0360-212 Winter 2017 Lab Assignment 3

Due@Bb: Feb 13, 11:59pm

Remember to comment your code.

For all the questions below, if there is any ambiguity in the descriptions make your own assumptions and document them in your code as comments.

Question 1:

Define a class called ***"Date"*** with the following suggested methods and instance variables:

- Three integer (or string) instance variables to store the day, the month and the year.
- A constructor with three int type parameters that stores day, month, and year, respectively. You are free to define other constructors when appropriate.
- Methods to set/get each instance variables.
- A method ***"toString"*** with no arguments which returns the date in the form of a string. The format of the string is as follows: name of month + space + date + comma + space + apostrophe + last 2 digits of year.
- A method ***"LessThan"*** which takes in one parameter of class Date. If the date stored in the current object corresponds to a date earlier than that represented by the parameter, the method should return true. Otherwise, the method should return false. You may assume that if the constructor receives a year argument greater than or equal to "50", the argument corresponds to a year in the twentieth century. Otherwise, the argument corresponds to a year in the twenty first century.

Besides designing and implementing the ***"Date"*** class, you also need to write another class such as ***"Date_Test"*** to test the ***"Date"*** class you write. Comment both ***"Date"*** and ***"Date_Test"*** classes. In your ***"Date_Test"*** class, demonstrates how you use various methods you defined in the ***"Date"*** class.

Examples

1. Consider three date objects d1, d2 and d3 corresponding to the dates 26th May '99, 8th June '94 and 15th March '00, respectively.

Date(25, 05, 99) will create a Date object, say, d4 corresponding to the date 25th May, 1999. Method call d4.toString() will return "May 25, '99". Method call d4.lessThan(d1) will return true. Method call d4.lessThan(d2) will return false. Method call d4.lessThan(d3) will return true.

2. Consider three date objects d1, d2 and d3 corresponding to the dates 26th May '99, 8th June '94 and 15th March '00. Date(25, 01, 00) will create a Date object d5 corresponding to the date 25th January, 2000. Method call d5.toString() will return "January 25, '99". Method call d5.lessThan(d1) will return false. Method call d5.lessThan(d2) will return false. Method call d5.lessThan(d3) will return true.

3. Consider three date objects d1, d2 and d3 corresponding to the dates 26th May '99, 8th June '94 and 15th March '00, respectively. Date(26, 05,99) will create a Date object d6 corresponding to the date 26th May, 1999. Method call d6.toString() will return "May 26, '99". Method call d6.lessThan(d1) will return false. Method call d6.lessThan(d2) will return false. Method call d6.lessThan(d3) will return true.

Question 2:

In this question, we are not giving out specifications of a class; you have to come up with your own design based on the following description.

1. A client can create an account by providing his/her SIN no. (a 9 digits number), name, and address. These 3 pieces of information must be provided before an account can be created. An account no. is created by randomly generating an integer between 1000-9999. This integer is then concatenated with the client's SIN no. The resulting concatenation is the account no. The initial balance of the account is zero.
2. A client can make inquiries on his/her account balance.
3. A client can change his name, address and other non-essential account information such as phone no. email address, etc. (this implies that your design of the account class should contain phone no., email address as instance variables.)
4. A client can deposit/withdraw from his/her account. A withdraw cannot be made if not sufficient fund is available in the account.

(Note: your design of the account class should at least be able to perform the above functions. However, you are free to incorporate any design that you think appropriate for a bank account.)

The design of the account class means that you have to decide on what instance variables the account class should have, what methods the class should implement to manipulate the instance variables. Upon finishing writing the account class, you should again write a test class such as "Account_Test" to test each method/functionality of your account class

---

Question 3:

Define a class called Fraction. This class is used to represent a ratio of two integers. Include methods that allow the user to set the numerator and the denominator. Also include a method that returns the value of numerator divided by denominator as a double. Include an additional method that outputs the value of the fraction reduced to lowest terms (e.g., instead of outputting 20/60, the method should output 1/3). This will require finding the greatest common divisor for the numerator and denominator, then dividing both by that number. You can get an idea of the algorithm from http://www.java67.com/2012/08/java-program-to-find-gcd-of-two-numbers.html

---

Question 4:

Write a method *squareOfAsterisks* that displays a solid square (the same number of rows and columns) of asterisks whose side is specified in integer parameter side. For example, if side is 4, the method should display

```
****
****
****
****
```

Incorporate this method into an Java Program that reads an integer value for side from the user and outputs the asterisks with the *squareOfAsterisks* method.

Question 5:

Write a method *minimum3* that returns the smallest of three floating point numbers. Use the *Math.min* method to implement minimum3. Incorporate the method into a Java program that reads three values from the user, determines the smallest value and displays the result.

Question 6:

Write an application that plays "guess the number" as follows: Your program chooses the number to be guessed by selecting a random integer in the range 1 to 1000. The application displays the prompt Guess a number between 1 and 1000. The player inputs a first guess. If the player's guess is incorrect, your program should display Too high. Try again. or Too low. Try again. to help the player "zero in" on the correct answer. The program should prompt the user for the next guess. When the user enters the correct answer, display Congratulations. You guessed the number!, and allow the user to choose whether to play again.

Question 7:

Create a class called Complex for performing arithmetic with complex numbers. Complex numbers have the form

realPart + imaginaryPart * i

where i is $\sqrt{-1}$

Write a program to test your class.

Use floating-point variables to represent the data of the class.

Provide a constructor that enables an object of this class to be initialized when it's declared.

Provide a no-argument constructor with default values in case no initial values are provided.

Provide methods that perform the following operations:

a) Add two Complex numbers: The real parts are added together and the imaginary parts are added together.

b) Subtract two Complex numbers: The real part of the right operand is subtracted from the real part of the left operand, and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.

c) Print Complex numbers in the form (realPart, imageinaryPart).