Vending Machine

Scenario

The vending machine system allows users to purchase products conveniently by selecting an item, inserting coins, and receiving the desired product. The system ensures smooth operation by checking product availability, validating the inserted amount, and dispensing both the product and any necessary change.

1. A customer approaches the vending machine and views the list of available products along with their prices.
2. The customer selects a product using the **selection panel**.
3. The machine verifies if the selected product is in stock.
    - If the product is unavailable, the system notifies the customer, prompting them to choose another item.
    - If the product is available, the machine displays its price.
4. The customer inserts coins into the **coin collector**.
5. The machine counts the inserted coins and checks if the total matches the product price.
    - If the inserted amount is exact, the machine proceeds to dispense the product.
    - If excess coins are inserted, the machine calculates and returns the appropriate change.
    - If the amount is insufficient, the system prompts the customer to insert additional coins.
6. Once the payment is verified, the vending machine signals the **product dispenser** to release the selected item.
7. If applicable, the **coin dispenser** returns any extra coins as change.
8. The transaction is completed when the customer collects the product and any returned change.

UML Diagrams with plantUML codes

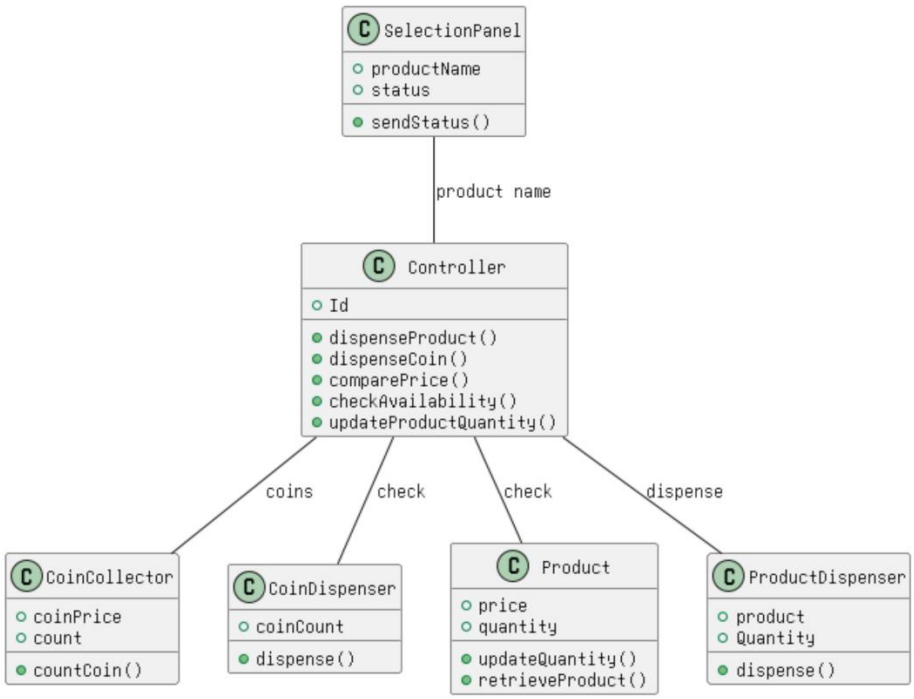
1) Class Diagram

PlantUML code:

```
@startuml
class SelectionPanel {
  + productName
  + status
  + sendStatus()
}

class Controller {
  + Id
  + dispenseProduct()
  + dispenseCoin()
  + comparePrice()
  + checkAvailability()
  + updateProductQuantity()
}

class CoinCollector {
  + coinPrice
```

```
  + count
  + countCoin()
}

class CoinDispenser {
  + coinCount
  + dispense()
}

class Product {
  + price
  + quantity
  + updateQuantity()
  + retrieveProduct()
}

class ProductDispenser {
  + product
  + Quantity
  + dispense()
}

SelectionPanel -- Controller : "product name"
Controller -- CoinCollector : "coins"
Controller -- CoinDispenser : "check"
Controller -- Product : "check"
Controller -- ProductDispenser : "dispense"

@enduml
```

Diagram:

