CMSC 131
Object-Oriented Programming I

Two-Dim Arrays I

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

Multidimensional Arrays

- We have discussed the notions of:

  Array of primitive types: Consider the declaration:

  char[ ] c = new char[3];

  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:

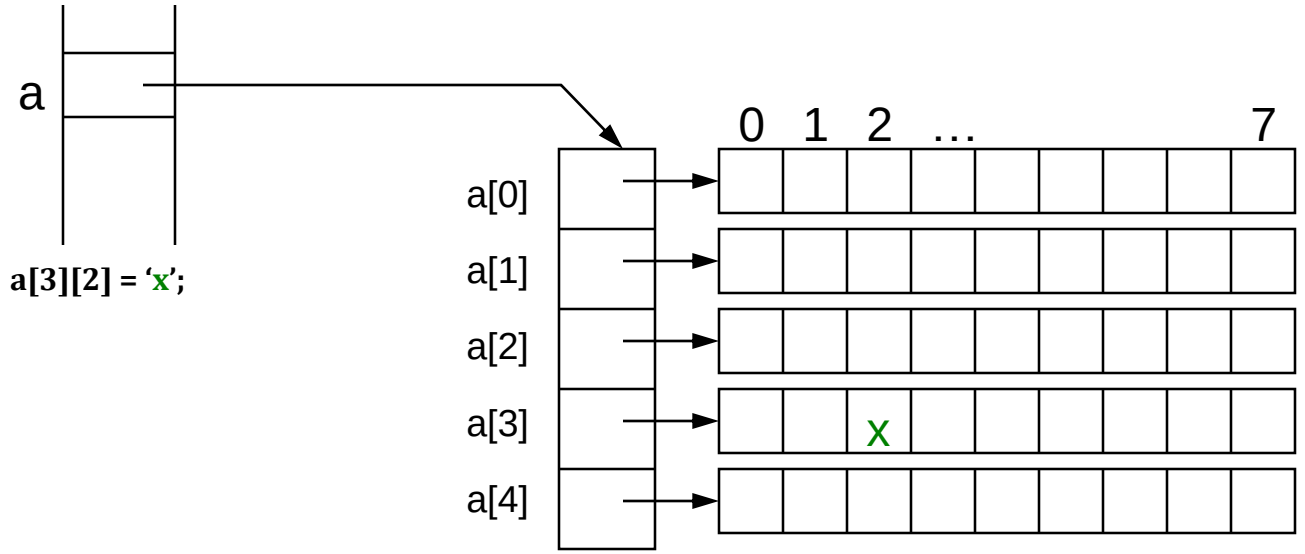
  String[ ] s = new String[4];

  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**

Two-Dimensional Arrays

- Java's stores a 2-dim array as an **array of array references**.
  char[ ] [ ] a = new char[5][8];



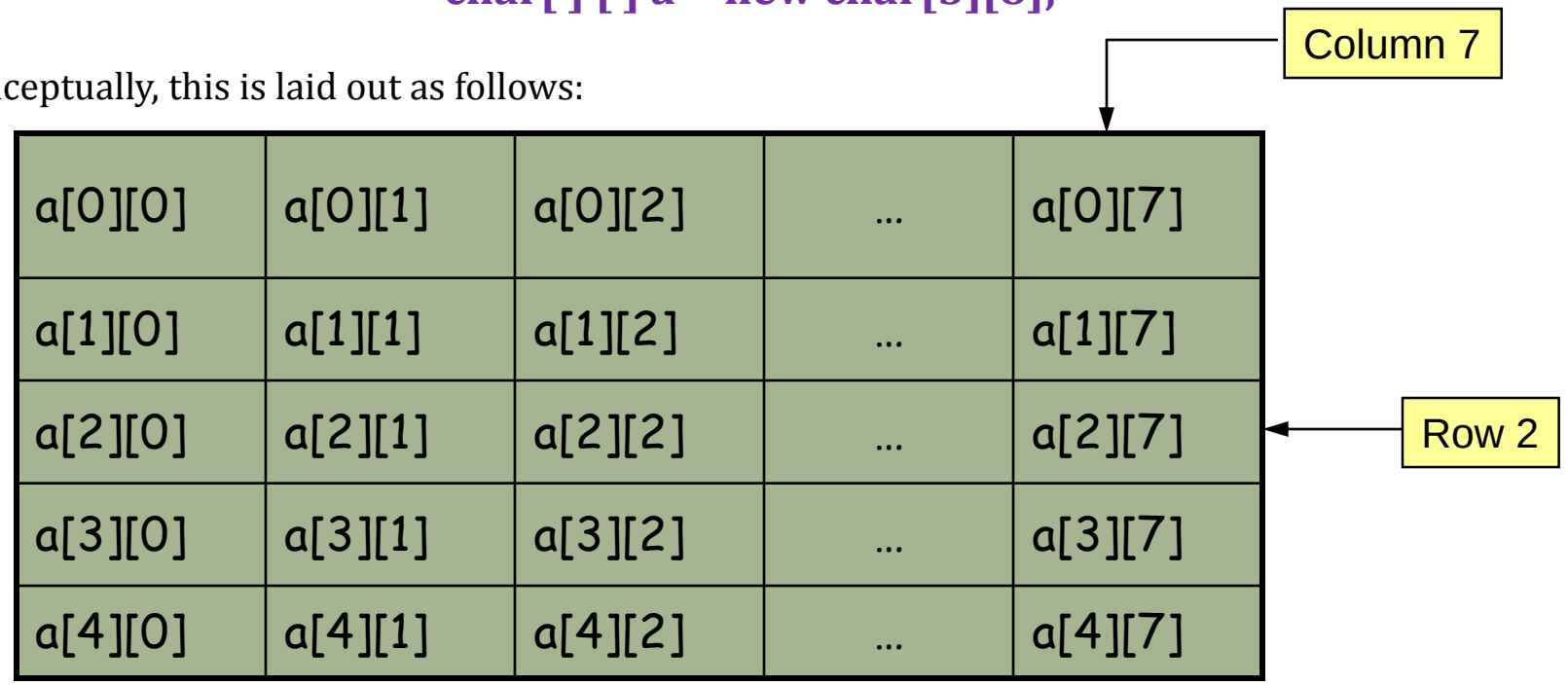
- Java allocates space for the **array of array references**, and then allocates space for the **individual arrays**
- **a:** is of type char[ ][ ], an array of array of characters (whole page)
- **a[4]** : is of type char[ ], an array of characters (a single line)
- **a[4][3]:** is of type char, a single character (character 3 of line 4)

Conceptual Layout

- 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 1st index is the row, the 2nd is the column

Multidimensional Array Length

- Consider the declaration:

  char[ ] [ ] a = new char[2][3];

- What is the meaning of **a.length**?
  - 2? 3? 6?
  - Undefined?
- **Ans**: 2. This is clear from the illustration on the previous page. Array **a** is an **array of 2 references** to other arrays
- What is the meaning of **a[1].length**?
- **Ans**: 3, because **a[1]** is an **array of 3 characters**

Initialization and Iterating

- We can initialize two-dimensional arrays using { }
- Nested for loops are used to iterate over the elements of the array
- We can iterate row by row or column by column
- As it was the case for one-dimensional arrays we can pass them to methods and any changes done to the elements in the method apply to the original array
- We can also pass rows of a two-dimensional array to methods
- **Example:** TwoDimArrayExample.java