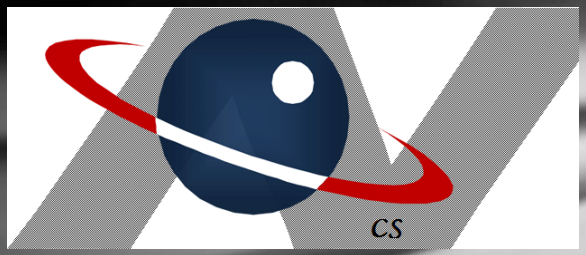
O(N) CS LESSONS

Lesson 6A – Loops



By John B. Owen

Topic List

- Objectives
- Loop structure – 4 parts
- Three loop styles
- Example of a while loop
- Example of a do while loop
- Comparison – while vs do while
- Example of a for loop
- Three loops…compare and contrast
- Lesson Summary / Labs
- Contact Information for supplementary materials

Objectives

- In this lesson you will learn about using a loop.
- This technique is very powerful and useful in many ways.
- Sometimes it is called “iteration”, another name for **looping**.

Loop structure

- A loop is a powerful control structure in programming that allows a task to be repeated several times, such as outputting a row of 20 stars, or counting from 1 to 20, or inputting from the keyboard 5 numbers…the possibilities are many.

Four parts of a loop

Every loop has four basic parts that MUST be present in order for the loop to work properly and effectively.

- START
- CHECK
- ACTION
- STEP

The START

- This is where the loop begins its process, and only happens ONCE.
- The process is often managed by a **loop control variable**, most often an integer that begins at zero.
- `int x = 0;` is a very common start to a loop

The CHECK

This is a boolean expression which, while its condition is TRUE, allows the loop process to continue, but stops the process when the condition becomes FALSE.

An example of a "check" **expression** would be `x<10`.

The CHECK

- If the current value of **x** is less than 10, the loop continues.
- When the value of **x** becomes equal to or greater than 10, the loop will stop.

The ACTION

This is the task to be repeated inside the loop, and can be any number of things, for example:

- Output or calculate something
- Read data from the keyboard or a file, or write data to a file
- Animation control

The STEP

- The STEP is generally some incremental process that gradually moves the loop towards its conclusion, finally reaching a FALSE condition for the check, thus stopping the process and continuing with the rest of the program.

The STEP

- If this does not happen, a FALSE check is never reached and is always TRUE, and the loop never stops.
- This is called an **INFINITE LOOP**, not a good thing!

The STEP

The Step is most often a mathematical process of adding to or subtracting from the value of a **loop control** variable, which eventually (hopefully) will reach a value that will cause the CHECK condition to be FALSE, causing the loop to stop.

The STEP

`x=x+1` is a very common "step" which increases the value of x by 1…

`x+=1` is a shortcut for this…

as is `x++`

Three styles of loops

In Java there are three primary styles of loops:

- ***for***
- ***while***
- ***do while***

All three work in similar ways, but have slight differences.

Newer loop style

There is a fourth style of loop that has emerged in the later versions of Java called the for each loop.

This loop style is used primarily to process collections of data, such as arrays, ArrayLists, etc., and will be discussed at a later time.

Three styles of loops

- Quite likely the most commonly used structure is the “for” loop, but the “while” and “do while” loop structures are sometimes better suited for use in some cases.

Three styles of loops

- For now equal attention will be given to each style of loop, and as you gain experience in using each, and insight into the particular characteristics of each, you will become better prepared to decide which is best for any particular situation.

The while loop

Here is an example of a "while" loop that will output the word "HELLO" five times, along with the current value of x, the loop control variable:

whileLoopExample.java

```
1   import static java.lang.System.*;
2   public class whileLoopExample
3   {
4      public static void main (String [] args)
5      {
6        //start loop control variable at 1
7        int x = 1;
8        //check if condition is true…if so continue,
9        //otherwise stop the loop action and fall through
10       //to the next program statement
11       while (x<=5)
12       {
13           out.println("HELLO "+x); //action
14           x=x+1;  //step…increase x adding 1
15       }
16       //first statement following the loop
17       out.println("Loop is finished...x = "+x);
18     }
19  }
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

The while loop

- Notice that the loop control variable **x**, is given a starting value of 1.
- When the condition of the boolean expression (x<=5) is first ***checked***, it is TRUE, because **x** currently is storing the value 1, which is indeed less than or equal to 5.

whileLoopExample.java

```
1   import static java.lan
2   public class whileLoop
3   {
4       public static void
5       {
6         //start loop control variable at 1
7         int x = 1;
8         //check if condition is true…if so continue,
9         //otherwise stop the loop action and fall through
10        //to the next program statement
11        while (x<=5)
12        {
13            out.println("HELLO "+x); //action
14            x=x+1;  //step…increase x adding 1
15        }
16        //first statement following the loop
17        out.println("Loop is finished...x = "+x);
18      }
19  }
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

The while loop

Since the boolean condition is TRUE, the inside block of the loop is executed, containing the ACTION and the STEP. "`HELLO 1`" is output, then **x**'s value is increased by 1, becoming 2.

whileLoopExample.java

```
1   import static java.lang.System.*;
2   public class whileLoopExample
3   {
4       public static void main (String [] args)
5       {
6         //start loop control variable at 1
7         int x = 1;
8         //check if condition is true…if so continue,
9         //otherwise stop the loop action and fall through
10        //to the next program statement
11        while (x<=5)
12        {
13            out.println("HELLO "+x); //action
14            x=x+1;  //step…increase x adding 1
15        }
16        //first statement following the loop
17        out.println("Loop is finished...x = "+x);
18      }
19  }
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

The while loop

When the closing brace is reached, the computer goes back to the "while" and re-checks the condition.

Since x contains the value 2, the condition is still TRUE, and the ACTION and STEP execute again.

whileLoopExample.java

