Software Systems Architectures

CS 586

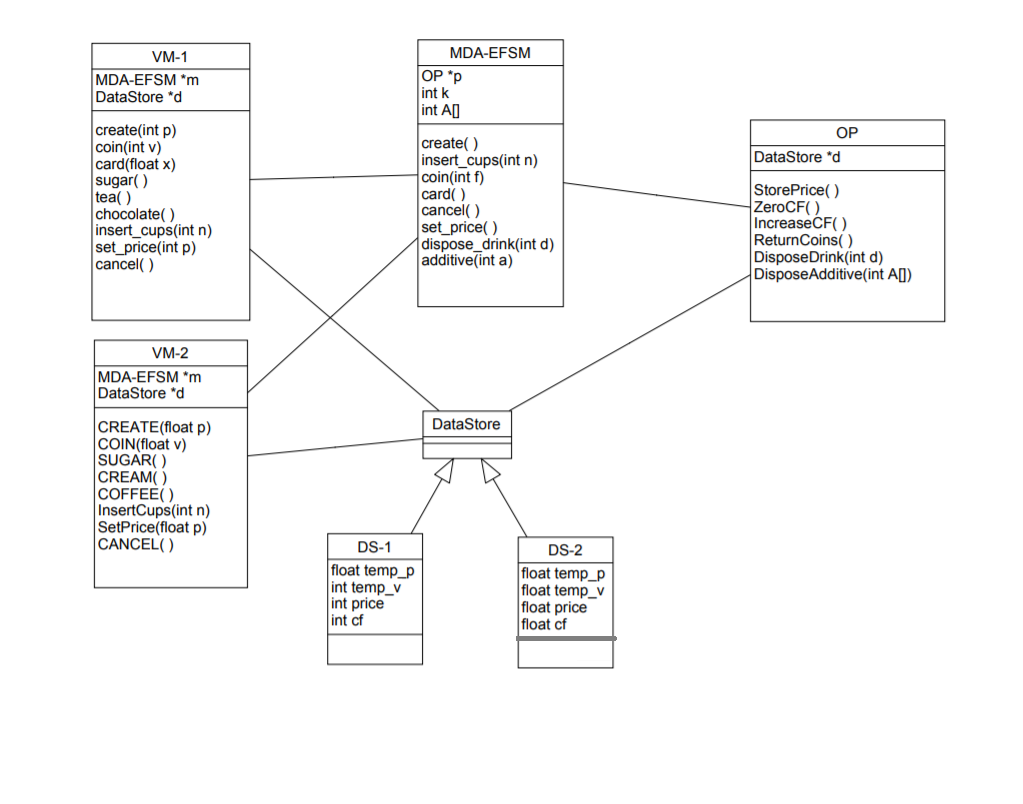
Instructor: Dr. Bogdan Korel

PROJECT

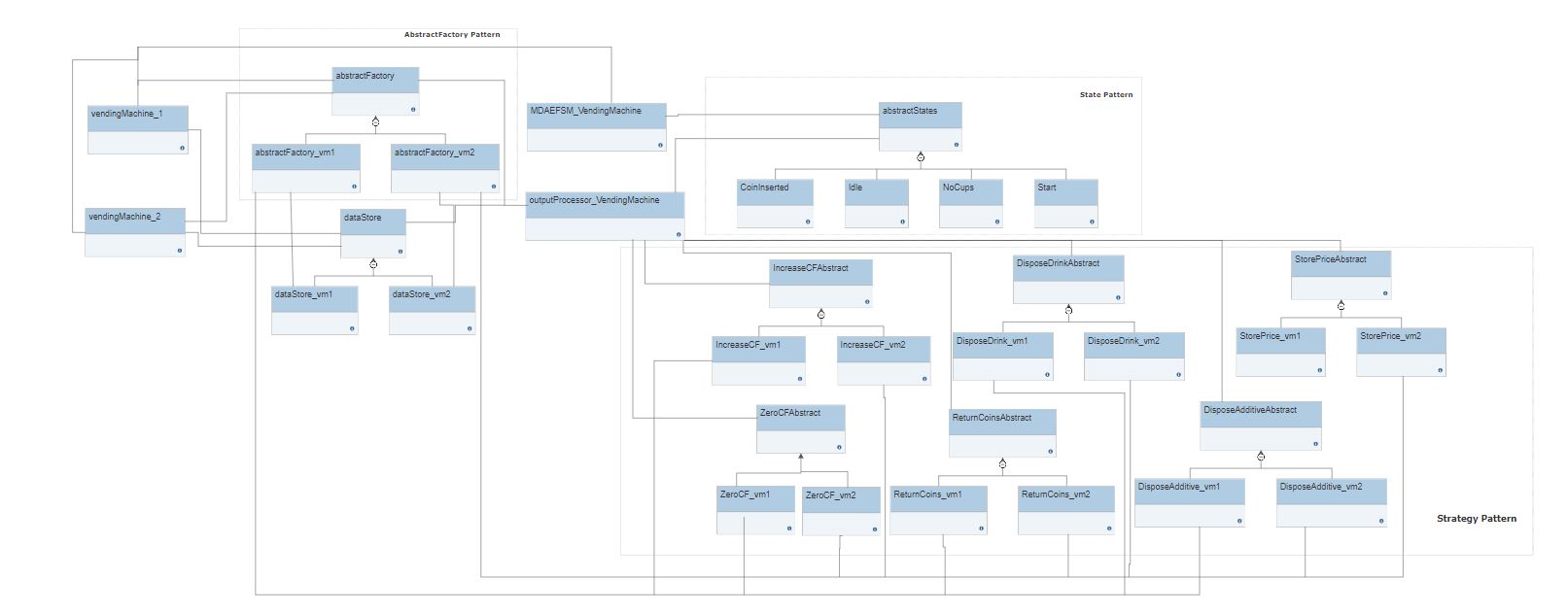
Pallavi Chandrashekar

A20427289

**Class Diagram**



**High Level Class Diagram**



**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

**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

**Operations of Input Processor (VendingMachine-1)**

**public** **void** create(**int** p) {

DS.setTemp\_p(p);

