CSE 142, Spring 2011

Programming Assignment #7: Personality Test (20 points)

Due: Tuesday, May 24, 2011, 9:00 PM

This assignment focuses on arrays and file/text processing. Turn in a file named `PersonalityTest.java`. You will also need input file `personality.txt` from the course web site. Save this file in the same folder as your program.

The assignment involves processing data from a personality test. There is a link on the course web site where you can take the personality test yourself. Student answers will be included in a data file distributed to the whole class.

Background About the Personality Test:

The Keirsey Temperament Sorter (keirsey.com) is a test that measures four independent dimensions of your 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

The test involves answering a 70-question survey. Each question has two answer choices, which we'll call "A" and "B". The person taking the test can leave a question blank, in which case the answer will be recorded with a dash (`-`).

Individuals are categorized as being on one side or the other for each dimension, and 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 the left-hand choices above: E, S, T, and J. The "B" answers correspond to the right-hand choices above: I, N, F, and P. For each dimension, we compute 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 the survey, in order from 1 to 70:

```
BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA
```

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

1. The first one 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 *(spaces added for emphasis)*:

```
index 0123456 7890123 4567890 1234567 8901234 5678901 2345678 9012345 6789012 3456789
dimen 1223344 1223344 1223344 1223344 1223344 1223344 1223344 1223344 1223344 1223344
char  BABAAAA BAAAAAA ABAAAAB BAAAAAA BAAAABA BAABAAA BABABAA BAAAAAA BAAAAAA BAAAAAA
```

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

Input Data:

The personality data consists of line pairs, one per person. The first line has the person’s name, and the second has the person's 70 answers (`'A'`, `'B'` or `'-'`). Notice that the A or B can be either upper or lowercase. A dash represents a question that was skipped by that person, as seen in Han Solo's data below.

Input file `personality.txt` (partial):

```
Betty Boop
BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA
Bugs Bunny
aabaabbabbbaaaabaaaabaaaaababbbaabaaaabaabbbabaaaabaabaaaaaabbaaaaabb
Luke Skywalker
bbbaaabbbbaaba-BAAAABBABBAAABBAABAAB-AAAAABBBABAABABA-ABBBABBABAA-AAAA
Han Solo
BA-ABABBB-bbbaababaaaabbaaabbaaabbabABBAAABABBAAABABAAAABBABAAABBABAAB
```
...

Program Behavior:

Your program begins with an introduction of your own creation, such as an explanation of the personality test or a fact about your own type. Next your program asks for the input file name to process. You may assume that the user will type the name of a valid existing file that is in the format described above. Your program opens this file and processes its data to compute information about each person's personality type.

Each pair of lines from the input file is turned into a group of lines in the console output with the name, count of As and Bs for each dimension, % Bs for each dimension (rounded to the nearest percent), and personality type. Questions left blank (indicated by a dash, `'-'`) do not contribute to the count of As/Bs, nor to the percent Bs for that dimension. If the person has the same number of As and Bs for a dimension, give them an "X", as with Han Solo below. Assume that the input file has no errors or illegal data, and that nobody has skipped all questions for a dimension (it would be impossible to determine a percentage of Bs in such a case).

Log of execution (partial; user input underlined):

```
<< your introduction message here >>

Input file name: personality.txt

Betty Boop:
answers: [1A-9B, 17A-3B, 18A-2B, 18A-2B]
percent B: [90, 15, 10, 10]
type: ISTJ

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

Luke Skywalker:
answers: [1A-8B, 7A-11B, 14A-5B, 15A-5B]
percent B: [89, 61, 26, 25]
type: INTJ

Han Solo:
answers: [2A-8B, 9A-9B, 11A-9B, 15A-5B]
percent B: [80, 50, 45, 25]
type: IXTJ
```
...

The input file name used in the log below is `personality.txt`, but the input file could have a different name. You should not place the string `"personality.txt"` in your program's code.

Implementation Guidelines and Hints:

A major purpose of this assignment is to test your understanding of arrays. You should use arrays to store the data for each of the four dimensions of the personality test, and to transform data from one form to another as follows:

- from the original 70 A/B answers to counts of As and Bs, grouped by dimension;
- from counts to percentages of Bs; and
- from percentages to a four-letter personality type string.

These transformations are summarized by the following diagram using Han Solo's data:

Answers:	`"BA-ABABBB-bbbaababaaaabbaaabbaaabbabABBAAABABBAAABABAAAABBABAAABBABAAB"`	
↓	<u>What is computed</u>	<u>Output</u>
A counts:	`{2, 9, 11, 15}`	
B counts:	`{8, 9, 9, 5}`	`Answers: [2A-8B, 9A-9B, 11A-9B, 15A-5B]`
↓		
B percent:	`{80, 50, 45, 25}`	`Percent B: [80, 50, 45, 25]`
↓		
Type:	`"I" "X" "T" "J"`	`Type: IXTJ`

To count As and Bs, read data from the input file one line at a time from your `Scanner`. Analyze each character within the overall line `String` as a `char` value using the line string's `charAt` method.

The process of converting the 70 answers into counts of As and Bs is one of the most challenging aspects of this program. Look at the lecture and section examples that perform tallying. Pay extra attention to this code to minimize redundancy.

For debugging it can be helpful to print your arrays. Recall that the method `Arrays.toString` converts an array to a printable `String`. For example, to print the contents of an array named `counts`, you could write the following code:

```
System.out.println("My counts are " + Arrays.toString(counts));
```

You must display percentages of Bs as integers. Use `Math.round` or `printf` to round numbers to the nearest integer. If you use `Math.round`, you must cast its result to type `int` if you want to store the result in an `int` variable or array:

```
double d = 3.14;
int n = (int) Math.round(d);
```

Style Guidelines:

Use a **class constant** in your program representing the number of dimensions in the test, as an `int` (with a value of 4). It will not be possible to change this constant and have the program adjust, but it makes the code more readable.

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, including arrays, as appropriate. The methods should be well-structured and avoid redundancy. No one method should do too large a share of the overall task. The Case Study in book section 7.5 is a good example of a larger program with methods that pass arrays as parameters.

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, or for `main` to open files and perform file I/O. But the `main` method should not perform too large a share of the overall work itself, such as examining each of the 70 characters of a person's answers to the personality test. Also avoid "chaining," when many methods call each other without ever returning to `main`. For reference, our solution is around 100 lines long and has 5 methods besides `main`, though you don't need to match this.

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 such as by creating a method, by using `for` loops over the elements of arrays, and/or by factoring `if/else` code as described in section 4.3 of the textbook.

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