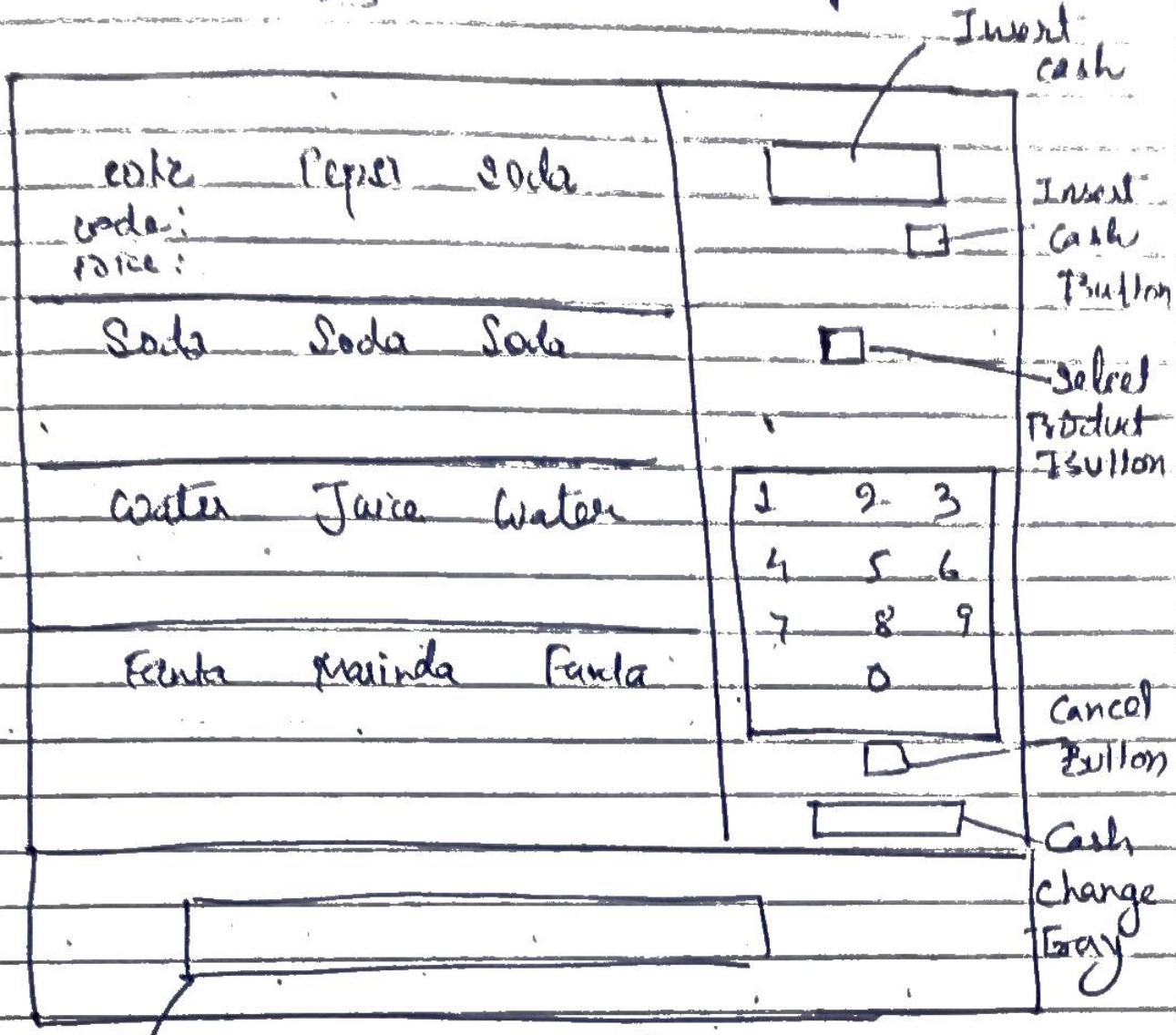
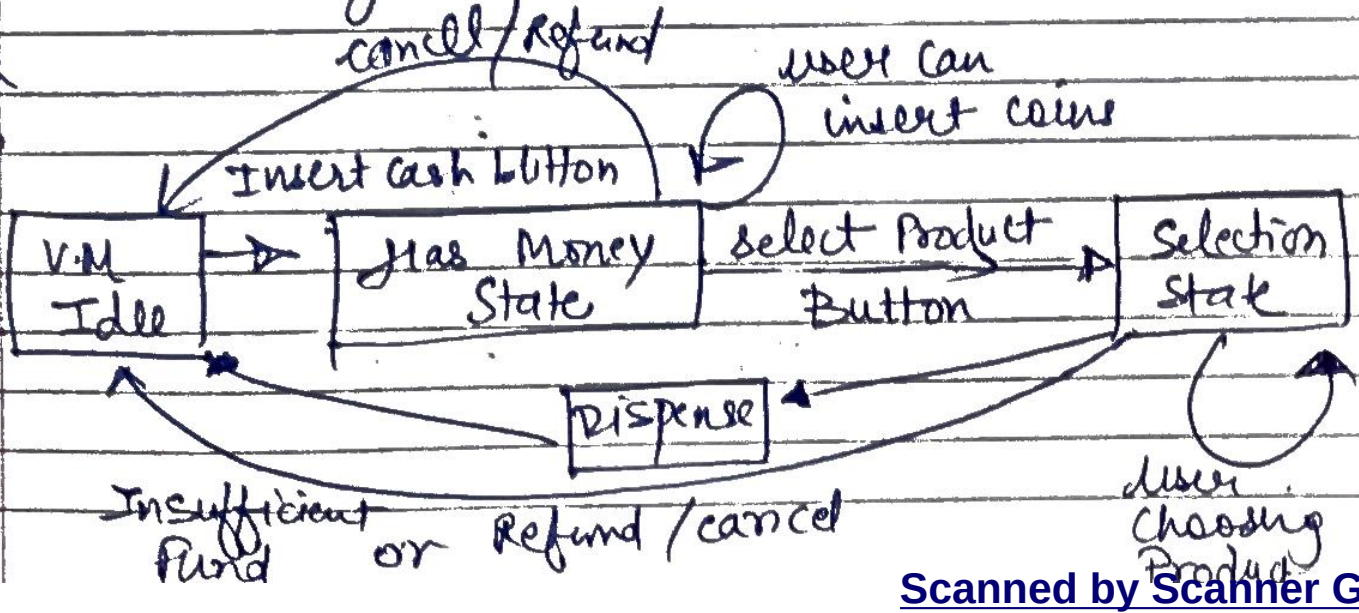
Vending Machine Design



