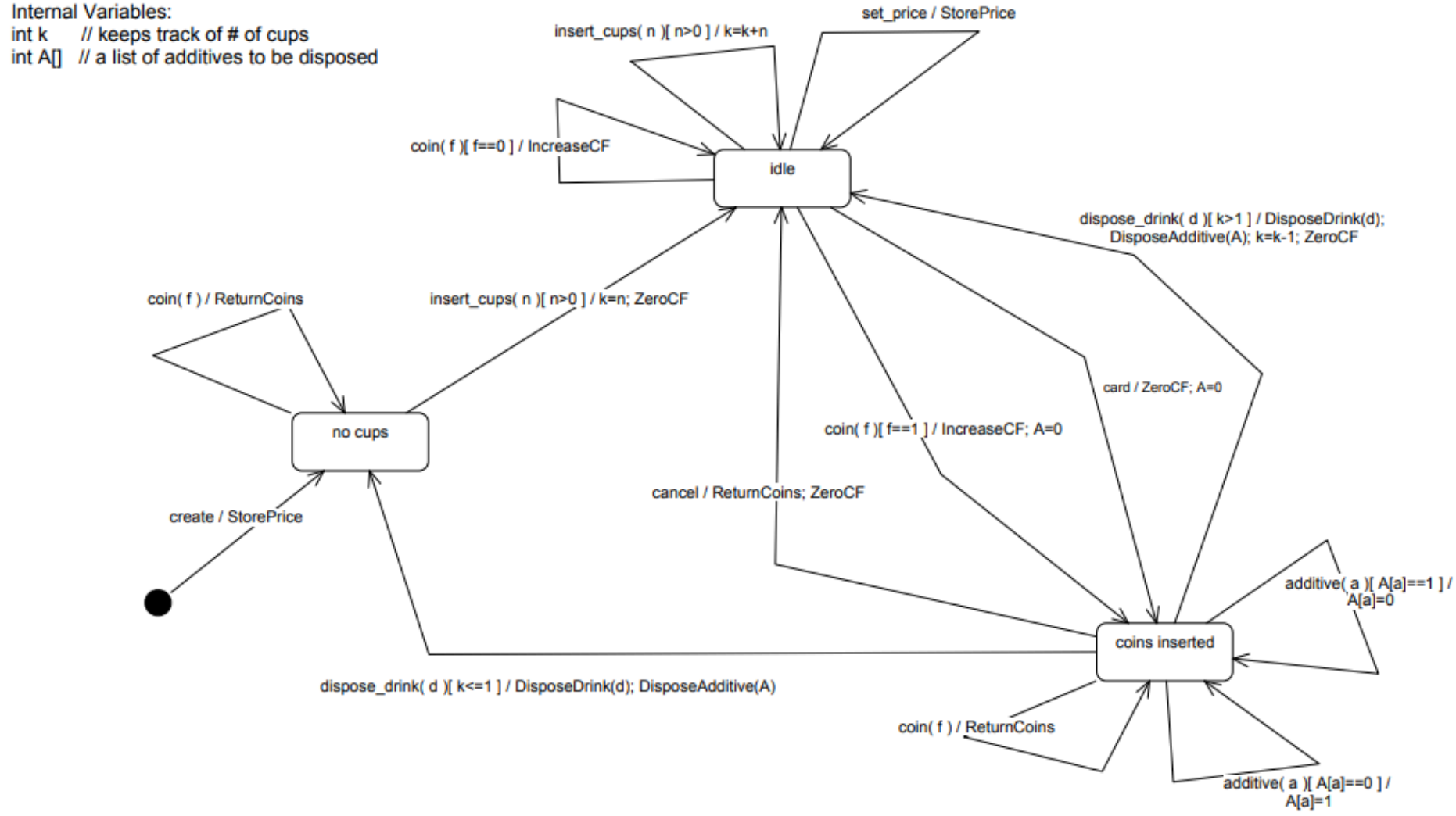
PROJECT DELIVERABLE
ANDRES GHERSI SAYAN
20539425

