# Nested Loops With Diamonds

# CSS 161 : Fundamentals of Computing

# Rob Nash, Senior Lecturer

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\*\*\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\*\*\*\*\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\*\*\*\*\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\*\*\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_\_\_\_

\_\_\_\*\*\*\*\*\*\*\*\_\_\*\*\*\*\*\*\*\*\_\_\_

\_\_\*\*\*\*\*\*\*\*\_\_\_\_\*\*\*\*\*\*\*\*\_\_

\_\*\*\*\*\*\*\*\*\_\_\_\_\_\_\*\*\*\*\*\*\*\*\_

## Summary

Extend your understanding of ***Repetition Control Structures*** by building a picture of diamonds. This picture will be configurable with respect both diamond size and the number of diamonds, as specified by a configuration file your program will parse.

## Work Items

1. Submit only your “Diamonds.java” program (“.java” text file) and “config.txt” file via the website’s dropbox.
2. Pay extra attention to code formatting, style, comments, and basic programming practices learned so far.

## Introduction – Sample Executions

Your assignment is to write a program that draws rows and rows of diamonds. It may be easier to view sample executions of such a program, and so I’ve included some below. Your program will read from an input file (“config.txt”) the size of each diamond and the number of rows of diamonds using the Scanner technique covered in class or lab. In addition, your software will read in two characters from the input file, and use those characters to draw the diamond picture.

## Building In Steps, From the Bottom Up

Before you draw the Mitsubishi Diamond, learn how to draw a triangle.

\_\_\_\_

\_\_\_

\_\_

\_

Next, learn to draw a box, composed of two different triangles. The example uses ‘\_’ for the first character and ‘\*’ for the second.

\_\_\_\_

\_\_\_\*

\_\_\*\*

\_\*\*\*

Next, mirror the triangles about the x-axis, so you can produce this box made of triangles, too.

\_\_\_\_

\*\_\_\_

\*\*\_\_

\*\*\*\_

Finally, compose the two boxes size-by-side so you can make half-a-diamond.

\_\_\_\_\_\_\_\_

\_\_\_\*\*\_\_\_

\_\_\*\*\*\*\_\_

\_\*\*\*\*\*\*\_

From a half-diamond box, mirror this output about the y-axis so you can complete one of the three diamonds that will be in your final picture.

\_\_\_\_\_\_\_\_

\_\_\_\*\*\_\_\_

\_\_\*\*\*\*\_\_

\_\*\*\*\*\*\*\_

\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*

\_\*\*\*\*\*\*\_

\_\_\*\*\*\*\_\_

\_\_\_\*\*\_\_\_  
\_\_\_\_\_\_\_\_

Continue to build your picture in pieces and then assemble them. Using methods is a great way to make this assignment easier - keep in mind you may have to pass data to the methods including the size of the diamond, the characters used to draw the diamond, and other relevant information.

## Empirical & Black Box Analysis

Black Box testing focuses on the external behaviours of a system while specifically ignoring the internal design details (which may not be available, either, as it is in this case). Studying the output in the figures examples below will reveal these ***invariants*** that should be built into your software:

1. The number of diamonds is directly controlled by the numDiamonds input variable.
2. The number of “stars” internal to a diamond is proportional to the size variable.
3. Ultimately, you’re building one big box where every line (or row) is composed of triangles, build with (nested) looping patterns similar to the following ***pseudocode***:

Loop(on number of rows)

Loop(draw left border(s)) (0 to many times)

**Loop**(draw diamond(s)) (0 to many times) //note this one step expands to a similar pattern

Loop(draw right border(s)) (0 to many times)

Note that the “draw diamond(s)” step bolded above actually represents multiple sub-steps, just as a method call actually represents multiple lines of code (although one line methods do exist). Every diamond(really, two triangles) is composed of a left border of spaces (char 1), the stars that compose the triangle (char 2), and then the right border spaces(char 1 again). Determining just how many spaces, stars, and then spaces to print is up to you. Using the pattern above as a ***template***, you should add the additional sub-steps omitted above before the **pseudocode** is detailed enough to be simply translated into code.