```
1   import static java.lan
2   public class whileLoop
3   {
4      public static void main (String [] args)
5      {
6        //start loop control variable at 1
7        int x = 1;
8        //check if condition is true…if so continue,
9        //otherwise stop the loop action and fall through
10       //to the next program statement
11       while (x<=5)
12       {
13          out.println("HELLO "+x); //action
14          x=x+1;  //step…increase x adding 1
15       }
16       //first statement following the loop
17       out.println("Loop is finished...x = "+x);
18     }
19  }
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

The while loop

This repetitive cycle continues until x contains the value 6, which causes **(x<=5)** to be FALSE, thus stopping the loop process.

The program control then falls through to the next command following the loop.

whileLoopExample.java

```
1   import static java.lan
2   public class whileLoop
3   {
4       public static void main (String [] args)
5       {
6         //start loop control variable at 1
7         int x = 1;
8         //check if condition is true…if so continue,
9         //otherwise stop the loop action and fall through
10        //to the next program statement
11        while (x<=5)
12        {
13            out.println("HELLO "+x); //action
14            x=x+1;  //step…increase x adding 1
15        }
16        //first statement following the loop
17        out.println("Loop is finished...x = "+x);
18      }
19  }
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

The while loop

If you trace the values of x, they start at 1, then become 2, 3, 4, 5, and finally 6. When the values were 1 through 5, the CHECK condition was true, and the ACTION and STEP inside the loop were executed, causing five outputs.

Sometimes it helps to do a tracing chart to see how the loop works, like this one:

x	x<=5	action
1	true	"HELLO 1"
2	true	"HELLO 2"
3	true	"HELLO 3"
4	true	"HELLO 4"
5	true	"HELLO 5"
6	false	loop ends

The do while loop

- Here is an example of a "do while" loop that does exactly the same thing as the while loop we just did.
- The "do while" loop performs the same task, but in a slightly different way.
- Can you tell the difference?

dowhileLoopExample.java

```
1   import static java.lang.System.*;
2   public class dowhileLoopExample
3   {
4      public static void main (String [] args)
5      {
6        //start loop control variable at 1
7        int x = 1;
8        do
9        {
10           out.println("HELLO "+x); //action
11           x=x+1;  //step…increase x adding 1
12       }
13       //check condition is at the end!
14       while (x<=5);
15       //first statement following the loop
16       out.println("Loop is finished...x = "+x);
17     }
18  }
19
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

The do while loop

- The first noticeable change is the word "do" at the beginning of the structure.
- The next major difference is that the **CHECK** happens at the end of the structure, AFTER the action and the step have occurred.
- Finally, the while statement finishes with a semi-colon, where in the first example there was NO semi-colon after the while statement.

dowhileLoopExample.java

```
1   import static java.lang.System.*;
2   public class dowhileLoopExample
3   {
4      public static void main (String [] args)
5      {
6        //start loop control variable at 1
7        int x = 1;
8        do
9        {
10           out.println("HELLO "+x); //action
11           x=x+1;  //step…increase x adding 1
12       }
13       //check condition is at the end!
14       while (x<=5);
15       //first statement following the loop
16       out.println("Loop is finished...x = "+x);
17     }
18  }
19
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

The do while loop

- The most significant difference between the "while" and the "do while" loop is that the "do while" loop will ALWAYS execute at least once since the CHECK happens at the end.

dowhileLoopExample.java

```
1   import static java.lang.System.*;
2   public class dowhileLoopExample
3   {
4      public static void main (String [] args)
5      {
6        //start loop control variable at 1
7        int x = 1;
8        do
9        {
10           out.println("HELLO "+x); //action
11           x=x+1;  //step…increase x adding 1
12       }
13       //check condition is at the end!
14       while (x<=5);
15       //first statement following the loop
16       out.println("Loop is finished...x = "+x);
17     }
18  }
19
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

while vs do while – a comparison

- To demonstrate the difference between the "while" and the "do while" loops, the two previous program examples have been altered, with the value of x starting at 6. Check out the results!

```
  //start loop control variable at 6
  int x = 6;
  //check if condition is true…if so continue,
  //otherwise stop the loop action and fall through
  //to the next program statement
  while (x<=5)
  {
      out.println("HELLO "+x); //action
      x=x+1;  //step…increase x adding 1
  }
  //first statement following the loop
  out.println("Loop is finished...x = "+x);
}
```

```
C:\Program Files (x86)\Xinox Software\J...
Loop is finished...x = 6
Press any key to continue...
```

```
//start loop control variable at 6
int x = 6;
do
{
    out.println("HELLO "+x); //action
    x=x+1;  //step…increase x adding 1
}
//check condition is at the end!
while (x<=5);
//first statement following the loop
out.println("Loop is finished...x = "+x);
```

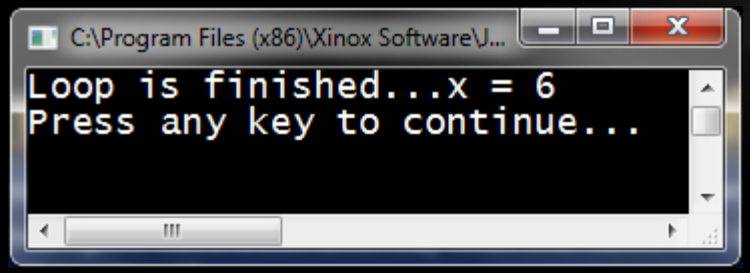
```
C:\Program Files (x86)\Xinox Software\J...
HELLO 6
Loop is finished...x = 7
Press any key to continue...
```

