ITP20003 Java Programming

Array

(Chapter 7)

This slide is primary taken from the instructor's resource of Java: Introduction to Problem Solving and Programming, 7th ed. by Savitch and then edited partly by Shin Hong

Creating and Accessing Arrays

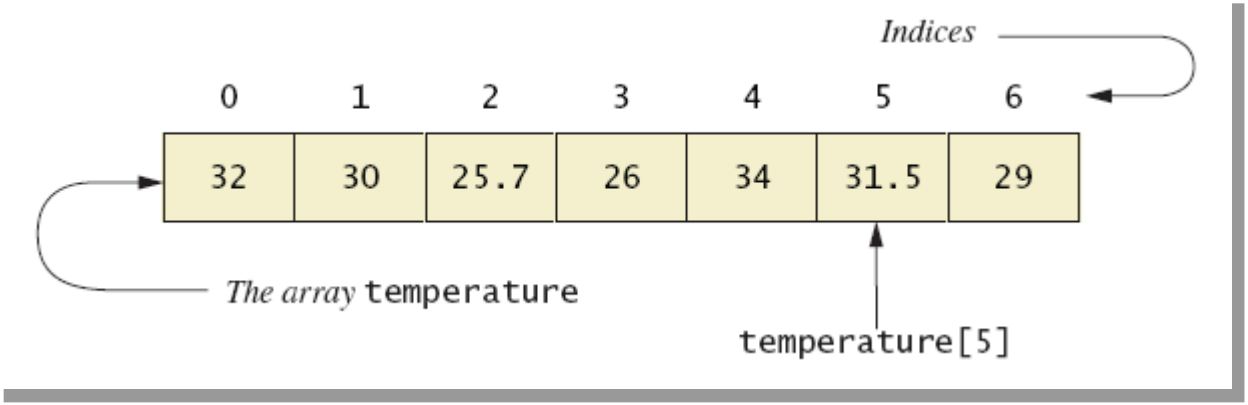
- An array is a special kind of object
- Think of as collection of variables of same type
- Creating an array with 7 variables of type double

```
double[] temperature = new double[7];
```

- To access an element use
  - The name of the array
  - An index number enclosed in braces
- Array indices begin at zero

Creating and Accessing Arrays

- Figure 7.1 A common way to visualize an array



- Note sample program, listing 7.1
  `class ArrayOfTemperatures`

```
import java.util.Scanner;
public class ArrayOfTemperatures
{
    public static void main (String [] args)
    {
        double[] temperature = null ;

         temperature = new double [7];

         System.out.println(temperature[0]) ;

        Scanner keyboard = new Scanner (System.in);
        System.out.println ("Enter 7 temperatures:");
        double sum = 0;
        for (int index = 0 ; index < 7 ; index++)
        {
            temperature [index] = keyboard.nextDouble ();
            sum = sum + temperature [index];
        }
        double average = sum / 7;
        System.out.println ("The average temperature is " + average);
        System.out.println ("The temperatures are");
        for (int index = 0 ; index < 7 ; index++)
        {
            if (temperature [index] < average)
                System.out.println (temperature [index] + " below average.");
            else if (temperature [index] > average)
                System.out.println (temperature [index] + " above average.");
            else //temperature[index] == average
                System.out.println (temperature [index] + " the average.");
        }
        System.out.println ("Have a nice week.");
    }
}
```

Creating and Accessing Arrays

```
Enter 7 temperatures:
32
30
25.7
26
34
31.5
29
The average temperature is 29.7428
The temperatures are
32.0 above average
30.0 above average
25.7 below average
26.0 below average
34.0 above average
31.5 above average
29.0 below average
Have a nice week.
```

Sample screen output

Array Details

- Syntax for declaring an array with **`new`**

```
Base_Type[] Array_Name = new Base_Type[Length];
```

- The number of elements in an array is its length
- The type of the array elements is the array's base type

