CSE 142, Summer 2019
Programming Assignment #7: Personality Test (20 points)
Due: Tuesday, August 13, 2019, 11:59 PM

This assignment will give you practice with input files, arrays, and producing an external output file. Turn in a file named `Personality.java`. You will also need the two input files `personality.txt` and `bigdata.txt` from the course web site. Save these files in the same folder as your program.

You are going to write a program that processes an input file of data for a personality test known as the Keirsey Temperament Sorter. (You can find out more about the Keirsey Temperament Sorter at http://www.keirsey.com.) The Keirsey personality test involves answering 70 questions each of which have two answers. The Keirsey test measures four independent dimensions of personality:

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

Individuals are categorized as being on one side or the other of each of these dimensions. 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. Usually the letter used is the first letter of the corresponding word, but notice that because the letter "I" is used for "Introvert", the letter "N" is used for "iNtuition."

If you are interested in taking the personality test yourself, you will find a link from the class webpage to an online form with the 70 questions. We will release a data file called `bigdata.txt` that includes data from students in the class.

Program Overview:

The input file will contain a series of line pairs, one per person. The first line will have the person's name (possibly including spaces) and the second line will have a series of 70 answers all in a row (all either "A", "B" or "-"). Your job is to compute the scores and overall result for each person and to report this information to an output file.

The Keirsey test involves 70 questions answered either A or B. The A answers correspond to extrovert, sensation, thinking and judging (the left-hand answers in the list above). The B answers correspond to introvert, intuition, feeling and perceiving (the right-hand answers in the list above). For each of these dimensions, we determine a number between 0 and 100 and indicate whether they were closer to the A side or the B side. The number is computed by figuring out what percentage of B answers the user gave for that dimension (rounded to the nearest integer).

Let's look at a specific example. Suppose that someone's answers divide up as follows:

Dimension	# of A answers	# of B answers	% B	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

These numbers correspond to the answers given by the first person in the sample input file ("Betty Boop"). We count up the number of each type of answer from the user for each of the four dimensions. 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, the person gave 9 "B" answers out of 10 total, which is 90%, which means they end up on the B side which is "Introvert" or I. In the Sensing/iNtuition dimension the person gave 3 "B" answers out of 20 total, which is 15%, which means they end up on the A side with is "Sensing" or S. The overall scores for this person are the percentages (90, 15, 10, 10) which works out to a personality type of ISTJ.

Some people will end up with a percentage of 50 in one or more dimensions. This represents a tie, where the person doesn't clearly fall on either side. **In this case we use the letter "X"** to indicate that the person is in the middle for this particular dimension. The last two entries in the sample input file end up with X's in their personality type.

Program Behavior:

Take a moment to compare the sample input file and the sample output file and you will see that each pair of lines in the input file produces a single line of output in the output file that reports the person's name, the list of B percentages for each personality dimension, and the personality type. **You are required to exactly reproduce the format of this output file. You are also required to exactly reproduce the sample log of execution.** Here, you will provide a short introduction and then ask the user for the names of the input file and output file.

To count the number of A and B answers for each dimension, you need to know something about the structure of the test. *You will get the best results if you take the test without knowing about the structure, so you might want to take the test **before** you read what follows.* The test has 10 groups of 7 questions with a repeating pattern in each group of 7 questions. The first question in each group is an Introvert/Extrovert question (questions 1, 8, 15, 22, etc). The next two questions are for Sensing/iNtuition (questions 2, 3, 9, 10, 16, 17, 23, 24, etc). The next two questions are for Thinking/Feeling (questions 4, 5, 11, 12, 18, 19, 25, 26, etc). And the final two questions in each group are for Judging/Perceiving (questions 6, 7, 13, 14, 20, 21, 27, 28, etc). Notice that there are half as many Introvert/Extrovert questions as there are for the other three dimensions. The seventy letters in the input file appear in question order (first letter for question 1, second letter for question 2, third letter for question 3, etc).

Remember that the user might leave a question blank, in which case you will find a dash in the input file for that question. Dash answers are not included in computing the percentages. For example, if for one of the dimensions you have 6 A answers, 9 B answers and 5 dashes, you would compute the percentage of B answers as 9 of 15, or 60%.

Implementation Guidelines, Hints, and Development Strategy:

The main purpose of this assignment is to demonstrate your understanding of arrays and array traversals with `for` loops. Therefore, you should use arrays to store the various data for each sequence. In particular, **your answer counts and answer percentages should be stored using arrays**. Additionally you should **use arrays and `for` loops** to transform the data from one form to another as follows:

- from the answer string to A/B counts; then
- from A/B counts to A/B percentages

You begin with a `String` that has 70 characters in it. You should then convert that `String` into two sets of counts: the number of A answers for each dimension and the number of B answers for each dimension. You should next convert those counts into a set of percentages. Then, you finally convert the set of percentages into a `String` that represents the personality type. If you work through the process outlined here step by step, we expect that the program will be easier to implement. The following diagram summarizes the different transformations:

Data	**Example**
String (70 chars)	`"BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA"`
↓	↓
counts of A/B	`{1, 17, 18, 18}, {9, 3, 2, 2}`
↓	↓
B percentages	`{90, 15, 10, 10}`
↓	↓
Personality type	`"ISTJ"`

Recall that you can print any array nicely formatted by using the method `Arrays.toString`. For example:

```
int[] numbers = {10, 20, 30, 40};
System.out.println("my data is " + Arrays.toString(numbers); // my data is [10, 20, 30, 40]
```

Notice that the letters "A" and "B" in the sample input file sometimes appear as uppercase letters and sometimes appear as lowercase letters. **Your program must recognize them correctly in either case.**

You should round percentages to the nearest integer when printing to the output file. You can use the `Math.round` method to do so, but you will have to cast the result to an `int` afterwards since the `Math.round` method returns a double:

```
int percent = (int) Math.round(percentage);
```

You should generate an output file by constructing an object of type `PrintStream` and writing to it in the same way you write to `System.out` (with `print` and `println` statements). See section 6.4 of the book for examples. It is a good idea to send your output to `System.out` while you are developing your program and send it to the output file using a `PrintStream` object only after you have thoroughly tested your program.