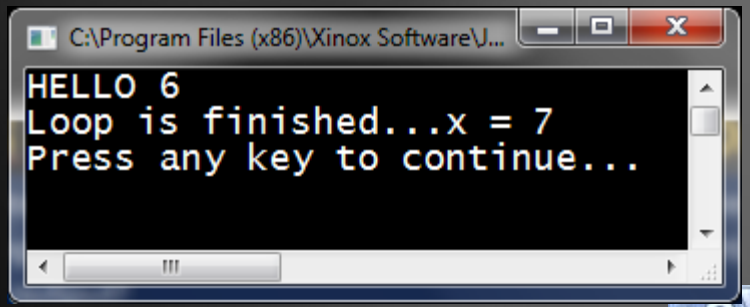
while vs do while – a comparison

- Notice carefully that the body of the **while** loop never executes since the first CHECK is false due to the fact that x has an initial value of 6.
- However, the body of the **do while** executes once with the x value of 6 since the CHECK does not happen until the end of the structure.

```
{
  //start loop control variable at 6
  int x = 6;
  //check if condition is true…if so continue,
  //otherwise stop the loop action and fall through
  //to the next program statement
  while (x<=5)
  {
      out.println("HELLO "+x); //action
      x=x+1;  //step…increase x adding 1
  }
  //first statement following the loop
  out.println("Loop is finished...x = "+x);
}
```

```
//start loop control variable at 6
int x = 6;
do
{
    out.println("HELLO "+x); //action
    x=x+1;  //step…increase x adding 1
}
//check condition is at the end!
while (x<=5);
//first statement following the loop
out.println("Loop is finished...x = "+x);
```





The for loop

- The for loop is clearly the most elegant of the three loop structures, in that it handles three of the four parts of the loop **all in one compound statement**, as you can see in the example below.
- It follows the same general pattern as the while loop:
  - Start, then check, action if true, then step

```
1   import static java.lang.System.*;
2   public class forLoopExample
3   {
4      public static void main (String [] args)
5      {
6        //loop control variable declared, but not initialized
7        int x;
8        //start,check, and step all encompassed in one compound statement
9        for(x=1;x<=5;x++)
10           out.println("HELLO "+x); //action
11
12       //first statement following the loop
13       out.println("Loop is finished...x = "+x);
14     }
15  }
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

The for loop

- It continues the three part circular action pattern of **check, action, step** until the check is FALSE, at which time the loop stops and program control falls through to the next statement.

```
//loop control variable declared, but not in
int x;
//start,check, and step all encompassed in
for(x=1;x<=5;x++)
    out.println("HELLO "+x); //action

//first statement following the loop
out.println("Loop is finished...x = "+x);
```

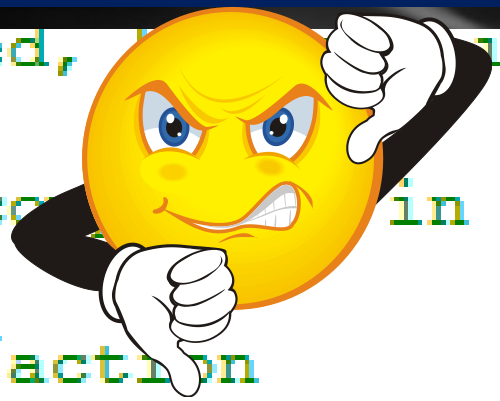
```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

The for loop

- Notice very carefully that ***THERE IS NO SEMICOLON*** at the end of the for statement
- Putting a semi-colon here is a common error among beginners…***don't do it!!!***

```
//loop control variable declared,
int x;
//start,check, and step all enco          in
for(x=1;x<=5;x++)
    out.println("HELLO "+x); //action

//first statement following the loop
out.println("Loop is finished...x = "+x);
```



```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...x = 6
Press any key to continue...
```

The for loop – internal LCV

- Another way to structure the for loop is to declare the loop control variable ***inside*** the for statement, as shown below.

forLoopExample.java

```
1   import static java.lang.System.*;
2   public class forLoopExample
3   {
4      public static void main (String [] args)
5      {
6        //start,check, and step all encompassed in
7        for(int x=1;x<=5;x++)
8            out.println("HELLO "+x); //action
9
10       //first statement following the loop
11       out.println("Loop is finished...");
12     }
13  }
14
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...
Press any key to continue..
```

The for loop – internal LCV

- There is one subtle difference in the output. Can you see it?

forLoopExample.java

```
1   import static java.lang.System.*;
2   public class forLoopExample
3   {
4      public static void main (String [] args)
5      {
6        //start,check, and step all encompassed in
7        for(int x=1;x<=5;x++)
8            out.println("HELLO "+x); //action
9
10       //first statement following the loop
11       out.println("Loop is finished...");
12     }
13  }
14
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...
Press any key to continue..
```

The for loop – internal LCV

- The value of x is omitted from the "Loop is finished…" statement!!

forLoopExample.java

```
1   import static java.lang.System.*;
2   public class forLoopExample
3   {
4      public static void main (String [] args)
5      {
6        //start,check, and step all encompassed in
7        for(int x=1;x<=5;x++)
8            out.println("HELLO "+x); //action
9
10       //first statement following the loop
11       out.println("Loop is finished...");
12     }
13  }
14
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...
Press any key to continue..
```

