ECS705/ECS717

Lab Sheet 5: Arrays and ArrayLists

Essential exercises:

Exercise 28.

a) Complete the following Java program. This program should:
   - Reads an array of numeric grades between 0 and 100 from the command line.
     - i.e. `java LetterGrades 50 60 20 95`
   - Write a `for` loop that prints out the index, the numerical score and letter grade equivalent for each element. (Grade A: 70-100, Grade B: 60-69, Grade C: 55-59, Grade D: 50-54, Grade F: 0-49)

```
public class LetterGrades{
    public static void main(String[] args){
       //Write your code here
    }
}
```

So `java LetterGrades 50 60 20 95` would print off:

```
Student 0 score is 50 and grade is D
Student 1 score is 60 and grade is B
Student 2 score is 20 and grade is F
Student 3 score is 95 and grade is A
```

b) Write another version of this program using `ArrayList`.

Exercise 29.

Download the file `RandomArray.java` from the course webpage. This class **will NOT** currently compile. You need to fill in the THREE incomplete methods so that when you run the program, it will produce similar output as following:

```
>java RandomArray 5
9 7 2 1 4
Sum: 23
Mean: 4.6

>java RandomArray 8
0 9 5 3 5 6 0 8
Sum: 36
Mean: 4.5
```

i. Complete the method `public void printArray()`
   This method simply prints out the value of each array element.

ii. Complete the method `public int calSum()`
   This method calculates the sum of all values in the array and returns the sum.

iii. Complete the method `public double calMean()`
This method calculates the mean of all values in the array and return the mean. [hint: think about reuse, you should call the method `calSum()` to get the sum].

Exercise 30.

Download the `Student.java` and `StudentList.java` from the course webpage and save them in the same folder.

`Student.java` is a completed file, do NOT modify it. This class defines the `Student`, with three attributes: *name*, *email* and *year of registration* to the course. It has an accessor(getter) for `name`. It has a `toString()` method to give a `String` representation of the `Student` objects.

You need to modify the `StudentList.java`. This class will NOT currently compile. You need to fill in the THREE incomplete methods so that when you run the program, it will produce the output as following[note the text in blue italic should be your own details]:

```
>java StudentList
John Smith has been added to the student list
Tom Will has been added to the student list
your name has been added to the student list
--Begin--
Name:John Smith Email:js@qmul.ac.uk Year:2008
Name:Tom Will Email:tw@qmul.ac.uk Year:2007
Name:your name Email:your email Year:your year
--End--
Tom Will has been removed from the student list
--Begin--
Name:John Smith Email:js@qmul.ac.uk Year:2008
Name:your name Email:your email Year: your year
--End--
```

a) Complete the method `public void printList()`
This method simply prints out each `Student` in the list.

b) Complete the method `public void addToList(Student s)`
This method takes a `Student` object from the parameter and adds the given student to the list. Then print out a message to confirm "*Name* has been added to the list".

c) Complete the method `public void removeFromList(Student s)`
This method takes a `Student` object from the parameter and removes the given student from the list. Then print out a message to confirm "*Name* has been removed from the list".

d) Complete the `main` method by filling in the code to create object `s3` with your own details?

Exercise 31.

i. Define a class, `StudentMarks`, which has an array attribute which can hold seven double values representing individual marks on seven assignments. Define an accessor for the array and methods to retrieve and set the marks for individual assignments. The seven marks should be entered from the keyboard. In addition, define a method to return the mean value of the seven marks. Write a class `StudentMarksMain` with a `main` method to set some sample mark values via the class's methods, and check that your class works properly.

ii. Add a further method to the StudentMarks class to return the index of the highest of the seven assignment marks held in the array. This method should use a for loop to find the correct index.

iii. In the class just written, which index is returned if there are two equal highest marks for a student; that of the first or the second occurrence? Add a further method so that the other occurrence is returned instead.

Desirable exercises:

Exercise 32.

Download the `YearlyRainfall.java, RainfallMain.java` and those precompiled classes for display from the course webpage. Complete the implementation of the `YearlyRainfall` class provided in outline in `YearlyRainfall.java. YearlyRainfall` should define an array attribute of twelve integers to hold the monthly rainfall values for a single year. Its public methods should be implemented as follows:

- **getMonthAmount** – takes a single integer argument (within the index bounds of the array attribute: 0 for January etc) and returns the integer rainfall amount for that month. Return –1 for an invalid month.
- **setMonthAmount** – takes an integer argument as the month and a second integer argument representing the rainfall value to be set for that particular month. The method should set the rainfall figure for the given month to the given value. Do nothing for an invalid month or amount.
- **getMean** – takes no argument and returns the mean monthly value for the whole year (the sum of the individual monthly values, divided by the number of months). The result should be returned as a **double**.

Use the **RainfallDisplay** attribute to display a chart of your data. An instance and accessor has already been defined in the outline. It has the following methods in its public interface:

```
// Display each value in data in a separate bar.
// The number of bars will be equal to the length of data.
// A copy of the data is *not* made.
      public void setData(int[] d)
// Provide a set of labels for the bars.
      public void setBarLabels(String[] labels)
// Redisplay the data after it has been altered.
      public void repaint()
```

You should send the display object a **repaint** message any time that the data stored in the YearlyRainfall's array attribute is modified, so that it can update its appearance. The code in `RainfallMain.java` provides a simple main method to create a `YearlyRainfall` object. This main method allows a user to interactively modify the `YearlyRainfall` object. You should feel free to alter this for testing purposes, if you wish.

Optional exercises:

Exercise 33.

i. Design a class, `DayTemperatures`, to record a set of hourly temperature readings over the course of a day. The class should include a method, `hottesthours`, that returns an array of the hottest hours. The length of the array will vary according to the number of hours in the day that have the same highest temperature. You may assume that these hours are numbered according to the 24 hour clock, that is they have values in the range **0 .. 23**. It should also include a method, `meanTemperature`, that returns the mean temperature for the day.
ii. Add the class, `WeeklyTemperatures`, to your solution to store a week's set of `DayTemperatures`. The class should include a method, `meanDailyTemperature`, that returns the mean value of the individual daily mean temperatures.