2022 Spring -- CSCI 1301L
Lab 12: Finding the Sums Program

Introduction

There is a puzzle game called "Word Search". In this game, the player is given a two-dimensional array of letters. The goal is to find the words that are spelled horizontally, vertically, and/or diagonally. We will implement a similar version of "Word Search" puzzle game. This program will use one-digit numbers instead of letters. The goal is to find the continuous blocks of cells where the sum of numbers equals to an input integer value. For example, if we wish to find the block areas where sum is 9, we may find that in the line of 1, 2, 3, 4, 5, there are (2, 3, 4) and (4, 5) that meet our set goal. To simplify the problem of this lab, we only deal with horizontal and vertical lines of the two-dimensional array to find the sums.

Take a look at the following example:

Horizontal Sums sumToFind = 20	Input Array	Vertical Sums sumToFind = 20
0 0 0 0 0 0 0 0 0 0	7 3 8 5 6 7 4 1 9 5	0 0 0 0 6 0 0 1 0 5
0 1 6 1 8 4 0 0 0 0	8 1 6 1 8 4 6 9 9 6	8 1 0 0 8 0 0 9 0 6
0 2 4 8 6 1 1 0 0 0	9 2 4 8 6 1 1 3 6 2	9 2 0 0 6 0 1 3 0 2
3 6 8 3 0 0 0 0 0 0	3 6 8 3 1 9 2 7 9 6	3 6 8 0 1 9 2 7 9 6
0 7 7 6 3 5 6 4 2 0	5 7 7 6 3 5 6 4 2 1	0 7 7 0 3 5 6 4 2 1
6 4 5 5 0 0 0 0 0 0	6 4 5 5 7 6 8 1 9 7	6 4 5 0 7 6 8 1 9 0
0 0 5 4 3 7 1 0 0 0	8 4 5 4 3 7 1 2 1 8	8 4 0 0 3 0 1 2 1 0
6 8 6 0 8 6 2 4 6 2	6 8 6 7 8 6 2 4 6 2	6 8 0 0 8 0 2 4 6 0
0 0 0 0 0 8 2 2 8 0	7 8 6 8 3 8 2 2 8 5	0 8 0 0 3 0 0 2 8 0
0 7 7 6 0 2 9 9 0 0	8 7 7 6 6 2 9 9 5 8	0 0 0 0 6 0 0 0 5 0
Note: 8 6 2 4 6 2 contains two overlapping sums that equal 20: 8 + 6 + 2 + 4 and 6 + 2 + 4 + 6 + 2		

The middle section is the input array (original two-dimensional data). The left section is the output that displays the values that are used (at least once) for getting the sum of 20. Those values that are not used to get the sum of 20 will be displayed as 0. Note that there may be overlapping sums that equal to 20, such as the highlighted numbers 8, 6, 2, 4, 6, 2. The right section is for the vertical sums, processed in the same way as for the horizontal sums.

Lab objectives

After completing this lab, you should be able to
(1) gain in-depth knowledge and experience in programming with two-dimensional arrays,
(2) use nested loops to iterate through the values of two-dimensional arrays in rows and columns, and
(3) further practice using methods, designing algorithms, and object-oriented programming.

Assignment

Summary: You will work on a two-dimensional (2D) input array of integers (**m** rows by **n** columns). The integers in the input array are from 1 to 9 (both inclusive). You will first write a method to convert such a 2D array to a neatly printable String. You will then write two methods that find the horizontal and vertical sums for that 2D input array and an input integer. For example, if the input integer is 20, then the method will find all adjacent values (horizontally and/or vertically) in the input array that equal to 20 and put these values into the output array. Other values not used for getting the sum of 20 will be set to zero (0) in the same position of the new array.

1. Create a class **FindTheSums**. Follow the instruction below to write methods within the class.

2. Think and design a working algorithm (e.g. in the form of pseudocode or drawing a logical workflow) to understand how the continuous blocks of numbers can be selected for getting the sum of an input value. Your scratch paper will not be checked for grading. However, it is encouraged to work on a logical solution first before moving to actual coding.

