University of Washington
Computer Science & Engineering 142 (Computer Programming I), Summer 2005

Programming Assignment #7 (Personality Test)
Due: Tuesday, 8/9/2005, 6:00 PM

This assignment will give you practice with arrays and producing an external output file. Turn in a Java class named `PersonalityTest` stored in a file named `PersonalityTest.java`. You will also need the input file `personality.txt` and support file `Scanner.java` from the course web site.

Background Information:

The Keirsey Temperament Sorter (http://www.keirsey.com/) is a personality test that involves answering 70 questions, each of which has two answer choices. We will refer to the choices as the "A" answer and the "B" answer. The person taking the test is allowed to leave a question blank, in which case his/her 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 of each of these dimensions. The corresponding letters are put together to form a personality type. For example, if you are an Extravert, iNtuitive, Thinking, Perceiving person then you are referred to as an ENTP.

The "A" answers correspond to extravert, sensation, thinking and judging (the left-hand choices in the list above). The "B" answers correspond to introvert, intuition, feeling and perceiving (the right-hand choices in the list above). For each of these dimensions, we determine a percentage between 0 and 100, to indicate whether the person is closer to the "A" side or the "B" side. The number is calculated by figuring out what percentage of B answers the user gave for that dimension (rounded to the nearest whole number).

Suppose that someone's answers divide up as follows (These are the answers given by "Betty Boop").

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 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, Betty gave 9 "B" answers out of 10 total (90%), which means she is on the B side, which is "Introvert" or I. In the Sensing/iNtuition dimension, Betty gave 3 "B" answers out of 20 total (15%), which means she is on the A side, which 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.

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. Submit your answers before the assignment is due, if you want to have them included in a big data file that will be distributed to the whole class.

Mechanics of the Personality Test:

(If you choose to take the personality test yourself, you will get the best results if you take the test without knowing about its structure. Therefore, you might want to take the test first before you read what follows.)

Suppose that the example person named "Betty Boop" gave the following answers for the 70 questions, in order:

```
BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA
```

Consider question 1 to be the first and question 70 to be the last. 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 <u>two</u> questions are for Sensing/iNtuition (questions 2 and 3, 9 and 10, 16 and 17, 23 and 24, etc).
3. The next <u>two</u> questions are for Thinking/Feeling (questions 4 and 5, 11 and 12, 18 and 19, 25 and 26, etc).
4. And the final <u>two</u> questions in each group 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:

```
1223344122334412233441223344122334412233441223344122334412233441223344
BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA
```

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

Program Behavior:

Your task is to write a Java program that processes an input file of data for the Keirsey personality test. The input file will contain a series of line pairs, one pair per person. The first line will have the person's name (possibly including spaces), and the second line will have a series of 70 letters all in a row (all either "A", "B" or "-"). The format will match the following example:

Example 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-BABBBBBBBBBBB
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
```

Notice that the letters "A" and "B" in the sample input file might appear as either uppercase or lowercase letters. Your program must recognize them in either case. Also notice the dashes that represent a question that the person skipped.

If the file does not exist, you should print an error message and prompt again for the file name. You may assume that the input file has no errors. In particular, you may assume that the file 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. You may also assume that nobody has skipped ALL of the questions for a given dimension (it would be impossible to determine a percentage in that case).

Your job is to read the file name from the user, compute the scores and overall result for each person, and to output a report of this information into an output file. Your program begins by asking for the file name that stores the personality data. Then each pair of lines in the input file is turned into a group of 3 lines in the output file that reports the person's name, the count of how many A and B answers the person gave for each dimension, the list of percentages of Bs for each dimension, and the personality type. You are required to exactly reproduce the format of this output. **If the person has the same number of A and B answers for a given dimension, give them an "X" for that dimension** (as with Princess Leia, shown below).

Example log of execution (user input underlined):

