CSCI 1301 – Programming Principles I
Georgia Southern University
Department of Computer Science
Fall 2024

Lab 11
Point Value: 20 points
Due: Friday November 15, 2024, end of lab

Objectives

1. Work with defining, creating, and using arrays.
2. Incorporate arrays into working programs that include methods, looping, conditional statements, and variables.
3. Use the debugger to effectively diagnose and fix errors in code.
4. Archive Java source code files and other documents into a zip file.

Description

Complete the following steps as described and upload requested materials to Folio in the appropriate dropbox in a single zip file. Submit Java source code files to Gradescope.

- Students should work in groups of two for this lab. When submitting your zip file, fill out the comment header and indicate both group member names.

- Your submitted zip file should be named Lab#LastName1LastName2.zip. For instance, Lab11JonesSmith. Any requested screenshots and/or answers to questions should be included in a separate Word document.

- All programs should have a comment header. That comment header should include information similar to:
```
/**
 * File: PAssign0.java
 * Class: CSCI 1301
 * Author: Christopher Williams
 * Created on: Jun 6, 2016
 * Last Modified: Aug 16, 2018
 * Description:  Display three messages to the console
 */
```

- Before writing any code, for each problem, discuss with your partner how you plan to approach solving the given problem. Once you both agree on a way to attempt the problem, only then should code be written.

- Pay attention to names and any other requested material. Failure to include material will result in the loss of points as described below.

Lab Problems
For all problems, use inline comments to describe major actions. Ask for assistance if needed.

1. Processing 2D Arrays – The pirate Blackbeard has a problem – his ship is sinking! He quickly needs to decide which cargo to toss overboard and which to keep. He will accomplish this by comparing weights and values. If the cargo weighs more than 100 lbs. and has a value less than $1500, then it must be thrown overboard.

   Blackbeard hands you a 2D array of values where the first array element corresponds to weights, and the second array element corresponds to values. Create a class named Lab11Prob01.java and write code to determine and display which bags to keep and which to toss overboard.

```
int[][] pirateBooty = {
   { 110, 1600 }, { 101, 1400 }, { 200, 50 }, { 322, 700 }, { 57, 500 },
   { 625, 1500 }, { 300, 320 }, { 50, 210 }, { 100, 105 }, { 90, 400 },
   { 30, 2000 }, { 200, 1300 },
};
```

   HINTS
   - Consider whether an inner loop is necessary to solve this problem before attempting to code a solution.
   - Review how printf() works with padding to line up the right-aligned Cargo numbers.
   - Downloading the PDF might make copying the above array easier.

   Expected Output:
```
Cargo  1: keep
Cargo  2: toss
Cargo  3: toss
Cargo  4: toss
Cargo  5: keep
Cargo  6: keep
Cargo  7: toss
Cargo  8: keep
Cargo  9: keep
Cargo 10: keep
Cargo 11: keep
Cargo 12: toss
```

   After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

   After instructor check-off, BOTH students should submit a final, identical version to Gradescope. This version is used for your final grade.

2. Processing 2D Arrays – Create a class named Lab11Prob02 that contains a method that takes a 2D array of integers as a parameter. This method should be named printRowMax() that prints the maximum value found in each row of the 2D array. This method should not return anything.

   For instance, for the array { {2, 9, 4}, {-512, -1024, -2048}, {5, -8, 17} }, it should print 9, -512, 17 each on its own line with the descriptive text shown in the output. Make sure to choose intelligent initial values so that your code works all possible array values. This method should work for any 2D array of integers.

   HINT: It would be advisable to create a separate method to print whatever 2D array you use to test your method and manually verify that your output is correct.

   Expected Output (for example given in problem):
   ```
   Row 1 Max: 9
   Row 2 Max: -512
   Row 3 Max: 17
   ```

   After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

   After instructor check-off, BOTH students should submit a final, identical version to Gradescope. This version is used for your final grade.

3. Processing 2D Arrays – Create a class named Lab11Prob03 that contains a method that takes a 2D array of integers as a parameter. This method should be named transpose2DArray() that transposes (swaps rows into columns) the original 2D array and returns that transposed 2D array. You may assume all “interior” arrays have the same length.

   For instance, for the array { { 1, 2, 3 }, { 5, 4, 6 }, { 9, 8, 7 }, { 12, 10, 11 } }, the transposed copy would be { { 1, 5, 9, 12 }, { 2, 4, 8, 10 }, { 3, 6, 7, 11 } }:

   ```
    1  2  3            1 5 9 12
    5  4  6      =>    2 4 8 10
    9  8  7            3 6 7 11
   12 10 11
   ```

   It would be advisable to print whatever array that is passed to your method and the array that is returned to your method to manually verify that your transpose is correct.

   HINTS (Read all below, each are VERY important):

- Think very carefully about the dimensions of the new copied transposed array before implementing this method.
- Mapping out the indices on paper of the original location vs. the transposed location will help you arrive at a solution.
- This solution can use a row-major or column-major traversal of either array but should account for two things:
    - Depending on which array you use to control your loops, your loop continuation condition may or may not be the same as the examples used in the slides (there will still be some hard-coded values). During your remapping of indices, think hard about what elements need to be visited in what order in which array to make this problem easier.
    - Adjust the new array positions based on the indices being used in the problem.

After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

After instructor or TA check-off, BOTH students should submit a final, identical version to Gradescope. This version is used for your final grade.

4. Demonstrate all working programs to the instructor or TA to receive credit for each problem. Turn in each problem inside the zip file detailed below.

5. Name the zip file Lab11LastName1LastName2.zip (e.g. Lab11JonesSmith) and submit that zip file to Folio. Both students must submit the exact same zip file for either to receive a grade.

Grade Breakdown

Demonstrate running application to instructor or TA:

Problem 1	6 points
Problem 2	6 points
Problem 3	6 points
Zipped source files (in Folio):	2 points
Total Possible:	20 points

Last modified: November 10, 2024