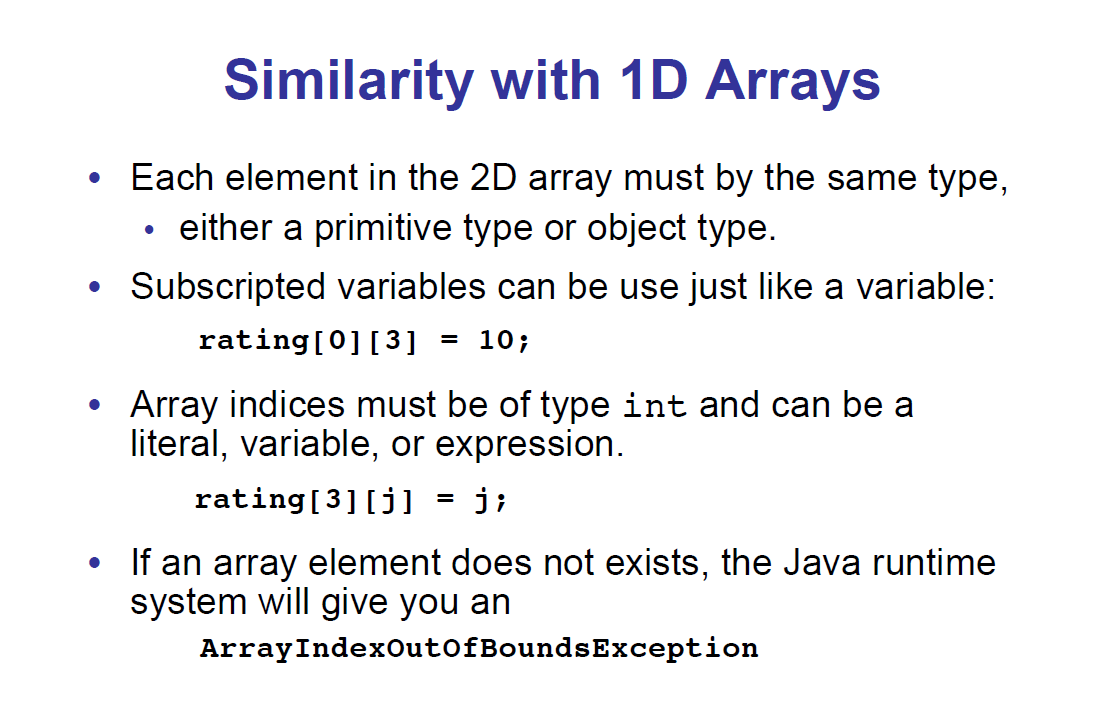
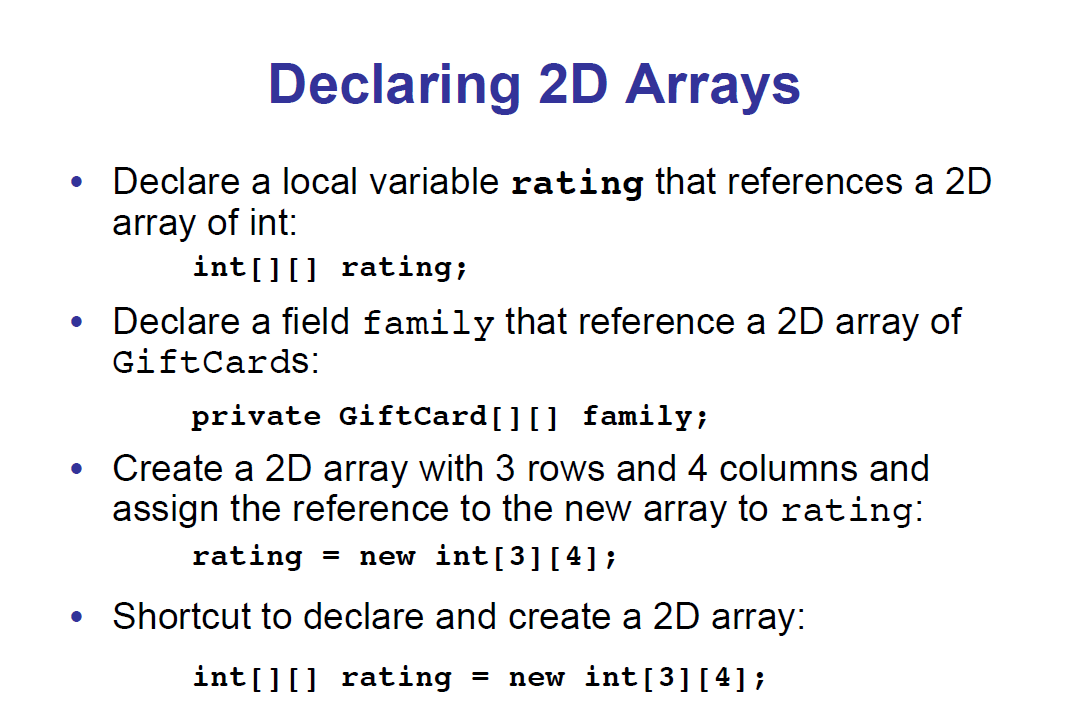
This this software project, we are going to create a 2-dimensional array that tracks ratings of multiple movies by multiple reviewers. Before we get into the details of that project, let us review a bit about 2-dimensional arrays in Java:



The above is a good summary of how 2-dim arrays in java are similar to 1-dim arrays with which you are already familiar.

On the next page there iis a description of how to declare and define a 2-dim array in java:

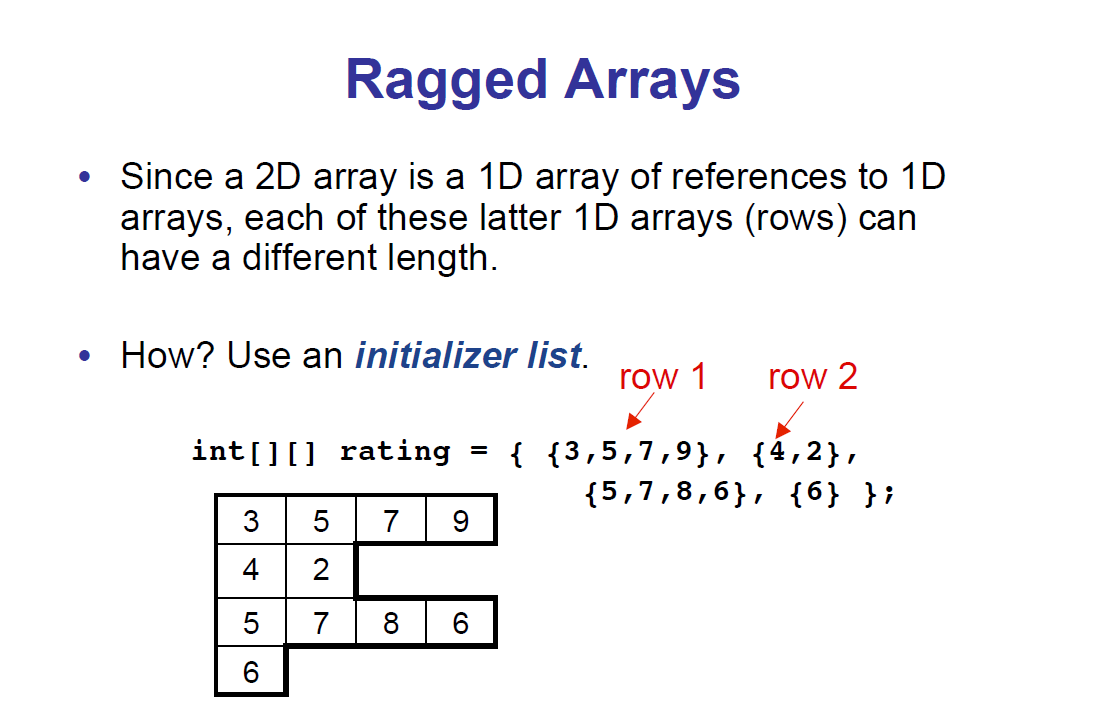


Notice that the declaration shown in the first and second bullet items, above, do not define memory for the two dimensional arrays they will eventually reference - these declarations only tells the compiler that “rating” and “family” will point to a two-dimensional arrays.

The third bullet item shows how we can define memory for a 2-dim array pointer that was declared previously.

The fourth bullet item shows how we can declare a pointer and define the memory for a 2-dim array all on one line.

