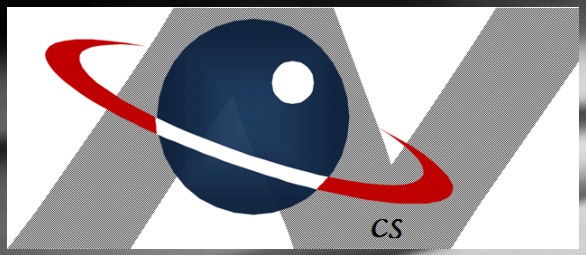
O(N) CS LESSONS

Lesson 3B – File Input



By John B. Owen

Table of Contents

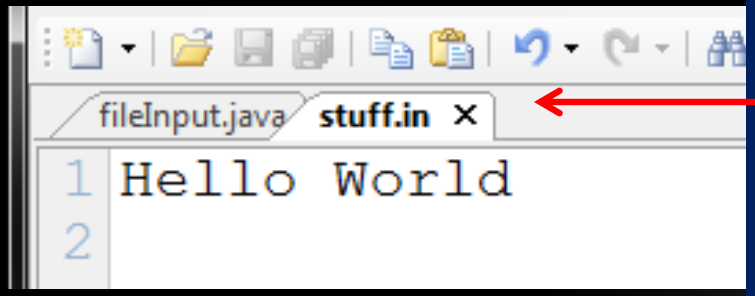
Objectives

- Students will understand file input using the Scanner class.
- A few "previews" of some advanced file processing techniques will be presented
- Students will also do various labs using all previously learned techniques.

File Input Setup

- Software packages quite often involve multiple files that interact with each other, with data values input from and stored to files other than the executable file.
- The process for inputting data from a file is quite easy.
- See the next slide for a very simple example.

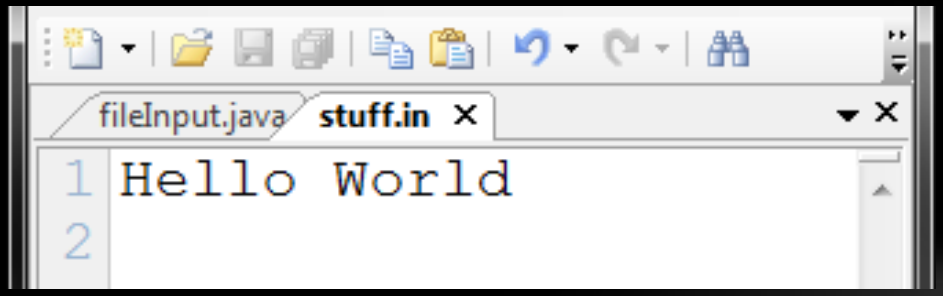
File Input Setup



A data file is simply a text file as shown on the left, like "stuff.in". The extension can be .in, .dat, .txt, or any extension that is not a normally reserved one like .exe or .bat.

```
fileInput.java | stuff.in

1  import java.util.*;
2  import java.io.*;
3  public class fileInput
4  {
5     public static void main (String [] args)
6         throws IOException
7     {
8           Scanner fileIn = new Scanner
9               (new File("stuff.in"));
10          String s = fileIn.next();
11          System.out.println(s);
12    }
13 }
```

File Input Setup



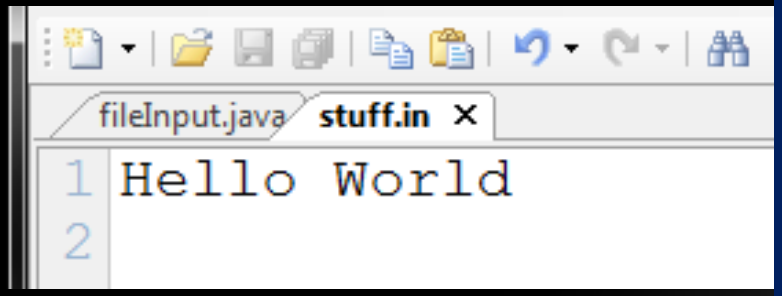
Below is a program showing the steps necessary for file input.

- Two Java package import statements are required:
  - import java.util.*
  - import java.io.*

fileInput.java | stuff.in

```
1   import java.util.*;
2   import java.io.*;
3   public class fileInput
4   {
5      public static void main (String [] args)
6          throws IOException
7      {
8            Scanner fileIn = new Scanner
9                (new File("stuff.in"));
10           String s = fileIn.next();
11           System.out.println(s);
12     }
13  }
```

File Input Setup

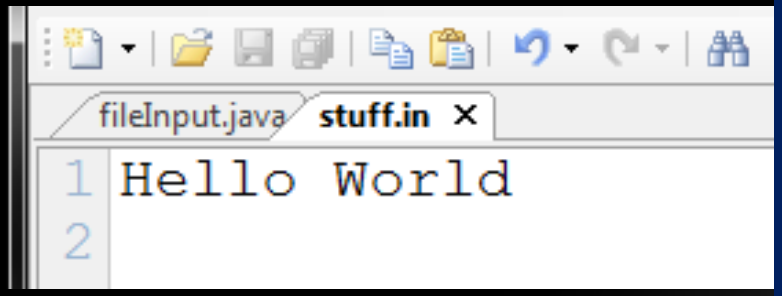


The statement ***throws IOException*** is required after the main heading. This is necessary to enable file input…a more complete explanation for this requirement will be provided in a later lesson on Exception Handling.

