MICP Week 7 Homework

Design a vending machine

#TEBOW IT

T - Talk

Who is going to use it? Anyone who wants to buy items
What is it? general vending machine
Where will it be used? Anywhere needed
When will it be used? Anytime needed
How will it be used? As described in the use case list
Why will it be used? To buy items

Expected behavior?

- select item and get price
- accept bills/coins
- dispense items purchased and return change
- refund when cancelling the request

Edge cases:

- sold out
- not fully paid
- not enough changes

E-Examples

Code	Behavior
vendingM = VendingMachine (item, money)	- creating object
vendingM.handlesale()	- processing sale

B-Brute Force

Item

State

- name
- price
- available

Behavior

- getName
- getPrice
- getAvailibility

Money

State

- type
- balance
- change
- bought

Behavior

- getType
- getBalance
- getChange
- getIfSold

Vending Machine

State

- Item
- Money

Behavior

- handle sale

O-Optimize

Item

State

- name
- price
- availabie
- count

Behavior

- get name
- get price
- get availibility
- get count

Item Request

State

- name
- price paid

Behavior

- get name
- get price paid

Money

State

- type
- balance
- change

Behavior

- get type
- get balance
- get change

Transaction

state

- bought

Behavior

- check cancellations

Vending Machine

State

- Item
- Item Req
- Money
- Transaction

Behavior

- handle sale - exceptions, issue change, sell item

I-Implement

```
class Item:
    def __init__(self,name, price, available, count):
        self.name = name
        self.price = price
        self.available = available
        self.count = count
    def getItemName(self):
        return name
    def getItemPrice(self):
        return price
    def getItemAvailability(self):
        return available
    def getCount(self):
        return count
```

—x—

```
class ItemRequest:
    def __init__ (self,name,price_paid):
        self.name = name
        self.price_paid = price_paid.
    def getItemName(self):
        return name
    def getPrice(self):
        return price_paid
```

—x—

```
class Money:
    def __init__(self, type, balance, change):
            self.type = type
            self.balance = balance
            self.change = change
    def getType(self):
            return type
    def getBalance(self):
            return balance
    def getChange(self):
            return change
```

—x—

```
class Transaction:
    def __init__(self, bought):
            self.bought = bought
    def getBought(self):
            return bought
```

—x—

```
class vendingMachine:
    def __init__(item, itemReq, money, trans):
            self.Item = item
            self.ItemRequest = itemReq
```

```
        self.Money = money
        self. Transaction = trans
def issueChange(self):
        requiredchange = self. ItemRequest.getPricePaid() -
                                self. Item. getItemPrice()
        self.Money.balance - = requiredchange
        return requiredChange

def  sellItem(self):
        self.Item . count - =1
        if self.Item.count is 0:
                self.Item.Available = False
        return self.ItemRequest.name

def handleSale(self):
        if not self. Item. getItemAvailibility():
            success = False
            print( self. ItemRequest. getPrice()) #refund
            return success
        if not ((self. ItemRequest .getPrice() - self. Item.
                    getItemPrice ()) < = self.Money.balance):
            success = False
            print(self.ItemRequest.getPrice())
            return success
        if not self.Transaction.bought:
            success = False
            print(self.ItemRequest.getPrice())
            return success
    self.issueChange()
    self.sellItem()
    success = True
    return success
```

— X —

Test