Kaitlin Hoffmann

Office Hours:

SH 243 MR 11:00 - 12:30 PM via appointment https://calendly.com/hoffmank4/15min

Email: hoffmank4@newpaltz.edu

For TA Office Hours and Email – Please see syllabus

NEW! Supplemental Instruction: Sign up at my.newpaltz.edu

NESTED LOOPS

COMPUTER SCIENCE I

OBJECTIVES

- Nested loops

A LOOP CAN BE NESTED INSIDE ANOTHER LOOP

- Nested loops consist of an ***outer loop*** and one or more ***inner loops***. Each time the outer loop is repeated, the inner loops are reentered, and started anew.

- These will be useful when sorting arrays

- A useful and popular way to learn to use nested loops is with ***pattern printing***

A LOOP CAN BE NESTED INSIDE ANOTHER LOOP

- What will the output of this be?

```
for(int i = 0; i < 3; i++) {
    for(int j = 0; j<5; j++) {
        System.out.print("* ");
    }
    System.out.println();
}
```

```
for(int i = 0; i < 3; i++) {
    for(int j = 0; j<5; j++) {
        System.out.print("* ");
    }
    System.out.println();
}
```

Important: i is **constant** while in the second loop

The 1st time we loop, i is **0** – for(int j = **0**; **j<=5**; j++) * * * * *

The 2nd time we loop, i is **1** – for(int j = 0; **j<=5**; j++) * * * * *

The 3rd time we loop, i is **2** – for(int j = 0; **j<=5**; j++) * * * * *

A LOOP CAN BE NESTED INSIDE ANOTHER LOOP

- Notice how the first loop is creating the **rows**
- The second loop is creating the **columns**
- i will stay **constant** as the second loop is executing

```
for(int i = 0; i < 3; i++) {
    for(int j = 0; j<5; j++) {
        System.out.print("* ");
    }
    System.out.println();
}
```

Output:

```
* * * * *
* * * * *
* * * * *
```

EXERCISE 1

- Write a program that will output the pattern below:

Output:

```
* * * *
* * * *
* * * *
* * * *
* * * *
```

EXERCISE 1 – SOLUTION

- Like before, the first loop is creating the **rows**
- The second loop is creating the **columns**

```
for(int i = 0; i < 5; i++) {
    for(int j = 0; j<4; j++) {
        System.out.print("* ");
    }
    System.out.println();
}
```

Output:

```
* * * *
* * * *
* * * *
* * * *
* * * *
```

A LOOP CAN BE NESTED INSIDE ANOTHER LOOP

- How could we create the pattern below?

```
*
* *
* * *
* * * *
```

WHAT IS THE PATTERN?

- Let's break this down. How many rows are there? **4**
- How many columns are there? **Starts at 1 then goes to 4**
- What have we established stays constant inside the second loop? **i – we can use this to our advantage in the second loop!**

```
*
* *
* * *
* * * *
```

DETERMINE THE ROWS

- The rows are easy to do – just have the first loop continue until the amount of rows you want, in this case, 4:

```
for(int i = 0; i<4; i++) {

}
```

```
*
* *
* * *
* * * *
```

DETERMINE THE COLUMNS

- The columns take a second to think about…however, we can use the i index to our advantage and only have the second loop print while j<=i since i starts at 0, then 1, up until 4:

```
for(int i = 0; i<4; i++) {
    for(int j = 0; j<=i; j++) { // we have j<=i because the highest
        System.out.print("* "); // i can be is 3 -> (0, 1, 2, 3)
    }
    System.out.println();
}
```

The 1st time we loop, i is **0** – for(int j = 0; **j<=0**; j++) *****

The 2nd time we loop, i is **1** – for(int j = 0; **j<=1**; j++) *** ***

The 3rd time we loop, i is **2** – for(int j = 0; **j<=2**; j++) *** * ***

The 4th time we loop, i is **3** – for(int j = 0; **j<=3**; j++) *** * * ***

TRY THIS

- Try to create the pattern below.
- Remember, the first loop is the **row**, the second loop is the **column**.

```
* * * *
* * *
* *
*
```

TRY THIS

```
for(int i = 0; i<4; i++) {
    for(int j = 4; j>i; j--) {
        System.out.print("* ");
    }
    System.out.println();
}
```

```
* * * *
* * *
* *
*
```

