Lesson 4.06 — Nested Loop Algorithms & Rectangular Arrays

Overview

Objectives — *Students will be able to...*

- **Correctly adjust** nested loop headers for use with arrays.
- **Correctly construct** two-dimensional arrays.

Assessments — *Students will...*

- **Complete** WS 4.6

Homework — *Students will...*

- **Read** BJP 10.1 up to "Adding to and Removing from an ArrayList"
- **Complete** self-check problems #1 - 6

Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Classroom copies** of WS 4.6
- **Array whiteboards**

Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction/Review of objects & string processing	10min
Round Robin	35min
Paper selection & grade announcement	3min

If the instruction takes longer than expected, assign the worksheet to small groups so students can work together collaboratively. This should allow you to complete the lesson in one class period. Alternatively, you can assign students to finish the worksheet at home for homework, or make the worksheet shorter.

Procedure

There are several ways you can teach today's class. You should first check in with your students to see how prepared they are for today's lesson. If students understood most of what they read for homework last night, you can ask students for specific questions, cover only those topics, then move on to the Round-Robin activity. If your class is mostly confused, you can re-teach all of the content, working through several examples with Think Pair Shares before breaking into the Round Robin Activity. An optional Programming Challenge is included at the end of the lesson if your lesson finishes early.

Bell-work and Attendance [5 minutes]

Introduction/Review of Objects & String Processing [10 minutes]

Emphasize with students...

Content - Advanced programming structures - Uses of pre-built data structures Multidimensional arrays have quite a few uses. They can serve as grids for seating plans or for board games such as chess and checkers. Some students enjoy recreating the game of battleship using two dimensional arrays.

As you learn about their use, remember to consider how you might use multidimensional arrays in your future projects.

---

1. Two dimensional arrays come in handy when tracking certain types of data; adjust the explanation so it's relevant to your students (a grocery store, a Starbucks, a farmer's market).
   - Previously, we might have used an array to keep track of sandwich orders in line at a bodega:
   - Suppose we have several workers at the bodega making sandwiches, and we want to keep track of the orders each one is working on. We could use multiple arrays, or an array of arrays.
   - To traverse multidimensional arrays, we need a new tool to help us fill them.
2. Demonstrate a basic nested loop used to switch numbers in an array. Begin by using the array whiteboard to demonstrate what you want the code to do:
   - Challenge the students to come up with pseudocode (or actual code, if they're ready for it) that prints out all of the inversions in this array. An inversion is a pair of numbers in which the first number in the list is greater than the second number. Be sure to emphasize that no elements are being moved or shifted. The output for the array above would be:

     ```
     (4, 3)
     (4, 2)
     (4, 1)
     (3, 2)
     (3, 1)
     (2, 1)
     ```

   - An array

     would output:

     ```
     (3, 1)
     (3, 2)
     (4, 2)
     ```

3. As your students are working on this problem, encourage them to use the whiteboard arrays to organize their thoughts and visualize the problem. Their pseudocode should look something like this:

   ```
   for (every possible first value) {
       for (every possible second value) {
           if (first value > second value) {
               print (first, second).
           }
       }
   }
   ```

   - As a whole group, or in individual groups (depending on familiarity with the material), construct the final code:

     ```
     for (int i = 0; i < data.length - 1; i++) {
         for (int j = i + 1; j < data.length; j++) {
             if (data[i] > data[j]) {
                 System.out.println ("(" + data[i] + ", " + data[j] + ")");
             }
         }
     }
     ```

     As a "Tricky Code Cheat Sheet" tip, you might discuss the general reminder that for an inversion, the second value has to appear after the first value in the list. This means that you only want to compare values that come after the first value, so the inner loop initializes at `j = i + 1`.

If students ask why the outer loop ends at `data.length - 1`, have them trace the code and predict output using their whiteboards. They should notice that since only pairs of numbers are being compared, the last element of the array will never be a possible first value.

