CSC 1302 Principles of Computer Science II

Assignment 5: Java Class Variables/Methods

(Due on 11:59 pm, 7/13/2021)

Purpose:

Instance variables belong to the instance of a class, i.e., an object. Every object of that class has its own copy of that instance variable. Changes made to a particular instance variable donot reflect in other instances of that class.

Class variables are known as static variables. There is only one copy of that variable that is shared with all instances of that class. If changes are made to that variable, all other instances will see the change.

Instance methods belong to the object of the class, can only be called after an object of its class is created. Static/class methods can be called without creating an object of class. They are referenced by the class name itself or reference to the Object of that class.

In this assignment, we will practice how to use multiple classes and objects as well as class variables/methods. Once a class is available, it's like we have a new data type. Variables can be defined with a type of this class and can refer to an object of this class. Methods can use this class as parameter type or return type.

Program #1:

1. Write a PaypalAccount class to include both *balance* and *accountID* as the instance variables. Make sure each instance of this account will have a **unique** *accountID.* In other words, different account object should have different *accountID* (hint: class variable).

2. Write a Bank class with main method. In the main method, ask the user to input how many accounts (say *numOfAccount*) to be generated in the bank (assuming less than 1000). Then create an array to hold these *numOfAccount* of Account objects. For each Account object, generate a random balance in the range of 0.0-1000.0.
    a. Assume that your GSU campus ID is *abc-de-fghi*; search the array to see if there is an account with *accountID* as *abc* (the first three digits of your campus ID). If there is not an account with *accountID* as *abc*, then set the *accountID* of the last account in the array as *abc;* transfer all the balance of the first account to the account with *accountID* of *abc*
    b. Set the *balance* of the account with *accountID* of *abc* to be *efghi/100.0* (i.e., your last 5 digits of your campus ID divided by 100.0); and print out the information of this account

c. Find the average account balance of all the accounts in the array and print it out.
d. Find the account with the largest balance, print out its accountID and balance.
e. Find the account with the lowest balance, print out its accountID and balance.

Here is an example of the screenshot when running the program:

Program #2:

Write a class called YourFirstNameLine that represents a line segment between two Points. Your Line objects should have the following methods:

- public YourFirstNameLine (Point p1, Point P2): construct a new Line that contains the given two Points
- public Point getStartPoint(): return this Line's first endpoint
- public Point getEndPoint(): return this Line's second endpoint
- public String toString(): return a String representation of this Line, such as "[(2,3), (4,7)]"
- public double getSlope(): return the slope of this Line. The slope of a line between points (x1, y1) and (x2, y2) is equal to (y2-y1)/(x2-x1). If x2 equals x1, the denominator is zero and the slope is undefined, so you may return a zero in this case
- public void draw (Graphics g): draw a line in a DrawingPanel.

Write a client class called YourFirstNameLineTest with main method to test your Line class. In the main method, you need create a line object with the last four numbers of your Panther ID. For example, if your pantherid ends with xxxxx1234, then create and draw a line of between point (1,2) and point (3,4). (Hint: you will need the Point.java, DrawingPanel.java, YourFirstNameLine.java and YourFirstNameLineTest.java)

Criteria:

1. Upload all of the .java and the .class files to the CSc1302 dropbox on http://icollege.gsu.edu.
2. Your assignment will be graded based on the following criteria: (a) Are your programs runnable without errors? (b) Do your programs complete the tasks with specified outputs? (c) Do you follow the specified rules to define your methods and programs? (d) Do you

provide necessary comments include the programmer information, date, title of the program and brief description of the program.

3. Make sure that both the .java and .class files are named and uploaded to icollege correctly. If any special package is used in the program, be sure to upload the package too. Should you use any other subdirectory (whatsoever) your program would not be graded, and you will receive a **0 (zero)**.
4. No copying allowed. If it is found that students copy from each other, all of these programs will get **0**.