fileInput.java | stuff.in

```
1  import java.util.*;
2  import java.io.*;
3  public class fileInput
4  {
5     public static void main (String [] args)
6        throws IOException
7     {
8           Scanner fileIn = new Scanner
9              (new File("stuff.in"));
10          String s = fileIn.next();
11          System.out.println(s);
12    }
13 }
```

File Input Setup



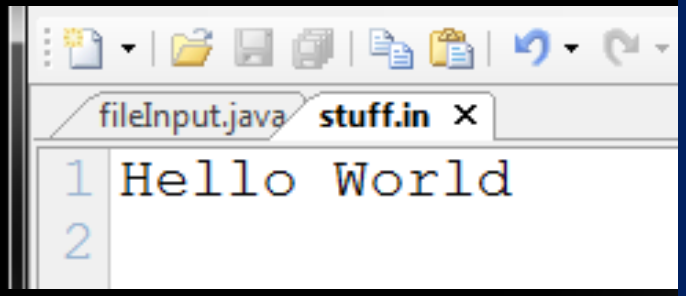
The "link" between the main file and the data file is made by this statement, which creates a new Scanner object, like **`fileIn`**, to the data file, "**`stuff.in`**".

Note: The data file must be in the ***same folder*** as the executable file, otherwise a "***path***" must precede it inside the quotes.

Note: This lesson series will NOT be using paths.

```
fileInput.java | stuff.in

1  import java.util.*;
2  import java.io.*;
3  public class fileInput
4  {
5     public static void main (String [] args)
6           throws IOException
7     {
8           Scanner fileIn = new Scanner
9                 (new File("stuff.in"));
10          String s = fileIn.next();
11          System.out.println(s);
12    }
13 }
```

File Input Setup

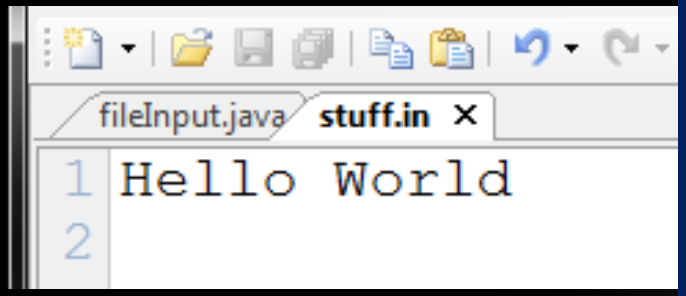


Once the two Java packages are imported, the exception handling statement is attached, and the Scanner file object created and linked, the rest of the program works just like keyboard input.

fileInput.java | stuff.in

```
1  import java.util.*;
2  import java.io.*;
3  public class fileInput
4  {
5     public static void main (String [] args)
6         throws IOException
7     {
8           Scanner fileIn = new Scanner
9               (new File("stuff.in"));
10          String s = fileIn.next();
11          System.out.println(s);
12    }
13 }
```

File Input Process



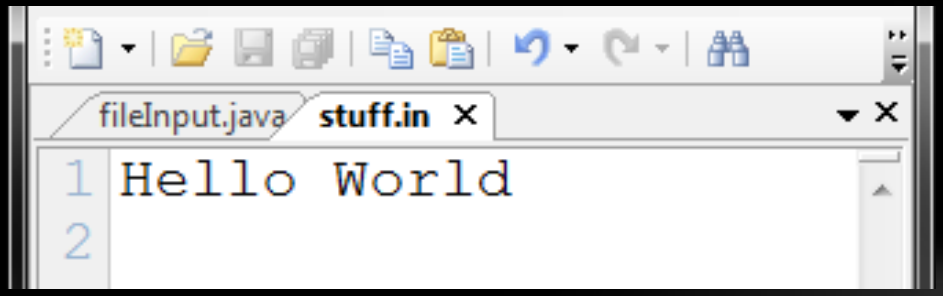
The **fileIn** object uses the ***next*** command in the same way the keyboard object **kb** used it in the previous lesson.

Remember that the ***next*** command only takes the first word from the data file since it uses the first space as the delimiter, just as we learned in the keyboard input process in the previous lesson.

Note: The lab solutions later on will simply use the letter f instead of fileIn as the file input object.

```
fileInput.java | stuff.in
1  import java.util.*;
2  import java.io.*;
3  public class fileInput
4  {
5     public static void ma
6        throws IOExceptio
7     {
8        Scanner fileIn = new Scanner
9           (new File("stuff.in"));
10       String s = fileIn.next();
11       System.out.println(s);
12    }
13 }
```

File Input Process

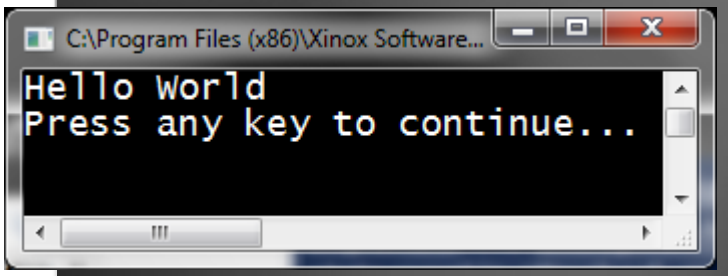


With the ***nextLine*** command replacing ***next***, you can see the entire line is retrieved from the data file, again, just as it did for keyboard input.

