CS 586 – Software Systems Architecture

Project Report for design of Vending Machine Components Using Model Driven Architecture

Submitted By
Adarsh Mathad Vijayakumar
CWID: A20424847

1. MDA-EFSM model for the Vending Machine components

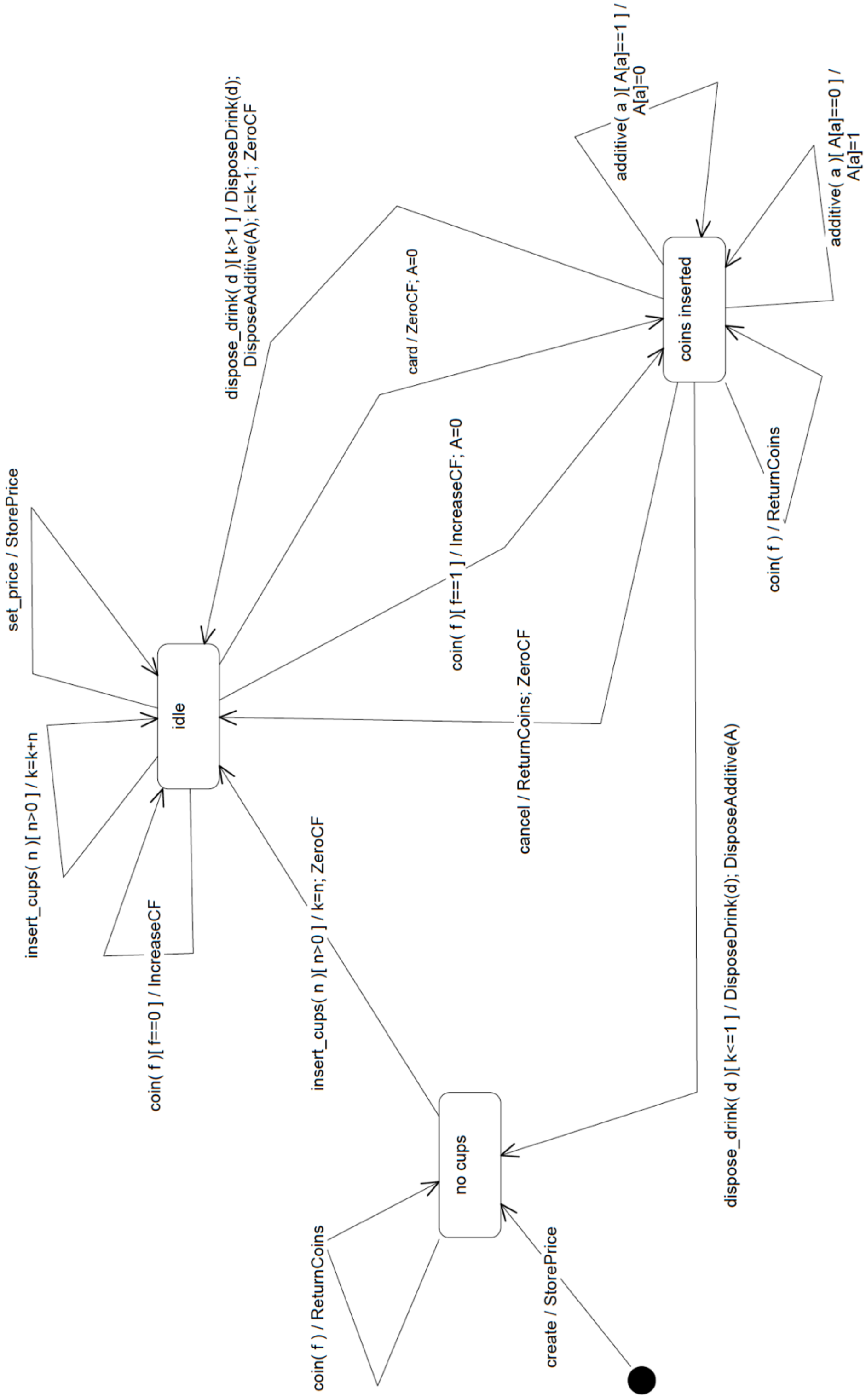
a) List of Meta Events for the MDA EFSM :

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()
6. set_price()
7. dispose_drink(int d) // d represents a drink id
8. additive(int a) // a represents additive id

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

1. StorePrice() //copy the price into a variable in data store
2. ZeroCF() // zero Cumulative Fund cf. i.e cf = 0
3. IncreaseCF() // increase Cumulative Fund cf by cf + v
4. ReturnCoins() // 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


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

Internal variable: int k – keeps track of the number of cups
Internal variable int A[] – List of additives to be disposed

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

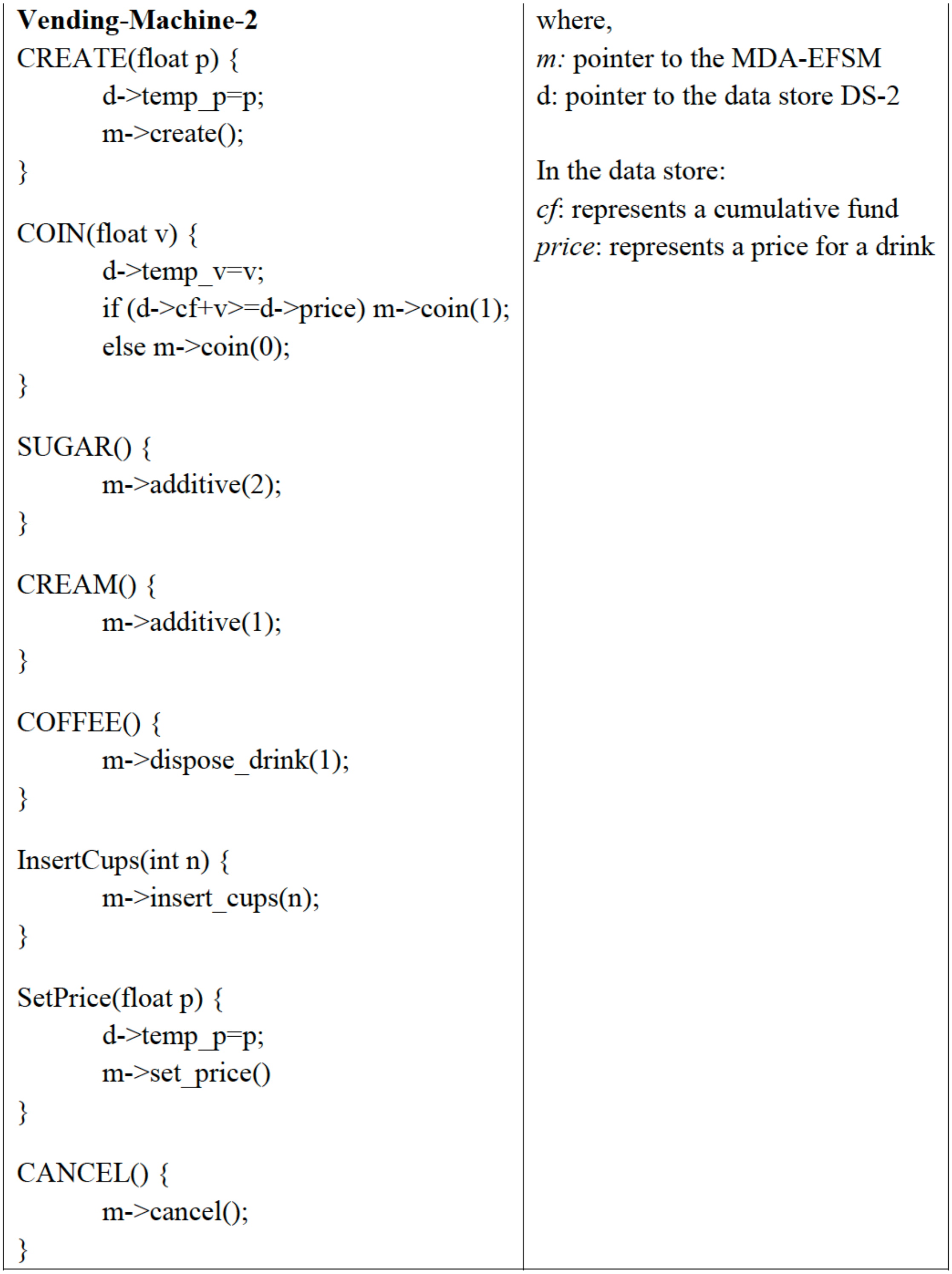
Vending-Machine-1

```
create(int 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(float x) {
        if (x>=d->price) m->card();
}
sugar() {
        m->additive(1);
}

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

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

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

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

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

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 a price for a drink

Pseudocode of the vending machine 2 class:

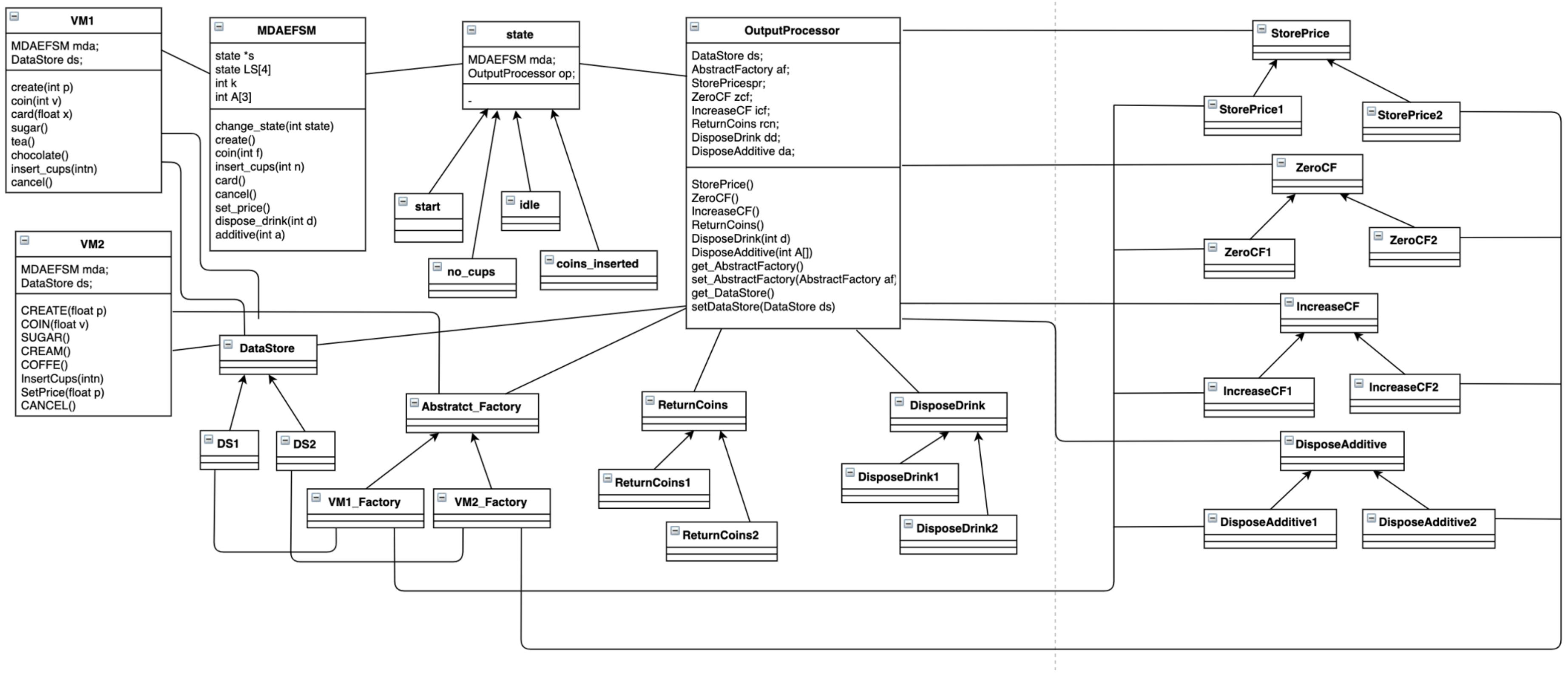


