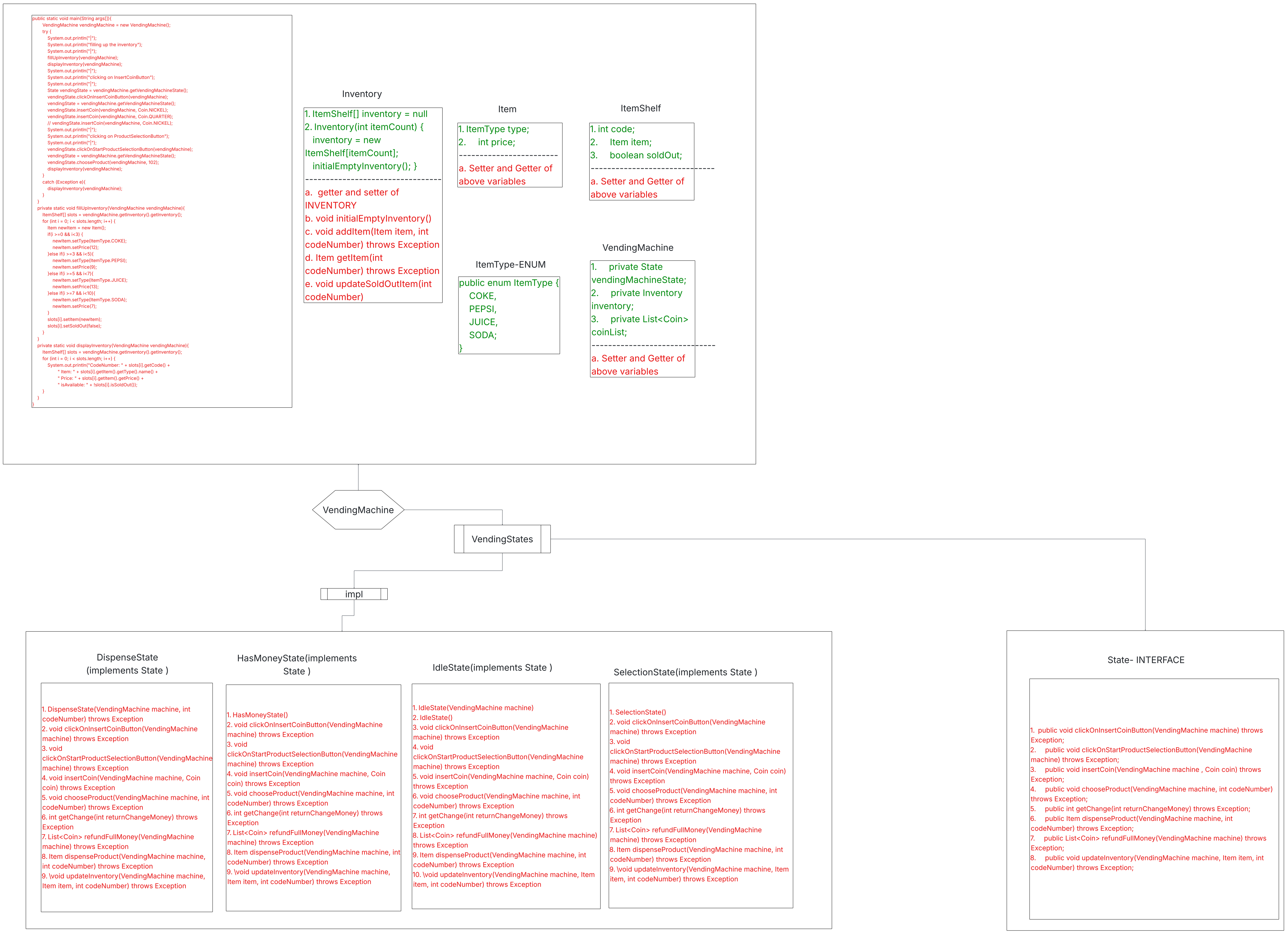

```
public static void main(String args[]){
    VendingMachine vendingMachine = new VendingMachine();
    try {
      System.out.println("|");
      System.out.println("filling up the inventory");
      System.out.println("|");
      fillUpInventory(vendingMachine);
      displayInventory(vendingMachine);
      System.out.println("|");
      System.out.println("clicking on InsertCoinButton");
      System.out.println("|");
      State vendingState = vendingMachine.getVendingMachineState();
      vendingState.clickOnInsertCoinButton(vendingMachine);
      vendingState = vendingMachine.getVendingMachineState();
      vendingState.insertCoin(vendingMachine, Coin.NICKEL);
      vendingState.insertCoin(vendingMachine, Coin.QUARTER);
      // vendingState.insertCoin(vendingMachine, Coin.NICKEL);
      System.out.println("|");
      System.out.println("clicking on ProductSelectionButton");
      System.out.println("|");
      vendingState.clickOnStartProductSelectionButton(vendingMachine);
      vendingState = vendingMachine.getVendingMachineState();
      vendingState.chooseProduct(vendingMachine, 102);
      displayInventory(vendingMachine);
    }
    catch (Exception e){
      displayInventory(vendingMachine);
    }
  }
  private static void fillUpInventory(VendingMachine vendingMachine){
    ItemShelf[] slots = vendingMachine.getInventory().getInventory();
    for (int i = 0; i < slots.length; i++) {
      Item newItem = new Item();
      if(i >=0 && i<3) {
        newItem.setType(ItemType.COKE);
        newItem.setPrice(12);
      }else if(i >=3 && i<5){
        newItem.setType(ItemType.PEPSI);
        newItem.setPrice(9);
      }else if(i >=5 && i<7){
        newItem.setType(ItemType.JUICE);
        newItem.setPrice(13);
      }else if(i >=7 && i<10){
        newItem.setType(ItemType.SODA);
        newItem.setPrice(7);
      }
      slots[i].setItem(newItem);
      slots[i].setSoldOut(false);
    }
  }
  private static void displayInventory(VendingMachine vendingMachine){
    ItemShelf[] slots = vendingMachine.getInventory().getInventory();
    for (int i = 0; i < slots.length; i++) {
      System.out.println("CodeNumber: " + slots[i].getCode() +
          " Item: " + slots[i].getItem().getType().name() +
          " Price: " + slots[i].getItem().getPrice() +
          " isAvailable: " + !slots[i].isSoldOut());
    }
  }
}
```