Note: As discussed in Lesson 3A, the same nextLine "quirk" issue exsists with file input as did with keyboard input, so be aware of that and use the same fix as necessary.

fileInput.java

```
1   import java.util.*;
2   import java.io.*;
3   public class fileInput
4   {
5       public static void main (String [] args)
6           throws IOException
7       {
8           Scanner fileIn = new Scanner
9               (new File("stuff.in"));
10          String s = fileIn.nextLine();
11          System.out.println(s);
12      }
13  }
```



File Input Process

Here you see an example of a person's return address stored in a file, retrieved by the program, and output to the screen.

fileInput.java | stuff.in

```
1 Sam Keystone
2 121 S Main Street
3 Brooks, Texas 79621
4
```

fileInput.java | stuff.in

```
1  import java.util.*;
2  import java.io.*;
3  public class fileInput
4  {
5     public static void main (String [] args)
6         throws IOException
7     {
8           Scanner fileIn = new Scanner
9               (new File("stuff.in"));
10          String name   = fileIn.nextLine();
11          String street = fileIn.nextLine();
12          String city   = fileIn.nextLine();
13          System.out.printf("%s\n%s\n%s\n",
14                  name,street,city);
15    }
```

```
Sam Keystone
121 S Main Street
Brooks, Texas 79621
Press any key to continue...
```

Advanced Techniques

fileInput.java | stuff.in

```
1 Smith, Joe
2 Sanders, Steve
3 Baker, Gerald
4 White, Tiffany
5 Kellogg, Susie
```

This program example gives you a glimpse of how powerful file input can be, using some advanced processes. Although you won't learn these techniques in depth yet, their function is fairly evident if you examine them carefully.

You need to learn this technique for your first lab test.

fileInput.java | stuff.in

```
1  import java.util.*;
2  import java.io.*;
3  public class fileInput
4  {
5     public static void main (String [] args)
6         throws IOException
7     {
8           Scanner fileIn = new Scanner
9               (new File("stuff.in"));
10          ArrayList <String>names
11              = new ArrayList<String>();
12          while(fileIn.hasNext())
13              names.add(fileIn.nextLine());
14          Collections.sort(names);
15          for(String s:names)
16              System.out.println(s);
17    }
```

```
Baker, Gerald
Kellogg, Susie
Sanders, Steve
Smith, Joe
White, Tiffany
Press any key to co
```

Advanced Techniques

fileInput.java | stuff.in

```
1 Smith, Joe
2 Sanders, Steve
3 Baker, Gerald
4 White, Tiffany
5 Kellogg, Susie
```

The data file contains several names, not in alphabetical order. The program uses a ***while loop*** to read all of the names and adds each to an ***ArrayList of Strings***. The list is sorted in alpha order, and then output using a ***for-each*** loop.

fileInput.java | stuff.in

```
1  import java.util.*;
2  import java.io.*;
3  public class fileInput
4  {
5     public static void main (String [] args)
6        throws IOException
7     {
8        Scanner fileIn = new Scanner
9           (new File("stuff.in"));
10       ArrayList <String>names
11          = new ArrayList<String>();
12       while(fileIn.hasNext())
13          names.add(fileIn.nextLine());
14       Collections.sort(names);
15       for(String s:names)
16          System.out.println(s);
17    }
```

C:\Program Files ...

```
Baker, Gerald
Kellogg, Susie
Sanders, Steve
Smith, Joe
White, Tiffany
Press any key to co
```

Advanced Techniques

fileInput.java | stuff.in

```
1 Smith, Joe
2 Sanders, Steve
3 Baker, Gerald
4 White, Tiffany
5 Kellogg, Susie
```

These four advanced techniques – the ***while loop***, the ***ArrayList, Collections.sort,*** and the ***for-each loop*** – will all be studied later on in much more detail.

fileInput.java | stuff.in

```
1  import java.util.*;
2  import java.io.*;
3  public class fileInput
4  {
5     public static void main (String [] args)
6         throws IOException
7     {
8           Scanner fileIn = new Scanner
9               (new File("stuff.in"));
10          ArrayList <String>names
11              = new ArrayList<String>();
12          while(fileIn.hasNext())
13              names.add(fileIn.nextLine());
14          Collections.sort(names);
15          for(String s:names)
16              System.out.println(s);
17    }
```

C:\Program Files ...

```
Baker, Gerald
Kellogg, Susie
Sanders, Steve
Smith, Joe
White, Tiffany
Press any key to co
```

Advanced Techniques

Here is essentially the same program, using integers instead.

fileInput.java | stuff.in

```
1  import java.util.*;
2  import java.io.*;
3  public class fileInput
4  {
5     public static void main (String [] args)
6         throws IOException
7     {
8           Scanner fileIn = new Scanner
9               (new File("stuff.in"));
10          ArrayList <Integer> nums
11              = new ArrayList<Integer>();
12          while(fileIn.hasNext())
13              nums.add(fileIn.nextInt());
14          Collections.sort(nums);
15          for(Integer i:nums)
16              System.out.println(i);
17    }
18 }
```

