstract method to set the value of temporary variable float temp_ value.
getFloatTotal()-This is abstract method to get the value of float total.
getIntTotal()-This is abstract method to get the value of int total.
setTotal(int)-This is abstract method to set the value of int total.
setTotal(float)-This is abstract method to set the value of float total.
getFloatPrice()-This is abstract method to get the value of float price
getIntPrice()-This is abstract method to get the value of int price.
setPrice(int)-This is abstract method to set the value of int Price
setPrice(float)-This is abstract method to set the value of float Price

Class DataStore2(This is an abstract class and is used to store platform dependent data.)

getIntTemp_price()-This is abstract method to get the value of temporary variable int temp_price.
setTemp_price(int)-This is abstract method to set the value of temporary variable int temp_ price.
getFloatTemp_ price ()-This is abstract method to get the value of temporary variable float temp_ price.
setTemp_ price (float)-This is abstract method to set the value of temporary variable float temp_ price.
getIntTemp_value()-This is abstract method to get the value of temporary variable int temp_value
setTemp_ value (int)-This is abstract method to set the value of temporary variable int temp_ value.
getFloatTemp_ value ()-This is abstract method to get the value of temporary variable float temp_ value.
setTemp_ value (float)-This is abstract method to set the value of temporary variable float temp_ value.
getFloatTotal()-This is abstract method to get the value of float total.
getIntTotal()-This is abstract method to get the value of int total.
setTotal(int)-This is abstract method to set the value of int total.
setTotal(float)-This is abstract method to set the value of float total.
getFloatPrice()-This is abstract method to get the value of float price
getIntPrice()-This is abstract method to get the value of int price.
setPrice(int)-This is abstract method to set the value of int Price

setPrice(float)-This is abstract method to set the value of float Price

Class OutputProcessor- This class is the Output processor which is used to execute actions called by the mdaefsm.

Attributes

private DataStore ds;
private AbstractFactory af;
private StorePrice StorePrice;
private ZeroCF ZeroCF;
private ReturnCoins ReturnCoins;
private IncreaseCF IncreaseCF;
private DisposeDrink DisposeDrink;
private DisposeAdditive DisposeAdditive;

Methods()

StorePrice()-This method creates StorePrices object using AbstractFactory class and It executes the storePrices() method of StorePrices class.
ZeroCF()-This method creates ZeroCf object using AbstractFactory class and It executes the ZeroCF() method of ZeroCf class.
ReturnCoins()-This method creates ReturnCoinobject using AbstractFactory class and It executes the ReturnCoin () method of ReturnCoinclass.
IncreaseCf()-This method creates IncreaseCf object using AbstractFactory class and It executes the IncreaseCf () method of IncreaseCf class.
DisposeDrink(int)-This method creates DisposeDrink object using AbstractFactory class and It executes the DisposeDrink () method of DisposeDrink class.
DisposeAdditive(int)-This method creates DisposeAdditive object using AbstractFactory class and It executes the DisposeAdditive () method of DisposeAdditive class.
getDataStore()-Get DataStore object
setDataStore(DataStore)-set DataStore object
getAbstractFactory()-Get AbstractFactory object
setAbstractFactory(AbstractFactory)-set AbstractFactory object

Class StorePrice(Interface class to store price)

StorePrice()-This is an Interface method for storing price.

Class StorePrice1(Interface class to store price)

StorePrice()-This is an Interface method for storing price.

Class StorePrice2(Interface class to store price)

StorePrice()-This is an Interface method for storing price.

Class ReturnChange(Interface class to return coins**)**
ReturnChange()-Interface metod for returning change

Class ReturnChange1(Interface class to return coins**)**
ReturnChange()-Interface metod for returning change

Class IncreaseTotal(Interface class to increasec**)**
Attributes

DataStore *ds- Pointer to DataStore
Method
IncreaseCF ()-This Interface method is used to increase cf
getDataStore()-Get DataStore object
setDataStore(DataStore ds)-set DataStore object

Class IncreaseTotal1(Interface class to increasec**)**
Attributes
DataStore *ds- Pointer to DataStore
Method
IncreaseCF ()-This Interface method is used to increase cf
getDataStore()-Get DataStore object
setDataStore(DataStore ds)-set DataStore object

Class IncreaseTotal2(Interface class to increasec**)**
Attributes
DataStore *ds- Pointer to DataStore
Method
IncreaseCF ()-This Interface method is used to increase cf
getDataStore()-Get DataStore object
setDataStore(DataStore ds)-set DataStore object

