Simulate **supermarket self-service checkout** using the Python 3 programming language.

**Class Design**

***Insert your list/table of possible product properties here…***

## Product Properties (All)

|  |  |  |  |
| --- | --- | --- | --- |
| **productItemName** | **productItemCode** | **productItemSize** | **productItemDescription** |
| **productItemCost** | **productItemColor** | **productItemCompany** |  |
| **productItemWeight** | **productItemType** | **productItemCategory** |  |

***Insert your list/table of key product properties here…***

## Product Properties (Key)

|  |  |  |  |
| --- | --- | --- | --- |
| **productItemCode** | **productItemName** | **productItemCost** | **productItemWeight** |

## Product Class Diagram

+productItemName: String

+productItemCost: Number

+productItemWeight: Number

+productItemCode: Number

+\_\_str\_\_(in self:InstanceObject,out String)

+\_\_init\_\_(in self:InstanceObject,in productItemCode:Number,

in productItemName:String,in productItemWeight:Number,

in productItemCost:Number)

## CheckoutRegister Class Diagram

***Complete the class diagram of your final CheckoutRegister class here…***

+priceArray: Number[]

+itemsPurchased: Product[]

+finalList: Product[]

+price: Number

+totalCost: Number

+\_\_init\_\_(in self:InstanceObject)

+save\_product\_list (in self:InstanceObject,

in finalProductList:Product[])

+accept\_payment(in self:InstanceObject,in amountEntered:Number)

+scan\_item(in self:InstanceObject,in code:Number)

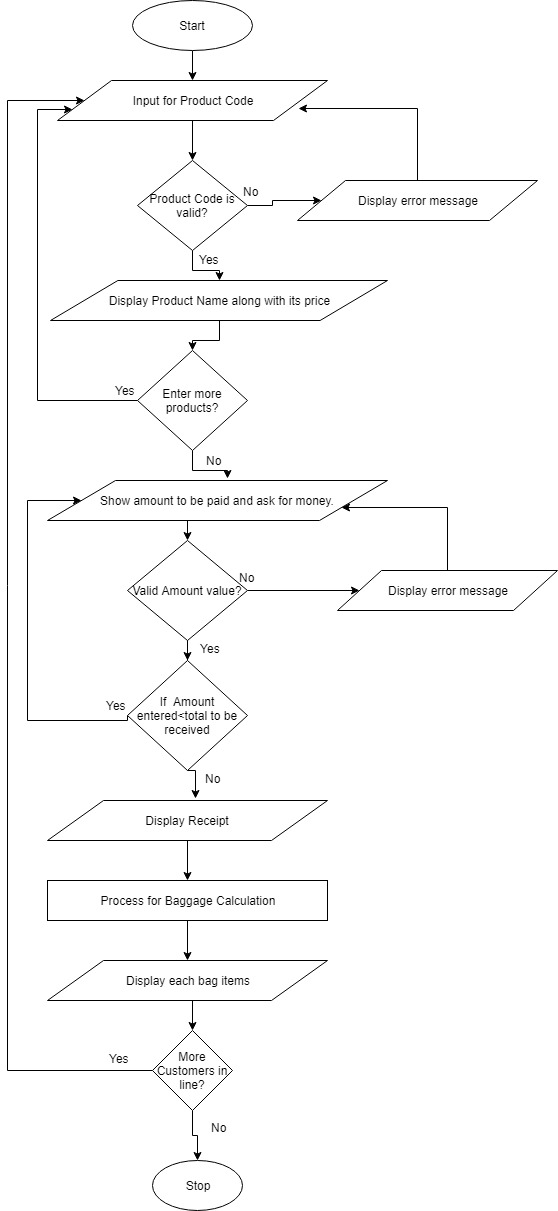
+print\_receipt(in self:InstanceObject)

+payBill(in self:InstanceObject,in finalPrice:Number)

+get\_value (in self,in prompt:String)

+bag\_products(in items\_list: item[],

in self:Object)

****

**# Function to: Validate amount inserted by customer**

**def** get\_value**(**prompt**):**

**# Initialize amount variable with 0**

amt **=** float**(**0.0**)**

**# Looping infinitely until user does not enter correct amount**

**while** **True:**

**try:**

**# Storing amount in amount variable after entered by customer**

amt **=** float**(**input**(**prompt**))**

**# If amount entered by customer is negative then continue the loop**

**if** amt **<** 0.0**:**

**print(**"Don't give negative money!"**)**

**continue**

**# Break the loop if amount is non-negative**

**break**

**# Handle Error for wrong amount enetered**

**except** ValueError**:**

**print(**Please enter a valid amount value. '**)**

**# Return amount entered**

**return** amount

**# Function to: Check the bag products in each bag**

**def** bag\_products**(**items\_list**):**

**# Bagged Item Array and Non Bagged Item Array and a variable to store MAX allowed weight in each bag**

baggedItems = []

nonBaggedItems = []

BAG\_WEIGHT\_MAXIMUM = 5.0

**# Iterating for each item in items\_list**

**for** item **in** items\_list**:**

**# Add item in nonBaggedItems Array if weight of item is greater than maximum allowed weight and remove it from items\_list**

**if** item.productItemWeight > BAG\_WEIGHT\_MAXIMUM:

items\_list.remove(item)

nonBaggedItems.append(item)

**# Creating array for present bag contents and present bag total weight**

presentBagContents = []

presentBagWeight = 0.0

**# Iterate items\_list till the size of list is greater than 0**

**while len(items\_list) > 0:**

**# Pick the first item in list and store it in a temporary variable temp\_item and remove it from items\_list**

temp\_item = items\_list[0]

items\_list.remove(temp\_item)

**# Check if sum of present bag weight and recently added item weight is less than maximum bag weight or not**

**if presentBagWeight + temp\_item.productItemWeight <= BAG\_WEIGHT\_MAXIMUM:**

**# If it is less than add temp\_item in current bag content list and add its weight in total bag weight**

presentBagContents.append(temp\_item)

presentBagWeight += temp\_item.productItemWeight

**# If items\_list is empty add the present bag in baggedItems**

**if (len(items\_list) == 0):**

**baggedItems.append(presentBagContents)**

**# if weight of temp item cannot be accommodated in presentBagContents then, append the present bag in baggedItems**

**else:**

baggedItems.append(presentBagContents)

**# Re-initialize the current\_bg\_contents list and current\_bag\_weight to 0 for another bag**

presentBagContents = []

presentBagWeight = 0.0

**#Show the data of each bag**

**for** index**,** bag **in** enumerate**(**baggedItems**):**

output **=** ' Your Bag ' **+** str**(**index **+** 1**)** **+** ' consists of: '

**# Display item name for all items in baggedItems array**

**for** product **in** bag**:**

output **+=** product**.**productItemName**+** '\t'

**print(**output**,** '\n'**)**

**# Display item name for each nonBaggedItems in nonBaggedItems array**

**for item in nonBaggedItems:**

**output += item.productItemName + '\t'**

**print(output,'\n')**