MD.create();

}

**public** **void** coin(**int** v) {

DS.setTemp\_v(v);

cf = DS.getCF();

price = DS.getPrice();

**if**(cf+v >= price)

{

MD.coin(1);

}

**else**

{

MD.coin(0);

}

}

**public** **void** card(**float** x) {

price = DS.getPrice();

**if**(x>=price)

MD.card();

}

**public** **void** dispose\_drink(**int** i) {

MD.dispose\_drink(i);

}

**public** **void** additive(**int** i) {

MD.additive(i);

}

**public** **void** set\_price(**int** p) {

DS.setTemp\_p(p);

MD.set\_Price();

}

**public** **void** insert\_cups(**int** n) {

MD.Insert\_cups(n);

}

**public** **void** cancel() {

MD.Cancel();

}

**Operations of Input Processor (VendingMachine-2)**

**public** **void** CREATE(**float** p) {

DS.setTemp\_p1(p);

MD.create();

}

**public** **void** COIN(**float** v) {

DS.setTemp\_v(v);

cf = DS.getCF1();

price = DS.getPrice1();

**if**((cf+v)>= price) {

MD.coin(1);

}

**else**

{

MD.coin(0);

}

}

**public** **void** additive(**int** i) {

MD.additive(i);

}

**public** **void** dispose\_drink(**int** i) {

MD.dispose\_drink(i);

}

**public** **void** InsertCups(**int** n) {

MD.Insert\_cups(n);

}

**public** **void** SetPrice(**float** p) {

DS.setTemp\_p1(p);

MD.set\_Price();

}

**public** **void** CANCEL() {

MD.Cancel();

}

**Class description:**

**Class vendingMachine\_1**

|  |
| --- |
| vendingMachine\_1 |
| dataStore DS  MDAEFSM\_VendingMachine MD  abstractFactory AF |
| void create(int p)  void coin(int v)  void card(float x)  void sugar()  void tea()  void chocolate()  void insert\_cups(int n)  void set\_price(int p)  void cancel() |

**dataStore** DS// Pointer to dataStore for vm1

**MDAEFSM\_VendingMachine** MD // pointer to MDA-EFSM

**abstractFactory AF** // pointer to abstractFactory for vm1

**Operations**

**void** setMDAEFSM(MDAEFSM\_VendingMachine MVM) //set MDAEFSM pointer to object created in driver.

**void** setAbstractFactory(abstractFactory vm1) // set vm1 factory ponter to object created in driver.

**void** setDataStore() // get datastore for vm1 from abstractfactory.

**void** create(int p) // set temp\_p and invokes create in MDAEFSM.

**void** coin(int v) // set temp\_v and get the value of cf and price values from dataStore of vm1 and compare the sum of v and cf with price. If sum value is greater than price then coin is invoked in MDAEFSM with 1 as parameter else coin is invoked in MDAEFSM with 0 as parameter.

**void** card(float x) // get the value of price from dataStore of vm1 and compare it with x. If x is greater than or equal to price then card is invoked in MDAEFSM.

**void** sugar() // invokes additive in MDAEFSM with 1 as parameter.

**void** tea() // invokes dispose\_drink in MDAEFSM with 1 as parameter.

**void** chocolate() // invokes dispose\_drink in MDAEFSM with 2 as parameter.

**void** insert\_cups(int n) // invokes Insert\_cups in MDAEFSM with n a parameter.

**void** set\_price(int p) //set temp\_p in datastore and invokes set\_Price in MDAEFSM with p as parameter.

void cancel() // invokes Cancel in MDAEFSM.

**Class vendingMachine\_2**

|  |
| --- |
| vendingMachine\_2 |
| dataStore DS  MDAEFSM\_VendingMachine MD  abstractFactory AF |
| void CREATE(float p)  void COIN(float v)  void SUGAR()  void CREAM()  void COFFEE()  void InsertCups(int n)  void SetPrice(int p)  void CANCEL() |

**dataStore** DS // Pointer to dataStore for vm2

**MDAEFSM\_VendingMachine** MD // pointer to MDA-EFSM

**abstractFactory** AF // pointer to abstractFactory for vm2

**Operations**

**void** setMDAEFSM(MDAEFSM\_VendingMachine MVM) – set MDAEFSM pointer to object created in driver.

**void** setAbstractFactory(abstractFactory vm2) – set vm2 factory ponter to object created in driver.

**void** setDataStore() // get datastore for vm2 from abstractfactory.

**void** COIN(float v) //sets temp\_v value as v. Get the value of cf and price and adds cf and v and compare them with price value. If result is greater than price value, then Coin(1) is invoked in MDAEFSM else Coin(0) is invoked in MDAEFSM.

**void** CREATE(float p) // set temp\_p and invokes create in MDAEFSM.void COIN(float v) – set temp\_v and get the value of cf and price values from dataStore of vm2 and compare the sum of v and cf with price. If sum value is greater than price then coin is invoked in MDAEFSM with 1 as parameter else coin is invoked in MDAEFSM with 0 as parameter.