```
What is the name of the input file? notfound.txt
File not found.  Please try again.
What is the name of the input file? foo.txt
File not found.  Please try again.
What is the name of the input file? personality.txt
What is the name of the output file? output.txt
```

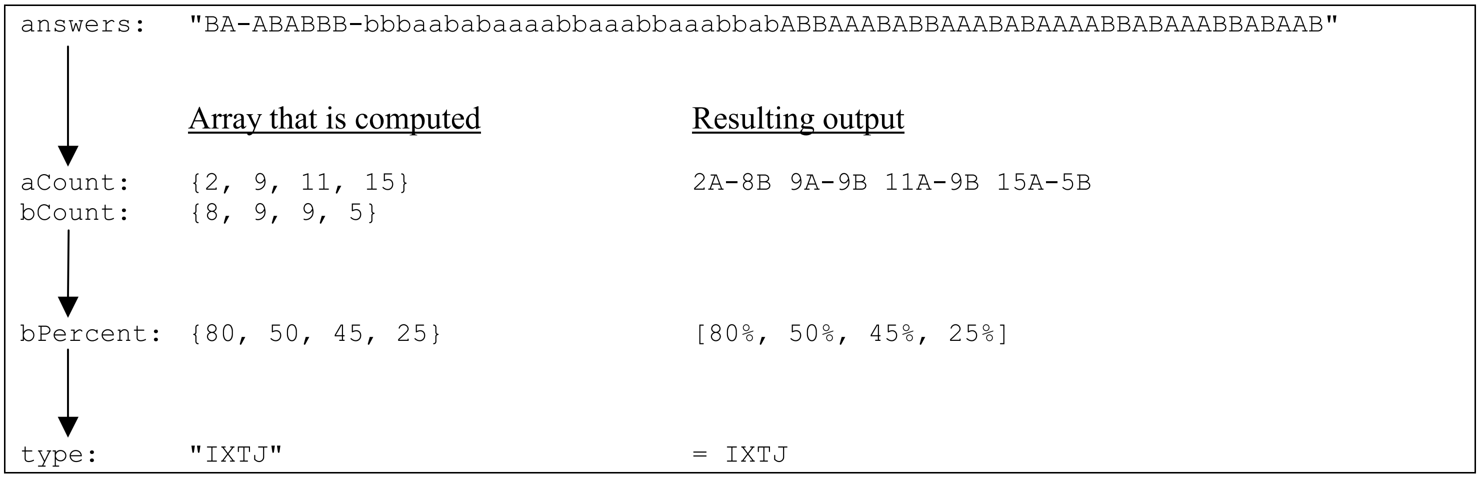
Example contents of output file `output.txt`:

```
Betty Boop:
    1A-9B 17A-3B 18A-2B 18A-2B
    [90%, 15%, 10%, 10%] = ISTJ
Snoopy:
    7A-3B 11A-9B 14A-6B 6A-14B
    [30%, 45%, 30%, 70%] = ESTP
Bugs Bunny:
    8A-2B 11A-9B 17A-3B 9A-11B
    [20%, 45%, 15%, 55%] = ESTP
Daffy Duck:
    0A-10B 16A-1B 16A-4B 17A-1B
    [100%, 6%, 20%, 6%] = ISTJ
The frumious bandersnatch:
    1A-6B 1A-19B 5A-15B 4A-14B
    [86%, 95%, 75%, 78%] = INFP
Minnie Mouse:
    3A-6B 13A-5B 13A-6B 18A-1B
    [67%, 28%, 32%, 5%] = ISTJ
Luke Skywalker:
    1A-8B 7A-11B 14A-5B 15A-5B
    [89%, 61%, 26%, 25%] = INTJ
Han Solo:
    2A-8B 9A-9B 11A-9B 15A-5B
    [80%, 50%, 45%, 25%] = IXTJ
Princess Leia:
    2A-8B 10A-10B 9A-9B 19A-1B
    [80%, 50%, 50%, 5%] = IXXJ
```

Implementation Guidelines:

One of the things to keep in mind for this program is that you are transforming data from one form to another. You start with a String that has 70 characters in it. You convert the answers in that String into two arrays of 4 counters (how many A answers for each dimension, how many B answers for each dimension). You convert that into an array of 4 percentages of Bs. And you finally convert that into a String that represents the personality type. If you work through this step by step, the problem will be easier to solve.

The transformation of the original answer data into a personality type can be roughly summarized by the following figure (which uses the data from "Han Solo" from the preceding input file):



```
answers:   "BA-ABABBB-bbbaababaaaabbaaabbaaabbabABBAAABABBAAABABAAAABBABAAABBABAAB"