You should read the user's answers from the input file using `nextLine()`. This will read an entire line of input and return it as a `String`.

You may assume that the input file has no errors and is in the format described. In particular, you may assume that the file exists, that it is composed of pairs of lines, and that the second line in each pair will have exactly 70 characters that are either A, B or dash (although the A's and B's might be either uppercase or lowercase or a combination). You may also assume that nobody has zero answers for a given dimension (it would be impossible to determine a percentage in that case).

The sample input and output files provide just a few simple examples of how this program works. We will be using a set of much more extensive files to test your program. As mentioned earlier, we will include data from people in the class to make this file.

Style Guidelines:

Your program is likely to have the number 4 in several places because of the four dimensions of this test. **You should introduce a class constant to make this more readable instead of using 4 itself.** It won't be possible, however, to change this constant to some other number and have the program function properly. The constant is helpful for documentation purposes, but it won't make the program particularly flexible.

We will grade your method structure strictly on this assignment. Use at least **four nontrivial methods** besides `main`. These methods should use parameters and returns, parameters and returns of type array, when appropriate. The methods should be well-structured and avoid redundancy. No one method should do too large a share of the overall task. The textbook's case study at the end of Chapter 7 is a good example of a larger program with methods that pass arrays as parameters. In particular, we require that you have **one method to compute the A/B counts** for a single person and **one method to compute the personality type** for a single person.

Your `main` method should be a concise summary of the overall program. It is okay for `main` to contain some code such as `println` statements. But `main` should not perform too large a share of the overall work itself, such as examining each character of an input line. Also avoid "chaining," when many methods call each other without ever returning to `main`.

We will also check strictly for redundancy on this assignment. If you have a very similar piece of code that is repeated several times in your program, eliminate the redundancy by creating a method, by using `for` loops over the elements of arrays, by factoring `if/else` code as described in section 4.3 of the textbook, and/or other approaches.

Since arrays are a key component of this assignment, part of your grade comes from using arrays properly. For example, you should reduce redundancy by using **traversals** over arrays (`for` loops over the array's elements). This is preferable to writing out a separate statement for each array element (a statement for element `[0]`, then another for `[1]`, then for `[2]`, etc.). Also carefully consider how arrays should be passed as parameters and/or returned from methods as you are decomposing your program. Recall that arrays use *reference semantics* when passed as parameters, meaning that an array passed to a method can be modified by that method and the changes will be seen by the caller.

You are limited to features in Chapters 1 through 7 of the textbook. Follow all previous style guidelines for indentation, names, variables, types, line lengths, and comments (at the beginning of your program, on each method, and on complex sections of code).

Input file personality.txt

```
Betty Boop
BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA
Snoopy
AABBAABBBBBABABAAAAABABBAABBAAAABBBAAABAABAABABAAAABAABBBBAAABBAABABBB
Bugs Bunny
aabaabbabbbaaaabaaaabaaaaababbbaabaaaabaabbbbabaaaabaabaaaaaabbaaaaabb
Daffy Duck
BAAAAA-BAAAABABAAAAAABA-AAAABABAAAABAABAA-BAAABAABAAAAAABA-BAAABA-BAAA
The frumious bandersnatch
-BBaBAA-BBbBBABBBBA-BaBBBBBbbBBABBBBBBABB-BBBaBBABBBBBBB-BABBBBBBBBBB
Minnie Mouse
BABA-AABABBBAABAABA-ABABAAAB-ABAAAAAA-AAAABAAABAAABAAAAAB-ABBAAAAAAAAA
Luke Skywalker
bbbaaabbbbaaba-BAAAABBABBAAABBAABAAB-AAAAABBBABAABABA-ABBBABBABAA-AAAA
Han Solo
BA-ABABBB-bbbaababaaaabbaaabbaaabbabABBAAABABBAAABABAAAABBABAAABBABAAB
Princess Leia
BABBAAABBBBAAABBA-AAAABABBABBABBAAABAABAAABBBA-AABAABAAAABAAAAABABBBAA
```

Output file for personality.txt

```
Betty Boop: [90, 15, 10, 10] = ISTJ
Snoopy: [30, 45, 30, 70] = ESTP
Bugs Bunny: [20, 45, 15, 55] = ESTP
Daffy Duck: [100, 6, 20, 6] = ISTJ
The frumious bandersnatch: [86, 95, 75, 78] = INFP
Minnie Mouse: [67, 28, 32, 5] = ISTJ
Luke Skywalker: [89, 61, 26, 25] = INTJ
Han Solo: [80, 50, 45, 25] = IXTJ
Princess Leia: [80, 50, 50, 5] = IXXJ
```

Sample log (user input bold and underlined)

```
This program processes a file of answers to the
Keirsey Temperament Sorter.  It converts the
various A and B answers for each person into
a sequence of B-percentages and then into a
four-letter personality type.

input file name? personality.txt
output file name? output.txt
```