```
Vending-Machine-2
CREATE(float p) {
        d->temp_p=p;
        m->create();
}

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

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

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

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

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

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

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

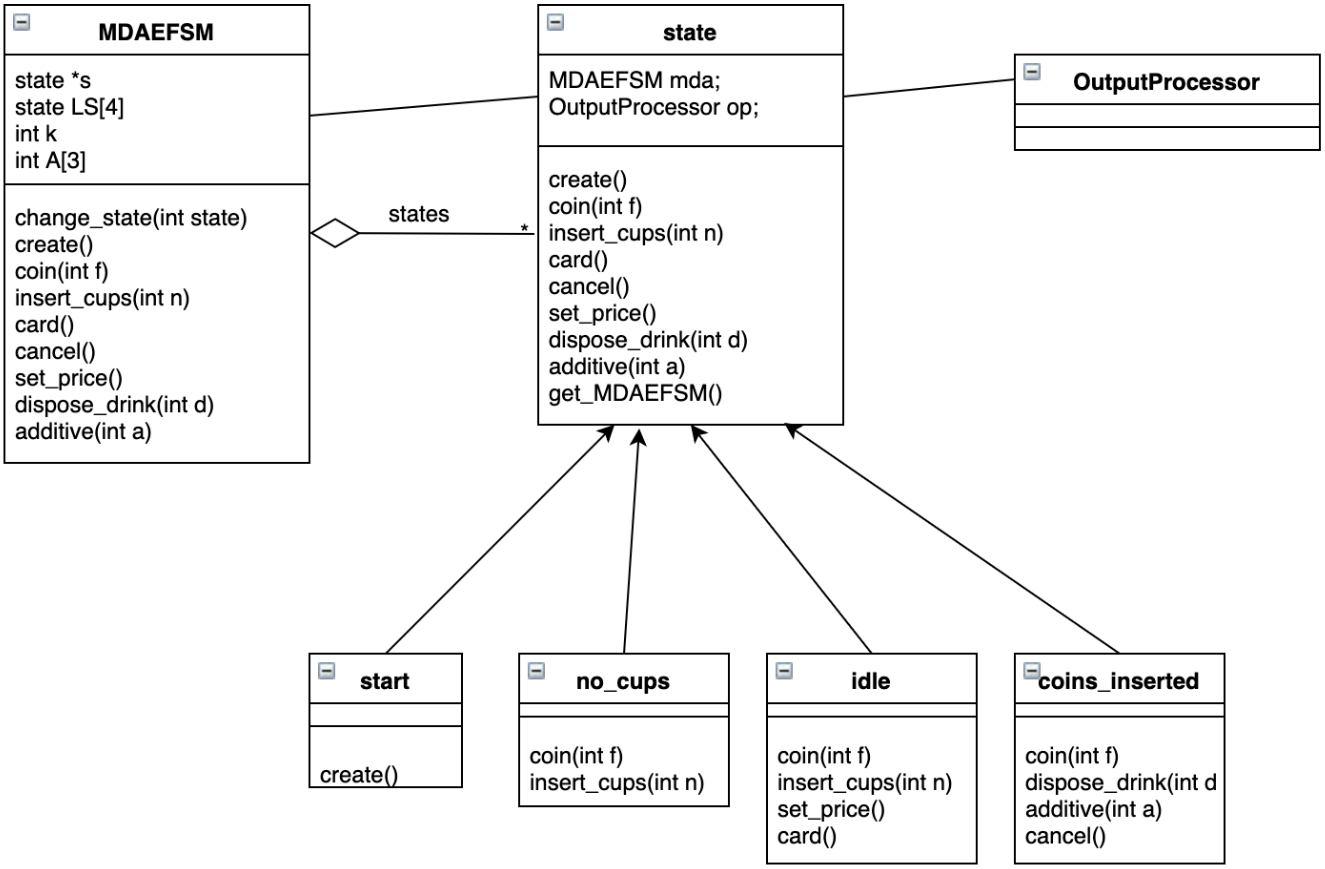
where,
m: pointer to the MDA-EFSM
d: pointer to the data store DS-2

In the data store:
cf: represents a cumulative fund
price: represents a price for a drink

