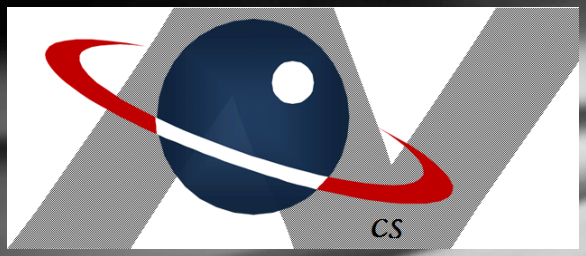
O(N) CS LESSONS



Lesson 6C – Loops 3

Advanced File Processing

By John B. Owen

All rights reserved

©2011, revised 2014

Table of Contents

Objectives

- In the previous lesson, you learned ways to use loops in problem solving that require repetitive actions, such as counting and accumulating.
- In this lesson you will learn about more file processing techniques in addition to what you have been using.

Objectives

- Finally, you will be given more practice in using the file input processes you have learned in this lesson.

File input setup - reminder

As we have learned before, the setup steps required for file input are listed below:

- import java.io.*; //package that contains the File class
- import java.util.*; // package that contains the Scanner class
- throws IOException //*added to the main method header*
- Scanner f = new Scanner (new File("fileName.in")); //links program file to data file

Two file input scenarios

An input text file can be arranged in several different ways. Two ways we will explore are:

- File Scenario #1 - An unknown quantity (one or more) of data items arranged vertically or horizontally (what we have been doing so far)
- File Scenario #2 - A single value N at the top of the file, followed by N sets of data, usually arranged vertically

File input scenario #1 – Unknown quantity of values

- This first example is the one we have been using for a while - reading an unknown quantity of values input from a data file using the ***while(f.hasNext())*** process. We will now study in detail how it actually works.

File input scenario #1 – Unknown quantity of values

- To make it a bit more interesting, we'll test these values to see if each is a prime number, using a method we will design called ***isPrime***.
- This method requires the use of a loop to test for the factors of a number.
- See the example on the next slide.

File input scenario #1 – Unknown quantity of values

ops3.java × MyLoops3.in

```
 */
public class MyLoops3
{
  /**This method returns a boolean value, true indicat
   *integer parameter is a prime number, and false that
    */
   public static boolean isPrime(int X)
   {
     int i=2;
     while(i<X)
     {
         if(X%i==0)
             return false;
         i++;
     }
     return true;
   }
   public static void main (String [] args)
     throws IOException
   {
     Scanner f = new Scanner(new File("MyLoops3.in"));
     while (f.hasNext())
     {
         int x = f.nextInt();
         out.printf("It is %s that %d is a prime number.\n",
             isPrime(x),x);
     }
   }
```

- This is the file scenario we have been using exclusively in lessons so far: an unknown quantity of values, usually arranged vertically in the data file.
- A ***while(f.hasNext())*** loop is required to process all of these.

MyLoops3.java | MyLoops3.in ×

```
1 2
2 5
3 9
4 17
5 23
6 45
7 97
8
```

File input scenario #1 – Unknown quantity of values

ops3.java × MyLoops3.in

```
 */
public class MyLoops3
{
  /**This method returns a boolean value, true indicat
   *integer parameter is a prime number, and false tha
    */
   public static boolean isPrime(int X)
   {
     int i=2;
     while(i<X)
     {
         if(X%i==0)
             return false;
         i++;
     }
     return true;
   }
   public static void main (String [] args)
     throws IOException
   {
     Scanner f = new Scanner(new File("MyLoops3.in"));
     while (f.hasNext())
     {
         int x = f.nextInt();
         out.printf("It is %s that %d is a prime number.\n",
             isPrime(x),x);
     }
   }
```

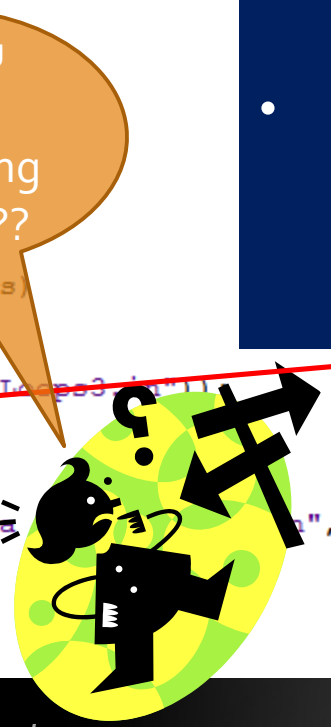
- The ***hasNext()*** method belongs to the Scanner object ***f*** and simply returns a boolean value, indicating whether or not there is a "next" item in the data file.
- Let's discuss how this works…