**void** SUGAR() // invokes additive in MDAEFSM with 2 as parameter.

**void** CREAM() // invokes additive in MDAEFSM with 1 as parameter.

**void** COFFEE() // invokes dispose\_drink in MDAEFSM with 1 as parameter.

**void** InsertCups(int n) // invokes Insert\_cups in MDAEFSM with n a parameter.

**void** SetPrice(int p) //set temp\_p in datastore and invokes set\_Price in MDAEFSM with p as parameter.

**void** CANCEL() // invokes Cancel in MDAEFSM.

**Class MDAEFSM\_VendingMachine**

|  |
| --- |
| static int k  static int A[] = new int[3]  abstractStates AS  abstractStates abStates[] |
| **void** setk(int n)  **int** getk()  **void** setStateList (abstractStates abStates[])  **void** changeState(int index)  **void** Cancel()  **void** set\_Price()  **void** Insert\_cups(int n)  **void** dispose\_drink(int i)  **void** additive(int i)  **void** card()  **void** coin(int i)  **void** create()  **void** setAtoZero()  **void** updateA(int index, int val) |

static int k // No of cups

static int A[] = new int[3] // Additives list

abstractStates AS // pointer to abstractStates.

abstractStates abStates[]

**Operations**

**void** setk(int n) // sets the value of k.

**int** getk() // returns the value of k.

**void** setStateList (abstractStates abStates[]) // sets the list of states.

**void** changeState(int index) // changes state based on the input

**void** Cancel() // calls Cancel of abstractStates class.

**void** set\_Price() // calls set\_Price of abstractStates class.

**void** Insert\_cups(int n) // calls insert\_cups of abstractStates class.

**void** dispose\_drink(int i) // calls dispose\_drink of abstractStates class.

**void** additive(int i) //calls additive of abstractStates class.

**void** card() //calls card of abstractStates class.

**void** coin(int i) // calls coin of abstractStates class.

**void** create() // calls create of abstractStates class.

**void** setAtoZero() //sets value of A to zero.

**void** updateA(int index, int val) // changes the value of A to given value.

**Class outputProcessor\_VendingMachine**

|  |
| --- |
| **dataStore** DS  **DisposeAdditiveAbstract** DAA  **DisposeDrinkAbstract** DDA  **IncreaseCFAbstract** ICFA  **ReturnCoinsAbstract** RCA  **StorePriceAbstract** SPA  **ZeroCFAbstract** ZCFA  **abstractFactory** AF |
| **void** setDataStore()  **void** setFactory(abstractFactory AF)  **void** StorePrice()  **void** ReturnCoins()  **void** IncreaseCF()  **void** ZeroCF()  **void** DisposeDrink(int drink)  **void** DisposeAdditive(int[] a) |

**dataStore** DS // pointer to dataStore.

**DisposeAdditiveAbstract** DAA // pointer to DisposeAdditiveAbstract class.

**DisposeDrinkAbstract** DDA // pointer to DisposeDrinkAbstract class.

**IncreaseCFAbstract** ICFA // pointer to IncreaseCFAbstract class.

**ReturnCoinsAbstract** RCA // pointer to ReturnCoinsAbstract class.

**StorePriceAbstract** SPA // pointer to StorePriceAbstract class.

**ZeroCFAbstract** ZCFA // pointer to ZeroCFAbstract class.

**abstractFactory** AF; // pointer to abstract factory.

**Operations**

**void** setDataStore() // get datastore value from abstract factory and sets it to DS.

**void** setFactory(abstractFactory AF) // sets the factory object as passed value

**void** StorePrice() //gets StorePrice class object and calls the function StorePrice

**void** ReturnCoins() //gets ReturnCoins class object and calls the function ReturnCoins

**void** IncreaseCF() //gets IncreaseCF class object and calls the function IncreaseCF

**void** ZeroCF() //gets ZeroCF class object and calls the function ZeroCF

**void** DisposeDrink(int drink)//gets DisposeDrink class object and calls the function DisposeDrink

**void** DisposeAdditive(int[] a) //gets DisposeAdditive class object and calls the function DisposeAdditive

**Class dataStore**

|  |
| --- |
| dataStore |
|  |
| **void** setTemp\_p(**int** p)  **int** getTemp\_p()  **void** setPrice(**int** price)  **int** getPrice()  **void** setTemp\_v(**int** v)  **int** getTemp\_v()  **void** setCF(**int** cf)  **int** getCF()  **void** setTemp\_p1(**float** p)  **float** getTemp\_p1()  **void** setPrice1(**float** price)  **float** getPrice1()  **void** setTemp\_v1(**float** v)  **float** getTemp\_v1()  **void** setCF1(**float** cf)  **float** getCF1(); |

**Operations**

This is abstract class of dataStore.

**Class dataStore\_vm1**

|  |
| --- |
| dataStore\_vm1 |
| int temp\_p, temp\_v, cf, price |
| **void** setTemp\_p(**int** p)  **int** getTemp\_p()  **void** setPrice(**int** price)  **int** getPrice()  **void** setTemp\_v(**int** v)  **int** getTemp\_v()  **void** setCF(**int** cf)  **int** getCF() |

int temp\_p, temp\_v, cf ,price // variables used by vendingMachine 1.

**Operations**

**void** setTemp\_p(**int** p) //Method stores the value passed to temp\_p variable.

**int** getTemp\_p() // Method is used to get the temp\_p value.

**void** setPrice(**int** price) //Method stores the value passed to price variable.

**int** getPrice() // Method is used to get the price value.

**void** setTemp\_v(**int** v) // Method stores the value passed to temp\_v variable.

**int** getTemp\_v() // Method is used to get the temp\_v value.

**void** setCF(**int** cf) // Method stores the value passed to cf variable.

**int** getCF() // Method is used to get the cf value.

**Class dataStore\_vm2**

|  |
| --- |
| dataStore\_vm2 |
| float temp\_p, temp\_v, cf, price |
| **void** setTemp\_p(**float** p)  **float** getTemp\_p()  **void** setPrice(**float** price)  **float** getPrice()  **void** setTemp\_v(**float** v)  **float** getTemp\_v()  **void** setCF(**float** cf)  **float** getCF() |

float temp\_p, temp\_v, cf ,price // variables used by vendingMachine 2.

