**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) // 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**

**MDA-EFSM Actions:**

1. StorePrice()

2. ZeroCF() // zero Cumulative Fund cf

3. IncreaseCF() // increase Cumulative Fund cf

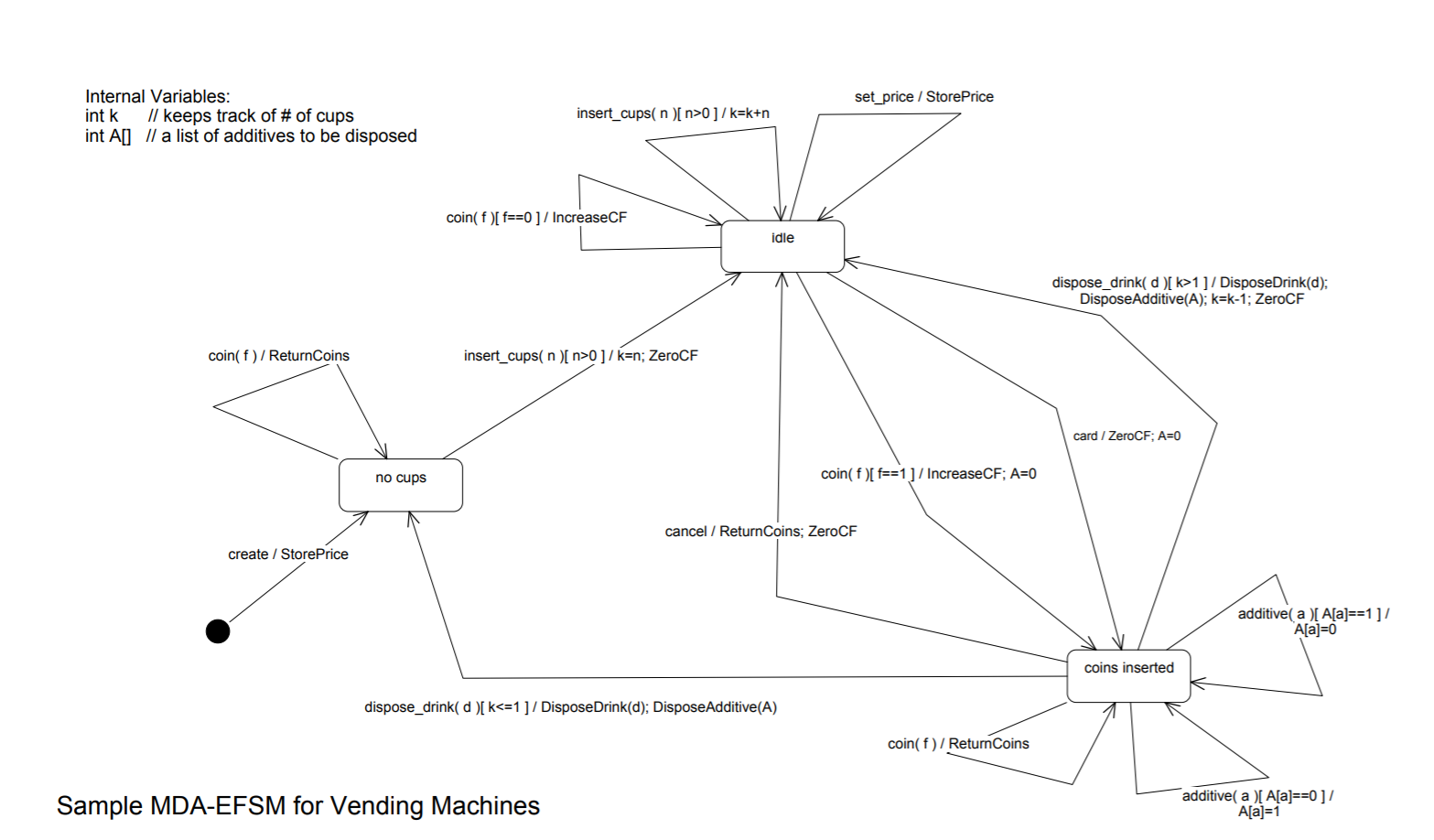
4. ReturnCoins() // return coins inserted for a drink