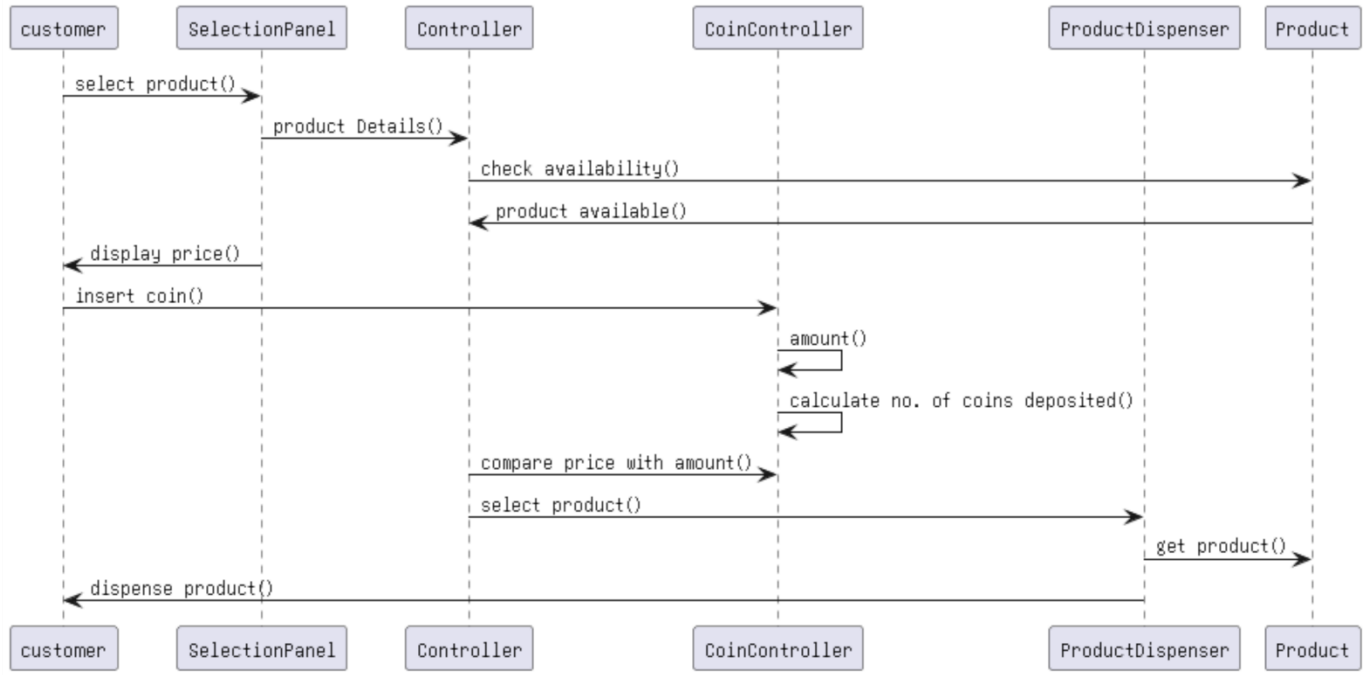


2) Sequence Diagram

PlantUML code:

```
@startuml
participant customer
participant SelectionPanel
participant Controller
participant CoinController
participant ProductDispenser
participant Product

customer -> SelectionPanel : select product()
SelectionPanel -> Controller : product Details()
Controller -> Product : check availability()
Product -> Controller : product available()
SelectionPanel -> customer : display price()
customer -> CoinController : insert coin()
CoinController -> CoinController : amount()
CoinController -> CoinController : calculate no. of coins deposited()
Controller -> CoinController : compare price with amount()
Controller -> ProductDispenser : select product()
ProductDispenser -> Product : get product()
ProductDispenser -> customer : dispense product()

@enduml
```

Diagram:

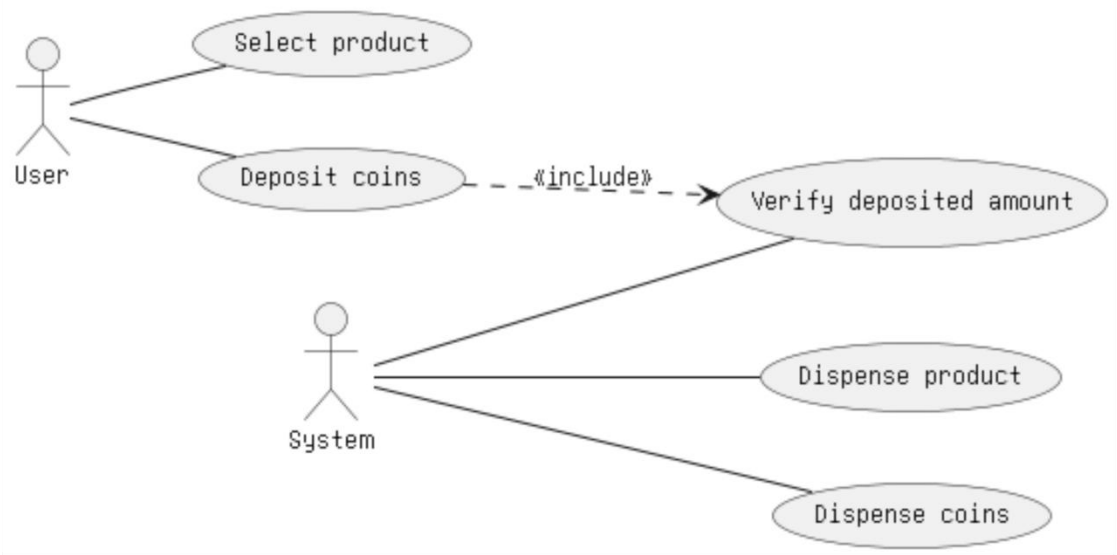


3) Use Case Diagram

PlantUML code:

```
@startuml
left to right direction

actor User
actor System

User -- (Select product)
User -- (Deposit coins)
(Deposit coins) ..> (Verify deposited amount) : <<include>>
System -- (Verify deposited amount)
System -- (Dispense product)
System -- (Dispense coins)

@enduml
```