The for loop – internal LCV

- Since the loop control variable x is internal (was declared inside the loop structure), it only EXISTS inside the loop structure, therefore is not available for any statements outside the loop.

forLoopExample.java

```
1    import static java.lang.System.*;
2    public class forLoopExample
3    {
4       public static void main (String [] args)
5       {
6         //start,check, and step all encompassed in
7         for(int x=1;x<=5;x++)
8             out.println("HELLO "+x); //action
9
10        //first statement following the loop
11        out.println("Loop is finished...");
12      }
13   }
14
```

```
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO 5
Loop is finished...
Press any key to continue..
```

The for loop – internal LCV

- Any attempt to access the variable x outside the loop structure results in a compile error, as you can see below.

forLoopExample.java

```
1   import static java.lang.System.*;
2   public class forLoopExample
3   {
4      public static void main (String [] args)
5      {
6        //start,check, and step all encompassed in one compound st
7        for(int x=1;x<=5;x++)
8            out.println("HELLO "+x); //action
9
10       //first statement following the loop
11       out.println("Loop is finished...x = "+x);
12     }
```

Build Output

```
--------------------Configuration: <Default>--------------------
C:\Users\John\Desktop\OWEN CS Lessons\javaFiles\LessonFiles\forLoopExample.java:11: err
        out.println("Loop is finished...x = "+x);
                                            ^
  symbol:   variable x
  location: class forLoopExample
1 error

Process completed.
```

Three loops…compare and contrast

Clearly all three loops accomplish the main goal – the "iteration" or "repetition" of program statements. They just do it in slightly different ways.

Some key differences among the three loops are:

- It is possible for the "while" loop and the "for" loop NEVER to execute. This happens when the FIRST CHECK of the boolean expression evaluates to FALSE. These are called PRETEST loops.

Three loops…compare and contrast

- On the other hand, the "do while" loop ALWAYS executes at least once since the FIRST CHECK happens only ***after*** the action of the loop is executed. It is called a POSTTEST loop.
- Another difference between the loops is that most of the time, both the "while" and "do while" loops require more than one statement in the body of the loop (ACTION and STEP), which means they must be included in a "block"…between the "{" and "}".

Three loops…compare and contrast

- Since the "for" loop encompasses the START, CHECK, and STEP inside the loop structure, and often only has one statement in the ACTION step, it does not need the braces for just one statement.
- However, the "for" loop CAN have multiple action statements, thus requiring the use of block brackets.

Lesson Summary

- In this lesson, you learned about the three basic loop structures: while, do while, and for.
- You learned how each loop has four parts, arranged in different ways..start, check, action, step.
- Now it is time to practice several examples.

Labs

- All the labs for this lesson will be created in a class called MyLoops.
- Create a folder to contain all of these labs, and then create a class called MyLoops to contain all of the static methods you will create.

Labs

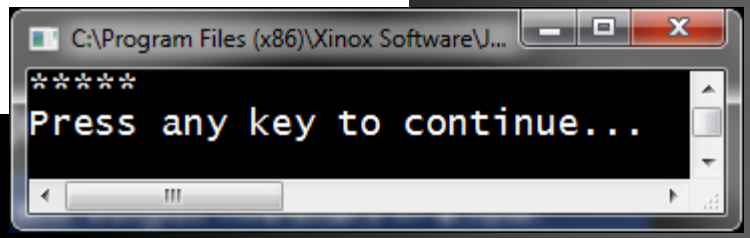
- The difference between these methods and the ones you did in the last lesson is that these will be ***void methods***, not return methods.
- Void methods simply DO something inside the method, instead of returning an answer.
- Some examples will be shown to help you get started.

Method 1 – *row5Stars*

MyLoops.java

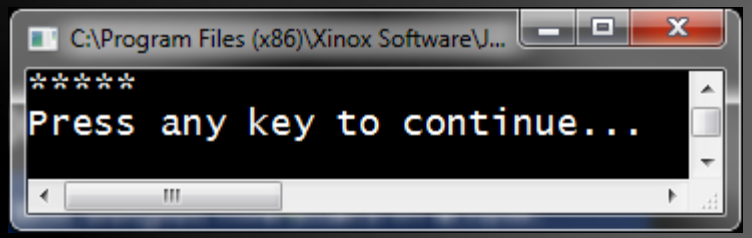
```
1   import static java.lang.System.*;
2   /**Loops class for Lesson 6A, by John Owen
3    */
4   public class MyLoops
5   {
6      /**This method outputs five stars in a row
7       *using a while loop.
8       */
9      public static void row5Stars()
10     {
11       int x=1;
12       while(x<=5)
13       {
14           out.print("*");
15           x++;
16       }
17       out.println();
18     }
19     public static void main (String [] args)
20     {
21       row5Stars();
22     }
23  }
24
```

- WAM (write a method) that will output five stars in a row using a ***while loop***.
- There is no input for this method…just simple output.
- Notice the way the method is called…this is how you call a void method…simply state its name.



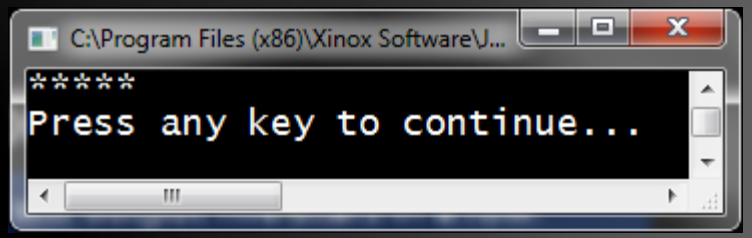
Method 2 – ***row5Stars2***

- WAM that will output five stars in a row using a ***do while loop***. Copy and paste the row5Stars() method and make the adjustments needed to make it a do while loop.