4. Ask a volunteer student to help you build a physical representation of a multidimensional array (an array of arrays). Demonstrate the code used to create arrays of multiple dimensions, and have students gather array whiteboards from around the room to show what the arrays would look like in memory:

   - `double`: one double value
   - `double[]`: a 1 dimensional array of doubles
   - `double[][]`: a 2 dimensional array (grid) of doubles
   - `double[][][]`: a 3 dimensional array (cube) of doubles
   - An array constructed with the code below has 2 rows and 3 columns:

     ```
     double[][] ages = new double[2][3];
     ```

     Have students index the array for you:
   - You might opt to give students this general formula for the syntax of declaring and constructing a multidimensional array:

5. Run through some examples with your students:

   - To access the 1st element of the 2nd row: `ages[1][0]`
   - To access all elements of the 1st row: `ages[0]`
   - To access all elements of the 2nd row: `ages[1]`

6. Multidimensional arrays can also be passed as parameters. Have students trace the flow of control and predict the output of the code below:

```
public static void print (double[][] grid) {
    for (int i = 0; i < grid.length; i++) {
        for (int j = 0; j < grid[i].length; j++) {
            System.out.print(grid[i][j] + " ");
        }
        System.out.println();
    }
}
```

   - Ask students why you referenced `grid.length` in the outer loop (the number of rows), and what `grid[i].length` refers to (the number of columns).
   - As a final tip, let students know that for multidimensional arrays, `Arrays.toString` won't work correctly, and they should use `Arrays.deepToString` instead.

Round Robin [45 minutes]

1. Round-robin is a drilling and error-checking exercise used with worksheets. At minimum, there should be 1 question for each student (*e.g.* a class of 15 students would need a worksheet with 15 or more questions). Students write their name on the worksheet, complete the first problem, then pass the paper to the student on the right (or whatever direction you choose). The next student first checks the previous answer, correcting it if need be, then completes the second question. Each student then passes on the paper again. By the end of the exercise, each student has checked and completed each question on the worksheet.

2. The hook is that you choose only ONE worksheet from the pile to grade. All students get a grade from that one worksheet. This keeps students invested throughout the exercise. Advanced students will check questions throughout the whole worksheet, and all students will try their best to catch their own (and others') mistakes, since the whole class shares the randomly-selected-paper's grade.

3. You should time each question/checking interval, and call "TIME!" when it is time for students to pass along papers.

   a. Questions 1–4 should take 1 minute each.
   b. Questions 5–9 should take 2 minutes each.
   c. Questions 10–12 should take 3 minutes each.
   d. Questions 13–14 should take 4 minutes each.
   e. Questions 15–16 [Bonus] should take 5 minutes each.

   Adjust the timing on these questions as needed, but try to keep a brisk pace. Part of the engagement factor is the sense of urgency.

Paper Selection & Grade Announcement [3 minutes]

If time allows, randomly select the worksheet and announce the class grade with a bit of fanfare, congratulating the class on a job well done. If there are any incorrect answers, use the time at the end of class to review the correct solutions or take questions.

Accommodation and Differentiation

To optimize this exercise, you might consider rearranging students (or creating a passing-path) that mixes students of different coding abilities. The advanced students can use the extra time to correct mistakes made by others; if they are sitting in proximity to the student that made the error, they will have a better chance of explaining the correct answer to them.

Due to the brisk pace of the round-robin rotation, there shouldn't be too much down time for any one student. If your students are finishing faster than the time intervals indicated, reduce the amount of time allotted to maintain a sense of urgency/keep students on task. If finished early, offer a Programming Challenge while you grade the sample worksheet.

Common Mistakes

Two dimensional arrays common mistakes: http://interactivepython.org/runestone/static/JavaReview/Array2dBasics/a2dMistakes.html

Videos

- BJP 7-3-2, *Tallying with an Array* http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoPlayer.php?id=c7-3-2

Forum discussion

Lesson 4.06 Nested Loop Algorithms & Rectangular Arrays (TEALS Discourse account required)