fileInput.java | stuff.in

```
1  45
2  12
3  42
4  77
5  84
6  36
7  57
```

C:\Progr...

```
12
36
42
45
57
77
84
Press any key t
```

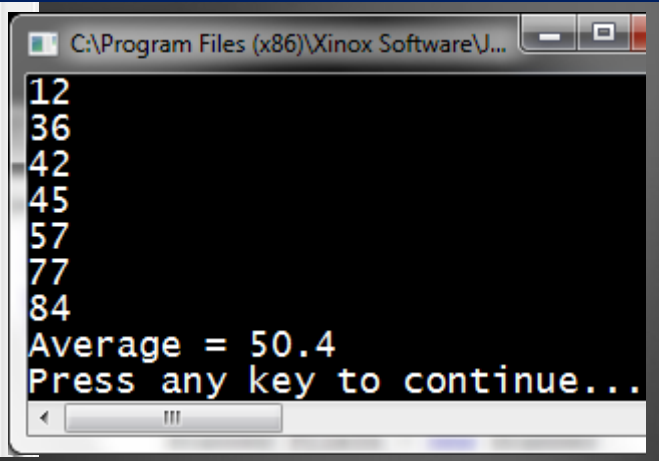
Advanced Techniques

.java × stuff.in

```
import java.util.*;
import java.io.*;
public class fileInput
{
   public static void main (String [] args)
       throws IOException
   {
         Scanner fileIn = new Scanner
             (new File("stuff.in"));
         ArrayList <Integer> nums
             = new ArrayList<Integer>();
         while(fileIn.hasNext())
             nums.add(fileIn.nextInt());
         Collections.sort(nums);
         int sum=0,count=0;
         for(Integer i:nums)
         {
             System.out.println(i);
             sum+=i;
             count++;
         }
         double average = (double)sum/count;
         System.out.printf("Average = %.1f\n",average);
}
```

Expanding further into advanced techniques, this program introduces ***counting*** and ***accumulating*** inside a loop. Using two utility variables, ***sum*** and ***count***, as each value is output, its value is accumulated into a total amount, and counted, so that the average of all the number can be calculated and output at the end.

You need to learn this technique for your second lab test.

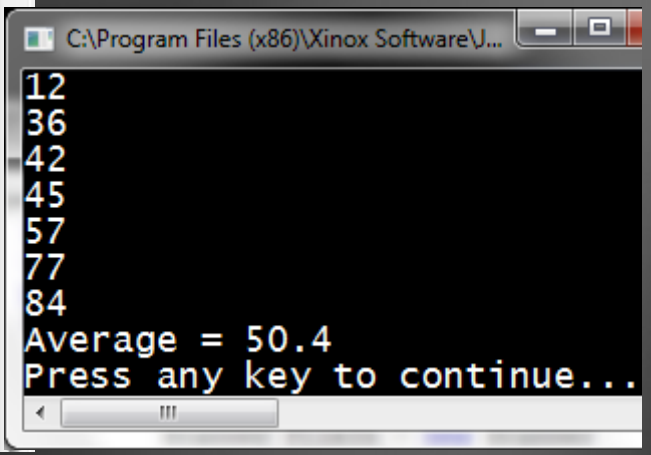


Advanced Techniques

.java × stuff.in

```
import java.util.*;
import java.io.*;
public class fileInput
{
   public static void main (String [] args)
       throws IOException
   {
         Scanner fileIn = new Scanner
             (new File("stuff.in"));
         ArrayList <Integer> nums
             = new ArrayList<Integer>();
         while(fileIn.hasNext())
             nums.add(fileIn.nextInt());
         Collections.sort(nums);
         int sum=0,count=0;
         for(Integer i:nums)
         {
             System.out.println(i);
             sum+=i;
             count++;
         }
         double average = (double)sum/count;
         System.out.printf("Average = %.1f\n",average);
}
```

It is necessary to ***double*** cast the ***sum*** variable so that decimal division takes place, otherwise any fractional part would be discarded since both ***sum*** and ***count*** are integer values. (See ***Lesson 2C, More Operations*** to review division rules)



Lesson Summary

- This lesson showed you how to input data from a file, including all of the necessary setup steps, as well as the input commands, which were exactly the same as the ones you learned for keyboard input.
- A few "previews of advanced techniques" were also shown.

Labs

- File input will be the primary input mechanism for labs from here on.
- For this lesson's lab assignment, simply ***redo all the labs from Lesson 3A***, with file input instead. The labs are relisted on the next few slides, with file input instructions instead of keyboard input.
- Also, type in the three Advanced Techniques shown in this lesson, exactly as you see them.

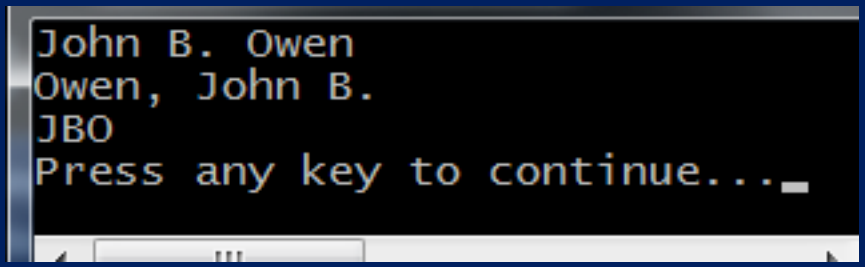
Lab 3B-1 (same as Lab3A1, using file input instead)

WAP to input FROM A FILE your first name, middle initial, and last name into three separate variables called *first* (String), *middle*(char), and *last*(String). Output your name three different ways as shown below. (Hint: The `String.charAt(0)` command can also be used in output statements to show the first letter of a word.)

```
Input file name "lab3B1.in":
John
B
Owen
```

(Use your own name in this data file)

```
John B. Owen
Owen, John B.
JBO
```



Lab 3B-1

- Here is the entire program to get you going.