MyLoops3.java MyLoops3.in ×

```
1 2
2 5
3 9
4 17
5 23
6 45
7 97
8
```

File input scenario #1 – Unknown quantity of values

```
ps3.java ×  MyLoops3.in
 */
public class MyLoops3
{
  /**This method returns a boolean value, true indicat
   *integer parameter is a prime number, and false tha
    */
   public static boolean isPrime(int X)
   {
     int i=2;
     while(i<X)
     {
          if(X%i==0)
              return false;
          i++;
     }
     return true;
   }
   public static void main (String [] a
     throws IOException
   {
     Scanner f = new Scanner(new File("My
     while (f.hasNext())
     {
          int x = f.nextInt();
          out.printf("It is %s that %d is
              isPrime(x),x);
     }
   }
```



- When the Scanner object ***f*** is first constructed using the "new" command, and linked to the datafile, an invisible ***data pointer*** points to the first element in the data file.
- The ***hasNext()*** method "asks" the ***data pointer*** if it is pointing to anything.

MyLoops3.java | MyLoops3.in ×

```
1 2
2 5
3 9
4 17
5 23
6 45
7 97
8
```

File input scenario #1 – Unknown quantity of values

ops3.java × MyLoops3.in

```
 */
public class MyLoops3
{
  /**This method returns a boolean value, true indicat
   *integer parameter is a prime number, and false tha
    */
   public static boolean isPrime(int X)
   {
     int i=2;
     while(i<X)
     {
         if(X%i==0)
             return false;
         i++;
     }
     return true;
   }
   public static void main (String [] args)
     throws IOException
   {
     Scanner f = new Scanner(new File("MyLoops3.in"));
     while (f.hasNext())
     {
         int x = f.nextInt();
         out.printf("It is %s that %d is a prime number.\n",
             isPrime(x),x);
     }
   }
```

- If the data pointer IS pointing to an element, the ***hasNext()*** method returns a ***true*** to the while loop, which then performs the action statements inside the loop.
- The "step" in this loop is the ***nextInt*** command, which does two things.

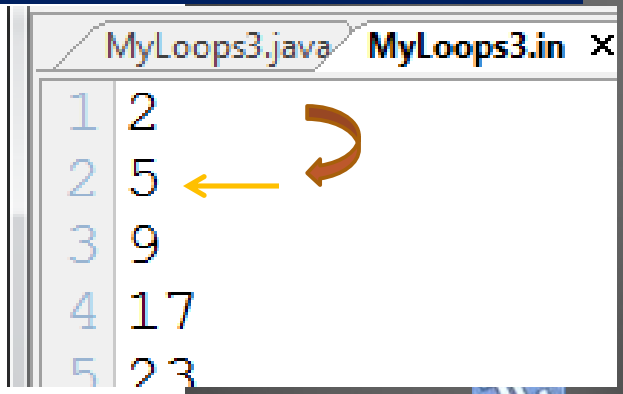
MyLoops3.java MyLoops3.in ×

```
1 2
2 5
3 9
4 17
5 23
6 45
7 97
8
```

File input scenario #1 – Unknown quantity of values

```
ops3.java ×  MyLoops3.in
 */
public class MyLoops3
{
  /**This method returns a boolean value, true indicat
   *integer parameter is a prime number, and false that
    */
   public static boolean isPrime(int X)
   {
     int i=2;
     while(i<X)
     {
          if(X%i==0)
              return false;
          i++;
     }
     return true;
   }
   public static void main (String [] args)
     throws IOException
   {
     Scanner f = new Scanner(new File("MyLoops3.in"));
     while (f.hasNext())
     {
          int x = f.nextInt();
          out.printf("It is %s that %d is a prime number.\n",
              isPrime(x),x);
     }
   }
```

- The ***nextInt*** command retrieves the integer to which the ***data pointer*** is pointing, which is then assigned to the variable
- The data pointer then "hops over (or down)" to the "next" integer, thus accomplishing the stepping action required for the loop.



