

**Topics:** ch. 7, Arrays and producing an external output file, selection, iteration & data file Processing loops, File input/output, Token/Line-based input, try/catch ifs, random numbers, and using object analysis, structured, top-down design, methods, expressions, variables, loops, class constants, formatting, Graphics, interactive programming. The assignment involves computing statistics from personality test data.

**Specifications:**

1. Create a program **named Sp9YourInitials\_Personality.java** (e.g. Ed Bob Hyde would use Sp9EBH\_Personality.java ).
2. You must **meet all requirements in Java\_Program\_Style\_Requirements\_GradingComponents**.
3. You must submit the .java file, printed, easy to read listing with line numbers and screen-captures of 3 runs.
4. At the top of your console output, display author, title and a brief introduction; but, no offensive remarks.
5. The main method must clearly show the overall program structure -- all major activities, not just header/body/trailer.
6. Where a program repeatedly gets user input and displays results, the main method must show that loop.
7. The main method must not read input, nor print nor draw output, itself; it must delegate that work.
8. You must have one method to do text output and a separate method to do any required graphical output.
9. The code that asks the user for input must not be in the same method as any code to read data from a file.
10. You must put any data files in the same directory as your program.
11. For graphic output, DrawingPanel.java in the same directory as your program.
12. You must structure your solution using static methods that accept parameters and return values where needed.
13. You must have **at least 3 non-trivial methods other than main** in your program – see the next specification.
14. You should define other methods as needed for structure or to eliminate redundancy.
15. You must use the input file `personality.txt` from the course web site.
16. At the top of your output, display author, title and a brief introduction; but, no offensive remarks.
17. You must use arrays to store the various data for each of the four dimensions of the personality test.
18. You must use a class constant for the number of dimensions (4) in the personality test. It will not be possible to change this constant and have the program function properly, it is for readability purposes.

**Background Information:**

The Keirsey Temperament Sorter is a personality test that involves answering 70 questions. You may want to read more about the topic (<http://www.keirsey.com/>) but that is not necessary to complete this assignment. 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. **E**xtrovert versus **I**ntrovert (E vs I): what energizes you
2. **S**ensation versus **i**Ntuition (S vs N): what you focus on
3. **T**hinking versus **F**eeling (T vs F): how you interpret what you focus on
4. **J**udging versus **P**erceiving (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 **E**xtrovert, **i**Ntuitive, **T**hinking, **P**erceiving 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:

BABAAAABAAAAAABAAAABBAAAAABAAAABABAABAAAABABABAABAAAAAABAAAAAABAAAAA

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 & 3, 9 & 10, 16 & 17, 23 & 24, etc).
3. The next two questions are for Thinking/Feeling (questions 4 & 5, 11 & 12, 18 & 19, 25 & 26, etc).
4. The next two questions are for Judging/Perceiving (questions 6 & 7, 13 & 14, 20 & 21, 27 & 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  
BABAAAABAAAAAABAAAABBAAAAABAAAABABAABAAAABABABAABAAAAAABAAAAAABAAAAA

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 Keirsej 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

**Samples from input file: personality.txt**

```

Betty Boop
BABAAAABAAAAAABAAAABBAAAAABAAAABABAABAAAABABABAABAAAAAABAAAAAABAAAAA
Bugs Bunny
aabaabbabbbbaaaabaaaabaaaababbbaabaaaabaabbbbabaaaabaabaaaaabbbaaaabb
Han Solo
BA-ABABBB-bbbaababaaaabbaaabbbaabbabABBAAAABABBAAAABABAAAABBABAAABBABAAB

```

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.

**Sample log of execution (user input underlined):**

Personality Inventory Program by Phineas Taylor

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 as described in the book (6.1), or by using a `try/catch` statement as described in the book (6.4). Use a `PrintStream` to write to the output file as described in the book (6.4).

**Samples from 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

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"):

answers: "BA-ABABBB-bbbaababaaaabbaaabbabABBAABABBAABABAAAABBABAAABBABAAB"

	<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