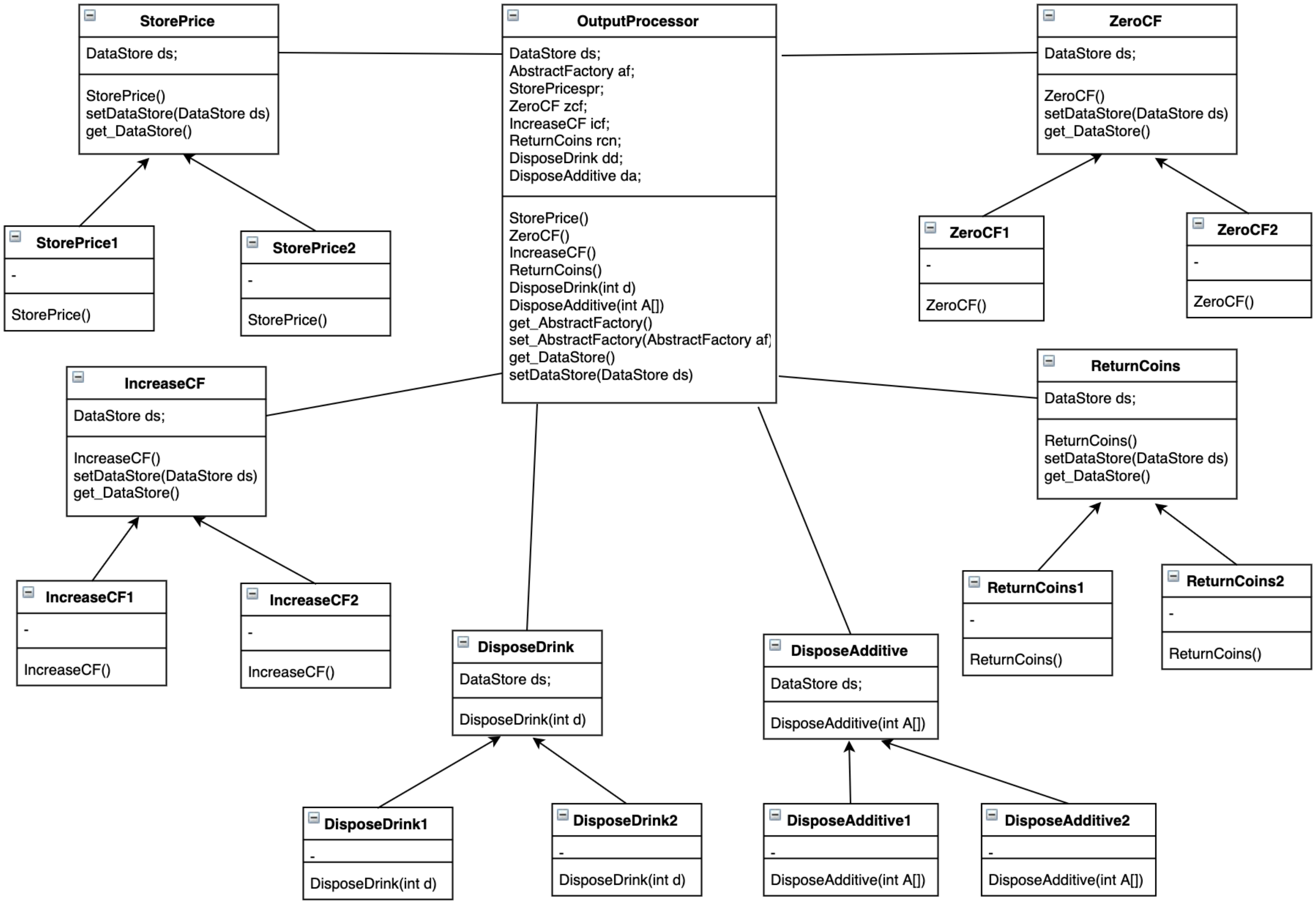
2. Class diagram(s) of the MDA of the Vending Machine components


VM1
MDAEFSM mda;
DataStore ds;
create(int p)
coin(int v)
card(float x)
sugar()
tea()
chocolate()
insert_cups(intn)
cancel()
VM2
MDAEFSM mda;
DataStore ds;
CREATE(float p)
COIN(float v)
SUGAR()
CREAM()
COFFE()
InsertCups(intn)
SetPrice(float p)
CANCEL()
MDAEFSM
state *s
state LS[4]
int k
int A[3]
change_state(int state)
create()
coin(int f)
insert_cups(int n)
card()
cancel()
set_price()
dispose_drink(int d)
additive(int a)
state
MDAEFSM mda;
OutputProcessor op;
-
start
idle
no_cups
coins_inserted
OutputProcessor
DataStore ds;
AbstractFactory af;
StorePricespr;
ZeroCF zcf;
IncreaseCF icf;
ReturnCoins rcn;
DisposeDrink dd;
DisposeAdditive da;
StorePrice()
ZeroCF()
IncreaseCF()
ReturnCoins()
DisposeDrink(int d)
DisposeAdditive(int A[])
get_AbstractFactory()
set_AbstractFactory(AbstractFactory af)
get_DataStore()
setDataStore(DataStore ds)
DataStore
DS1
DS2
Abstratct_Factory
VM1_Factory
VM2_Factory
ReturnCoins
ReturnCoins1
ReturnCoins2
DisposeDrink
DisposeDrink1
DisposeDrink2
StorePrice
StorePrice1
StorePrice2
ZeroCF
ZeroCF1
ZeroCF2
IncreaseCF
IncreaseCF1
IncreaseCF2
DisposeAdditive
DisposeAdditive1
DisposeAdditive2

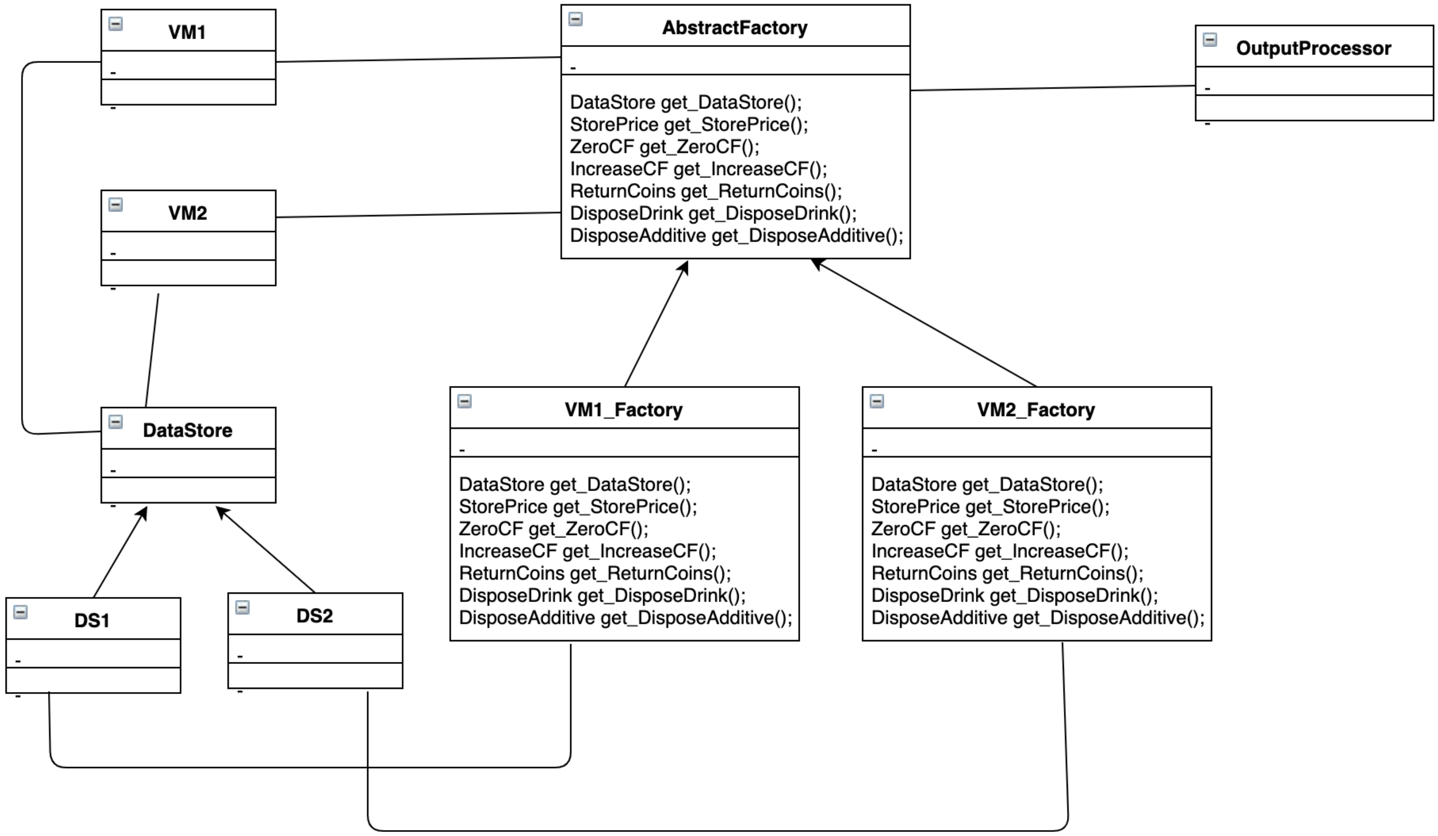
Class diagram for the MDA EFSM model (State Pattern)