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**



**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

**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

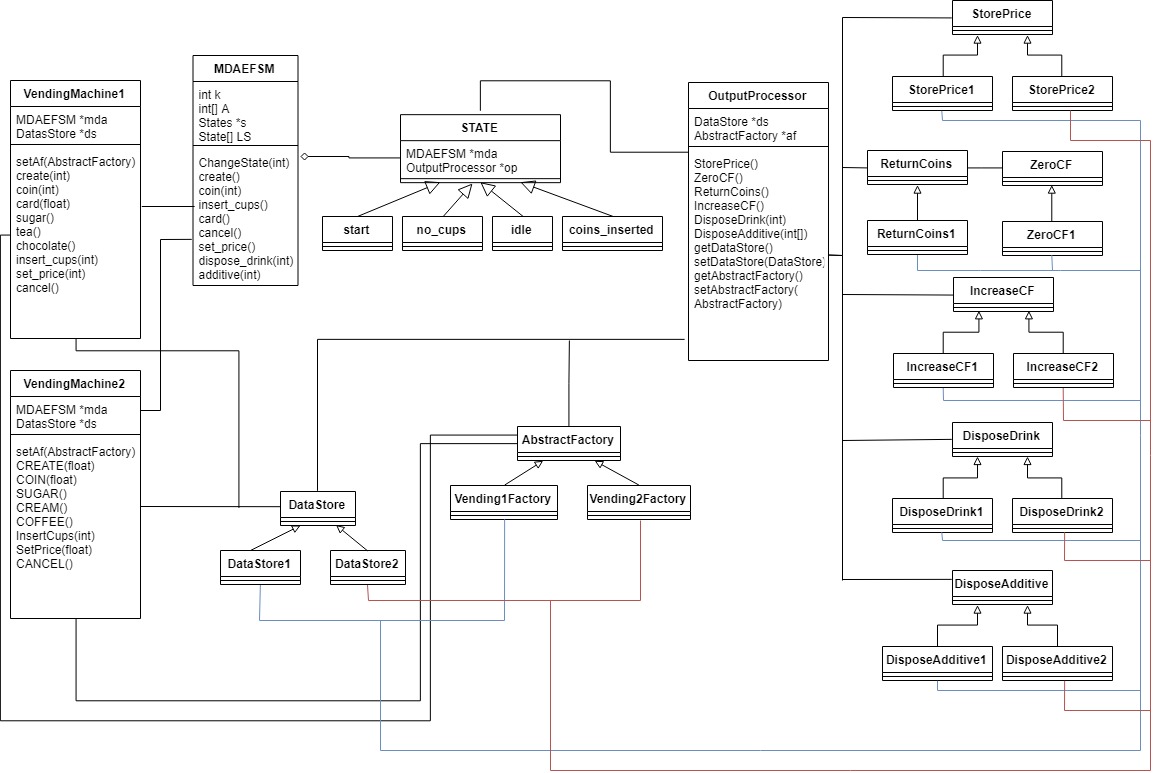
d: pointer to the data store DS-2

In the data store:

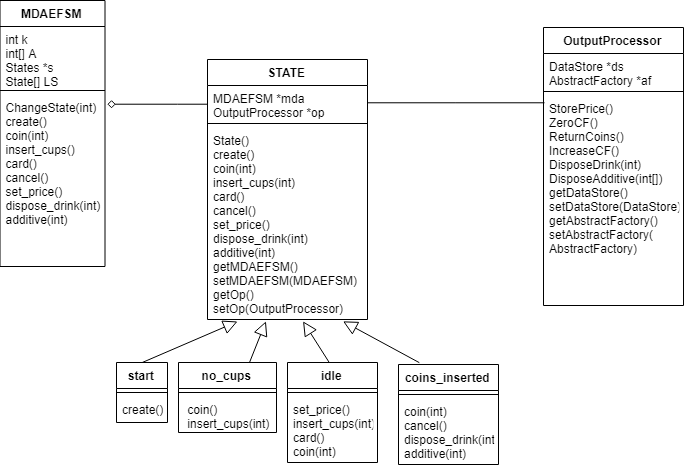
cf: represents a cumulative fund

price: represents a price for a drink

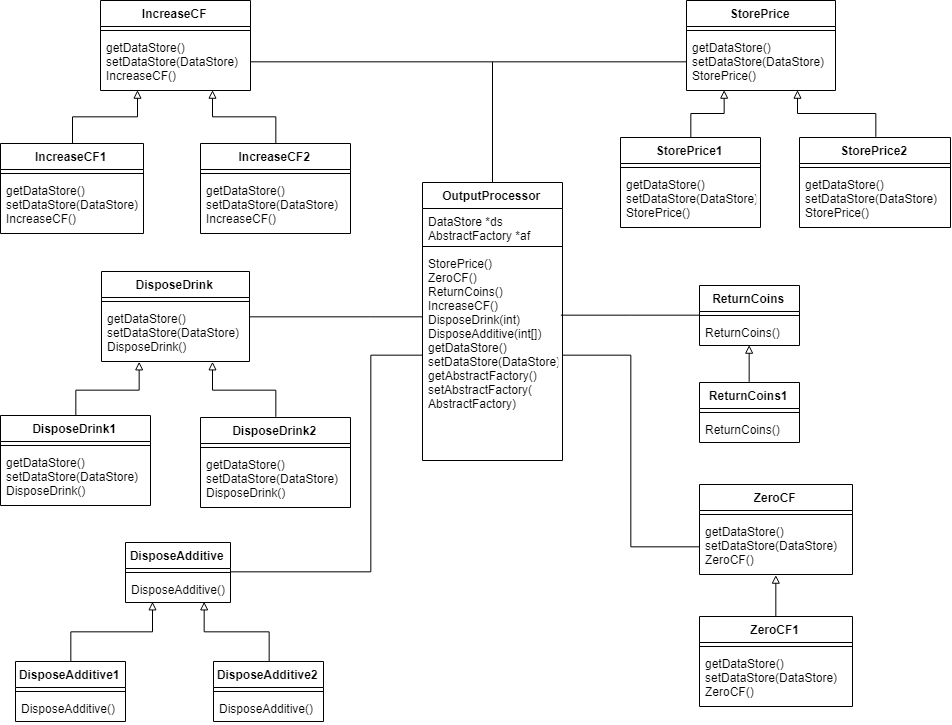
**2. Class diagram(s) of the MDA of the Vending Machine components. In your design, you MUST use the following OO design patterns:**

