- Previous Lecture:
- Polymorphism
- Object class
- Abstract
- Today's Lecture:
- 2-d array
- Reading: Sec 6.4


## Multi-dimensional array

- Can have as many dimensions as you want
- A 2-d array is a 1-d array of 1-d arrays. Each 1-d array has its own constant length $\Rightarrow$ you can have a ragged (not rectangular) 2-d array.


## 2-d arrays

- A 1-d array is a list of values (references)
- A 2-d array is a table of values (references)
- 2-d array is referenced using two index values
- 2-d array in Java is really a 1-d array of 1-d arrays (i.e., an array of objects)
- Orientation (row, column) is only how we choose to visualize the "table"
- Convention: row-major

$$
\text { May } 2,2006
$$



## Accessing a 2-d array

Given a reference $x$ that points to a 2-d int array. . .
What is its height (\# of rows)?
What is $x[0]$ ?
What is the length of the first row?
How to access last element in the second row?
5. How to access last element in last row?

## Example 1

Given a 2-d integer array $x$, calculate the sum of all entries in the array. Assume the array is rectangular.

## Example 2

Given a 2-d array m, re-order the rows such that the row with the highest row sum is the first row.
$m a k 12$


## Example 2

Given a 2-d array m, re-order the rows such that the row with the highest row sum is the first row.
//calculate row sums
//find index of row with max sum
//swap row of max sum with row 0

## What if . . .

- The array is ragged instead of rectangular? Suppose all rows exist but the rows have different lengths.
- Not all rows exist and the existing rows have different lengths?


## Example 3

Given a 2-d array m, re-order the columns such that the column with the highest column sum is the first column. Assume $m$ is rectangular.

How will the code differ from the previous question?

```
public class Array2d {
        public static void main(String[] args) {
            int[][] m= new int[][]{new int[]{1,3,5,9},
                        new int[]{2,100},
                            new int[]{2,2,3} };
            int[] rsum= new int[m.length]; //array of row sums
            for (int r=0; r<m.length; r++) {
                //calculate row sum
                //if current row sum is highest, update highSum, highIndex
            }
            //swap row with highest sum into first row
            //Print 2-d array
            for (int r=0; r<m.length; r++){
            for (int c=0; c<m[r].length; c++)
                System.out.print(m[r][c]+ " ");
                System.out.println();
            }
    }
}
```