Method 3 – ***row5Stars3***

- WAM that will output five stars in a row using a ***for loop***. Again, copy and paste and make the adjustments.



Method 4 – *rowNStars*

- WAM that will receive an integer parameter N and output N stars in a row using a ***while loop***.
- Use a while loop in the main method to input values from a data file.
- A portion of the solution is shown.

MyLoops.java | MyLoops4.in

```
21      /**This method receives an integer parameter N and
22       *outputs N stars in a row
23       *using a while loop.
24       */
25      public static void rowNStars(int N)
26      {
27        int x=1;
28        while(x<=
29        {
30
31
32        }
33        out.println();
34      }
35      public static void main (String [] args)
36        throws IOException
37      {
38        Scanner f = new Scanner(new File("MyLoops4.in"));
39        while (f.hasNext())
40        {
41            int n = f.nextInt();
42            rowNStars(n);
43        }
44      }
45  }
```

MyLoops.java | MyLoops4.in

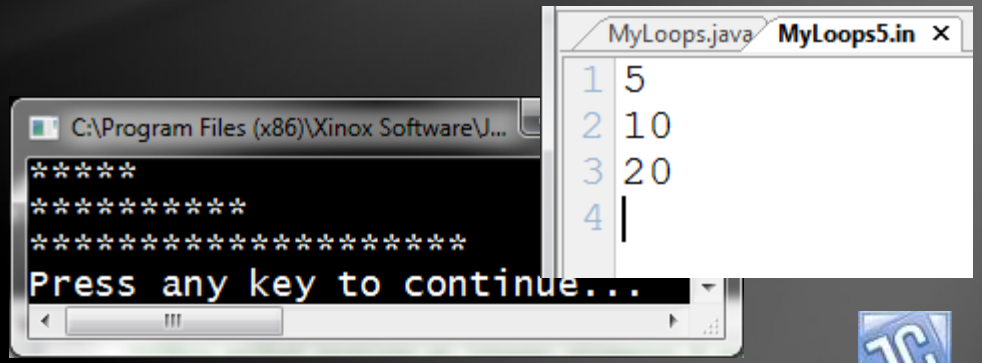
```
1  5
2  10
3  20
4
```

C:\Program Files (x86)\Xinox Software\J...

```
*****
**********
**********
Press any key to continue...
```

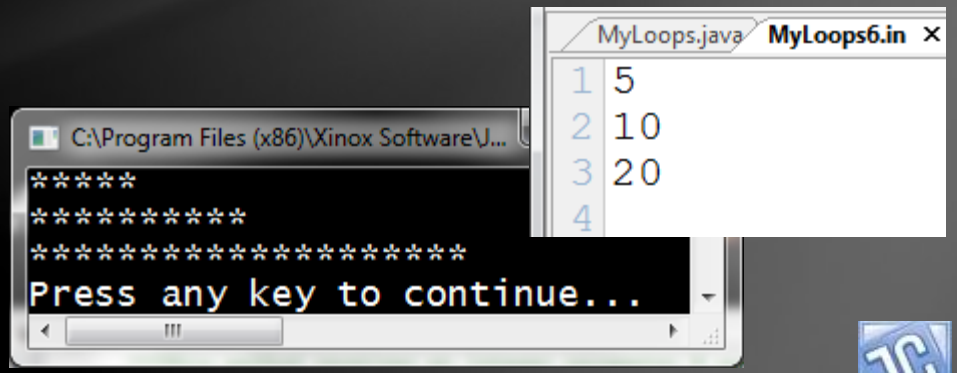
Method 5 – ***rowNStars2***

- WAM that will receive an integer parameter N and output N stars in a row using a ***do while loop***.



Method 6 – ***rowNStars3***

- WAM that will receive an integer parameter N and output N stars in a row using a ***for loop***.



Method 7 – *colNStars*

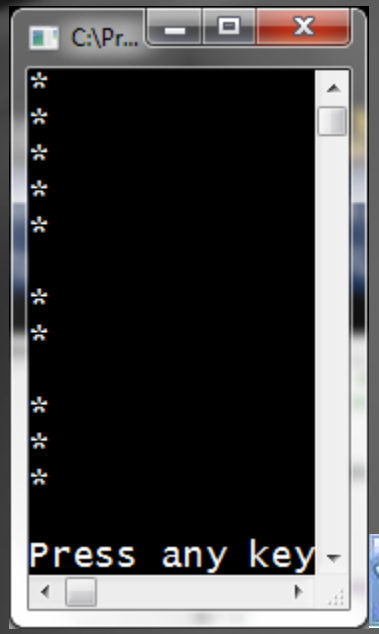
- WAM that will receive an integer parameter N and output N stars in a ***COLUMN*** using a ***while loop***.

MyLoops.java | MyLoops7.in

```
35      /**This method receives an integer parameter N and
36       *outputs N stars in a column
37       *using a while loop.
38       */
39      public static void colNStars(int N)
40      {
41        int x=1;
42        while(x<=
43        {
44
45
46        }
47        out.println();
48      }
49      public static void main (String [] args)
50        throws IOException
51      {
52        Scanner f = new Scanner(new File("MyLoops7.in"));
53        while (f.hasNext())
54        {
55            int n = f.nextInt();
56            colNStars(n);
57        }
58      }
59  }
```

MyLoops.java | MyLoops7.in

```
1 5
2 2
3 3
4
```



Method 8 – ***colNValues***

- WAM that will receive an integer parameter N and output the ***VALUES FROM 1 TO N*** in a ***COLUMN*** using a ***do while loop***.