****

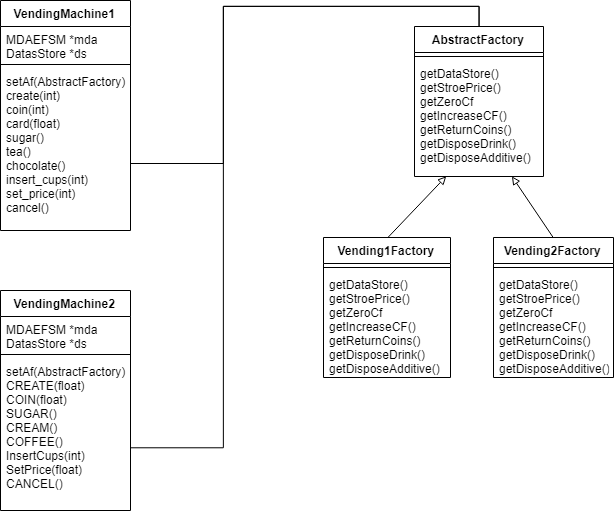
**a. State pattern**

****

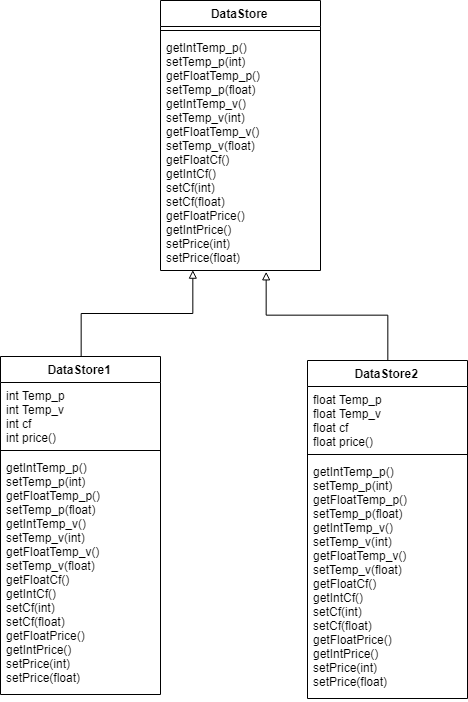
**b. Strategy pattern**

****

**c. Abstract factory pattern**

****

**DataStore**

****

**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.**

**DRIVER**

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

**INPUT PROCESSOR**

|  |  |
| --- | --- |
| Class VendingMachine1 |  |
| Purpose | This class supports all the operations Vending Machine 1 should provide |
| Attributes |  |
| MDAFSM \*mda | Pointer to MDAEFSM object. |
| DataStore \*ds | Pointer to DataStore object |
| Methods |  |
| 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. |
| chocolate() | This method is used to dispose chocolate |
| 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 |  |
| Purpose | This class supports all the operations Vending Machine 2 should provide |
| Attributes |  |
| 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 |  |
| Purpose | This class contains the events which would be triggered by the input processor vm1 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 |  |
| Purpose | It represents the state for MDAEFSM. It’s a abstract class. |
| 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 |  |
| Purpose | extends of State class and represents start state. |
| Methods |  |
| create() | Stores the price and changes the state to no\_cups |

|  |  |
| --- | --- |
| Class no\_cups |  |
| Purpose | extends of State class and represents no\_cups state. |
| Methods |  |
| 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 |  |
| Purpose | extends of State class and represents idle state. |
| Methods |  |
| 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 |  |
| Purpose | extends of State class and represents coins\_inserted state. |
| Methods |  |
| 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 |  |
| Purpose | This is a abstract class is used to create DataStore and actions objects. Abstract Factory design pattern. |
| Abstract Method |  |
| 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) |
| getZeroCf() | This is an abstract method to create and return ZeroCF object (OutputProcessor) |
| getIncreaseCf() | This is an abstract method to create and return IncreaseCF object (OutputProcessor) |
| getReturnCoins() | 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 | Concreate Factory |
| Purpose | This class is used to create the data store and actions objects for VendingMachine1 |
| Method |  |
| getDataStore() | This is an method to create and return DataStore object |
| getStorePrice() | This is an method to create and return StorePrice object (OutputProcessor) |
| getZeroCf() | This is an method to create and return ZeroCF object (OutputProcessor) |
| getIncreaseCf() | This is an method to create and return IncreaseCF object (OutputProcessor) |
| getReturnCoins() | 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 Vending2Factory | Concreate Factory |
| Purpose | This class is used to create the data store and actions objects for VendingMachine2 |
| Method |  |
| getDataStore() | This is an method to create and return DataStore object |
| getStorePrice() | This is an method to create and return StorePrice object (OutputProcessor) |
| getZeroCf() | This is an method to create and return ZeroCF object (OutputProcessor) |
| getIncreaseCf() | This is an method to create and return IncreaseCF object (OutputProcessor) |
| getReturnCoins() | 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 |  |
| Purpose | This is an abstract class and is used to store platform dependent data. |
| Method |  |
| getIntTemp\_p() | This is abstract method to get the value of temporary variable int temp\_p. |
| setTemp\_p(int) | This is abstract method to set the value of temporary variable int temp\_p. |
| getFloatTemp\_p() | This is abstract method to get the value of temporary variable float temp\_p. |
| setTemp\_p(float) | This is abstract method to set the value of temporary variable float temp\_p. |
| getIntTemp\_v() | This is abstract method to get the value of temporary variable int temp\_v |
| setTemp\_v(int) | This is abstract method to set the value of temporary variable int temp\_v. |
| getFloatTemp\_v() | This is abstract method to get the value of temporary variable float temp\_v. |
| setTemp\_v(float) | This is abstract method to set the value of temporary variable float temp\_v. |
| getFloatCf() | This is abstract method to get the value of float cf. |
| getIntCf() | This is abstract method to get the value of int cf. |
| setCf(int) | This is abstract method to set the value of int cf. |
| setCf(float) | This is abstract method to set the value of float cf. |
| 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 |  |
| Purpose | This class is used to store platform dependent data for vm1 |
| Method |  |
| getIntTemp\_p() | This method is used to get the value of temporary variable int temp\_p. |
| setTemp\_p(int) | This method is used to set the value of temporary variable int temp\_p. |
| getIntTemp\_v() | This method is used to get the value of temporary variable int temp\_v. |
| setTemp\_v(int) | This method is used to set the value of temporary variable int temp\_v. |
| getIntCf() | This method is used to get the value of variable int Cf. |
| setCf(int) | This method is used to set the value of variable int Cf. |
| getIntPrice() | This method is used to get the value of variable int price. |
| setPrice(int) | This method is used to set the value of variable int price. |