REPORT - PART 1:

a) Meta Events and b) Meta Actions

META EVENTS	
create()	Starts the machine and instantiates objects
insert_cups(int n)	n represents # of cups
coin(int f)	f=1: sufficient funds inserted for a drink f=0: not sufficient funds for a drink
card()	Card is used to pay
cancel()	Cancels the transaction
set_price()	Sets up the product price from temporary
dispose_drink(int d)	d represents a drink id
additive(int a)	a represents additive id
META ACTIONS	
StorePrice()	Stores the price
ZeroCF()	zeroes Cumulative Fund cf
IncreaseCF()	increases Cumulative Fund cf
ReturnCoins()	returns coins inserted for a drink
DisposeDrink(int d)	disposes a drink with d id
DisposeAdditive(int A[])	disposes marked additives in A list, where additive with i id is disposed when A[i]=1

c) State Diagram



d) Pseudo-Code of Input Processors (VM1, VM2)

CLASS VM1

```
m: EFSM
d: DS1
---
initialize(af: AbstractFactory)
    d = af.createDataStore()
    op = NEW OutputProcessor(af)
    m = NEW EFSM(op)
END
---
create(p: FLOAT)
    IF p > 0 THEN
        d.temp_p = p
        m.create()
    END IF
END
---
coin(v: FLOAT)
    IF v > 0 THEN
        d.temp_v = v
        IF (d.cf + v >= d.price) THEN
            m.coin(1) // Sufficient
        ELSE
            m.coin(0) // Insufficient
        END IF
    END IF
END
---
card(x: FLOAT)
    IF x >= d.price THEN
        m.card()
    END IF
END
---
sugar()
    m.additive(1)
END
---
```

```
cappuccino()
    m.dispose_drink(1)
END
```

```
chocolate()
    m.dispose_drink(2)
END
```

```
insert_cups(n: INTEGER)
    m.insert_cups(n)
END
```

```
set_price(p: FLOAT)
    IF p > 0 THEN
        d.temp_p = p
        m.set_price()
    END IF
END
```

```
cancel()
    m.cancel()
END
```

```
CLASS VM2
```

```
m: EFSM
d: DS2
```

```
initialize(af2: AbstractFactory)
    d = af2.createDataStore()
    op = NEW OutputProcessor(af2)
    m = NEW EFSM(op)
END
```

```
CREATE(p: INTEGER)
    IF p > 0 THEN
        d.temp_p = p
        m.create()
    END IF
```

```
END
```

```
COIN(v: INTEGER)
    IF v > 0 THEN
        d.temp_v = v
        IF (d.cf + v >= d.price) THEN
            m.coin(1) // Sufficient
        ELSE
            m.coin(0) // Insufficient
        END IF
    END IF
END
```

```
SUGAR()
    m.additive(2)
END
```

```
CREAM()
    m.additive(1)
END
```

```
COFFEE()
    m.dispose_drink(1)
END
```