Class ZeroTotal(Interface class to ZeroTotal**)**

ZeroTotal()-Interface method to set total to 0

Class ZeroTotal1(Interface class to ZeroTotal**)**

ZeroTotal()-Interface method to set total to 0

Class DisposeDrink(Interface class to DisposeDrink**)**

DisposeDrink (int)- Interface method to dispose drink

Class DisposeDrink1(Interface class to DisposeDrink**)**

DisposeDrink (int)- Interface method to dispose drink

Class DisposeDrink2(Interface class to DisposeDrink**)**

DisposeDrink (int)- Interface method to dispose drink

Class DisposeAdditive(Interface class to DisposeAdditive**)**

DisposeAdditive (int[]) Interface method to dispose additive

Class DisposeAdditive1(Interface class to DisposeAdditive**)**

DisposeAdditive (int[]) Interface method to dispose additive

Class DisposeAdditive2(Interface class to DisposeAdditive**)**

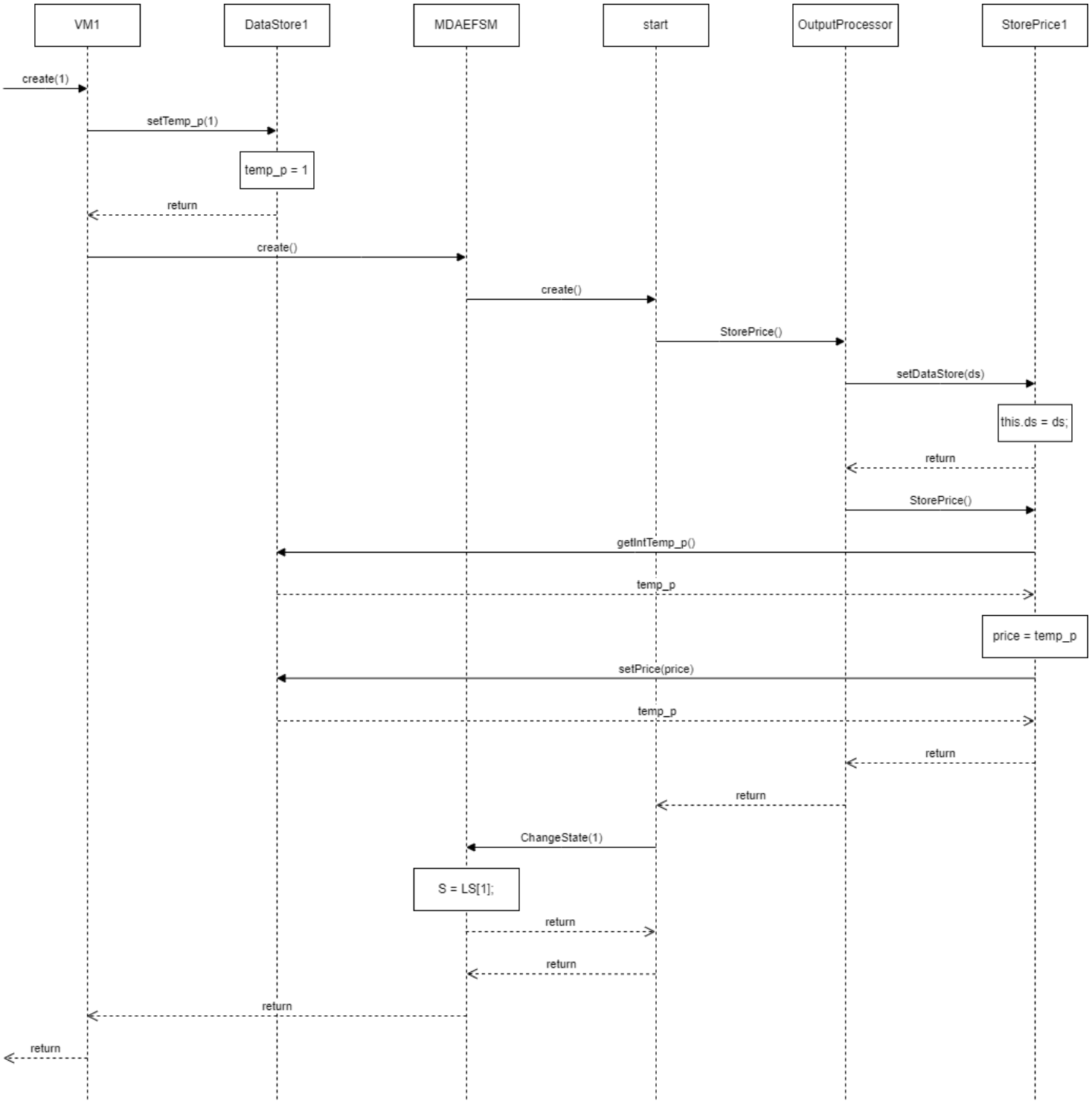
DisposeAdditive (int[]) Interface method to dispose additive

