Lab 0

CSC 202 Data Structures

Winter 2024

1. The first task is quite easy. It will show how actually classes look like and how we call different functions in classes by creating objects. Copy paste the following code in PyCharm IDE or IDE of your choice and remove any errors that may come up. I need to know the output of this program. Just find out the final output and paste it as a comment in the end. Go through each line and understand it as you will need this understanding to do the tasks later in this lab.

```
class Flower:
#Common base class for all Flowers
 def __init__(self, petalName, petalNumber, petalPrice):
 self.name = petalName
 self.petals = petalNumber
 self.price = petalPrice
 def setName(self, petalName):
 self.name = petalName
 def setPetals(self, petalNumber):
 self.petals = petalNumber
 def setPrice(self, petalPrice):
 self.price = petalPrice
 def getName(self):
 return self.name
 def getPetals(self):
 return self.petals
 def getPrice(self):
 return self.price
#This would create first object of Flower class
f1 = Flower("Sunflower", 2, 1000)
print ("Flower Details:")
print ("Name: ", f1.getName())
print ("Number of petals:", f1.getPetals())
print ("Price:",f1.getPrice())
print ("\n")
#This would create second object of Flower class
f2 = Flower("Rose", 5, 2000)
f2.setPrice(3333)
f2.setPetals(6)
print ("Flower Details:")
print ("Name: ", f2.getName())
print ("Number of petals:", f2.getPetals())
print ("Price:",f2.getPrice())
```

2. By using the concepts, you have learned in the above code. Write a class called Product.

    - The class should have fields called name, amount, and price, holding the product's name, the number of items of that product in stock, and the regular price of the product.
    - There should be a method get_price that receives the number of items to be bought and returns the cost of buying that many items, where the regular price is charged for orders of less than 10 items, a 10% discount is applied for orders of between 10 and 99 items, and a 20% discount is applied for orders of 100 or more items.
    - Include a method named **make_purchase** in your class that takes the quantity of items to be purchased as an argument and updates the **balance** attribute by deducting the total cost of the items. The regular prices for the products are as follows: shirt ($50), wristwatch ($120), textbook ($80), history book ($50), and memory card ($140).
3. Write a class called Converter. The user will pass a length and a unit when declaring an object from the class—for example, c = Converter(9, 'inches').
    - The possible units are inches, feet, yards, miles, kilometers, meters, centimeters, and millimeters.
    - For each of these units there should be a method that returns the length converted into those units. For example, using the Converter object created above, the user could call c.feet() and should get 0.75 as the result.

Test Cases

Many people tend to focus on writing code as the singular activity of a programmer, but testing is one of the most important tasks that one can perform while programming. Proper testing provides a degree of confidence in your solution. Systematic testing helps you to discover and then debug your code. Writing high quality test cases can greatly simplify the tasks of both finding and fixing bugs and, as such, will save you time during development. However, testing does not always guarantee that your program is correct.

For this part of the lab, you will practice writing some simple test cases to gain experience with the unit test framework. I recommend watching the first 20 minutes or so of the following video if you need more guidance on testing in Python. https://www.youtube.com/watch?v=6tNS--WetLI