           Array that is computed            Resulting output

aCount:    {2, 9, 11, 15}                    2A-8B 9A-9B 11A-9B 15A-5B
bCount:    {8, 9, 9, 5}

bPercent:  {80, 50, 45, 25}                  [80%, 50%, 45%, 25%]

type:      "IXTJ"                            = IXTJ
```

We suggest that you start this program by writing the code to read the input file. Initially, make your program print its output to the console, rather than to the output file, for easier debugging. Write the code that prints each person's name followed by a colon. Next, write the code that passes over the person's 70 answers and counts the number of As and Bs for each personality dimension. Put the A and B counts into two arrays of size 4. You can then print this information as shown in the previous sample file output. Once you have this output working correctly, you can convert your counts of As and Bs into a new array of percentages of Bs for each dimension.

Read each line from the `Scanner` by calling its `nextLine` method. This will read an entire line of input and return it as a string. You may use the string's `charAt` method to get the individual characters of this string, or you can convert the string into an array of characters by calling the string's `toCharArray` method. Java has a method `Math.round` that will help you to round percentages to the nearest whole number.

You must re-prompt the user when an invalid input file name is entered (such as an input file that does not exist or cannot be read). You can test for such an invalid file by either calling the methods of `File` objects as described in book section 6.2.3, or by using a `try/catch` statement as described in book section 6.5. You do not need to handle any errors related to the output file; you may assume that the user will type a valid output file name that can be opened for writing. Use a `PrintStream` to write to the output file as described in book section 6.4.

The sample input and output files provide just a few simple examples of how this program works. We will be using a much more extensive file for testing your program. We will include data from people in the class to make this file. We will make the data file and its output available to students at the end of the coming weekend.

You may assume that the input file has no errors. In particular, you may assume that the file 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 As and Bs might be in either uppercase form or lowercase form 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).

Stylistic Guidelines:

A major portion of this assignment is to demonstrate your understanding of arrays. Therefore, you should use arrays to store the various data for each of the four dimensions of the personality test.

We will be grading on your appropriate use of control structures like loops and if/else statements, your ability to avoid redundancy and your ability to break down the overall problem into methods that each solve part of the overall problem. No one method should be overly long. The `main` method in particular should not directly perform file input/output; perform these tasks in other methods. You should be able to come up with at least **three** different methods other than `main` that each perform some nontrivial part of the problem.

You should use **a global constant for the number of dimensions (4)** in the personality test. It will not be possible to change this constant to some other number and have the program function properly, but the constant is helpful for documentation purposes, even though it will not make the program particularly flexible.

You should keep in mind the ideas we have been stressing all quarter. You don't want to have redundant code. You don't want to have any one method be overly long. You want to break the problem down into logical sub-problems so that someone reading your code can see the sequence of steps it is performing.

You are required to properly indent your code and will lose points if you make significant indentation mistakes. See section 2.5.3 of the book for an explanation and examples of proper indentation. You should also use white space to make your program more readable, such as between operators and their operands, between parameters, and blank lines between groups of statements or methods.

Give meaningful names to methods and variables in your code. Follow Java's naming standards about the format of ClassNames, methodAndVariableNames, and CONSTANT_NAMES. Localize variables whenever possible -- that is, declare them in the smallest scope in which they are needed.

Include a comment at the beginning of your program with basic information and a description of the program and include a comment at the start of each method. Also put brief comments inside the methods explaining the more complex sections of code.

Submission and Grading:

Name your file `PersonalityTest.java` and it them in electronically from the "Assignments" link on the course web page. You do not have to turn in `Scanner.java`. This assignment is worth 20 points total.

Part of your program's score will come from its "external correctness." External correctness measures whether your log of execution and output file match exactly what is expected, including identical prompting for user input, identical console output, and identical `output.txt` file output.

The rest of your program's score will come from its "internal correctness." Internal correctness measures whether your source code follows the stylistic guidelines specified in this document. This includes using `while` or `do/while` loops to capture indefinite repetition, using `for` loops to capture definite repetition, representing the structure and redundancy of the program using appropriate parameterized static methods with return values as needed, using global constants to replace otherwise "magic number" values, commenting, naming identifiers, and indentation of your source code.