Final Project Report

Yuqi Zhou

A20423555

25 April

1 MDA-EFSM model for the Vending Machine components

a). Meta events list for the MDA-EFSM.

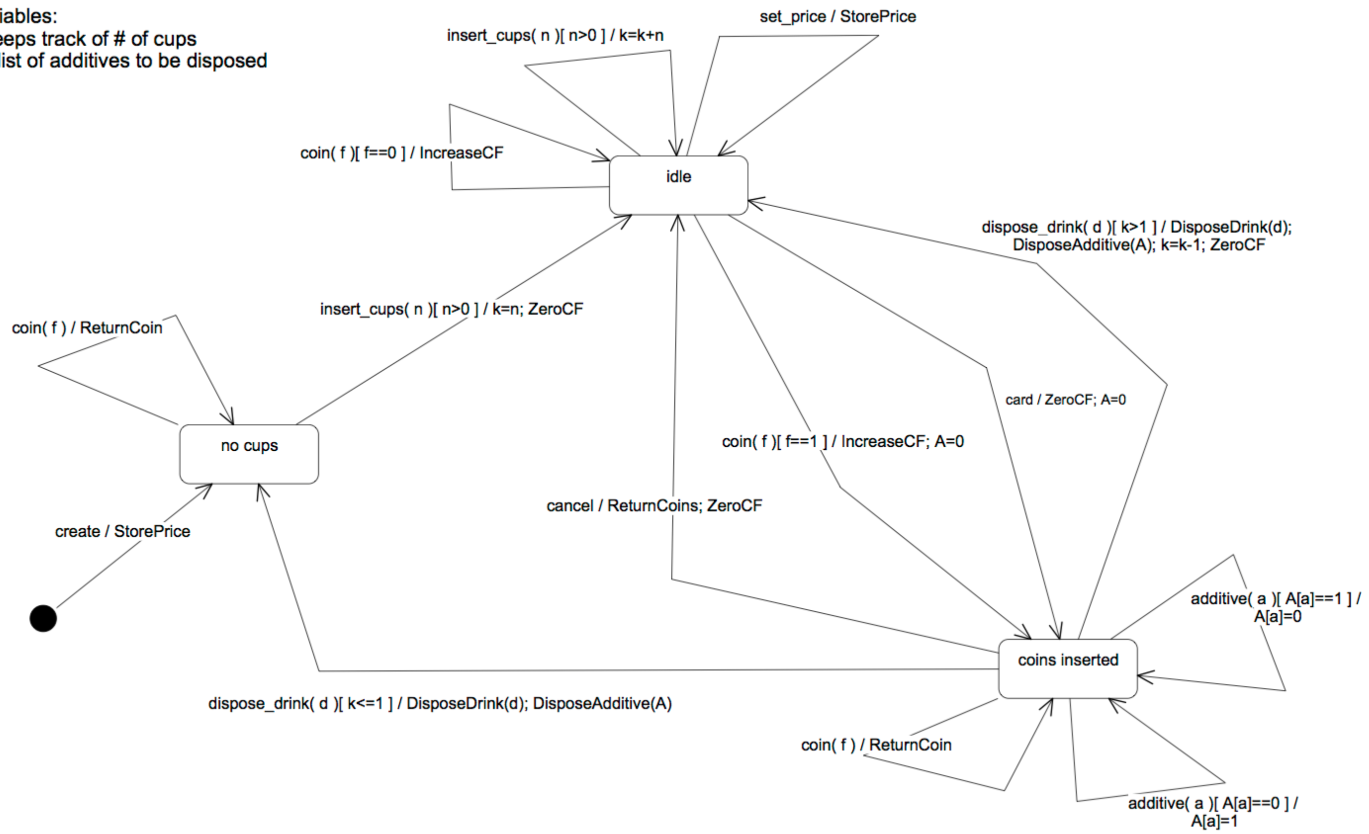
```
1 create()
2 insert_cups(int n)          // n represents # of cups
3 coin(int f)                 // f=1: sufficient funds inserted for a drink
                              // f=0: not sufficient funds for a drink
4 card()
5 cancel(float x)
6 cancel()
7 set_price()
8 dispose_drink(int d)        // d represents a drink id
9 additive(int a)             // a represents additive id
```

b). Meta actions list for the MDA-EFSM.

```
1 StorePrice()
2 ZeroCF()                    // zero Cumulative Fund cf
3 IncreaseCF()                // increase Cumulative Fund cf
4 ReturnCoins(int r)          // return coins inserted for a drink
5 DisposeDrink(int d)         // dispose a drink with d id
6 DisposeAdditive(int A[])    // dispose additives in A list
```

c). State diagram of the MDA-EFSM

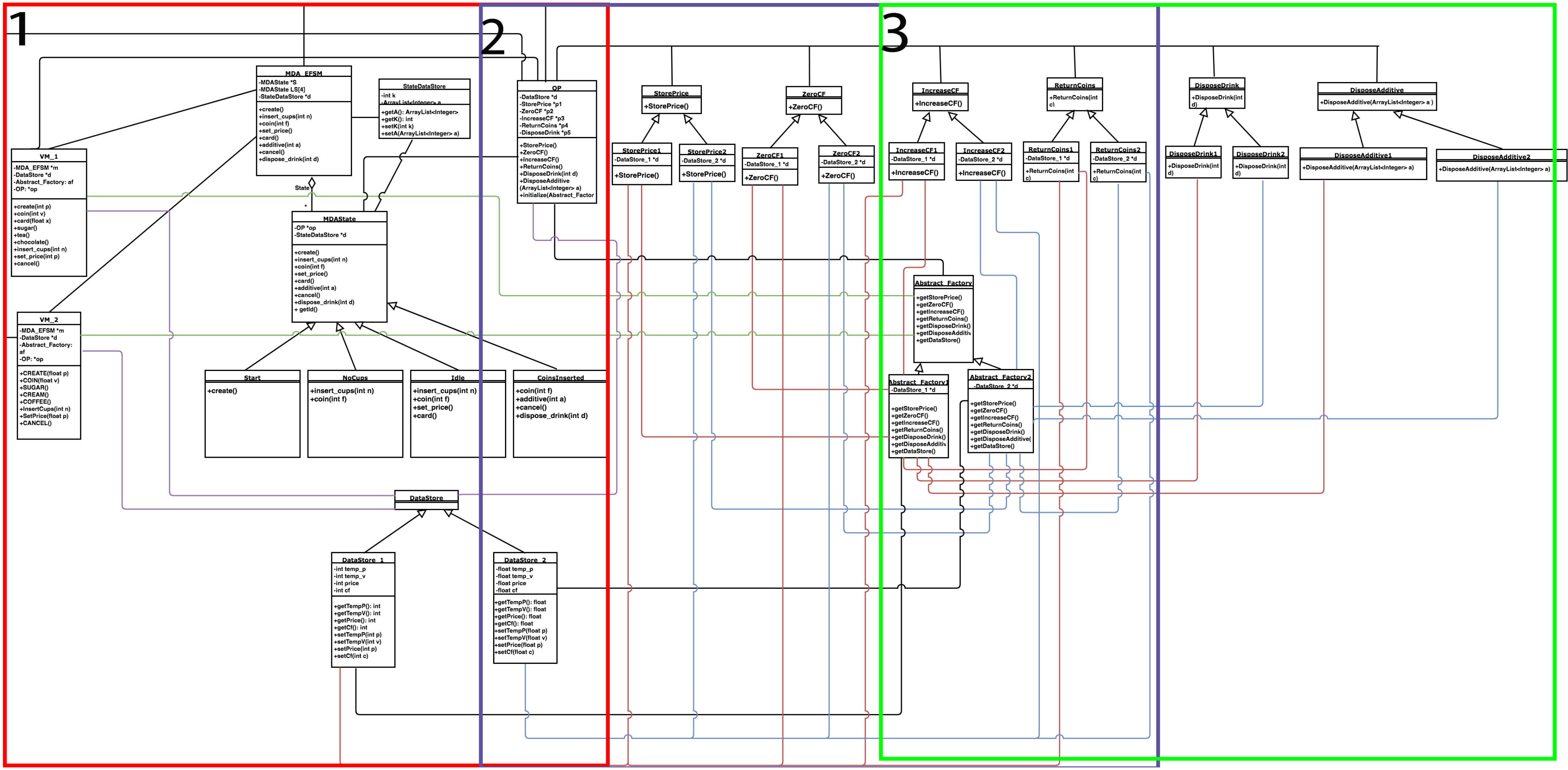
Internal Variables:
int k // keeps track of # of cups
int A[] // a list of additives to be disposed

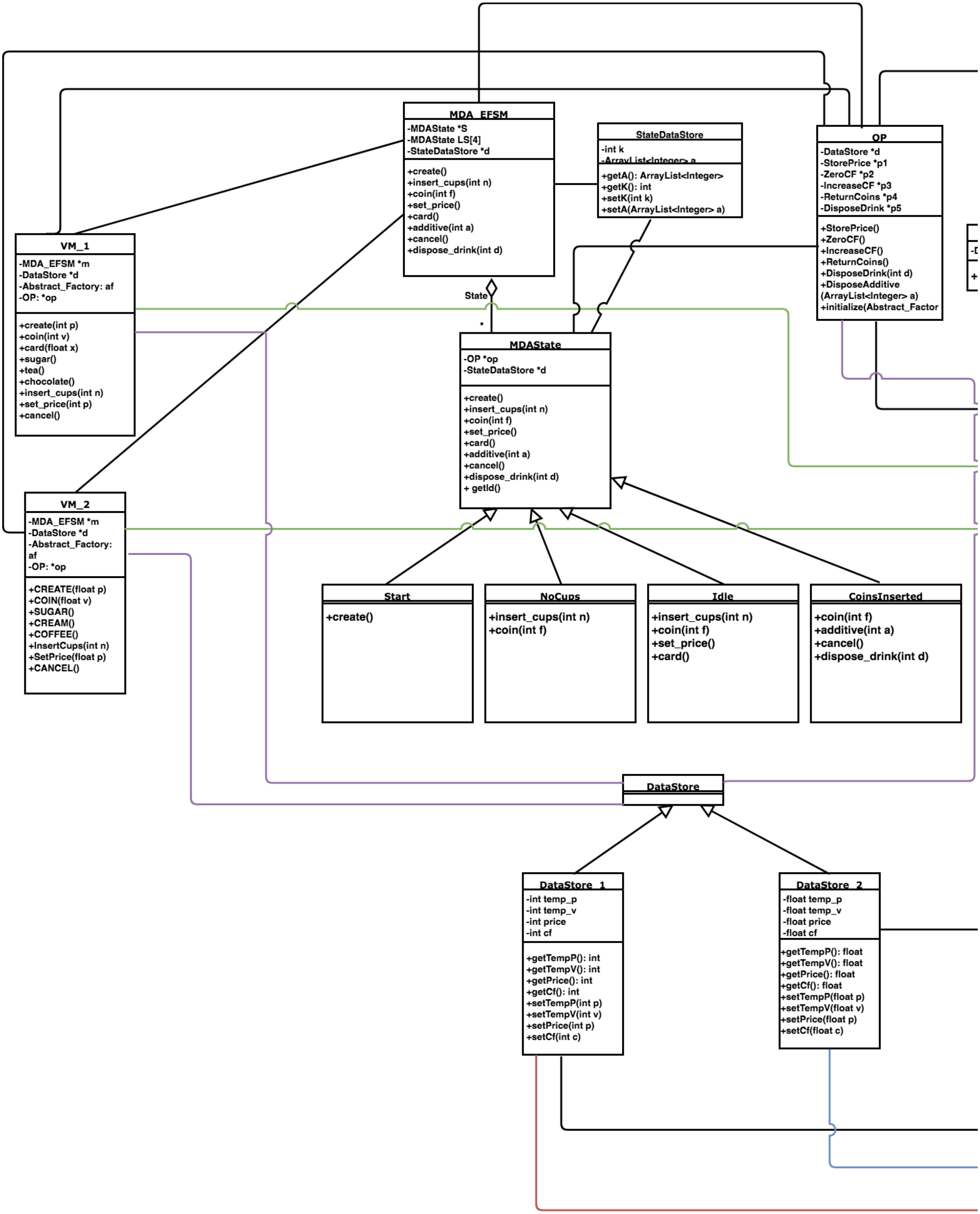

insert_cups(n)[n>0] / k=k+n
set_price / StorePrice
coin(f)[f==0] / IncreaseCF
idle
dispose_drink(d)[k>1] / DisposeDrink(d); DisposeAdditive(A); k=k-1; ZeroCF
insert_cups(n)[n>0] / k=n; ZeroCF
coin(f) / ReturnCoin
card / ZeroCF; A=0
no cups
coin(f)[f==1] / IncreaseCF; A=0
cancel / ReturnCoins; ZeroCF
create / StorePrice
additive(a)[A[a]==1] / A[a]=0
coins inserted
dispose_drink(d)[k<=1] / DisposeDrink(d); DisposeAdditive(A)
coin(f) / ReturnCoin
additive(a)[A[a]==0] / A[a]=1

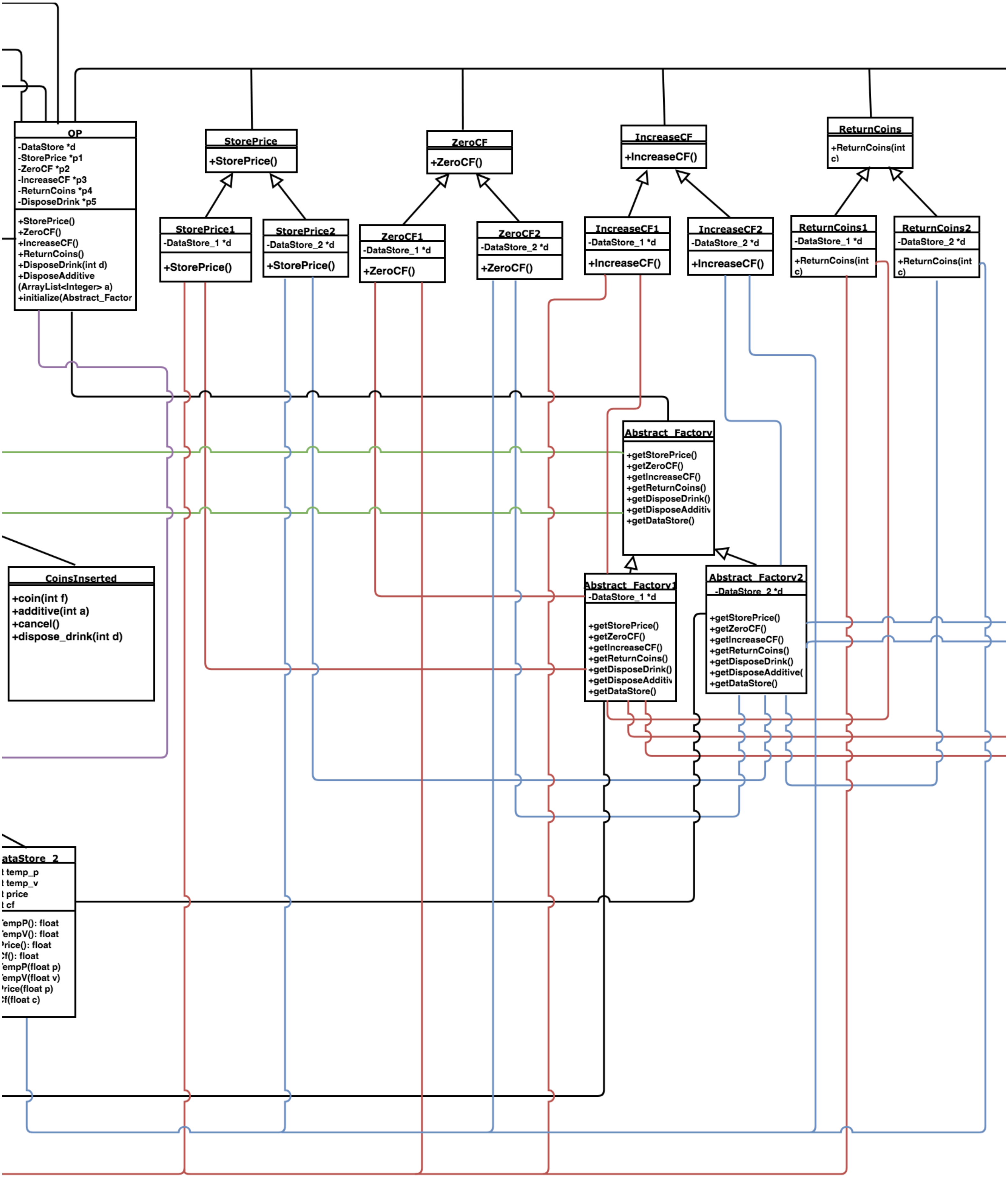
d). Pseudo-code of all operations of Input Processors of Vending Machines: VM-1 and VM-2

