CS 112 Programing II
Lab Session 3: Object-Oriented Thinking

Objectives

So far you have learnt how to **define classes, create objects and access class members**. During your lectures you have also been introduced to the power of using object-oriented paradigm in programming. This lab session will provide you with more practice on the role of **class user** rather than **class developer** which is equally important. In fact, being familiar with how to use a class will help you design/implement better classes in the future. The objective of this lab session is also to **familiarize you with class associations**. By the end of this session, you should be able to:

- identify class associations and understand how they are represented in Java.
- use already defined classes which involve associations with other classes.

Exercise 1

Open **Lab_3_1** project (attached as softcopy with this sheet) and take a few minutes reading and understanding the files contained in the project. Do not worry if some parts of the code do not make sense at this stage. You will eventually understand them. After reading the files, please do the following:

1. The project has some issues and thus does not compile. Error messages will guide you to resolve the issues so read them by hovering the mouse over the problematic lines and try to work out why the project does not compile and resolve the issues accordingly. Files should be kept within their packages (i.e. do not move them). *(15 minutes)*

Once the code compiles, run the project and the output should look similar to the one given below:

```
Course Name: Programming II

Course Code: CS112

Course Professor: Nora

List of students:

 - Name: Sara, ID: 123145, level: 4, GPA: 4.5

 - Name: Sahar, ID: 123412, level: 2, GPA: 3.5

 - Name: Amnah, ID: 133345, level: 4, GPA: 3.75

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

2. Look closely at the data fields in the **Course** class. What type of relationships are represented using the **Professor** object and **Student** array of objects in the **Course** class? Identify the *aggregat**ing** class* and the *aggregat**ed** classes*. Visualize the relationships using UML *(note that we often omit class data fields and methods when we model relationships in UML)*. *(15 minutes)*

Please note that for simplicity we have omitted in this project the courses a student takes as well as the courses a professor teaches.

3. Note how the **Course** object is created within the testing class (testing.CourseTest.java). Also note how the course data fields, particularly the **Student** array of objects and the **Professor** object are set and then printed out.
The for loop used in the main to iterate over all students in the array **studentList** has a slightly different syntax to the for loop you have used before. It is often called for-each and it provides an easy way to traverse between array elements. Read more at: https://www.geeksforgeeks.org/for-each-loop-in-java/. *(15 minutes)*

4. Now it is your turn to create your own **Course** object, set its data fields and print out its details. *(30 minutes)*

5. **Reflection:** Look through your use of the class in the testing class and reflect on some possible improvements that can be done to the classes implementation which may facilitate the use of them. *(do that at home! ☺)*

Homework

Design a class named StackOfStrings that contains:

a. A private array elements to store strings in the stack
b. A private data field size to store the number of strings in the stack
c. A constructor to construct an empty stack with a default capacity of 4
d. A constructor to construct an empty stack with a specified capacity
e. A method empty() that returns true if the stack is empty
f. A method push(String value) that stores value at the top of the stack. This method checks if the stack is full before pushing to it. In case the stack is full, it
   i. copies the contents of the stack to a new temp array
   ii. creates a new elements array with double the current capacity
   iii. copies back temp to elements

   iv. places value at the top of the new stack

g. A method pop() that removes the string at the top of the stack and returns it. In case the stack is empty, it returns "Stack is EMPTY"
h. A method peek() that returns the string at the top of the stack without removing it from the stack. In case the stack is empty, it returns "Stack is EMPTY"
i. A method printStack() as shown below:

```
public void printStack(){
    System.out.print("Stack (top to bottom) : ");
    for (int i=size-1; i>-1; i--)
        System.out.print(elements[i] + " ");
    System.out.println();
}
```

1. Draw the UML diagram for the StackOfStrings class.
2. Implement the StackOfStrings class.
3. Use the test program shown on the following page which creates a StackOfStrings object, and then allows the user to push, pop or peek at the stack.
4. Draw the UML diagram for the StackOfStrings object created in the test program.

```
import java.util.Scanner;

public class Test {

    public static void main(String[] args) {
        StackOfStrings stack = new StackOfStrings();

        System.out.println("STACK OF STRINGS\nType PSH followed by a space "
           + "and a string and a carriage return or\nType POP followed "
           + "by a carriage return or\nType PEK followed by a carriage "
           + "return or \nType STP followed buy a carriage return");

        Scanner input = new Scanner(System.in);

        while (true) {
            System.out.print("Enter command: ");
            String s = input.nextLine();
            String stackCommand = s.substring(0, 3);

            switch (stackCommand) {
                case "PSH":
                    stack.push(s.substring(s.indexOf(" ") + 1));
                    stack.printStack();
                    break;
                case "POP":
                    System.out.println(stack.pop());
                    stack.printStack();
                    break;
                case "PEK":
                    System.out.println(stack.peek());
                    stack.printStack();
                    break;
                case "STP":
                    System.exit(0);
                default:
                    System.out.println("Illegal input. Try again.");
            }
        }
    }
}

class StackOfStrings {
  // Implement the class here
}
```

Submission Guidelines

1. Have your project saved in a folder with your full name.

2. Zip the folder and send it to this email: m.k.alaofi@gmail.com with the email title being **CS112 Lab03 H.W – [YOUR FULL NAME] – [YOUR SECTION]**. E.g CS112 Lab03 H.W – Reham Alhazmi – CS4K