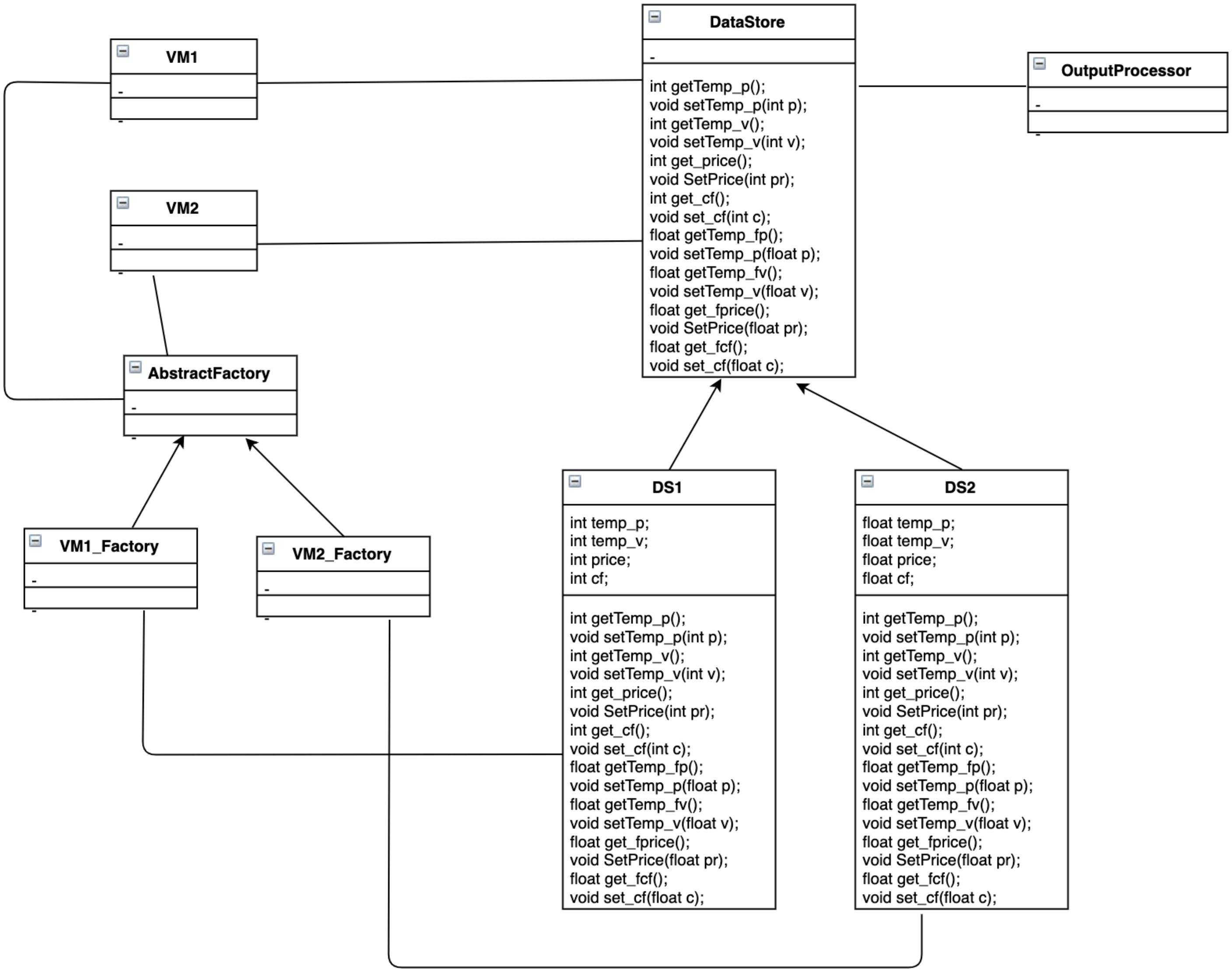
Class diagram for the Output Processor (Strategy Pattern)



Class diagram for the Abstract Factory Pattern


VM1
-
-
AbstractFactory
-
DataStore get_DataStore();
StorePrice get_StorePrice();
ZeroCF get_ZeroCF();
IncreaseCF get_IncreaseCF();
ReturnCoins get_ReturnCoins();
DisposeDrink get_DisposeDrink();
DisposeAdditive get_DisposeAdditive();
OutputProcessor
-
-
VM2
-
-
DataStore
-
-
VM1_Factory
-
DataStore get_DataStore();
StorePrice get_StorePrice();
ZeroCF get_ZeroCF();
IncreaseCF get_IncreaseCF();
ReturnCoins get_ReturnCoins();
DisposeDrink get_DisposeDrink();
DisposeAdditive get_DisposeAdditive();
VM2_Factory
-
DataStore get_DataStore();
StorePrice get_StorePrice();
ZeroCF get_ZeroCF();
IncreaseCF get_IncreaseCF();
ReturnCoins get_ReturnCoins();
DisposeDrink get_DisposeDrink();
DisposeAdditive get_DisposeAdditive();
DS1
-
-
DS2
-
-

Class diagram for Data Store



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

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

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

Class Driver	
Purpose	This class is used to run the GasPump components.
Operations	
main(String[] args)	This method is used to run the GasPump components.

Class VM1	
Purpose	This class represents Vending Machine1 and supports all the operations provided by Vending Machine1. This is a part of input processor.
Attributes	
MDAEFSM *mda	Pointer to MDAEFSM object.
DataStore *ds	Pointer to DataStore object.
Operations	
create(int p)	starts a vending machine application, where p is an initial price of a drink
coin(int v)	a coin with value v is inserted
card(float x)	credit card is swiped, where x is an available fund
sugar()	sugar button is pressed
tea()	tea button is pressed
chocolate()	chocolate button is pressed
insert_cups(int n)	n cups are inserted into the vending machine
set_price(int p)	new price of a cup of tea/chocolate is set to value p
Cancel()	cancel selection for a cup of tea or hot chocola

Class VM2	
Purpose	This class represents Vending Machine2 and supports all the operations provided by Vending Machine2. This is a part of input processor.
Attributes	
MDAEFSM *mda	Pointer to MDAEFSM object.
DataStore *ds	Pointer to DataStore object.
Operations	
CREATE(float p)	starts a vending machine application, where p is an initial price of a drink
COIN(float v)	a coin with value v is inserted
SUGAR()	sugar button is pressed

CREAM()	cream button is pressed
COFFEE()	coffee button is pressed
InsertCups(int n)	n cups are inserted into the vending machine
SetPrice(float p)	new price of a cup of coffee is set to value p
CANCEL()	cancel selection for a cup of coffee

Class MDAEFSM	
Purpose	This class represents the MDAEFSM. It supports the MDAEFSM events. This class is also a context class of State Pattern.
Attributes	
State[] LS	Stores the objects of different state classes.
State *state	Pointer to current state of MDAEFSM.
int A[]	Array to manage the selection, de-selection of additives
Operations	
Change_State(int state)	This method is used to change state.
insert_cups(int n)	This method is used to insert the cups into the vending machine
coin(int f)	This method accepts a coin of value f
card()	This method accepts a card
cancel()	This method cancels the selection done and return to the previous state
set_price()	This method is used to set a new price for a drink
dispose_drink(int d)	This method disposes the drink d
additive(int a)	This method is used to select an additive

Class State	
Purpose	This class is state class of State Pattern. It represents the state for MDAEFSM.
Attributes	
MDAEFSM *mda	Pointer to MDAEFSM object.
OutputProcessor *op	Pointer to OutputProcessor class object.
Abstract Operations	
create()	This method starts the vending machine
coin(int f)	This method accepts a coin of value f
insert_cups(int n)	This method is used to insert the cups into the vending machine
set_price()	This method is used to set a new price for a drink
card()	This method accepts a card
additive(int a)	This method is used to select an additive
cancel()	This method cancels the selection done and return to the previous
dispose_drink(int d)	This method disposes the drink d
Operations	
get_MDAEFSM()	This method is used to get MDAEFSM object.
set_MDAEFSM(MDAEFSM mda)	This method is used to set MDAEFSM object.
get_OutputProcessor()	This method is used to get OutputProcessor object.
set_OutputProcessor(OutputProcessor op)	This method is used to set OutputProcessor object.