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

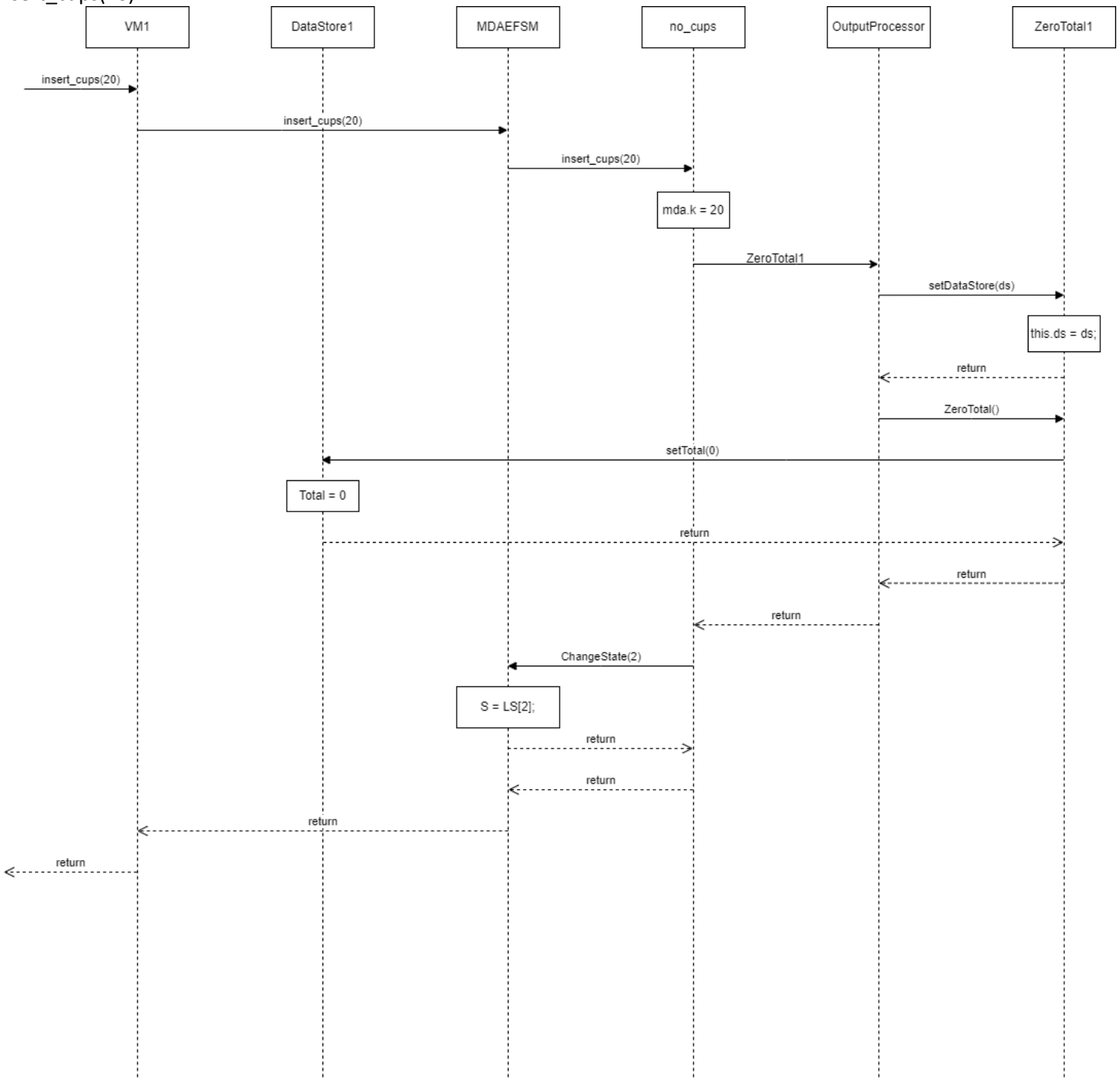
a. Scenario-I

create(1), insert_cups(20), coin(0.5), coin(0.5), sugar(), latte()

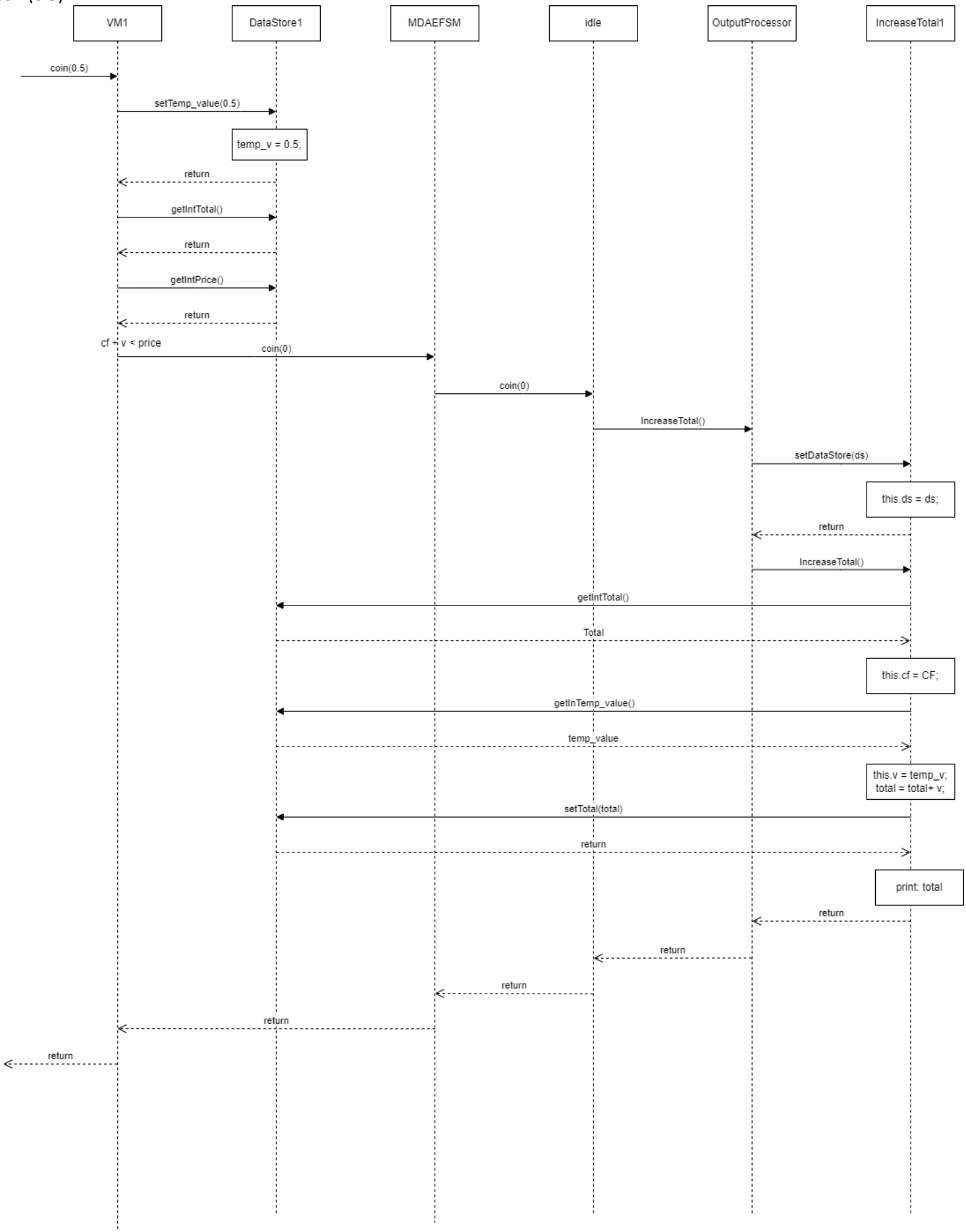
create(1)