**Operations**

**void** setTemp\_p(f**loat** p) //Method stores the value passed to temp\_p variable.

**float** getTemp\_p() // Method is used to get the temp\_p value.

**void** setPrice(**float** price) //Method stores the value passed to price variable.

**float** getPrice() // Method is used to get the price value.

**void** setTemp\_v(**float** v) // Method stores the value passed to temp\_v variable.

**float** getTemp\_v() // Method is used to get the temp\_v value.

**void** setCF(**float** cf) // Method stores the value passed to cf variable.

**int** getCF() // Method is used to get the cf value.

**Abstract Factory Pattern**

**Class abstractFactory**

|  |
| --- |
| abstractFactory |
|  |
| **dataStore** getDataStore()  **DisposeAdditiveAbstract** getDisposeAdditive\_Obj()  **DisposeDrinkAbstract** getDisposeDrink\_Obj()  **IncreaseCFAbstract** getIncreaseCF\_Obj()  **ReturnCoinsAbstract** getReturnCoins\_Obj()  **StorePriceAbstract** getStorePrice\_Obj()  **ZeroCFAbstract** getZeroCF\_Obj(); |

This is abstract class of abstractFactory pattern. Abstract factory pattern is used to create objects required for the classes of VendingMachine.

**Class abstractFactory\_vm1**

|  |
| --- |
| abstractFactory\_vm1 |
| **dataStore\_vm1** DS  **DisposeAdditive\_vm1** DA  **DisposeDrink\_vm1** DD  **IncreaseCF\_vm1** ICF  **ReturnCoins\_vm1** RC  **StorePrice\_vm1** SP  **ZeroCF\_vm1** ZCF |
| **dataStore\_vm1** getDataStore()  **DisposeAdditive\_vm1**  getDisposeAdditive\_Obj()  **DisposeDrink\_vm1** getDisposeDrink\_Obj()  **IncreaseCF\_vm1** getIncreaseCF\_Obj()  **ReturnCoins\_vm1** getReturnCoins\_Obj()  **StorePrice\_vm1** getStorePrice\_Obj()  **ZeroCF\_vm1** getZeroCF\_Obj() |

**dataStore\_vm1** DS // pointer to dataStore\_vm1 class of vm1.

**DisposeAdditive\_vm1** DA // pointer to DisposeAdditive\_vm1 class of vm1.

**DisposeDrink\_vm1** DD // pointer to DisposeDrink\_vm1 class of vm1.

**IncreaseCF\_vm1** ICF // pointer to IncreaseCF\_vm1 class of vm1.

**ReturnCoins\_vm1** RC // pointer to ReturnCoins\_vm1 class of vm1.

**StorePrice\_vm1** SP // pointer to StorePrice\_vm1 class of vm1.

**ZeroCF\_vm1** ZCF // pointer to ZeroCF\_vm1 class of vm1.

**Operations**

**dataStore\_vm1** getDataStore() // Method is used to get the dataStore assigned to VendingMachine 1, create object for the datastore and return the dataStore object

**DisposeAdditive\_vm1**  getDisposeAdditive\_Obj() // Method is used to create an object for DisposeAdditive\_vm1 class and return the object.

**DisposeDrink\_vm1** getDisposeDrink\_Obj() // Method is used to create an object for DisposeDrink\_vm1 class and return the object.

**IncreaseCF\_vm1** getIncreaseCF\_Obj() // Method is used to create an object for IncreaseCF\_vm1 class and return the object.

**ReturnCoins\_vm1** getReturnCoins\_Obj() // Method is used to create an object for to ReturnCoins\_vm1 class and return the object.

**StorePrice\_vm1** getStorePrice\_Obj() // Method is used to create an object for StorePrice\_vm1 class and return the object.

**ZeroCF\_vm1** getZeroCF\_Obj() // Method is used to create an object for ZeroCF\_vm1 class and return the object.

**Class abstractFactory\_vm2**

|  |
| --- |
| abstractFactory\_vm2 |
| **dataStore\_vm2**DS  **DisposeAdditive\_vm2** DA  **DisposeDrink\_vm2** DD  **IncreaseCF\_vm2** ICF  **ReturnCoins\_vm2** RC  **StorePrice\_vm2** SP  **ZeroCF\_vm2** ZCF |
| **dataStore\_vm2** getDataStore()  **DisposeAdditive\_vm2**  getDisposeAdditive\_Obj()  **DisposeDrink\_vm2** getDisposeDrink\_Obj()  **IncreaseCF\_vm2** getIncreaseCF\_Obj()  **ReturnCoins\_vm2** getReturnCoins\_Obj()  **StorePrice\_vm2** getStorePrice\_Obj()  **ZeroCF\_vm2** getZeroCF\_Obj() |

**dataStore\_vm2** DS // pointer to dataStore\_vm2 class of vm2.

**DisposeAdditive\_vm2** DA // pointer to DisposeAdditive\_vm2 class of vm2.

**DisposeDrink\_vm2** DD // pointer to DisposeDrink\_vm2 class of vm2.

**IncreaseCF\_vm2** ICF // pointer to IncreaseCF\_vm2 class of vm2.

**ReturnCoins\_vm2** RC // pointer to ReturnCoins\_vm2 class of vm2.

**StorePrice\_vm2** SP // pointer to StorePrice\_vm2 class of vm2.

**ZeroCF\_vm2** ZCF // pointer to ZeroCF\_vm2 class of vm2.

**Operations**

**dataStore\_vm2** getDataStore() // Method is used to get the dataStore assigned to VendingMachine 2, create object for the datastore and return the dataStore object.

**DisposeAdditive\_vm2**  getDisposeAdditive\_Obj() // Method is used to create an object for DisposeAdditive\_vm2 class and return the object.