Inventory

```
1. ItemShelf[] inventory = null
2. Inventory(int itemCount) {
   inventory = new ItemShelf[itemCount];
   initialEmptyInventory(); }
---------------------------------
a.  getter and setter of INVENTORY
b. void initialEmptyInventory()
c. void addItem(Item item, int codeNumber) throws Exception
d. Item getItem(int codeNumber) throws Exception
e. void updateSoldOutItem(int codeNumber)
```

Item

```
1. ItemType type;
2.    int price;
------------------------
a. Setter and Getter of above variables
```

ItemShelf

```
1. int code;
2.    Item item;
3.    boolean soldOut;
-----------------------------
a. Setter and Getter of above variables
```

ItemType-ENUM

```
public enum ItemType {
   COKE,
   PEPSI,
   JUICE,
   SODA;
}
```

VendingMachine

```
1.    private State vendingMachineState;
2.    private Inventory inventory;
3.    private List<Coin> coinList;
-----------------------------
a. Setter and Getter of above variables
```

DispenseState (implements State)

```
1. DispenseState(VendingMachine machine, int codeNumber) throws Exception
2. void clickOnInsertCoinButton(VendingMachine machine) throws Exception
3. void clickOnStartProductSelectionButton(VendingMachine machine) throws Exception
4. void insertCoin(VendingMachine machine, Coin coin) throws Exception
5. void chooseProduct(VendingMachine machine, int codeNumber) throws Exception
6. int getChange(int returnChangeMoney) throws Exception
7. List<Coin> refundFullMoney(VendingMachine machine) throws Exception
8. Item dispenseProduct(VendingMachine machine, int codeNumber) throws Exception
9. \void updateInventory(VendingMachine machine, Item item, int codeNumber) throws Exception
```

HasMoneyState(implements State)

```
1. HasMoneyState()
2. void clickOnInsertCoinButton(VendingMachine machine) throws Exception
3. void clickOnStartProductSelectionButton(VendingMachine machine) throws Exception
4. void insertCoin(VendingMachine machine, Coin coin) throws Exception
5. void chooseProduct(VendingMachine machine, int codeNumber) throws Exception
6. int getChange(int returnChangeMoney) throws Exception
7. List<Coin> refundFullMoney(VendingMachine machine) throws Exception
8. Item dispenseProduct(VendingMachine machine, int codeNumber) throws Exception
9. \void updateInventory(VendingMachine machine, Item item, int codeNumber) throws Exception
```

IdleState(implements State)

```
1. IdleState(VendingMachine machine)
2. IdleState()
3. void clickOnInsertCoinButton(VendingMachine machine) throws Exception
4. void clickOnStartProductSelectionButton(VendingMachine machine) throws Exception
5. void insertCoin(VendingMachine machine, Coin coin) throws Exception
6. void chooseProduct(VendingMachine machine, int codeNumber) throws Exception
7. int getChange(int returnChangeMoney) throws Exception
8. List<Coin> refundFullMoney(VendingMachine machine) throws Exception
9. Item dispenseProduct(VendingMachine machine, int codeNumber) throws Exception
10. \void updateInventory(VendingMachine machine, Item item, int codeNumber) throws Exception
```

SelectionState(implements State)

```
1. SelectionState()
2. void clickOnInsertCoinButton(VendingMachine machine) throws Exception
3. void clickOnStartProductSelectionButton(VendingMachine machine) throws Exception
4. void insertCoin(VendingMachine machine, Coin coin) throws Exception
5. void chooseProduct(VendingMachine machine, int codeNumber) throws Exception
6. int getChange(int returnChangeMoney) throws Exception
7. List<Coin> refundFullMoney(VendingMachine machine) throws Exception
8. Item dispenseProduct(VendingMachine machine, int codeNumber) throws Exception
9. \void updateInventory(VendingMachine machine, Item item, int codeNumber) throws Exception
```

State- INTERFACE

```
1.  public void clickOnInsertCoinButton(VendingMachine machine) throws Exception;
2.    public void clickOnStartProductSelectionButton(VendingMachine machine) throws Exception;
3.    public void insertCoin(VendingMachine machine , Coin coin) throws Exception;
4.    public void chooseProduct(VendingMachine machine, int codeNumber) throws Exception;
5.    public int getChange(int returnChangeMoney) throws Exception;
6.    public Item dispenseProduct(VendingMachine machine, int codeNumber) throws Exception;
7.    public List<Coin> refundFullMoney(VendingMachine machine) throws Exception;
8.    public void updateInventory(VendingMachine machine, Item item, int codeNumber) throws Exception;
```