|  |  |
| --- | --- |
| Class DataStore2 |  |
| Purpose | This class is used to store platform dependent data for vm2 |
| Method |  |
| getFloatTemp\_p() | This method is used to get the value of temporary variable float temp\_p. |
| setTemp\_p(float) | This method is used to set the value of temporary variable float temp\_p. |
| getFloatTemp\_v() | This method is used to get the value of temporary variable float temp\_v. |
| setTemp\_v(float) | This method is used to set the value of temporary variable float temp\_v. |
| getFloatCf() | This method is used to get the value of variable float Cf. |
| setCf(float) | This method is used to set the value of variable float Cf. |
| getFloatPrice() | This method is used to get the value of variable float price. |
| setPrice(float) | This method is used to set the value of variable float price. |

|  |  |
| --- | --- |
| Class OutputProcessor |  |
| Purpose | 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; | pointer to DataStore  pointer to AbstractFactory  pointer to StorePrice  pointer to ZeroCF  pointer to ReturnCoins  pointer to IncreaseCF  pointer to DisposeDrink  pointer to 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 |  |
| Purpose | Interfaceclass to store price |
| Attributes |  |
| DataStore \*ds | Pointer to DataStore |
| Method |  |
| StorePrice() | This is an Interfacemethod for storing price. |
| getDataStore() | Get DataStore object |
| setDataStore(DataStore ds) | set DataStore object |

|  |  |
| --- | --- |
| Class StorePrice1 |  |
| Purpose | This class implements StorePrice |
| Method |  |
| StorePrice() | This method is used for storing the integer price |

|  |  |
| --- | --- |
| Class StorePrice2 |  |
| Purpose | This class implements StorePrice |
| Method |  |
| StorePrice() | This method is used for storing the float price |

|  |  |
| --- | --- |
| Class ReturnCoins | Interface class to return coins |
| Purpose |  |
| Method |  |
| ReturnCoins() | Interface metodfor returning coins |

|  |  |
| --- | --- |
| Class ReturnCoins1 | This class implements ReturnCoins |
| Purpose |  |
| Method |  |
| ReturnCoins() | Displays a message to return coins |

|  |  |
| --- | --- |
| Class IncreaseCF | Interface class to increasecf |
| Purpose |  |
| Attributes |  |
| DataStore \*ds | Pointer to DataStore |
| Method |  |
| IncreaseCF () | This Interfacemethod is used to increase cf |
| getDataStore() | Get DataStore object |
| setDataStore(DataStore ds) | set DataStore object |

|  |  |
| --- | --- |
| Class IncreaseCF1 |  |
| Purpose | This class implemets IncreaseCF |
| Method |  |
| IncreaseCF () | This methods adds the coins to cumulative funds integers only. |
| getDataStore() | Get DataStore object |
| setDataStore(DataStore ds) | set DataStore object |

|  |  |
| --- | --- |
| Class IncreaseCF2 |  |
| Purpose | This class implemets IncreaseCF |
| Method |  |
| IncreaseCF () | This methods adds the coins to cumulative funds float. |
| getDataStore() | Get DataStore object |
| setDataStore(DataStore ds) | set DataStore object |

|  |  |
| --- | --- |
| Class ZeroCF | Interface class to ZeroCF |
| Purpose |  |
| Method |  |
| ZeroCF () | Interface method to set cf to 0 |

|  |  |
| --- | --- |
| Class ZeroCF1 | This class implemets ZeroCF |
| Purpose |  |
| Method |  |
| ZeroCF () | This method reads the value of cf from the datastore and sets it to 0 |

|  |  |
| --- | --- |
| Class DisposeDrink | Interface class to DisposeDrink |
| Purpose |  |
| Method |  |
| DisposeDrink (int) | Interface method to dispose drink |

|  |  |
| --- | --- |
| Class DisposeDrink1 |  |
| Purpose | Implements DisposeDrink |
| Method |  |
| DisposeDrink (int) | Based on the integer passed a drink to disposed |

|  |  |
| --- | --- |
| Class DisposeDrink2 |  |
| Purpose | Implements DisposeDrink |
| Method |  |
| DisposeDrink (int) | Based on the integer passed a drink to disposed |

|  |  |
| --- | --- |
| Class DisposeAdditive | Interface class to DisposeAdditive |
| Purpose |  |
| Method |  |
| DisposeAdditive (int[]) | Interface method to dispose additive |

|  |  |
| --- | --- |
| Class DisposeAdditive1 | This class implements from DisposeAdditive |
| Purpose |  |
| Method |  |
| DisposeAdditive (int[]) | Based on the items chosen in array different additives are added. |

