

## NESTED LOOP STRUCTURE

### Objectives

- 1) Understand how nested loop structure works
- 2) Discuss when to use a nested loop structure
- 3) Create programs utilizing nested loop structures

### Executing Nested Loop Structure

Just like selection structures, loop structures can also be nested. That is, a loop structure can be written as one of the statements of another loop structure.

For example:

```
for(int i=1; i<=3; i++) {
    for(int j=1; j<=i; j++) {
        System.out.print("* ");
    }
    System.out.println("");
}
```

How to execute the given nested loop structure:

Step 1: Execute the initialization part of the **outer loop**. In our example, the variable i will be initialized to 1.

Step 2: Test the condition part of the **outer loop**. If the condition  $i \leq 3$  evaluates to true, proceed to Step 3. Otherwise, terminate the loop.

Step 3: Execute the statements of the outer loop.

Step 3.1: Execute the initialization part of the **inner loop**. In our example, the variable j will be initialized to 1.

Step 3.2: Test the condition part of the **inner loop**. If the condition  $j \leq i$  evaluates to true, proceed to Step 3.3. Otherwise, proceed to Step 3.6.

Step 3.3: Execute the **statements** of the inner loop.

Step 3.4: Perform the incrementation part of the **inner loop**. This part will increment the value of variable j by 1 as indicated by the statement  $j++$ .

Step 3.5: Repeat Step 3.2.

Step 3.6: Execute the **other statements** of the outer loop.

Step 4: Perform the incrementation part of the **outer loop**. This part will increment the value of variable i by 1 as indicated by the statement  $i++$ .

Step 5: Repeat Step 2.

If you have traced the code correctly, your output should be similar to the one below:



CCS102- COMPUTER PROGRAMMING 1 LABORATORY MANUAL

**When to use Nested Loops?**

Nested loops are useful when for each pass through the outer loop, you need to repeat some action on the data in the outer loop. For example, you want to count how many vowels can be found in a given String. The outer loop would read the contents of the given String character per character and the inner loop would check each character against a list of vowel characters.

**References:**

[1] Oracle Java Documentation. 2021. The if-then and if-then-else Statements. Retrieved February 1, 2022 from <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html>

PAMANTASAN NG CABUYAO – COLLEGE OF COMPUTING AND ENGINEERING **65**  
CCS102- COMPUTER PROGRAMMING 1 LABORATORY MANUAL

**Lesson 6 NESTED LOOP STRUCTURE**

**Activity #1**

<b>Name (LN, FN MI.)</b>	<b>Merdelyn Blan Charles Nathaniel Dueñas</b>			<b>Score:</b>	
<b>Section:</b>	<b>1BSIT-B</b>	<b>Instructor :</b>	<b>Sharmaine Maglapuz</b>	<b>Sem./ A.Y.</b>	<b>2nd Sem 2022-20 23</b>
<b>Project Link:</b>	<b><a href="https://www.codiva.io/p/bbf0b0e3-e3d0-400f-8762-6ae15fbb38c7">https://www.codiva.io/p/bbf0b0e3-e3d0-400f-8762-6ae15fbb38c7</a></b>				

**INSTRUCTION:**

Create a Java program that will use nested loops to display the given output below:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

GRADING RUBRIC:

TRAITS	GRADING SCALE				WEIGHT	SCORE
	NEEDS IMPROVEMENT 1	FAIR 2	GOOD 3	EXCELLENT 4		
FUNCTIONALITY	The program produced an incorrect pattern.	The program produced the given pattern but with two missing lines.	The program produced the given pattern but with one missing line.	The program produced the exact given pattern.	4	
READABILITY	The code is poorly organized and very difficult to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is fairly easy to read.	The code is exceptionally well organized and very easy to follow.	2	
EFFICIENCY	The code appears to be patched together.	The code is unnecessarily long.	The code is fairly efficient without sacrificing readability.	The code is extremely efficient without sacrificing readability.	4	
TOTAL						

```

int num1=0;

    for (int i = 1; i <= 5; ++i, num1 = 0) {
        for (int space = 1; space <= 5 - i; ++space) {
            System.out.print(" ");
        }

        while (num1 != 2 * i - 1) {
            System.out.print("* ");
            ++num1;
        }

        System.out.println();
    }
}
}
}

```

SAMPLE OUTPUT





Lesson 6 LOOP STRUCTURE

Activity #2

Name (LN, FN MI.)	Merdelyn Blan Charles Nathaniel Dueñas			Score:	
Section:	1BSIT-B	Instructor :	Sharmaine Maglapuz	Sem./ A.Y.	2nd Sem 2022-20 23
Project Link:	<a href="https://www.codiva.io/p/2dbb7a8c-f2a4-4333-902f-3aa4488266ae">https://www.codiva.io/p/2dbb7a8c-f2a4-4333-902f-3aa4488266ae</a>				

INSTRUCTION:

Create a Java program that will use nested loops to display the given output below:

GRADING RUBRIC:

TRAITS	GRADING SCALE				WEIGHT	SCORE
	NEEDS IMPROVEMENT 1	FAIR 2	GOOD 3	EXCELLENT 4		
FUNCTIONALITY	The program did not accomplish more than 2 of the requirements.	The program did not accomplish 2 of the requirements.	The program did not accomplish 1 of the requirements.	The program accomplished all of the program requirements.	4	
READABILITY	The code is poorly organized and very difficult to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is fairly easy to read.	The code is exceptionally well organized and very easy to follow.	2	
EFFICIENCY	The code appears to be patched together.	The code is unnecessarily long.	The code is fairly efficient without sacrificing readability.	The code is extremely efficient without sacrificing readability.	4	
TOTAL						

**File: OddEven.java**

```
class Pyramid {  
    public static void main(String[] args) {  
  
        int row = 5, n = 0, count0 = 0, count1 = 0;  
  
        for (int i = 1; i <= row; ++i) {  
            for (int space = 1; space <= row - i; ++space) {  
                System.out.print(" ");  
                ++count0;  
            }  
  
            while (n != 2 * i - 1) {  
                if (count0 <= row - 1) {  
                    System.out.print((i + n) + " ");  
                    ++count0;  
                } else {  
                    ++count1;  
                    System.out.print((i + n - 2 * count1) + " ");  
                }  
  
                ++n;  
            }  
            count1 = count0 = n = 0;  
  
            System.out.println();  
        }  
    }  
}
```

SAMPLE OUTPUT

```
      1
    2 3 2
  3 4 5 4 3
4 5 6 7 6 5 4
5 6 7 8 9 8 7 6 5
```

