CSC 1051 - Lab 11

In this lab we will learn how to access text files using Scanner to get their contents. We will also try reading data directly from a webpage.
Before you begin, create a folder for your Lab 11 files. It is important to save all the data and program files for this lab in the same folder.

Warm-up 1: Compute average of 10 numbers
Implement and test a Java program AverageNumbers.java using the algorithm:

Constants:
MAX = 10

Variables:
count, sum, n (integers)

Algorithm:
count = 0
sum = 0

while (count<MAX)
   input n
   sum = sum + n
   count = count + 1
print “Average = “ + sum/MAX

Warm-up 2: Compute average word length of 10 words
Implement and test a Java program AverageWordLength.java using the algorithm:

Constants:
MAX = 10

Variables:
count, sum, n (integers)
word (String)

Algorithm:
count = 0
sum = 0

while (count<MAX)
   input word
   n = number of characters in word
   sum = sum + n
   count = count + 1
print “Average = “ + sum/MAX

1 You can start by modifying the previous program, but be sure to keep both versions.
1. Reading a text file and converting to uppercase

www.csc.villanova.edu/~map/1051/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.inp");
        fileScan = new Scanner (myFile);

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

Download and compile this program; create a small text file named `sample.inp` and save it in the same folder as the program (you can create the file in Notepad or directly in JGrasp – under File-> New, choose “Plain text” instead of “Java” for the file type). Run FileInput – what does it do?
2. Reading word by word, instead of line by line

Modify the program so that instead of using `nextLine()`, it simply uses `next()`.

`next()` obtains and returns the next token from the scanner. A “token” is any sequence of characters that does not include white space (space, tab or newline). Thus, it can be used to pick out words from sentences. Play around with your input file `sample.inp`. Does it contain punctuation, spaces, tabs, or other symbols? If not, add some and observe what counts as “token”.

Here is a summary of some of the Scanner methods:
(Consult the Java API for full details: [http://docs.oracle.com/javase/6/docs/api/](http://docs.oracle.com/javase/6/docs/api/))

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hasNext()</code></td>
<td>Returns true if this scanner has another token in its input.</td>
</tr>
<tr>
<td><code>hasNextDouble()</code></td>
<td>Returns true if the next token in this scanner's input can be interpreted as a double value using the <code>nextDouble()</code> method.</td>
</tr>
<tr>
<td><code>hasNextInt()</code></td>
<td>Returns true if the next token in this scanner's input can be interpreted as an int value in the default radix using the <code>nextInt()</code> method.</td>
</tr>
<tr>
<td><code>hasNextLine()</code></td>
<td>Returns true if there is another line in the input of this scanner.</td>
</tr>
<tr>
<td><code>next()</code></td>
<td>Finds and returns the next complete token from this scanner.</td>
</tr>
<tr>
<td><code>nextDouble()</code></td>
<td>Scans the next token of the input as a double.</td>
</tr>
<tr>
<td><code>nextInt()</code></td>
<td>Scans the next token of the input as an int.</td>
</tr>
<tr>
<td><code>nextInt(int radix)</code></td>
<td>Scans the next token of the input as an int.</td>
</tr>
<tr>
<td><code>nextLine()</code></td>
<td>Advances this scanner past the current line and returns the input that was skipped.</td>
</tr>
</tbody>
</table>
3. Adapting programs to read from a file
Using FileInput.java as a model, modify your program AverageNumbers.java and program AverageWordLength.java to input from a text file, instead of from interactive input. Here are some pointers:

- Do not eliminate the Scanner object that you used for interactive input (you can still input some things interactively, eg, the name of the file containing your input data); instead, add another Scanner object (similar to filescan, in FileInput.java) to input from a file.
- Eliminate the constant MAX that controls how many items are to be input; instead, use the same technique as in FileInput.java to input numbers or words until you reach the end of the file (you should still use the variable count to keep track of how many items have been input and to calculate the average at the end of the program).
- Create files numbers.inp and wordyStuff.inp to test your programs.
- Once you have the programs working, try it with some “real” data files, for example one of the papers or reports you have written. You can create the input file by copy/paste into a new text file in JGrasp or by saving a Word file as plain text. (The file extension does not have to be .inp, it is common to use .dat or .txt for text files containing data.)

4. (Optional) Input directly from a website
Would you like your program to access a website directly? Here is how. You need to

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

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

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

```java
String myurl = "http://www.csc.villanova.edu/~map/1051/proj09.html";
InputStream inStream = new URL(myurl).openStream();
Scanner webScan = new Scanner (inStream);
```

c) Now you can use webScan as any other Scanner object, to input from a webpage as if it were any other text file. This works with most pages, as long as they can be read as text.