COSC1073 Programming 1, Semester 1 2021

Week 4 Assessment

Create a Java Project prefixed with your student ID followed by -Week4, for example:

s1234567-Week4

Create a git repository and publish it to the **rmit-p1-s1-2021** organisation on GitHub with the same name as a **private** repository. The repository URL for the example project name above would be:

https://github.com/rmit-p1-s1-2021/s1234567-Week4

Instructions to create a git repository and publishing it to the **rmit-p1-s1-2021** organisation on GitHub can be found in TuteLab 1 and TuteLab 2.

Create a class called **Program** which contains the **main** method and another class called **RandomSequence**; you can use the default package.

Implement the **RandomSequence** class to contain the following:

- Private instance variable called **sequence** which is an int array.
- Private instance variable called **bound** which is an int.
- Public constructor that has 2 arguments, **length** and **bound** both of which are int.
  The constructor should initialise both instance variables and populate the **sequence** array with random values from 0 (inclusive) up to the **bound** (exclusive).
- Public method called **generateSequence**; this method will populate the **sequence** array with random values from 0 (inclusive) up to the **bound** (exclusive).
- Public method called **printSequence**; this method prints the **sequence** array with each value separated by a comma and space and ends with a new line. Note: The last value in the array is not followed by a command and space, it is only followed by a new line.
- Public method called **findMax**; this method returns an int which is the highest value in the **sequence** array.
- Public method called **findMin**; this method returns an int which is the lowest value in the **sequence** array.

Note: Random values can be generated using the built-in Random class; import with java.util.Random. Example of using Random to generate a value stored into r from 0 (inclusive) to 10 (exclusive):

```
Random random = new Random();
int x = 10;
int r = random.nextInt(x);
```

Write the **main** method to prompt the user to enter a length and a bound, both of which must be positive integers (i.e., whole numbers). You can use the Scanner class to get input from the user. You must validate the user input and if a number less than or equal to zero is entered an error message should be displayed and the user prompted to try again; this repeats until valid input has been entered.

Note: You can assume all input will be integers, i.e., scanner.nextInt(); will not fail due to mismatched input.

After valid input is entered create a **RandomSequence** instance with the provided length and bound values and call the **printSequence** method, followed by **findMax** and **findMin** (you need to print the returned value of the find methods). Then call the **generateSequence** method and print "Generated new sequence." afterwards. Again, call **printSequence** followed by **findMax** and **findMin** (you need to print the returned value of the find methods).

Lastly the program should display "Program ending." before terminating.

See below for sample output of the program.

Sample Output

Sample output #1; invalid input:

```
Enter length; must be positive: 0
Invalid, please try again.
Enter length; must be positive: 10

Enter bound; must be positive: -5
Invalid, please try again.
Enter bound; must be positive: 100

67, 89, 78, 86, 34, 32, 34, 31, 10, 46
Max: 89
Min: 10
Generated new sequence.
14, 14, 55, 75, 40, 82, 19, 38, 35, 23
Max: 82
Min: 14

Program ending.
```

Sample output #2; different length and bound:

```
Enter length; must be positive: 5

Enter bound; must be positive: 50

6, 25, 7, 13, 1
Max: 25
Min: 1
Generated new sequence.
36, 4, 36, 28, 23
Max: 36
Min: 4

Program ending.
```

Marking Guide

Implementation of the RandomSequence class	[2 marks]
Getting and validating input from the user	[1 mark]
Calling the RandomSequence methods as specified	[1 mark]
Correct output	[1 mark]
Code quality; GitHub repository, indentation, consistency, variable names, code reuse, etc…	[1 mark]
Total	**[6 marks]**

Submission Instructions

Export the Eclipse project as an archive; the file name should be the same as the project name, for example **s1234567-Week4.zip** and submit the zip file to Canvas under Assignments.

Instructions to export an Eclipse project can be found on Canvas under Exporting Projects.

Academic Integrity and Plagiarism (Standard RMIT Warning)

Your code will be automatically checked for similarity against other submissions so please make sure your submitted project is entirely your own work.

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge, and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and / or ideas of others you have quoted (i.e., directly copied), summarised, paraphrased, discussed, or mentioned in your assessment through the appropriate referencing methods.
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source.
- Copyright material from the internet or databases.
- Collusion between students.

For further information on our policies and procedures, please refer to the University website.

Assessment Declaration

When you submit your project electronically, you agree to the RMIT Assessment Declaration.