BRAC UNIVERSITY

Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Classwork No:	**06**
Topic:	**OOP (Class variable and Class method)**
Number of tasks:	**4**

Task 1

We know that Nike is opening their official outlets in Bangladesh. So let's construct a NikeBangladesh class so that they can keep track of their inventory and sales here,

Hint:
productSold()/restockProducts(): takes in a dictionary with product name and quantity, and updates the instance and class variables accordingly

Driver Code	Output

print("xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx")
NikeBangladesh.status()
dhaka = NikeBangladesh("Dhaka Banani")
chittagong = NikeBangladesh("Chittagong GEC")
print("xxxxxxxxxxxxxx2xxxxxxxxxxxxxxxx")
dhaka.details()
print("xxxxxxxxxxxxxx3xxxxxxxxxxxxxxxx")
chittagong.details()
print("xxxxxxxxxxxxxx4xxxxxxxxxxxxxxxx")
dhaka.restockProducts(
{"Air Jordan":1200,"Cortez":200,"Zoom Kobe":200})
chittagong.restockProducts(
{"Air Jordan":1000,"Cortez":250,"Zoom Kobe":100})
print("xxxxxxxxxxxxxx5xxxxxxxxxxxxxxxx")
NikeBangladesh.status()
print("xxxxxxxxxxxxxx6xxxxxxxxxxxxxxxx")
dhaka.productSold({"Air Jordan":760,"Cortez":90})
chittagong.productSold({"Air Jordan":520,"Zoom
Kobe":70})
print("xxxxxxxxxxxxxx7xxxxxxxxxxxxxxxx")
NikeBangladesh.status()
	xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx
	Nike Bangladesh Status:
	Branches Opened: []
	Currently Stocked
	{'Air Jordan': 0, 'Cortez': 0, 'Zoom Kobe': 0}
	Sold: 0
	xxxxxxxxxxxxxx2xxxxxxxxxxxxxxxx
	Nike Dhaka Banani outlet:
	Products Currently Stocked:
	{'Air Jordan': 0, 'Cortez': 0, 'Zoom Kobe': 0}
	Sold: 0
	xxxxxxxxxxxxxx3xxxxxxxxxxxxxxxx
	Nike Chittagong GEC outlet:
	Products Currently Stocked:
	{'Air Jordan': 0, 'Cortez': 0, 'Zoom Kobe': 0}
	Sold: 0
	xxxxxxxxxxxxxx4xxxxxxxxxxxxxxxx
	xxxxxxxxxxxxxx5xxxxxxxxxxxxxxxx
	Nike Bangladesh Status:
	Branches Opened: ['Dhaka Banani', 'Chittagong GEC']
	Currently Stocked
	{'Air Jordan': 2200, 'Cortez': 450, 'Zoom Kobe': 300}
	Sold: 0
	xxxxxxxxxxxxxx6xxxxxxxxxxxxxxxx
	xxxxxxxxxxxxxx7xxxxxxxxxxxxxxxx
	Nike Bangladesh Status:
	Branches Opened: ['Dhaka Banani', 'Chittagong GEC']
	Currently Stocked
	{'Air Jordan': 920, 'Cortez': 360, 'Zoom Kobe': 230}
	Sold: 1440

Task 2

Implement the Question and Quiz(except attempt() method) classes with necessary properties as described in this collab file.

Notes:

1. Please copy and paste the code given in the above-mentioned link into your own colab file and write the missing code.
2. You DO NOT need to implement the attempt() method in Quiz class as it is already implemented. DO NOT change any code written in the attempt() method.
3. You DO NOT need to use print() function in any of the parts you are instructed to write. You need to implement the following:
    a. Implement the __init__() method of Question class.
    b. Declare the class variables of Quiz class.
    c. Implement __init__(),add_question(), create_from_data()
4. DO NOT change the driver code.
5. Relationship between Question and Quiz class: A Quiz can HAVE multiple Questions.

Task 3

Suppose you are the owner of a restaurant named **Foodiz**. First, you started operating at Mohakhali. After noticing its increasing popularity, you opened 2 more branches of **Foodiz** at Banani and Gulshan. As your business is growing fast, you plan to keep track of sales of every branch and how much they contribute to the total revenue.

To implement your idea, you are planning to design a **'Foodiz'** class with the following features:

1. You need to maintain a class variable named **total_sell** to keep track of the total sells(revenue) of all branches.
2. As you need to know the individual contributions of each branch to the total revenue of **Foodiz,** you also need to maintain a class variable(list) named **branch_info** that will store references of each object(each branch) of **Foodiz** class that will later be used to calculate the contribution of each branch to total sales.
3. While creating a branch, you need to pass its name (branch name) to the constructor. You can create any branch in the following way:
   ```
   mohakhali = Foodiz('Mohakhali')
   ```

4. There will be 2 instance variables in the **Foodiz** class: **branch_name, branch_sell,** where **branch_name** will store the branch name passed to the constructor and **branch_sell** will store the sell information of each branch based on the quantity passed to the **sell_quantity**() method described in point 5. Initially, the default value of **branch_sell** will be 0.