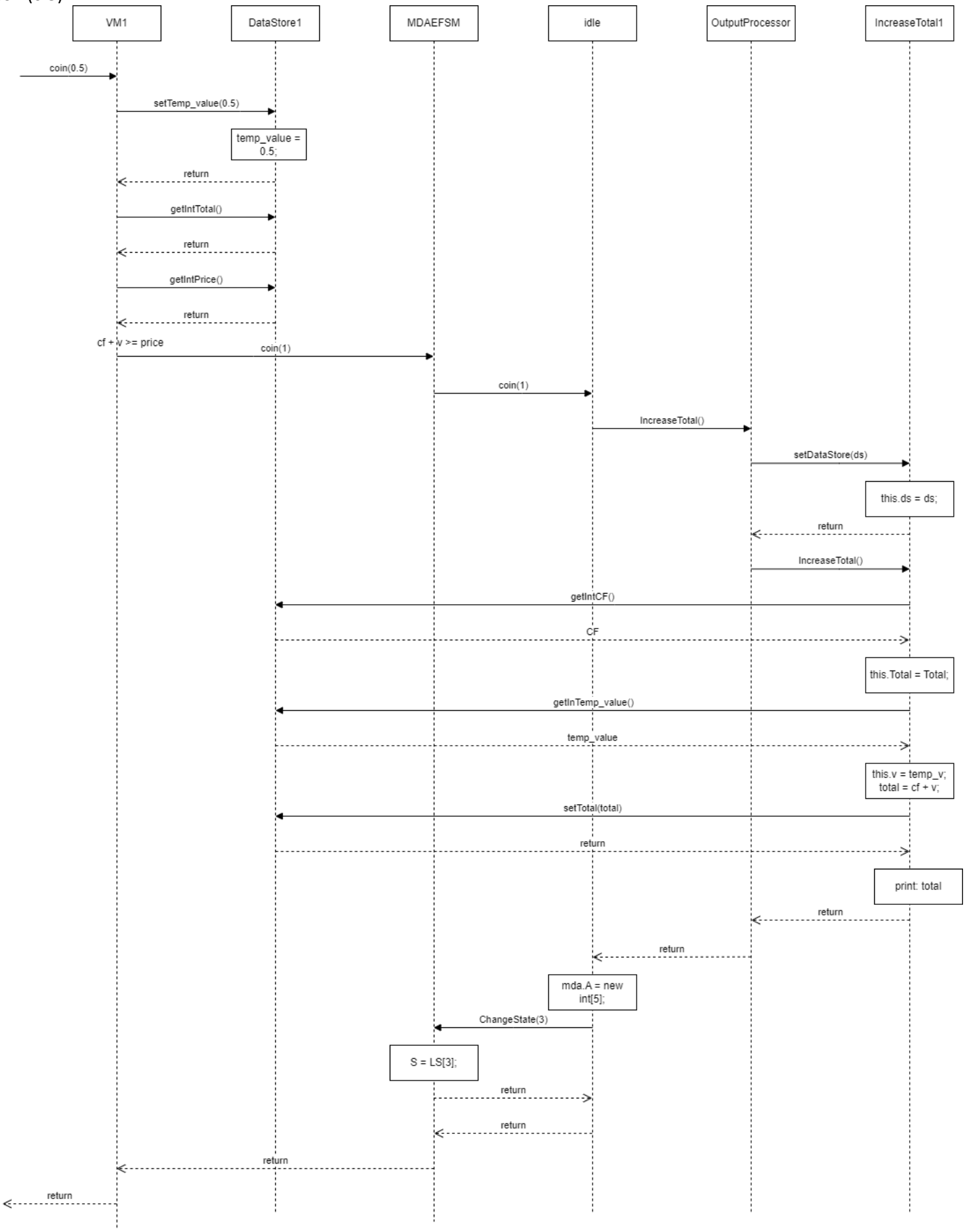
insert_cups(20)



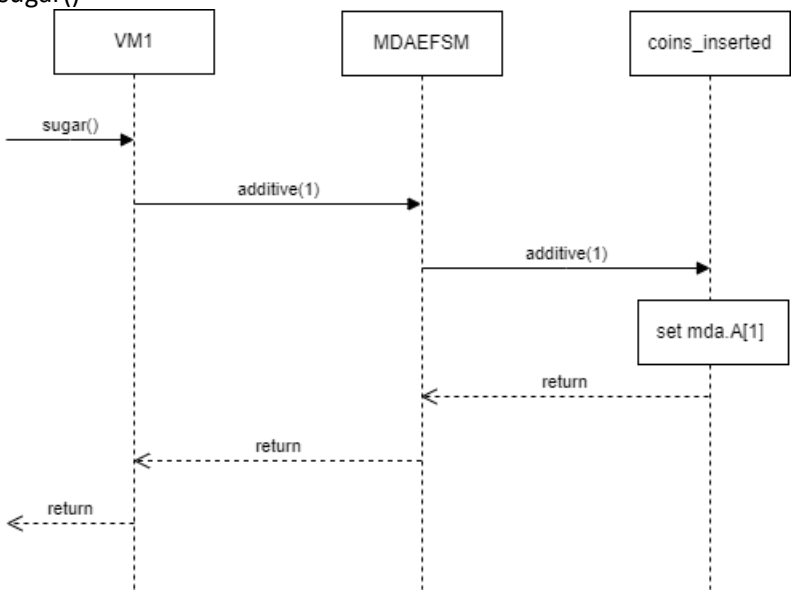
coin(0.5)


VM1
DataStore1
MDAEFSM
idle
OutputProcessor
IncreaseTotal1
coin(0.5)
setTemp_value(0.5)
temp_v = 0.5;
return
getIntTotal()
return
getIntPrice()
return
cf + v < price
coin(0)
coin(0)
IncreaseTotal()
setDataStore(ds)
this.ds = ds;
return
IncreaseTotal()
getIntTotal()
Total
this.cf = CF;
getInTemp_value()
temp_value
this.v = temp_v;
total = total+ v;
setTotal(total)
return
print: total
return
return
return
return
return

