SFWR ENG/COMP SCI 2S03 Principles of Programming

Dr. Ridha Khedri

Department of Computing and Software, McMaster University
Canada L8S 4L7, Hamilton, Ontario

Acknowledgments: Material based on *Java actually: A Comprehensive Primer in Programming* (Chapter 6)

Topics Covered

Introduction and Learning Objectives

- We often need to organize values so that they can be processed
- Such an organization of values is called a data structure
- An **array** is a simple form of data structure
- How to declare and access arrays?

Introduction and Learning Objectives

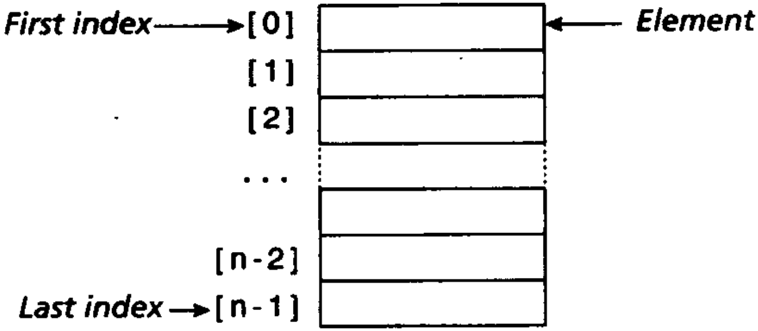
Learning Objectives:

- Using arrays to organize a collection of values
- Declaring array references, creating arrays, and using these references
- Initializing an array
- Iterating over an array
- Creating and using multidimensional arrays
- Generating pseudo-random numbers using the Random class

Arrays as data structures

- An array is a fixed-length sequence of elements
- All elements of an array have the same type
  (An array has a fixed length)
- Each array element can store a value
- The type of the elements is called the (array) element type
- The elements are numbered
- A position in the array is called the index
- The index 0 indicates the position of the first element

Arrays as data structures



Arrays as data structures

- We can create arrays of integers, floating-point numbers, characters, and Boolean values **(primitive data type)**
- We can also create arrays of objects
  - The array does not actually contain the objects
  - It contains only references to objects
- Arrays are themselves objects in Java
- THEREFORE, we can create arrays of arrays

Creating and using arrays

Declaring array reference variables

- An array reference variable is a reference (refer to objects that are arrays)
- In a reference declaration, we have to specify the reference type

```
int[ ] noOfTextMessages;
```

Creating arrays

- We use the *new* operator to create an array

```
noOfTextMessages = new int[7];
```

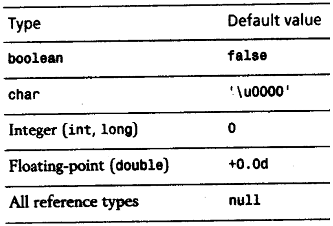
Arrays
Creating and using arrays

Default initialization

- The array creation expression says nothing about which seven int values are stored in the array

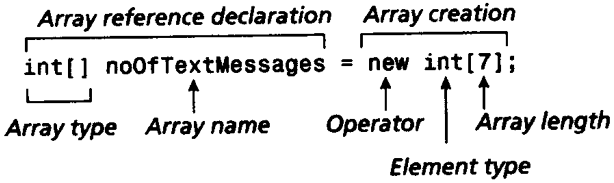
  noOfTextMessages = new int[7];

- **Rule:** When an array is created as above, the elements are automatically initialized to the default value for the element type



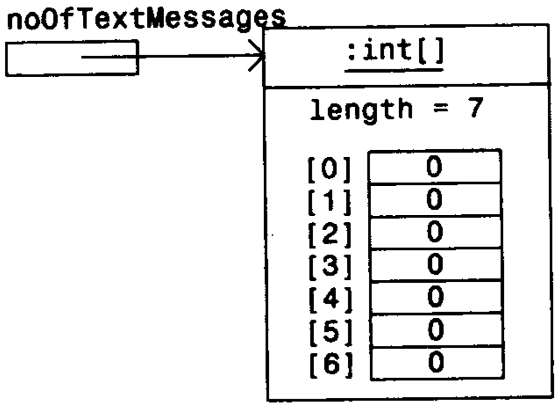
Type	Default value
boolean	false
char	'\u0000'
Integer (int, long)	0
Floating-point (double)	+0.0d
All reference types	null

Creating and using arrays



```
int[] noOfTextMessages = new int[7];
```

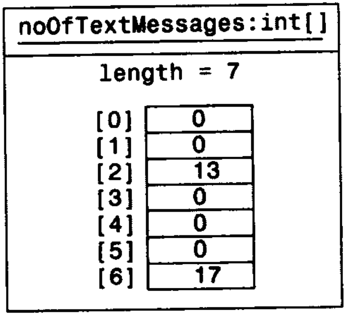
Arrays
Creating and using arrays



After executing:

```
noOfTextMessages = new int[7];
```

Arrays
Creating and using arrays



After executing:

```
noOfTextMessages[2] = 13;
noOfTextMessages[6] = noOfTextMessages[2] + 4;
```

