Design Vending Machine

22 June 2024 21:54

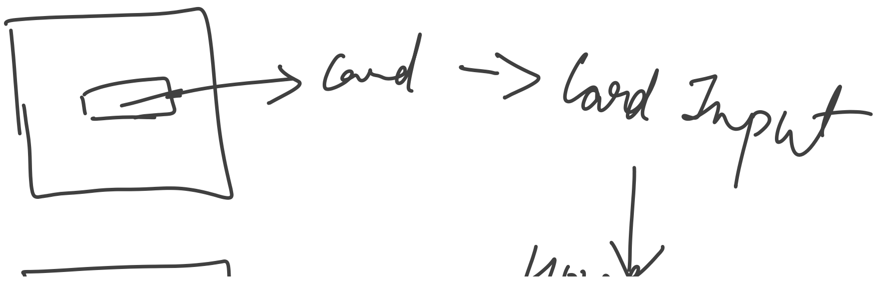
101 - 60₹

102 - Coke

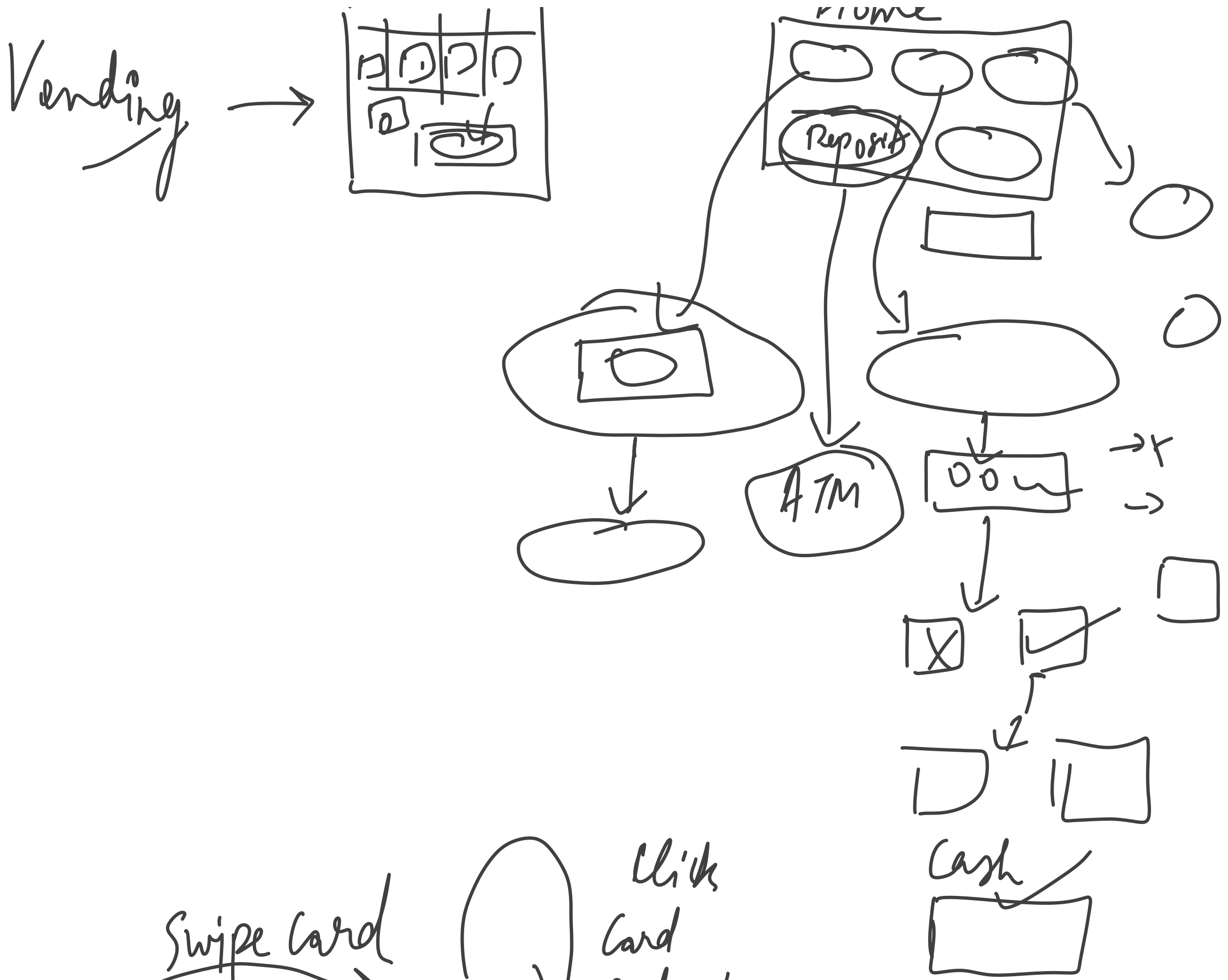
Requirements:

- Should be able to click on insert button cash.
- Able to Insert Cash.
- Should be able to do product selection.
- Able to Select some product – 123
- Checker - Check if the amount is more than the product price
- I click on cancel, entire amount needs to be refunded.
- It will dispense the product
- Dispense the change in the cash dispenser

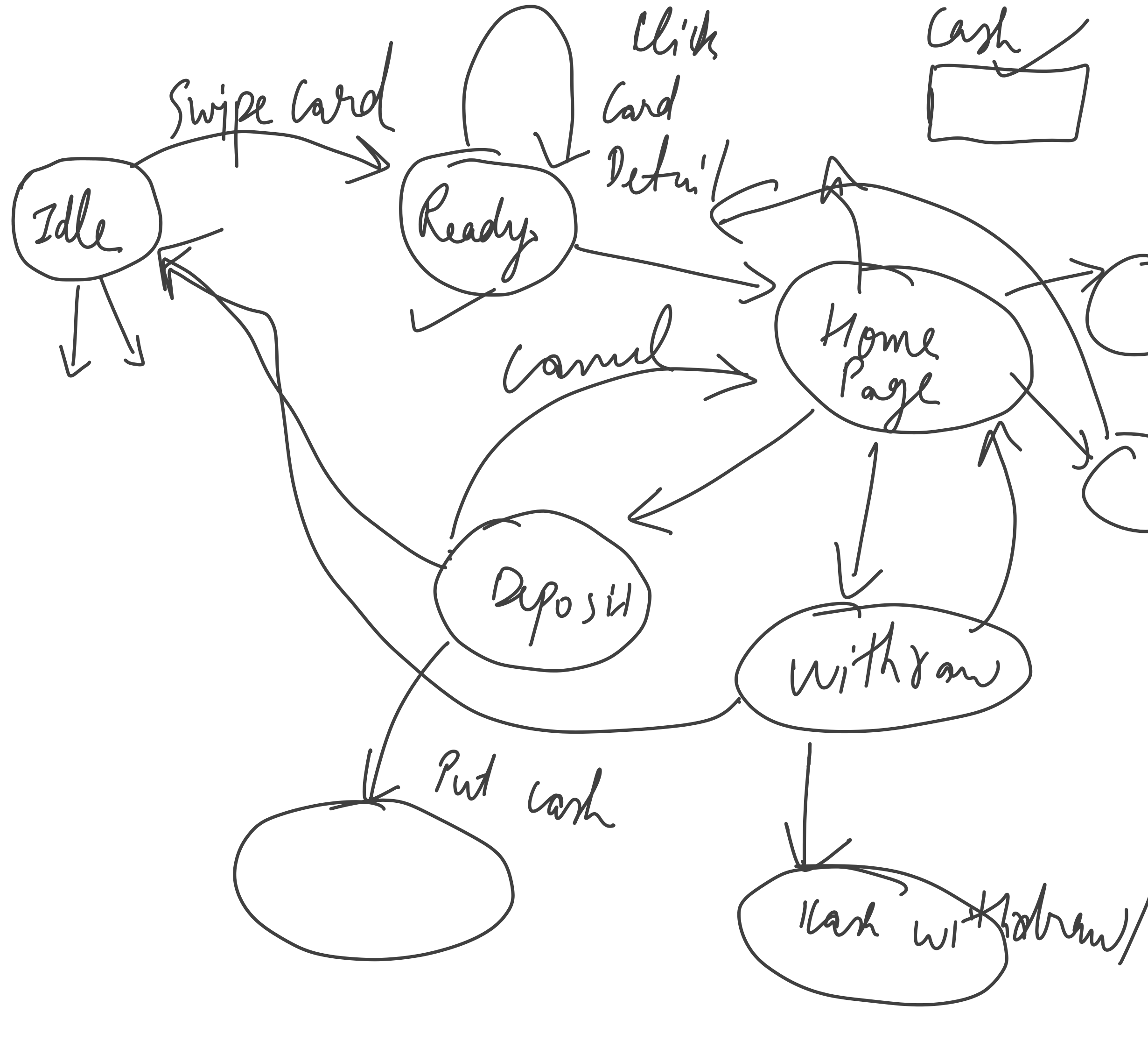
ATM ✓



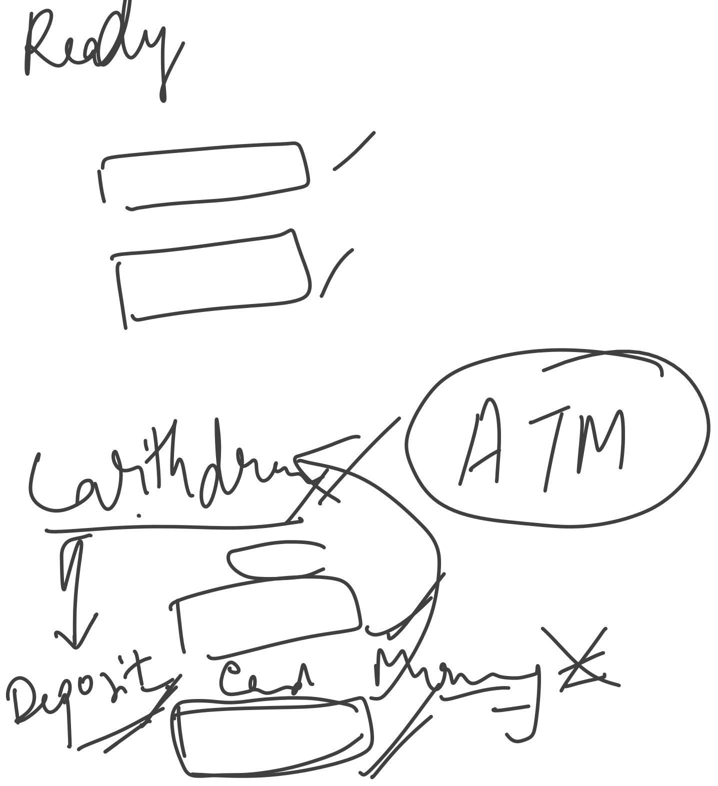
Vending

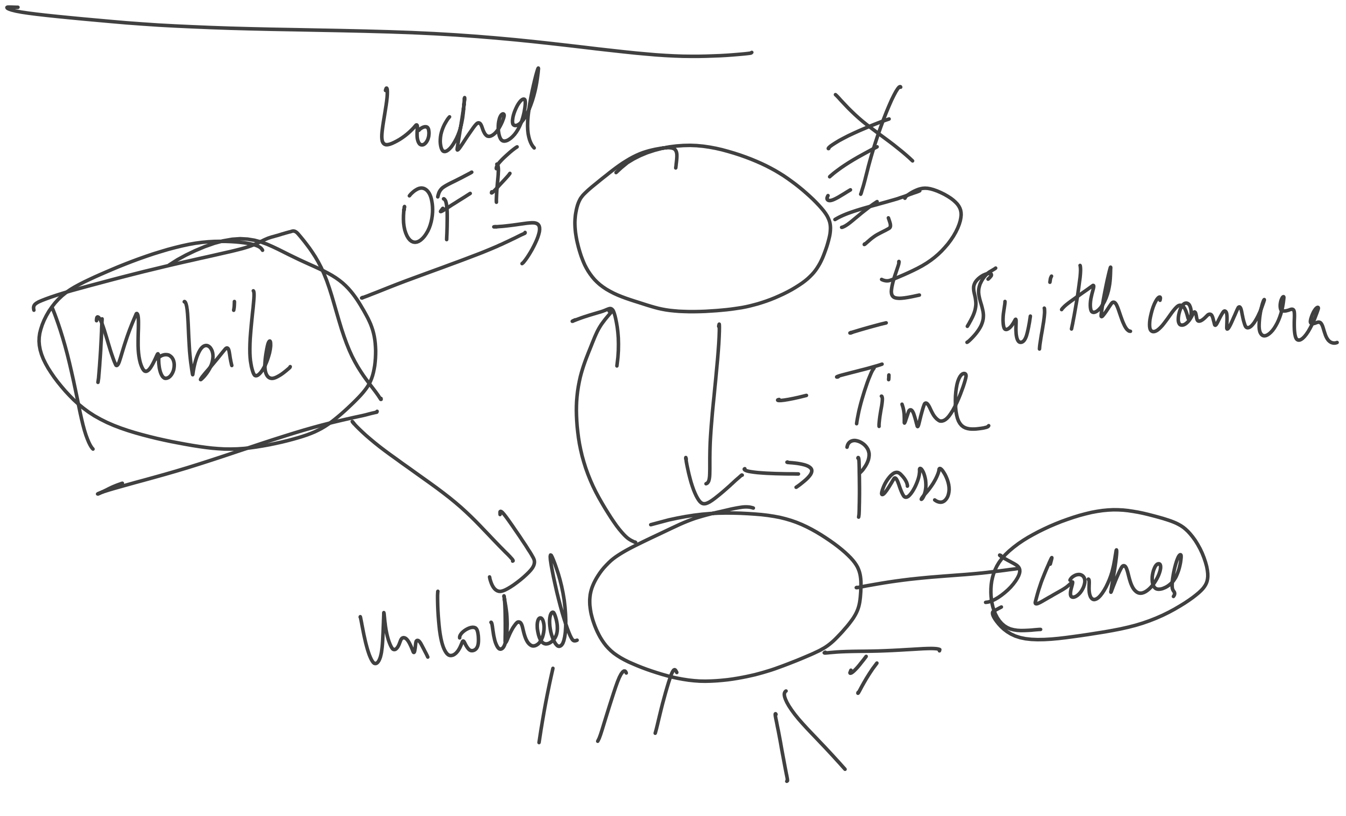


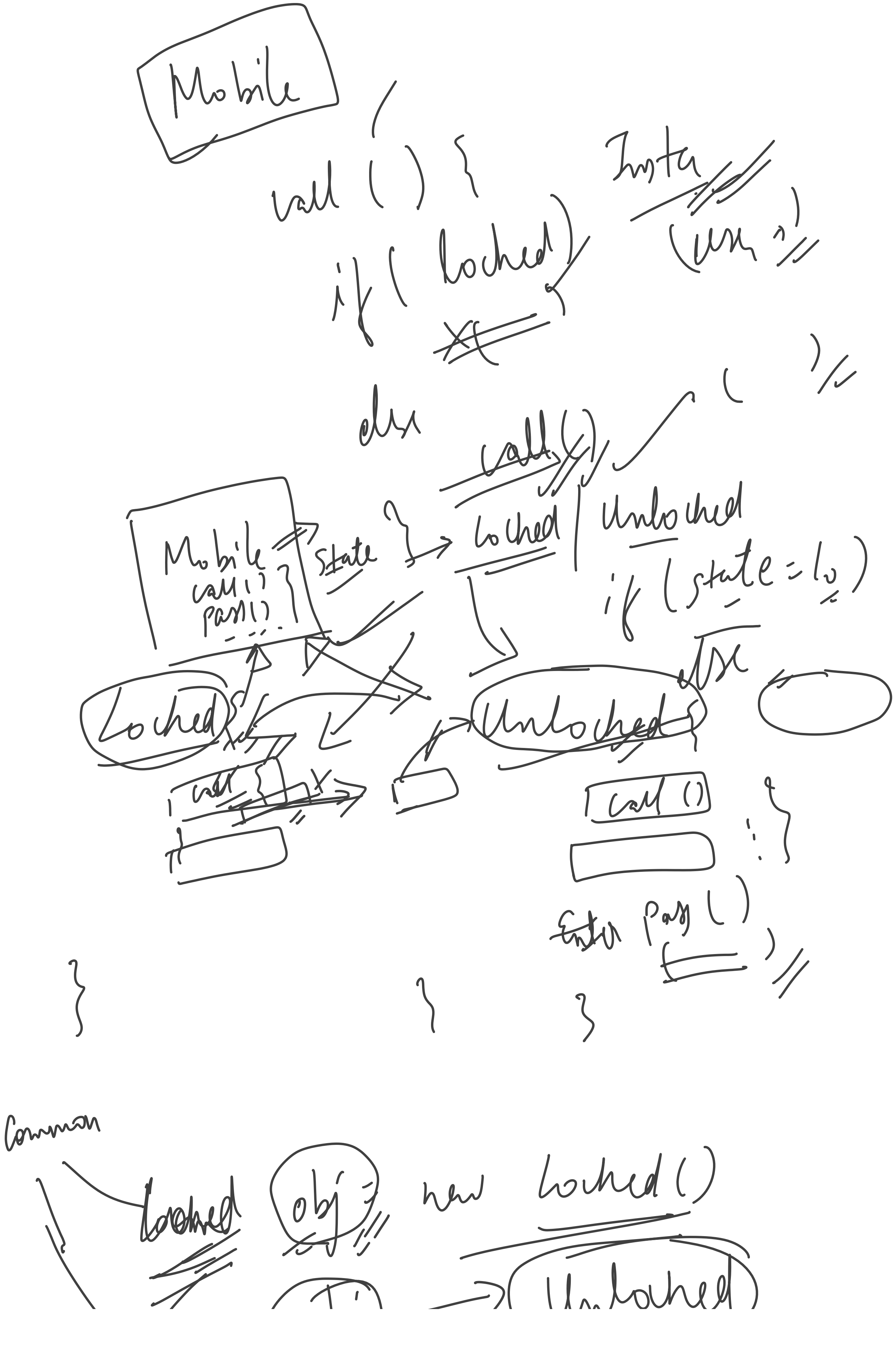
Home
Reposit
ATM
Cash



Idle
Swipe card
Ready
Click
Card
Detail
Home
Page
cancel
Deposit
Withraw
Put cash