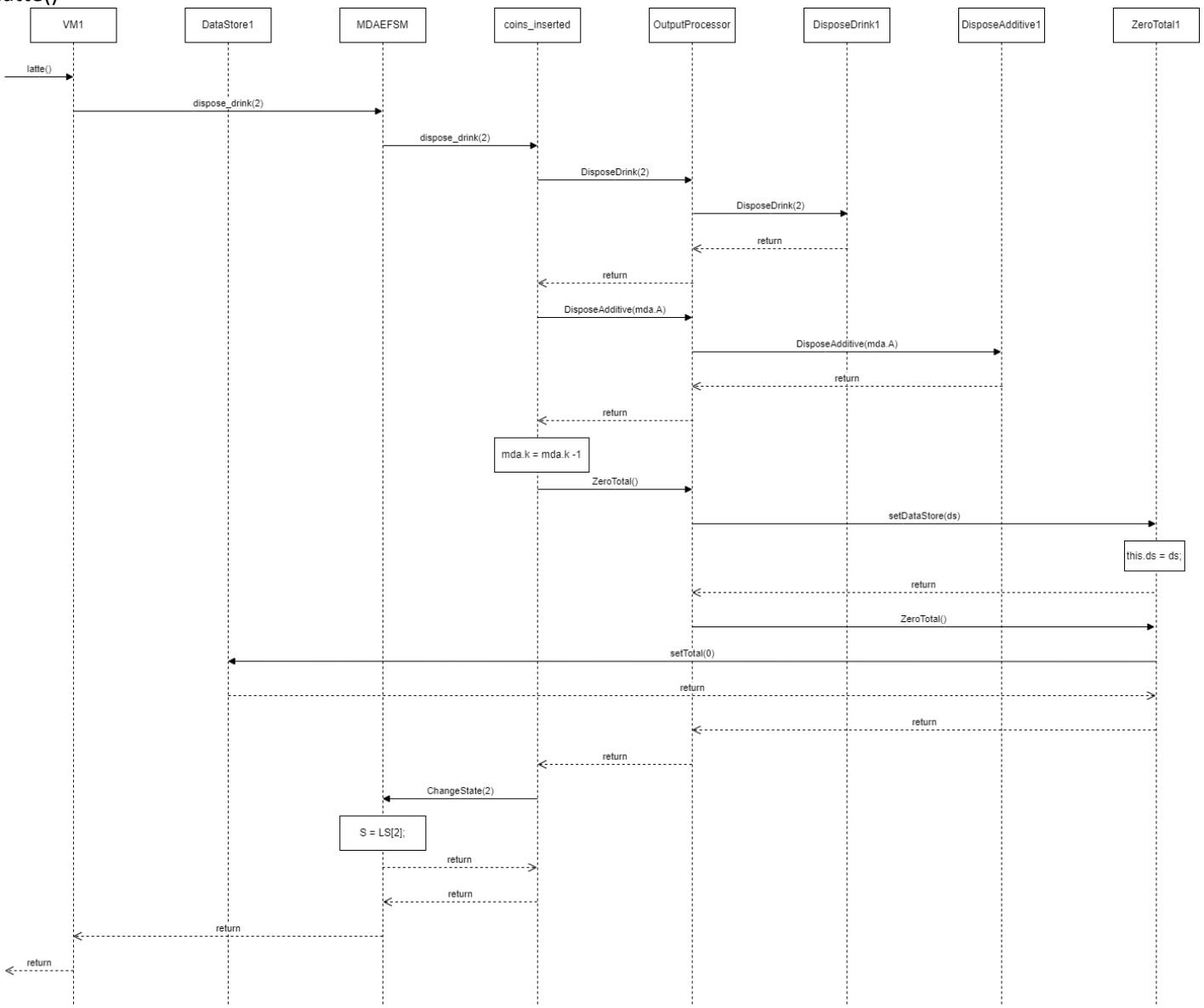
coin(0.5)


VM1
DataStore1
MDAEFSM
idle
OutputProcessor
IncreaseTotal1
coin(0.5)
setTemp_value(0.5)
temp_value = 0.5;
return
getIntTotal()
return
getIntPrice()
return
cf + v >= price
coin(1)
coin(1)
IncreaseTotal()
setDataStore(ds)
this.ds = ds;
return
IncreaseTotal()
getIntCF()
CF
this.Total = Total;
getInTemp_value()
temp_value
this.v = temp_v;
total = cf + v;
setTotal(total)
return
print: total
return
return
mda.A = new int[5];
ChangeState(3)
S = LS[3];
return
return
return
return

sugar()


VM1
MDAEFSM
coins_inserted
sugar()
additive(1)
additive(1)
set mda.A[1]
return
return
return