3. Write three public static methods: **arrayToString, HorizontalSums,** and **verticalSums** in the **FindTheSums** class.
   a. **public static String arrayToString (int[][] a)**
      This method returns a String that is a neat representation of all values in the array **a**. There is a single space between two columns of values. There is no space before the first column's value or after the last column's value. There is no newline before the first row or after the last row.
   b. **public static int[][] horizontalSums (int[][] a, int sumToFind)**
      This method creates an output array **b** that has the same dimensions as **a**. For each **a[i][j]**, if **a[i][j]** is part of the horizontal sum in a that equals to **sumToFind**, then assign **a[i][j]** to **b[i][j]**; otherwise, **b[i][j]** is 0. The method returns **b**.
   c. **public static int[][] verticalSums (int[][] a, int sumToFind)**
      This method creates an output array **b** that has the same dimensions as **a**. For each **a[i][j]**, if **a[i][j]** is part of the vertical sum in a that equals to **sumToFind**, then assign **a[i][j]** to **b[i][j]**; otherwise, **b[i][j]** is 0. The method returns **b**.

4. We provide a tester file for you to test your program for functional correctness: *FindTheSumsTester.java* (No need to modify this file!). You may, but are not required, use this file to check whether the execution output matches the output from manual calculation. Also see the example output at the end of this document. Note that valid and successful results from the tester program will NOT guarantee that your program has no problem at all. You should supply your own inputs and test your program thoroughly.

It is very important that you are NOT allowed to use any methods from java.util.Arrays or any Java Stream APIs throughout the entire program code.

5. Once you make the program working correctly. Understand the following statement for Academic Honesty and add it into the top of your source code to submit. The following lines should be added ABOVE the original first line of code.

```
/*
 * [CLASS/FILE NAME].java
 * Author: [YOUR NAME]
 * Statement of Academic Honesty:
 *
 * The following code represents my own work. I have neither
 * received nor given inappropriate assistance. I have not copied
 * or modified code from anywhere other than the authorized
 * sources. I recognize that any unauthorized sharing, assistance,
 * or plagiarism will be handled in accordance with both the
 * University of Georgia's Academic Honesty Policy and the
 * policies of this course. I recognize that my work is based on
 * an assignment created by the Department of Computer
 * Science at the University of Georgia. Any publishing or posting
 * of source code at any time for this project is prohibited.
 */
```

Replace [CLASS/FILE NAME] with the required name according to the assignment instruction. Replace [YOUR NAME] with your actual name.

Submission instruction

After you have completed and thoroughly tested your program, upload and submit the file *FindTheSums.java*. You do not submit the driver class file that contains the main method.

Grading

(1) Points are AWARDED when your program passes test cases that are determined or chosen by the grader/grading program. Test cases include using two 2D arrays with different sizes to check the arrayToString method (1 * 2 = 2 points), the horizontalSums method (2 * 2 = 4 points), the verticalSums methd (2 * 2 = 4 points).

(2) Points are DEDUCTED when your program fails to meet certain requirements:

- (-1 point) If the source file(s)/class(es) are named incorrectly (case matters!)
- (-1 point) If your source file(s) have a package declaration at the top
- (-1 point) If any source file you submit is missing your Statement of Academic Honesty at the top of the source file. All submitted source code files must contain your Statement of Academic Honesty at the top of each file.
- (-1 point) If you have more than one instance of Scanner in your program. Your program should only have one instance of Scanner.

• (-1.5 points) Inconsistent I/O (input/output) that does not match our instructions or examples exactly (unless otherwise stated in an assignment's instructions). Your program's I/O (order, wording, formatting, etc.) must match our examples and instructions.
• If your (-1 point) comments or (-1 point) variables are "lacking". "Lacking" means that your grader can find any lines of code or variables that take more than 10 seconds to understand, and there is no comment, or the variable name does not make sense (variable names like b, bb, bbb, etc. will almost never be acceptable).
• (-1 point) Indentation is not consistent throughout your source code (e.g. you should NOT use a combination of tabs and spaces in the files.)
• (-10 points) If you use a non-standard Java library or class (StringBuilder class, Arrays class, any class in java.util.stream) or other code specifically disallowed by the lab/project.