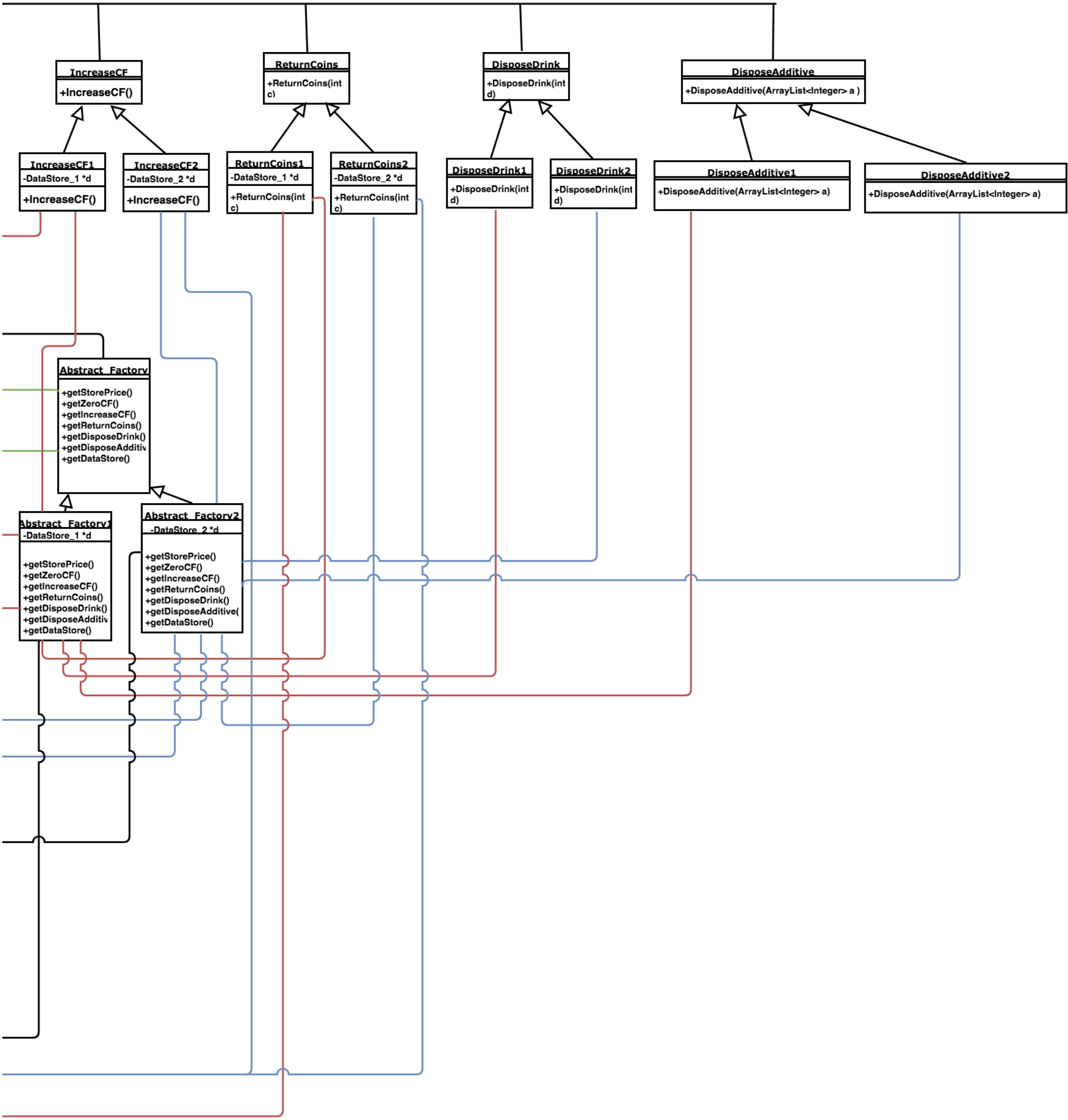
```
CLASS VM-1:

MDA_EFSM *m
DataStore *d
Abstract_Factory *af
OP *op

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

card(float x) {
   if x >= d->price then
      m->card();
   endif
}

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

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

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

tea() {
   m->dispose_drink(1);
}
chocolate() {
   m->dispose_drink(2);
}
sugar() {
   m->additive(1);
}
cancel() {
   m->cancel();}


CLASS VM-2

MDA_EFSM *m
DataStore *d
Abstract_Factory af
OP *op


CREATE(float p) {
   d->temp_p = p;
   m->create();
}
InsertCups(int n) {
   m->insert_cups(n);
}

SetPrice(float p) {
   d->temp_p = p;
   m->set_price();
}
COIN(float v) {
   d->temp_v = v;
   if v + d->cf >= d->price then
      m -> coin(1);
   else
      m->coin(0);
   endif
}
COFFEE() {
   m->dispose_drink(1);
}
SUGAR() {
   m->additive(2);
}
CREAM() {
   m->additive(1);
}
CANCEL() {
   m->cancel();
}
```

2 Class diagram(s) of the MDA of the Vending Machine components.


1
2
3
VM_1
-MDA_EFSM *m
-DataStore *d
-Abstract_Factory: af
-OP: *op
+create(int p)
+coin(int v)
+card(float x)
+sugar()
+tea()
+chocolate()
+insert_cups(int n)
+set_price(int p)
+cancel()
VM_2
-MDA_EFSM *m
-DataStore *d
-Abstract_Factory: af
-OP: *op
+CREATE(float p)
+COIN(float v)
+SUGAR()
+CREAM()
+COFFEE()
+InsertCups(int n)
+SetPrice(float p)
+CANCEL()
MDA_EFSM
-MDAState *S
-MDAState LS[4]
-StateDataStore *d
+create()
+insert_cups(int n)
+coin(int f)
+set_price()
+card()
+additive(int a)
+cancel()
+dispose_drink(int d)
StateDataStore
-int k
-ArrayList<Integer> a
+getA(): ArrayList<Integer>
+getK(): int
+setK(int k)
+setA(ArrayList<Integer> a)
State
*
MDAState
-OP *op
-StateDataStore *d
+create()
+insert_cups(int n)
+coin(int f)
+set_price()
+card()
+additive(int a)
+cancel()
+dispose_drink(int d)
+ getId()
Start
+create()
NoCups
+insert_cups(int n)
+coin(int f)
Idle
+insert_cups(int n)
+coin(int f)
+set_price()
+card()
CoinsInserted
+coin(int f)
+additive(int a)
+cancel()
+dispose_drink(int d)
DataStore
DataStore_1
-int temp_p
-int temp_v
-int price
-int cf
+getTempP(): int
+getTempV(): int
+getPrice(): int
+getCf(): int
+setTempP(int p)
+setTempV(int v)
+setPrice(int p)
+setCf(int c)
DataStore_2
-float temp_p
-float temp_v
-float price
-float cf
+getTempP(): float
+getTempV(): float
+getPrice(): float
+getCf(): float
+setTempP(float p)
+setTempV(float v)
+setPrice(float p)
+setCf(float c)
OP
-DataStore *d
-StorePrice *p1
-ZeroCF *p2
-IncreaseCF *p3
-ReturnCoins *p4
-DisposeDrink *p5
+StorePrice()
+ZeroCF()
+IncreaseCF()
+ReturnCoins()
+DisposeDrink(int d)
+DisposeAdditive
(ArrayList<Integer> a)
+initialize(Abstract_Factor
StorePrice
+StorePrice()
StorePrice1
-DataStore_1 *d
+StorePrice()
StorePrice2
-DataStore_2 *d
+StorePrice()
ZeroCF
+ZeroCF()
ZeroCF1
-DataStore_1 *d
+ZeroCF()
ZeroCF2
-DataStore_2 *d
+ZeroCF()
IncreaseCF
+IncreaseCF()
IncreaseCF1
-DataStore_1 *d
+IncreaseCF()
IncreaseCF2
-DataStore_2 *d
+IncreaseCF()
ReturnCoins
+ReturnCoins(int c)
ReturnCoins1
-DataStore_1 *d
+ReturnCoins(int c)
ReturnCoins2
-DataStore_2 *d
+ReturnCoins(int c)
DisposeDrink
+DisposeDrink(int d)
DisposeDrink1
+DisposeDrink(int d)
DisposeDrink2
+DisposeDrink(int d)
DisposeAdditive
+DisposeAdditive(ArrayList<Integer> a)
DisposeAdditive1
+DisposeAdditive(ArrayList<Integer> a)
DisposeAdditive2
+DisposeAdditive(ArrayList<Integer> a)
Abstract_Factory
+getStorePrice()
+getZeroCF()
+getIncreaseCF()
+getReturnCoins()
+getDisposeDrink()
+getDisposeAdditiv
+getDataStore()
Abstract_Factory1
-DataStore_1 *d
+getStorePrice()
+getZeroCF()
+getIncreaseCF()
+getReturnCoins()
+getDisposeDrink()
+getDisposeAdditiv
+getDataStore()
Abstract_Factory2
-DataStore_2 *d
+getStorePrice()
+getZeroCF()
+getIncreaseCF()
+getReturnCoins()
+getDisposeDrink()
+getDisposeAdditive(
+getDataStore()


