Chapter 9

Multidimensional arrays

Lesson page 9-1. Multidimensional arrays

Activity 9-1-1 Declaring and creating two-dimensional arrays

Question 1. Array `fred` has 46 rows.

Question 2. `Object[][] b= new Object[2][3];`

Activity 9-1-2 Referencing an array element

Question 3. Variables `r` and `c` must be of type `int`. Variable `r` must be in the range `0..b.length-1`; `c`, in the range `0..b[r].length-1`.

Activity 9-1-3 Referencing the number of rows and columns

Question 4. The underlined expressions are: `b.length`; `b[0].length`

Question 5. Here's the completed program segment:

```
// Add one to each element of rectangular
// int array b
    for (int i= 0; i!=b.length; i= i+1) {
        for (int j= 0; j!=b[i].length; j= j+1) {
            b[i][j]= b[i][j]+1;
        }
    }
```

Activity 9-1-4 Exercises on two-dimensional arrays

Activity 9-1-5 A non-Java notation for a subarray

Question 6. The initialized section is: `b[0..1][0..2]`.

Question 7. The assertions are:

1. `b[0][1..2]`
2. `b[0..1][0]`
3. `b[1][0..2]`

4. `0 < i <= 1 && 0 < j <= 2 && b[i-1][j-1] ≤ b[i][j]`

Activity 9-1-6 Two-dimensional array initializers

Question 8. The dimensions are:

1. 2 rows, 1 column
2. 3 rows, 3 columns
3. 2 rows, 4 columns
4. This is a 3-dimensional array. The first dimension has size 3, the second demension has size 2, and the third dimension has size 1.

Question 9. `int[][] oz= {{1,3,5}, {3,5,7}, {5,7,11}, {7,11,13}};`

Lesson page 9-2. Some programs that use two-dimensional arrays

Activity 9-2-1 Printing a table of values

Question 1. First, in the conditions of the second **for** loop, **int** `c` is not an array, so `c[r].length` is an invalid statement. Try `b.[r].length` instead.

Second, in the body of the second **for** loop, there is no variable `d`! This should be `b[r][c]`.

If you fix those errors, you will still get an out-of-bounds exception. The condition of the second **for** loop should be `c != b[r].length`.

Finally, if your entire method looked like the following:

```
public static void main(String[] arguments) {

    Integer[][] b= new Integer[3][2];

    // Integer[][] b= new Integer[3][2];
    // Print arry b, one row per line
    // Precede each row with the integer 1+(row number)
    // inv: rows 0, ..., r-1 have been printed
    for (int r= 0; r!=b.length; r++) {
        // Print row b[r] (on the next line)
            System.out.print((1+r) + " ");
            // inv: elements b[r,0..c-1] have been printed
            for (int c= 0; c!=b[r].length; c++) {
                System.out.print(" " + b[r][c]);
            }
    }
}
```

Then the output would be the following, because array `b` has not been initalized:

```
1  null null2  null null3  null null
```

Try inserting code to initalize the elements of `b` to 0 and run again. Remember that the elements of `b` are of class `Integer`.

Activity 9-2-2 A schema for processing a two-dimensional array

Question 2. The ordering of the elements of a rectanuglar array in which the elements of row 0 come first, then the elements of row 1, etc.

Question 3. The ordering of the elements of a rectangular array in which the elements of column 0 come first, then the elements of column 1, etc.

Question 4. Here is the schema:

```
// Process the elements of d[0..][0..], in row-major order
    // inv: d[0..r-1] and d[r][0..c-1] have been processed
    for (int r= 0; r!= d.length; r++) {
        for (int c= 0; c!= d[r].length; c++) {
            Process d[r][c]
        }
    }
```

Question 5. Remember: since `d` is a square array, the number of rows in `d` equals the number of columns.

```
// Process the elements of square array d[0..][0..]
// in column-major order.
    for (int c= 0; c!=d.length; c++) {
        for (int r= 0; r!=d.length; r++) {
            Process d[r][c]
        }
    }
```

Activity 9-2-3 An interest(ing) table

Question 6. Here's the program:

```
// For each element d[r][c], print 1+2+...+d[r][c]
    // inv: the sums for d[0..r-1] and d[r][0..c-1]
            have been printed
    for (int r= 0; r!= d.length; r++) {
        for (int c= 0; c!= d[r].length; c++) {
            int n= d[r][c];
            System.out.print((n*(n+1)/2) + " ");
        }
        System.out.println();
    }
```

Activity 9-2-4 Class Coordinates

Question 7. Fields `r` and `c` are **public**.