Rough Flow



Page No.
Date / /

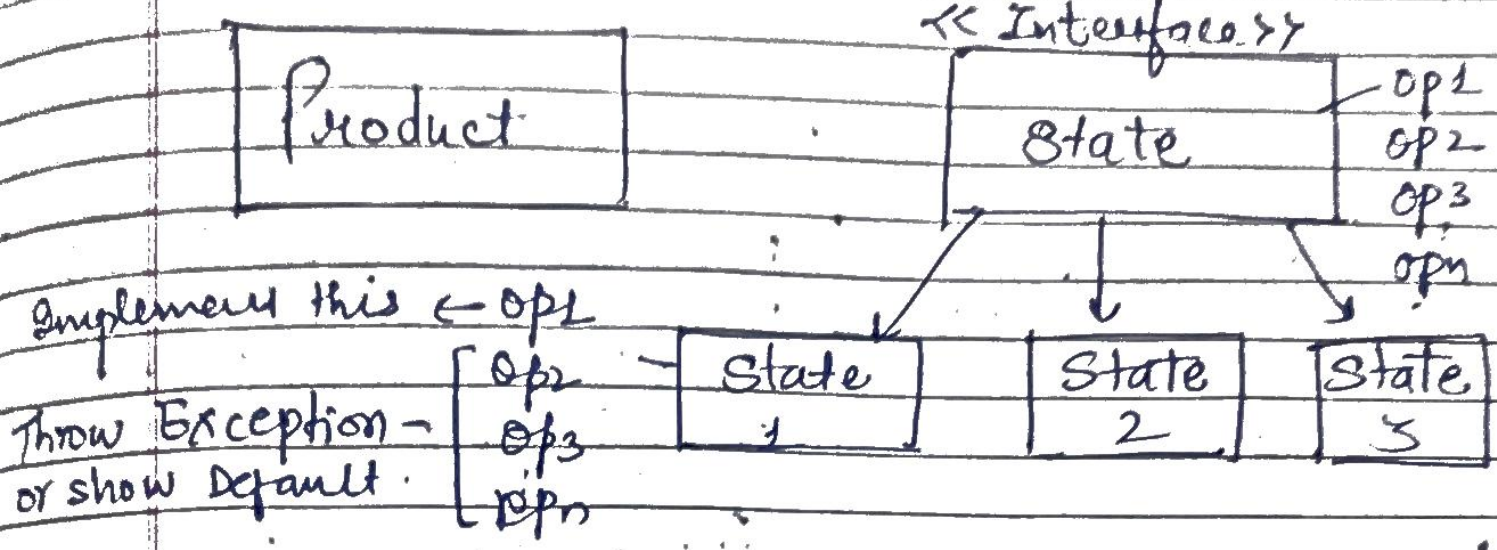
State	operation
Idle	Press insert Cash Button
Has Money	Insert Coin, Select Product Button Cancel Button
Selection State	choose Product; Cancel / Refund Return change
Dispense State	Product Dispense.