MDA_EFSM
-MDAState *S
-MDAState LS[4]
-StateDataStore *d
+create()
+insert_cups(int n)
+coin(int f)
+set_price()
+card()
+additive(int a)
+cancel()
+dispose_drink(int d)
StateDataStore
-int k
-ArrayList<Integer> a
+getA(): ArrayList<Integer>
+getK(): int
+setK(int k)
+setA(ArrayList<Integer> a)
OP
-DataStore *d
-StorePrice *p1
-ZeroCF *p2
-IncreaseCF *p3
-ReturnCoins *p4
-DisposeDrink *p5
+StorePrice()
+ZeroCF()
+IncreaseCF()
+ReturnCoins()
+DisposeDrink(int d)
+DisposeAdditive
(ArrayList<Integer> a)
+initialize(Abstract_Factor
-D
+
VM_1
-MDA_EFSM *m
-DataStore *d
-Abstract_Factory: af
-OP: *op
+create(int p)
+coin(int v)
+card(float x)
+sugar()
+tea()
+chocolate()
+insert_cups(int n)
+set_price(int p)
+cancel()
State
*
MDAState
-OP *op
-StateDataStore *d
+create()
+insert_cups(int n)
+coin(int f)
+set_price()
+card()
+additive(int a)
+cancel()
+dispose_drink(int d)
+ getId()
VM_2
-MDA_EFSM *m
-DataStore *d
-Abstract_Factory:
af
-OP: *op
+CREATE(float p)
+COIN(float v)
+SUGAR()
+CREAM()
+COFFEE()
+InsertCups(int n)
+SetPrice(float p)
+CANCEL()
Start
+create()
NoCups
+insert_cups(int n)
+coin(int f)
Idle
+insert_cups(int n)
+coin(int f)
+set_price()
+card()
CoinsInserted
+coin(int f)
+additive(int a)
+cancel()
+dispose_drink(int d)
DataStore
DataStore_1
-int temp_p
-int temp_v
-int price
-int cf
+getTempP(): int
+getTempV(): int
+getPrice(): int
+getCf(): int
+setTempP(int p)
+setTempV(int v)
+setPrice(int p)
+setCf(int c)
DataStore_2
-float temp_p
-float temp_v
-float price
-float cf
+getTempP(): float
+getTempV(): float
+getPrice(): float
+getCf(): float
+setTempP(float p)
+setTempV(float v)
+setPrice(float p)
+setCf(float c)


OP
-DataStore *d
-StorePrice *p1
-ZeroCF *p2
-IncreaseCF *p3
-ReturnCoins *p4
-DisposeDrink *p5
+StorePrice()
+ZeroCF()
+IncreaseCF()
+ReturnCoins()
+DisposeDrink(int d)
+DisposeAdditive
(ArrayList<Integer> a)
+initialize(Abstract_Factor
StorePrice
+StorePrice()
ZeroCF
+ZeroCF()
IncreaseCF
+IncreaseCF()
ReturnCoins
+ReturnCoins(int c)
StorePrice1
-DataStore_1 *d
+StorePrice()
StorePrice2
-DataStore_2 *d
+StorePrice()
ZeroCF1
-DataStore_1 *d
+ZeroCF()
ZeroCF2
-DataStore_2 *d
+ZeroCF()
IncreaseCF1
-DataStore_1 *d
+IncreaseCF()
IncreaseCF2
-DataStore_2 *d
+IncreaseCF()
ReturnCoins1
-DataStore_1 *d
+ReturnCoins(int c)
ReturnCoins2
-DataStore_2 *d
+ReturnCoins(int c)
Abstract_Factory
+getStorePrice()
+getZeroCF()
+getIncreaseCF()
+getReturnCoins()
+getDisposeDrink()
+getDisposeAdditiv
+getDataStore()
CoinsInserted
+coin(int f)
+additive(int a)
+cancel()
+dispose_drink(int d)
Abstract_Factory1
-DataStore_1 *d
+getStorePrice()
+getZeroCF()
+getIncreaseCF()
+getReturnCoins()
+getDisposeDrink()
+getDisposeAdditiv
+getDataStore()
Abstract_Factory2
-DataStore_2 *d
+getStorePrice()
+getZeroCF()
+getIncreaseCF()
+getReturnCoins()
+getDisposeDrink()
+getDisposeAdditive(
+getDataStore()
ataStore_2
t temp_p
t temp_v
t price
t cf
'empP(): float
'empV(): float
'rice(): float
:f(): float
'empP(float p)
'empV(float v)
'rice(float p)
:f(float c)


IncreaseCF
+IncreaseCF()
ReturnCoins
+ReturnCoins(int c)
DisposeDrink
+DisposeDrink(int d)
DisposeAdditive
+DisposeAdditive(ArrayList<Integer> a)
IncreaseCF1
-DataStore_1 *d
+IncreaseCF()
IncreaseCF2
-DataStore_2 *d
+IncreaseCF()
ReturnCoins1
-DataStore_1 *d
+ReturnCoins(int c)
ReturnCoins2
-DataStore_2 *d
+ReturnCoins(int c)
DisposeDrink1
+DisposeDrink(int d)
DisposeDrink2
+DisposeDrink(int d)
DisposeAdditive1
+DisposeAdditive(ArrayList<Integer> a)
DisposeAdditive2
+DisposeAdditive(ArrayList<Integer> a)
Abstract_Factory
+getStorePrice()
+getZeroCF()
+getIncreaseCF()
+getReturnCoins()
+getDisposeDrink()
+getDisposeAdditiv
+getDataStore()
Abstract_Factory1
-DataStore_1 *d
+getStorePrice()
+getZeroCF()
+getIncreaseCF()
+getReturnCoins()
+getDisposeDrink()
+getDisposeAdditiv
+getDataStore()
Abstract_Factory2
-DataStore_2 *d
+getStorePrice()
+getZeroCF()
+getIncreaseCF()
+getReturnCoins()
+getDisposeDrink()
+getDisposeAdditive(
+getDataStore()