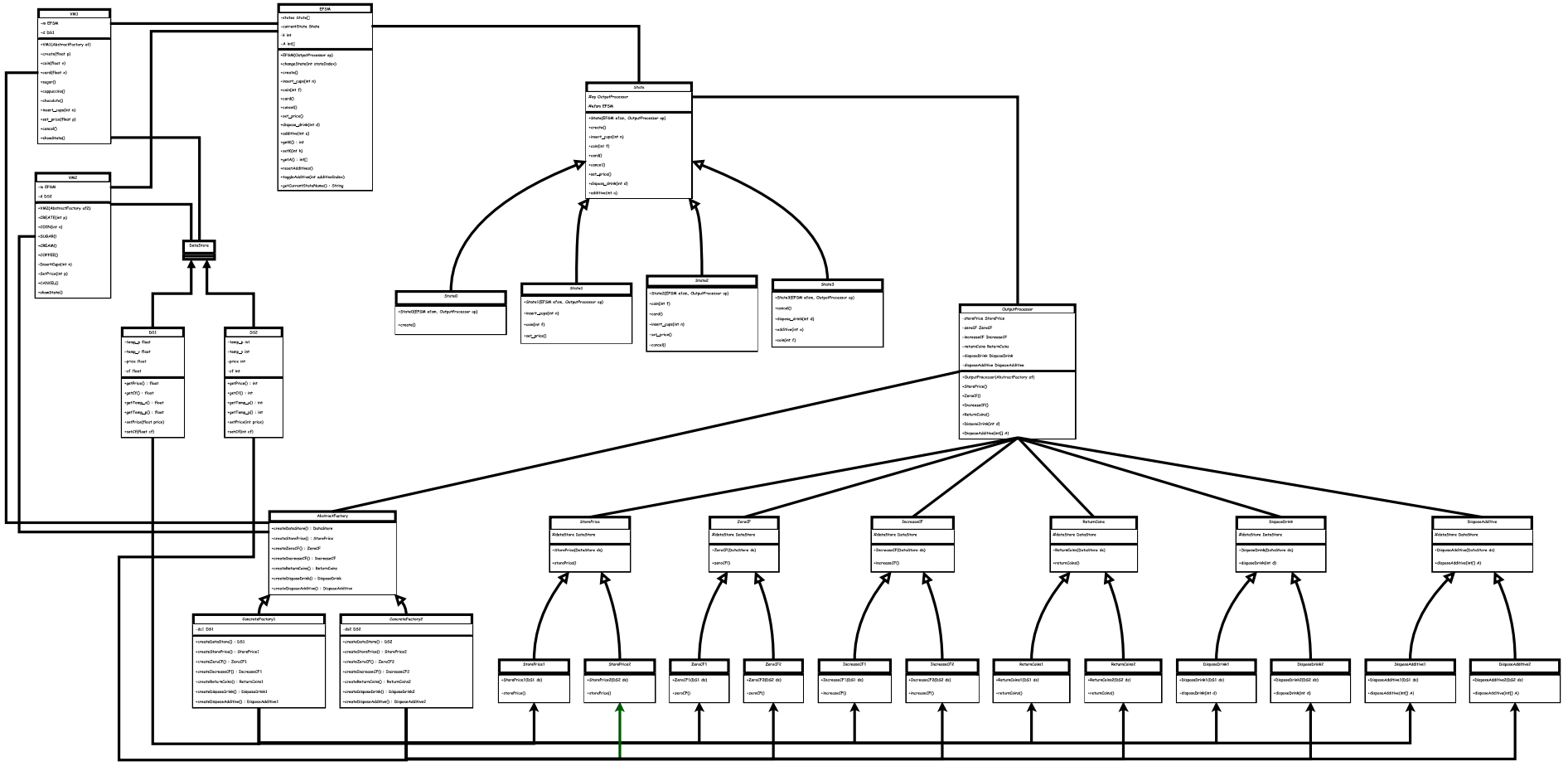
```
InsertCups(n: INTEGER)
    m.insert_cups(n)
END
```

```
SetPrice(p: INTEGER)
    IF p > 0 THEN
        d.temp_p = p
        m.set_price()
    END IF
END
```

```
CANCEL()
    m.cancel()
END
```

REPORT PART 2: CLASS DIAGRAM

(inserted next page)