The 1st time we loop, i is **0** – for(int j = 4; **j>i**; j- -) `* * * *`

The 2nd time we loop, i is **1** – for(int j = 4; **j>1**; j- -) `* * *`

The 3rd time we loop, i is **2** – for(int j = 4; **j>2**; j- -) `* *`

The 4th time we loop, i is **3** – for(int j = 4; **j>3**; j- -) `*`

QUESTION

- How could we put this program in a **method** that will allow a user to enter the number of rows and columns they want?

```
for(int i = 0; i<4; i++) {
    for(int j = 4; j>i; j--) {
        System.out.print("* ");
    }
    System.out.println();
}
```

```
* * * *
* * *
* *
*
```

QUESTION

```java
public class Main {

    public static void main(String[] args) {

        printPattern( n: 5);

    }
    public static void printPattern(int n) {
        for(int i = 0; i<n; i++) {
            for(int j = n; j>i; j--) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

Output

```
* * * * *
* * * *
* * *
* *
*
```

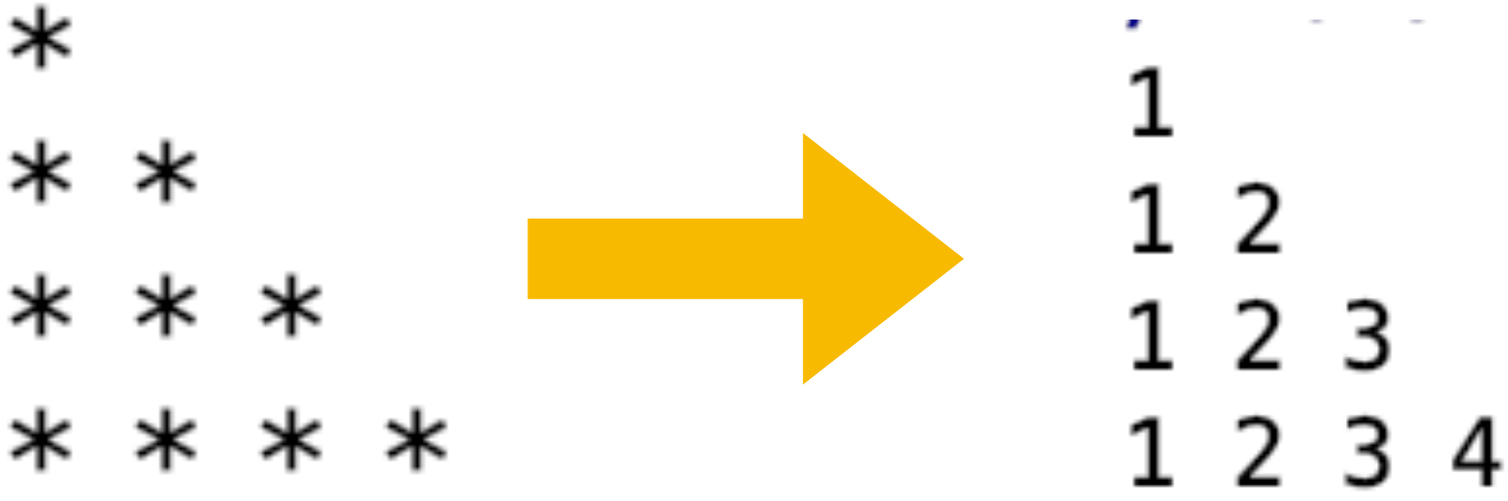
PRINTING NUMBERS

- How could we create the pattern below?
- What is the pattern? **Starts at 1 then increases by 1 every loop**

```
1
1 2
1 2 3
1 2 3 4
```

PRINTING NUMBERS

- We can use the same format as the first triangle pattern, ***except*** we will be printing one of the variables.




