Java 8 Streams Lab Suggestions

Simple Streams

For these exercises, use start a stream from a list containing a small number of instances of the Student class (with name, grade, and a list of courses represented as String)

For each exercise ensure the stream pipeline ends with exactly this:

```
    .forEach(System.out::println);
```

1) Display all student names
2) Display all students shown with grade, in the form:
```
    Fred has grade 3.2
```
3) Display all "smart" students with grade and course count, e.g.:
```
    Fred has grade 3.2 and takes 3 classes
```
4) Display all the courses taken by all students (including the duplicates)
5) Use the API docs to modify 4 to exclude duplicate course names
6) Use the API docs to modify 5 to show the courses in alphabetical order
7) Display all student/course combinations, in this form:
```
    Fred takes Math
    Fred takes Physics
    Jim takes Art History
    Sheila takes Math
```

Monte Carlo

Complete this exercise “entirely with streams”. That is, avoid any loops, and avoid the use of "substring".

Simulate throwing ten dice at a time (invoking `ThreadLocalRandom.current()` will provide a useful random number generator). Add up the face value of all those dice, which should result in a total in the range 10-60. Repeat this “throwing” operation 10,000 times or more, and build a table representing the count of how many times each face-total shows up. Using that table, display a bar chart of the frequencies, something like:

```
10: *
[...]
26: **********************
27: ****************************
```

```
28: **********************************
29: *************************************
30: **************************************
```

...etc...

You should notice that the shape shown is something like the normal distribution or “bell curve”.

Pride and Prejudice

Obtain a copy of the plain text form of Jane Austen’s novel “Pride and Prejudice”. This can be downloaded from gutenberg.org. Open the file to give a stream of lines.

Starting from that stream, separately achieve the following results:

1) Print the most frequent 200 words, with counts, in descending order
2) Build & print a table of word-length frequency
3) Build & print a table of frequency of counts-of-distinct-letters in the words. I.e. the word "hello" has four distinct letters, while letters has five.
4) Build a table of letter frequency for the book & print in descending order.
5) Calculate the ratio of "distinct letters to actual letters" in the books words.

You will likely find it helpful to check the API documentation for:

```
    java.nio.file.Files
    java.util.regex.Pattern
```