Square Brackets with Arrays

- With a data type when declaring an array

```
int [ ] pressure;
```

- To enclose an integer expression to declare the length of the array

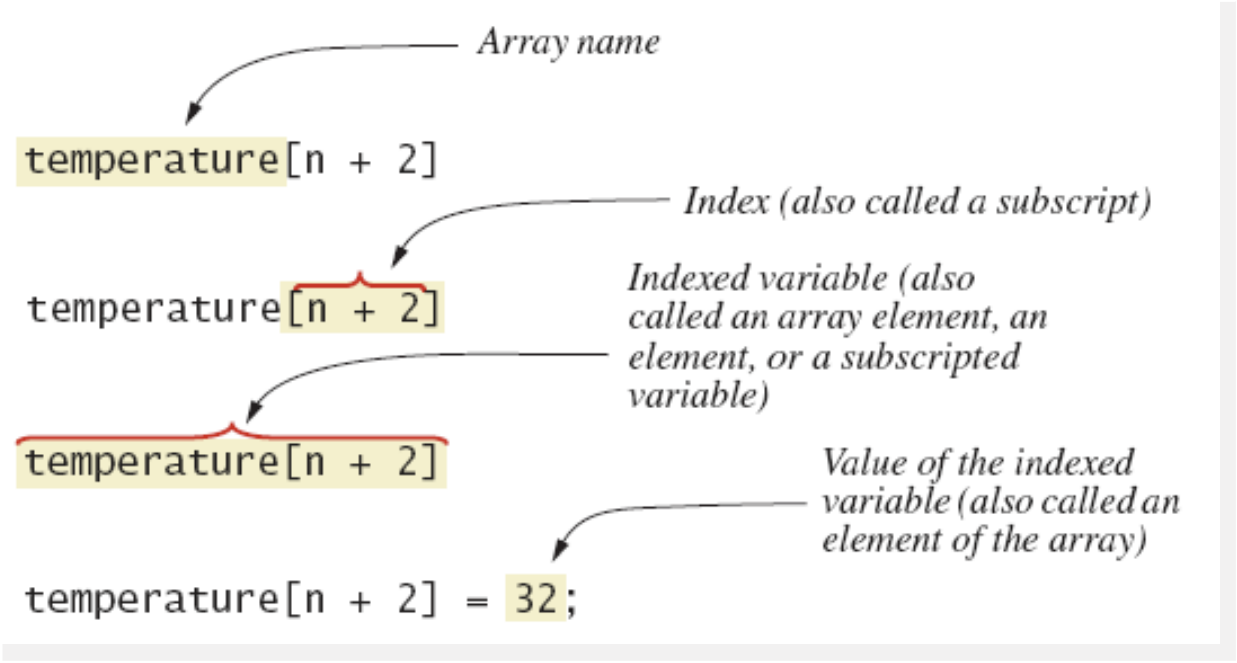
```
pressure = new int [100];
```

- To name an indexed value of the array

```
pressure[3] =
keyboard.nextInt();
```

Array Details

- Figure 7.2 Array terminology



The Instance Variable `length`

- As an object an array has only one public instance variable
  - Variable **`length`**
  - Contains number of elements in the array
  - It is final, value cannot be changed
- Note revised code, listing 7.2
  `class ArrayOfTemperatures2`

More About Array Indices

- Index of first array element is 0
- Last valid Index is **`arrayName.length - 1`**
- Array indices must be within bounds to be valid
  - When program tries to access outside bounds, run time error occurs
- OK to "waste" element 0
  - Program easier to manage and understand
  - Yet, get used to using index 0

Initializing Arrays

- Possible to initialize at declaration time

```
double[] reading = {3.3, 15.8, 9.7};
```

- Also may use normal assignment statements
  - One at a time
  - In a loop

```
int[] count = new int[100];
for (int i = 0; i < 100; i++)
    count[i] = 0;
```

Indexed Variables as Method Arguments

- Indexed variable of an array
  - Example … **`a[i]`**
  - Can be used anywhere variable of array base type can be used
- View program using indexed variable as an argument, listing 7.5
  `class ArgumentDemo`

Entire Arrays as Arguments

- Note – array parameter in a method heading does not specify the length
  - An array of any length can be passed to the method
  - Inside the method, elements of the array can be changed
- When you pass the entire array, do not use square brackets in the actual parameter

Arguments for Method main

- Recall heading of method **`main`**
  `public static void main (String[] args)`
- This declares an array
  - Formal parameter named **`args`**
  - Its base type is **`String`**
- Thus possible to pass to the run of a program multiple strings
  - These can then be used by the program

Array Assignment and Equality

- Arrays are objects
  - Assignment and equality operators behave (misbehave) as specified in previous chapter
- Variable for the array object contains memory address of the object
  - Assignment operator = copies this address
  - Equality operator == tests whether two arrays are stored in same place in memory

