Lecture 28: Two-Dimensional Arrays

Last time:
1. Code comments
2. Javadoc comments

Today:
1. Project #6 due
2. 2-dimensional arrays
Project #6 Assigned!

- Project due Monday, 4/16 at 11 pm
- Project is closed
  - You must complete the project by yourself
  - Assistance can only be provided by teaching assistants (TAs) and instructors
  - You must not look at other students' code
- Start now!
  - Read entire assignment from beginning to end before starting to code
  - Check out assignment now from CVS
  - Follow the instructions exactly, as much of grading is automated
Recall Arrays

- Arrays: sequences of elements from the same base type
  
  ```java
  int[] a;  // array of ints
  Date[] d;  // array of Dates
  ```

- Base type may be:
  - Primitive (i.e. `int`)  
  - Reference (i.e. `Date`, other objects)

- Arrays are also objects
  - Arrays created using `new`
  - Array elements stored on heap
  - Array variables store references to heap
Can You Create Arrays of Arrays?

- Yes!
  - Arrays are objects
  - Arrays of objects may be created
  - So arrays of arrays can also be created

- Syntax

  ```java
  char[][] a; // Array of char arrays
  a = new char[3][]; // Create array of 3 arrays
  ```
Example

```java
char[][] a;
a = new char[3][];
a[0] = new char[4];
a[1] = new char[6];
a[2] = new char[3];
a[1][3] = 'a';
```

- This array has **two dimensions**: rows, columns
- This kind of array is called **ragged** because the rows are of unequal length
Notation

char[][] a;
a = new char[3][];
a[0] = new char[4];
a[1] = new char[6];
a[2] = new char[3];

- a is an array of arrays of char
- a[0], a[1], ... refer to arrays of char
- a[0][0], a[0][1], ... refer to char variables

In a[i][j]:
- i is the row
- j is the column
Questions

```java
char[][] a;
a = new char[3][];
a[0] = new char[4];
a[1] = new char[6];
a[2] = new char[3];
```

- What does `a[1][2] = 'a';` do? Set element in row 2, column 3 to `a`
- What does `a.length` return? 3
- What does `a[1].length` return? 6
Example:
RandomRaggedArray.java

import java.util.Random;
public class RandomRaggedArray {
    public static void main(String[] args) {

        Random generator = new Random();
        int rows = generator.nextInt(10) + 1;
        int[][] a = new int[rows][];

        for (int i = 0; i < rows; i++) {
            a[i] = new int[generator.nextInt(10) + 1];
            for (int j = 0; j < a[i].length; j++)
                a[i][j] = generator.nextInt(10);
        }

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < a[i].length; j++)
                System.out.print(a[i][j] + "  ");
            System.out.println();
        }
    }
}
Example:
InitialValueRagged.java

public class InitialValueRagged {

    public static void main(String[] args) {
        int a[][] = {{2, 3, 5}, {7}, {8, 9, 10, 23, 55}};

        // Output formatting:
        //  - Output elements of each row on same line
        //  - Consecutive rows should be on consecutive lines

        for (int i = 0; i < a.length; i++) {
            // Print row i; note "print" rather than "println"
            for (int j = 0; j < a[i].length; j++)
                System.out.print(a[i][j] + " ");

            // Now print carriage return at end of row
            System.out.println();
        }
    }
}
Rectangular Arrays

- Often we want 2-dimensional arrays in which rows have the same length
  - Tables
  - Matrices (what are these?)
- Java has a special short-hand syntax for creating rectangular arrays
  ```java
  int[][] a = new int[2][4];  // 2 rows, 4 cols
  ```
  Equivalent to:
  ```java
  int[][] a = new int[2][];
  a[0] = new int[4];
  a[1] = new int[4];
  ```
- The short-hand takes care of allocating each row, initializing each cell in each row
Example

```java
int[][] a = new int[2][4];
```

- Note each cell is initialized to default value (0)
- Each row is a 1-dim array
Example:
TwoDimensionalArray.java

```java
import java.util.Random;

public class TwoDimensionalArray {

    public static void main(String[] args) {
        Random random = new Random();
        final int ROWS = 10;
        final int COLS = 40;

        char[][] a = new char[ROWS][COLS];

        for (int i = 0; i < ROWS; i++)
            for (int j = 0; j < COLS; j++)
                a[i][j] = (char)(random.nextInt(26) + 65);

        for (int i = 0; i < ROWS; i++) {
            for (int j = 0; j < COLS; j++)
                System.out.print(a[i][j]);
            System.out.println();
        }
    }
}
```
2-D Arrays of Objects Also Possible

String[][] s = new String[4][2];
s[0][0] = "Fred";
s[1][1] = "Jane";