5. To calculate the **branch_sell**, you can maintain an instance method named **sell_quantity(quantity)** method that takes a quantity(integer) as a parameter and calculates the **branch_sell** in the following way:

   branch_sell = quantity * 300

So, while creating a branch(object) of **Foodiz**, you are just passing its branch name and after that, you use the **sell_quantity()** method to calculate the **branch_sell**.

After creating each branch, you will need to show the individual branch information and also the information of all branches altogether. To implement this, you can maintain two methods.

1. You can create an instance method named **branch_information()**, that will print the **branch_name** and **branch_sell** of a specific individual branch.
2. You can create a class method named '**details()**' that will first print the total number of branches(objects) of **Foodiz** class and the total sells(**total_sell**) of **Foodiz**.
   After that, it will print the name of each branch, then the branch sells information, and then the contribution of each branch to the total sell of **Foodiz**.

   You can calculate the contribution of each branch to the total sell using the following formula:

   branch's contribution (in %) = (branch's sell / total sell) * 100

Now design the **Foodiz** class and the **required driver code** to implement the above idea. You will find a demo driver code along with the corresponding output below that will help you to understand the overall scenario a bit more clearly.

Demo Driver Code

```python
Foodiz.details()
print('1----------------------------------')
mohakhali = Foodiz('Mohakhali')
mohakhali.sellQuantity(25)
mohakhali.branchInformation()
print('2----------------------------------')
Foodiz.details()
print('3========================')
banani = Foodiz('Banani')
banani.sellQuantity(15)
banani.branchInformation()
print('4----------------------------------')
Foodiz.details()
print('5========================')
gulshan = Foodiz('Gulshan')
gulshan.sellQuantity(9)
gulshan.branchInformation()
print('6----------------------------------')
Foodiz.details()
```

Given Output

```
Total Number of branch(s): 0
Total Sell: 0 Taka
1---------------------------------------
Branch Name: Mohakhali
Branch Sell: 7500 Taka
2---------------------------------------
Total Number of branch(s): 1
Total Sell: 7500 Taka
########################
Branch Name: Mohakhali
Branch Sell: 7500 Taka
Branch consists of total sell's: 100.0%
3========================
Branch Name: Banani
Branch Sell: 4500 Taka
4---------------------------------------
Total Number of branch(s): 2
Total Sell: 12000 Taka
########################
Branch Name: Mohakhali
Branch Sell: 7500 Taka
Branch consists of total sell's: 62.5%
########################
Branch Name: Banani
Branch Sell: 4500 Taka
Branch consists of total sell's: 37.5%
5========================
Branch Name: Gulshan
Branch Sell: 2700 Taka
6---------------------------------------
Total Number of branch(s): 3
Total Sell: 14700 Taka
########################
Branch Name: Mohakhali
Branch Sell: 7500 Taka
Branch consists of total sell's:
51.02040816326531%
########################
Branch Name: Banani
Branch Sell: 4500 Taka
Branch consists of total sell's:
30.612244897959183%
########################
Branch Name: Gulshan
Branch Sell: 2700 Taka
Branch consists of total sell's:
18.367346938775512%
```

Task-4

```python
class A:
    temp = 4
    def __init__(self):
        self.y = self.temp - 2
        self.sum = self.temp + 1
        A.temp -= 2
        self.methodA(3, 4)
    def methodA(self, m, n):
        x = 0
        self.y = self.y + m + (self.temp)
        A.temp += 1
        x = x + 1 + n
        self.sum = self.sum + x + self.y
        print(x, self.y, self.sum)

class B:
    x = 0
    def __init__(self, b = None):
        self.y, self.temp, self.sum = 5, -5, 2

        if b == None:
            self.y = self.temp + 3
            self.sum = 3 + self.temp + 2
            self.temp -= 2
        else:
            self.sum = b.sum
            B.x = b.x
            b.methodB(2, 3)
    def methodA(self, m, n):
        x = 2
        self.y = self.y + m + (self.temp)
        self.temp += 1
        x = x + 5 + n
        self.sum = self.sum + x + self.y
        print(x, self.y, self.sum)
    def methodB(self, m, n):
        y = 0
        y = y + self.y
        B.x = self.y + 2 + self.temp
        self.methodA(self.x, y)
        self.sum = self.x + y + self.sum
        print(self.x, y, self.sum)
```

```python
a1 = A()
b1 = B()
b2 = B(b1)
b1.methodA(1, 2)
b2.methodB(3, 2)
```