**DisposeDrink\_vm2** getDisposeDrink\_Obj() // Method is used to create an object for DisposeDrink\_vm2 class and return the object.

**IncreaseCF\_vm2** getIncreaseCF\_Obj() // Method is used to create an object for IncreaseCF\_vm2 class and return the object.

**ReturnCoins\_vm2** getReturnCoins\_Obj() // Method is used to create an object for to ReturnCoins\_vm2 class and return the object.

**StorePrice\_vm2** getStorePrice\_Obj() // Method is used to create an object for StorePrice\_vm2 class and return the object.

**ZeroCF\_vm2** getZeroCF\_Obj() // Method is used to create an object for ZeroCF\_vm2 class and return the object.

**State Pattern**

**Class abstractStates**

|  |
| --- |
| abstractStates |
| **outputProcessor\_VendingMachine** OP  **MDAEFSM\_VendingMachine** MD |
| **void** Create()  **void** Coin(int i)  **void** Card()  **void** SetPrice()  **void** cancel()  **void** insert\_cups(int k)  **void** Additive(int additional)  **void** dispose\_drink(int drink) |

This is abstract class of states of vending machine.

**Class Start**

|  |
| --- |
| Start |
| **outputProcessor\_VendingMachine** OP  **MDAEFSM\_VendingMachine** MD |
| **void** Create()  **void** Coin(int i)  **void** Card()  **void** SetPrice()  **void** cancel()  **void** insert\_cups(int n)  **void** Additive(int additional)  **void** dispose\_drink(int drink) |

**outputProcessor\_VendingMachine** OP // pointer to output processor of vending machine.

**MDAEFSM\_VendingMachine** MD //pointer to MDAEFSM class.

**Operations**

**void** Create() // In this Start State we will override only Create() method. Which stores the price value to the dataStore associated with vendingmachine and change the state of the machine to NoCups state.

**Class NoCups**

|  |
| --- |
| NoCups |
| **outputProcessor\_VendingMachine** OP  **MDAEFSM\_VendingMachine** MD |
| **void** Create()  **void** Coin(int i)  **void** Card()  **void** SetPrice()  **void** cancel()  **void** insert\_cups(int n)  **void** Additive(int additional)  **void** dispose\_drink(int drink) |

**outputProcessor\_VendingMachine** OP // pointer to output processor of vending machine.

**MDAEFSM\_VendingMachine** MD //pointer to MDAEFSM class.

**Operations**

**void** insert\_cups(int n) // In this NoCups State we will override only insert\_cups(int n) method. If the value of n is greater than 0, then it stores 0 to cf in the dataStore. Change the value of k in MDAESFM and change the state to Idle.

**Class Idle**

|  |
| --- |
| Start |
| **outputProcessor\_VendingMachine** OP  **MDAEFSM\_VendingMachine** MD |
| **void** Create()  **void** Coin(int i)  **void** Card()  **void** SetPrice()  **void** cancel()  **void** insert\_cups(int n)  **void** Additive(int additional)  **void** dispose\_drink(int drink) |

**outputProcessor\_VendingMachine** OP // pointer to output processor of vending machine.

**MDAEFSM\_VendingMachine** MD //pointer to MDAEFSM class.

**Operations**

**void** Coin(int i) // In this Idle State we will override Coin(int i) method. If I value is 0, then value of cf is changed to cf+I in the dataStore. If I value is 1, then value of cf is changed to cf+I in the dataStore. A value is set to Zero in MDAEFSM and state is changed to CoinInserted.

**void** Card() //In this Idle state, we will override Card() method as well. Set cf to zero in dataStore. Changes state to CoinInserted.

**void** SetPrice() // In this Idle state, we will override SetPrice() method, which Stores sets the value of price in dataStore.

**void** insert\_cups(int n) // In this Idle state, we will override insert\_cups(int n) method also, we will change the value of k in MDAEFSM to k+n.

Class **CoinInserted**

|  |
| --- |
| CoinInserted |
| **outputProcessor\_VendingMachine** OP  **MDAEFSM\_VendingMachine** MD |
| **void** Create()  **void** Coin(int i)  **void** Card()  **void** SetPrice()  **void** cancel()  **void** insert\_cups(int n)  **void** Additive(int additional)  **void** dispose\_drink(int drink) |

**outputProcessor\_VendingMachine** OP // pointer to output processor of vending machine.

**MDAEFSM\_VendingMachine** MD //pointer to MDAEFSM class.

**Operations**

**void** cancel() // In CoinInserted state, we will override cancel() method, returns the coin, sets cf to zero and changes state to Idle state.

**void** Additive(int additional) // In this method, we will check the value of A[additional] in MDAEFSM, if value is 1, then we will deselect the additive. If value is 0, then we will select the additive.

**void** dispose\_drink(int drink) // In this Method, if value of k in MDAEFSM is less than or equal to 1, then we will call DisposeDrink in output Processor, we will call DisposeAdditive and change the state to NoCups.

If value of k in MDAEFSM is greater than 1, then call DisposeDrink in output Processor, we will call DisposeAdditive, set the value of k to k-1, set cf to zero and change the state to Idle.

**StrategyPattern**

**Class DisposeAdditiveAbstract**

|  |
| --- |
| DisposeAdditiveAbstract |
|  |
| abstract **void** DisposeAdditive(int[] a) |

This is abstract class of DisposeAdditive. Which is used to add additives based on the value in A MDAEFSM.

**Class DisposeAdditive\_vm1**

|  |
| --- |
| DisposeAdditive\_vm1 |
|  |
| **void** DisposeAdditive(int[] a) |

**Operations**

**void** DisposeAdditive(int[] a) // Based on value assigned in VendingMachine, We will add additive to the drink.

In vm\_1, we have only 1 additive “Sugar”. 1 is assigned to Sugar.