Cash withdrawl

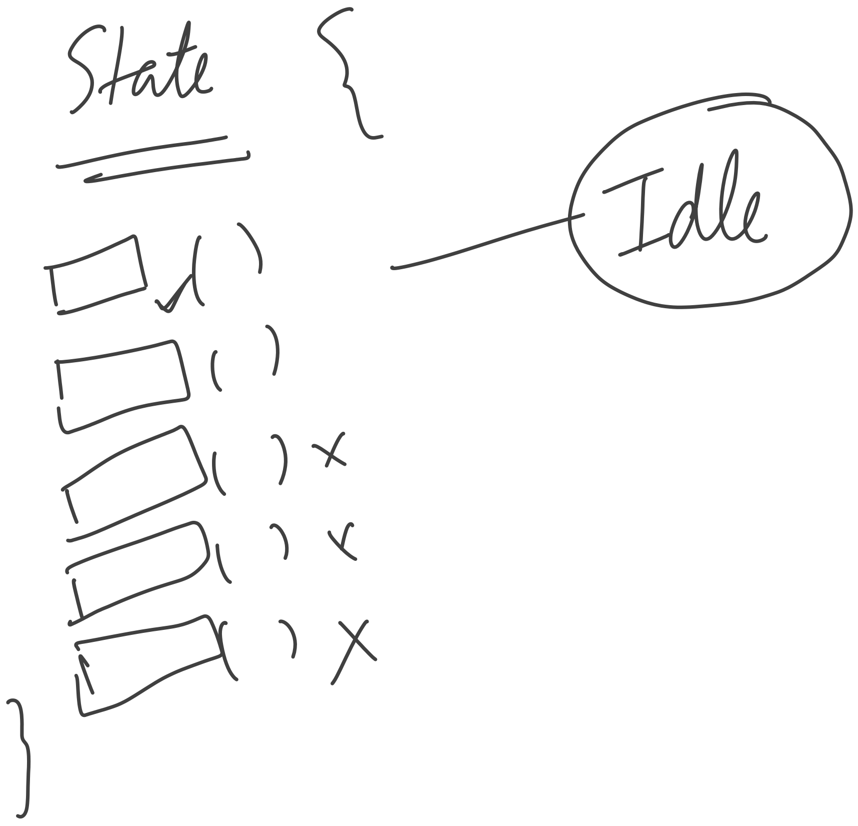

Ready
withdraw
ATM
Deposit
Card
Money


Locked
OFF
Mobile
Switch camera
Time
Pass
Unlocked
Camera



Mobile
call ( ) {
if (locked)
else
call()
Insta
(user ?
Mobile
call()
pass()
state
Locked | Unlocked
if (state = lo)
else
Locked
Unlocked
call
call ()
Enter pass ( )
Common
Locked obj = new Locked()
Unlocked

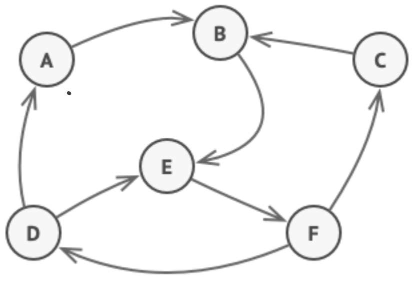
Locked obj [illegible] [illegible]



State Design Pattern

23 June 2024 12:29

State is a behavioral design pattern that lets an object alter its behavior when its internal state changes. It appears as if the object changed its class.



