CSE 142: Computer Programming I — Spring 2021

Take-home Assessment 7: Personality — *due May 25, 2021 11:59pm*

This assignment will assess your mastery of the following objectives:

- Write a functionally correct Java program to produce specified console and file output.
- Use `Scanner` and `File` to read input from a file.
- Use `PrintStream` and `File` to write output to a file.
- Use arrays to store and manipulate collections of related information.
- Use methods to manage information flow and add structure to programs.
- Follow prescribed conventions for spacing, indentation, naming methods, and header comments.

Background

Note: You do not need to read this section to complete the assessment, but it provides some helpful context that may make the assessment easier to understand.

In this assessment, you will be writing a program to process the results of a personality test known as the Keirsey Temperament Sorter. The Keirsey personality test involves answering 70 questions by choosing one of 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 can do so **in this survey**. We will release a data file called `bigdata.txt` that includes data from students and staff in the class.

Sample Console Output

```
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
```

Program Behavior

Overview

This program will process an input file containing the results of a personality test for a number of people and determine each person's Keirsey personality type. The program will begin by printing a short introduction, and then asking the user for an input file to read and an output file to print results to. The program will then read the given input file and process the responses to the personality test in that file, printing the results to the specified output file. Only the introduction and file prompting) should be printed to the console.

Almost all of your output on this assessment will go to a file, NOT the console.

Test Format

The Keirsey test involves 70 questions, each of which can be answered either "A" or "B" (case-insensitively). Each question related to one of the four dimensions (see below), and each answer corresponds to one option or the other for that dimension. The "A" answers correspond to extrovert, sensation, thinking, and judging (the left-hand types in the list above), whereas the "B" answers correspond to introvert, intuition, feeling, and perceiving (the right-hand types in the list above). Questions can also be skipped, indicated by a response of "-".

Sample Input File (personality.txt)

```
Betty Boop
BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA
Snoopy
AABBAABBBBBABABAAAAABABBAABBAAAABBBAAABAABAABABAAAABAABBBBAAABBAABABBB
Bugs Bunny
aabaabbabbbaaaabaaaabaaaaababbbaabaaaabaabbbbabaaaabaabaaaaaabbaaaaabb
Daffy Duck
BAAAAA-BAAAABABAAAAAAABA-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
```

The 70 questions in the Keirsey test are organized in ten groups of seven questions each. The first question in each group (i.e. questions 1, 8, 15, etc.) is related to the Extrovert/Introvert dimension. The next two questions in each group (questions 2 and 3, 9 and 10, 16 and 17, etc.) are related to Sensation/iNtuition. The next two questions (questions 4 and 5, 11 and 12, etc.) are related to Thinking/Feeling. The final two questions in each group (questions 6 and 7, 13 and 14, etc.) are related to Judging/Perceiving. Each person's responses will be provided in question order. That is, the first response is from question 1, the second response from question 2, and so on. As stated above, each response will be either "A" or "a" (for the "left-hand" response), "B" or "b" (for the "right-hand" response), or "-" (if the person skipped that question).

Notice that the A's and B's can be either upper- or lower-case!

Determining Personality Type

For each of the four dimensions, we determine the percentage of responses the person gave for that dimension that were "B" (rounded to the nearest integer). Smaller percentages indicate the person is closer to the "A" personality type for that dimension, while larger percentages indicate the person is closer to the "B" side. For each person in the input file, your program should output the person's name, the percentage of responses they gave for each dimension that were "B", and their overall personality type. See the sample output file below for the specific format.

Make sure that the format and structure of both your console output and your file output ***exactly*** match the given logs.

For example, consider the first person in the sample input file `personality.txt`, named Betty Boop. This person's responses are as follows:

Dimension	# of A responses	# of B responses	%age B	Result
Extrovert/Introvert	1	9	90%	I
Sensation/iNtuituion	17	3	15%	S
Thinking/Feeling	18	2	10%	T
Judging/Perceiving	18	2	10%	J

For each dimension, we count the total number of responses given, then determine the percentage of those responses that were "B". Based on this percentage, we assign the person to one side or the other of the dimension. If the percentage is less than 50, we assign them to the "A" side (the first option in each pair); if the percentage is greater than 50, the person is assigned to the "B" side (the second option in each pair). So, for example, Betty Boop has 90% "B" responses in the E/I dimension, so she is assigned to Introvert. On the other hand, she has only 15% "B" responses in the S/N dimension, so she is assigned to Sensing. It is also possible for a person to have exactly 50% "B" responses. In this case, the person is evenly split between the two sides of the dimension, and they are assigned "X" for that dimension. (See the sample input file for examples of this.)

Sample Output File

```
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
```

See the sample input and output files for more examples.

Questions that were skipped (i.e. those that have a response of "-") are considered as *neither* A nor B responses and should not be included at all in the percentage calculation. So, for example, if a person skipped the first question on the test, they would only have 9 responses for the E/I dimension, rather than 10. If 6 of their responses were B, their percentage of B responses would be $6/9 \approx 67\%$, not 60%.

When a person skips a question, it is like that question doesn't exist. Not all people will have the same number of responses for each dimension.

Input File Format

The input file will consist of pairs of lines, with each pair representing a single person. The first line in the pair will contain a person's name, which may include spaces. The second line will contain a sequence of 70 responses to the personality test questions. Each response will be either `"A"`, `"B"`, or `"-"`, and the responses will appear as a sequence of characters with no spaces. (See the sample input file for more details.)

You may assume that the input file is valid according to the format described above. Specifically, you may assume that:

- the file exists
- the file contains an even number of lines
- the second line in each pair contains exactly 70 characters, all of which are either "A", "a", "B", "b", or "-"
- each person's responses includes at least one non-dash response for each of the four dimensions

Development Strategy

As always, it is recommended that you approach this assessment in smaller pieces rather than all at once. We recommend the following strategy:

(a) **Compute counts**: Compute the A/B counts for each dimension from a single line of responses and print the results to the console (consider hard-coding the responses at this point)

(b) **Compute B percentages**: Compute the percentage of 'B' responses for each dimension and print to the console

(c) **Compute personality type**: Find the personality type and print to the console

(d) **File output**: Prompt for the output file name and print results to that file (still using a single line of input)

(e) **File input**: Prompt for the input file and read a single person (name and responses) from the file

(f) **Multiple people**: Read names and responses for multiple people from the input file

Hints

The string returned from `Arrays.toString` matches the format in which you should output arrays in your program.

The following suggestions and hints may help you be more successful on this assessment:

- You may find it easier to initially print all output to the console (`System.out`) while you are developing your program.
- You may want to initially "hard-code" the input and output filenames; in other words, you may want to just use fixed file names in your code rather than prompting the user to enter the file names.
- You can print arrays in a nice format using the `Arrays.toString` method. Remember that this method *returns* a `String`, but does not print anything on its own– you will need to print the result using one of the usual methods.
- Percentages should be rounded to the nearest integer. You can do this using the `Math.round` method, but this method returns a `double`, so you will potentially need to cast the result to an `int`. You can do this using code like the following:

  ```
  int roundedPercent = (int)Math.round(percentage);
  ```
- You should read *all* file and user input using the `Scanner` class and the `nextLine()` method.
- You should produce file output using the `PrintStream` class as described in class and in the textbook.

Implementation Guidelines

Required Methods

To receive full credit, your program must include the following methods:

- A method to compute the count of A and B responses for a single user
- A method to determine the final personality type for a single user

Your program *must* include these two methods, plus at least two others.

Each of these methods must perform *only* the tasks indicated, and must not contain any repeated logic. Your program must also include at least two (2) other non-trivial methods besides `main` besides these two. (Therefore, your program should have a total of at least four (4) non-trivial methods.) Each method should perform a single, coherent task and not do too much work.

Using Arrays

You *must* use arrays and traversals for these parts of your program to receive full credit.