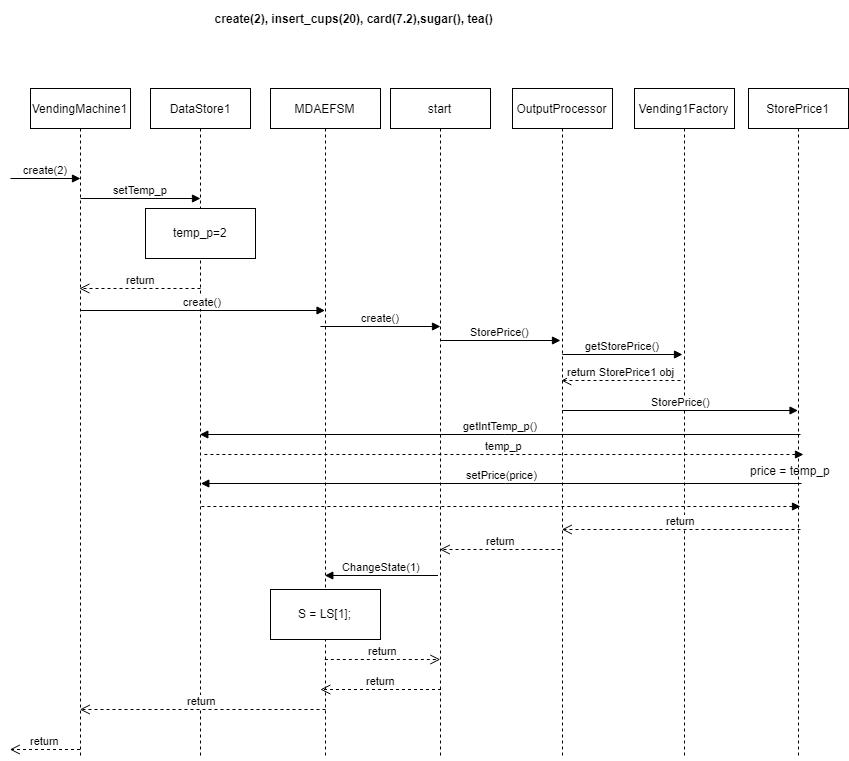
|  |  |
| --- | --- |
| Class DisposeAdditive2 | This class implements from DisposeAdditive |
| Purpose |  |
| Method |  |
| DisposeAdditive (int[]) | Based on the items chosen in array different additives are added. |

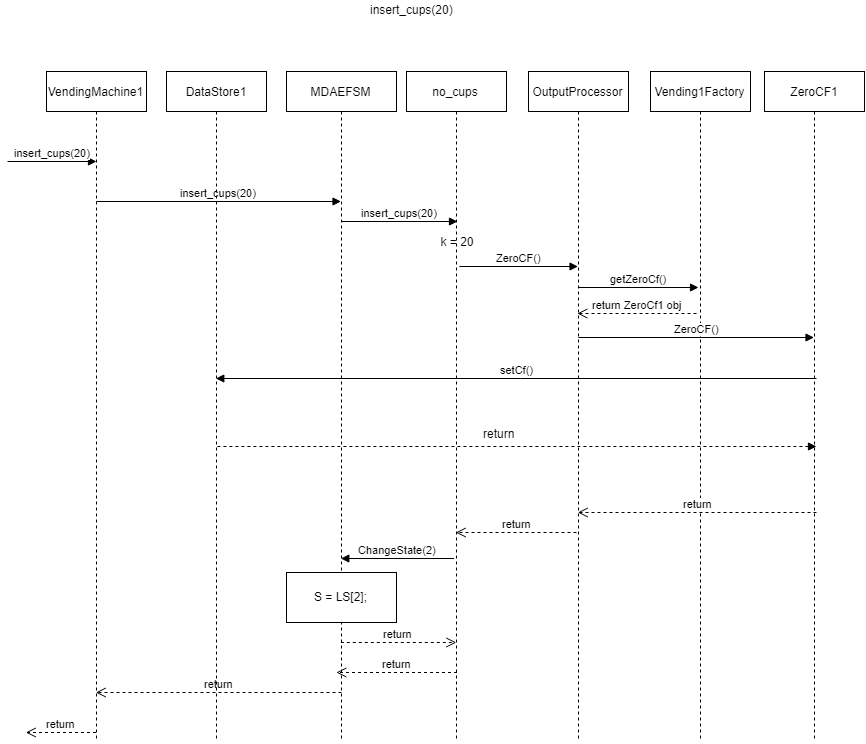
**4. Dynamics. Provide two sequence diagrams for two Scenarios:**

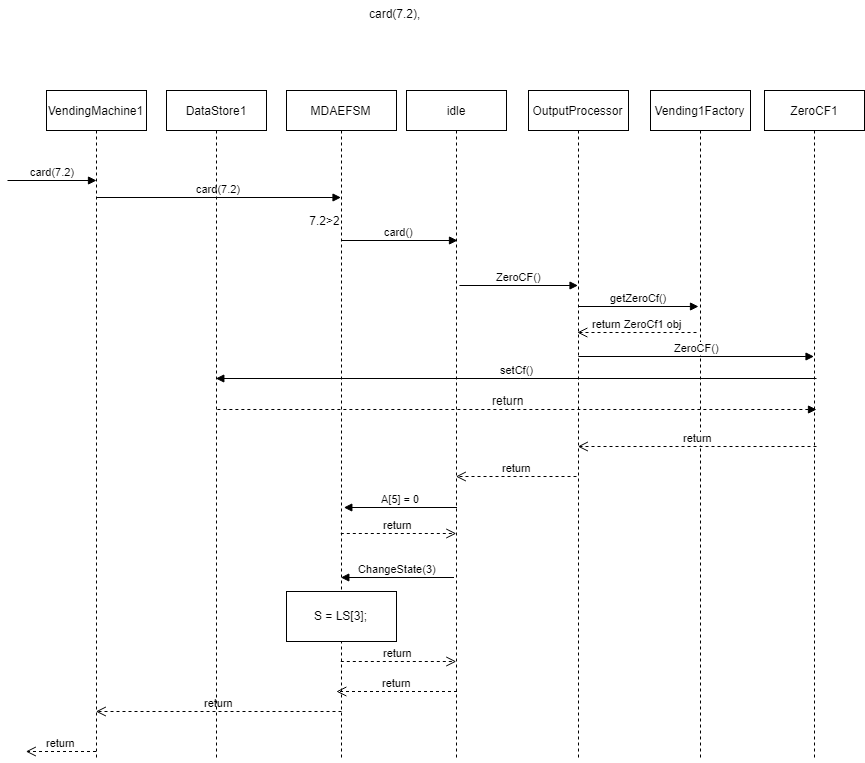
**a. Scenario-I should show as to how the cup of tea is disposed in the Vending Machine VM-1**