lab3B1.java | lab3B1.in

```
John
B
Owen
```

lab3B1.java | lab3B1.in

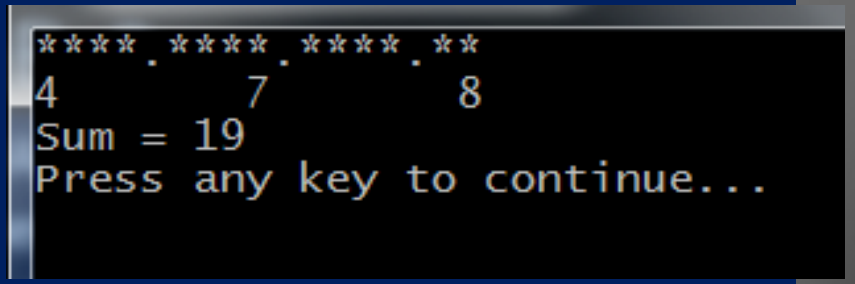
```
1  //Owen Lessons 2014-2015
2  //Copyright John B. Owen, All rights reserved
3
4  //Solution - Lab3B1 - Same as Lab 3A1, using file input
5     import java.util.*;
6     import java.io.*;
7     public class lab3B1
8     {
9        public static void main (String [] args)
10         throws IOException
11       {
12             String first,last;
13             char middle;
14             Scanner f = new Scanner(new File("lab3b1.in"));
15
16             //Input section
17             first = f.next();
18             middle = f.next().charAt(0);
19             last = f.next();
20
21             //Output section
22             System.out.println(first+" "+middle+". "+last);
23             System.out.println(last+", "+first+" "+middle+".");
24             System.out.println(""+first.charAt(0)+middle+last.charAt(0));
25       }
26    }
```

Look! NO PROMPTS!!

Lab 3B-2 (same as Lab3A2, using file input)

- WAP to input FROM A FILE three integers and output them in columns 8 spaces wide.
- Note: NO PROMPTS ARE NECESSARY WITH FILE INPUT.
- Use the "\t" escape command to tab 8 spaces.
- Assign to a variable the sum of all three values, and output it on the next line, as shown below.
- Document your source code, marking the three main sections with comments: Input, Process and Output

```
Input file name "lab3B2.in":
4 7 8

****.****.****.**
4       7       8
Sum = 19
```



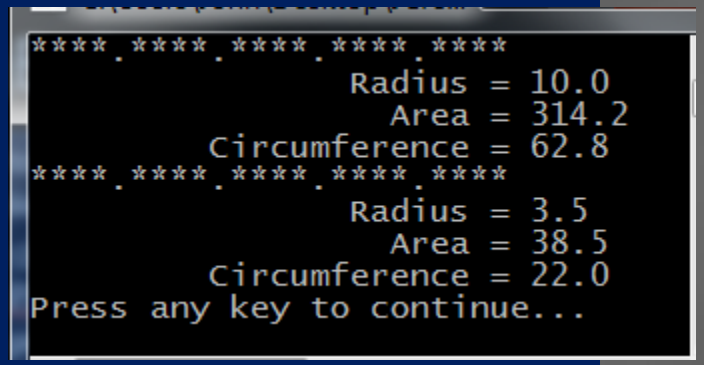
Lab 3B-3 (same as Lab3A3, using file input)

WAP to input FROM A FILE the radius values of two circles into decimal variables *rad1* and *rad2*. Calculate and assign to the variables *area1, area2, circ1,* and *circ2* the area and circumference of each circle. Use Math.PI in your calculations. Be sure to label and format each output value properly as shown below. Align the "=" signs in column 24.

Input file name "lab3B3.in":

```
10.0 3.5
```

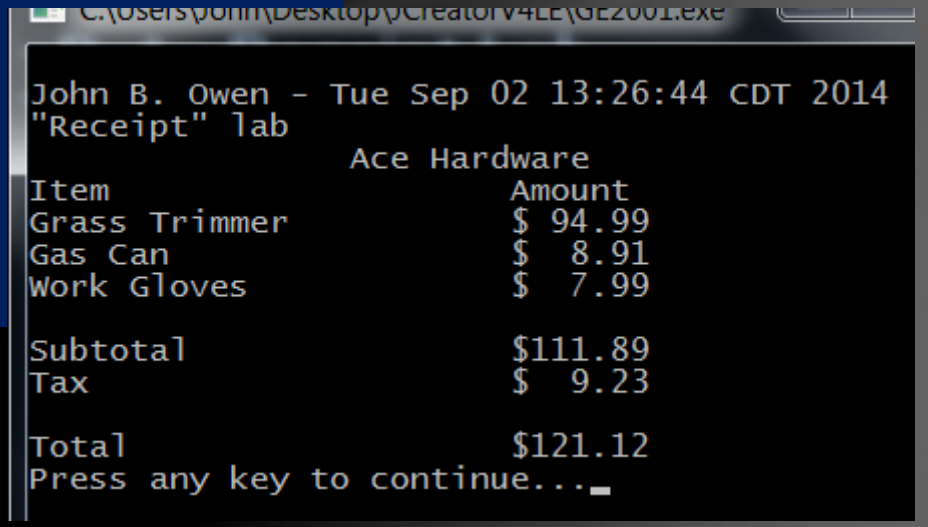
```
****.****.****.****.****
                Radius = 10.0
                  Area = 314.2
        Circumference = 62.8
****.****.****.****.****
                Radius = 3.5
                  Area = 38.5
        Circumference = 22.0
```



3B-4 - Receipt Lab – (version 4 of this lab, using file input)

Using file input this time, do this lab once more, with your receipt.
In the data file, list your items as shown below, with the description on one line, followed immediately by the price. You must handle the ***nextLine "quirk"*** in the same way as you did for keyboard input.