Class Start	
Purpose	This is a subclass of State class and represents Start state.
Operation	
create()	This method starts the vending machine

Class no_cups	
Purpose	This is a subclass of State class and represents no_cups state.
Operation	
coin(int f)	This method returns the coins
insert_cups(int n)	Inserts cups into the machine. Executes the ZeroCF() operation

Class idle	
Purpose	This is a subclass of State class and represents idle state.
Operation	
coin(int f)	This method used to pay for the drink. It executes IncreaseCF() operation.
insert_cups(int n)	Inserts cups into the machine.
set_price()	Used to set the price of the drink to a new price
Card()	Used to pay for the drink by card. It executes ZeroCF() operation

Class coins_inserted	
Purpose	This is a subclass of State class and represents coins_inserted state.
Operation	
coin(int f)	This operation returns the coins
Additive(int a)	This operation is used to select or de-select the additives
Cancel()	This method cancels the selection. It executes the ZeroCF() and ReturnCoins() operations
Dispose_drink(int d)	This operation disposes the drink selected, which is represented by parameter d

Class OutputProcessor	
Purpose	This class represents Output processor and used to execute actions.
Attributes	
DataStore *ds	Pointer to DataStore object.
AbstractFactory *af	Pointer to AbstractFactory object.
StorePrice spr	Pointer to StorePrice object
ZeroCF zcf	Pointer to ZeroCF object
IncreaseCF icf	Pointer to IncreaseCF object
ReturnCoins rcn	Pointer to ReturnCoins object
DisposeDrink dd	Pointer to DisposeDrink object
DisposeAdditive da	Pointer to DisposeAdditive object

Operations	
StorePrice()	This operation is used to execute the StorePrice action. It creates the object StorePrice using the AbstractFactory class. This executes the StorePrice() operation of the StorePrice class
ZeroCF()	This operation is used to execute the ZeroCF action. It creates the ZeroCF object using the AbstractFactory class. This executes the ZeroCF() operation of the ZeroCF class
IncreaseCF()	This operation is used to execute the IncreaseCF action. It creates the IncreaseCF object using the AbstractFactory class. This executes the IncreaseCF() operation of the IncreaseCF class
ReturnCoins()	This operation is used to execute the ReturnCoins action. It creates the ReturnCoins object using the AbstractFactory class. This executes the ReturnCoins() operation of the ReturnCoins class
DisposeDrink(int d)	This operation is used to execute the DisposeDrink action. It creates the DisposeDrink object using the AbstractFactory class. This executes the DisposeDrink() operation of the DisposeDrink class
DisposeAdditive(int A[])	This operation is used to execute the action. It creates the object using the AbstractFactory class. This executes the operation of the
getAbstractFactory()	Get the AbstractFactory object.
setAbstractFactory(AbstractFactory af)	Set the AbstractFactory object.
getDataStore()	Get the DataStore object.
setDataStore(DataStore ds)	Set the DataStore object.

Class DisposeAdditive	
Purpose	This is an abstract class to Dispose the additives selected
Attributes	
DataStore *ds	Pointer to DataStore object.
Abstract Operations	
DisposeAdditive(int A[])	This is an abstract operation for disposing the additives selected.

Class DisposeAdditive1	
Purpose	This class is subclass of DisposeAdditive. It is used to dispose the additives selected
Operation	
DisposeAdditive(int A[])	operation for disposing the additives selected

Class DisposeAdditive2	
Purpose	This class is subclass of DisposeAdditive. It is used to dispose the additives selected
Operation	
DisposeAdditive(int A[])	operation for disposing the additives selected

Class DisposeDrink	
Purpose	This is an abstract class to Dispose the drink selected
Attributes	
DataStore *ds	Pointer to DataStore object.
Abstract Operations	
DisposeDrink(int d)	This is an abstract operation for disposing the drinks selected.

Class DisposeDrink1	
Purpose	This class is subclass of DisposeDrink. It is used to dispose the drinks selected
Operation	
DisposeDrink(int d)	operation for disposing the drinks selected

Class DisposeDrink2	
Purpose	This class is subclass of DisposeDrink. It is used to dispose the drinks selected
Operation	
DisposeDrink(int d)	operation for disposing the drinks selected

Class IncreaseCF	
Purpose	This is an abstract class to increase the value of cf variable
Attributes	
DataStore *ds	Pointer to DataStore object.
Abstract Operations	
IncreaseCF()	This is an abstract operation used to increase the value of cf variable
Get_DataStore()	This method is used to get the DataStore object
Set_DataStore(DataStore ds)	This method is used to set the DataStore object

Class IncreaseCF1	
Purpose	This class is subclass of IncreaseCF. It is used to increase the value of cf variable
Operation	
IncreaseCF()	This operation is used to increase the value of cf variable

Class IncreaseCF2	
Purpose	This class is subclass of IncreaseCF. It is used to increase the value of cf variable
Operation	
IncreaseCF()	This operation is used to increase the value of cf variable

Class IncreaseCF	
Purpose	This is an abstract class to increase the value of cf variable
Attributes	
DataStore *ds	Pointer to DataStore object.
Abstract Operations	
IncreaseCF()	This is an abstract operation used to increase the value of cf variable
Get_DataStore()	This method is used to get the DataStore object
Set_DataStore(DataStore ds)	This method is used to set the DataStore object

Class ReturnCoins	
Purpose	This is an abstract class to return the coins back
Attributes	
DataStore *ds	Pointer to DataStore object.
Abstract Operations	
ReturnCoins()	This is an abstract operation used return the coins back
Get_DataStore()	This method is used to get the DataStore object
Set_DataStore(DataStore ds)	This method is used to set the DataStore object

Class ReturnCoins1	
Purpose	This class is subclass of ReturnCoins. It is used to return the coins back
Operation	
ReturnCoins()	This operation is used to return the coins back

Class ReturnCoins2	
Purpose	This class is subclass of ReturnCoins. It is used to return the coins back
Operation	
ReturnCoins()	This operation is used to return the coins back

Class StorePrice	
Purpose	This is an abstract class to store the price of the drink
Attributes	
DataStore *ds	Pointer to DataStore object.
Abstract Operations	

StorePrice()	This is an abstract operation used to store the price of the drink
Get_DataStore()	This method is used to get the DataStore object
Set_DataStore(DataStore ds)	This method is used to set the DataStore object

Class StorePrice1	
Purpose	This class is subclass of StorePrice. It is used to store the price of the drink
Operation	
StorePrice()	This operation is used to store the price of the drink

Class StorePrice2	
Purpose	This class is subclass of StorePrice. It is used to store the price of the drink
Operation	
StorePrice()	This operation is used to store the price of the drink

Class ZeroCF	
Purpose	This is an abstract class to set the value of cf to zero
Attributes	
DataStore *ds	Pointer to DataStore object.
Abstract Operations	
ZeroCF()	This is an abstract operation used to set the value of cf to zero
Get_DataStore()	This method is used to get the DataStore object
Set_DataStore(DataStore ds)	This method is used to set the DataStore object

Class ZeroCF1	
Purpose	This class is subclass of ZeroCF. It is used to set the value of cf to zero
Operation	
ZeroCF()	This operation is used to set the value of cf to zero

Class ZeroCF2	
Purpose	This class is subclass of ZeroCF. It is used to set the value of cf to zero
Operation	
ZeroCF()	This operation is used to set the value of cf to zero

Class AbstractFactory	
Purpose	This class is used to create DataStore and actions objects. It is a part of Abstract Factory design pattern.
Abstract Operations	
DataStore getDataStore()	This is an abstract method to create and return DataStore object.
StorePrice get_StorePrice()	This is an abstract method to create and return StorePrice object
ZeroCF get_ZeroCF()	This is an abstract method to create and return ZeroCF object
IncreaseCF get_IncreaseCF()	This is an abstract method to create and return IncreaseCF object
ReturnCoins get_ReturnCoins()	This is an abstract method to create and return ReturnCoins object
DisposeDrink get_DisposeDrink()	This is an abstract method to create and return DisposeDrink object
DisposeAdditive get_DisposeAdditive()	This is an abstract method to create and return DisposeAdditive object

Class VM1_Factory	
Purpose	This class is used to create the data store and actions objects for Vending Machine1. This class is concrete factory class for Vending Machine1 and this is a part of Abstract factory design pattern.
Abstract Operations	
DataStore getDataStore()	This is a method to create and return DataStore object.
StorePrice get_StorePrice()	This is a method to create and return StorePrice object
ZeroCF get_ZeroCF()	This is a method to create and return ZeroCF object
IncreaseCF get_IncreaseCF()	This is a method to create and return IncreaseCF object
ReturnCoins get_ReturnCoins()	This is a method to create and return ReturnCoins object
DisposeDrink get_DisposeDrink()	This is a method to create and return DisposeDrink object
DisposeAdditive get_DisposeAdditive()	This is a method to create and return DisposeAdditive object