```
for(int i = 0; i<4; i++) {
    for(int j = 0; j<=i; j++) { // we have j<=i because the highest
        System.out.print("* "); // i can be is 3 -> (0, 1, 2, 3)
    }
    System.out.println();
}
```

PRINTING NUMBERS

- Instead of starting at 0, what should our loops start at? **1**
- Instead of printing out * below, what should we print to get the pattern? **j**

```
for(int i = 0; i<4; i++) {
    for(int j = 0; j<=i; j++) { // we have j<=i because the highest
        System.out.print("* "); // i can be is 3 -> (0, 1, 2, 3)
    }
    System.out.println();
}
```

```
1
1 2
1 2 3
1 2 3 4
```

PRINTING NUMBERS

- Start at the first number you want to print (**1** in this case, then print the value of **j**)

```
for(int i = 1; i<5; i++) {
    for(int j = 1; j<=i; j++) {
        System.out.print(j + " ");
    }
    System.out.println();
}
```

The 1st time we loop, i is **1** – for(int j = 1; **j<=1**; j- -) **1**

The 2nd time we loop, i is **2** – for(int j = 1; **j<=2**; j- -) **1 2**

The 3rd time we loop, i is **3** – for(int j = 1; **j<=3**; j- -) **1 2 3**

The 4th time we loop, i is **4** – for(int j = 1; **j<=4**; j- -) **1 2 3 4**

YOUR TURN

- Try to create the pattern below:
- What should j start at this time and keep looping until?

```
4  3  2  1
4  3  2
4  3
4
```

YOUR TURN

```
for(int i = 1; i<=4; i++) {
    for(int j = 4; j>=i; j--) {
        System.out.print(j + " ");
    }
    System.out.println();
}
```

```
4 3 2 1
4 3 2
4 3
4
```

The 1st time we loop, i is **1** – for(int j = 4; **j>=1**; j- -) **4 3 2 1**

The 2nd time we loop, i is **2** – for(int j = 4; **j>=2**; j- -) **4 3 2**

The 3rd time we loop, i is **3** – for(int j = 4; **j>=3**; j- -) **4 3**

The 4th time we loop, i is **4** – for(int j = 4; **j>=4**; j- -) **4**

MULTIPLICATION TABLE

- How could we create the table below? Let's start with the **body** of the table.
- What's the pattern? **The rows start from 1 to 9, columns go from 1 through 9 and are multiplied by whatever the row is (i)**

```
                Multiplication Table
          1    2    3    4    5    6    7    8    9
------------------------------------------------------
1   |     1    2    3    4    5    6    7    8    9
2   |     2    4    6    8   10   12   14   16   18
3   |     3    6    9   12   15   18   21   24   27
4   |     4    8   12   16   20   24   28   32   36
5   |     5   10   15   20   25   30   35   40   45
6   |     6   12   18   24   30   36   42   48   54
7   |     7   14   21   28   35   42   49   56   63
8   |     8   16   24   32   40   48   56   64   72
9   |     9   18   27   36   45   54   63   72   81
```

MULTIPLICATION TABLE

- Let's start with the rows:

```
for(int i = 1; i <= 9; i++) {

}
```

```
                Multiplication Table
           1   2   3   4   5   6   7   8   9
   ------------------------------------------
1  |       1   2   3   4   5   6   7   8   9
2  |       2   4   6   8  10  12  14  16  18
3  |       3   6   9  12  15  18  21  24  27
4  |       4   8  12  16  20  24  28  32  36
5  |       5  10  15  20  25  30  35  40  45
6  |       6  12  18  24  30  36  42  48  54
7  |       7  14  21  28  35  42  49  56  63
8  |       8  16  24  32  40  48  56  64  72
9  |       9  18  27  36  45  54  63  72  81
```

MULTIPLICATION TABLE

```
for(int i = 1; i <= 9; i++) {
    for(int j = 1; j <= 9; j++) {
        System.out.print((j * i) + " ");
    }
    System.out.println();
}
```

- Now the columns. We need to have j go from 1 through 9 and multiply by whatever i is.
- Remember i stays constant while in the inner loop, and j increases by 1.

```
                Multiplication Table
          1   2   3   4   5   6   7   8   9
------------------------------------------
1 |       1   2   3   4   5   6   7   8   9
2 |       2   4   6   8  10  12  14  16  18
3 |       3   6   9  12  15  18  21  24  27
4 |       4   8  12  16  20  24  28  32  36
5 |       5  10  15  20  25  30  35  40  45
6 |       6  12  18  24  30  36  42  48  54
7 |       7  14  21  28  35  42  49  56  63
8 |       8  16  24  32  40  48  56  64  72
9 |       9  18  27  36  45  54  63  72  81
```

MULTIPLICATION TABLE

- Now we can fix the format with printf:

```
                Multiplication Table
          1   2   3   4   5   6   7   8   9
------------------------------------------------
1 |       1   2   3   4   5   6   7   8   9
2 |       2   4   6   8  10  12  14  16  18
3 |       3   6   9  12  15  18  21  24  27
4 |       4   8  12  16  20  24  28  32  36
5 |       5  10  15  20  25  30  35  40  45
6 |       6  12  18  24  30  36  42  48  54
7 |       7  14  21  28  35  42  49  56  63
8 |       8  16  24  32  40  48  56  64  72
9 |       9  18  27  36  45  54  63  72  81
```

```
for(int i = 1; i <= 9; i++) {
    for(int j = 1; j <= 9; j++) {
        System.out.printf("%4d", j * i);
    }
    System.out.println();
}
```

MULTIPLICATION TABLE

```
                Multiplication Table
     1   2   3   4   5   6   7   8   9
-------------------------------------------
1 |  1   2   3   4   5   6   7   8   9
2 |  2   4   6   8  10  12  14  16  18
3 |  3   6   9  12  15  18  21  24  27
4 |  4   8  12  16  20  24  28  32  36
5 |  5  10  15  20  25  30  35  40  45
6 |  6  12  18  24  30  36  42  48  54
7 |  7  14  21  28  35  42  49  56  63
8 |  8  16  24  32  40  48  56  64  72
9 |  9  18  27  36  45  54  63  72  81
```

- We can replicate the table by using an extra loop above to print the row number. Printf can be utilized here as well.
- I wouldn't test you on making a perfectly formatted table, but I may want you to know how to do the body for the math with loops.

```java
//header
System.out.printf("%32s\n", "Multiplication Table");
System.out.print("    ");
for(int i = 1; i<=9; i++) {
    System.out.printf("%4d", i);
}
System.out.println("\n-----------------------------------");
//body
for(int i = 1; i <= 9; i++) {
    System.out.print(i + " | "); //added this for column number
    for(int j = 1; j <= 9; j++) {
        System.out.printf("%4d", j * i);
    }
    System.out.println();
}
```

YOUR TURN

- Replicate the **body** of the addition table below.

```
                Addition Table
      0   1   2   3   4   5   6   7   8   9  10
-------------------------------------------------
0   |   0   1   2   3   4   5   6   7   8   9  10
1   |   1   2   3   4   5   6   7   8   9  10  11
2   |   2   3   4   5   6   7   8   9  10  11  12
3   |   3   4   5   6   7   8   9  10  11  12  13
4   |   4   5   6   7   8   9  10  11  12  13  14
5   |   5   6   7   8   9  10  11  12  13  14  15
6   |   6   7   8   9  10  11  12  13  14  15  16
7   |   7   8   9  10  11  12  13  14  15  16  17
8   |   8   9  10  11  12  13  14  15  16  17  18
9   |   9  10  11  12  13  14  15  16  17  18  19
10  |  10  11  12  13  14  15  16  17  18  19  20
```

YOUR TURN

- Replicate the **body** of the addition table below.

```
                Addition Table
       0   1   2   3   4   5   6   7   8   9  10
-------------------------------------------------
0  |    0   1   2   3   4   5   6   7   8   9  10
1  |    1   2   3   4   5   6   7   8   9  10  11
2  |    2   3   4   5   6   7   8   9  10  11  12
3  |    3   4   5   6   7   8   9  10  11  12  13
4  |    4   5   6   7   8   9  10  11  12  13  14
5  |    5   6   7   8   9  10  11  12  13  14  15
6  |    6   7   8   9  10  11  12  13  14  15  16
7  |    7   8   9  10  11  12  13  14  15  16  17
8  |    8   9  10  11  12  13  14  15  16  17  18
9  |    9  10  11  12  13  14  15  16  17  18  19
10 |   10  11  12  13  14  15  16  17  18  19  20
```

```
System.out.printf("%32s\n","Addition Table");
System.out.print("     ");
//header
for(int i = 0; i<=10; i++) {
    System.out.printf("%4d", i);
}
System.out.println("\n-----------------------------------------");
//body
for(int i = 0; i <= 10; i++) {
    System.out.printf("%-2d%s", i, " | ");
    for(int j = 0; j<=10; j++) {
        System.out.printf("%4d", i+j);
    }
    System.out.println();
}
```