North South University
Department of Electrical and Computer Engineering

CSE 215: Assignment 03 (Problem Set)
Course Instructor: Dr. Mohammad Rezwanul Huq

Problem Solving on Classes and Objects

Submission Instructions:

- Write the solution of each problem named as A03_P01.java and so on.
- Zip all your .java files. Remember, only the java files. Name the zip file according to your student id such as 1911234042.zip.
- Upload the zip file in the following link. https://goo.gl/m99ftH (Select Assignment 02).
- Solution must be uploaded by **Sunday 11:59:59 PM, 24 March 2019.**
- **Failure to submit the weekly assignment will result into deducting marks from both theory and lab performance.**

Problem Set

1. (*The **Rectangle** class*) Design a class named **Rectangle** to represent a rectangle. The class contains:
   - Two **double** data fields named **width** and **height** that specify the width and height of the rectangle. The default values are **1** for both **width** and **height**.
   - A no-arg constructor that creates a default rectangle.
   - A constructor that creates a rectangle with the specified **width** and **height**.
   - A method named **getArea()** that returns the area of this rectangle.
   - A method named **getPerimeter()** that returns the perimeter.

   Implement the class. Write a test program that creates two **Rectangle** objects—one with width **4** and height **40** and the other with width **3.5** and height **35.9**. Display the width, height, area, and perimeter of each rectangle in this order.

2. (*The **Stock** class*) Design a class named **Stock** that contains:
   - A string data field named **symbol** for the stock's symbol.
   - A string data field named **name** for the stock's name.
   - A **double** data field named **previousClosingPrice** that stores the stock price for the previous day.
   - A **double** data field named **currentPrice** that stores the stock price for the current time.
   - A constructor that creates a stock with the specified symbol and name.
   - A method named **getChangePercent()** that returns the percentage changed from **previousClosingPrice** to **currentPrice**.

   Implement the class. Write a test program that creates a **Stock** object with the stock symbol **ORCL**, the name **Oracle Corporation**, and the previous closing price of **34.5**. Set a new current price to **34.35** and display the price-change percentage.

3. **(Account class)** Design a class named **Account** that contains:

   - A private **int** data field named **id** for the account (default **0**).
   - A private **double** data field named **balance** for the account (default **0.0**).
   - A private **double** data field named **annualInterestRate** that stores the current interest rate (default **0.0**).
   - A private **Calendar** data field named **dateCreated** that stores the date when the account was created. Use Calendar type object.
   - A no-arg constructor that creates a default account.
   - A constructor that creates an account with the specified id, initial balance and annual interest rate. Within the constructor, assign the value of dateCreated using Calendar.getInstance().
   - The accessor and mutator methods for **id**, **balance**, and **annualInterestRate**.
   - The accessor method for **dateCreated**.
   - A method named **getMonthlyInterestRate()** that returns the monthly interest rate. **monthlyInterestRate** is **annualInterestRate / 12**. Note that **annualInterestRate** is a percentage, e.g., like 4.5%. You need to divide it by 100
   - The method **getMonthlyInterestAmount()** is to return monthly interest amount, not the interest rate. Monthly interest amount is **balance * monthlyInterestRate**.
   - A method named **withdraw** that withdraws a specified amount from the account.
   - A method named **deposit** that deposits a specified amount to the account.

   Write a test program that creates an **Account** object with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the **withdraw** method to withdraw $2,500, use the **deposit** method to deposit $3,000, and print the balance, the monthly interest, and the date when this account was created.

4. (*The **Location** class*) Design a class named **Location** for locating a maximal value and its location in a two-dimensional array. The class contains public data fields **row**, **column**, and **maxValue** that store the maximal value and its indices in a two-dimensional array with **row** and **column** as **int** types and **maxValue** as a **double** type.

   Write the following method that returns the location of the largest element in a two-dimensional array:

   public static Location locateLargest(**double**[][] a)

   The return value is an instance of **Location**. Write a test program that prompts the user to enter a two-dimensional array and displays the location of the largest element in the array. Here is a sample run:

```
Enter the number of rows and columns in the array:  3 4 Enter
Enter the array:
23.5 35 2 10 Enter
4.5 3 45 3.5 Enter
35 44 5.5 9.6 Enter
The location of the largest element is 45 at (1, 2)
```

5. (*The **Person**, **Student**, **Employee**, **Faculty**, and **Staff** classes*) Design a class named **Person** and its two subclasses named **Student** and **Employee**. Make **Faculty** and **Staff** subclasses of **Employee**. A person has a name, address, phone number, and email address. A student has a class status (freshman, sophomore, junior, or senior). Define the status as a constant. An employee has an office, salary, and date hired. Use java.util.Calendar to create an object for date hired. A faculty member has office hours and a rank. A staff member has a title. Override the **toString** method in each class to display the class name and the person's name.

   Write a test program that creates a **Person**, **Student**, **Employee**, **Faculty**, and **Staff**, and invokes their **toString()** methods.