3 The purpose of each class and the responsibility of each operation supported by each class

a). CLASS MDA_EFSM

operations

create()
// Call *create* function of the current state and if the current state id is 0 then change state to "NoCups"

insert_cups(int n)
// If the number of cups to insert is greater than 0 then call *insert_cups* function and if the current state id is 1 then change state to "Idle"

coin(int f)
// Call *coin* function of the current state. If f is equal to 1 and the current state id is 2 then change state to "CoinsInserted"

card(float x)
// Call *card* function of the current state. If the current state id is 2 then change state to "CoinsInserted"

cancel()
// Call *cancel* function of the current state

set_price()
// Call *set_price* function of the current state

dispose_drink(int d)
// Call current state's *dispose_drink* function. If the number of cups is greater than 1
and the current state id is 3 then change state to "Idle" else if the number of cups is less than or equal to 1 then change state to "NoCups"

additive(int a)
// Call *additive* function to add or remove additive a

b). State pattern

CLASS MDAState
operations
create()
// abstract operation
insert_cups(int n)
// abstract operation
coin(int f)
// abstract operation

```
set_price()
 // abstract operation
card()
//abstract operation
additive(int a)
//abstract operation
cancel()
//abstract operation
dispose_drink(int d)
// abstract operation
getId()
// return state id


CLASS Start                 // state pattern, state Start, state id: 0

operations

create()                    // Call function op->StorePrice()

CLASS NoCups                // state pattern, state NoCups,state id: 1

operations

coin(int f)                 // Call function op->ReturnCoins(0), return temp_v coins

insert_cups(int n)          // Call functions d->setK(n) and op->ZeroCF()

CLASS Idle                  // state pattern, state Idle, state id: 2

operations

coin(int f)                 // Call op->IncreaseCF()

insert_cups(int n)
// If the number of cups to insert is greater than 0 then add the number of cups by n

set_price()                 // Call function op->StorePrice(), store price

card()                      // Call function op->ZeroCF(), set cf to zero

CLASS CoinsInserted         // state pattern, state CoinsInserted, state id: 3

operations

coin(int f)                 // Call function op->ReturnCoins(0), return temp_v coins

additive(int a)             // Remove additive or add additive to additive list
```

dispose_drink(int d)
//Call functions op->DisposeDrink(d) and op->DisposeAdditive(this.d->getA()). If the number of cups greater than 1, Reduce the number of cups by one and zero Cumulative Fund cf

cancel()
//Call function op->ReturnCoins(1)(return cf coins) and zero Cumulative Fund cf

```
Class StateDataStore:       //used to store data used by MDA_EFSM and State patterns
numberOfCups
ArrayList<Integer> additive                       // additive list
getAdditive()                                     // return additive list
setAdditive(ArrayList<Integer> additive)          //set additive list
getNumberOfCups()                                 // return numberOfCups
setNumberOfCups(int numberOfCups)                 //set numberOfCups
```

c). Abstract Factory pattern

```
CLASS Abstract_Factory
operation

getStorePrice()                    // abstract operation
getZeroCF()                        // abstract operation
getIncreaseCF()                    // abstract operation
getReturnCoins()                   // abstract operation
getDisposeDrink()                  // abstract operation
getDisposeAdditive()               // abstract operation
getDataStore()                     // abstract operation


CLASS Abstract_Factory1

DataStore *d;

getStorePrice()                    //Return new StorePrice1

getZeroCF()                        //Return new ZeroCF1

getIncreaseCF()                    //Return new IncreaseCF1

getReturnCoins()                   //Return new ReturnCoins1

getDisposeDrink()                  //Return new DisposeDrink1;

getDisposeAdditive()               //Return new DisposeAdditive1

getDataStore()                     //Return new DataStore_1


CLASS Abstract_Factory2:

DataStore *d;

getStorePrice()                    //Return new StorePrice2

getZeroCF()                        //Return new ZeroCF2

getIncreaseCF()                    //Return new IncreaseCF2

getReturnCoins()                   //Return new ReturnCoins2

getDisposeDrink()                  //Return new DisposeDrink2
```

getDisposeAdditive() //Return new DisposeAdditive2
getDataStore() //Return new DataStore_2

d) Strategy pattern

CLASS StorePrice:
StorePrice()

CLASS StorePrice1:

DataStore_1 *d;

StorePrice() // Store temp_p to price in DataStore_1 and reset temp_p

CLASS StorePrice2:

DataStore_2 *d;

StorePrice() // Store temp_p to price in DataStore_2 and reset temp_p

CLASS ZeroCF:
ZeroCF()

CLASS ZeroCF1:
DataStore_1 *d;
ZeroCF1() // Zero Cumulative Fund cf in DataStore_1

CLASS ZeroCF2:
DataStore_2 *d;
ZeroCF2() // Zero Cumulative Fund cf in DataStore_2

CLASS ReturnCoins:
ReturnCoins(int c)

CLASS ReturnCoins1:
DataStore_1 *d
ReturnCoins(int c) // Return coins inserted for a drink. If c is equal to 0 then return temp_v coins else if c is equal to 1 then return cf coins

CLASS ReturnCoins2:
DataStore_2 *d
ReturnCoins(int c) // Return coins inserted for a drink. If c is equal to 0 then return temp_v coins else if c is equal to 1 then return cf coins

CLASS IncreaseCF:
IncreaseCF()

CLASS IncreaseCF1:
DataStore_1 *d
IncreaseCF() //Increase Cumulative Fund cf in DataStore_1

CLASS IncreaseCF2:
DataStore_2 *d
IncreaseCF() //Increase Cumulative Fund cf in DataStore_2

CLASS DisposeDrink:
DisposeDrink(int d)

CLASS DisposeDrink1:
DisposeDrink(int d) // id == 1 dispose tea, id == 2 dispose chocolate

CLASS DisposeDrink2:
DisposeDrink(int d) // id == 1 dispose coffee

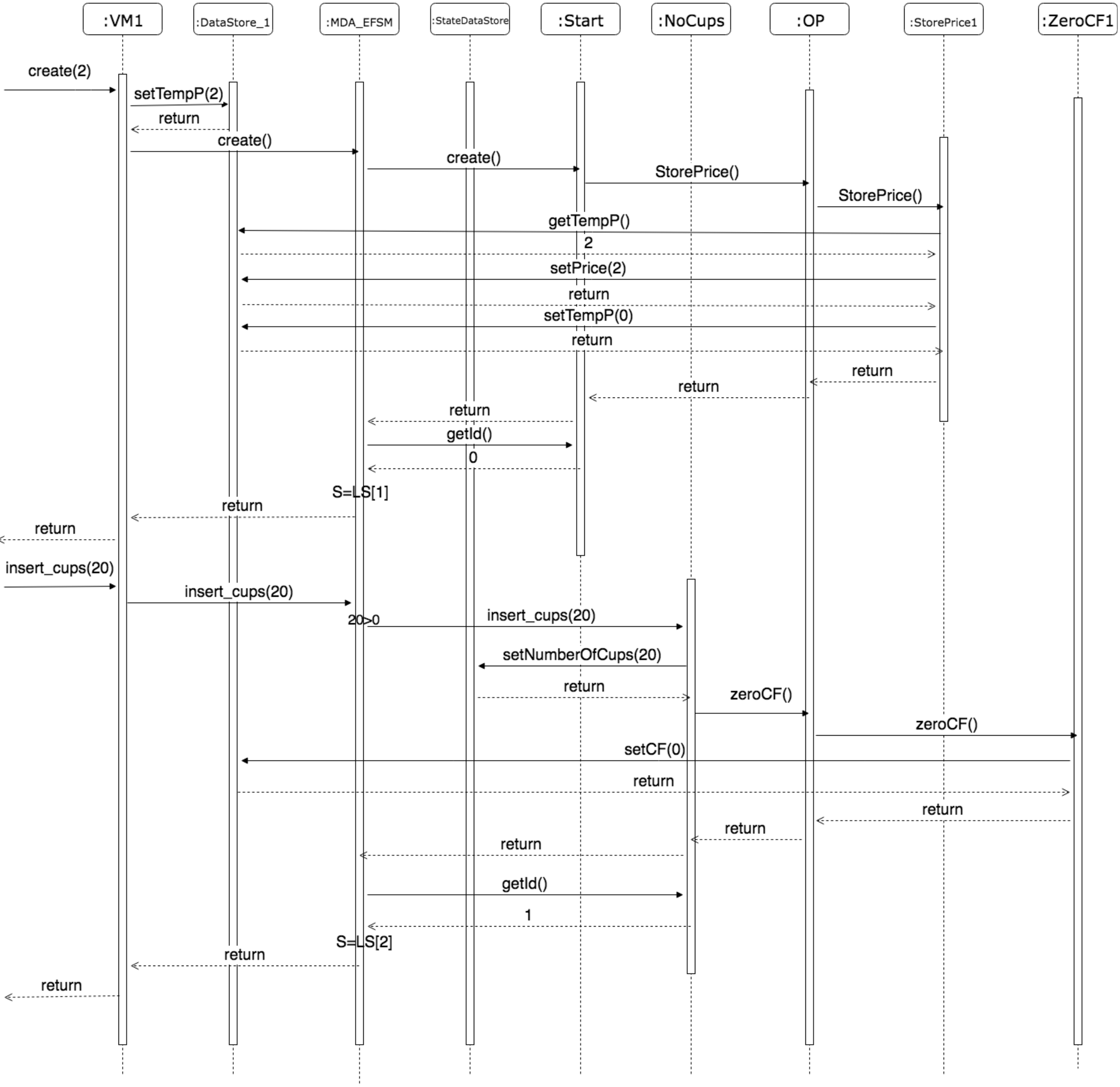
CLASS DisposeAdditive:
DisposeAdditive()

