Two-Dimensional Arrays

- A one-dimensional array stores a list of elements
- A two-dimensional array can be thought of as a table of elements, with rows and columns

To be precise, in Java a two-dimensional array is an array of arrays

A two-dimensional array is declared by specifying the size of each dimension separately:

```java
int[][] scores = new int[12][50];
```

A array element is referenced using two index values:

```java
value = scores[3][6];
```

The array stored in one row can be specified using one index

Arrays in Java

- Java represents 2D arrays as an array of arrays!

- In other words, a 2D integer array is really a 1D array of references to 1D integer arrays.

- The concept generalizes to N-dimensions
Two-Dimensional Arrays

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>int[]</td>
<td>2D array of integers, or array of integer arrays</td>
</tr>
<tr>
<td>table[]</td>
<td>int[]</td>
<td>array of integers</td>
</tr>
<tr>
<td>table[5]</td>
<td>int[]</td>
<td>array of integers</td>
</tr>
<tr>
<td>table[5][12]</td>
<td>int</td>
<td>integer</td>
</tr>
</tbody>
</table>

Example of a regular 2D array

Note: In Java the first index should be 0 not 1!

Multidimensional Arrays

- An array can have many dimensions – if it has more than one dimension, it is called a **multidimensional array**
- Each dimension subdivides the previous one into the specified number of elements
- Each dimension has its own `length` constant
- Because each dimension is an array of array references, the arrays within one dimension can be of different lengths
  - these are sometimes called **ragged arrays**

Example of a Ragged Array

Note: In Java the first index should be 0 not 1!

Two-Dimensional Arrays

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>int[]</td>
<td>2D array of integers, or array of integer arrays</td>
</tr>
<tr>
<td>table[]</td>
<td>int[]</td>
<td>array of integers</td>
</tr>
<tr>
<td>table[5]</td>
<td>int[]</td>
<td>array of integers</td>
</tr>
<tr>
<td>table[5][12]</td>
<td>int</td>
<td>integer</td>
</tr>
</tbody>
</table>

3D Array Example
Arrays as Parameters

- An entire array can be passed as a parameter to a method
- Like any other object, the reference to the array is passed, making the formal and actual parameters aliases of each other
- Therefore, changing an array element within the method changes the original
- An individual array element can be passed to a method as well, in which case the type of the formal parameter is the same as the element type

Java Example: Printing an Array

ArrayList Class

- The ArrayList class is part of the java.util package
- Like an array, it can store a list of values and reference each one using a numeric index
- However, you cannot use the bracket syntax with an ArrayList object
- Furthermore, an ArrayList object grows and shrinks as needed, adjusting its capacity as necessary
**The ArrayList Class**

- Elements can be inserted or removed with a single method invocation
- When an element is inserted, the other elements "move aside" to make room
- Likewise, when an element is removed, the list "collapses" to close the gap
- The indexes of the elements adjust accordingly

**The ArrayList Class**

- An ArrayList stores references to the Object class, which allows it to store any kind of object
- See Beatles.java (page 405)
- We can also define an ArrayList object to accept a particular type of object
- The following declaration creates an ArrayList object that only stores Family objects
  ```java
  ArrayList<Family> reunion = new ArrayList<Family>
  ```
- This is an example of generics, which are discussed further in Chapter 12

**ArrayList Efficiency**

- The ArrayList class is implemented using an underlying array
- The array is manipulated so that indexes remain continuous as elements are added or removed
- If elements are added to and removed from the end of the list, this processing is fairly efficient
- But as elements are inserted and removed from the front or middle of the list, the remaining elements are shifted

**Example:** Beatles.java (page 405)
Linear Search

- The most basic
- Very easy to implement
- The array DOESN'T have to be sorted
- All array elements must be visited if the search fails
- Could be very slow

Example: Successful Linear Search

Example: Failed Linear Search

Java Example: Linear Search

Java Example: Finding the minimum number in an array of unsorted integers

THE END