Finite-State Machine.

The main idea is that, at any given moment, there's a *finite* number of *states* which a program can be in. Within any unique state, the program behaves differently, and the program can be switched from one state to another instantaneously. However, depending on a current state, the program may or may not switch to certain other states. These switching rules, called *transitions*, are also finite and predetermined.

Real-World Analogy

The buttons and switches in your smartphone behave differently depending on the current state of the device:

- When the phone is unlocked, pressing buttons leads to executing various functions.
- When the phone is locked, pressing any button leads to the unlock screen.
- When the phone's charge is low, pressing any button shows the charging screen.

State machines are usually implemented with lots of conditional statements (if or switch) that select the appropriate behavior depending on the current state of the object. Usually, this "state" is just a set of values of the object's fields. Even if you've never heard about finite-state machines before, you've probably implemented a state at least once. Does the following code structure ring a bell?

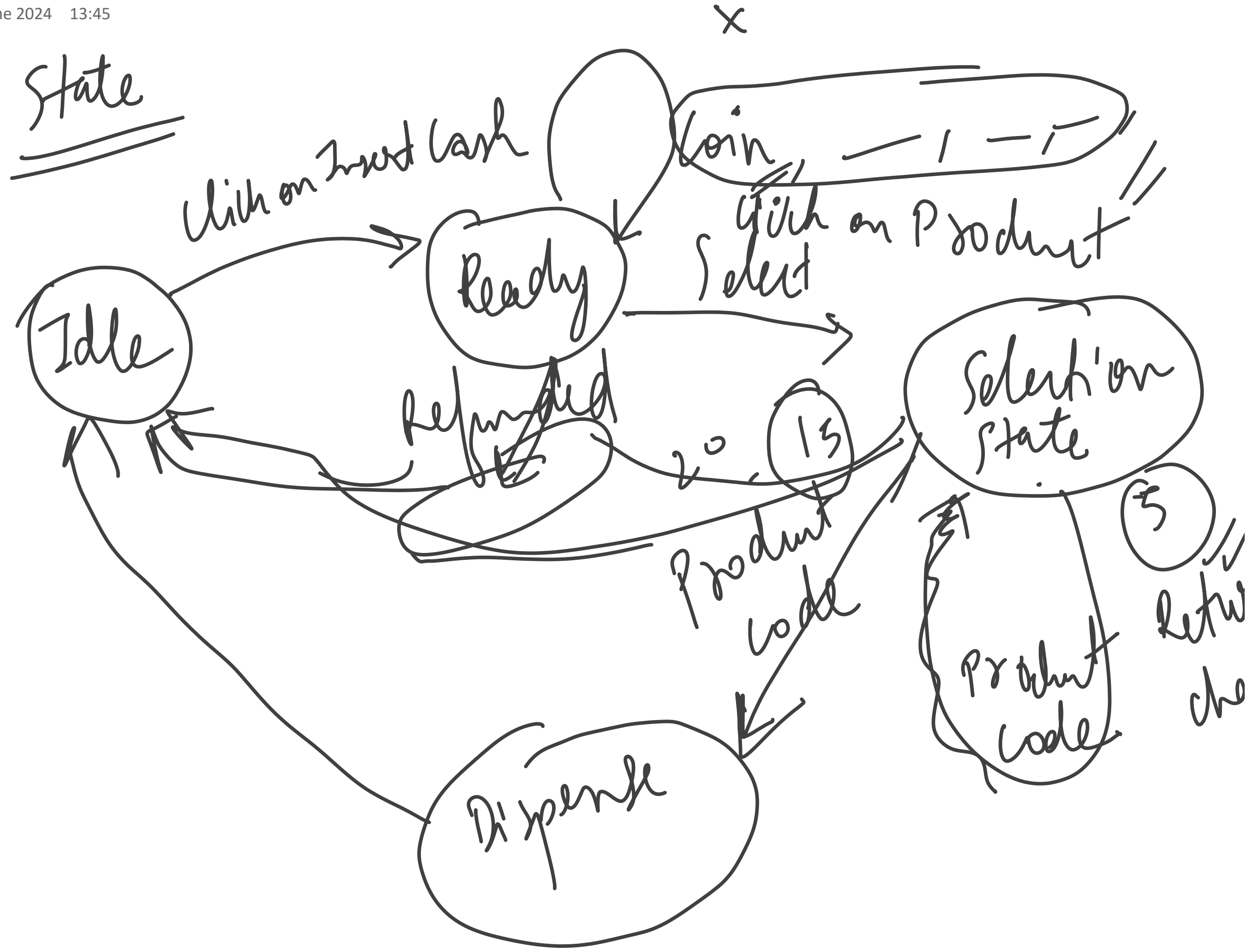
```
class Document is
    field state: string
    // ...
    method publish() is
        switch (state)
            "draft":
                state = "moderation"
                break
            "moderation":
                if (currentUser.role == "admin")
                    state = "published"
                break
            "published":
                // Do nothing.
                break
    // ...
```