Class VM2_Factory	
Purpose	This class is used to create the data store and actions objects for Vending Machine1. This class is concrete factory class for Vending Machine1 and this is a part of Abstract factory design pattern.
Abstract Operations	
DataStore getDataStore()	This is a method to create and return DataStore object.
StorePrice get_StorePrice()	This is a method to create and return StorePrice object
ZeroCF get_ZeroCF()	This is a method to create and return ZeroCF object
IncreaseCF get_IncreaseCF()	This is a method to create and return IncreaseCF object
ReturnCoins get_ReturnCoins()	This is a method to create and return ReturnCoins object
DisposeDrink get_DisposeDrink()	This is a method to create and return DisposeDrink object
DisposeAdditive get_DisposeAdditive()	This is a method to create and return DisposeAdditive object

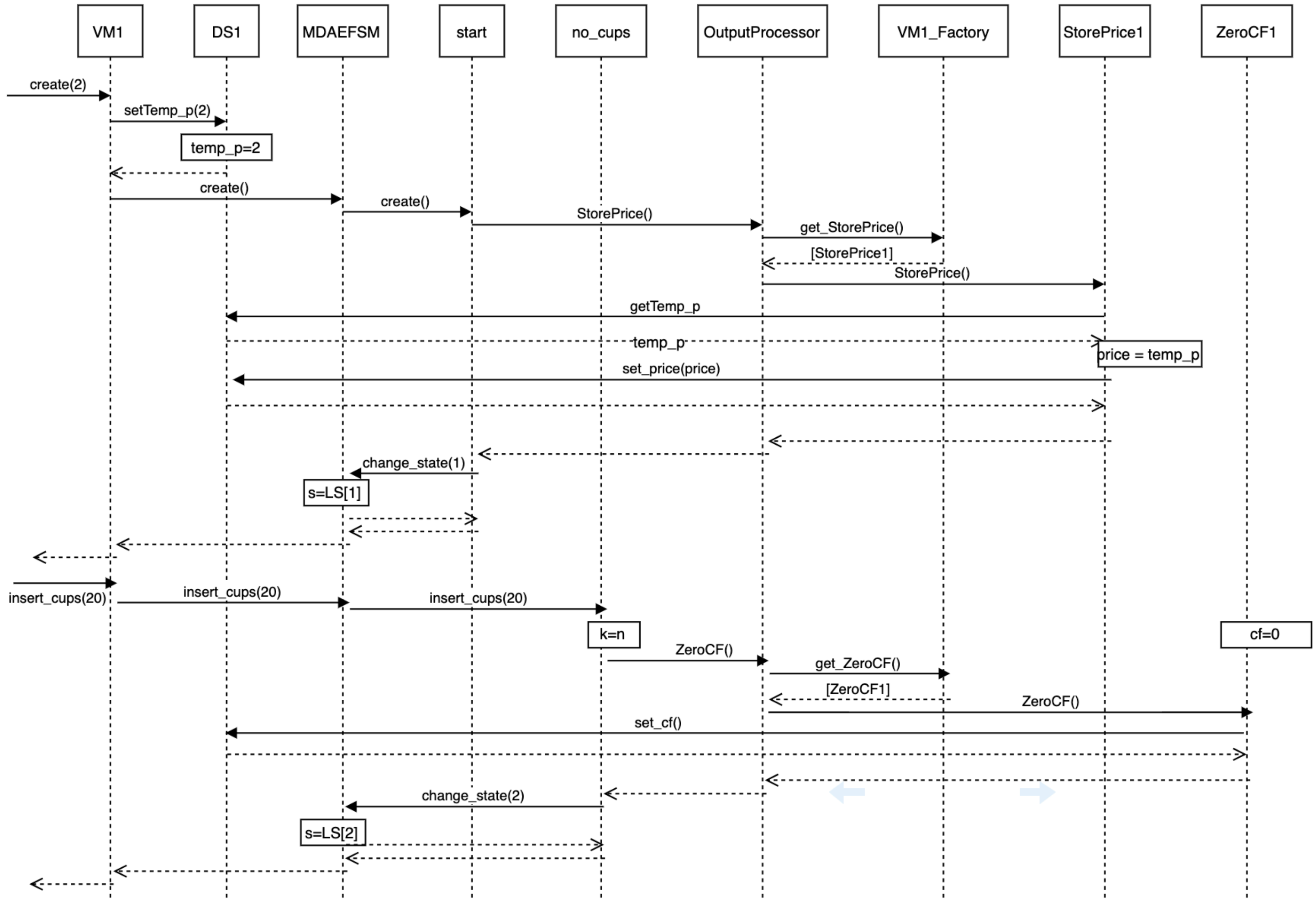
Class DataStore	
Purpose	This is an abstract class and is used to store data (platform dependent data).
Abstract Operations	
int getTemp_p()	This is abstract method to get the value of the int temp_p
int getTemp_v()	This is abstract method to get the value of the int temp_v
int get_price()	This is abstract method to get the value of the int price
int get_cf()	This is abstract method to get the value of the int cf
float getTemp_fp()	This is abstract method to get the value of the float temp_p
float getTemp_fv()	This is abstract method to get the value of the float temp_v
float get_fprice()	This is abstract method to get the value of the float price
float get_fcf()	This is abstract method to get the value of the float cf
void setTemp_p(int p)	This is abstract method to set the value of int temp_p
void setTemp_v(int v)	This is abstract method to set the value of int temp_v
void SetPrice(int pr)	This is abstract method to set the value of int price
void set_cf(int c)	This is abstract method to set the value of int cf
void setTemp_p(float p)	This is abstract method to set the value of float temp_p
void setTemp_v(float v)	This is abstract method to set the value of float temp_v
void SetPrice(float pr)	This is abstract method to set the value of float price
void set_cf(float c)	This is abstract method to set the value of float cf

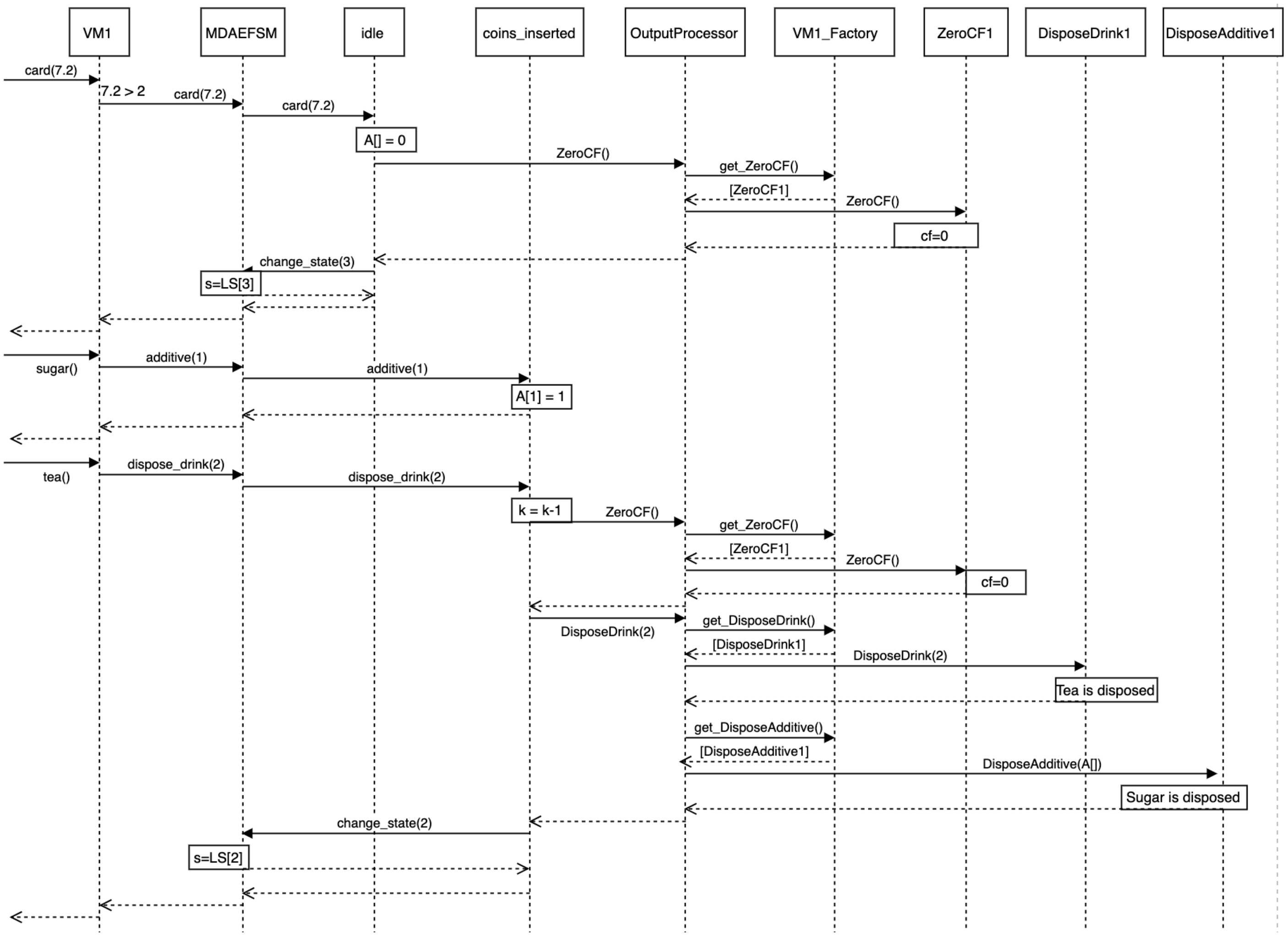
Class DS1	
Purpose	This class is used to store data (platform dependent data) for Vending Machine 1 class VM1
Abstract Operations	
int getTemp_p()	This is method to get the value of the int temp_p
int getTemp_v()	This is method to get the value of the int temp_v
int get_price()	This is method to get the value of the int price
int get_cf()	This is method to get the value of the int cf
float getTemp_fp()	This is method to get the value of the float temp_p
float getTemp_fv()	This is method to get the value of the float temp_v
float get_fprice()	This is method to get the value of the float price
float get_fcf()	This is method to get the value of the float cf
void setTemp_p(int p)	This is method to set the value of int temp_p
void setTemp_v(int v)	This is method to set the value of int temp_v
void SetPrice(int pr)	This is method to set the value of int price
void set_cf(int c)	This is method to set the value of int cf
void setTemp_p(float p)	This is method to set the value of float temp_p
void setTemp_v(float v)	This is method to set the value of float temp_v
void SetPrice(float pr)	This is method to set the value of float price
void set_cf(float c)	This is method to set the value of float cf