CLASS DisposeAdditive1:
DisposeAdditive(ArrayList a) //If list a!=null then add sugar

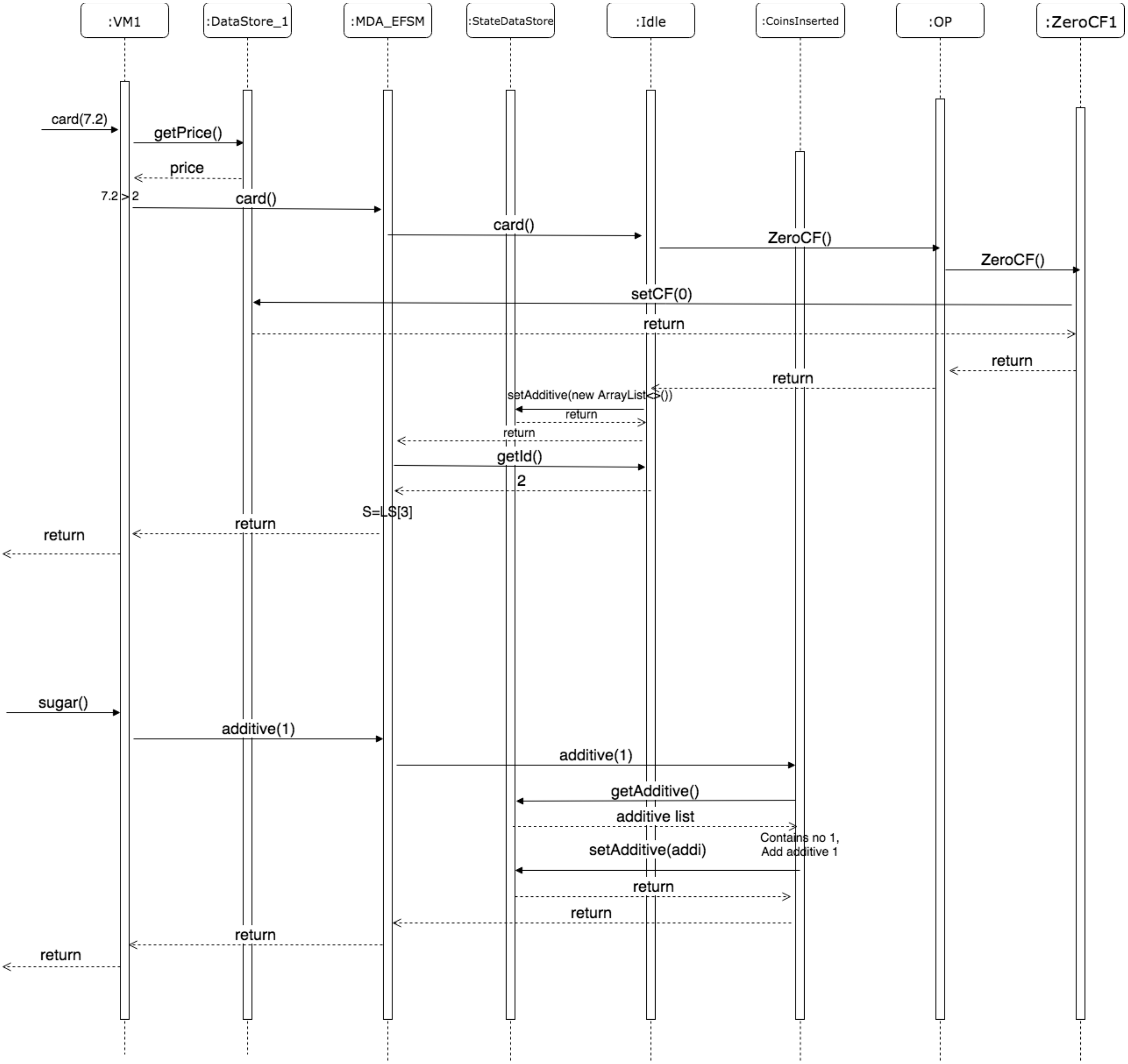
CLASS DisposeAdditive2:
DisposeAdditive(ArrayList a)
//list a contains 1, add cream. list a contains 2, add sugar

4 Sequence diagrams for two Scenarios

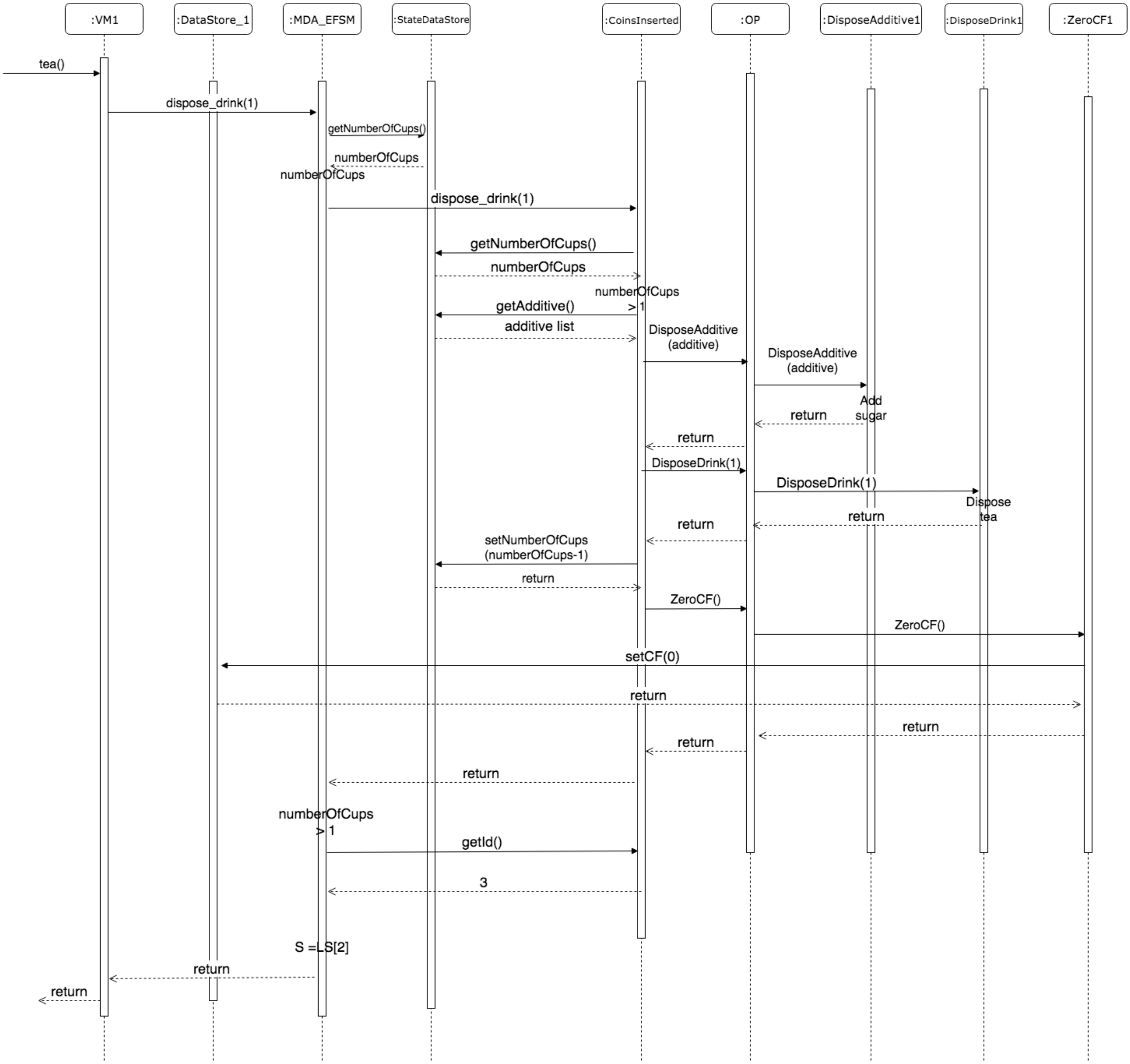
a. Scenario-One VM1 Operations: create(2), insert_cups(20)