The biggest weakness of a state machine based on conditionals reveals itself once we start adding more and more states and state-dependent behaviors to the Document class. Most methods will contain monstrous conditionals that pick the proper behavior of a method according to the current state. Code like this is very difficult to maintain because any change to the transition logic may require changing state conditionals in every method.

The problem tends to get bigger as a project evolves. It’s quite difficult to predict all possible states and transitions at the design stage. Hence, a lean state machine built with a limited set of conditionals can grow into a bloated mess over time.

Step by Step solution:

23 June 2024 13:45



Req

- There should be ability to add coins
- Choose Product
- Cancel Transaction
- Dispense Product

Classes

- Product — Name / Type

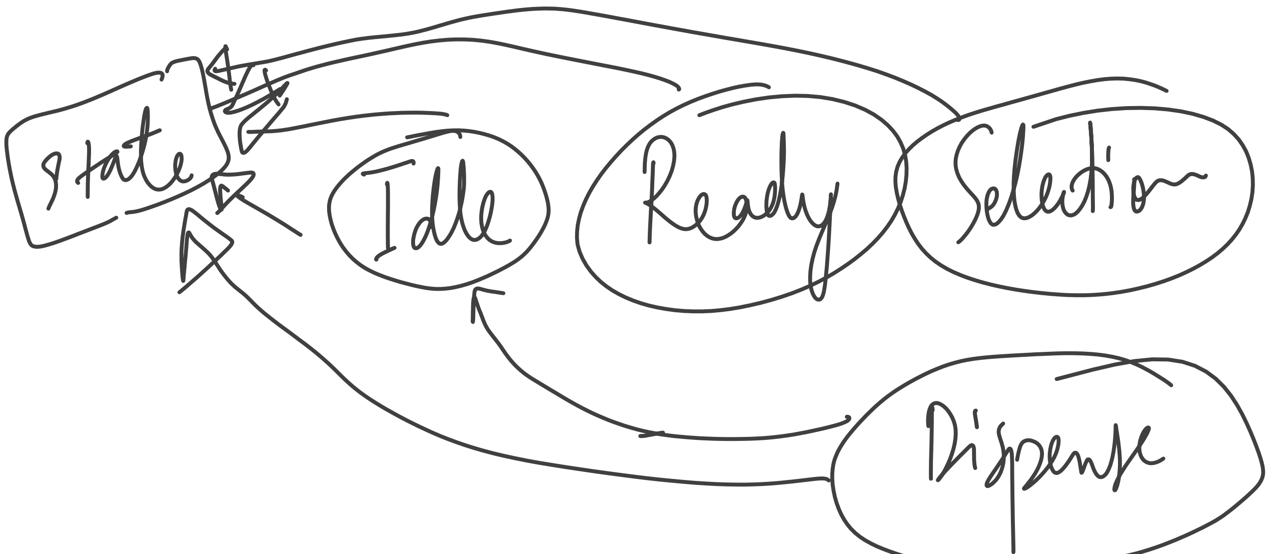
= → Price

- ItemShelf – code
  → isavail

- Inventory → list<ItemShelf>

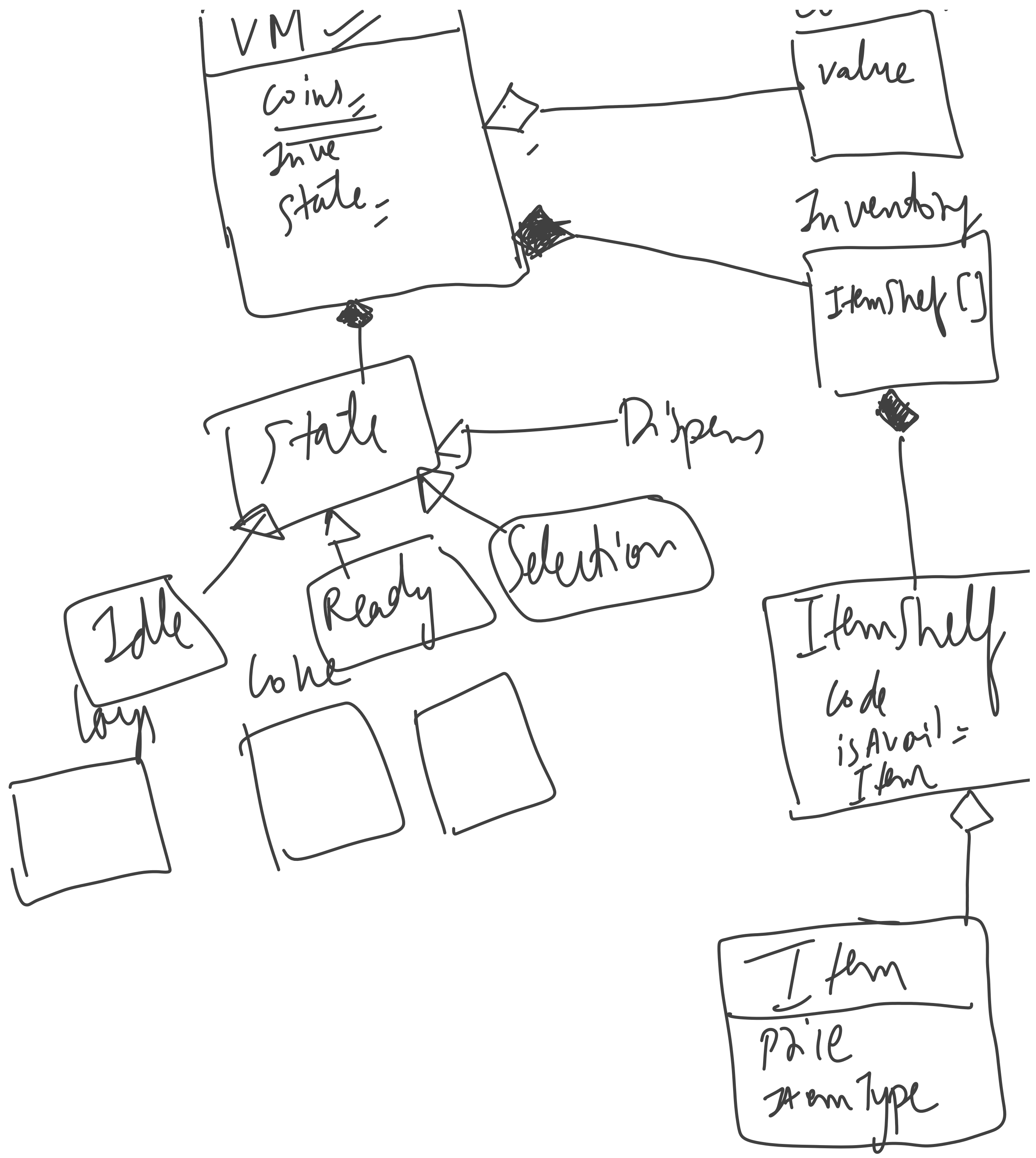
- Vending Machine
  ↳ list<Coin> coins
  → Inventory
  → State

~~25~~
20 → 15 → $\textcircled{5}$



Class Diagram

Payment
Coin


VM
coins
Inve
state
value
Inventory
ItemShelf []
State
Dispens
Idle
Ready
Selection
Coin
ItemShelf
code
isAvail
Item
Item
price
ItemType

CodeLink

23 June 2024 16:57

Only check this link after you are done with your own implementation:
https://github.com/amankumarkeshu/AlgoLLD/tree/main/src/VendingMachineDesign/