Diagram:



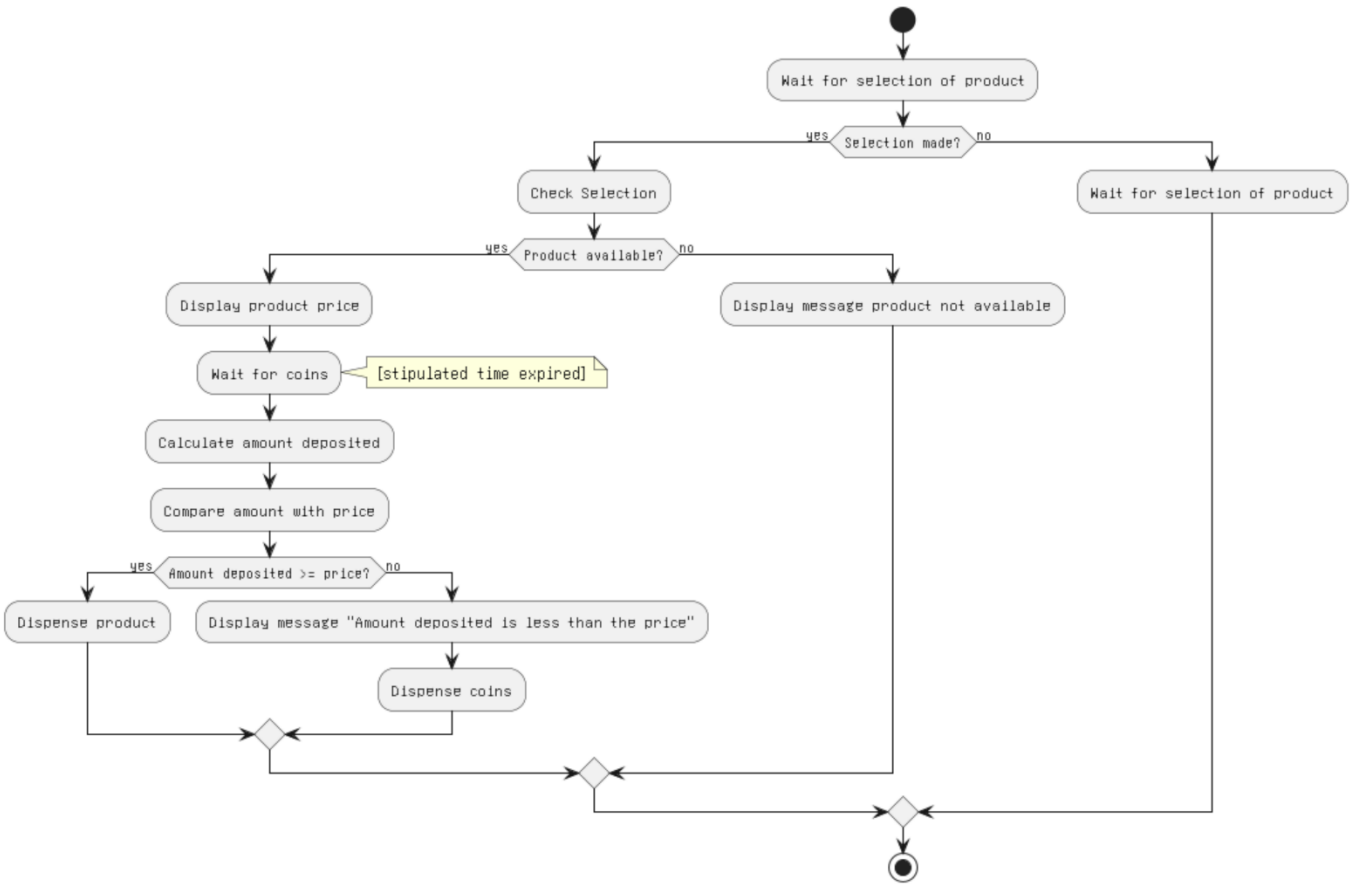
4) Activity Diagram

PlantUML code:

```
@startuml
start
:Wait for selection of product;

if (Selection made?) then (yes)
   :Check Selection;
   if (Product available?) then (yes)
      :Display product price;
      :Wait for coins;
```

```
    note right: [stipulated time expired]
    :Calculate amount deposited;
    :Compare amount with price;

    if (Amount deposited >= price?) then (yes)
      :Dispense product;
    else (no)
      :Display message "Amount deposited is less than the price";
      :Dispense coins;
    endif

  else (no)
    :Display message product not available;
  endif
else (no)
  :Wait for selection of product;
endif

stop
@enduml
```