On the next page, we will show how we can declare, define, and initialize 2-dim arrays in Java on a single line:

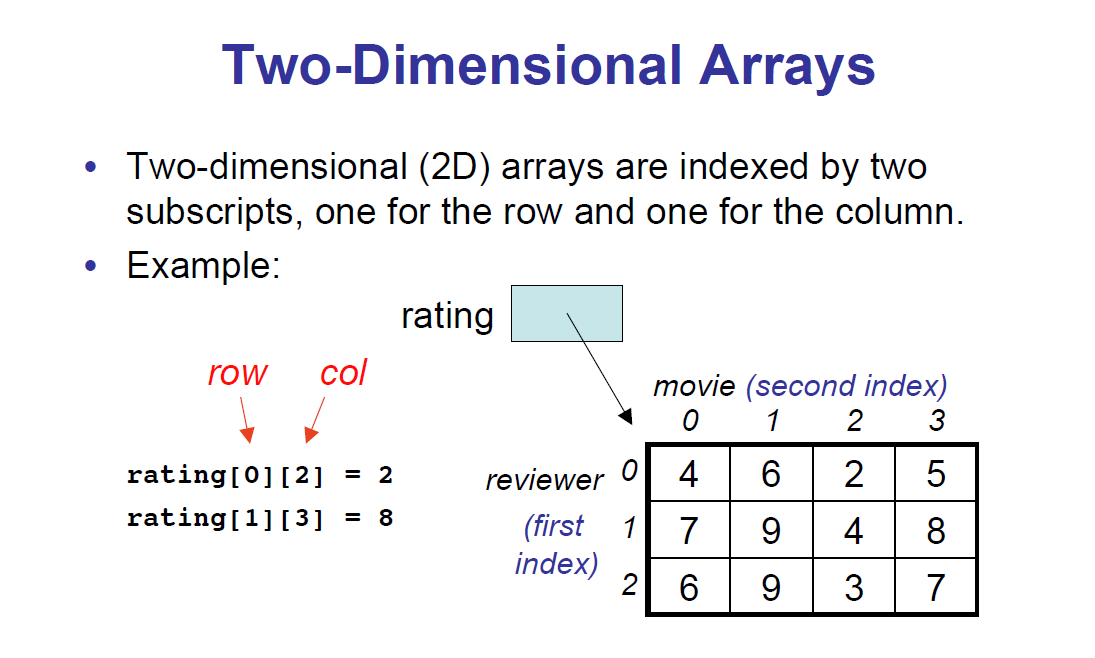


We have used the familiar bracket notation that we used with one-dimensional arrays to initialize two-dimensional arrays.

Notice how the rows of a two-dimensional java array do not all have to be the same size. But this is an advanced concept that will NOT be tested on the AP exam. But I wanted to show it to you here in case you ever encounter it in the field.

The array we build for our movie rating project will have the same number of elements in each row, unlike the array shown above.

**Project Directions:** You are to write several methods that manipulate and report statistics on a given 2-dimensional array. The arrays that you will be working with look like this:



Notice the two-dimensional array “rating” keeps track of multiple movies (each movie has a separate column number) and the ratings given by multiple reviewers (each reviewer has a separate row number).

So looking at the example, above, we can see that reviewer #2 loved movie #1 and gave it a rating of 9.

On the next page, there is a description of all the methods you have to write for this project.

You have to write the following methods. Your methods should not assume that a known number of movies or reviewers is present in the array. You must ask the array how many elements are in each row and column. You can assume that each row is of the same size.

a) **public static double averageRatingByReviewer( int [] [] ratings, int reviewerIndex)**

This method takes a two dimensional ratings array as its first argument and also takes a reviewer number as its second argument. This method should return the average rating by this reviewer over all the movies this reviewer has rated. For example, a call to this method with the rating array shown at the bottom of this page and with the reviewer index set to 2 would return a value of 6.25 because the average of 6,9,3, and 7 is 6.25.

If this method is called with an invalid reviewer index, it should return a value of -1. You can assume that the rating array will never be empty - that at least one movie review will be included.

b) **public static int aboveX( int [] [] ratings, int x)**

This method should search the entire ratings array and return the number of ratings that are larger than x. So for the ratings array shown at the bottom of this page, a call to this method with x set to 8 would return 2 because there are only 2 ratings above an 8 in this ratings array.

c) **public static double averageRating(int [] [] ratings, int movieIndex)**

This method should return the average rating for the movie indexed by the second parameter.

So a call to this method with the movieIndex parameter set to 1 would return an 8.0 because the average of 6, 9, and 9 is 8.0.

When you test your code, set up the runner class (main method) by declaring, defining and initializing a two-dimensional ratings array identical to the one shown below:

