CMSC 131
Object-Oriented Programming I

Two-Dim Arrays II

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

Announcements

- Do not post any messages in Piazza under the Annoucements folder or the clarifications folder for a project. Thanks

About Arrays

- Zero-size arrays
  - What space do they occupy?
  - Advantages/disadvantages when compare with null
    - By returning an array of zero size you can iterate through it
- Visualization of arrays through the debugger
  - Let's see an array through the debugger

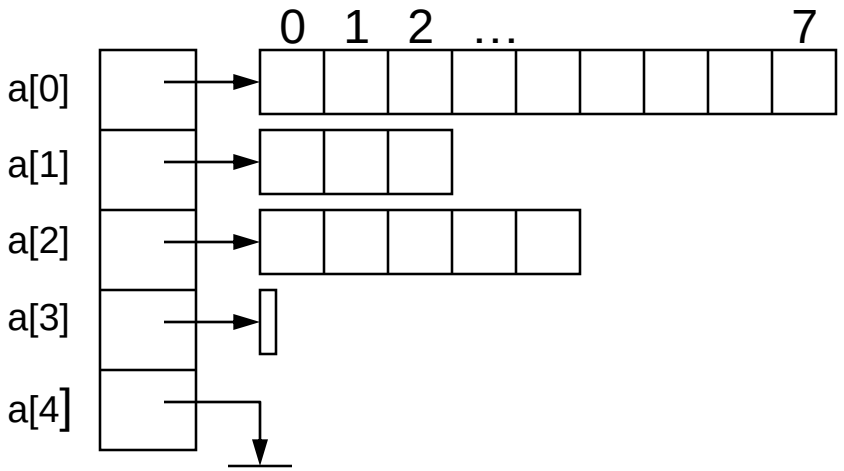
Announcement

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

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.

```
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;
```

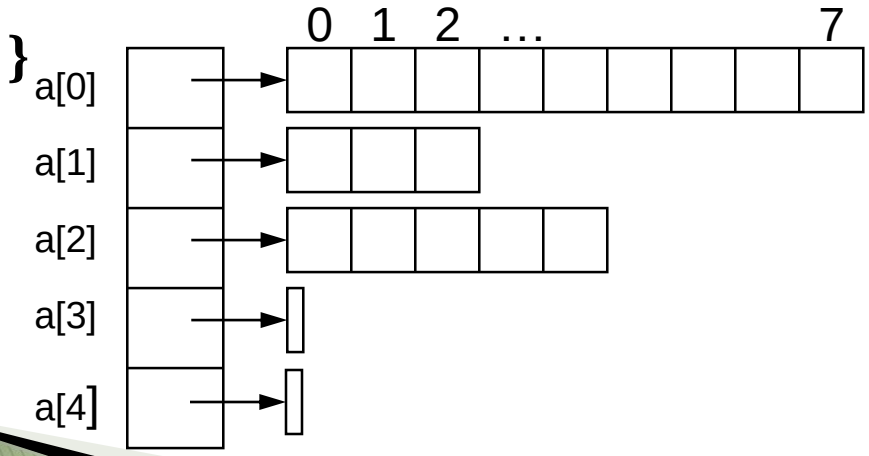


Two-Dimensional Arrays and Loops

- 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

```
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



Would the above nested loop work when a[4] is null?

Multidimensional Initializers

- **1-dim Initializer**: recall

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

- **2-dimensional Initializer**:

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

Note: The size of the array is determined by the initializer.

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

- **Example (Initializers.java):** Print the array.

```
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:

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:

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 use pass by value to pass two-dimensional arrays
  Notice we can pass a row of a two-dimensional array
- Let's see a two-dimensional array through the debugger

Miscellaneous

- 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