**Class DisposeAdditive\_vm2**

|  |
| --- |
| DisposeAdditive\_vm2 |
|  |
| **void** DisposeAdditive(int[] a) |

**Operations**

**void** DisposeAdditive(int[] a) // Based on value assigned in VendingMachine, We will add additive to the drink.

In vm\_2, we have 2 additives “Cream” and “Sugar”. 2 is assigned for Sugar and 1 is assigned for Cream.

**Class DisposeDrinkAbstract**

|  |
| --- |
| DisposeAdditiveAbstract |
|  |
| abstract **void** DisposeDrink(int drink) |

This is abstract class of DisposeDrink. Which is used to add drink based on value passed.

**Class DisposeDrink\_vm1**

|  |
| --- |
| DisposeDrink\_vm1 |
|  |
| **void** DisposeDrink(int drink) |

**Operations**

**void** DisposeDrink(int drink) // Based on value Passed, it will dispose the drink.

In vm\_1, we have 2 drinks “Tea” and “Coffee”. 1 is assigned to Tea and 2 for Coffee.

**Class DisposeDrink\_vm2**

|  |
| --- |
| DisposeDrink\_vm2 |
|  |
| **void** DisposeDrink(int drink) |

**Operations**

**void** DisposeDrink(int drink) // Based on value Passed, it will dispose the drink.

In vm\_2, we have 1 drink “Coffee”. 1 is assigned to Coffee.

**Class IncreaseCFAbstract**

|  |
| --- |
| IncreaseCFAbstract |
|  |
| abstract **void** IncreaseCF(dataStore DS) |

This is abstract class of IncreaseCF. Which is used to change the value of cf in datastore.

**Class IncreaseCF\_vm1**

|  |
| --- |
| IncreaseCF\_vm1 |
|  |
| **void** IncreaseCF(dataStore DS) |

**Operations**

**void** IncreaseCF (dataStore DS) // This method changes the cf value in datastore. Get the value of temp\_v from datastore, get cf value from datastore and set cf to cf+temp\_v.

**Class IncreaseCF\_vm2**

|  |
| --- |
| IncreaseCF\_vm2 |
|  |
| **void** IncreaseCF(dataStore DS) |

**Operations**

**void** IncreaseCF (dataStore DS) // This method changes the cf value in datastore. Get the value of temp\_v from datastore, get cf value from datastore and set cf to cf+temp\_v.

**Class ReturnCoinsAbstract**

|  |
| --- |
| ReturnCoinsAbstract |
|  |
| abstract **void** ReturnCoins() |

This is abstract class of ReturnCoins. Which is used to return coins in the dataStore.

**Class ReturnCoins\_vm1**

|  |
| --- |
| ReturnCoins |
|  |
| **void** ReturnCoins() |

**Operations**

**void** ReturnCoins() // This method is used to return coins in dataStore vm1.

**Class ReturnCoins\_vm2**

|  |
| --- |
| ReturnCoins |
|  |
| **void** ReturnCoins() |

**Operations**

**void** ReturnCoins() // This method is used to return coins in dataStore vm2.

**Class StorePriceAbstract**

|  |
| --- |
| StorePriceAbstract |
|  |
| abstract **void** StorePrice(dataStore DS) |

This is abstract class of StorePrice. Which is used to store the value of price in datastore.

**Class StorePrice\_vm1**

|  |
| --- |
| StorePrice\_vm1 |
|  |
| **void** StorePrice(dataStore DS) |

**Operations**.

**void** StorePrice(dataStore DS) // This method is used to set the value of price in dataStore. It will take the value of temp\_p from the dataStore and assign it to price value.

**Class StorePrice\_vm2**

|  |
| --- |
| StorePrice\_vm2 |
|  |
| **void** StorePrice(dataStore DS) |

**Operations**.

**void** StorePrice(dataStore DS) // This method is used to set the value of price in dataStore. It will take the value of temp\_p from the dataStore and assign it to price value.

**Class ZeroCFAbstract**

|  |
| --- |
| ZeroCFAbstract |
|  |
| **void** ZeroCF(dataStore DS) |

This is abstract class of ZeroCF. Which is used to set the value of cf to zero.