## Program Requirements

There exists many ways to build this code, but the requirements stay the same. Your software should:

1. Read in 4 data items from a sequential text file using a Scanner.
2. Draw a picture composed of diamonds that matches the examples provided here exactly.
   1. How large each diamond will be (size)
   2. How many diamonds will be displayed on the row.
   3. What character should be used to draw the diamonds (char 1)
   4. A second character used to draw the space around the diamonds (char 2)

## 84% Target – Start Here

This uses only 3 of the 4 input variables to produce the logo: size, and the two characters to draw it. Said another way, you can get up to a B- in credit if you are only able to draw only a single diamond logo. You should first build this software, since it is easy to make this do multiple logos once you have gotten just one done.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| numRows | size | char one | char two | output |
| 1 | 2 | ‘\*’ | ‘\_’ | \_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\*\*\_\_\_\_\_  \_\_\_\_\_\*\*\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_  \_\_\*\*\*\*\*\*\*\*\_\_  \_\*\*\*\*\_\_\*\*\*\*\_ |
| 1 | 3 | ‘\_’ | ‘\*’ | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\_\_\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\_\_\_\_\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\_\_\_\_\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\_\_\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\_\_\_\_\_\_\_\_\_\_\_\_\*\*\*  \*\*\_\_\_\_\_\_\*\*\_\_\_\_\_\_\*\*  \*\_\_\_\_\_\_\*\*\*\*\_\_\_\_\_\_\* |
| 1 | 6 | ‘ ‘ | ‘\*’ | \*\*  \*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*  \*\*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\* |

## 100% Target (ONLY Approach Once 84% is Done)

This uses all 4 input variables to adjust the size of the logo and the number of consecutive logos drawn back-to-back in the x-axis.

**int** size = 5;

**char** a = '0';

**char** b = '1';

**int** numDiamondRows = 2;

111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111

111111111111110011111111111111111111111111110011111111111111111111111111110011111111111111111111111111110011111111111111

111111111111100001111111111111111111111111100001111111111111111111111111100001111111111111111111111111100001111111111111

111111111111000000111111111111111111111111000000111111111111111111111111000000111111111111111111111111000000111111111111

111111111110000000011111111111111111111110000000011111111111111111111110000000011111111111111111111110000000011111111111

111111111110000000011111111111111111111110000000011111111111111111111110000000011111111111111111111110000000011111111111

111111111111000000111111111111111111111111000000111111111111111111111111000000111111111111111111111111000000111111111111

111111111111100001111111111111111111111111100001111111111111111111111111100001111111111111111111111111100001111111111111

111111111111110011111111111111111111111111110011111111111111111111111111110011111111111111111111111111110011111111111111

111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111

111110000000000000000000011111111110000000000000000000011111111110000000000000000000011111111110000000000000000000011111

111100000000001100000000001111111100000000001100000000001111111100000000001100000000001111111100000000001100000000001111

111000000000011110000000000111111000000000011110000000000111111000000000011110000000000111111000000000011110000000000111

110000000000111111000000000011110000000000111111000000000011110000000000111111000000000011110000000000111111000000000011

100000000001111111100000000001100000000001111111100000000001100000000001111111100000000001100000000001111111100000000001

**int** size = 7;

**char** a = '\*';

**char** b = ' ';

**int** numDiamondRows = 3;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*

\*\*\*\*\*\* \*\* \*\*\*\*\*\*\*\*\*\*\*\* \*\* \*\*\*\*\*\*\*\*\*\*\*\* \*\* \*\*\*\*\*\*

\*\*\*\*\* \*\*\*\* \*\*\*\*\*\*\*\*\*\* \*\*\*\* \*\*\*\*\*\*\*\*\*\* \*\*\*\* \*\*\*\*\*

\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*

\*\*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*\*\* \*\*\*

\*\* \*\*\*\*\*\*\*\*\*\* \*\*\*\* \*\*\*\*\*\*\*\*\*\* \*\*\*\* \*\*\*\*\*\*\*\*\*\* \*\*

\* \*\*\*\*\*\*\*\*\*\*\*\* \*\* \*\*\*\*\*\*\*\*\*\*\*\* \*\* \*\*\*\*\*\*\*\*\*\*\*\* \*

## Hints

* Don’t wait till the last minute to get started or get help from the instructor.
* Use comments in your code.
* Hardcode the size and char one and two at first; later, read them from a file.
* Break up this program into smaller functions and call those
  + First, build a function to make the simple triangle described in the step-by-step analysis above
    - Note that you can (and should) call a function inside of a loop