File input scenario #1 – Unknown quantity of values

```
*/
public class MyLoops3
{
  /**This method returns a boolean value, true indicat
   *integer parameter is a prime number, and false that
    */
   public static boolean isPrime(int X)
   {
     int i=2;
     while(i<X)
     {
         if(X%i==0)
             return false;
         i++;
     }
     return true;
   }
   public static void main (String [] args)
     throws IOException
   {
     Scanner f = new Scanner(new File("MyLoops3.in"));
     while (f.hasNext())
     {
         int x = f.nextInt();
         out.printf("It is %s that %d is a prime number.\n",
             isPrime(x),x);
     }
   }
```

- The output statement processes the input value and shows the results so far while the ***data pointer*** is ready with the next data element in the file.

```
It is true that 2 is a prime number.
```

MyLoops3.in

```
2
5
9
17
23
45
97
```

File input scenario #1 – Unknown quantity of values

- This two-part retrieval and stepping action continues as long as the ***data pointer*** still has something to point to.

ps3.java × MyLoops3.in

```
 */
public class MyLoops3
{
  /**This method returns a boolean value, true indicat
   *integer parameter is a prime number, and false tha
    */
   public static boolean isPrime(int X)
   {
     int i=2;
     while(i<X)
     {
         if(X%i==0)
             return false;
         i++;
     }
     return true;
   }
   public static void main (String [] args)
     throws IOException
   {
     Scanner f = new Scanner(new File("MyLoops3.in"));
     while (f.hasNext())
     {
         int x = f.nextInt();
         out.printf("It is %s that %d is a prime number.\n",
             isPrime(x),x);
     }
   }
```

```
It is true that 2 is a prime number.
It is true that 5 is a prime number.
It is false that 9 is a prime number.
It is true that 17 is a prime number.
It is true that 23 is a prime number.
It is false that 45 is a prime number.
It is true that 97 is a prime number.
```

MyLoops3.java MyLoops3.in ×

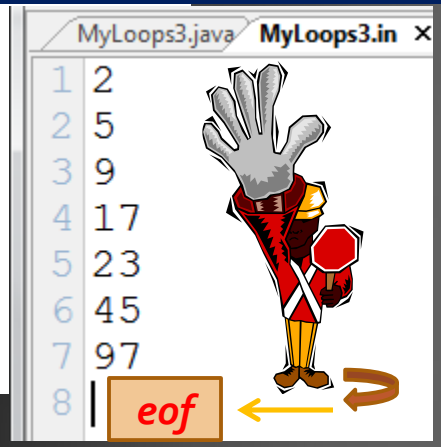
```
1 2
2 5 ←
3 9 ←
4 17 ←
5 23 ←
6 45 ←
7 97 ←
8
```

File input scenario #1 – Unknown quantity of values

ops3.java × MyLoops3.in

```
 */
public class MyLoops3
{
  /**This method returns a boolean value, true indic
   *integer parameter is a prime number, and false t
    */
   public static boolean isPrime(int X)
   {
     int i=2;
     while(i<X)
     {
         if(X%i==0)
             return false;
         i++;
     }
     return true;
   }
   public static void main (String [] args)
     throws IOException
   {
     Scanner f = new Scanner(new File("MyLoops3.in"));
     while (f.hasNext())
     {
         int x = f.nextInt();
         out.printf("It is %s that %d is a prime number.\n",
             isPrime(x),x);
     }
   }
```

- When the ***data pointer*** runs out of elements in the file and is now pointing to the ***eof*** (an invisible "end-of-file marker"), it causes the ***hasNext()*** command to return a ***false*** to the while loop, causing the loop to ***stop*** and the entire file input process to finish.



File input scenario #1 – Unknown quantity of values - summary

- To summarize this input process, the four parts of the while(f.hasNext()) loop are:
- **Start** – Scanner statement linking the file input object to the file
- **Check** – use of the hasNext() command
- **Action** – using the first of the two-step action of one of the next commands to retrieve a data element
- **Step** – using the second action of the next command to hop to the next element

File input scenario #2 – N sets of data

- To demonstrate this file input scenario, a common one used in programming competitions such as UIL and ACM, we'll use a simple situation.
- The data file contains an initial value N, which indicates there are N sets of data to follow.
- A ***for*** loop is appropriate to use, but any loop will do.
- Let's discuss this process…

MyLoops3.java | MyLoops3.in

```
22        public static void main (String [] args)
23          throws IOException
24        {
25          Scanner f = new Scanner(new File("MyLoops3.in"));
26          int N = f.nextInt();
27          for(int x=0;x<N;x++)
28          {
29              String animal = f.next();
30              out.printf("I love my %s.\n",animal);
31          }
32        }
33    }
```