Gotcha – Don't Exceed Array Bounds

- The code below fails if the user enters a number like 4. Use input validation.

```
Scanner kbd = new Scanner(System.in);
int[] count = {0,0,0,0};

System.out.println("Enter ten numbers between 0 and 3.");
for (int i = 0; i < 10; i++)
{
 int num = kbd.nextInt();
 count[num]++;
}
for (int i = 0; i < count.length; i++)
   System.out.println("You entered " + count[i] + " " + i + "'s");
```

Gotcha – Creating an Array of Objects

- When you create an array of objects Java does not create instances of any of the objects! For example, consider the code:

```
SalesAssociate[] team = new SalesAssociate[10];
```

- We can't access team[0] yet; it is **null**. First we must create references to an object:

```
team[0] = new SalesAssociate("Jane Doe", 5000);
team[1] = new SalesAssociate("John Doe", 5000);
```

- we can now access team[0].getName() or team[1].getSalary()

Methods that Return Arrays

- A Java method may return an array
- View example program, listing 7.7
  `class ReturnArrayDemo`
- Note definition of return type as an array
- To return the array value
  - Declare a local array
  - Use that identifier in the **`return`** statement

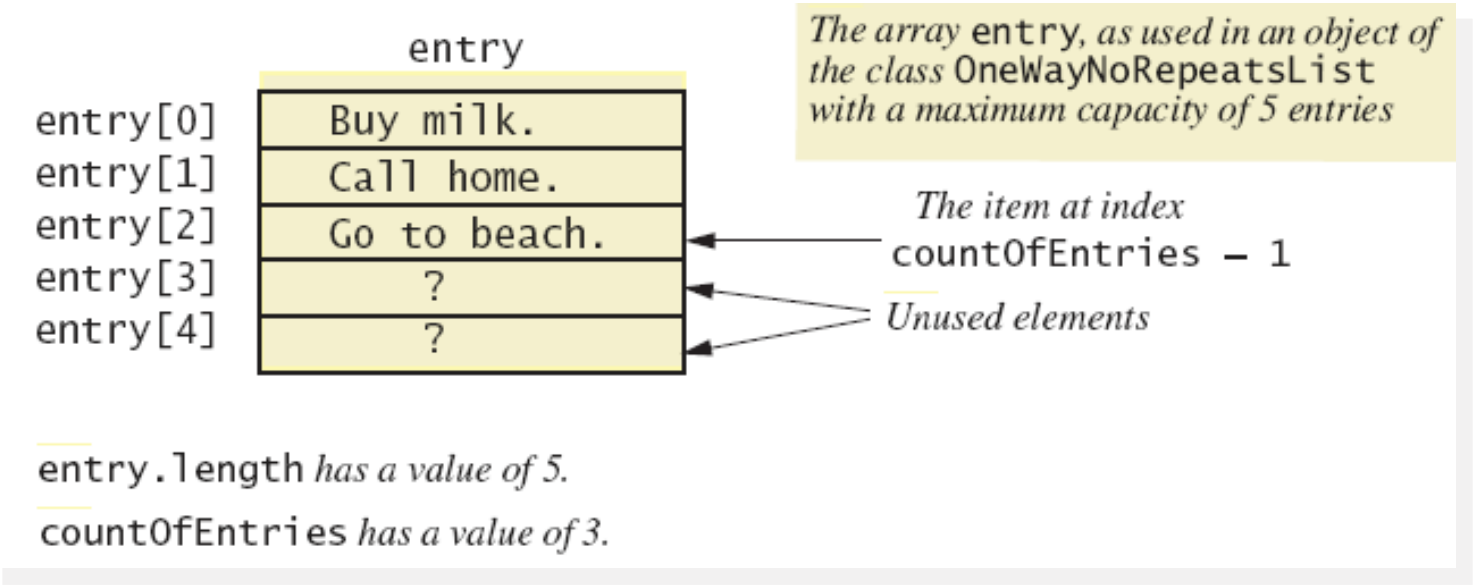
```
import java.util.Scanner;

public class ReturnArrayDemo {
    public static void main (String [] args) {
        Scanner keyboard = new Scanner (System.in);

        System.out.println ("Enter your score on exam 1:");

        int firstScore = keyboard.nextInt ();

        int [] nextScore = new int [3];
        for (int i = 0 ; i < nextScores.length ; i++)
            nextScore [i] = firstScore + 5 * i;

        double [] averageScore = getArrayOfAverages (firstScore, nextScores);
        for (int i = 0 ; i < nextScore.length ; i++)
        {
            System.out.println ("If your score on exam 2 is " + nextScore [i]);
            System.out.println ("your average will be " + averageScore [i]);
        }
    }


    public static double [] getArrayOfAverages (int firstScore, int [] nextScores)
    {
        double [] temp = new double [nextScores.length];
        for (int i = 0 ; i < temp.length ; i++)
            temp [i] = getAverage (firstScore, nextScores[i]);
        return temp;
    }


    public static double getAverage (int n1, int n2)
    {
        return (n1 + n2) / 2.0;
    }
}
```