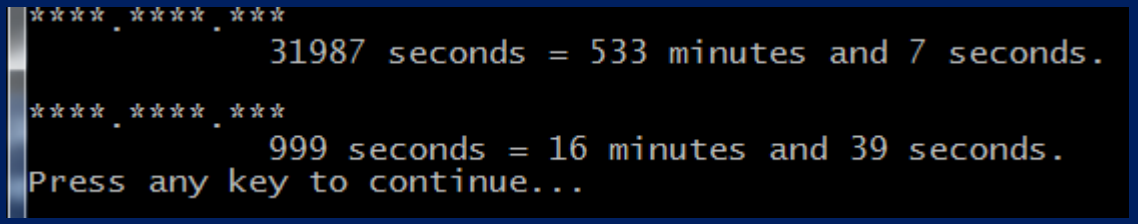
```
Input file name "lab3B4.in":
Grass Trimmer
94.99
Gas Can
8.91
Work Gloves
7.99
```



Lab 3B-5 (same as lab3A5, using file input)

WAP to input FROM A FILE the number of seconds and convert to the number of minutes and seconds that value represents. Use / (div) to determine the number of minutes and % (mod) to determine the number of seconds. Indent to create EXACT OUTPUT. Make your program run with two sets of data, indented and formatted exactly as shown.

Input file name "lab3B5.in":
31987 999



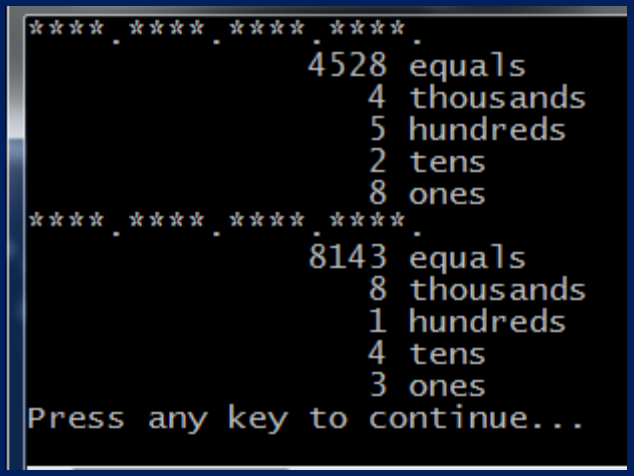

```
****.****.***
            31987 seconds = 533 minutes and 7 seconds.
****.****.***
            999 seconds = 16 minutes and 39 seconds.
```

Lab 3B-6 (same as lab3A6, using file input)

WAP to print the thousands, the hundreds, the tens, and the ones digits separately for an input 4-digit integer. Print each digit on a separate line with a label indicating its place value. Use / and % operators to mathematically separate out the digits. Create EXACT OUTPUT. DO NOT trivialize this task by inputting the value as a String to use the charAt method. Make your program run with two sets of data, FROM A FILE, indented and formatted exactly as shown on the next slide.

Lab 3B-6

```
Input file name "lab3B6.in":
4528 8143
****.****.****.****.
                4528 equals
                   4 thousands
                   5 hundreds
                   2 tens
                   8 ones
 ****.****.****.****.
                8143 equals
                   8 thousands
                   1 hundreds
                   4 tens
                   3 ones
```



Lab 3B-7 `(same as lab3A7, using file input)`

WAP to "make change" for a decimal input value FROM A FILE, such as `388.97` (`38897` pennies), to the smallest possible number of bills and coins. Do NOT use half-dollars or two-dollar bills.

- You must use / and % to mathematically accomplish this task.
- Run your program a second time to convert the input value `412.42`.
- Make your program run with two sets of data, indented and formatted exactly as shown.
- Hint: Input the money value into a double, multiply the value and cast it into an integer that represents the value in pennies, and then mathematically use / and % in combination (see Lesson 2C) to calculate each denomination.

Lab 3B-7

```
Input file name "lab3B7.in":
388.97 412.42

****.****.****.****.****.
             $388.97 = 38897 pennies
                   3 = Benjamin(s)   - hundred
                   1 = Grant(s)      - fifty
                   1 = Jackson(s)    - twenty
                   1 = Hamilton(s)   - ten
                   1 = Lincoln(s)    - five
                   3 = Washington(s)- one
                   3 = quarter(s)
                   2 = dime(s)
                   0 = nickel(s)
                   2 = penny(s)
               17 money items in all
```

Lab 3B-7

```
****.****.****.****.****.
            $412.42 = 41242 pennies
                  4 = Benjamin(s)   - hundred
                  0 = Grant(s)      - fifty
                  0 = Jackson(s)    - twenty
                  1 = Hamilton(s)   - ten
                  0 = Lincoln(s)    - five
                  2 = Washington(s)- one
                  1 = quarter(s)
                  1 = dime(s)
                  1 = nickel(s)
                  2 = penny(s)
           12 money items in all
```

Lab 3B-7


**** **** **** **** ****
$388.97 = 38897 cents
= 3 Benjamin(s) - hundred
= 1 Grant(s) - fifty
= 1 Jackson(s) - twenty
= 1 Hamilton(s) - ten
= 1 Lincoln(s) - five
= 3 Washington(s)- one
= 3 Quarter(s)
= 2 Dime(s)
= 0 Nickel(s)
= 2 Pennies(s)
17 money items in all
**** **** **** **** ****
$412.42 = 41242 cents
= 4 Benjamin(s) - hundred
= 0 Grant(s) - fifty
= 0 Jackson(s) - twenty
= 1 Hamilton(s) - ten
= 0 Lincoln(s) - five
= 2 Washington(s)- one
= 1 Quarter(s)
= 1 Dime(s)
= 1 Nickel(s)
= 2 Pennies(s)
12 money items in all
Press any key to continue...

