CSCI 1301: Introduction to Computing and Programming

Lab 08 – Loops; One-Dimensional Arrays

Introduction

This lab assignment continues to give you practice using loops, particularly loops with variable termination conditions, and it also provides you an opportunity to use one-dimensional arrays.

Recall that an array is used to store a collection of data. The data can be values of Java primitive data types or else objects (for instance, **`String`** objects), but the data stored must all be of the same type. An array is created in Java using a statement of the form

BaseType`[]` *arrayName* **`=`** **`new`** *BaseType*`[`*length*`];`

where *BaseType* is the underlying type of data stored in the array, *arrayName* is just a variable name used to refer to the array, and *length* is a positive integer indicating the number of elements in the array. An array element can be referred to using *arrayName*`[`*i*`]`, where *i* is an integer index (starting with 0). The length of the array is stored in the variable *arrayName*`.`**`length`**.

The arrays used in this assignment will all be one-dimensional (that is, they won't be arrays of arrays), and they will all use Java primitive data types as the base type.

For the assignment, you are to create a program using two arrays of type **`double`** to store multiple values for *x* and *y*, where *y* is defined as the below function.

$$y = 20.0 * |\sin x|$$

It is assumed that the angles for the sine function are measured in radians. In the program, both the values for *x* and *y* are computed based on input from the user. Once the values are found, the program should print them out as indicated by the examples. The program should also print out a graphical representation of the function, using a sequence of '*'s to indicate the magnitude of *y*.

Lab Objectives

By the end of the lab, you should be able to:

- create nested loops and loops with variable termination conditions;
- create and manipulate one-dimensional arrays.

Prerequisites

The lab deals with material from Chapter 4 (loops) and Section 7.1 (arrays).

What to Submit

The file **`StarGraph.java`** should be submitted to eLC for grading.

Instructions

1. The class should be declared as public and called **`StarGraph.`**
2. You must include a comment stating your name, the date, program purpose, and containing a statement of academic honesty as shown in Lab 02. When writing, you must also abide by the Java formatting and naming conventions.

3. The program should prompt the user for the following information:
    - the size ***N*** of the array (that is, the number values of x for which y will be computed);
    - x_{min}, a minimum value for x;
    - an *increment* for x (a value by which x is increased).

   The messages to display to the user are shown in the examples below.
4. Note that the size of the arrays must be a positive integer. If the user enters 0 or a negative integer, then the program should print out an error message and terminate the program, as indicated in the examples.
5. The increment value should be a positive floating point value (a **`double`**). If the user enters 0 or a negative number, then the program should print an error message and terminate.
6. Once ***N***, x_{min}, and the increment are established, the program should create two double arrays, one to hold the x values and one to hold the y values. The size of both arrays will be N.

   The first element of the x-array will hold x_{min}, and each successive element will be incremented by the value provided by the user. The values for the y-array will be defined using the corresponding value from the x-array and the function defined earlier. To compute the value for y, the program should use the **`Math.abs`** and **`Math.sin`** methods.

   Observe that in order to populate the arrays with values, loops must be used.
7. Once computed, the values of x and y should be printed to the console. Each pair of values for x and y stored in the arrays should be printed. Format the values of x and y to have 3 digits after the decimal point.
8. After the values have been printed, a simple graph of the function should be plotted using asterisks (*). Each successive line corresponds to an x value, and the number of '*'s on the line corresponds to the whole-number part of y (the fractional part is ignored/truncated). To compute the number of '*'s, you should not round the value for y (instead, truncate it).

Additional Requirements

These are things that make the graders lives easier, and ultimately, you will see in the real world as well. Remember the teaching staff does not want to touch your code after they gave you requirements; they want to see the perfect results they asked for! Here is a checklist of things you can **lose points** for:

- (-1 point) If the source file(s)/class(es) are named incorrectly (case matters!)
- (-1 point) If your source file(s) have a package declaration at the top
- (-1 point) If any source file you submit is missing your Statement of Academic Honesty at the top of the source file. All submitted source code files must contain your Statement of Academic Honesty at the top of each file.
- (-1 point) If you have more than one instance of Scanner in your program. Your program should only have one instance of Scanner.
- (-1 point) Inconsistent I/O (input/output) that does not match our instructions or examples exactly (unless otherwise stated in an assignment's instructions). Your program's I/O (order, wording, formatting, etc.) must match our examples and instructions.