Partially Filled Arrays

- Array size specified at definition
- Not all elements of the array might receive values
  - This is termed a *partially filled array*
- Programmer must keep track of how much of array is used

Partially Filled Arrays

- Figure 7.4 A partially filled array



Sorting, Searching Arrays: Outline

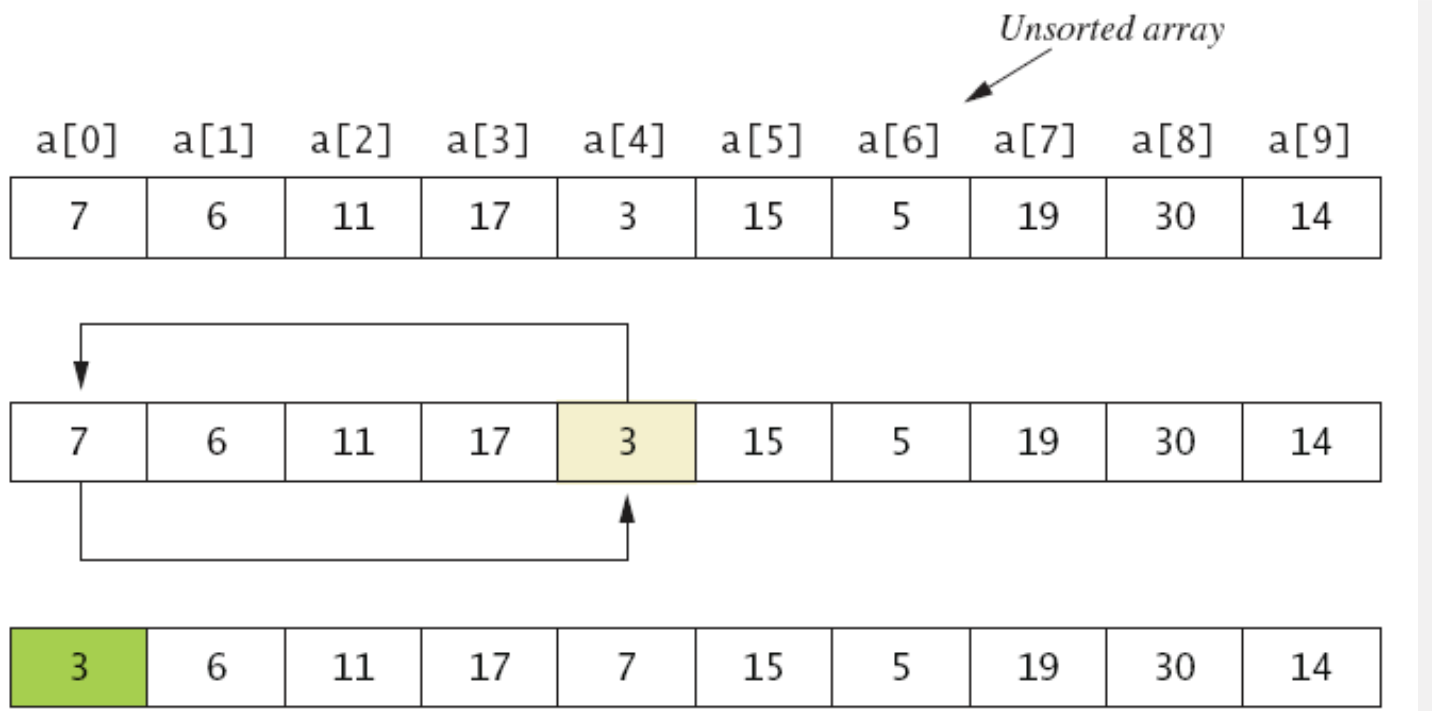
- Selection Sort
- Other Sorting Algorithms
- Searching an Array

Selection Sort

- Consider arranging all elements of an array so they are ascending order
- Algorithm is to step through the array
  - Place smallest element in index 0
  - Swap elements as needed to accomplish this
- Called an interchange sorting algorithm

