Arrays

Chapter 7

Array Definitions

pp. 376 - 377

- An array is a structured data type
  - Contains many variables of the same data type (called elements)
  - Allows manipulation of all variables with one name
  - Individual elements accessed by an index
  - Legal index values are \(0, \ldots, \text{length} - 1\) (zero-based indexing)
  - Index bounds are checked

\[
\text{scores} = \begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9
\end{bmatrix}
\]

Arrays

Always are instantiated objects

- Reference definition is distinct from object instantiation
  - `int[] numberOfDays;` // reference variable definition
  - `numberOfDays = new int[12];` // object instantiation
  - `int[] numberOfDays = new int[12];` // both

- Static instantiation and initialization
  - `int[] numberOfDays = {3, 12, 31, 30, 31, 30, 31, 30, 31};`
  - `numberOfDays.length;` // capacity, not filled slots

- Bracket placement
  - `int[] numberOfDays;` // preferred
  - `int numberOfDays[];`
  - Useful for method arguments: `void func(int[] scores) {...}`
  - Useful for method return types: `int[] func() {...}`

One-dimensional Array

\[
\text{int[] test = new int[10];}
\]

Two-dimensional Array

\[
\text{float[] testScore = new float[10][4];}
\]

Three-dimensional Array

\[
\text{double[][] classScore = new double[5][5][5];}
\]
Array Syntax

Using arrays

- Arrays are often used with for-loops
- One loop per dimension
- Index goes from 0 to array.length - 1
- Examples
  - for (int i = 0; i < test.length; i++)
    System.out.println(test[i]);
  - for (int i = 0; i < testScore.length; i++)
    for (int j = 0; j < testScore[i].length; j++)
      System.out.println(testScore[i][j]);

The For-Each Loop

Adapted from C#

```java
for <type> <name> : <array> { ... }
for (int i : scores)
  System.out.println(i);
for (String s : args)
  System.out.println(s);
```

Arrays of Objects

Arrays of objects are really arrays of references to objects

```java
Employee[] emp;
emp = new Employee[5];
for (int i = 0; i < emp.length; i++)
  emp[i] = new Employee();
```

Multidimensional Arrays

An array of references

- Array of references to arrays
- First subscript (i) indexes array of references (i.e., rows)
- Second subscript (j) indexes an element in the row
- Rows may be different lengths (each row is instantiated separately)

Command-Line Arguments

Array of Strings (see p. 414)

```java
public static void main (String[] args)
  c:\> java ArgDemo hello world from CS1400
```

Example-- Matrix Addition

Arrays on either side of the assignment operator

```java
double[][] a = new double[m][n];
double[][] b = new double[m][n];
double[][] c = new double[m][n];
for (int i = 0; i < m; i++)
  for (int j = 0; j < n; j++)
    c[i][j] = a[i][j] + b[i][j];
```
Example-- Matrix Multiplication

Arrays and loops: frequent companions

\[ C = AB - c_{ij} = \sum_{k=0}^{p-1} a_{ik} b_{kj} \]

```java
double[][] a = new double[m][p];
double[][] b = new double[p][n];
double[][] c = new double[m][n];
for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
        c[i][j] = 0;
    for (int k = 0; k < p; k++)
        c[i][j] += a[i][k] * b[k][j];
```

Sorting

java.util.Arrays

- static void sort(type[] a)
- static void sort(type[] a, int fromIndex, int toIndex)
  - fromIndex - the index of the first element (inclusive) to be sorted
  - toIndex - the index of the last element (exclusive) to be sorted
- type can be any built-in type
  - The sorting algorithm is a tuned quicksort
  - This algorithm offers n log(n) performance
- type can be Object
  - The sorting algorithm is a modified mergesort
  - This sort is guaranteed to be stable: equal elements will not be reordered as a result of the sort
  - guaranteed n * log(n) performance
  - All elements in the array must implement the Comparable interface.

Searching

java.util.Arrays

- static int binarySearch(type[] a, type key)
  - The array must be sorted into ascending order according to the natural ordering of its elements
  - If the array contains multiple elements with the specified value, there is no guarantee which one will be found
  - Returns the index if key is found, otherwise a negative value
  - type may be any built-in type
  - type may be a class type
    - Class must implement Comparable interface from java.lang
      ```java
      interface Comparable<type>
      {
          public int compareTo(type o);
      }
      ```