- (-2 points) If your submission is late
- (-7 points) If the source file(s) are not submitted before the specified deadline's late period ends (48 hours after the deadline) or if they do not compile.
- If your (-1 point) comments or (-1 point) variables are "lacking"
    - Here, "lacking" means that you or a TA can find **any** lines of code or variables that take more than 10 seconds to understand, and there is no comment, or the variable name does not make sense (variable names like **b**, **bb**, **bbb**, etc. **will almost never be acceptable)**
- (-1 point) Indentation is not consistent throughout your source code
    - Refresh your memory of indentation patterns in chapter 2 in the course textbook
    - Be careful of a combination of tabs and spaces in your files (use one or the other)
- (-7 points) If you use a non-standard Java library or class (StringBuilder class, Arrays class, any class in java.util.stream) or other code specifically disallowed by the lab/project. For this assignment, you are **NOT** permitted to use a **break** or **continue** statement inside of the body of a loop, and using one of these statements in the body of a loop will result in a grade of zero on this assignment.

If any of the above do not make sense to you, talk to a TA or ask your lab instructor.

eLC Submission and Grading

After you have completed and thoroughly tested **`StarGraph.java`** using both the examples included here and your own, submit the program to ***eLC*** in order to receive credit for the lab. Always double check that your submission was successful on ***eLC***!

The lab will be graded according to the following guidelines.

- Students may earn 3 points for lab attendance and 7 points if their program passes various test cases that we use for grading. **Some test cases may use values taken from the examples in this document and some test cases may not**. You should come up with additional test cases to check that your program is bug free.
- All source code must adhere to good Java programming style standards as discussed in lecture class and in the course textbook readings.
- Points will be deducted for not following any of the aforementioned instructions or requirements.

Sample Input and Output

These are sample program runs. Input is underlined. Your input and output must be consistent with what is shown here. Each example is a single execution of a correctly working program.

Example 1

```
Please enter the number of x values to process: 10
Enter a minimum value for x: 0
Enter the amount to increment x: 0.25

Values
x: 0.000, y: 0.000
x: 0.250, y: 4.948
```

```
x: 0.500, y: 9.589
x: 0.750, y: 13.633
x: 1.000, y: 16.829
x: 1.250, y: 18.980
x: 1.500, y: 19.950
x: 1.750, y: 19.680
x: 2.000, y: 18.186
x: 2.250, y: 15.561

Graph
:
:****
:*********
:*************
:****************
:******************
:*******************
:*******************
:******************
:***************
```

Example 2

```
Please enter the number of x values to process: 10
Enter a minimum value for x: 0
Enter the amount to increment x: 0.31415926535

Values
x: 0.000, y: 0.000
x: 0.314, y: 6.180
x: 0.628, y: 11.756
x: 0.942, y: 16.180
x: 1.257, y: 19.021
x: 1.571, y: 20.000
x: 1.885, y: 19.021
x: 2.199, y: 16.180
x: 2.513, y: 11.756
x: 2.827, y: 6.180

Graph
:
:******
:***********
:****************
:*******************
:********************
:*******************
:****************
:***********
:******
```

Example 3

```
Please enter the number of x values to process: 5
Enter a minimum value for x: 0
Enter the amount to increment x: 3.14159265359

Values
x: 0.000, y: 0.000
x: 3.142, y: 0.000
x: 6.283, y: 0.000
x: 9.425, y: 0.000
x: 12.566, y: 0.000

Graph
:
:
:
:
:
```

Example 4

```
Please enter the number of x values to process: 9
Enter a minimum value for x: -1.5707963267
Enter the amount to increment x: 0.78539816339

Values
x: -1.571, y: 20.000
x: -0.785, y: 14.142
x: 0.000, y: 0.000
x: 0.785, y: 14.142
x: 1.571, y: 20.000
x: 2.356, y: 14.142
x: 3.142, y: 0.000
x: 3.927, y: 14.142
x: 4.712, y: 20.000

Graph
:********************
:**************
:
:**************
:********************
:**************
:
:**************
:********************
```