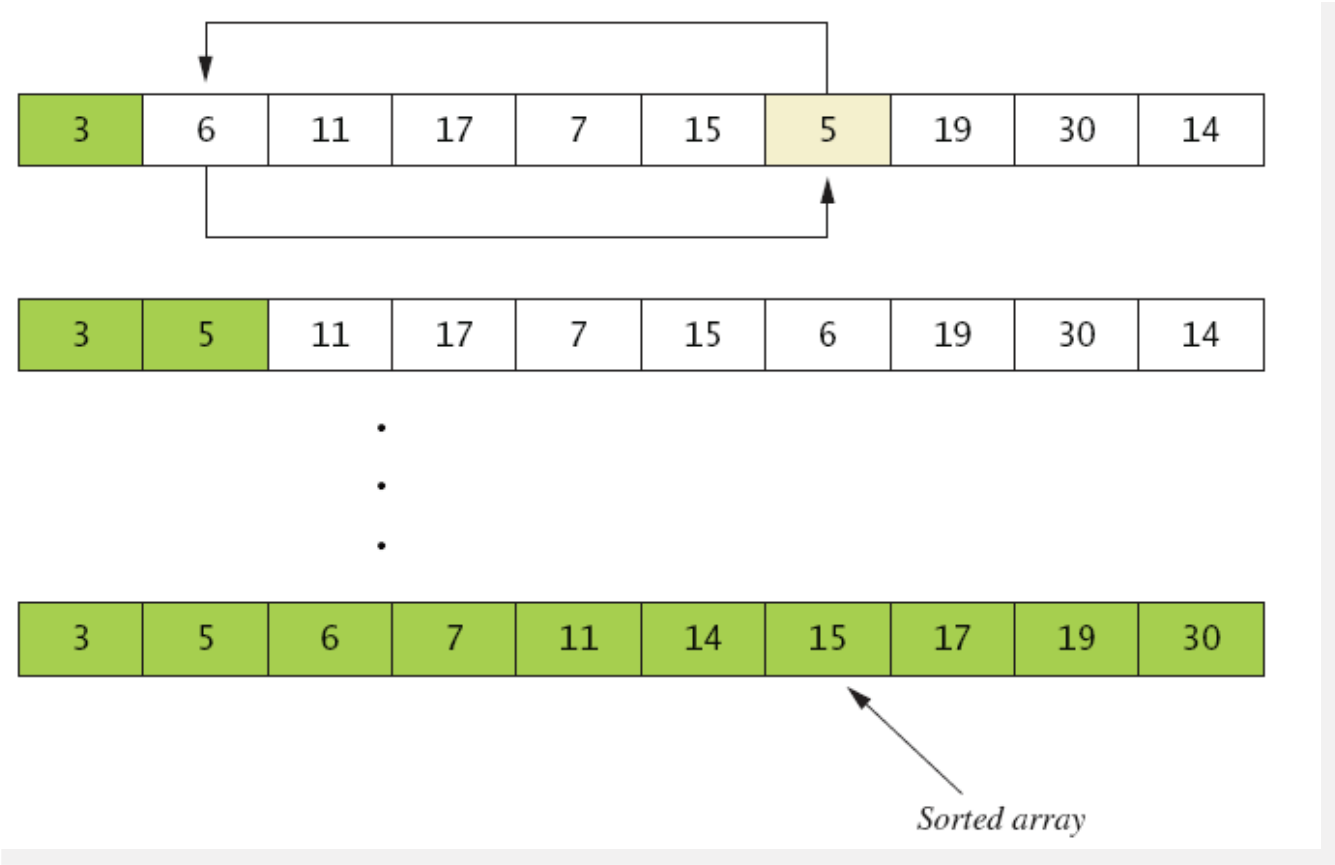
Selection Sort

- Figure 7.5a



Selection Sort

- Figure 7.5b



Selection Sort

- Algorithm for selection sort of an array

```
for (index = 0; index < a.length – 1; index++)
{// Place the correct value in a[index]:
    indexOfNextSmallest = the index of the smallest value among
                          a[index], a[index+1], ..., a[a.length – 1]
    Interchange the values of a[index] and a[indexOfNextSmallest].
    // Assertion: a[0] <= a[1] <=...<= a[index] and these
    // are the smallest of the original array elements.
    // The remaining positions contain the rest of the
    // original array elements.
}
```