VM1
-e: EFSM
-c: DS1
+VM1(AbstractFactory af)
+create(float p)
+coin(float n)
+cancel(float n)
+again()
+capputchi()
+hacuba()
+insert_cups(int n)
+set_price(float p)
+cancel()
+showstate()
EFSM
-states: State[]
-currentState: State
-S: int
-A: int[]
+EFSM(OutputProcessor op)
+changeState(int stateIndex)
+create()
+insert_cups(int n)
+coin(int f)
+card()
+cancel()
+set_price()
+dispose_drink(int d)
+additive(int n)
+getQ : int
+setQ(int k)
+getA() : int[]
+resetAdditives()
+toggleAdditive(int additiveIndex)
+getCurrentStateName() : String
State
#Key: OutputProcessor
#efsm: EFSM
+State(EFSM efsm, OutputProcessor op)
+create()
+insert_cups(int n)
+coin(int f)
+card()
+cancel()
+set_price()
+dispose_drink(int d)
+additive(int n)
VM2
-e: EFSM
-d: DS2
+VM2(AbstractFactory af2)
-CREATE(int p)
-COIN(int v)
-SUGAR()
-CREAM()
-COFFEE()
-InsertCups(int n)
-SetPrice(int p)
-CANCEL()
+showState()
DataStore
DS1
-temp_a: float
-temp_r: float
-price: float
-cf: float
+getPrice() : float
+getCF() : float
+getTemp_a() : float
+getTemp_g() : float
+setPrice(float price)
+setCF(float cf)
DS2
-temp_a: int
-temp_r: int
-price: int
-cf: int
+getPrice() : int
+getCF() : int
+getTemp_a() : int
+getTemp_g() : int
+setPrice(int price)
+setCF(int cf)
State0
+State0(EFSM efsm, OutputProcessor op)
+create()
State1
+State1(EFSM efsm, OutputProcessor op)
+insert_cups(int n)
+coin(int f)
+set_price()
State2
+State2(EFSM efsm, OutputProcessor op)
+coin(int f)
+card()
+insert_cups(int n)
+set_price()
-cancel()
State3
+State3(EFSM efsm, OutputProcessor op)
+cancel()
+dispose_drink(int d)
+additive(int n)
+coin(int f)
OutputProcessor
-storePrice: StorePrice
-zeroCF: ZeroCF
-increaseCF: IncreaseCF
-returnCoins: ReturnCoins
-disposeDrink: DisposeDrink
-disposeAdditive: DisposeAdditive
+OutputProcessor(AbstractFactory af)
+StorePrice()
+ZeroCF()
+IncreaseCF()
+ReturnCoins()
+DisposeDrink(int d)
+DisposeAdditive(int[] A)
AbstractFactory
+createDataStore() : DataStore
+createStorePrice() : StorePrice
+createZeroCF() : ZeroCF
+createIncreaseCF() : IncreaseCF
+createReturnCoins() : ReturnCoins
+createDisposeDrink() : DisposeDrink
+createDisposeAdditive() : DisposeAdditive
ConcreteFactory1
-ds1: DS1
+createDataStore() : DS1
+createStorePrice() : StorePrice1
+createZeroCF() : ZeroCF1
+createIncreaseCF() : IncreaseCF1
+createReturnCoins() : ReturnCoins1
+createDisposeDrink() : DisposeDrink1
+createDisposeAdditive() : DisposeAdditive1
ConcreteFactory2
-ds2: DS2
+createDataStore() : DS2
+createStorePrice() : StorePrice2
+createZeroCF() : ZeroCF2
+createIncreaseCF() : IncreaseCF2
+createReturnCoins() : ReturnCoins2
+createDisposeDrink() : DisposeDrink2
+createDisposeAdditive() : DisposeAdditive2
StorePrice
#DataStore DataStore
+StorePrice(DataStore ds)
+storePrice()
ZeroCF
#DataStore DataStore
-ZeroCF(DataStore ds)
+zeroCF()
IncreaseCF
#DataStore DataStore
+IncreaseCF(DataStore ds)
+increaseCF()
ReturnCoins
#DataStore DataStore
+ReturnCoins(DataStore ds)
+returnCoins()
DisposeDrink
#DataStore DataStore
+DisposeDrink(DataStore ds)
+disposeDrink(int d)
DisposeAdditive
#DataStore DataStore
+DisposeAdditive(DataStore ds)
+disposeAdditive(int[] A)
StorePrice1
-StorePrice1(DS1 ds)
-storePrice()
StorePrice2
+StorePrice2(DS2 ds)
+storePrice()
ZeroCF1
+ZeroCF1(DS1 ds)
+zeroCF()
ZeroCF2
+ZeroCF2(DS2 ds)
+zeroCF()
IncreaseCF1
+IncreaseCF1(DS1 ds)
+increaseCF()
IncreaseCF2
+IncreaseCF2(DS2 ds)
+increaseCF()
ReturnCoins1
+ReturnCoins1(DS1 ds)
+returnCoins()
ReturnCoins2
+ReturnCoins2(DS2 ds)
+returnCoins()
DisposeDrink1
+DisposeDrink1(DS1 ds)
+disposeDrink(int d)
DisposeDrink2
+DisposeDrink2(DS2 ds)
+disposeDrink(int d)
DisposeAdditive1
+DisposeAdditive1(DS1 ds)
+disposeAdditive(int[] A)
DisposeAdditive2
+DisposeAdditive2(DS2 ds)
+disposeAdditive(int[] A)