Example 5

```
Please enter the number of x values to process: 20
Enter a minimum value for x: 0
Enter the amount to increment x: 0.25

Values
x: 0.000, y: 0.000
x: 0.250, y: 4.948
x: 0.500, y: 9.589
x: 0.750, y: 13.633
x: 1.000, y: 16.829
x: 1.250, y: 18.980
x: 1.500, y: 19.950
x: 1.750, y: 19.680
x: 2.000, y: 18.186
x: 2.250, y: 15.561
x: 2.500, y: 11.969
x: 2.750, y: 7.633
x: 3.000, y: 2.822
x: 3.250, y: 2.164
x: 3.500, y: 7.016
x: 3.750, y: 11.431
x: 4.000, y: 15.136
x: 4.250, y: 17.900
x: 4.500, y: 19.551
x: 4.750, y: 19.986

Graph
:
:****
:*********
:*************
:****************
:******************
:*******************
:*******************
:******************
:***************
:***********
:*******
:**
:**
:*******
:***********
:***************
:*****************
:*******************
:*******************
```

Example 6

```
Please enter the number of x values to process: 20
Enter a minimum value for x: 0
Enter the amount to increment x: 0.78539816339

Values
x: 0.000, y: 0.000
x: 0.785, y: 14.142
x: 1.571, y: 20.000
x: 2.356, y: 14.142
x: 3.142, y: 0.000
x: 3.927, y: 14.142
x: 4.712, y: 20.000
x: 5.498, y: 14.142
x: 6.283, y: 0.000
x: 7.069, y: 14.142
x: 7.854, y: 20.000
x: 8.639, y: 14.142
x: 9.425, y: 0.000
x: 10.210, y: 14.142
x: 10.996, y: 20.000
x: 11.781, y: 14.142
x: 12.566, y: 0.000
x: 13.352, y: 14.142
x: 14.137, y: 20.000
x: 14.923, y: 14.142

Graph
:
:**************
:********************
:**************
:
:**************
:********************
:**************
:
:**************
:********************
:**************
:
:**************
:********************
:**************
:
:**************
:********************
:**************
```

Example 7

```
Please enter the number of x values to process: 10
Enter a minimum value for x: -5
Enter the amount to increment x: 0.20

Values
x: -5.000, y: 19.178
x: -4.800, y: 19.923
x: -4.600, y: 19.874
x: -4.400, y: 19.032
x: -4.200, y: 17.432
x: -4.000, y: 15.136
x: -3.800, y: 12.237
x: -3.600, y: 8.850
x: -3.400, y: 5.111
x: -3.200, y: 1.167

Graph
:*******************
:*******************
:*******************
:*******************
:*****************
:***************
:************
:********
:*****
:*
```

Example 8

```
Please enter the number of x values to process: 20
Enter a minimum value for x: -1.5707963267
Enter the amount to increment x: 0.15707963267

Values
x: -1.571, y: 20.000
x: -1.414, y: 19.754
x: -1.257, y: 19.021
x: -1.100, y: 17.820
x: -0.942, y: 16.180
x: -0.785, y: 14.142
x: -0.628, y: 11.756
x: -0.471, y: 9.080
x: -0.314, y: 6.180
x: -0.157, y: 3.129
x: 0.000, y: 0.000
x: 0.157, y: 3.129
x: 0.314, y: 6.180
x: 0.471, y: 9.080
x: 0.628, y: 11.756
```

```
x: 0.785, y: 14.142
x: 0.942, y: 16.180
x: 1.100, y: 17.820
x: 1.257, y: 19.021
x: 1.414, y: 19.754

Graph
:********************
:*******************
:*******************
:*****************
:****************
:**************
:***********
:*********
:******
:***
:
:***
:******
:*********
:***********
:**************
:****************
:*****************
:*******************
:*******************
```

Example 9

```
Please enter the number of x values to process: 10
Enter a minimum value for x: 1.57079632679
Enter the amount to increment x: 1.57079632679

Values
x: 1.571, y: 20.000
x: 3.142, y: 0.000
x: 4.712, y: 20.000
x: 6.283, y: 0.000
x: 7.854, y: 20.000
x: 9.425, y: 0.000
x: 10.996, y: 20.000
x: 12.566, y: 0.000
x: 14.137, y: 20.000
x: 15.708, y: 0.000

Graph
:********************
:
:********************
:
:********************
:
:********************
:
:********************
:
```

Example 10

```
Please enter the number of x values to process: 0
The number of x values must be an integer greater than 0.
```

Example 11

```
Please enter the number of x values to process: 100
Enter a minimum value for x: -50
Enter the amount to increment x: 0
The increment must be a decimal number greater than 0.
```