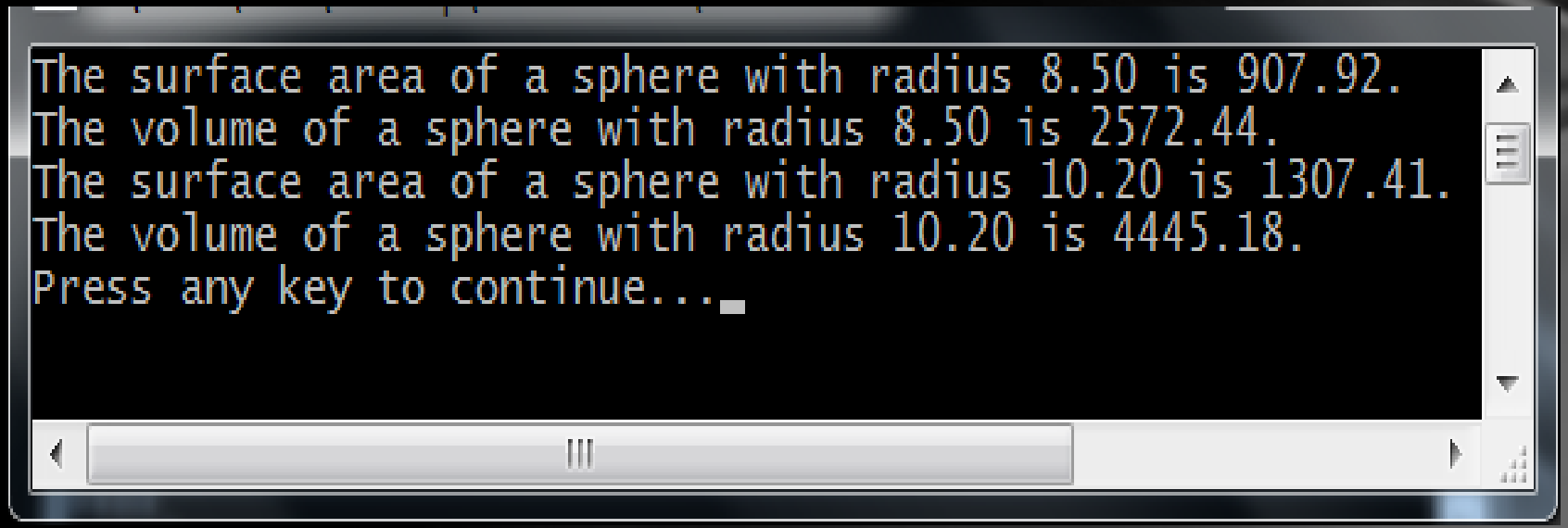
Lab 3B-8 (same as lab3A8, using file input)

WAP to input FROM A FILE the radius of a sphere, and then calculate and output the surface area and volume of that sphere.

- Make your program accept two input values and produce two outputs.
- DO NOT ask your instructor or your neighbor for these formulas. Instead, **find the formulas for surface area and volume of a sphere yourself** in a math textbook or on the internet.

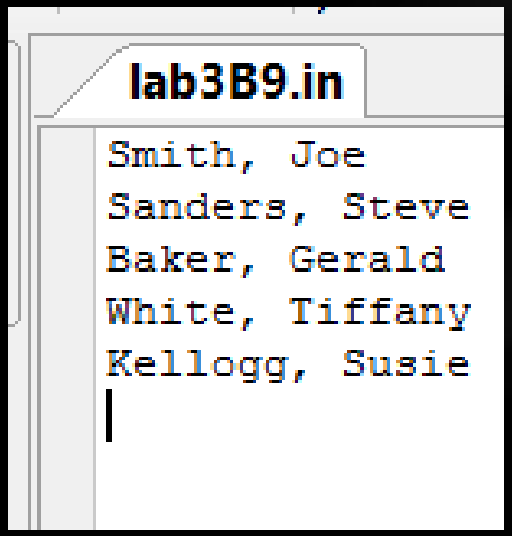
```
Input file name "lab3B8.in":
8.5 10.2

The surface area of a sphere with radius 8.50 is 907.92.
The volume of a sphere with radius 8.50 is 2572.44.
The surface area of a sphere with radius 10.20 is 1307.41.
The volume of a sphere with radius 10.20 is 4445.18.
Press any key to continue...
```

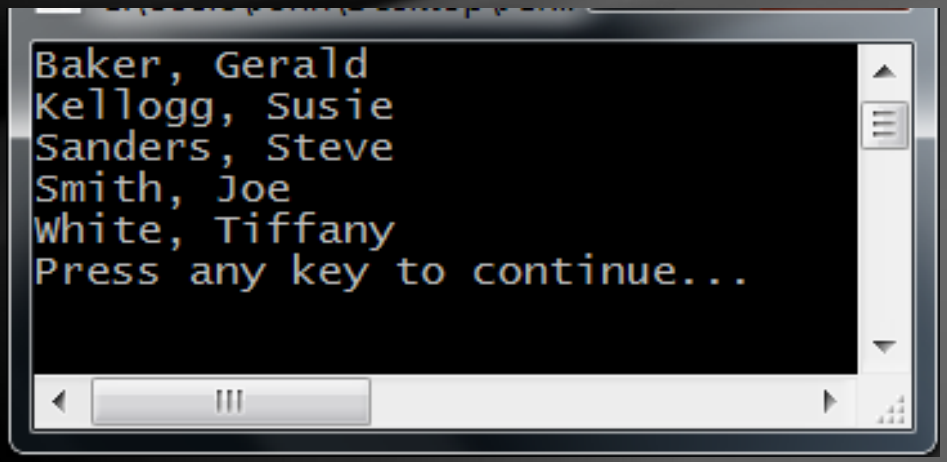
Lab 3B-8


The surface area of a sphere with radius 8.50 is 907.92.
The volume of a sphere with radius 8.50 is 2572.44.
The surface area of a sphere with radius 10.20 is 1307.41.
The volume of a sphere with radius 10.20 is 4445.18.
Press any key to continue...