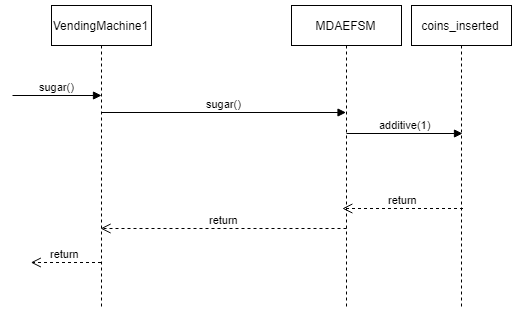
**component, i.e., the following sequence of operations is issued:**

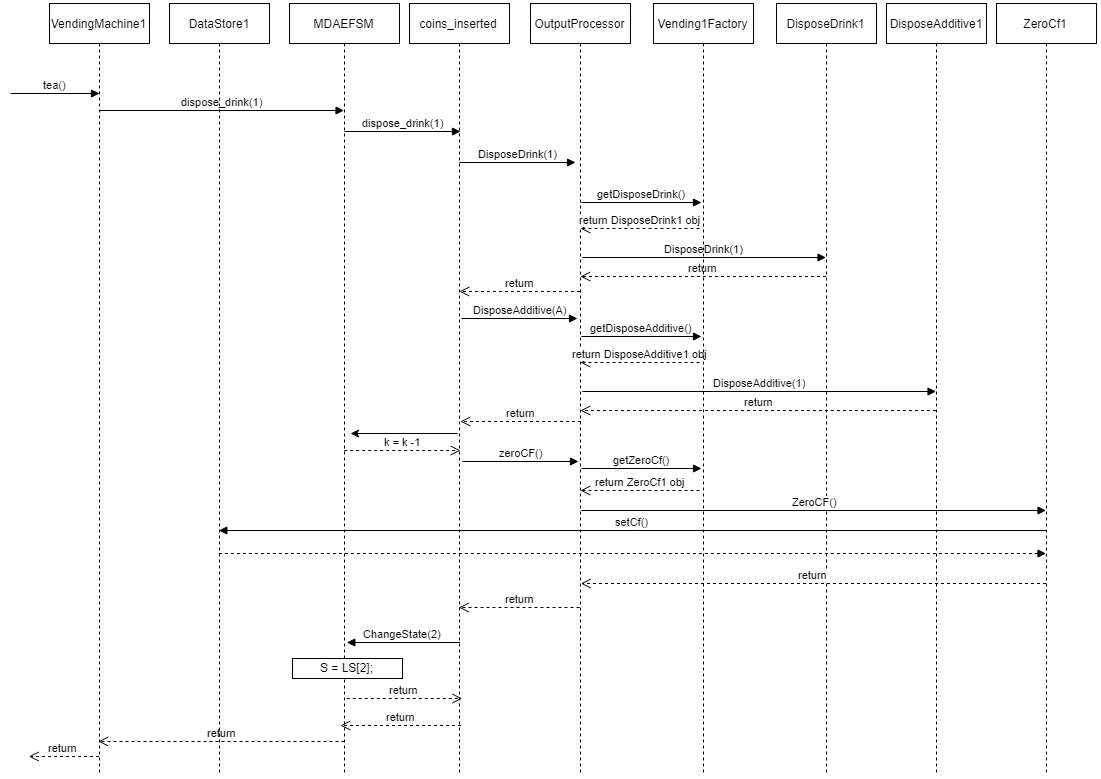
**create(2), insert\_cups(20), card(7.2), sugar(), tea()**