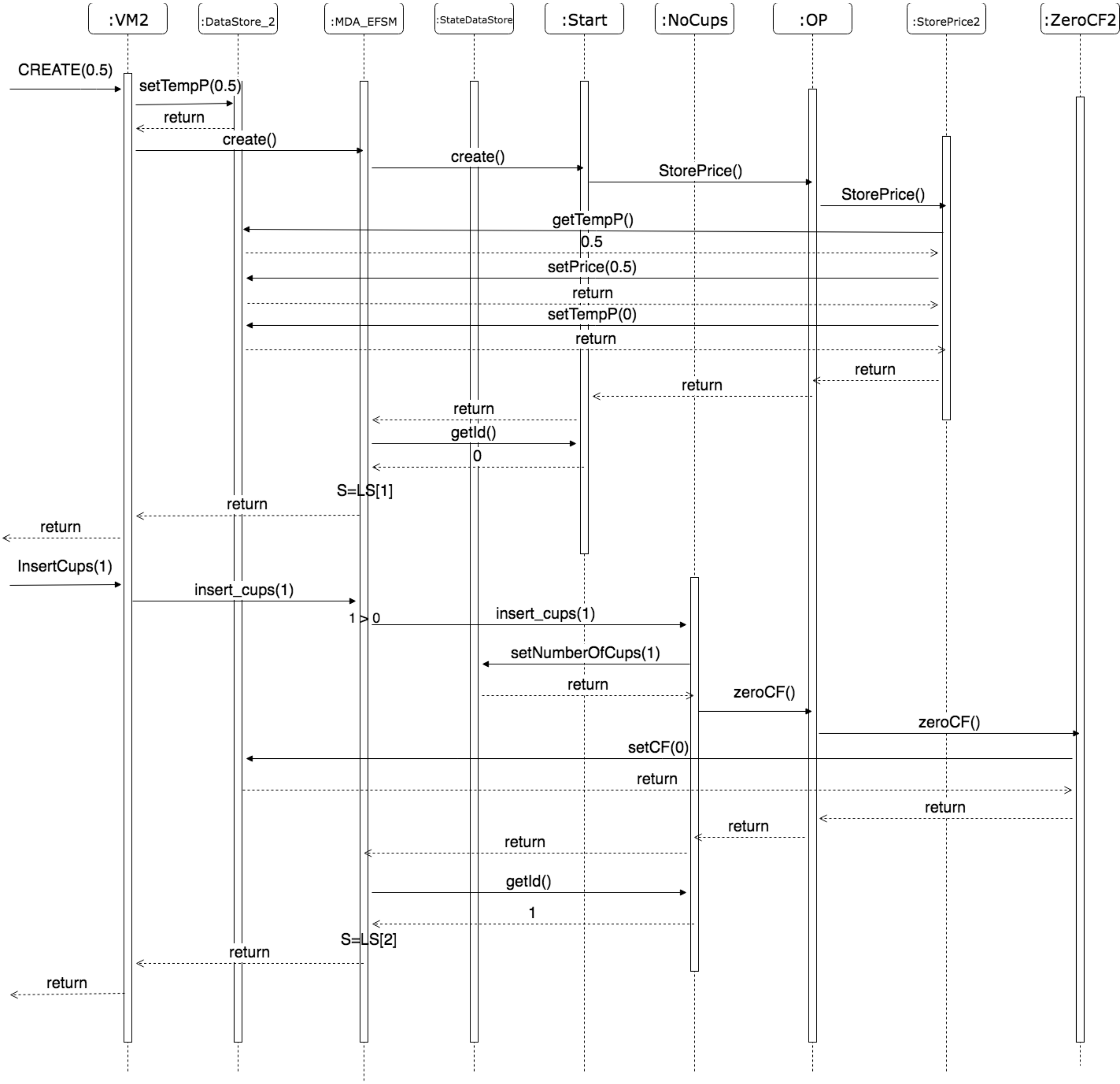
Operations: card(7.2), sugar()



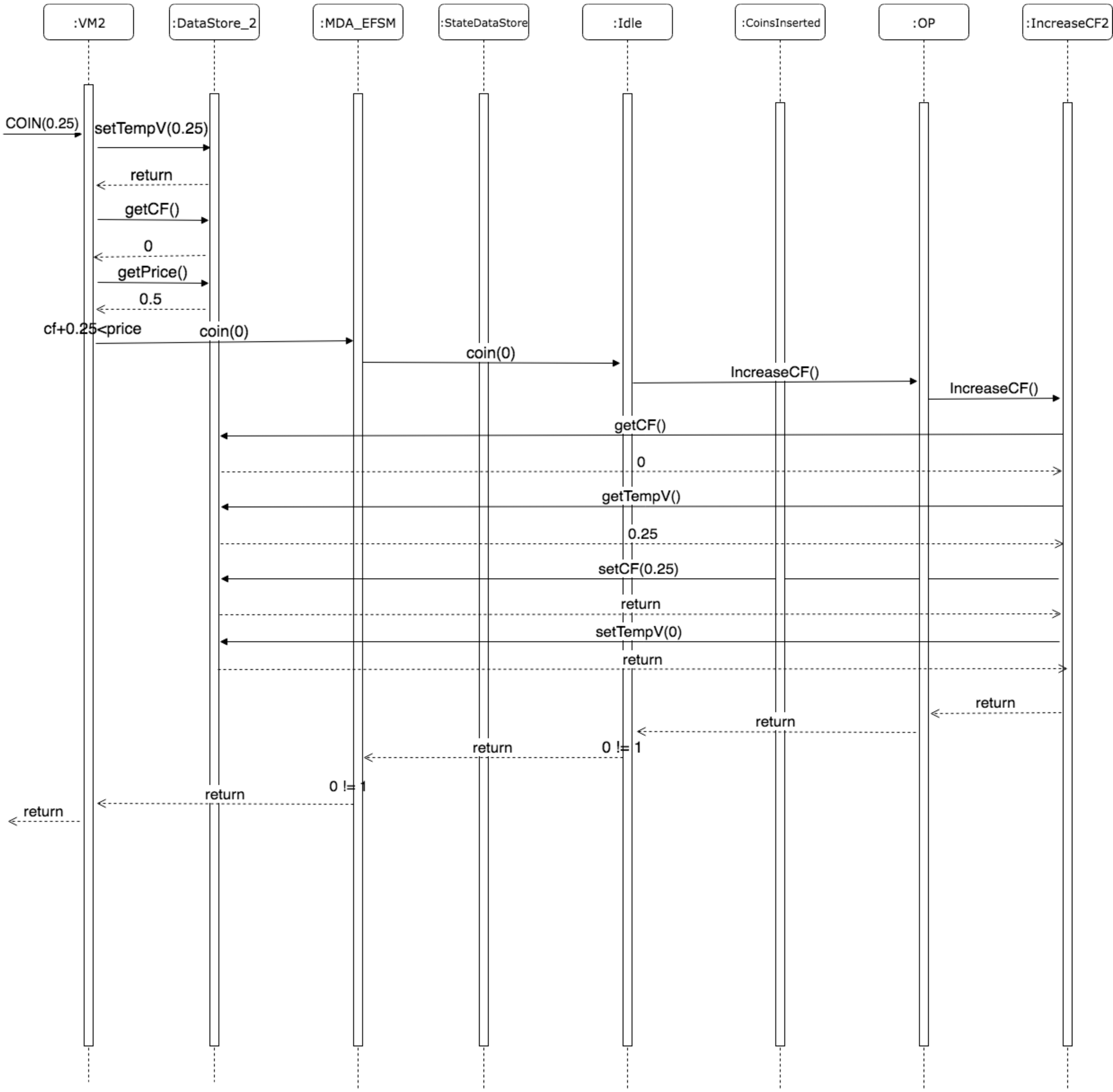
Operations: tea()