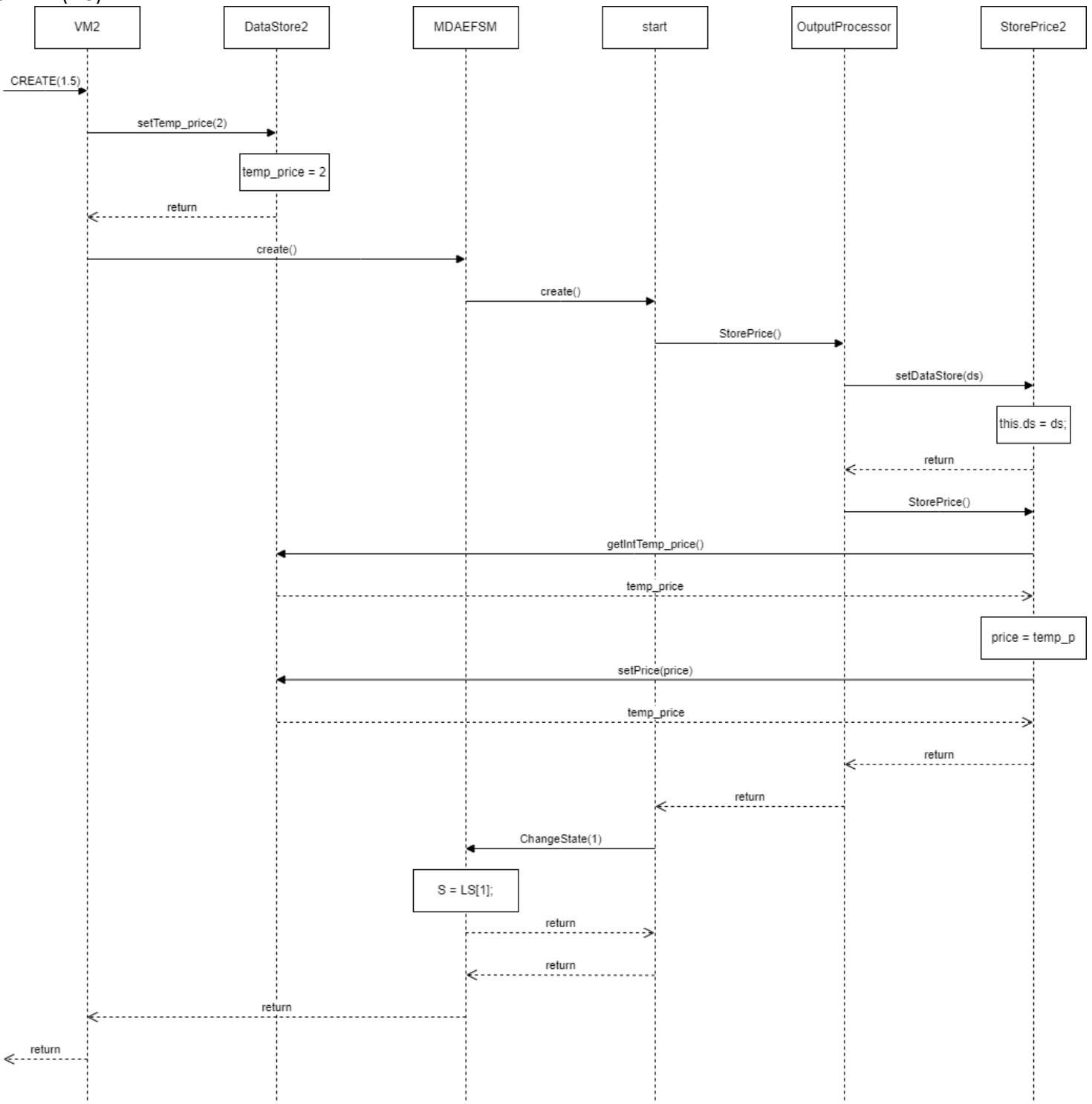
latte()


VM1
DataStore1
MDAEFSM
coins_inserted
OutputProcessor
DisposeDrink1
DisposeAdditive1
ZeroTotal1
latte()
dispose_drink(2)
dispose_drink(2)
DisposeDrink(2)
DisposeDrink(2)
return
return
DisposeAdditive(mda.A)
DisposeAdditive(mda.A)
return
return
mda.k = mda.k -1
ZeroTotal()
setDataStore(ds)
this.ds = ds;
return
ZeroTotal()
setTotal(0)
return
return
return
ChangeState(2)
S = LS[2];
return
return
return
return

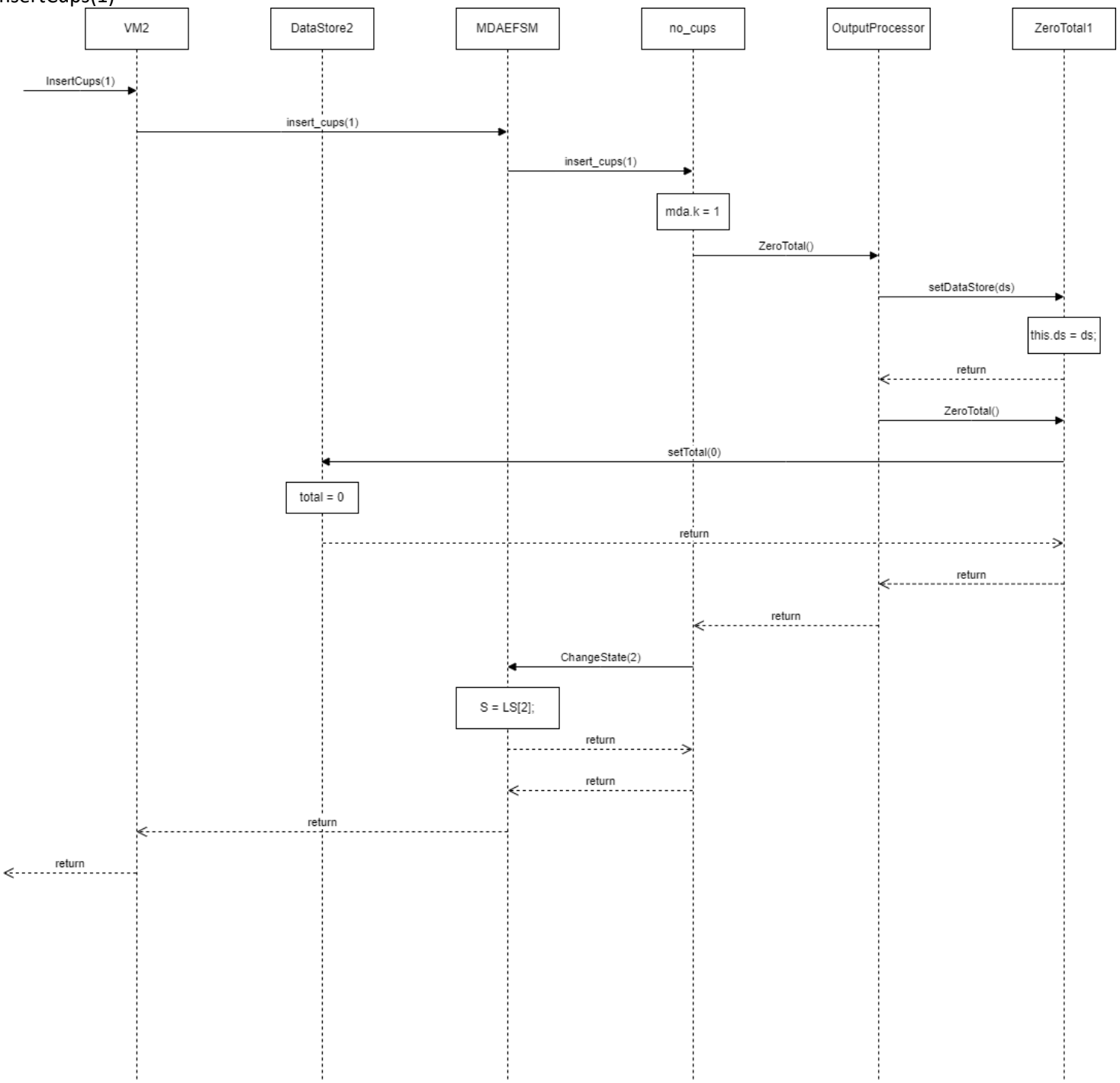
b. Scenario-II should show as to how a cup of coffee is disposed in the Vending Machine VM-2 component, i.e., the following sequence of operations is issued:

CREATE(1.5), InsertCups(1), CARD(10), CREAM(), COFFEE()

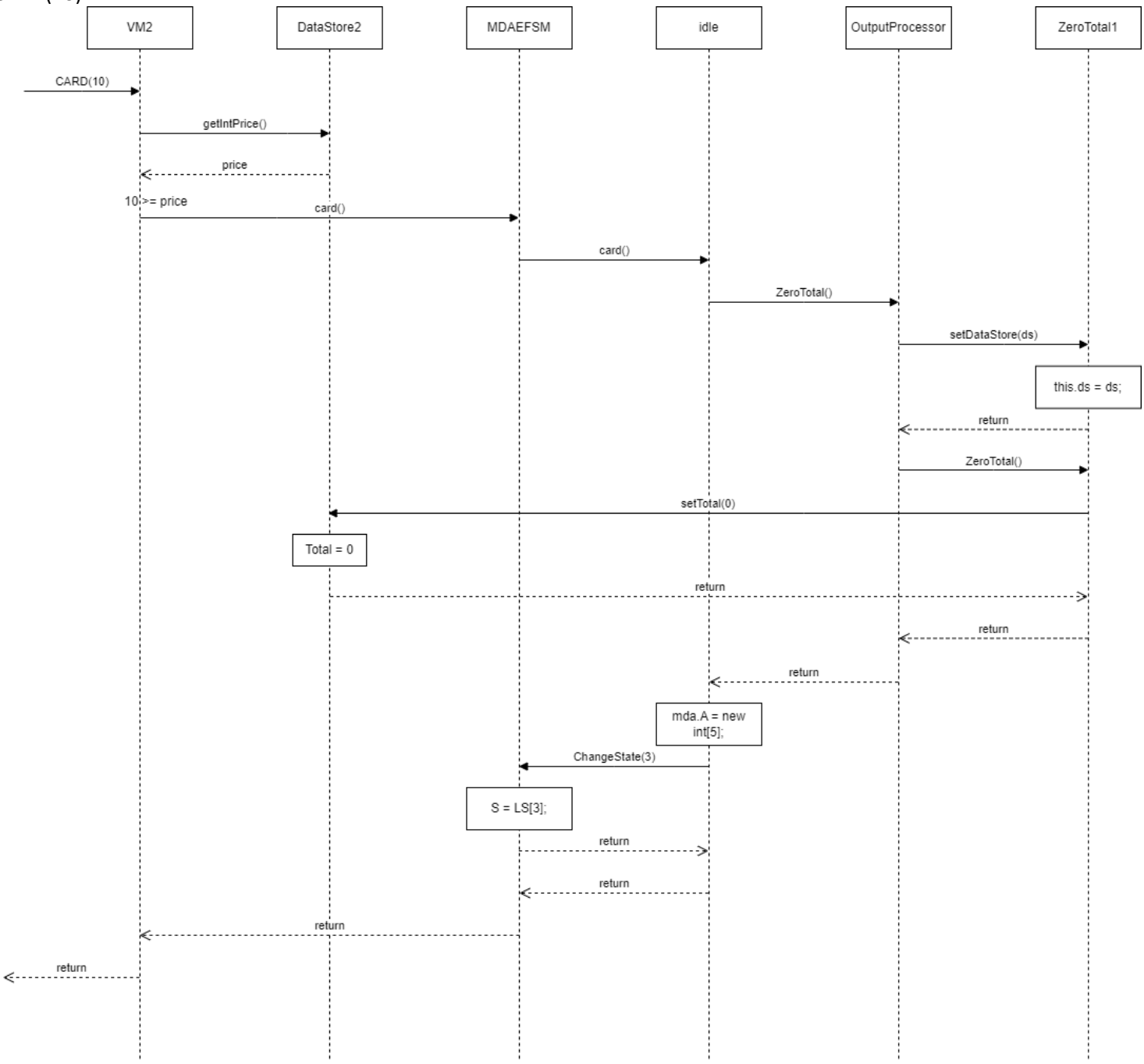
CREATE(1.5)