MyLoops3.java | MyLoops3.in

```
1 4
2 dog
3 cat
4 horse
5 boat
```

```
I love my dog.
I love my cat.
I love my horse.
I love my boat.
Press any key to continue...
```

File input scenario #2 – N sets of data

- For simplicity's sake, we'll just focus on the file input process without any method calls or other processes for now.
- After the Scanner ***f*** object is constructed and linked, the initial integer N is input from the file.
- Instead of using the ***hasnext()*** feature as we did before, we'll now just set up a ***for*** loop to input N sets of data. Again, any loop will do.

MyLoops3.java

```
22        public static void main (String [] args)
23          throws IOException
24        {
25          Scanner f = new Scanner(new File("MyLoops3.in"));
26          int N = f.nextInt();
27          for(int x=0;x<N;x++)
28          {
29              String animal = f.next();
30              out.printf("I love my %s.\n",animal);
31          }
32        }
33    }
```

MyLoops3.in

```
4
dog
cat
horse
boat
```

```
I love my dog.
I love my cat.
I love my horse.
I love my boat.
Press any key to continue...
```

File input scenario #2 – N sets of data

- Notice carefully that the loop control variable starts at zero, and the check is ***x<N***. This achieves the correct number of input steps, the same as if you started at 1 and made the check ***x<=N.***
- Since N has a value of 4, 4 sets of data are input, and four outputs are the result, as you can see below…very simple and straightforward.

MyLoops3.java

```
22        public static void main (String [] args)
23          throws IOException
24        {
25          Scanner f = new Scanner(new File("MyLoops3.in"));
26          int N = f.nextInt();
27          for(int x=0;x<N;x++)
28          {
29              String animal = f.next();
30              out.printf("I love my %s.\n",animal);
31          }
32        }
33    }
```

MyLoops3.in

```
4
dog
cat
horse
boat
```

```
I love my dog.
I love my cat.
I love my horse.
I love my boat.
Press any key to continue...
```

Warning: manage the white space!

- Here is a reminder and a warning that was first introduced in Lesson 3B:
  - When a data file is arranged in this fashion with an initial integer N followed by several words or phrases requiring the ***nextLine()*** command to harvest, you must manage the whitespace character left behind by the ***nextInt()*** command.
  - To do this, simply follow the ***nextInt()*** command with a ***nextLine()*** command, and all will be well.

```
int N = f.nextInt();
f.nextLine();
```

File input scenario #2 – N sets of data - summary

- Here is a summary of this file input process, which is essentially that of the normal for loop, with an additional part at the Start:
- **Start** – The for loop control variable x starts at zero AND the Scanner statement linking the file input object to the file.
- **Check** – x < N
- **Action** – use any of the next commands to retrieve a data element
- **Step** – x++

File input scenario #2 – N sets of data - variation

- A shorter variation of this scenario uses the while loop:

```
int N = f.nextInt();
while(N-->0)
{
        //next command
}
```

- The beauty of this variation is that the check and the step are contained inside the while statement.

File input scenario #2 – N sets of data - variation

- Here's how this works:

```
int N = f.nextInt();
while(N-->0)
{
        //next command
}
```

  - The **N--** part of the statement is a **post-decrement operation**, where the value of N is used first in the comparison, then is immediately decremented, thus providing both the check, and then the step down.

File input scenario #2 – N sets of data - variation

- Here's how this works:

```
int N = f.nextInt();
while(N-->0)
{
        //next command
}
```

  - If N has a value of 3, indicating 3 data elements to follow, the value of N is checked first with a **true** result, then immediately decremented, becoming 2, and a data element is retrieved. With a value of 2 N is checked again as **true** , then goes down to 1, with the next data element retrieved. At 1 N still checks as **true**, goes to zero, with the final data element retrieved. At zero, the check is finally false, and the process stops AFTER the three data elements are taken.

Third file input scenario

- File Scenario #3 - An unknown quantity of values all on one line separated by spaces, harvested using a special advanced technique called "split" (we'll devote an entire lesson – 6D – for this process, which will require a brief introduction to arrays)

Lesson Summary

- In this lesson, you learned additional techniques for inputting data into your program from external data files, using several scenarios that are commonly used.
- Now it is time to practice with several examples.