MyLoops.java | MyLoops8.in

```
1 5
2 2
3 3
4
```

```
1
2
3
4
5

1
2

1
2
3

Press any key
```

Method 9 –*MtoNValues*

- WAM that will receive two integer parameters M and N and output the ***VALUES FROM M TO N*** in a ***ROW*** using a ***for loop***. There should be exactly one space after each value.

MyLoops.java | MyLoops9.in

```
1 5 10
2 2 7
3 3 21
4
```

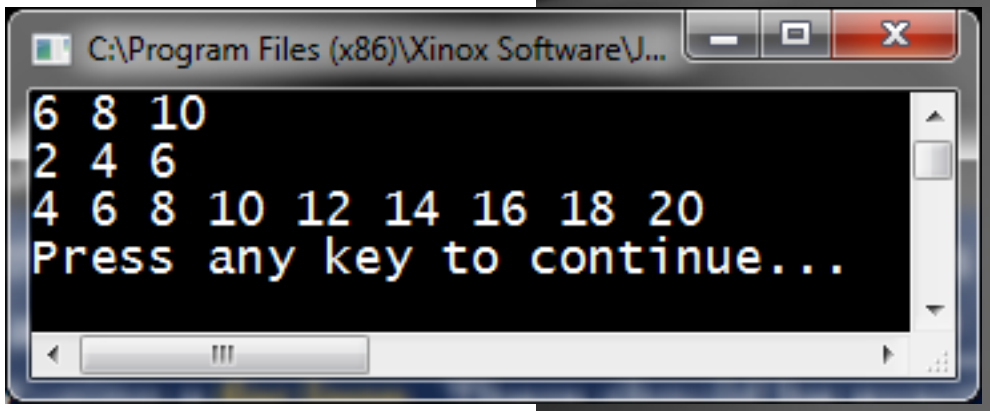
```
C:\Program Files (x86)\Xinox Software\JCreatorV5LE\GE2001.exe
5 6 7 8 9 10
2 3 4 5 6 7
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Press any key to continue...
```

Method 10 – ***MtoNEvens***

- WAM that will receive two integer parameters M and N and output the ***EVEN VALUES FROM M TO N*** in a ***ROW*** using a ***while loop***. There should be exactly one space after each value. *Hint: use an if statement with the mod (%) operation.*

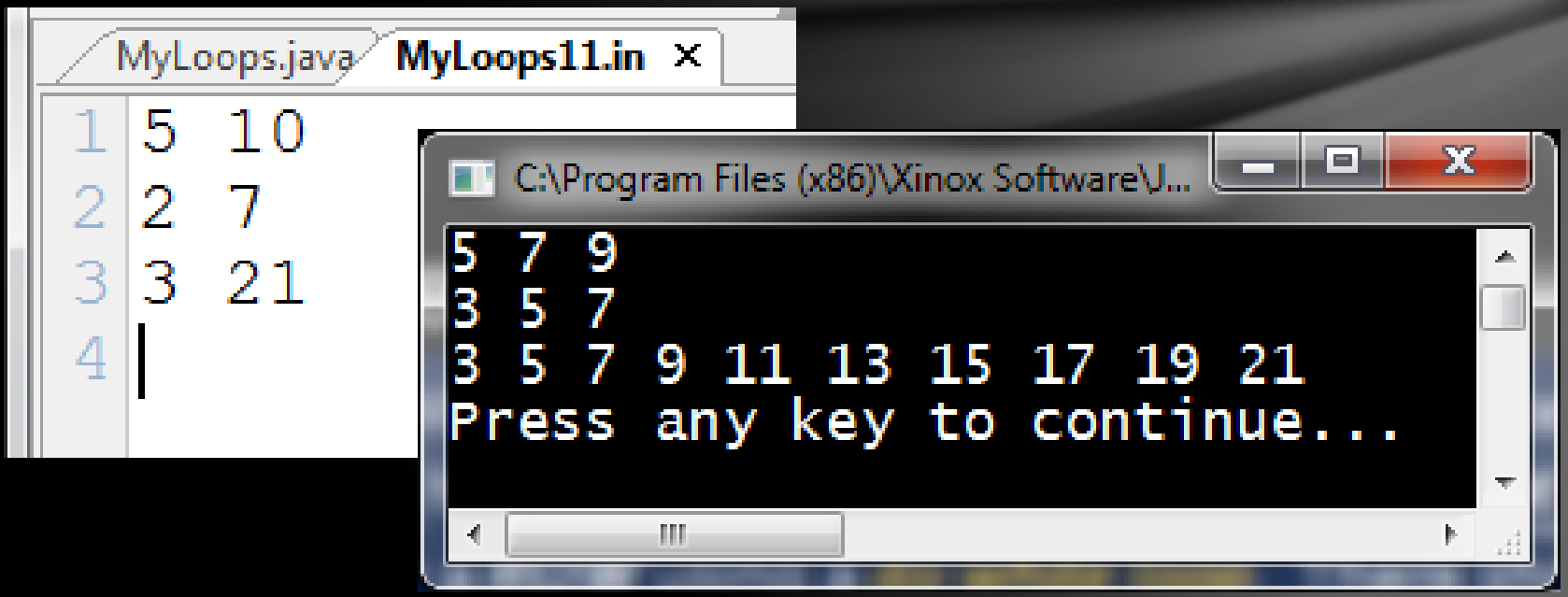
MyLoops.java | MyLoops10.in

```
1  5  10
2  2  7
3  3  21
4
```



Method 11 – *MtoNOdds*

- WAM that will receive two integer parameters M and N and output the ***ODD VALUES FROM M TO N*** in a ***ROW*** using a ***do while loop***. There should be exactly one space after each value.