Class DS2	
Purpose	This class is used to store data (platform dependent data) for Vending Machine 1 class VM1
Abstract Operations	
int getTemp_p()	This is method to get the value of the int temp_p
int getTemp_v()	This is method to get the value of the int temp_v

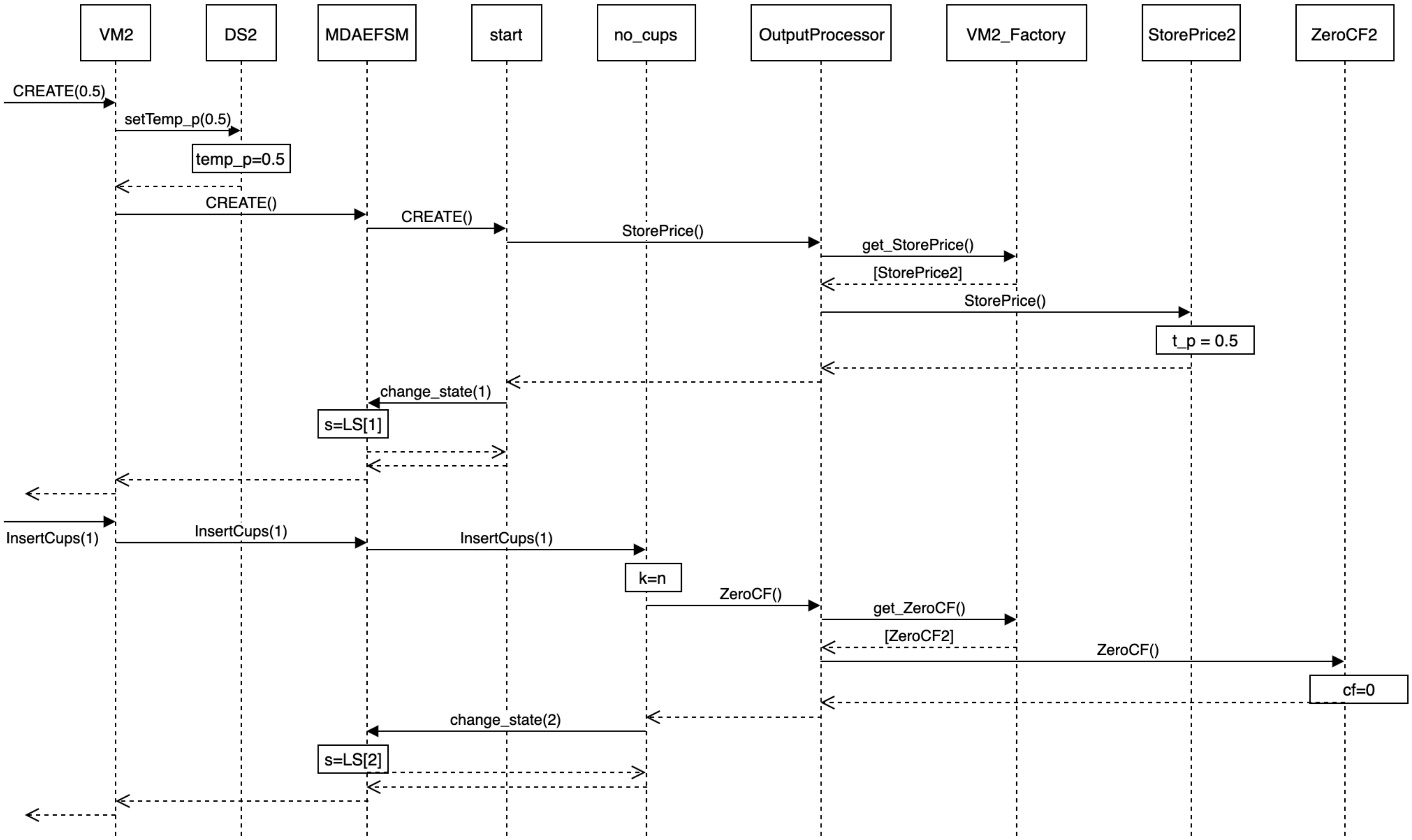
int get_price()	This is method to get the value of the int price
int get_cf()	This is method to get the value of the int cf
float getTemp_fp()	This is method to get the value of the float temp_p
float getTemp_fv()	This is method to get the value of the float temp_v
float get_fprice()	This is method to get the value of the float price
float get_fcf()	This is method to get the value of the float cf
void setTemp_p(int p)	This is method to set the value of int temp_p
void setTemp_v(int v)	This is method to set the value of int temp_v
void SetPrice(int pr)	This is method to set the value of int price
void set_cf(int c)	This is method to set the value of int cf
void setTemp_p(float p)	This is method to set the value of float temp_p
void setTemp_v(float v)	This is method to set the value of float temp_v
void SetPrice(float pr)	This is method to set the value of float price
void set_cf(float c)	This is method to set the value of float cf

