Lab 11 Name:_____________________________ Checked:_______

Objectives:

Learn more about iterators and using in novel ways, including reading data from a String or directly from a webpage.

Preparation: Review the basics of processing a text file, line by line

1) In part (g) of Lab 3 we explored how to input values from a file and store them in an array. Review the steps necessary to set up your Scanner to input from a file.

2) Try this program: www.csc.villanova.edu/~map/2014/f14/examples/FileInput.java

```
//********************************************************************
//  FileInput.java       Author: MAP
//  Demonstrates the use of Scanner to read text file input.
//********************************************************************

import java.util.Scanner;
import java.io.*;

public class FileInput
{
   //-----------------------------------------------------------------
   //  Reads text from a file and prints it in uppercase.
   //-----------------------------------------------------------------
   public static void main (String[] args) throws IOException
   {
      String line;
      Scanner fileScan;

      File myFile = new File("sample.txt");
      fileScan = new Scanner (myFile);

      // Read and process each line of the file
      while (fileScan.hasNext())
      {
         line = fileScan.nextLine();
         System.out.println (line.toUpperCase());
      }

   }
}
```

1) Download and compile this program; create a small text file named **`sample.txt`** to test it. Run **`FileInput`** – what does it do?

____

2) Modify it to use the parameter **`args[0]`** of **`main()`** as the file name. Do this as follows:

- Replace the use of

  `File myFile = new File("sample.txt");`

  with

  `File myFile = new File(args[0]);`

In jGrasp, select "Run Arguments" from the Build menu, and provide the file name as an argument (parameter) to **`main`** by typing **`sample.txt`** in the box that appears above your program. In this way, you can run your program with different files, without modifying the code. Try it running **`FileInput.java`** with the program itself (**`FileInput.java`**) as its input!

•

A. Scanning from a String

Just as we can use a Scanner to input from a file or System.in, we can also use a Scanner to "input" from a String!

1) Try this code: **Lab11a.java**

```
//*******************************************************************
//  Lab11a.java         MA Papalaskari
//  Simple example: scanning from a String
//*******************************************************************

import java.util.Scanner;

public class Lab11a
{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);

      System.out.print("Please type 3 words:  ");
      String line = scan.nextLine();

      Scanner scanLine = new Scanner(line);

      String word1 = scanLine.next();
      String word2 = scanLine.next();
      String word3 = scanLine.next();

      System.out.println("Word 1:  " + word1);
      System.out.println("Word 2:  " + word2);
      System.out.println("Word 3:  " + word3);

   }
}
```

Run : **Lab11a.java** – what does it do?

____

____

B. Scanning from a String and doing something useful

Next, we will create **Lab11b.java** by modifying Lab11a.java so that it does something more interesting with the input. Our new program will treat the input as a command for a simple numeric computation.
For example, the input might be:
`55 * 83`
We want the program to compute and print the product 4565. First, run Lab11a.java with this input and observe how it picks out the "55", "*", "83" as **`word1`**, **`word2`**, and **`word3`**, respectively. Note that the code uses **`scanLine.next()`** which produces **`String`** tokens and that was fine because **`word1`**, **`word2`**, and **`word3`** are Strings. But now you want to use the values 55 and 83 as numbers, so the variables have to be of type

`double` (we could use **`int`**, but **`double`** will allow you to handle a wider range of values), and you need to obtain their values using **`scanLine.nextDouble()`** instead of **`scanLine.next()`**.

Can you use these ideas to create a simple calculator? Change the prompt from "Please enter 3 words" to "Calculate: "
Note that you can test the value of word2.charAt(0) to see if it is equal to '+', '*', etc, and, accordingly, compute the result. (If you want to be able to handle more than 2 operators, it is best to use a switch statement.)

Sample runs:

```
----jGRASP exec: java Lab11b
Calculate:
9.3 + 44.7
= 54.0
```

```
----jGRASP exec: java Lab11b
Calculate:
55 * 83
= 4565.0
```

C. Input data into an array

The technique described in Parts A and B is also useful for processing data organized in columns and inputting that into an array. Go back to the code for Part A (NOT Part B) and modify the code so that it inputs 8 words into an array of 8 Strings. (Be sure to replace the variables word1, word2 etc by an array word[0], word[1], etc. and use a for-loop to get the input. The words should then be printed *backwards*.

Tab delimited data:
Sometimes the input tokens can contain spaces. For example, the "words" could be country names:

```
 India United States France China Germany Greece South Korea Brazil
```

These are still just 8 countries! In such situations, a tab can be used as a delimiter, so the String would be stored as:

```
"India\tUnited States\tFrance\tChina\tGermany\tGreece\tSouth Korea\tBrazil"
```

In order for your Scanner to use a delimiter other than whitespace, you need to specify this before doing any input:

```
scanLine.useDelimiter("\t");
```

Sample run:

```
----jGRASP exec: java Lab11c
Enter 8 country names, all in one line, separated by tabs:
England  France   Japan   India    Greece   United States  South Korea   Sierra Leone
Sierra Leone
South Korea
United States
Greece
India
Japan
France
England
```

Note: these are tab characters

D. Processing data from text files, organized in columns (Combine Parts A & C)

The technique described in Part C is useful for processing text files containing data organized in columns. We now modify **`FileInput.java`** (from the preparation steps, above) so that after it inputs each line, it uses the technique of **`Lab11c.java`** (i.e., a second Scanner) to "scan" 8 words from each line in the file and store these words in an array, then print the contents of the array *backwards*. Try this with the following file: http://www.csc.villanova.edu/~map/2014/f14/examples/eightwords.txt

Sample output:

```
Line: England   France  Japan   United Arab Emirates    Greece  United States   South Korea
Sierra Leone
Sierra Leone
South Korea
United States
Greece
United Arab Emirates
Japan
France
England

Line: apple     orange  asian pear      fig     persimmon       grape   raspbery
pineapple
pineapple
raspbery
grape
persimmon
fig
asian pear
orange
apple

Line: black     white   gray    light gray      dark gray       red     blue    green
green
blue
red
dark gray
light gray
gray
white
black
```

E. (Optional) Input directly from a website

Would you like your program to access a website directly? Here is how. You need to

1) Add another import directive at the beginning or your program:

```
import java.net.URL;
```

2) Set up your Scanner to read from the url instead of a file. Here is an example:

```
String myurl =     "
http://www.csc.villanova.edu/~map/2014/f14/examples/eightwords.txt";
InputStream inStream = new URL(myurl).openStream();
Scanner webScan = new Scanner (inStream);
```

3) Now you can use `webScan` as any other `Scanner` object, to input from a webpage as if it were any other text file. (Try it running **`FileInput.java`** with input **`FileInput.java`** but this time, use the version that is online.)
This technique will work with most webpages, as long as they can be read as text (including html files).