**Class ZeroCF\_vm1**

|  |
| --- |
| ZeroCF\_vm1 |
|  |
| **void** ZeroCF(dataStore DS) |

**Operations**

**void** ZeroCF(dataStore DS) // This method is used to set the value of cf to zero in datastore.

**Class ZeroCF\_vm2**

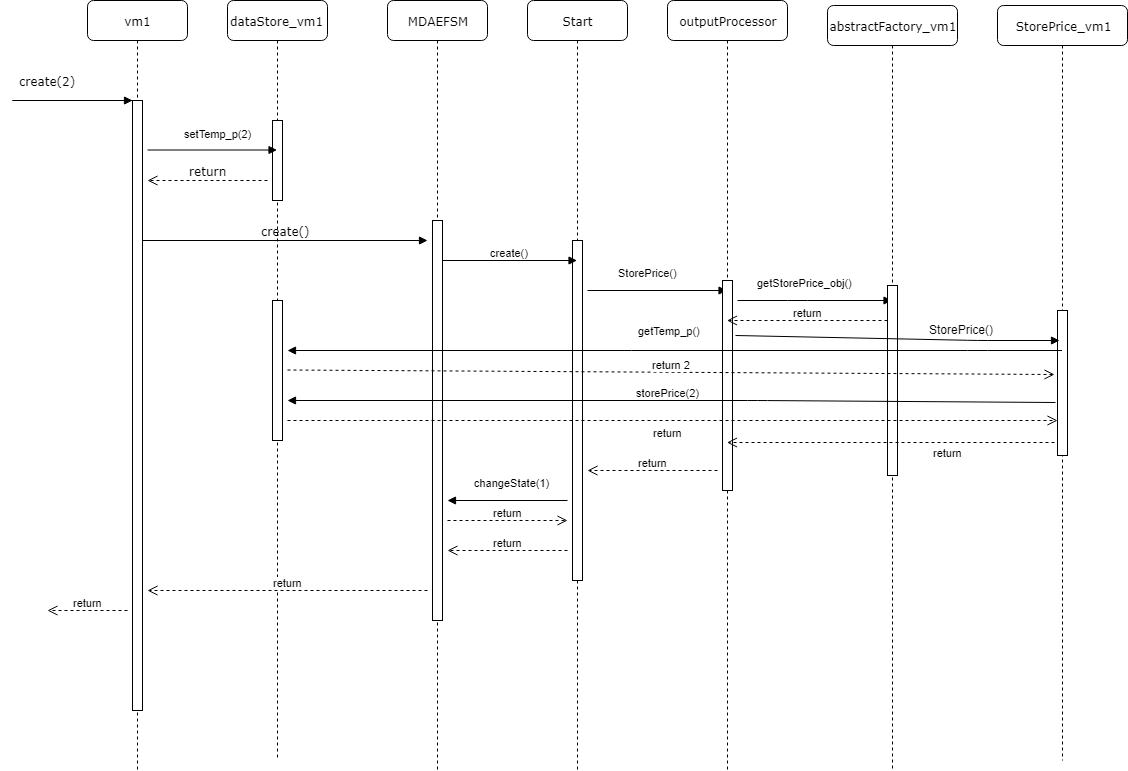
|  |
| --- |
| ZeroCF\_vm2 |
|  |
| **void** ZeroCF(dataStore DS) |

