Computer Science

COMPSCI 105 Semester 1 2014

Assignment ONE

Due Date

Due: **11:30pm, Wednesday 30th Apr, 2014**
Worth: **7% of your final mark**

Introduction

The aim of this assignment is to exercise the student's understanding of ArrayLists, Exceptions, FileIO and Nested Statements.

In Q1 you will try to handle all exceptions that may occur when reading a line from a file and writing into a file the number of strings provided in line 1 of the file.
In Q2 you will order matching strings from the first (one pattern) and second line (many patterns) of an input file. The modified lines will be written in an output file.
In Q3 you will order all matching strings from the first and second line of an input file. The modified lines will be written in an output file.
In Q4 you will read the first 3 lines of a file. The first line will provide the number of strings to match. The second line will contain a list of patterns to match. The third line will contain the patterns to be matched. The outcome will be written in an ouput file.
In all 4 questions, exceptions will have to be handled and code commented.

What you are to do

Firstly, become familiar with the program supplied. Download all source files from the assignment course page. The programs provided are part of the lecturing material. Your assignment is divided into 3 stages for ease of completion. Please complete the assignment in order of the stages.

Stage 1: Exceptions (Self learning)

You are required to implement `try` and `catch` statements in the `ReadWriteTextFromToFile` to handle run-time and FileIO errors. `ReadWriteTextFromToFile` reads a set of strings separated by a space in the second line of the file input.txt and writes back N strings (from first to N strings read from input.txt) in the output.txt file. N can be obtained by reading the number in line 1 of the input.txt file.
For example with the following **Q1** input.txt file:
3
One Two Three Four

The ouput.txt file should look like:
One Two Three

Exception handling in Java can greatly improve software quality. It’s definitely valuable for you to think about exception handling design in this program. You are required to write code that catches an exception(s), and think about what the exception means. Does the type you caught match what your program is documented to throw? Do you know how to handle the exception such that your program can correctly and safely resume execution?

Stage 2: Matching one element (15 marks)

Reusing as much of the code completed in Q1 as possible, implement the program which will read a unique pattern (as a set of characters) from line 1 of the file **input**.txt. You will then read line 2 of the file and match any of the strings read with the pattern provided in line 1. Each matching pattern should be written back in line 1 of the file **output**.txt. The non-matched strings should be written as per their order of appearance in line 2 of the file **output**.txt. The count of matched and unmatched should be written in line 3 of the output.txt file. You will be required to handle exceptions of the type IOException and RuntimeException.

For example with the following **Q2** input.txt file:

```
One
Two Three One Two One
```

The ouput.txt file should look like:

```
One One
Two Three Two
2 3
```

Line 1 contains the matched patterns; line 2 contains the unmatched patterns. Line 3 first provides the count of matched patterns then of unmatched patterns.

Stage 3: Matching several elements (20 marks)

Reusing as much of the code completed in Q2, implement the program which will read all patterns (as sets of characters separated by a space) from line 1 of the file **input**.txt. You will then sequentially match each of the above patterns to all strings read from line 2 of the file input.txt. You will sequentially write in the **output**.txt file the matching patterns in separate lines. All unmatched patterns will be written in the following line of the file. The count of each matched and unmatched patterns will be written in the last line of file **output**.txt. You will be required to handle exceptions of the type IOException and RuntimeException.

For example with the following **Q3** input.txt file:

```
One Two
Two Three One Two One
```

The ouput.txt file should look like:

```
One One
Two Two
Three
2 2 1
```

Line 1 contains the strings matched with the first pattern read in line 1 of input.txt; line 2 contains the strings matched with the second pattern read in line 1 of input.txt. Line 3 contains the unmatched patterns. Line 4 provides in order the count of matched patterns for each patterns read. The last number is the count of unmatched patterns.

Stage 4: Matching N elements (20 marks)

This program only differs from the programme just developed in stage 3 by the number of patterns to be matched as retrieved from line 3 of the file **input**.txt. Reusing as much of the code completed in Q3, implement the program which will read N patterns (as sets of characters separated by a space) from line 2 of the file input.txt. The number N of patterns to read will be obtained from line 1 of the file input.txt. You will then sequentially match N patterns to all strings read from line 3 of the file input.txt. You will sequentially write in the output.txt file the matching patterns in separate lines. All unmatched patterns will be written in the following line of the file. The count of each matched and unmatched patterns will be written in the last line of file output.txt. Be aware that N may be larger than the actual number of patterns to match. In this case you should ensure that all patterns from line 2 are taken into account. You will be required to handle exceptions of the type IOException and RuntimeException.

For example with the following **Q4** input.txt file:

```
1
One Two
Two Three One Two One
```

The ouput.txt file should look like:

```
One One
Two Three Two
2 3
```

For example with the following input.txt file:

```
2
One Two
Two Three One Two One
```

The ouput.txt file should look like:

```
One One
Two Two
Three
2 2 1
```

For example with the following input.txt file:

```
3
One Two
Two Three One Two One
```

The ouput.txt file should look like:

```
One One
Two Two
Three
2 2 1 0
```

Line 1 contains the number of strings matched with the first pattern read in line 1 of input.txt; line 2 contains the strings matched with the second pattern read in line 1 of input.txt. Line 3 contains the strings matched with the third pattern read in line 1 of input.txt unmatched patterns. **There are no unmatched patterns**. Line 4 provides in order the count of matched patterns for each patterns read. The last number is the count of unmatched patterns.

Submission

You may electronically submit your assignment through the Assignment Dropbox (http://adb.auckland.ac.nz) at any time from the first submission date up until the final date. Submit the latest version of your assignment; do **NOT** submit separate code for each stage!

Please double check that you have included all the files required to run your program in the zip file before you submit it. No marks will be awarded if your program does not compile and run.

Remember that the Forum (https://forums.cs.auckland.ac.nz/) is the ideal place to ask and share questions about the assignment but bear in mind that NO code may be posted there.

Warning

This assignment is to be done individually. No marks will be given to students who provide solution code to others or accept such code.

- The work done on this assessment must be your own work. Think carefully about any problems you come across, and try to solve them yourself before you ask anyone for help. Under no circumstances should you take or pay for an electronic copy of someone else's work, modify it, and submit it as your own.
- Penalties for copying will be severe – to avoid being caught copying, don't do it.
- To ensure you are not identified as cheating you should follow these points:
    - Always do individual labs by yourself.
    - Never show your code to another person.
    - Never put your code in a public place (e.g., forum, your web site).
    - Never leave your computer unattended. You are responsible for the security of your account.

If you have any doubts as to what counts as individual work, please read the departmental undergraduate handbook and/or see the lecturer.

Marking Scheme

Style: 15 marks

Marks	Description
6	Comments at the top of each class (containing your name, UPI, date, and a brief description of the class)
3	The program contains appropriate comments
3	Correct indentation
3	Good layout structure in your program

Correctness – 55 marks

	Stage 1: Exception Handling
15	Stage 2: Compare one pattern
20	Stage 3: Compare all patterns
20	Stage 4: Compare N patterns