CS 112 Programing II

Lab Session 6: ArrayList and Exception Handling

Objectives

So far you have developed your understanding of object-oriented programming and its main concepts, i.e., encapsulation, inheritance and polymorphism. You have previously used an array to store, retrieve and manipulate objects (e.g. the array of Student objects (studentsList) in Course class and the array of String objects (elements) in the stack of Strings example). You have probably noticed that the array size is fixed, i.e., once the array is created its size is fixed and cannot be changed. The ArrayList class (*java.util.ArrayList*) is provided by Java to help you create arrays with unlimited number of objects.

By the end of this lab session you should be able to:

- create an array list using the class ArrayList.
- store, retrieve and manipulate (add, remove and clear) objects from an array list.
- handle the exceptions that might be thrown in the context of array list.

Exercise 1 *(ArrayList)*

Open Lab_3_1 project that you have used in both lab 3 and lab 4 sessions and update the code as follows: *(you may want to refer to Figure 11.3 in your textbook for a refresher!)*

- In the Course class, replace the array of Student objects (studentList) with an array list to store the Student objects. You have to create an ArrayList of Student objects and store its reference in studentList. Use the statement below to accomplish this step:

```
private ArrayList<Student> studentList=new ArrayList<>();
```

- Add another array list named waitingList to store students who are waiting to be added to the course.
- Update the *setters* and *getters* to accommodate for the changes you have made above.
- Create a *constructor* that creates a Course object with specified *name*, *code* and *professor*.
- In the Course class, add the following methods that manipulate the studentList and the waitingList:

```
    public void addToWaitingList(Student s) {
       // Add the required code to add the Student s to the end of the waitingList  (One statement)
     }

    public void addToStudentList(Student s) {
       // Add the required code to add the Student s to the end of the studentList (One statement)
    }

    public void clearWaitingList() {
       // Add the required code to clear the waitingList (One statement)
    }

    public void AddStudentFromWList() {
       // Add the required code to move the 1st student s from the waitingList to the studentList (i.e., add
Student s to the studentList and remove Student s from the waiting list)(2 to 3 statements)
    }
```

- Replace your previous testing program with the following one:

```
public static void main(String[] args) {
       //Create a new Course object
       Course course=new Course("Programming II","CS112",new Professor("Nora",12));

       // Add 2 students to the course student list
       course.addToStudentList(new Student("Sara","123145",4.5,4));
       course.addToStudentList(new Student("Sahar","123412",2.5,3));

       // Add 2 students to the course waiting list
       course.addToWaitingList(new Student("Rawyah","145677",4.5,4));
       course.addToWaitingList(new Student("Awatif","245677",3.5,4));

       // Print student list and waiting list
       System.out.println("-- Student List --");
       for (Student student : course.getStudentList()) {
          System.out.println("Student ID: "+student.getId()+ ", Name: " + student.getName());
       }
       System.out.println("**********************************");
       System.out.println("-- Student Waiting List --");
       for (Student student : course.getWaitingList()) {
          System.out.println("Student ID: "+student.getId()+ ", Name: " + student.getName());
       }
       System.out.println("**********************************");

       // Add the 1st student from the waiting list to the student list
       course.AddStudentFromWList();
```

```
        // Print student list and waiting list
        System.out.println("**** The lists after adding one student from the waiting list ******");
        System.out.println("-- Student List --");
        for (Student student : course.getStudentList()) {
            System.out.println("Student ID: "+student.getId()+ ", Name: " + student.getName());
        }
          System.out.println("********************************");
        System.out.println("-- Student Waiting List --");
        for (Student student : course.getWaitingList()) {
            System.out.println("Student ID: "+student.getId()+ ", Name: " + student.getName());
        }
        System.out.println("********************************");
}
```

- Read and understand the code provided above and note how Student objects and Professor object are created.

- Run the code above and the output should be similar to the one below:

```
-- Student List --

Student ID: 123145, Name: Sara

Student ID: 123412, Name: Sahar

********************************

-- Student Waiting List --

Student ID: 145677, Name: Rawyah

Student ID: 245677, Name: Awatif

********************************

**** The lists after adding one student from the waiting list ******

-- Student List --

Student ID: 123145, Name: Sara

Student ID: 123412, Name: Sahar

Student ID: 145677, Name: Rawyah

********************************

-- Student Waiting List --

Student ID: 245677, Name: Awatif

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

Exercise 2 *(Exception Handling)*

- Note that after executing the code above the waiting list is having one student only (Awatif). Add the line below to add this student from the waiting list to the student list:

```
course.AddStudentFromWList();
```

- Repeat the same line and run your code. What do you notice? What exception is thrown? Which method is throwing the exception?
- Use try-catch to handle the exception.

Refer to the textbook exercises for more practice on exception handling.