SSA Final Project Report

```
- Thomas Ehling A20432671
```

3 Class & Operations descriptions

Abstract Factory Pattern :

VM_1 :

```
Class to handle user interactions for the VM1.
```

- **create**(int p) : Init and set the price to p
- **coin**(int v): Insert v coins.
- **card**(float x): Pay x with the card
- **sugar**(): Ask to add sugar as additive
- **tea**(): Ask to dispose a cup of tea
- **chocolate**(): Ask to dispose a cup of chocolate
- **insert_cups**(int n): Insert n cups
- **set_price**(int p): Set the price for a drink to p
- **cancel** (): Ask to cancel the current transaction.

VM_2:

```
Class to handle user interactions for the VM2.
```

- **CREATE**(float p) : Init and set the price to p
- **COIN**(float v): Insert v coins.
- **SUGAR**(): Ask to add sugar as additive
- **CREAM**(): Ask to add cream as additive
- **COFFEE**(): Ask to dispose a cup of coffee
- **InsertCups**(int n): Insert n cups
- **SetPrice**(float p): Set the price for a drink to p
- **CANCEL** (): Ask to cancel the current transaction.

Abstract_Factory:

```
Abstract class for the concrete factory objects
```

Fact_VM_1:

```
Concrete Factory object for VM1.
```

- **createDS**(): create an instance of DS_1 and return the pointer.
- **createSP**(): create an instance of StorePrice_Int and return the pointer.
- **createZCF**(): create an instance of ZeroCF_Int and return the pointer.
- **createICF**(): create an instance of IncreaseCF_Int and return the pointer.
- **createRC**(): create an instance of ReturnCoins_Int and return the pointer.
- **createDD**(): create an instance of DisposeDrink_1 and return the pointer.
- **createDA**(): create an instance of DisposeAdd_1 and return the pointer.

Fact_VM_2:

```
Concrete Factory object for VM2.
```

- **createDS**(): create an instance of DS_2 and return the pointer.
- **createSP**(): create an instance of StorePrice_Float and return the pointer.
- **createZCF**(): create an instance of ZeroCF_Float and return the pointer.
- **createICF**(): create an instance of IncreaseCF_Int and return the pointer.
- **createRC**(): create an instance of ReturnCoins_Float and return the pointer.
- **createDD**(): create an instance of DisposeDrink_2 and return the pointer.
- **createDA**(): create an instance of DisposeAdd_2 and return the pointer.

OP:

```
Output Processor. Get actions from State and redirect them to the
Strategy pattern classes with the pointers.
```

Arguments : pointers to DataStore and every Abstract classes of the strategy pattern.

- **StorePrice(), ZeroCF(), IncreaseCF()** : call by State, redirect to the right class with the corresponding pointer, putting the pointer to Dtastore as an argument.

- **ReturnCoins(),DisposeDrink(in d:int), DisposeAdditive(in int A[])**: call by State, redirect to the right

class with the corresponding pointer.

Strategy Pattern

DataStore:

```
Abstract class to store the data
```

DS_1:

```
Inherited from DataStore, it stores the data for the VM1.
```

Attributes :

- **temp_p** : temporary price value
- **temp_v** : temporary balance value
- **price** : price value
- **cf** : balance value

DS_2:

```
Inherited from DataStore, it stores the data for the VM2.
```

Attributes :

- **temp_p** : temporary price value
- **temp_v** : temporary balance value
- **price** : price value
- **cf** : balance value

A_DisposeAdditive:

```
Abstract class for the DisposeAdditive() function.
```

DisposeAdditive_1:

```
Concrete implementation of the DisposeAdditive() function for VM1.
```

- **DisposeAdditive**(A[]): Dispose additives listed in A.
-

DisposeAdditive_2:

```
Concrete implementation of the DisposeAdditive() function for VM1.
```

- **DisposeAdditive**(A[]): Dispose additives listed in A.