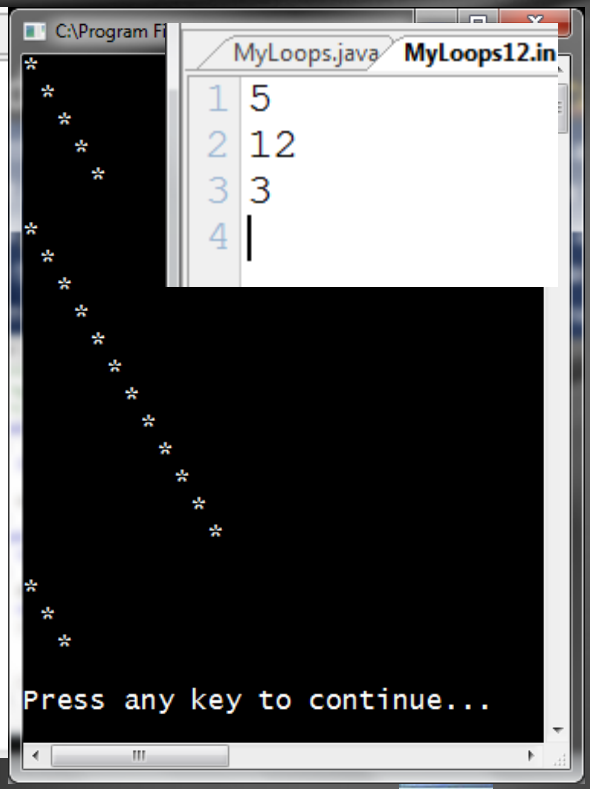


Method 12 – *diagonalNStars*

- WAM that will receive an integer parameter N and output a diagonal of stars using a ***for loop***. Use the special ***printf*** technique you learned in Lesson 1D where the field width specifier helps to create a custom indent. The complete solution is shown. An explanation is provided on the next slide

MyLoops.java | MyLoops12.in

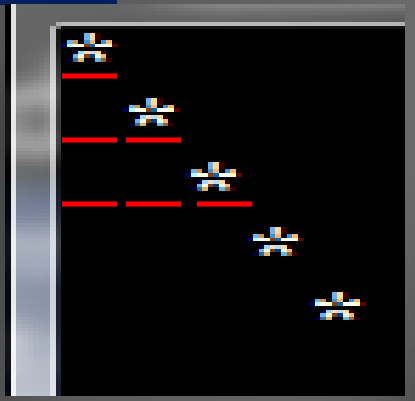
```
62        /**This method receives an integer parameters N
63         *and outputs an N size diagonal.
64         */
65        public static void diagonalNStars(int N)
66        {
67          for(int x=1;x<=N;x++)
68              out.printf("%"+x+"s\n","*");
69          out.println();
70        }
71        public static void main (String [] args)
72          throws IOException
73        {
74          Scanner f = new Scanner(new File("MyLoops12.in"));
75          while (f.hasNext())
76          {
77              int n = f.nextInt();
78              diagonalNStars(n);
79          }
80        }
81    }
```



Method 12 – *diagonalNStars*

- **Study carefully the indenting technique highlighted below.**
- The format string is a concatenation of the ***%s format specifier*** using the changing value of x as the indenting value, which creates the "diagonal" effect required of this method.
- When the value of x is 1, the field width is 1, and the star is placed in that field width, right on the left margin.
- When x increases in value, the field width also increases, and the star is placed at the end of each larger field width, thus creating the diagonal effect.

```
{
   for(int x=1;x<=N;x++)
        out.printf("%"+x+"s\n","*");
   out.println();
}
```



Method 13 – ***reverseDiagonalNStars***

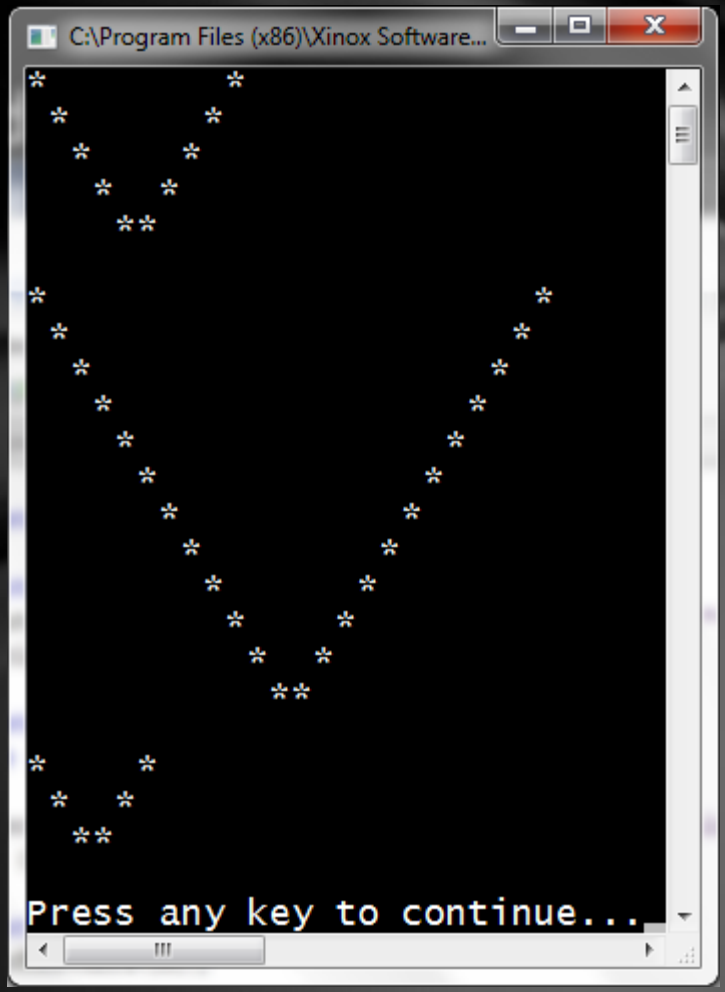
- WAM that will receive an integer parameter N and output a REVERSE diagonal of stars. Think about it, and good luck! *Hint: Reverse the loop.*