Lab 3B-9

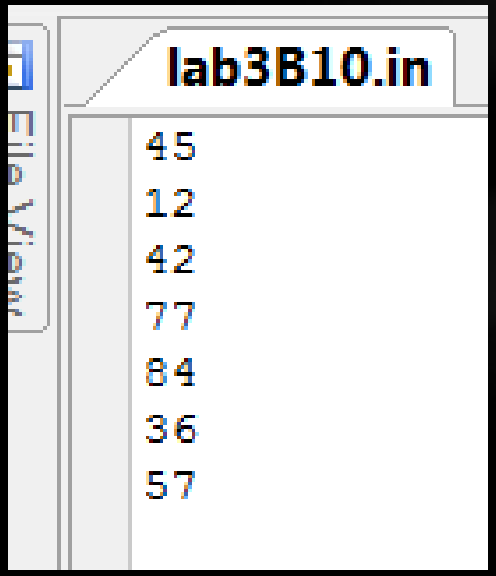
WAP that implements the first Advanced Technique demonstrated in this lesson, found on slides 13-15.

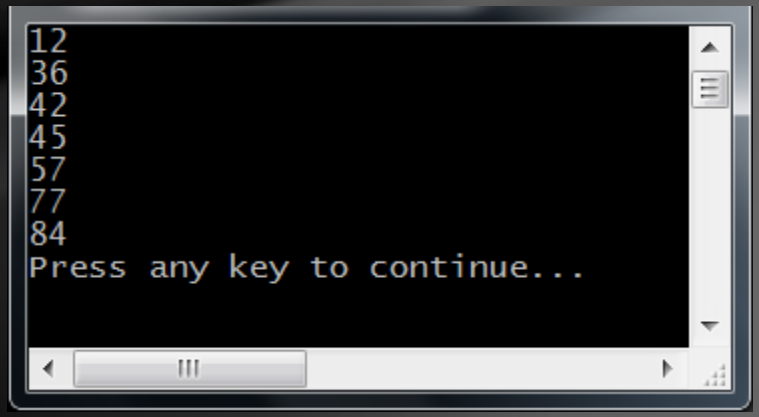




Lab 3B-10

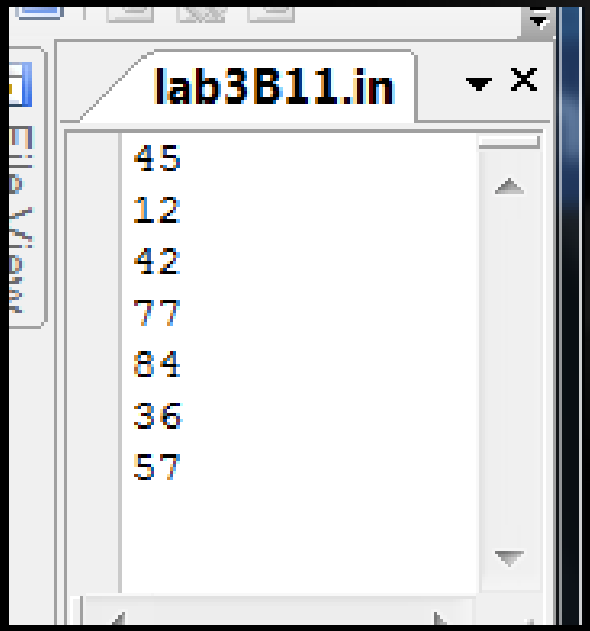
WAP that implements the second Advanced Technique demonstrated in this lesson, found on slide 16.

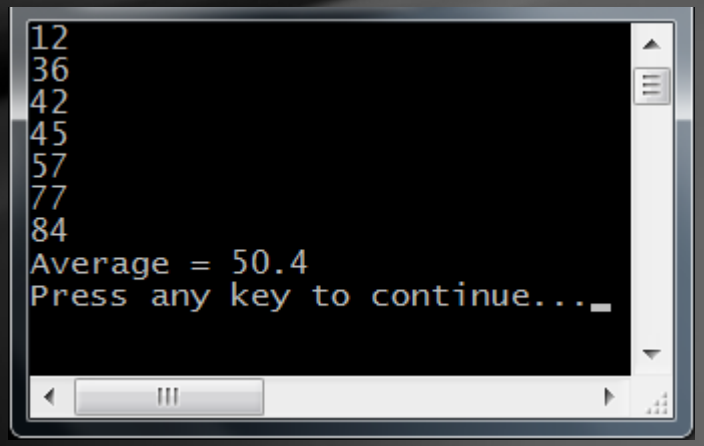




Lab 3B-11

WAP that implements the third Advanced Technique demonstrated in this lesson, found on slides 17 and 18.

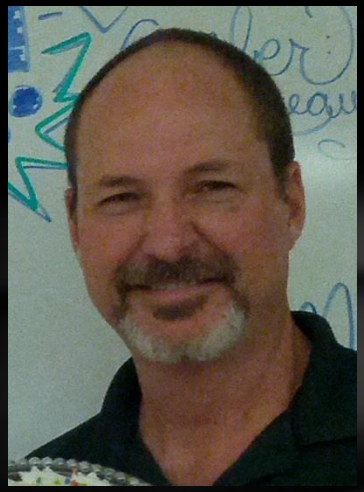




CONGRATULATIONS!

- You now understand how to manage file input.
- *Lesson 4A will introduce the use of the if and if else conditional statements.*

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