A_Dispose_Drink:

```
Abstract class for the DisposeDrink() function.
```

Dispose_Drink_1:

```
Concrete implementation of the DisposeDrink() function for VM1.
```

- **DisposeDrink**(i): Dispose the drink with ID i.

Dispose_Drink_2:

```
Concrete implementation of the DisposeDrink() function for VM2.
```

- **DisposeDrink**(i): Dispose the drink with ID i.

A_IncreaseCF:

```
Abstract class for the IncreaseCF() function.
```

IncreaseCF_Int:

```
Concrete implementation of the IncreaseCF () function for VM1.
```

- **IncreaseCF** (Datastore * ds): Increases cf in DS_1 using temp_v.

IncreaseCF_Float:

```
Concrete implementation of the IncreaseCF () function for VM2.
```

- **IncreaseCF** (Datastore * ds): Increases cf in DS_2 using temp_v.

A_ReturnCoins:

```
Abstract class for the ReturnCoins () function.
```

ReturnCoins_Int:

```
Concrete implementation of the ReturnCoins () function for VM1.
```

- **ReturnCoins** (): Returns coins.

ReturnCoins_Float:

```
Concrete implementation of the ReturnCoins () function for VM2.
```

- **ReturnCoins** (): Returns coins.

A_StorePrice:

```
Abstract class for the StorePrice() function.
```

StorePrice_Int:

```
Concrete implementation of the StorePrice () function for VM1.
```

- **StorePrice** (Datastore * ds): Stores the price in DS_1 from temp_p.

StorePrice_Float:

```
Concrete implementation of the StorePrice () function for VM2.
```

- **StorePrice** (Datastore * ds): Stores the price in DS_2 from temp_p.

A_ZeroCF:

```
Abstract class for the ZeroCF () function.
```

ZeroCF_Int:

```
Concrete implementation of the ZeroCF () function for VM1.
```

- **ZeroCF** (Datastore * ds): Resets cf to 0 in DS_1.

ZeroCF_Float:

```
Concrete implementation of the ZeroCF () function for VM2.
```

- **ZeroCF** (Datastore * ds): Resets cf to 0 in DS_1.

State Pattern : Decentralized version

MDA:

```
Model Driven Architecture class, interface between VM and states
```

Attributes :

- list *LS* : list of all states
- State *current* : pointer to the current state

Operations:

- **create(), insertcups(int n), coin(), card(), cancel(), set_price(), dispose_drink(int id), additive(int id)** : call the same function on the current state pointer
- **changeState(int ind)** : change the value of the current pointer to the state in LS[ind]. Call by the current state class.

State:

```
Abstract class for the model states, take order from MDA and redirect them to OP.
```

Attributes :

- OP * *op* : pointer to an instance of OP
- StatesData * *sd* : pointer to an instance of StatesData

Operations: abstract funcitons, detailed fot each states in the corresponding class.

StatesData:

```
Class for storing the data related to the State pattern
```

- int *k* : number of cups
- List a: list of all current additives.

StateStart:

```
Class for the State Start
```

- create(): call StorePrice() of OP, change the current state to StateNoCups

StateNoCups

```
Class for the State No_Cups
```

- coins(int f): call ReturnCoins()
- insert_cups(int n): if n>0, stores it into *k*, and change the current state to StateIdle

StateIdle:

```
Class for the State Idle
```

- coins(int f): call IncreaseCF, if f==1: change the current state to StateCoinInserted and reset *a*
- insert_cups(int n): if n>0, stores it into *k*
- set_price(): call Storeprice()
- card(): call ZeroCF() and reset *a*, change the current state to StateCoinInserted

StateCoinInserted:

```
Class for the State Coin_Inserted
```

- coins(int f): call ReturnCoins()
- cancel(): call ReturnCoins(), ZeroCF(), change the current state to StateIdle
- dispose_drink(int id): call DisposeDrink(id) and DisposeAdd(id),
    - if k<=1 : change current state to StateNoCups
    - if k>1: set k to k-1, call ZeroCF(), change the current state to StateIdle
- additive(int id): if a[id] is 0/1 set it to 1/0