****

****

****

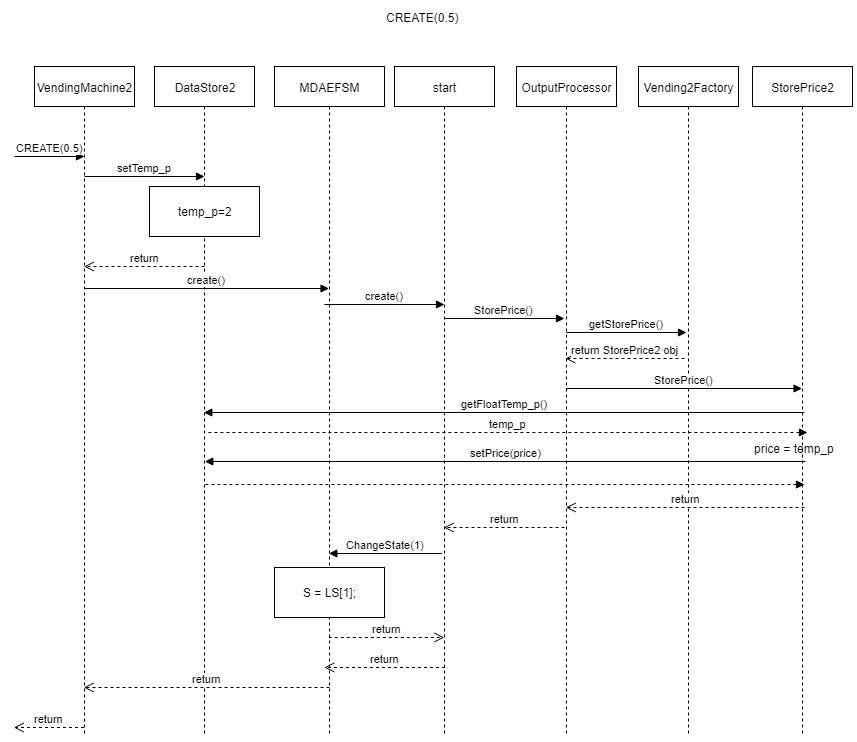
****

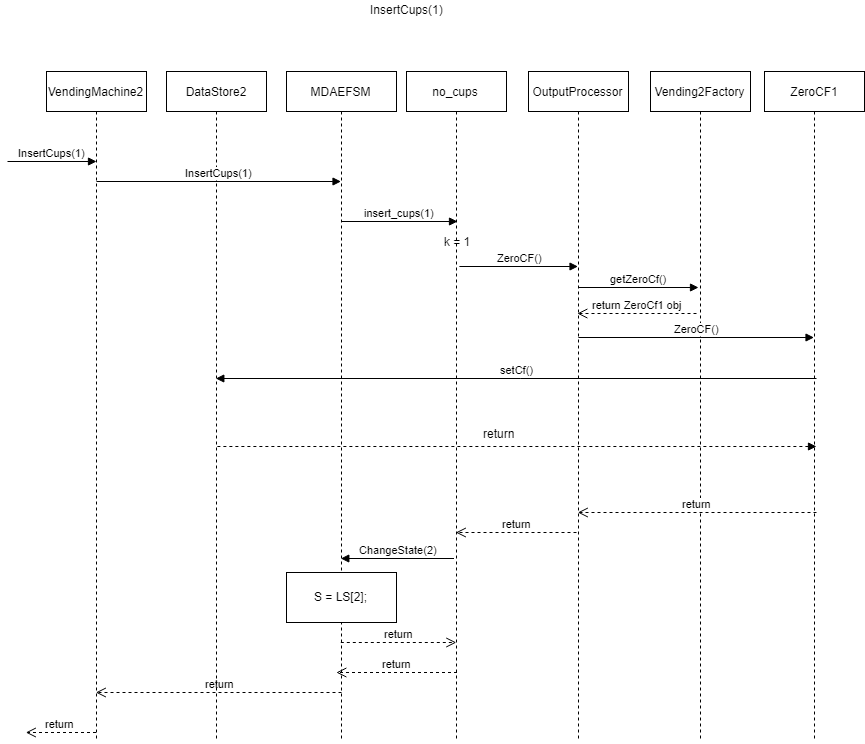
****

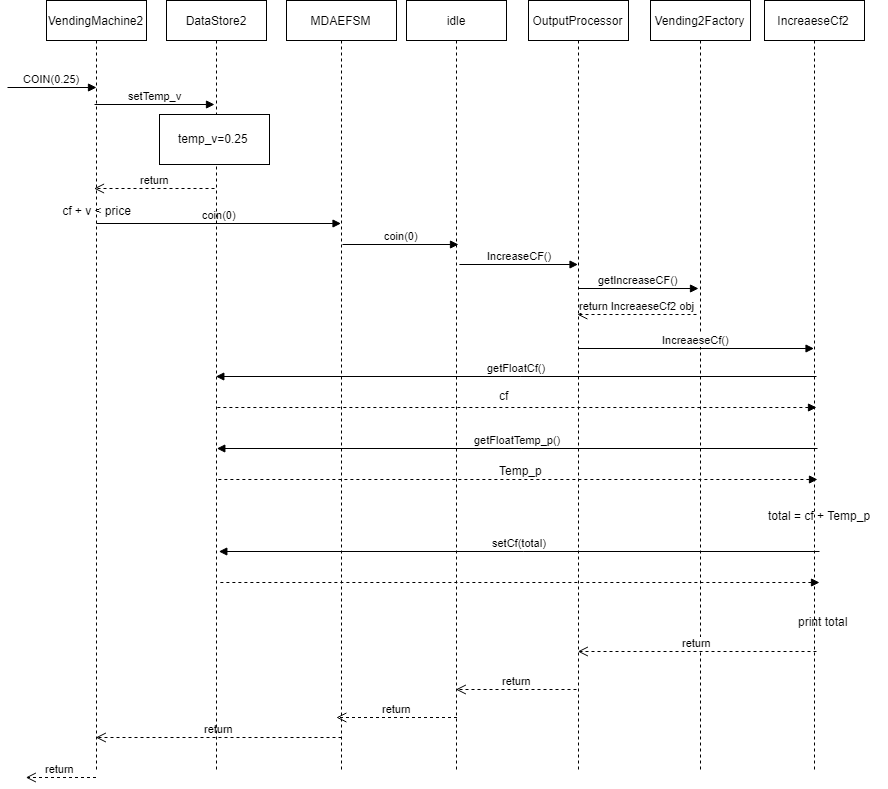
**b. Scenario-II should show as to how a cup of coffee is disposed in the Vending Machine VM-2**