Labs

- MyLoops3 will be the name of this series of labs. As you did before, create a separate folder and file called MyLoops3 and do your work there.
- The first method is done for you to help you get started.

Lab 1 – *Col to Row doubles*

- WAP in the main method to read and output each of the values in a data file using File Scenario #1 (unknown quantity of values). The values in the data file are arranged vertically.
- Output is to be in a single HORIZONTAL line with each value formatted to two decimal places and separated by single spaces. A solution is shown below.

MyLoops3.java | MyLoops3_1.in

```
4.7
1.9
3.4
5.9
1.3
6.3
```

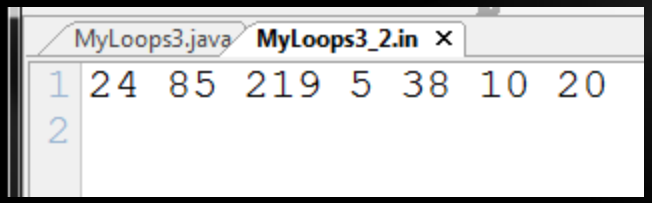
C:\Program Files (x86)\Xinox Software\JCreatorV5L...

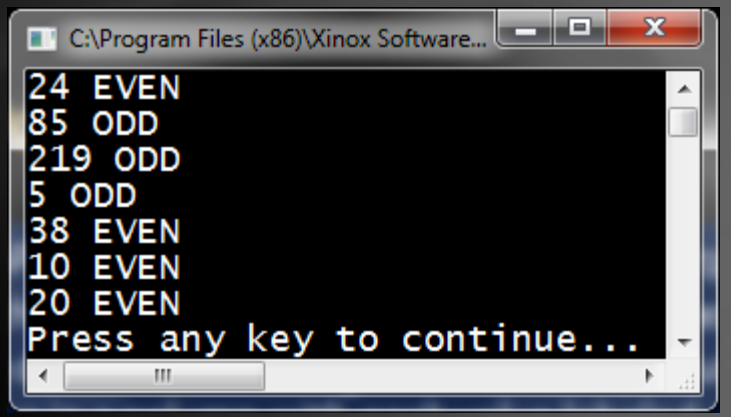
```
4.70 1.90 3.40 5.90 1.30 6.30
Press any key to continue...
```

```
    public static void main (String [] args)
      throws IOException
    {
      Scanner f = new Scanner(new File("MyLoops3_1.in"));
      while(f.hasNext())
      {
          double num = f.nextDouble();
          out.printf("%.2f ",num);
      }
      out.println();
    }
}
```

Lab 2 – *Odd/Even*

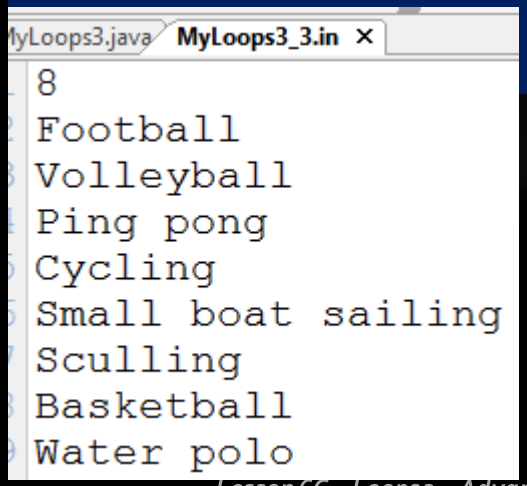
- WAP in the main method to read and output each of the values in a data file (again, use FS#1). The values in the data file are arranged horizontally.
- Output is to be in a column with each value labeled as ODD or EVEN. Define and use a static method called ***oddEven*** to receive an integer parameter and return the appropriate string, ODD or EVEN.