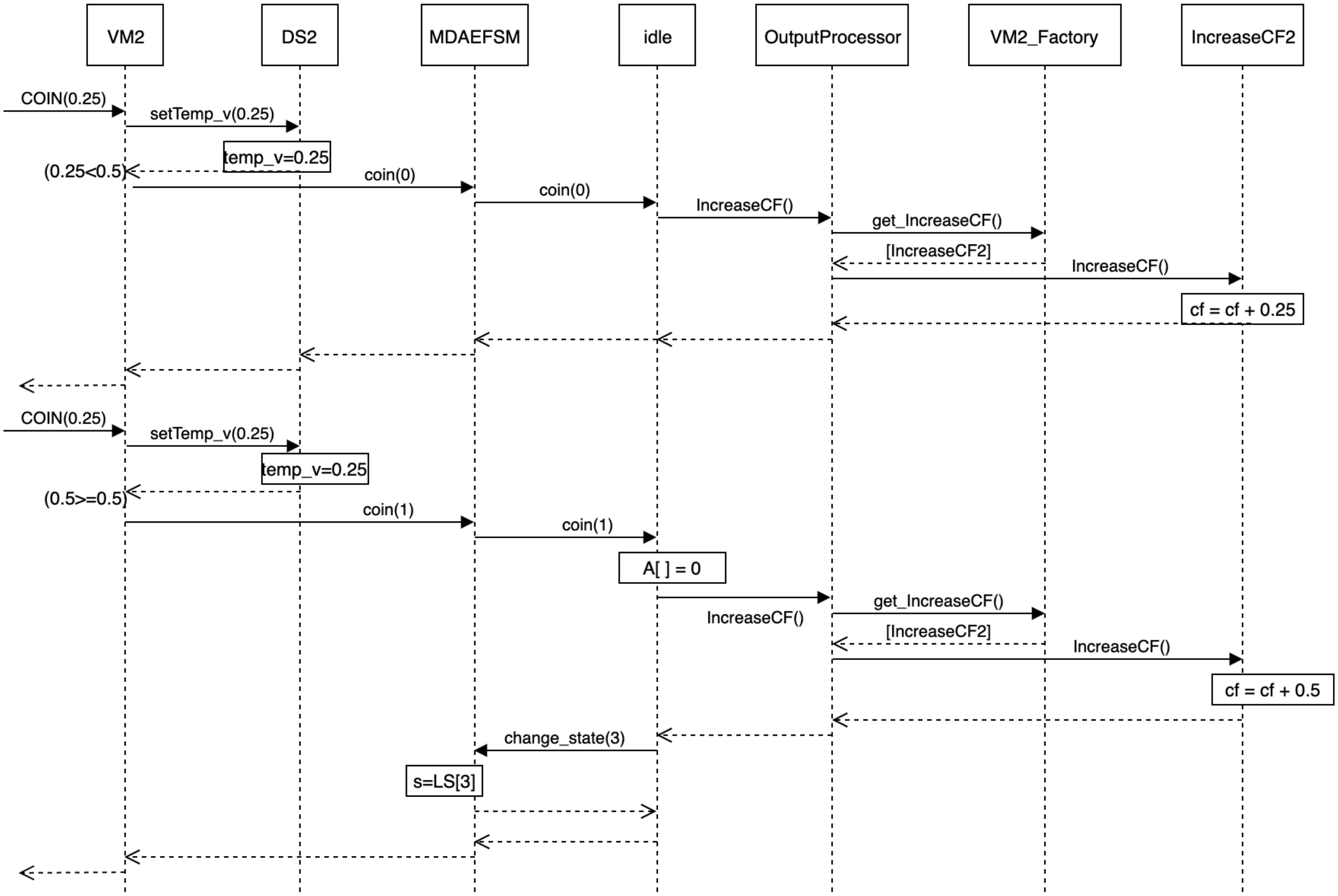
4. Provide two sequence diagrams for two Scenarios: create(2), insert_cups(20), card(7.2), sugar(), tea()

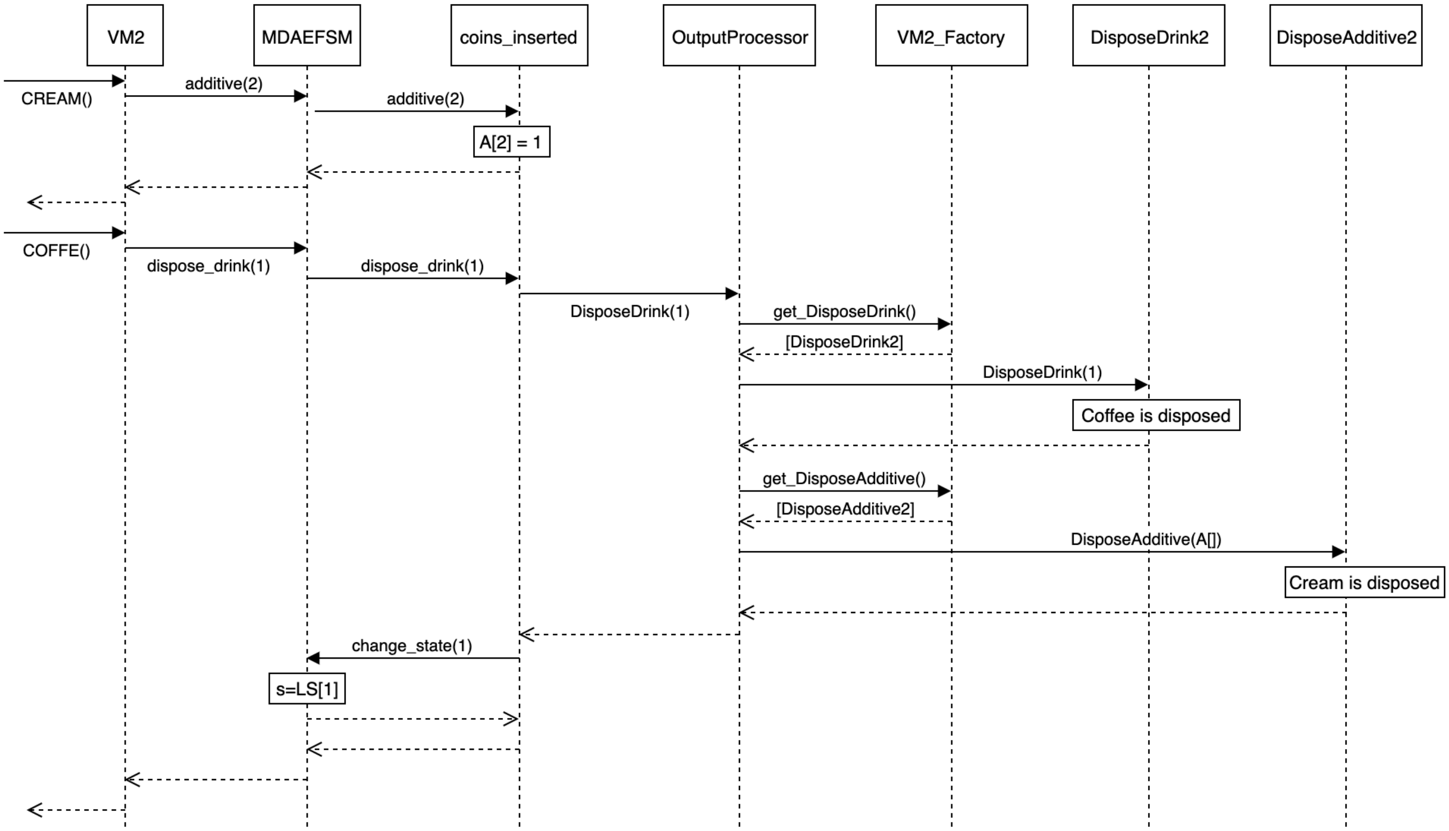



VM1
MDAEFSM
idle
coins_inserted
OutputProcessor
VM1_Factory
ZeroCF1
DisposeDrink1
DisposeAdditive1
card(7.2)
7.2 > 2
card(7.2)
card(7.2)
A[] = 0
ZeroCF()
get_ZeroCF()
[ZeroCF1]
ZeroCF()
cf=0
change_state(3)
s=LS[3]
additive(1)
sugar()
additive(1)
A[1] = 1
dispose_drink(2)
tea()
dispose_drink(2)
k = k-1
ZeroCF()
get_ZeroCF()
[ZeroCF1]
ZeroCF()
cf=0
get_DisposeDrink()
DisposeDrink(2)
[DisposeDrink1]
DisposeDrink(2)
Tea is disposed
get_DisposeAdditive()
[DisposeAdditive1]
DisposeAdditive(A[])
Sugar is disposed
change_state(2)
s=LS[2]

Scenario-II
CREATE(0.5), InsertCups(1), COIN(0.25), COIN(0.25), CREAM(), COFFEE()




VM2
DS2
MDAEFSM
idle
OutputProcessor
VM2_Factory
IncreaseCF2
COIN(0.25)
setTemp_v(0.25)
temp_v=0.25
(0.25<0.5)
coin(0)
coin(0)
IncreaseCF()
get_IncreaseCF()
[IncreaseCF2]
IncreaseCF()
cf = cf + 0.25
COIN(0.25)
setTemp_v(0.25)
temp_v=0.25
(0.5>=0.5)
coin(1)
coin(1)
A[ ] = 0
IncreaseCF()
get_IncreaseCF()
[IncreaseCF2]
IncreaseCF()
cf = cf + 0.5
change_state(3)
s=LS[3]


VM2
MDAEFSM
coins_inserted
OutputProcessor
VM2_Factory
DisposeDrink2
DisposeAdditive2
CREAM()
additive(2)
additive(2)
A[2] = 1
COFFE()
dispose_drink(1)
dispose_drink(1)
DisposeDrink(1)
get_DisposeDrink()
[DisposeDrink2]
DisposeDrink(1)
Coffee is disposed
get_DisposeAdditive()
[DisposeAdditive2]
DisposeAdditive(A[])
Cream is disposed
change_state(1)
s=LS[1]