Computer Science Division
Department of Mathematics
Faculty of Science

Level 3- Fall Semester
Course Code: Comp 301
Date: October 14, 2024

Sheet 2

Objective: upon successful completion of this sheet, students should be able to create user-defined classes, practice methods, deal with inner classes, and differentiate between access and non-access modifiers.

1. Design the "`Complex`" class to represent complex numbers and their operations. The class contains "`rel`" and "`img`" as data members and the following methods:
   - suitable constructor(s);
   - setters and getters;
   - `isReal()` and `isImaginary()` to check if the complex number is purely real or purely imaginary, respectively;
   - `equals(Complex c)` to compare two complex numbers;
   - `addTo(Complex c)` to add two complex numbers and return the result as a new complex number $(a + bi) + (c + di) = (a + c) + (b + d)i$;
   - `multiplyTo(Complex c)` to multiply two complex numbers and return the result as a new complex number $(a + bi)(c + di) = (ac - db) + (ad + bc)i$;
   - `conjugate()` to return the conjugate of the complex number (conjugate $((a + bi)) = (a - bi)$);
   - `magnitude()` to return the magnitude of the complex number; and
   - `display()` to print the complex number in the following form: `real`+`imag` i.

2. Design the "`Invoice`" class to represent an invoice of a sold item. The class should contain the following data members and methods.
   Data members: ID, description, a quantity of the item, and a price per item. In addition, the `invoices_count` data member to keep track of the number of invoices.
   All data members should only be accessible in the class itself.

Methods: suitable constructor(s), setters and getters, `subtotal()` to calculate the subtotal invoice amount according to the type and number of items sold, `total()` to calculate the total amount after applying the `5%` tax, and `display()` to show all invoice information.
Before setting the values for both quantity and price, you need to first check whether the entered value is positive or not. In case it is not, then it is set to zero.

Design also a static inner class named "`Cashier`" in the `invoice` class that contains the name of the cashier as data member and the `getInvoices()` method to get the number of invoices issued by that cashier.

3. Design the "`Author`" class, which contains the following data members and methods.
   Data members: name and email.
   Methods: suitable constructor(s), setters and getters, and display information about the Author class.

   Design also the "`Book`" class, which contains the following data members and methods.
   Data members: ISBN, name, an author object, publisher, and price.
   Methods: suitable constructor(s), setters and getters, display information about the Book class by overriding the `toString()` method, and check if two books are the same, in terms of their ISBN by overriding the `equals()` method.