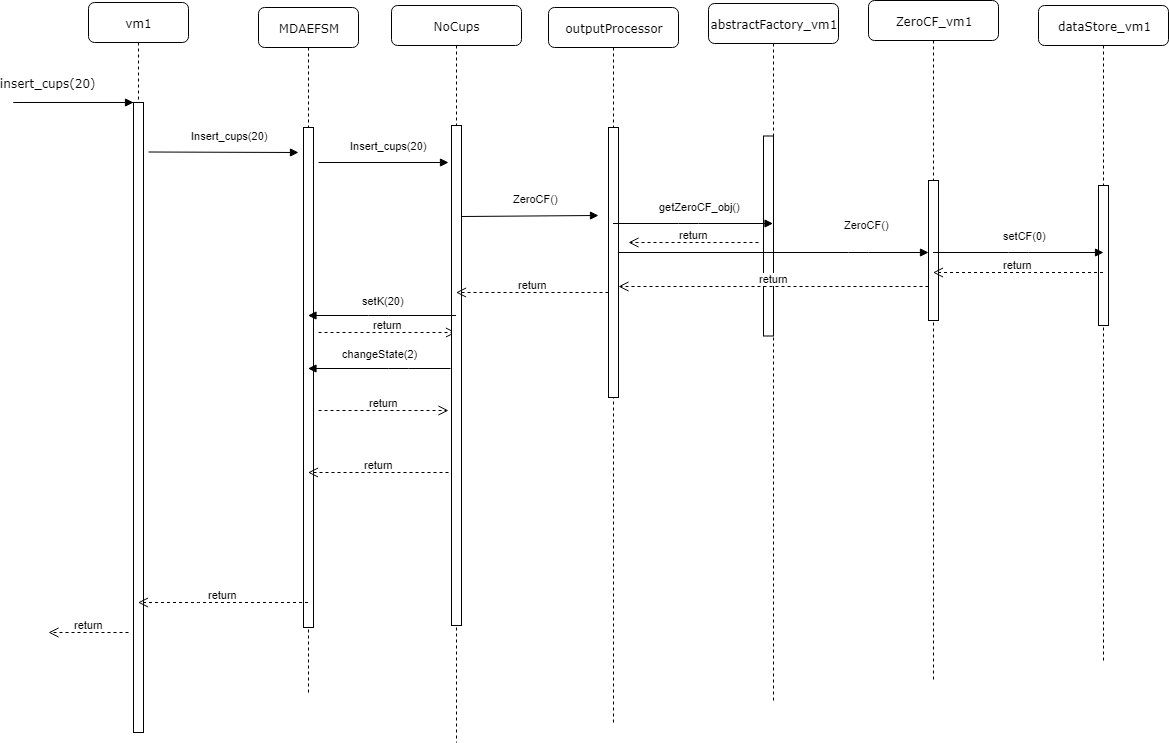
**Operations**

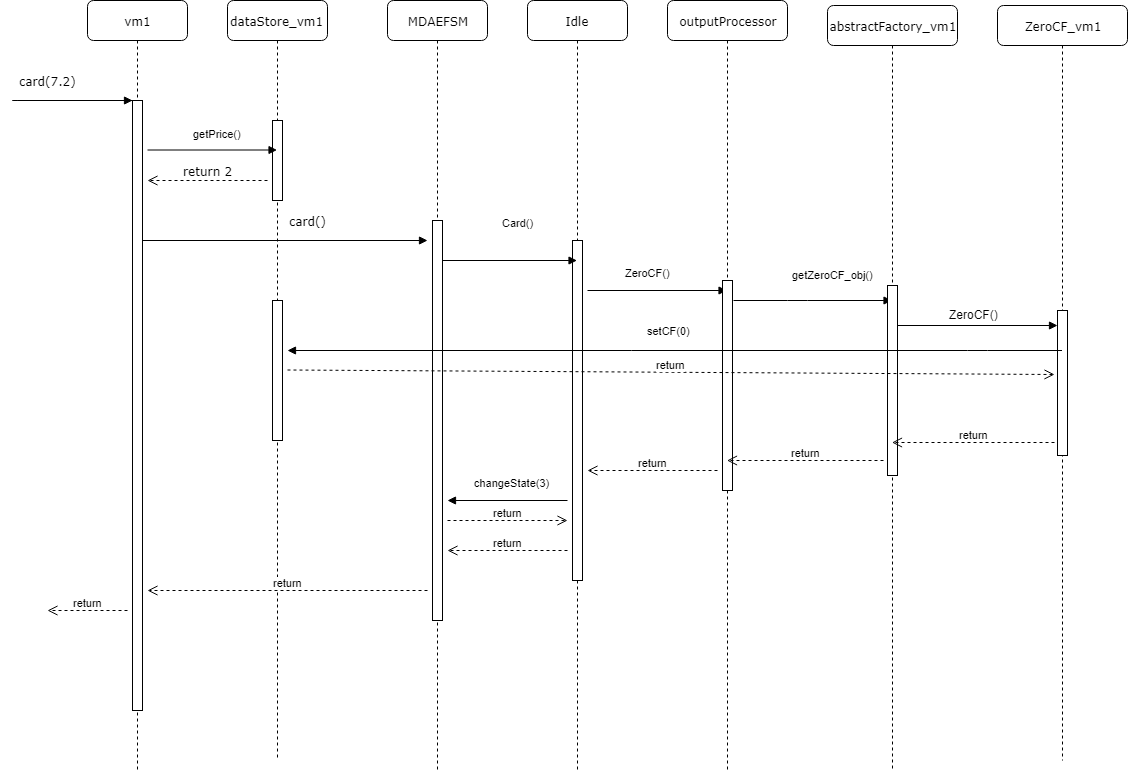
**void** ZeroCF(dataStore DS) // This method is used to set the value of cf to zero in datastore.

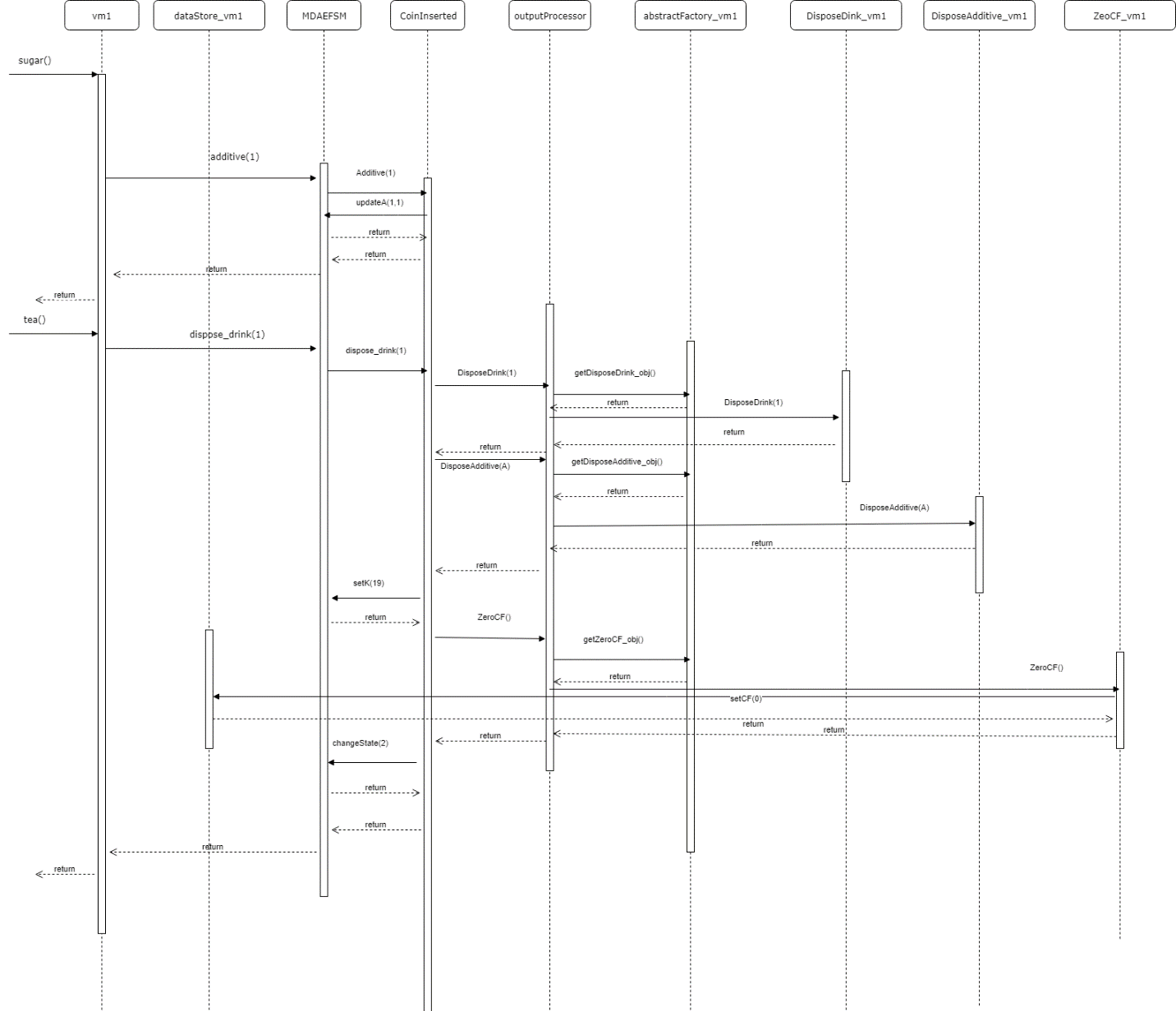
**Sequence Diagram**

VM1









VM2