Special notice regarding the submission:

A. Late submission penalty. Points will be deducted from the original grade. If your submission is after the posted deadline…

(1) within 2 hours: -1
(2) after 2 hours: assignment will not be accepted.

Late submissions will not be accepted for this lab since it is the last lab of the semester, per the course syllabus.

B. If the source code file is missing, partially or completely broken, unable to open (including using a wrong file format), unable to compile, irrelevant to the assignment, purposely made to contain malicious codes, or determined to be an academic misrepresentation or a case of plagiarism, you will receive 0 grade for this assignment.

Below are sample inputs/outputs. This is not a comprehensive list of test cases.

```
Testing arrayToString method:
arrayToString(array1) test passed
arrayToString(array2) test passed

Testing horizontalSums method:
array1:
3 2 1 1
2 5 6 2
1 2 9 8
horizontalSums(array1, 7):
3 2 1 1
2 5 0 0
0 0 0 0
array2:
7 3 8 5 6 7 4 1 9 5
8 1 6 1 8 4 6 9 9 6
9 2 4 8 6 1 1 3 6 2
3 6 8 3 1 9 2 7 9 6
5 7 7 6 3 5 6 4 2 1
```

6 4 5 5 7 6 8 1 9 7
8 4 5 4 3 7 1 2 1 8
6 8 6 7 8 6 2 4 6 2
7 8 6 8 3 8 2 2 8 5
8 7 7 6 6 2 9 9 5 8
horizontalSums(array2, 20):
0 0 0 0 0 0 0 0 0 0
0 1 6 1 8 4 0 0 0 0
0 2 4 8 6 1 1 0 0 0
3 6 8 3 0 0 0 0 0 0
0 7 7 6 3 5 6 4 2 0
6 4 5 5 0 0 0 0 0 0
0 0 5 4 3 7 1 0 0 0
6 8 6 0 8 6 2 4 6 2
0 0 0 0 0 8 2 2 8 0
0 7 7 6 0 2 9 9 0 0
horizontalSums(array2, 25):
0 0 0 0 0 0 0 0 0 0
0 0 6 1 8 4 6 0 0 0
0 2 4 8 6 1 1 3 6 0
0 0 0 0 0 0 0 0 0 0
5 7 7 6 0 0 0 0 0 0
0 0 0 0 0 0 8 1 9 7
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 8 6 8 3 8 2 2 8 5
0 0 0 0 0 2 9 9 5 0

Testing verticalSums method:
array1:
3 2 1 1
2 5 6 2
1 2 9 8
verticalSums(array1, 22):
0 0 0 0
0 0 0 0
0 0 0 0
array2:
7 3 8 5 6 7 4 1 9 5
8 1 6 1 8 4 6 9 9 6
9 2 4 8 6 1 1 3 6 2
3 6 8 3 1 9 2 7 9 6
5 7 7 6 3 5 6 4 2 1
6 4 5 5 7 6 8 1 9 7
8 4 5 4 3 7 1 2 1 8
6 8 6 7 8 6 2 4 6 2
7 8 6 8 3 8 2 2 8 5
8 7 7 6 6 2 9 9 5 8
verticalSums(array2, 14):
0 0 8 5 6 0 0 0 0 0
0 0 6 1 8 4 0 0 0 6
0 0 0 8 6 1 0 3 0 2
3 0 0 3 1 9 0 7 0 6
5 0 0 6 3 5 6 4 0 1
6 0 0 5 7 0 8 1 0 7
8 0 0 0 3 0 1 2 0 0

6 0 0 0 8 6 2 0 6 0
0 0 0 8 3 8 2 0 8 0
0 0 0 6 0 0 9 0 0 0
verticalSums(array2, 33):
0 0 8 0 0 0 0 1 9 0
0 0 6 0 0 0 0 9 9 0
0 0 4 8 0 0 0 3 6 0
0 0 8 3 0 9 0 7 9 0
0 0 7 6 0 5 0 4 2 0
0 0 0 5 0 6 0 1 9 0
0 0 0 4 0 7 0 2 1 0
0 0 0 7 0 6 0 4 6 0
0 0 0 8 0 0 0 2 0 0
0 0 0 0 0 0 0 0 0 0