Two-Dimensional Arrays

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane
Overview

- Multidimensional arrays
- Arrays of arrays
- Multidimensional initializers
- Ragged arrays
- Multidimensional Arrays of Objects
News

- Site with Java Examples
  http://www.java2s.com/

- Firesheep
  http://codebutler.com/firesheep
About Arrays

- Zero-size arrays
  - What space do they occupied?
  - Advantages/disadvantages when compare with null
- Visualization of arrays through the debugger
  - Let’s see an array through the debugger
We have discussed the notions of:

**Array of primitive types**: Consider the declaration:

```java
char[] c = new char[5];
```

c: is of type `char[]`, that is, an array of characters. 
c[2] and c[i]: are of type `char` (a single character)

**Array of class objects**:

```java
String[] s = new String[10];
```

s: is of type `String[]`, that is, an array of strings. 
s[3] and s[j]: are of type `String` (a single String).

Can we have an array of arrays? Yes! In Java this is called a **multidimensional array**.
Java’s stores a 2-dim array as an **array of array references**.

```java
char[][] a = new char[5][8];
```

Java allocates space for the **array of array references**, and then allocates space for the **individual arrays**

- **a**: is of type `char[][]`, an array of array of characters (whole page)
- **a[4]**: is of type `char[]`, an array of characters (a single line)
- **a[4][23]**: is of type `char`, a single character (character 23 of line 4)
Let’s be more concrete. Consider the following declaration:

```
char[ ] [ ] a = new char[5][8];
```

Conceptually, this is laid out as follows:

By convention the 1\textsuperscript{st} index is the row, the 2\textsuperscript{nd} is the column.
Consider the declaration:

```java
char[ ][ ] a = new char[5][8];
```

What is the meaning of `a.length`?
- 5? 8? 40?
- Undefined?

**Ans:** 5. This is clear from the illustration on the previous page. Array `a` is an array of 5 references to other arrays.

What is the meaning of `a[2].length`?

**Ans:** 8, because `a[2]` is an array of 8 characters.
Ragged Arrays

- When you allocate an array of arrays, do all the arrays have to be of the same size?
- No. When the arrays have different sizes, it is called a ragged array. You can specify their sizes individually.

```java
char[][] a = new char[5][];
a[0] = new char[8];
a[1] = new char[3];
a[2] = new char[5];
a[3] = new char[0];
a[4] = null;
```
Two-Dimensional Arrays and Loops

- Nested loops go hand in hand with two-dimensional arrays.
- The following is the standard nested loop to go row by row in a two-dimensional array:

```java
for (int row = 0; row < a.length; row++)
    for (int col = 0; col < a[row].length; col++)
        a[row][col] = ' ';  
```

When all rows have the same length, we could use `a[0].length`.  

[Diagram of a two-dimensional array with nested loops example]
1-dim Initializer: recall

```java
int[ ] quizScoresOne = { 90, 82, 75, 66 };
```

2-dimensional Initializer:

```java
int[ ][ ] quizScoresTwo = {
    { 90, 82, 75, 66 },
    { 85 },
    { 45, 77, 99 }
};
```

This allocates and initializes a **ragged array** with 3 rows.

Example (Initializers.java): Print the array.

```java
for ( int row = 0; row < quizScores.length; row++ ) {
    System.out.print( "Scores for student " + row+ ":" );
    for ( int col = 0; col < quizScores[row].length; col++ )
        System.out.print( " " + quizScores[row][col] );
    System.out.println();
}
```

Output:

```
Scores for student 0: 90 82 75 66
Scores for student 1: 85
Scores for student 2: 45 77 99
```
Multidimensional Arrays of Objects

- We have discussed the notion of two-dimensional arrays of primitives:

  ```java
  char[ ] [ ] page = new char[50][100];
  ```

- Can we have multidimensional arrays of objects? Yes!
Multidimensional arrays of objects behave exactly as multidimensional arrays of primitives except for one main difference: When you define the array:

  ```java
  ObjectType[][] var = new ObjectType[MAXROW][MAXCOL]
  ```

  you are actually creating a two-dimensional array of references to objects; those objects don’t exist yet!

- **Example:** TwoDimArrayObjects.java
- **Example:** PassingArrays.java
  - Notice that we can pass arrays as one-dimensional arrays
  - Notice we can pass a row of a two-dimensional array
- Let’s see a two-dimensional array through the debugger
In our examples, we have processed the two-dimensional array row by row but we could also process it column by column.

Notice that we can have sharing of objects by several array entries and sharing of rows.

Two-dimensional arrays examples online

http://www.java2s.com/Tutorial/Java/0140_Collections/0060_Multidimensional-Arrays.htm