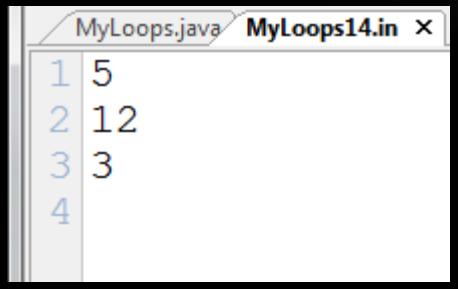
MyLoops.java | MyLoops13.in

```
1 5
2 12
3 3
4
```

Method 14 – *VofNStars*

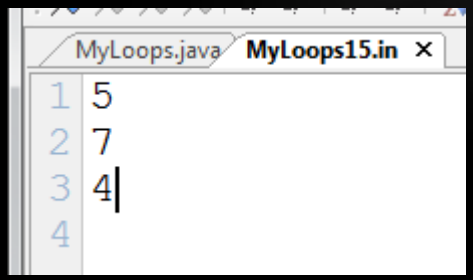
- WAM that will receive an integer parameter N and output V shape of N stars.
- *This is a kind of "rite of passage" for beginning programmers. It is a tough method, but if you think about it long enough, you'll make it work.*
- *Here's a hint: Use one forward loop with two outputs inside the loop.*

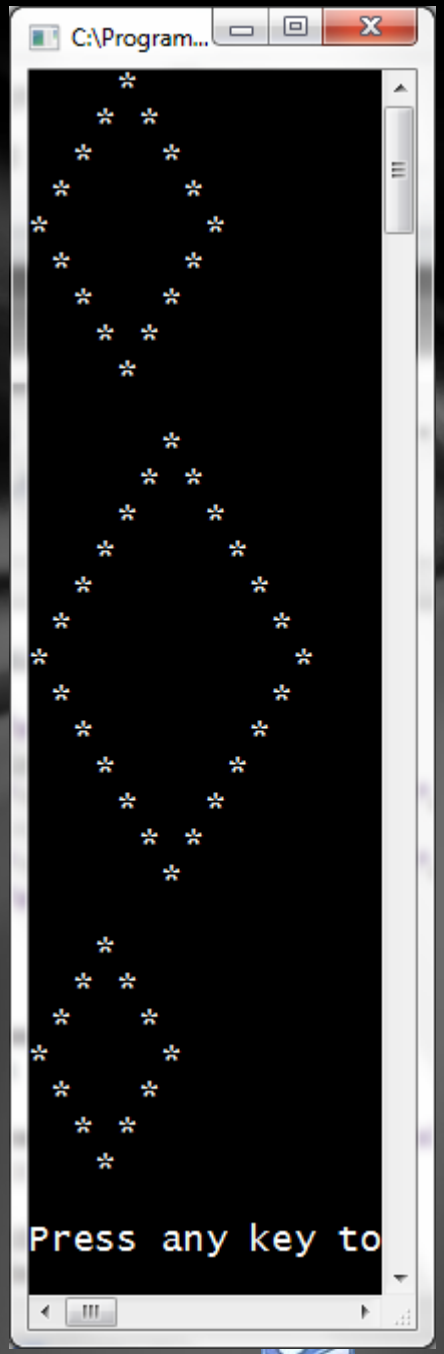




Method 15 – *diamondNStars*

- (BONUS) WAM that will receive an integer parameter N and output a diamond shape of N stars.
- *Notice carefully that the top, bottom and sides of the diamond come to a single point.*





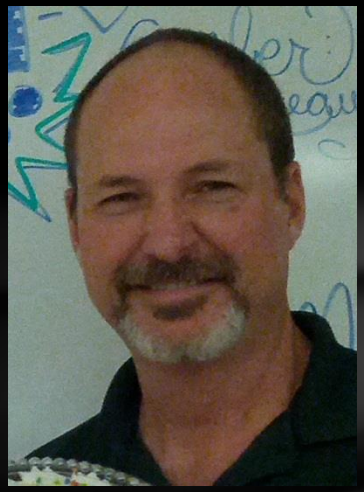
JavaDoc

- As you did in the last lesson, when you finish this lab series, complete the documentation for all of the methods, run the JavaDoc utility
- Make adjustments as you see fit to make sure the help file is indeed helpful.

CONGRATULATIONS!

- You have survived the first lesson on loops, probably the toughest lesson and lab series you have had so far.
- *The Lesson 6B will tell you more about loops that count and accumulate values.*

Thank you, and have fun!



To order supplementary materials for all the lessons in this package, such as lab solutions, quizzes, tests, and unit reviews, visit the O(N)CS Lessons website, or contact me at

John B. Owen
captainjbo@gmail.com