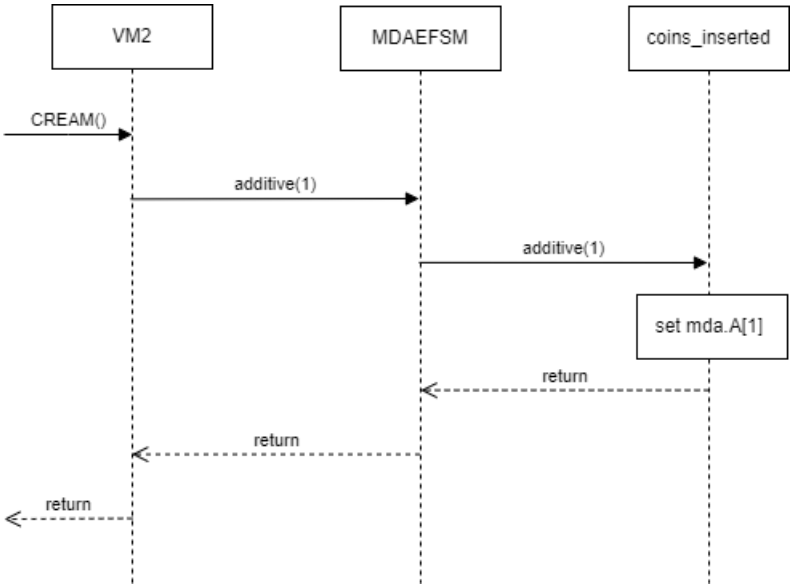
InsertCups(1)



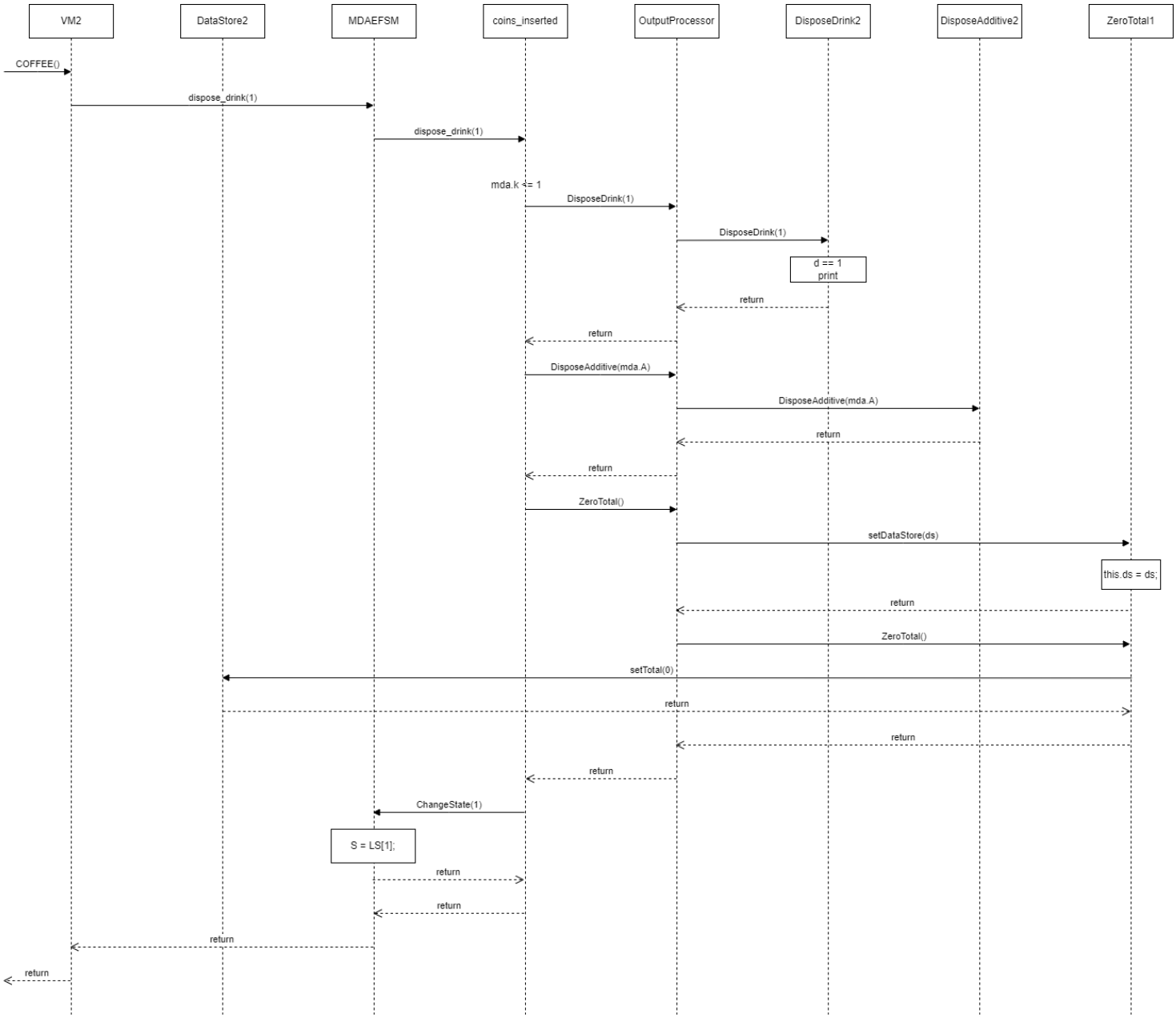
CARD(10)


VM2
DataStore2
MDAEFSM
idle
OutputProcessor
ZeroTotal1
CARD(10)
getIntPrice()
price
10 >= price
card()
card()
ZeroTotal()
setDataStore(ds)
this.ds = ds;
return
ZeroTotal()
setTotal(0)
Total = 0
return
return
return
mda.A = new int[5];
ChangeState(3)
S = LS[3];
return
return
return
return

CREAM()


VM2
MDAEFSM
coins_inserted
CREAM()
additive(1)
additive(1)
set mda.A[1]
return
return
return

COFFEE()


VM2
DataStore2
MDAEFSM
coins_inserted
OutputProcessor
DisposeDrink2
DisposeAdditive2
ZeroTotal1
COFFEE()
dispose_drink(1)
dispose_drink(1)
mda.k <= 1
DisposeDrink(1)
DisposeDrink(1)
d == 1
print
return
return
DisposeAdditive(mda.A)
DisposeAdditive(mda.A)
return
return
ZeroTotal()
setDataStore(ds)
this.ds = ds;
return
ZeroTotal()
setTotal(0)
return
return
return
ChangeState(1)
S = LS[1];
return
return
return
return