REPORT PART 3:

a) Description of Classes and Functions:

AbstractFactory: Abstract Factory interface.
- createDataStore(): Creates a DataStore object.
- createStorePrice(): Creates a StorePrice object.
- createZeroCF(): Creates a ZeroCF object.
- createIncreaseCF(): Creates an IncreaseCF object.
- createReturnCoins(): Creates a ReturnCoins object.
- createDisposeDrink(): Creates a DisposeDrink object.
- createDisposeAdditive(): Creates a DisposeAdditive object.

OutputProcessor: Executes concrete operation methods.
- StorePrice(): Calls storePrice method.
- ZeroCF(): Calls zeroCF method.
- IncreaseCF(): Calls increaseCF method.
- ReturnCoins(): Calls returnCoins method.
- DisposeDrink(int d): Calls disposeDrink method.
- DisposeAdditive(int[] A): Calls disposeAdditive method.

State: Abstract State class for the EFSM.
- create(): Handles create event.
- insert_cups(int n): Handles insert_cups event.
- coin(int f): Handles coin event.
- card(): Handles card event.
- cancel(): Handles cancel event.
- set_price(): Handles set_price event.
- dispose_drink(int d): Handles dispose_drink event.
- additive(int a): Handles additive event.

DisposeAdditive: Abstract class for the DisposeAdditive action.
    disposeAdditive(int[] A): Disposes selected additives.

DisposeDrink: Abstract class for the DisposeDrink action.
    disposeDrink(int d): Disposes the selected drink.

IncreaseCF: Abstract class for the IncreaseCF action.
    increaseCF(): Increases cumulative funds.

ReturnCoins: Abstract class for the ReturnCoins action.
    returnCoins(): Returns inserted coins.

StorePrice: Abstract class for the StorePrice action.
    storePrice(): Stores the temporary price.

ZeroCF: Abstract class for the ZeroCF action.
    zeroCF(): Resets cumulative funds to zero.

DataStore: Abstract Data Store class.

EFSM: The Extended Finite State Machine.
    changeState(int stateIndex): Changes the current state.
    create(): Triggers create event in current state.
    insert_cups(int n): Triggers insert_cups event in current state.
    coin(int f): Triggers coin event in current state.
    card(): Triggers card event in current state.
    cancel(): Triggers cancel event in current state.
    set_price(): Triggers set_price event in current state.
    dispose_drink(int d): Triggers dispose_drink event in current state.
    additive(int a): Triggers additive event in current state.
    getK(): Gets number of cups.