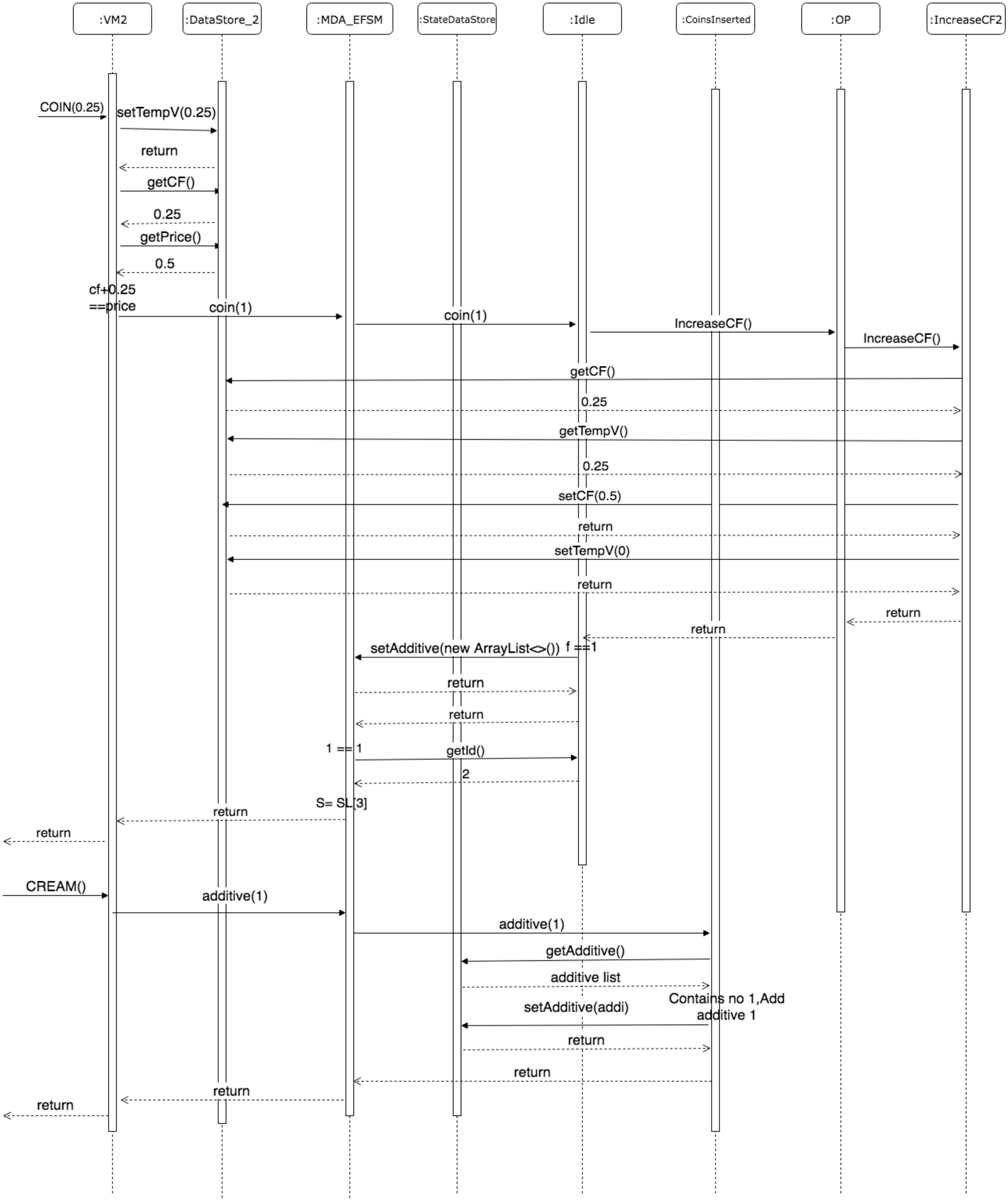
b. Scenario-Two VM2 Operations: CREATE(0.5), InsertCups(1)



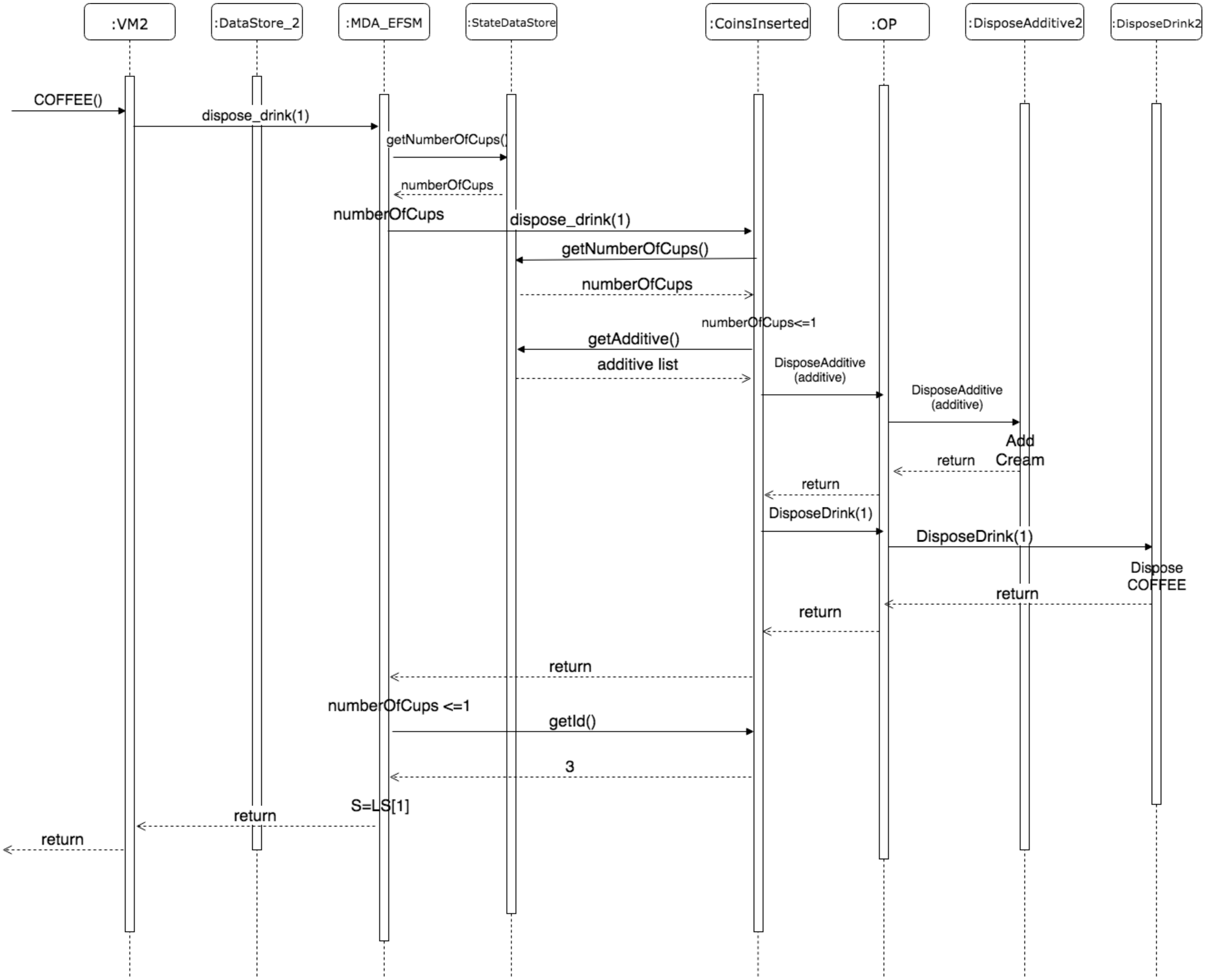
Operations: COIN(0.25)



Operations: COIN(0.25), CREAM()



Operations: COFFEE()


:VM2
:DataStore_2
:MDA_EFSM
:StateDataStore
:CoinsInserted
:OP
:DisposeAdditive2
:DisposeDrink2
COFFEE()
dispose_drink(1)
getNumberOfCups()
numberOfCups
numberOfCups
dispose_drink(1)
getNumberOfCups()
numberOfCups
numberOfCups<=1
getAdditive()
additive list
DisposeAdditive (additive)
DisposeAdditive (additive)
Add Cream
return
return
DisposeDrink(1)
DisposeDrink(1)
Dispose COFFEE
return
return
return
numberOfCups <=1
getId()
3
S=LS[1]
return
return