Activity 9-2-5 Row-major search

Question 8. The position is `(0,1)` for row-major and `(1,0)` for column-major order.

Question 9. Here is row-major search:

```
// = index (r,c) of the first (in a row-major order) element
// x in d (or the pair (d.length,0), if x is not in d)
public static Coordinates search(int[][] d, int x) {
    for (int i= 0; i!=d.length; i++) {
        for (int j= 0; j!=d[i].length; j++) {
            if (d[i][j] == x)
                { return new Coordinates(i, j); }
        }
    }
    return new Coordinates(d.length, 0);
}
```

Activity 9-2-6 Saddleback search

Question 10. The first two are arranged as required for saddleback search; the third is not.

Question 11. Here are the variables, showing how they changed:

d

1	2	3
2	3	4
3	4	5

r	c
0	2
1	

x = 4

Lesson page 9-3. The Java concept of a multidimensional array

Activity 9-3-1 The Java concept of a multidimensional array

Question 1. Here is a picture of array `d`:

d

1	2	3	
4	5		
6	7	8	9

Question 2.

`d.length`: 3
`d[0]`: the name of the object that contains array {1,2,3}

`d[1]`: the name of the object that contains array {`4,5`}
`d[2]`: the name of the object that contains array {`6,7,8,9`}

Hint: `d[0]` does not contain {`1,2,3`}. Try writing a test program that creates `d` and prints the values.

Activity 9-3-2 The lengths of rows and columns

Question 3. All six expressions evaluate to: `2`.

Activity 9-3-3 Ragged arrays

Question 4. A ragged array is a multidimensional array in which the rows have different lengths.

Question 5. Here is the method:

```
// Print x neatly: each row an a separate line, with
// adjacent elements separated by a comma.
public static void d2intPrint(int[][] x) {
    for (int i= 0; i!=x.length; i++) {
        for (int j= 0; j!=x[i].length; j++) {
            if (j != 0) { System.out.print(", "); }
            System.out.print(x[i][j]);
        }
        System.out.println();
    }
}
```

Activity 9-3-4 Exercises on Java ragged arrays

Lesson page 9-4. Programs that use ragged arrays

Activity 9-4-1 Pascal's triangle

Question 1. Combinatorics is the study of permutations and combinations of finite sets of objects. For example, a combinatorics type of problem would be: You have 3 shirts and 2 pairs of pants. How many different combinations of shirt and pants can you wear?

Question 2. "`n choose r`" denotes the number of subsets of size `r` that can be chosen from a set of size `n`.

Question 3. The expression is: `20 choose 10`.

Question 4. The first inner element, at position `(2,1)`, is the sum of the two above it. Every other inner element is the product of the two above it. There may be other correct answers.

```
Penultimate row:
 1   2    16   64   16    2    1

Last row:
 1   2    32 1024 1024 32    2    1
```

Activity 9-4-2 Implementing Pascal's triangle

Question 5. Here is the array and assignments to initialize it:

```
String[][] wallaceAve= new String[3][];
wallaceAve[0]= {"Harvey", "Sam"};
wallaceAve[1]= {"Henry"};
wallaceAve[2]= new String[15];
wallaceAve[2][6]= "George";
wallaceAve[2][9]= "Greg";
wallaceAve[2][14]= "Geoffrey";
```

Activity 9-4-3 Printing Pascal's triangle

Question 6. Here's method `d2objPrint`.

```
// Print x neatly: each row on a separate line, with
// adjacent elements separated by a comma.
public static void d2objPrint(Object[][] x) {
    for (int i= 0; i!=x.length; i++) {
        for (int j= 0; j!=x[i].length; j++) {
            if (j != 0) { System.out.print(", "); }
            System.out.print(x[i][j]);
        }
        System.out.println();
    }
}
```