Beni-Suef University

Faculty of Computers and Artificial Intelligence

First Term 2021/2022

CS 241 - Object Oriented Programming

SHEET #4

Due date: 13-12-2021

Problem 1:

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.

Draw the UML diagram for the class then 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.

Problem 2:

Create a class called MyDate that includes:

- Three instance variables: a month (int), a day (int) and a year (int).
- The class should have a constructor that initializes the three instance variables and ensures that the values provided are correct.
- Provide a set and a get method for each instance variable.
- Provide a method displayDate that displays the month, day and year separated by forward slashes(/).

Problem 3:

Design a class named Fan to represent a fan. The class contains:

- Three constants named SLOW, MEDIUM, and FAST with the values 1, 2, and 3 to denote the fan speed. A private int data field named speed that specifies the speed of the fan (the default is SLOW).

- A private boolean data field named on that specifies whether the fan is on (the default is false).
- A private double data field named radius that specifies the radius of the fan (the default is 5).
- A string data field named color that specifies the color of the fan (the default is blue).
- The accessor and mutator methods for all four data fields.
- A no-arg constructor that creates a default fan.
- A method named toString() that returns a string description for the fan. If the fan is on, the method returns the fan speed, color, and radius in one combined string. If the fan is not on, the method returns the fan color and radius along with the string “fan is off” in one combined string.

Draw the UML diagram for the class then implement the class. Write a test program that creates two Fan objects. Assign maximum speed, radius 10, color yellow, and turn it on to the first object. Assign medium speed, radius 5, color blue, and turn it off to the second object. Display the objects by invoking their toString method.

Problem 4:

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).
- A private double data field named annualInterestRate that stores the current interest rate (default 0). Assume that all accounts **have the same interest rate**.
- A private Date data field named dateCreated that stores the date when the account was created.
- A no-arg constructor that creates a default account.
- A constructor that creates an account with the specified id and initial balance.
- The accessor (get method) and mutator (set method) for id, balance, and annualInterestRate.
- The accessor method for dateCreated.
- A method named getMonthlyInterestRate() that returns the monthly interest rate.
- A method named getMonthlyInterest() that returns the monthly interest.
- A method named withdraw that withdraws a specified amount from the account. Ensure that the debit amount does not exceed the Account’s balance. If it does, the balance should be left unchanged and the method should print a message indicating “Debit amount exceeded account balance”.
- A method named deposit that deposits a specified amount to the account.
- Draw the UML diagram for the class then implement the class.

Note: The method getMonthlyInterest() is to return monthly interest, not the interest rate.

Monthly interest = balance * monthlyInterestRate.

MonthlyInterestRate = annualInterestRate / 12.

Note: annualInterestRate is a percentage, for example 4.5%. You need to divide it by 100.

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.

Problem 5:

Write a class named Complex for dealing with the complex numbers and contains:

- A private double data field named realPart for the real part.
- A private double data field named imaginaryPart for the imaginary part.
- A default constructor that initializes the number to (0, 0).
- A constructor with parameters **Complex(double realPart,double imaginaryPart)**.
- The accessor and mutator for realPart and imaginaryPart.
- A method **public Complex addComplex(Complex newNumber):** This method will compute the sum of the current complex number and the passed complex number. The method returns a new Complex number which is the sum of the two.
- A method **public Complex subComplex(Complex newNumber):** This method will compute the difference between the current complex number and the passed complex number. The method returns a new Complex number which is the difference of the two.
- A method **public void printComplex()** to print the complex number.
- Write a separate class ComplexTest with a main() method and test the Complex class methods.

Problem 6:

Create the Employee class that includes:

- Three instance variables (firstName, lastName and monthlySalary).
- The class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0.
- Write a test class named EmployeeTest that creates an **array of 10 objects** of class Employee and displays each Employee’s **yearly** salary. Then, give each Employee a 10% raise and display each Employee’s yearly salary again.

Problem 7:

Write the Time class which is designed in the following class diagram:

Time
-hour:int -minute:int -second:int
+Time(hour:int,minute:int,second:int) +getHour():int +getMinute():int +getSecond():int +setHour(hour:int):void +setMinute(minute:int):void +setSecond(second:int):void +setTime(hour:int,minute:int,second:int):void +toString():String +nextSecond():Time +previousSecond():Time

Note:

- You must perform all the input validations, for example the instance variable hour is in range 0-23.
- The method toString() will print the time of this instance in the format; "hh:mm:ss" with leading zero.
- The methods nextSecond() and previousSecond() will increment or decrement this instance by one second and return **this instance**.
- Write a test program that creates a Time object called t and invoke the method nextSecond() using the statement t.nextSecond().nextSecond().

Best Wishes
DR. Noha Yehia