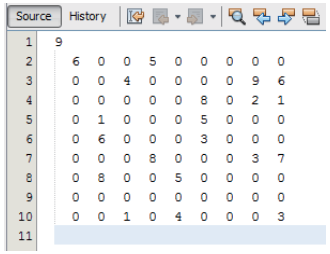
Lab 07 – File I/O

Objectives:

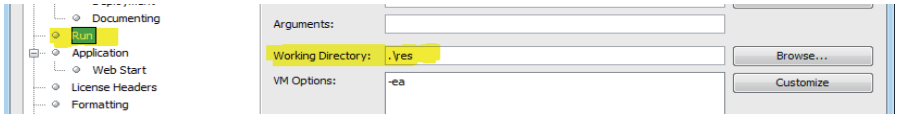
- Learn the concept of data streams and understand the difference between text data and binary data
- Learn to use various Java I/O classes for text and binary data

1. Write a class named *PuzzleRotator* that can read the Sudoku puzzle file format (9-killer.txt) and rotate the puzzle by 90-degree clockwisely. The puzzle is then saved to a new file using the original file name with the suffix "-rotated".



The file content of 9-killer-rotated.txt

You can put the files into the working directory as configured below.



[Discussion] Run your puzzle solver to solve the 9-killer and 9-killer-rotated puzzles. Which one is finished faster?

2. Run-Length Encoding (RLE) is a very simple form of data compression in which a run of data is stored as a pair of a symbol and a count, rather than as the original run. This is most useful on data that contains many such runs: for example, relatively simple graphic images such as icons, line drawings, and animations. For example, consider the following scan line where B represents a black pixel and W represents white:

WWWWWWBWWWWWWWWWWBBBWWWWWWBWWWWWWWWWW

When RLE data compression algorithm is applied to the above scan line, it becomes: **W6B1W9B3W6B1W9**

The new scan line is interpreted as six W's, one B, six W's, three B's, etc. Obviously, this representation is more compact and efficient.

You are given an RLE encoded file "**bean.dat**" of ***ASCII art***. The file is a binary file where every two bytes represent a codeword - the first byte is the symbol and the second byte represents the count by its binary value. Write a program to decode this file based on the RLE algorithm above. Save your decoded file as "**bean.txt**". Print the text to the output window and you should be able to see the ASCII drawing.

- END -