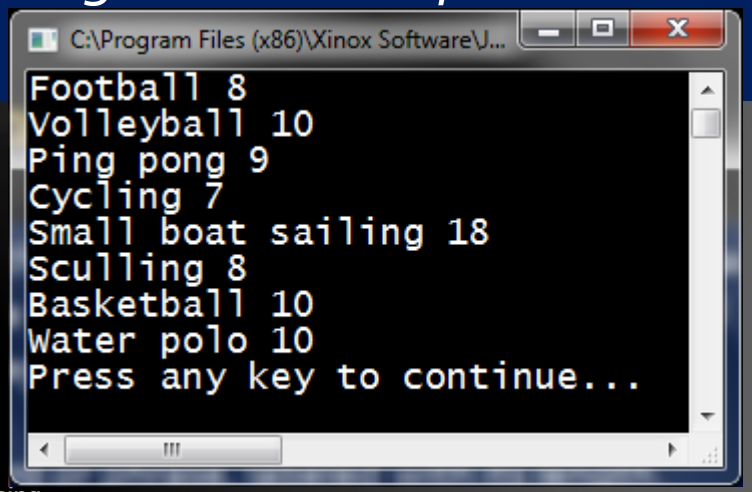




Lab 3 – *String length*

- WAP to display the lengths of the words stored in a data file. The data file will be File Scenario #2, with an initial N value at the beginning, followed by N words or phrases.
- Output each word or phrase, labeled with its length.
- *Hint 1: Use nextLine() since there are some lines with multiple words*
- *Hint 2: Remember to manage the white space!!!*





Lab 4 – *String length 2*

- Repeat Lab 3, but this time align all length values right justified in column 25, as shown below.
- To do this you will need to use the printf formatting process with some creative use of the String class methods.
- *Hint 1: Think about what column contains the ones digits of each number.*
- *Hint 2: Subtraction will be necessary.*

MyLoops3.java | MyLoops3_3.in

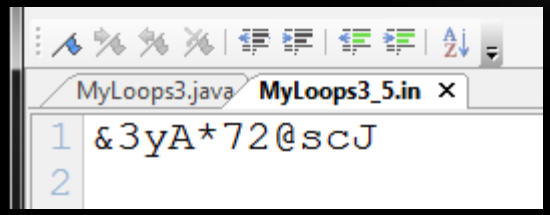
```
8
Football
Volleyball
Ping pong
Cycling
Small boat sailing
Sculling
Basketball
Water polo
```

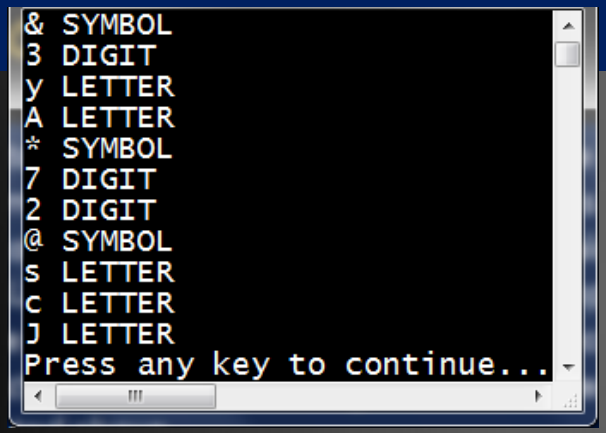
C:\Program Files (x86)\Xinox Software\JCrea...

```
---------*---------*----^----*
Football                    8
Volleyball                 10
Ping pong                   9
Cycling                     7
Small boat sailing         18
Sculling                    8
Basketball                 10
Water polo                 10
Press any key to continue...
```

Lab 5 – *Character Type*

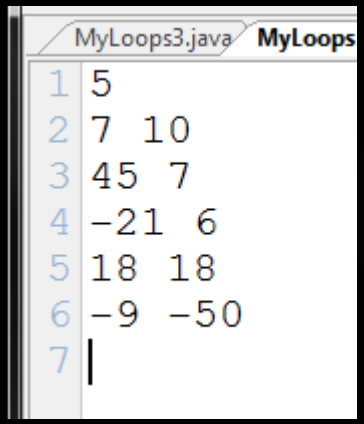
- WAP to read a single String from a data file and determine if each character in that String is a letter, a digit, or a symbol. Output each character along with the word LETTER, DIGIT, or SYMBOL as appropriate.
- Define and use the method ***charType*** to receive a single character and return a String with the correct "type" as indicated above.
- *Hint: The **Character** class would come in handy here. Examine the API to check out the Character class methods that might be helpful.*