You are **required** to use arrays to store and process most of the data throughout your program. Specifically, you must use arrays to store the following data:

- the A/B response counts for each dimension
- the percentage of B responses for each dimension

You must also use array traversals (via `for` loops) to process the data and arrays and transform it from one form to another. See the diagram below for the steps you should take to process and transform the data throughout your program:

Data	**Example**
String (70 characters)	`"BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA"`
↓	↓
A/B Counts (4 of each)	`[1, 17, 18, 18]`, `[9, 3, 2, 2]`
↓	↓
B Percentages (4)	`[90, 15, 10, 10]`
↓	↓
Personality Type	`"ISTJ"`

Class Constant

Because the Keirsey personality test assigns people on four dimensions, your program will likely include the number 4 in several places. To improve the readability of your code, you should include a class constant in your program to represent the number of dimensions (4). However, unlike previous constants, you should not expect to be able to change this value and have your program still function correctly. In this case, you will be using the constant purely for documentation and code quality purposes.

Permitted Java Features

For this assessment, you are restricted to Java concepts covered in chapters 1 through 7 of the textbook, and you MUST use arrays (see above). However, you **MAY NOT** use two-dimensional arrays. You also **MAY NOT** use the `toCharArray` method in the Java `String` class.

Code Quality Guidelines

In addition to producing the desired behavior, your code should be well-written and meet all expectations described in the grading guidelines and the Code Quality Guide. For this assessment, pay particular attention to the following elements:

Capturing Structure

Your `main` method in this program may have more code than it has in previous assessments. In particular, you may include a limited amount of output and some control flow constructs in `main`. However, your `main` method must remain a concise summary of your program's structure, and you must still utilize methods to both capture structure and eliminate redundancy. In particular, you should not process a line of input character-by-character in `main`. Your program should utilize parameters and return values effectively to produce a well-structured program as described above. Your methods should not accept unnecessary or redundant parameters, and should not use parameters or return values inappropriately (such as through "chaining").

Look for opportunities to eliminate redundancy in similar operations using parameters, array traversals, and other techniques.

Arrays

You should use arrays appropriately throughout the assessment, including in the required places described above. You must also reduce redundancy when processing arrays by using `for` loops to traverse the array, rather then writing separate code for each array element (referred to as "unrolling" the array). You should also properly use arrays as parameters and return values. Remember that arrays use *reference semantics*, meaning that when an array is passed as a parameter, its elements can be modified in the method and the changes will be seen by the caller.

Code Aesthetics

Your code should be properly indented, make good use of blank lines and other whitespace, and include no lines longer than 100 characters. Your class, methods, variables, and constant should all have meaningful and descriptive names and follow the standard Java naming conventions. (e.g. `ClassName`, `methodOrVariableName`, `CONSTANT_NAME`) See the Code Quality Guide for more information.

Commenting

Your code should include a header comment at the start of your program, following the same format described in previous assessments. Your code should also include a comment at the beginning of each method that describes that methods behavior. Method comments should also explicitly name and describe all parameters to that method and describe the method's return value (if it has one). Comments should be written in your own words (i.e. not copied and pasted from this spec) and should not include implementation details (such as describing loops or expressions included in the code). See the Code Quality Guide for examples and more information.

Running and Submitting

You can run your Personality Test program by clicking the "Activate the Terminal" message in Ed. (Note that this is a different process than previous assessments.) You may also need to click the arrow at the bottom of the window to reveal the terminal. This will compile and execute your code and show you any errors, or the output of your program if it runs correctly. If you believe your output is correct, you can submit your work by clicking the "Mark" button in the Ed lesson. You will see the results of some automated tests along with tentative grades. **These grades are not final until you have received feedback from your TA.**

You may submit your work as often as you like until the deadline; we will always grade your most recent submission. Note the due date and time carefully—**work submitted after the due time will not be accepted**.

Getting Help

If you find you are struggling with this assessment, make use of all the course resources that are available to you, such as:

- Reviewing relevant examples from lessons, section, and lab
- Reading the textbook
- Visiting support hours
- Posting a question on the message board

Collaboration Policy

Remember that, while you are encouraged to use all resources at your disposal, including your classmates, **all work you submit must be entirely your own**. In particular, you should **NEVER** look at a solution to this assessment from another source (a classmate, a former student, an online repository, etc.). Please review the full policy in the syllabus for more details and ask the course staff if you are unclear on whether or not a resource is OK to use.

Reflection

In addition to your code, you must submit answers to short reflection questions. These questions will help you think about what you learned, what you struggled with, and how you can improve next time. The questions are given in a `Personality Reflection` slide in the Ed lesson; type your responses directly into those textboxes.