Java Programming I: Exercise 14

The deadline for submissions is one week after the exercise.

The aim of this exercise is to learn a concept of Java I/O streams.

[Core Set, Problem 1] File I/O using Byte Stream

(4 points)

You should create a program to read a file "input.txt" using the *FileInputStream* class, and store input data to an array of integer.
After that, your program should write the reverse order of input data into the file "*output_q1.txt*" using the *FileOutputStream* class.

While working on this problem, pay attention to the examples on lecture slides 19, 20, 25, 26, and 28.

Your Java code should be saved to the *ReverseString.java* file.

Run:
% java ReverseString

input.txt is as folows:

```
There was once a poor shepherd boy who used to watch his flocks in the fields next to a dark forest near the foot of a mountain. One hot afternoon, he thought up a good plan to get some company for himself and also have a little fun. Raising his fist in the air, he ran down to the village shouting "Wolf, Wolf." As soon as they heard him, the villagers all rushed from their homes, full of concern for his safety, and two of his cousins even stayed with him for a short while. This gave the boy so much pleasure that a few days later he tried exactly the same trick again, and once more he was successful. However, not long after, a wolf that had just escaped from the zoo was looking for a change from its usual diet of chicken and duck. So, overcoming its fear of being shot, it actually did come out from the forest and began to threaten the sheep. Racing down to the village, the boy of course cried out even louder than before. Unfortunately, as all the villagers were convinced that he was trying to fool them a third time, they told him, "Go away and don't bother us again." And so the wolf had a feast.
```

Output File is as follows:

```
.tsaef a dah flow eht os dnA ".niaga su rehtob t地od dna yawa oG" ,mih dlot yeht ,emit driht a meht loof ot gniyrt saw eh taht decnivnoc erew sregalliv eht lla sa ,yletanutrofnU .erofeb naht reduol neve tuo deirc esruoc fo yob eht ,egalliv eht ot nwod gnicaR .peehs eht netaerht ot nageb dna tserof eht morf tuo emoc did yllautca ti ,tohs gnieb fo raef sti gnimocrevo ,oS .kcud dna nekcihc fo teid lausu sti morf egnahc a rof gnikool saw ooz eht morf depacse tsuj dah taht flow a ,retfa gnol ton ,revewoH .lufsseccus saw eh erom ecno dna ,niaga kcirt emas eht yltcaxe deirt eh retal syad wef a taht erusaelp hcum os yob eht evag sihT .elihw trohs a rof mih htiw deyats neve snisuoc sih fo owt dna ,ytefas sih rof nrecnoc fo lluf ,semoh rieht morf dehsur lla sregalliv eht ,mih draeh yeht sa noos sA ".floW ,floW" gnituohs egalliv eht ot nwod nar eh ,ria eht ni tsif sih gnisiaR .nuf elttil a evah osla dna flesmih rof ynapmoc emos teg ot nalp doog a pu thguoht eh ,noonretfa toh enO .niatnuom a fo toof eht raen tserof krad a ot txen sdleif eht ni skcolf sih hctaw ot desu ohw yob drehpehs roop a ecno saw erehT
```

Basic Knowledge
1. You can refer to lecture slides 1 to 28.

[Core Set, Problem 2] File I/O using Character Stream

(6 points)

You should create a program to read a file "input.txt" using *BufferedReader* with *FileReader* and *Scanner* classes. Your program should detect each word in the file.
For each word, your program should count word frequency in the file.
After that, the program should create the file "*output_q2.txt*" using *BufferedWriter* with *FileWriter* (and *PrintWriter*) and write the lines with the following structure into the file:

- First line: wordcount and number of words
- Secound line and after: word : counts, frequency ratio

The maximum number of words in the *input.txt* file is less than 200. You do not need to remove any marks such as periods, commas, semi-colons, etc.

While working on this problem, pay attention to the examples on lecture slides 34, 35, 37, and 38.

Your Java code should be saved to the *WordCounting.java* file.

Output File is as follows:

```
Wordcount: 216, Number of Words: 142
There : 1, 0.0046296296296296
was : 4, 0.018518518518518517
once : 2, 0.009259259259259259
a : 11, 0.050925925925925
poor : 1, 0.0046296296296296
shepherd : 1, 0.0046296296296296
boy : 3, 0.013888888888888888
who : 1, 0.0046296296296296

....
```

Basic Knowledge
1. You can refer to lecture slides 29 to 38.

[Advanced Set, Problem 3] Helping Doctors

(10 points)

Modern technologies are used nowadays to save lives of people. Home distance monitoring of the rate of breath is very helpful for the doctors to detect the condition of the patient. The average rate of breath within an hour is crucial in order to get the right treatment to the patient. Modern smartphones can be used to collect these data and send them to the hospital over the Internet. This approach is realistic. According to Statistical Handbook of Japan 2013, the number of mobile phone subscribers was 132.76 million in 2012. Japan's total population in 2012 was 127.52 million.

You task is to create a piece of software to analyze the data on the rate of breath received from the patient.

You should create two programs.

The first one is to generate data on the rate of breath received from one patient. This program should create a *DataStream* file consisting of 60 integer numbers. Each number should be a random value in range between 8 and 60. Each number means the rate of breath within one minute. The data in the file hold measurement within one hour.

The second program should read the *DataStream* file created by the first program, find the average value

for the rate of breath in a minute. If this value is more than 30, it should print on the screen the following message “Treatment is necessary: average rate of breath per minute is ” and the value for the average rate of breath per minute.

The name of the file for the first program: *GeneratingData.java*
The name of the file for the second program: *AnalyzingData.java*

You should answer the following question: How to debug the program *AnalyzingData.java*? In other words, how to make sure that the second program works well?

The file name for your answer is *HowToDebug.pdf*

Basic Knowledge
1. You can refer to lecture slides 39 to 49.