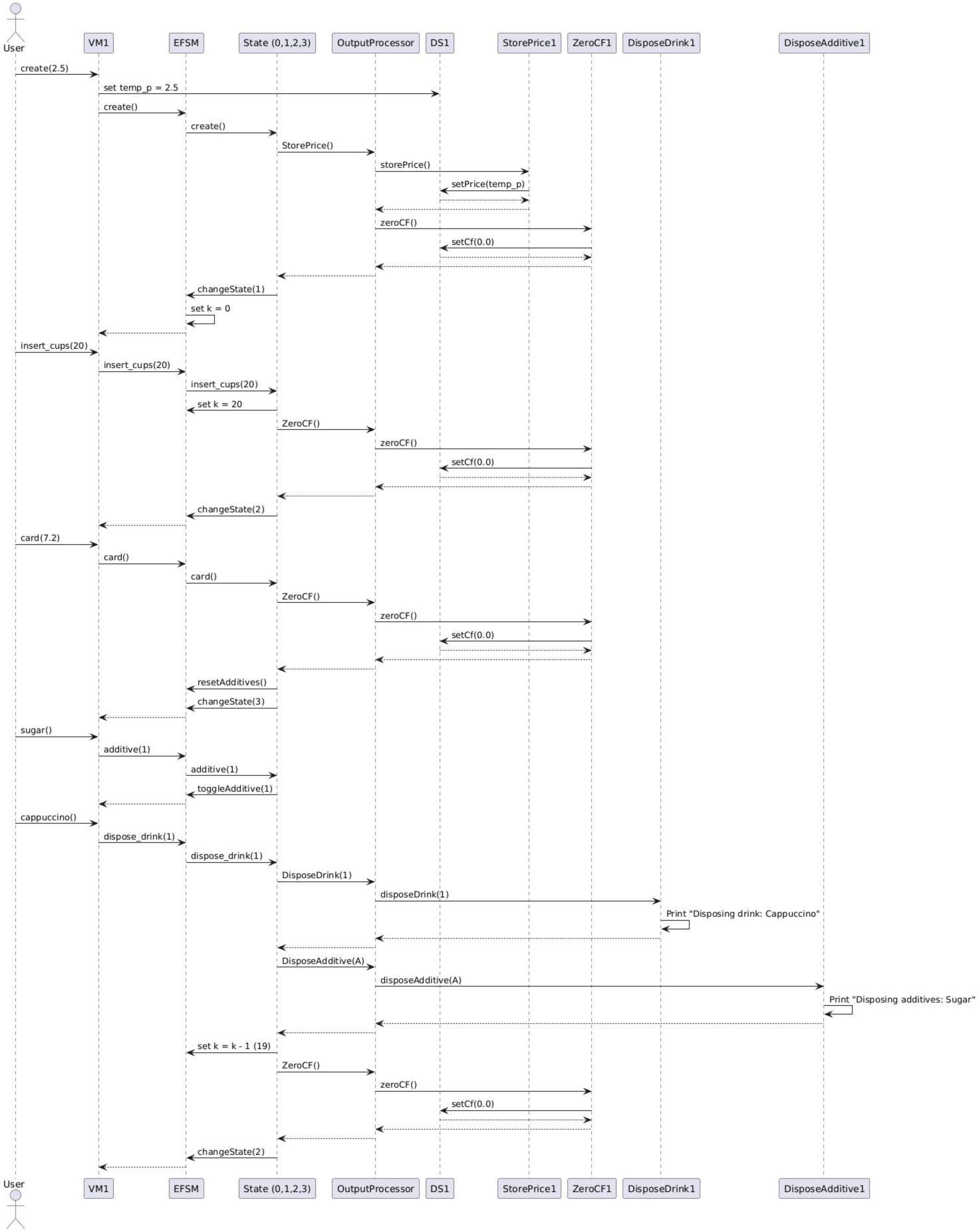
setK(int k): Sets number of cups.
getA(): Gets additives array.
resetAdditives(): Resets the additives array.
toggleAdditive(int additiveIndex): Toggles an additive.
getCurrentStateName(): Gets current state's name.

VM1: Vending Machine 1 implementation.
create(float p): Creates product with price.
coin(float v): Inserts a coin.
card(float x): Processes card payment.
sugar(): Selects sugar additive.
cappuccino(): Selects cappuccino drink.
chocolate(): Selects chocolate drink.
insert_cups(int n): Inserts cups.
set_price(float p): Sets product price.
cancel(): Cancels current transaction.
showState(): Displays current state info.

VM2: Vending Machine 2 implementation.
CREATE(int p): Creates product with price.
COIN(int v): Inserts a coin.
SUGAR(): Selects sugar additive.
CREAM(): Selects cream additive.
COFFEE(): Selects coffee drink.
InsertCups(int n): Inserts cups.
SetPrice(int p): Sets product price.
CANCEL(): Cancels current transaction.
showState(): Displays current state info.

REPORT PART 4: SEQUENCE 1 AND 2

VM-1 Cappuccino Sequence



VM-2 Coffee Sequence