Selection Sort

- View implementation of selection sort, listing 7.10
  `class ArraySorter`
- View demo program, listing 7.11
  `class SelectionSortDemo`

```
Array values before sorting:
7 5 11 2 16 4 18 14 12 30
Array values after sorting:
2 4 5 7 11 12 14 16 18 30
```

Sample screen output

Other Sorting Algorithms

- Selection sort is simplest
  - But it is very inefficient for large arrays
- Java Class Library provides for efficient sorting
  - Has a class called Arrays
  - Class has multiple versions of a sort method

Searching an Array

- Method used in **`OneWayNoRepeatsList`** is sequential search
  - Looks in order from first to last
  - Good for unsorted arrays
- Search ends when
  - Item is found … or …
  - End of list is reached
- If list is sorted, use more efficient searches

Multidimensional-Array Basics

- We can access elements of the table with a nested for loop
- Example:

```
for (int row = 0; row < 10; row++)
   for (int column = 0; column < 6; column++)
      table[row][column] =
          balance(1000.00, row + 1, (5 + 0.5 * column));
```

- View sample program, listing 7.12
  `class InterestTable`

Multidimensional-Array Basics

```
Balances for Various Interest Rates Compounded Annually
(Rounded to Whole Dollar Amounts)

Years   5.00%   5.50%   6.00%   6.50%   7.00%   7.50%
1       $1050   $1055   $1060   $1065   $1070   $1075
2       $1103   $1113   $1124   $1134   $1145   $1156
3       $1158   $1174   $1191   $1208   $1225   $1242
4       $1216   $1239   $1262   $1286   $1311   $1335
5       $1276   $1307   $1338   $1370   $1403   $1436
6       $1340   $1379   $1419   $1459   $1501   $1543
7       $1407   $1455   $1504   $1554   $1606   $1659
8       $1477   $1535   $1594   $1655   $1718   $1783
9       $1551   $1619   $1689   $1763   $1838   $1917
10       $1629   $1708   $1791   $1877   $1967   $2061
```

Sample screen output

```
public class InterestTable2
{
    public static final int ROWS = 10, COLUMNS = 6 ;

    public static void main (String [] args) {
        int [] [] table = new int [ROWS] [COLUMNS];
        for (int row = 0 ; row < ROWS ; row++)
            for (int column = 0 ; column < COLUMNS ; column++)
                table [row] [column] = getBalance (1000.00, row + 1, (5 + 0.5 * column));
        showTable (table);
    }

    public static void showTable (int [] [] anArray) {
        for (int row = 0 ; row < ROWS ; row++) {
            System.out.print ((row + 1) + " ");
            for (int column = 0 ; column < COLUMNS ; column++)
                System.out.print ("$" + anArray [row] [column] + " ");
            System.out.println ();
        }
    }

    public static int getBalance (double startBalance, int years, double rate) {
        double runningBalance = startBalance;
        for (int count = 1 ; count <= years ; count++)
            runningBalance = runningBalance * (1 + rate / 100);
        return (int) (Math.round (runningBalance));
    }
}
```

Multidimensional-Array Parameters and Returned Values

- Methods can have
  - Parameters that are multidimensional-arrays
  - Return values that are multidimensional-arrays
- View sample code, listing 7.13
  `class InterestTable2`

Java's Representation of Multidimensional Arrays

- Multidimensional array represented as several one-dimensional arrays
- Given

```
int [][] table = new int [10][6];
```

- Array table is actually 1 dimensional of type **`int[]`**
  - It is an array of arrays
- Important when sequencing through multidimensional array

Ragged Arrays

- Not necessary for all rows to be of the same length
- Example:

```
int[][] b;
b = new int[3][];
b[0] = new int[5]; //First row,  5 elements
b[1] = new int[7]; //Second row, 7 elements
b[2] = new int[4]; //Third row,  4 elements
```

Programming Example

- Employee Time Records
  - Two-dimensional array stores hours worked
    - For each employee
    - For each of 5 days of work week
  - Array is private instance variable of class
- View sample program, listing 7.14
  `class TimeBook`

Programming Example

```
Employee   1    2    3    Totals
Monday     8    0    9    17
Tuesday    8    0    9    17
Wednesday  8    8    8    24
Thursday   8    8    4    20
Friday     8    8    8    24
Total  =   40   24   38
```

Sample screen output

Programming Example

- Figure 7.8 Arrays for the class **`TimeBook`**