



COMPUTER SCIENCE 12B-1 (SPRING TERM, 2017) PROGRAMMING IN JAVA

PROGRAMMING ASSIGNMENT 1

Due Date: Friday February 1, 2017, 11:55pm

Overview:

This assignment will give you practice with arrays and producing an external output file. Turn in a file named `PersonalityTest.java`. You will also need the input file `personality.txt`. The assignment involves computing personality test data.

Background Information:

The Keirsey Temperament Sorter (<http://www.keirsey.com/>) is a personality test that involves answering 70 questions. Each question has two answer choices, which we will refer to as the "A" and "B" answer. The person taking the test is allowed to leave a question blank, in which case the answer will be recorded with a dash ("-").

The Keirsey test measures four independent dimensions of personality:

1. *Extrovert versus Introvert* (E vs I): what energizes you
2. *Sensation versus iNtuition* (S vs N): what you focus on
3. *Thinking versus Feeling* (T vs F): how you interpret what you focus on
4. *Judging versus Perceiving* (J vs P): how you approach life

Individuals are categorized as being on one side or the other for each dimension. The corresponding letters are put together to form a personality type. For example, if you are an Extrovert, iNtuitive, Thinking, Perceiving person then you are referred to as an ENTP. The "A" answers correspond to E, S, T, and J (the left-hand choices above). The "B" answers correspond to I, N, F, and P (the right-hand choices above). For each dimension, we determine a percentage of B answers the user gave for that dimension between 0 and 100, to indicate whether the person is closer to the "A" or "B" side.

Suppose that someone's answers are as follows (These are the answers given by "Betty Boop" later in this document).

Dimension	# of "A" answers	# of "B" answers	% of "B" answers	Result
Extrovert/Introvert	1	9	90%	I
Sensing/iNtuition	17	3	15%	S
Thinking/Feeling	18	2	10%	T
Judging/Perceiving	18	2	10%	J

We add up how many of each type of answer we got for each dimension. Then we compute the percentage of B answers for each dimension. Then we assign letters based on which side the person ends up on for each dimension. In the Extrovert/Introvert dimension, for example, Betty gave 9 "B" answers out of 10 total (90%), which means she is on the B side, which is "Introvert" or I. The overall percentages are (90, 15, 10, 10) which works out to a personality type of ISTJ.

Mechanics of the Personality Test:

Suppose that "Betty Boop" gave the following answers for the 70 questions, in order from 1 to 70:

BABAAAABAAAAAABAAAABBBAAAAAABAAAABABAABAAABABABABAAAAAABAAAAAABAAAAA

The questions are organized into 10 groups of 7 questions, with the following repeating pattern in each group:

1. The first question in each group is an Introvert/Extrovert question (questions 1, 8, 15, 22, etc).
2. The next two questions are for Sensing/iNtuition (questions 2 and 3, 9 and 10, 16 and 17, 23 and 24, etc).
3. The next two questions are for Thinking/Feeling (questions 4 and 5, 11 and 12, 18 and 19, 25 and 26, etc).
4. The next two questions are for Judging/Perceiving (questions 6 and 7, 13 and 14, 20 and 21, 27 and 28, etc).

In other words, if we consider the I/E to be dimension 1, the S/N to be dimension 2, the T/F to be dimension 3, and the J/P to be dimension 4, the map of questions to their respective dimensions would look like this:

122334412233441223344122334412233441223344122334412233441223344
BABAAAABAAAAAABAAAABBBAAAAAABAAAABABAABAAABABABABAAAAAABAAAAAABAAAAA

Notice that there are half as many Introvert/Extrovert questions as there are for the other three dimensions.

Program Behavior:

Your program will process a file of Keirsey test data. The file will contain line pairs, one per person. The first line has the person's name, and the second has the person's 70 answers (all "A", "B" or "-"). The "A" and "B" in the file can be upper or lowercase. A dash represents a question that was skipped. The format will match the following example:

Input file: personality.txt

Betty Boop
BABAAAABAAAAAABAAAABBBAAAAAABAAAABABAABAAABABABABAAAAAABAAAAAABAAAAA
Bugs Bunny
aabaabbabbbbaaaabaaaabaaaababbbaabaaaabaabbbbabaaaabaabaaaaabbbaaaabb
Han Solo
BA-ABABBB-bbbaababaaaabbbaabbbaabbabABBAABABBAABABAAAAABBABAAABBABAAB

Your program begins by asking for the input and output file names. If the input file does not exist, prompt again until a valid file name is typed. Each pair of lines from the input file is turned into a group of lines in the output file with the name, count of As and Bs for each dimension, % Bs for each dimension (rounded to the nearest whole percent), and personality type. You must exactly reproduce the following output format. If the person has the same number of As and Bs for a dimension, give them an "X" (as with Han Solo). Assume the input file has no errors and that nobody has skipped all questions for a dimension.

Log of execution (user input underlined):

```
Input file name: notfound.txt
File not found. Try again: foo.txt
File not found. Try again: personality.txt
Output file name: output.txt
```

Notice that you must re-prompt the user when an invalid input file name is entered. You can do this by calling the `exists` method of a `File` object. Use a `PrintStream` to write to the output file.

Resulting output file `output.txt`:

```
Betty Boop:
1A-9B 17A-3B 18A-2B 18A-2B
[90%, 15%, 10%, 10%] = ISTJ

Bugs Bunny:
8A-2B 11A-9B 17A-3B 9A-11B
[20%, 45%, 15%, 55%] = ESTP

Han Solo:
2A-8B 9A-9B 11A-9B 15A-5B
[80%, 50%, 45%, 25%] = IXTJ
```

Implementation Guidelines:

In this program you are transforming data from one form to another. The transformation of the original data into a personality type can be summarized by the following figure (using the data from "Han Solo"):

<pre>answers: "BA-ABABBB- bbbaababaaaaabbaaabbabABBAAABABBAABABAAAABBABAAABBABAAB"</pre>		
↓		
↓	<u>What is computed</u>	<u>Resulting output</u>
aCount:	{2, 9, 11, 15}	
bCount:	{8, 9, 9, 5}	2A-8B 9A-9B 11A-9B 15A-5B
↓		
bPercent:	{80, 50, 45, 25}	[80%, 50%, 45%, 25%]
↓		
type:	"IXTJ"	= IXTJ

Stylistic Guidelines:

You should have **at least three methods other than `main`** that perform some nontrivial part of the problem. Your `main` method should not directly perform file input/output.

For this assignment you are limited to language features in Chapters 1 through 7 of the textbook. Follow stylistic guidelines about removing redundancy, using proper data types, indentation, whitespace, identifier names, and commenting at the beginning of your program, on each method, and on complex sections of code.

Submission:

Your Java source code (program) should be submitted via Latte.

For late policy check the syllabus.

Grading:

You will be graded on

- **External Correctness:** The output of your program should match exactly what is expected. Programs that do not compile will not receive points for external correctness.
- **Internal Correctness:** Your source code should follow the stylistic guidelines shown in class. Also, remember to include the comment header at the beginning of your program.
- **One-on-one interactive grading:** By the end of the day that the assignment is due, please make an appointment with your TA for an interactive 10-15 minute grading session. You will receive an email notifying you of the name of the TA who has been assigned to you for this assignment with further instructions on setting up the appointment. (You will be meeting with a different TA for each assignment). One-on-one interactive grading will help you improve your programming skills and avoid repeating mistakes from one assignment to the next.