CS 112 Programing II

Lab Session 1: Objects and Classes (Part 1)

Objectives

The objective of today's lab session is to get you started using the concept of *object-oriented programming* to address larger scale software development. By the end of this lab you should develop better understanding of how to:

- define classes and create objects using different constructors.
- use UML graphical notations to describe classes and objects.
- access an object's data and methods using the object member access operator.

Exercise 1 *(30 minutes)*

Given the code below (attached as softcopy with this sheet), do the following:

1. Does this code compile and run?
2. If so, what is the output?
3. Provide comments explaining each line of code and summarize the general aim of the code.
4. Use UML graphical notations to describe the Rectangle class as well as the two created objects, i.e., rec1 and rec2.

******************** RectangleDemo.java ********************

```java
public class RectangleDemo {

    public static void main(String[] s)
    {
      Rectangle rec1 = new Rectangle();
      Rectangle rec2 = new Rectangle(3.5, 35.9);

      System.out.println("The perimeter of the first rectangle is: " +
              rec1.getPerimeter() + "\n");
      System.out.println("The perimeter of the second rectangle is: " +
              rec2.getPerimeter() + "\n");
      System.out.println("The area of the first rectangle is: " +
              rec1.getArea() + "\n");
      System.out.println("The area of the second rectangle is: " +
              rec2.getArea() + "\n");
    }
}
 class Rectangle {

    private double width;
    private double height;

    public Rectangle() {
        width = 1.0;
        height = 1.0;
    }

    public Rectangle(double w, double h) {
        width = w;
        height = h;
    }

    public double getArea() {
        return width * height;
    }

    public double getPerimeter() {
        return width * 2 + height * 2;
    }
}
```

Please note that for simplicity the rectangle class as well as the main class (which is used for testing the rectangle class) are both written in the same file. This is, however, not the case in real-world applications.

Exercise 2 *(45 minutes)*

The Account Class

Design a class named **Account** that contains:

a. A private **int** data field named **id** for the account (default 0)
b. A private **double** data field named **balance** for the account (default 0)
c. A private **Date** data field named **dateCreated** that stores the date when the account was created
d. A **no-arg constructor** that creates a **default account**
e. A **constructor** that creates an account with the **specified id and initial balance**
f. The **accessor and mutator methods** for **id and balance**.
g. The **accessor** method for **dateCreated**
h. A method named **withdraw()** that withdraws a specified amount from the account
i. A method named **deposit()** that deposits a specified amount to the account

- *Draw the UML diagram for the Account class*
- *Implement the Account class*
- *Use the test program given bellow that creates an Account object with an account ID of 1122, and a balance of $20,000. Use the withdraw() method to withdraw $2,500, use the deposit() method to deposit $3,000, and print the balance, and the date when this account was created.*

Hint: The Date class is included in the util package.

Use two separate files to implement the Account class and the Test class.

```
public class Test {
  public static void main (String[] args) {
    Account account = new Account(1122, 20000);

    account.withdraw(2500);
    account.deposit(3000);
    System.out.println("Balance is " + account.getBalance());
    System.out.println("This account was created at " +
      account.getDateCreated());
  }
}


Class Account {

  // Implement the class here

}
```