Keep Hope Alive

1 Introduction

In this timed lab you'll use command line arguments, parse strings, do simple calculations, and gain practice in modifying existing code. The bulk of this assignment is in `String` operations.

2 Problem Description

Georgia Tech students often try to optimize their effort in each course, for example, to ensure they continue to meet the minimum requirements for the HOPE scholarship. Being a good engineering student, you recognize the value of a program that could answer the question "what average do I need on the remainder of this course's assignments to make a particular grade?"

3 Solution Description

Modify the provided `HowToPass` class whose constructor takes a `String` of the form

```
Exams: <exam1>[, <exam2>, <exam3>]; Timed Labs: <tl1>[, <tl2>, <tl3>]; Homeworks: <hw1>[ <hw2>, ..., <hw10>]
```

and, when run from the console, prints a report that tells you what you need to make the grade cutoffs for CS 1331. Here's a sample program run:

```
$ java HowToPass "Exams: 90; Timed Labs: 80; Homeworks: 80, 0, 100"
Given your current scores:
Exams: [90]
TLs: [80]
HWs: [80, 0, 100]
and current average of 80.6
On remaining assignments you need:
a 98.0 average to finish with a 90.
a 79.5 average to finish with a 80.
a 60.9 average to finish with a 70.
a 42.4 average to finish with a 60.
```

It doesn't matter the order of your clauses, only that they begin with "Exams:", "Timed Labs:" or "Homeworks:" and are separated with semicolons. The hard parts of the `HowToPass` class are already implemented for you. You need to fill in the following methods:

3.1 `static double average(int ... nums)`

`average` should compute and return the arithmetic mean of `nums`. For example, if called with the arguments `10` and `20` `average` should return `15.0`.

3.2 `int[] extractScores(String label, String currentScores)`

`extractScores` should call `extractClause` to get the clause with the given `label`, then get the scores that come after the label and return an `int[]` with the scores. For example, if `extractScores` is called like this:

```
int[] homeworks = extractScores("Homeworks",
   "Exams: 90; Timed Labs: 80; Homeworks: 80, 0, 100");
```

Then the `homeworks` array would contain the `int` elements `80, 0, 100`.

3.3 `String extractClause(String label, String text)`

`extractClause` should return a substring of `text` that starts with `label` followed by a colon and ends with either a semicolon or the end of the string. For example, after the follwoing code:

```
String timedLabsClause = extractClause("Timed Labs",
   "Exams: 90; Timed Labs: 80; Homeworks: 80, 0, 100");
String homeworkClause = extractClause("Homeworks",
   "Exams: 90; Timed Labs: 80; Homeworks: 80, 0, 100");
```

`timeLabsClause` would be `"Timed Labs:  80"` and `homeworkClause` would be `"Homeworks:  80, 0, 100"`.

3.4 `public static void main(String ... args)`

The main method should instantiate a `HowToPass` object, passing the first command line argument to the constructor, then print the text returned by `HowToPass`'s `report` method to the console. The main method is very simple.

4 Tips

You may find the following `String` instance methods useful:

- `int indexOf(String text)` returns the index of the first occurence of `text` in the string on which it is called. For example, `"foo:bar".indexOf(":")` would return `3`.
- `String substring(begin, end)` returns a `String` which contains the characters on which `substring` is called beginning with the character at index `begin` and ending with the character at the index preceding `end`. For example, `"foo:bar".substring(3, 7)` would return `":bar"`.
- `String[] split(String delimiter)` returns an array of `String` elements from the `String` on which it is called, where the elements are separated by `delimiter`. For example, `"1, 2, 3".split(",")` would return a `String[]` with elements `"1"`, `" 2"`, and `" 3"`. Note the leading spaces.
- `String trim()` returns a copy of the `String` on which it is called, but with lead and trailing whitespace removed. For example, `" 2".trim()` woudl return `"2"`. Use `trim()` a lot.
- `boolean startsWith(String text)` returns `true` if the `String` on which it is called begins with `text`, `false` otherwise. For example, `"foo:bar".startsWith("foo")` would return `true`. `" foo:bar".startsWi` would return `false`. Read that last sentence carefully.

Additional tips:

- `Integer.parseInt(String text)` returns `int` value of the integer represented by `text`. For example, `Integer.parseInt("2")` returns the `int` value `2`. Note that `Integer.parseInt` is finnicky. See previous note on `String.trim()`.

5 Checkstyle

You must run checkstyle on your submission. The checkstyle cap for this assignment is **10** points.
Review the Style Guide and download the Checkstyle jar. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-6.2.1.jar *.java
Audit done. Errors (potential points off):
0
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off.
The Java source files we provide contain no Checkstyle errors. For this assignment, there will be a maximum of **10** points lost due to Checkstyle errors (1 point per error). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

6 Using the Submission Tool

Included with this timed lab is a handy-dandy submission tool. You will not submit to T-Square, but will use this instead. The commands are:

- `ant` or `ant compile` compiles your code. Basically just like running `javac HowToPass.java`, but you don't need to specify the java file. Just run `ant compile`
- `ant test` runs the provided JUnit tests. For this practice timed lab, the output will perfectly correspond to your hypothetical grade. For actual timed labs, output will roughly correspond - you can use this to see if you're doing something wrong, and get an idea of how well you're doing.
- `ant checkstyle` is a convenient way of running checkstyle without typing that obnoxious command.
- `ant run` will run the code.
- `ant submit` will actually submit your code to the GT Github repo. This will prompt you for your user name and password each time, but you can run it as many times as you want if you want to keep resubmitting as you make minor changes.

7 Confirm Your Submission

Your submission was pushed to a git repository on github.gatech.edu. After running `ant submit`, a new browser window should open at the repository, at https://github.gatech.edu/[your-username]/[your-username]-timedlab1