- If at one state some specific operations are performed then we will use State design Pattern

Ex. Design T.V

State	
ON	change channel, change volume off tv.
OFF	ON Button

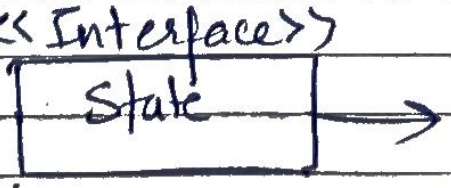
STATE DESIGN PATTERN



→ If the product has many states then the interface will define all operations that can be performed by product

→ If state 1 performs op1 then it will implement that and show default behaviour or exception for other operations.

State Design

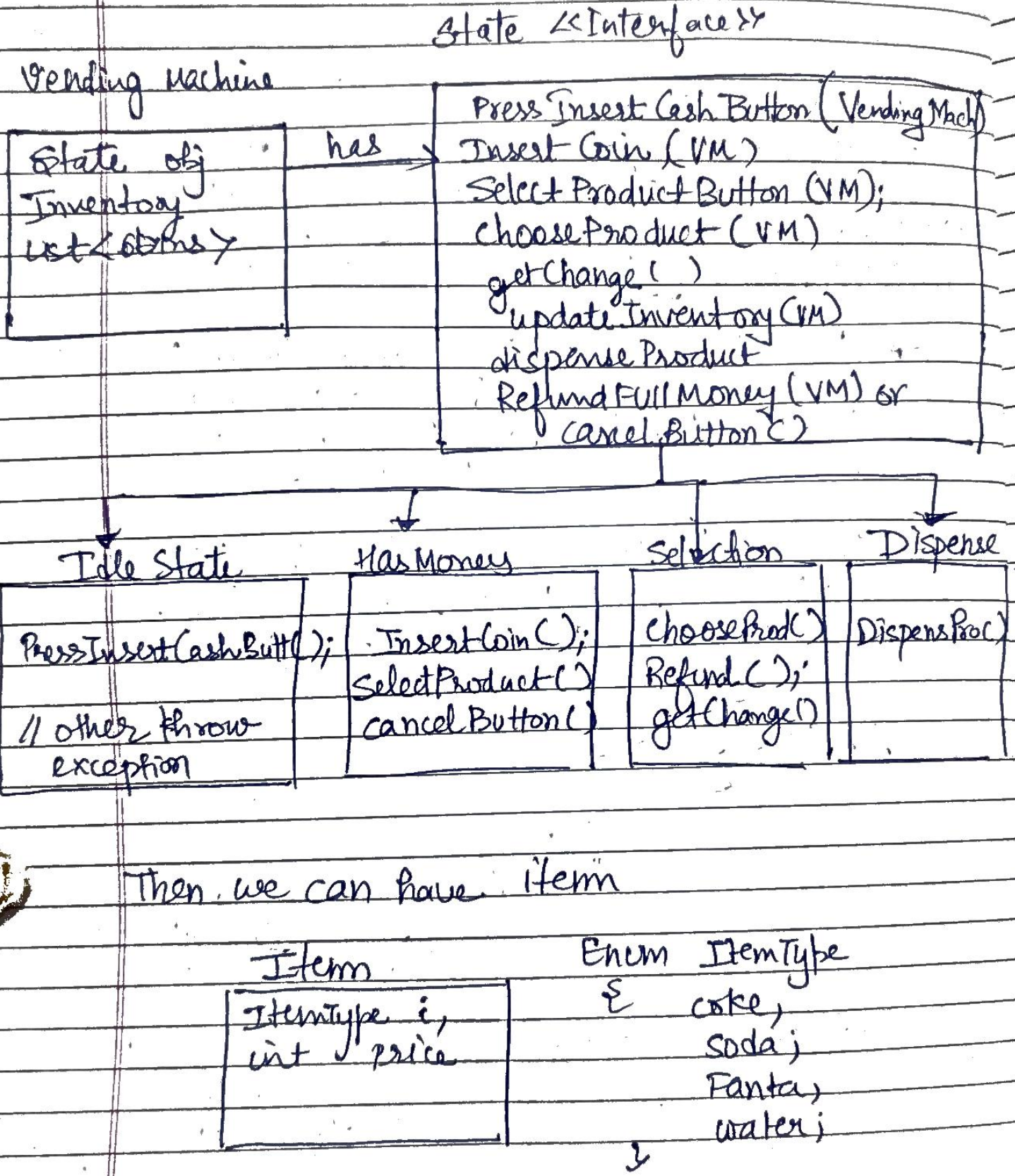


→ It define all operations that product can performs

State 1, state 2 ... → Concrete classes implement these operations

Page No.
Date / /

UML of VM



Then we can have item

Item
ItemType i;
int price

Enum ItemType
{ coke,
soda,
Fanta,
water;
}

Page No.
Date / /

Now these items are placed in a shelf

ItemShelf

```
Item i;
int code;
Bool isSoldout;
```

has → Item

Now the Inventory can have this Itemself

Inventory

```
ItemShelf[] items;
```

vending has Machine

has

Coins

```
{
  ,
}
```