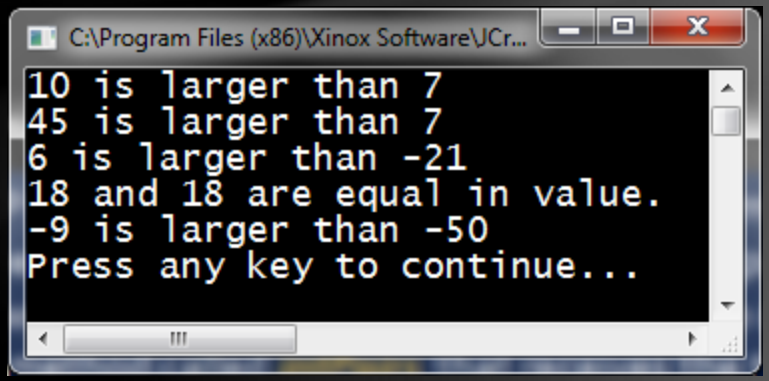




Lab 6 – *Integer pairs*

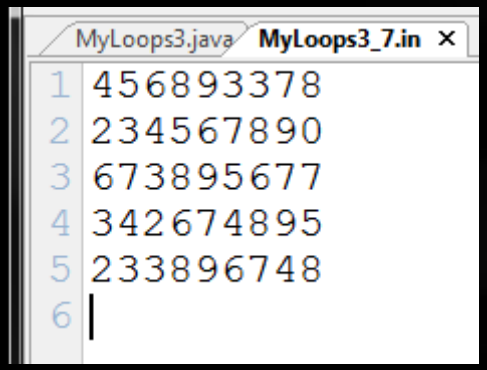
- WAP to read N pairs of integers and determine which is the larger and smaller value, or if they are equal, using the output format shown below. The entire resulting String should be calculated in and returned from a method called ***intPairs*** that receives the two integers and returns the appropriate String sentence. *Hint: The Math class would be very helpful here.*

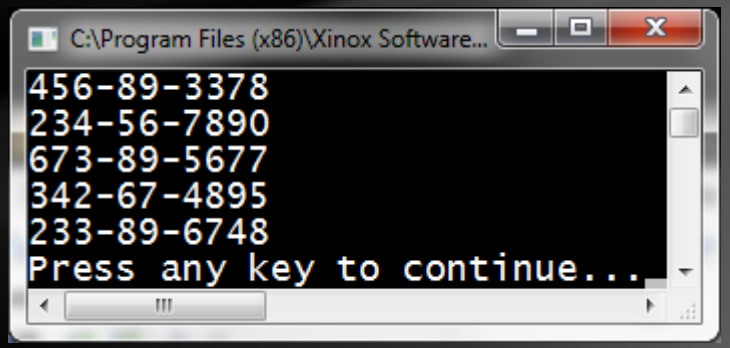




Lab 7 – *Social Security*

- WAP called ***ssNum*** that will read an unknown quantity of nine-digit numeric strings representing Social Security numbers and output that string properly formatted with dashes. For example, the string `370592433` would be formatted as `370-59-2433`.





Lab 8 – ***Area and Circumference***

- WAP that will read N ***diameter*** values and output in table format the ***radius***, ***area***, and ***circumference*** of the circle.
- All output - headings, values, alignment and decimal formatting - must match ***EXACTLY*** as shown.
- The area and circumference must be calculated in return methods called ***area*** and ***circ***, both of which receive a decimal value. Use Math.PI in your calculations.

MyLoops3.java | MyLoops3_8.in

```
1 6
2 3.5
3 10.16
4 4.006
5 23.99
6 96.1
7 31
8
```

C:\Program Files (x86)\Xinox Software\JCreatorV5LE\G...

```
RADIUS      AREA          CIRCUMFERENCE
======...==========...=============
  1.8         9.62            10.996
  5.1        81.07            31.919
  2.0        12.60            12.585
 12.0       452.01            75.367
 48.1      7253.32           301.907
 15.5       754.77            97.389
Press any key to continue...
```

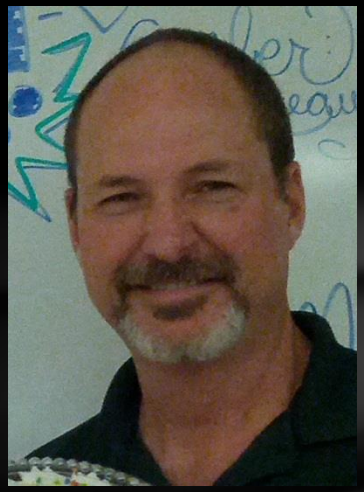
JavaDoc

- Complete the documentation for all of the methods and run the JavaDoc utility for this class.

CONGRATULATIONS!

- You now know how to use two different file processes to input data to your program.
- *Lesson 6D will explore a third file input process that uses the String "split" command to create an array of Strings.*

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