CS1003 Week 2 Exercise 1: Java I/O

As with all lab exercises, this exercise is not assessed. It is intended solely to help you understand the module material. There is probably more here than you will have time for during the lab hour; you are encouraged to complete it in your own time if you don't finish it.

The aim of this exercise is to practice Java file I/O. To familiarise yourself with the Linux environment, we recommend that you use one of the lab machines running Linux This exercise is intended to be done on a Linux/Unix machine. If you have a Unix/Linux environment on your own machine you may prefer to use that. However, you will need to use the lab machine to run *stacscheck* so it is a good idea to log into a lab machine to do this exercise.

We also recommend that you use a text editor such as *vi, nano* or *emacs* to write your code. If you prefer you can use Visual Studio, Eclipse or IntelliJ IDEA etc., however, it may make the last steps in the exercise more difficult.

1. Reading from a file

Open **Terminal** and use suitable **`mkdir`**, **`cd`** and **`touch`** commands to create a suitable directory for CS1003 exercises, for this week's exercise, and a suitable Java file for the exercise. For example, you could use:

```
mkdir cs1003-exercises; cd cs1003-exercises; mkdir W02-1-IO; cd W02-1-IO; touch W02_1_IO.java
```

Then copy the following file into your file space:

`https://studres.cs.st-andrews.ac.uk/CS1003/Exercises/W02/test-short.txt`

Write, compile and test a Java program that reads in all the contents of the file, line by line (do not use the bulk load option that you were shown in the practical), and prints out to the console the number of lines of text in the file. There should be 15 lines in the file. You may need to consult the API documentation to discover how to detect the end of the file. You may recall that you can compile and run your program from Terminal in the `W02-1-IO` directory using:

```
javac W02_1_IO.java
java W02_1_IO
```

2. Processing file data

Make a copy of your previous program. You could do this by e.g. copying your old program code to a method such as `public static void step1()` and calling `step1` from the `main` method, then copying that method to a new method called `step2`.

Modify your new program it so that it counts the total number of words[1] in the file. The methods *String.split()*, *String.charAt()*, or the class *Scanner* may be useful. There should be 47 words in the file. Add the word count to the program output.

3. More processing

Make a copy of your previous program, and modify it so that before printing out the numbers of lines and words in the input file, it prints out the contents of the file (like the command `cat` — try executing the command `cat test-short.txt` in a Terminal).

4. Command-line arguments

Make a copy of your previous program, and modify it so that it uses the first element of the `String[] args` parameter to the `main` method as the filename. It should treat `args[0]` as the filename. Assuming `test-short.txt` is in the directory containing your compiled class file, try running your program from the command line:

```
java W02_1_IO test-short.txt
```

[1] Where a word is defined as a sequence of any characters other than spaces and newlines.

5. Test with larger input data

Test your program with the input file:

`https://studres.cs.st-andrews.ac.uk/CS1003/Exercises/W02/test-long.txt`

There should be 3160 lines and 17305 words.

6. Submission

You are required to submit a zip file containing your exercise attempt to the appropriate slot in the Exercises category on MMS. Note that if you have done this work on the lab servers as suggested, you will need to upload your submission to your local machine before submitting to MMS.

To recursively zip the contents of a folder on Linux you can do this:

```
zip -r CS1003-W1-1-al.zip *
```

This will create a file called `CS1003-W1-1-al.zip` containing the entire contents of the current directory (files that match *).

Suppose that you have done that operation from this directory: `/home/al/CS1003` on the lab machine called `pc3-002-l`, you can copy the zip file from my local machine using `scp` as follows:

```
scp pc3-002-l.cs.st-andrews.ac.uk:/home/al/CS1003/CS1003-W1-1-al.zip .
```

and you will have a copy of the zip file in the working directory from which you executed the `scp` command. I can then submit the file to MMS using the web interface.

You could also use FileZilla instead of scp to download the file to your local machine (but you should really master scp).

7. Useful links

Working remotely: https://systems.wiki.cs.st-andrews.ac.uk/index.php/Working_remotely

Video tutorials: https://systems.wiki.cs.st-andrews.ac.uk/index.php/Video_tutorials

The host service: https://systems.wiki.cs.st-andrews.ac.uk/index.php/Linux_Host_service

Scp: https://systems.wiki.cs.st-andrews.ac.uk/index.php/How_to_use_the_Home_service_(OS_X)#Using_SCP.2C_SFTP_or_rsync

Filezilla: https://filezilla-project.org/

Using text editors on Linux:

Nano: https://www.nano-editor.org/docs.php

Vi/vim: https://www.vim.org/ https://www.vim.org/docs.php

Emacs: https://www.gnu.org/software/emacs/ https://www.gnu.org/software/emacs/documentation.html