Diagram:

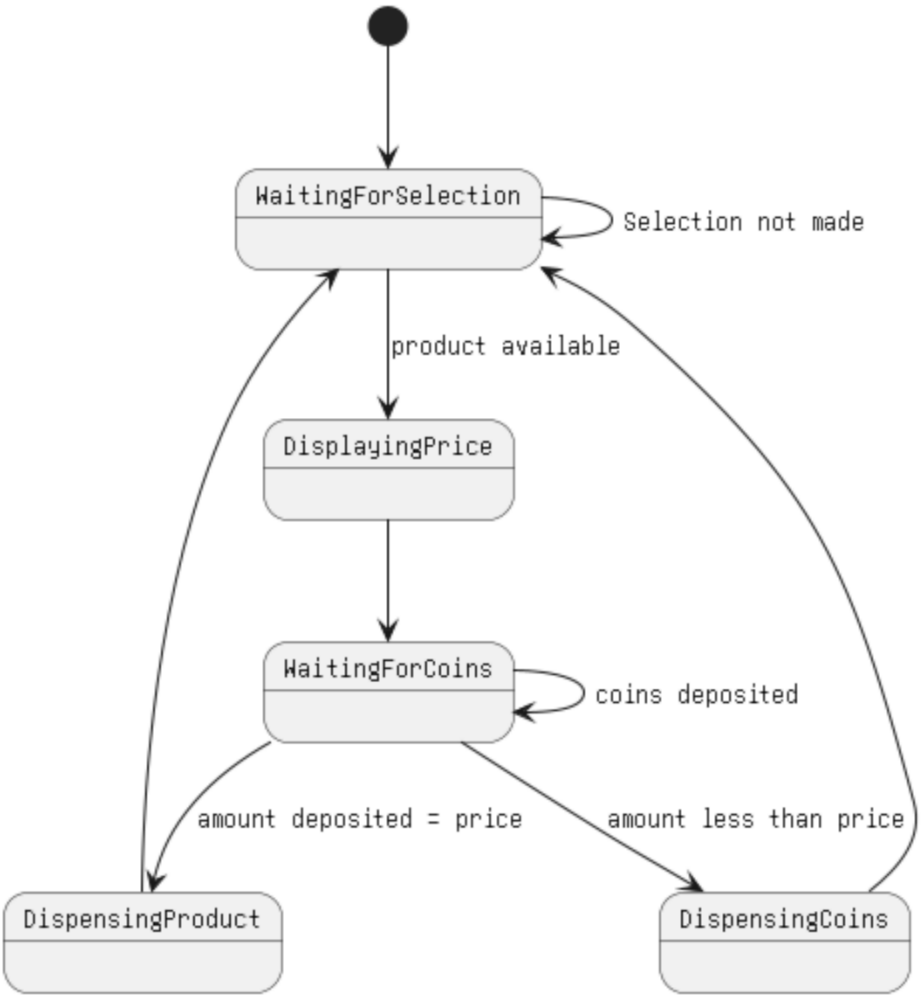


5) State Diagram

PlantUML code:

```
@startuml
[*] --> WaitingForSelection

WaitingForSelection --> WaitingForSelection : Selection not made
WaitingForSelection --> DisplayingPrice : product available

DisplayingPrice --> WaitingForCoins

WaitingForCoins --> DispensingProduct : amount deposited = price
WaitingForCoins --> DispensingCoins : amount less than price
WaitingForCoins --> WaitingForCoins : coins deposited

DispensingProduct --> WaitingForSelection
DispensingCoins --> WaitingForSelection
@enduml
```

Diagram:



6) Collaboration Diagram

PlantUML code:

```
@startuml
object Actor
object selectorPanel
object controller
object Product
object coinCollector
object product_dispenser

Actor --> selectorPanel : 1: select product()
selectorPanel --> controller : 2: product Details()
controller --> Product : 3: check availability of product()
Product --> controller : 4: product available()
controller --> selectorPanel : 5: display price()

Actor --> coinCollector : 6: insert coin()
coinCollector --> coinCollector : 7: calculate no of coins deposited()
coinCollector --> controller : 8: amount()

controller --> controller : 9: compare price with amount()

controller --> product_dispenser : 10: select product()
product_dispenser --> Product : 11: get product()
product_dispenser --> Actor : 12: dispense product()

@enduml
```

Diagram: