Topic: 2-d array of objects, review of String methods, instantiating 2-d arrays

Reading: Sec 9.3

Example: re-ordering rows

Given a 2-d `int` array `m`, re-order the rows such that the row with the highest row sum is the first row. Assume `m` is in row-majored order.

1	3	5	9
2	100		
2	2	3	

Thought question: what if you want to re-order the array such that the *column* with the highest *column sum* is the first column? How will the code fragment differ? What is the major difference?

`String` methods

Below are some useful methods of the `String` class. Let `s` refer to the `String` “`CS100J`” in the examples below.

Expression	*Returned value*
`s.length()`	
`s.charAt(0)`	
`s.indexOf(‘0’)`	
`s.equals(“CS100”)`	
`s.toLowerCase()`	
`s.toUpperCase()`	

You can learn more `String` methods from the API documentation. Use the API documentation as a resource, but don’t go memorizing all the methods!

Example: cubicle world

Given **`seat`**, a 2-d array of **`String`**s that stores a seating plan, complete the program fragment below to *find the row and seat number of the person whose name is given through user input.* Array **`seat`** has dimensions just big enough to store the entire seating plan including internal spaces. Assume all rows have `length` > 0.

row				
1	Alice	Dilbert	Dogbert	
2	Ratbert		Wally	
3	Asok	Carol	Catbert	P-H Boss
	1	2	3	4
		seat		

```
String target = JLiveRead.readString();

//Set Row#, seat# to -1 if target not found
int foundR= -1;  //Row# of target
int foundC= -1;  //Seat# of target
```

```
//Output location
if (foundR==-1)
  System.out.println(target + “ not found”);
else
  System.out.println(target + “ sits in row ” + foundR + “, seat ” + foundC);
```

Lab Exercise: Creating 2-d arrays

//Declare variable `table` to reference 2-d `int` array

//Set the no. of rows in table—instantiate array in 1st dimension

//Create the individual rows of `table`

//Assign random numbers to cells in `table`