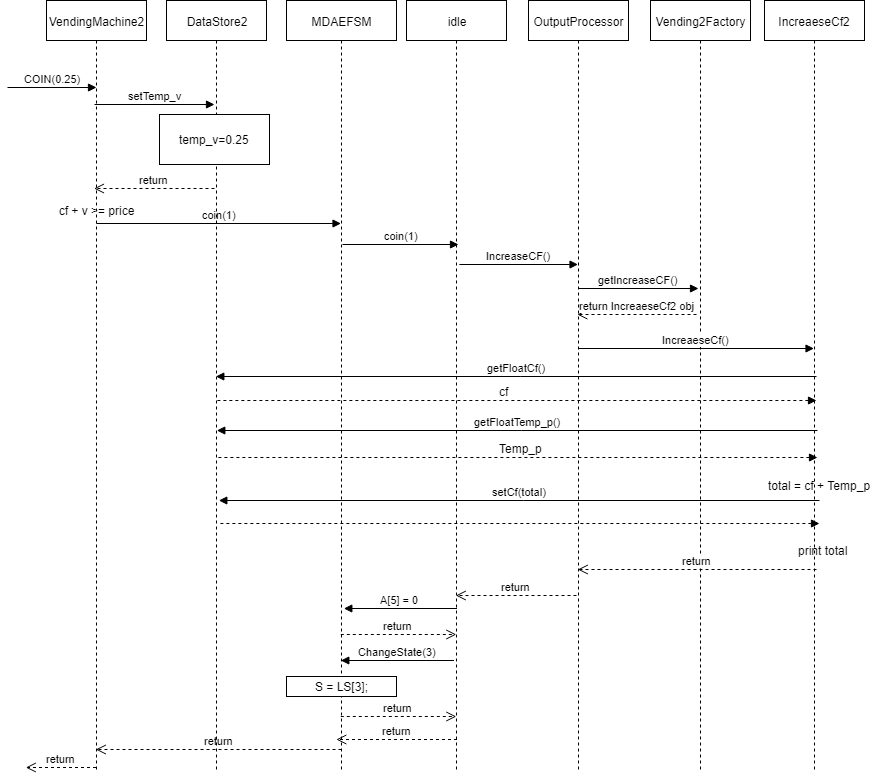
**component, i.e., the following sequence of operations is issued:**

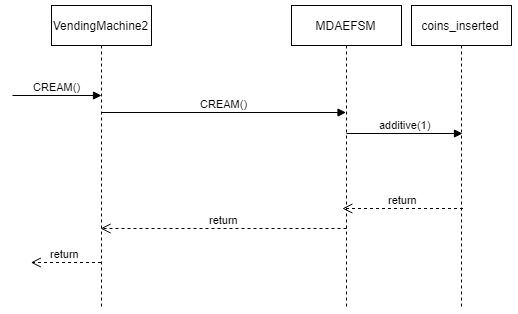
**CREATE(0.5), InsertCups(1), COIN(0.25), COIN(0.25), CREAM(), COFFEE()**

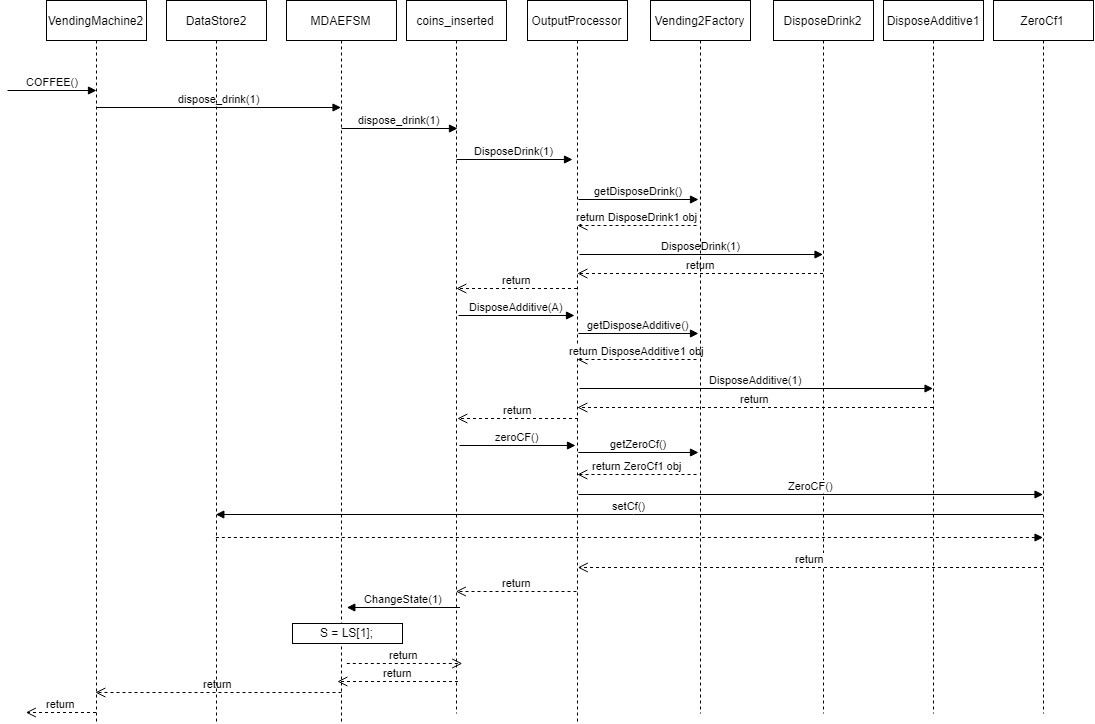
****

****

****



****

****