Creating and using arrays

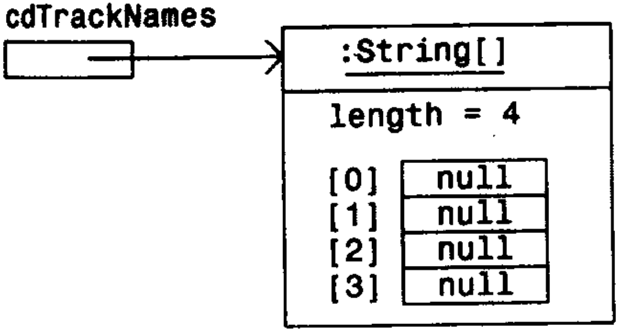
Arrays of objects

- In the same way as creating arrays of primitive types, we can create arrays of objects
- Combining the declaration of the array reference variable and the creation of the array:

```
String[ ] cdTrackNames = new String[4];
```

- What is the implicit default initial value of the four elements? **null**

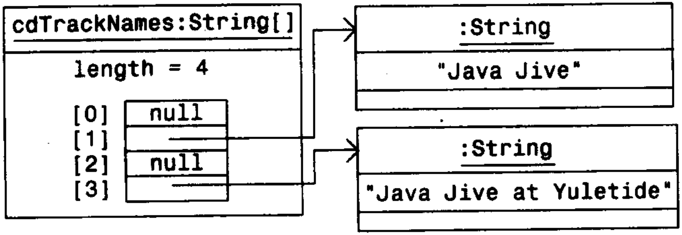
Creating and using arrays



After executing:

```
String[] cdTrackNames = new String[4];
```

Creating and using arrays



After executing:

```
cdTrackNames[1] = "Java Jive";
cdTrackNames[3] = cdTrackNames[1] + " at Yuletide";
```

The length field

- Each array has a field called length whose value is the array length
- The value is set when the array is created (cannot be changed)
- The value of this field can be accessed using the dot notation:

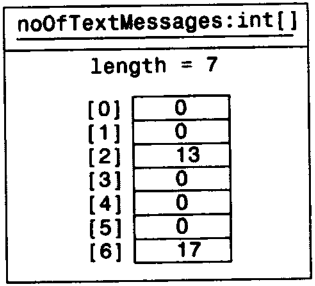
```
System.out.println(CdTrackNames.length);   // Prints 4
```

- Every array has a field called length (Every String class has a method called length ())

Creating and using arrays

Accessing an array element

- To access an element, we need to specify
  - the array reference
  - the index of the element in the array



After executing:

```
noOfTextMessages[2] = 13;
noOfTextMessages[6] = noOfTextMessages[2] + 4;
```

Array bounds

- An index does not need to be integer literal: It can be any arbitrary expression that evaluates to an int value
- $0 \leqslant$ **index value** $<$ **array length**
- At runtime, the index value is always checked before accessing the array
- An invalid index results in an out-of-bounds error: (ArrayIndexOutOfBoundsException)

Arrays

Creating and using arrays

```java
// Array initialisation
public class ArrayInitialisation {
  public static void main(String[] args) {

    final int NO_OF_TESTS = 5;

    // Array declaration:
    int[] testScores;  // (1) Only declaration, no array creation
    // Array creation, default initialisation and assignment:
    testScores = new int[NO_OF_TESTS];        // (2) Array length specified
    assert(testScores != null);
    assert(testScores.length == NO_OF_TESTS);
    assert(testScores[0] == 0);                // First value
    assert(testScores[NO_OF_TESTS - 1] == 0); // Last value
    // and the other elements are also initialised to the default value 0.

    // Combined (1) and (2).
    // Array declration, creation, default initialisation and assignment:
    int[] testScoresII = new int[NO_OF_TESTS];

    // Array declaration:
    int[] testScoresIII;  // (3) Only declaration, no array creation
    // Array creation, explicit initialisation and assignment:
    testScoresIII = new int[] {47, 55, 58, 41, 52}; // (4) Anonymous array
    assert(testScoresIII.length == NO_OF_TESTS);
    assert(testScoresIII[0] == 47);               // First value
    assert(testScoresIII[NO_OF_TESTS -1] == 52); // Last value
    // and the other elements are also explicitly initialised accordingly.

    // Combined (3) and (4)
    // Array declaration, creation, explicit initialisation
    // and assignment:
    int[] testScoresIV = new int[] {47, 55, 58, 41, 52};
    int[] testScoresV = {47, 55, 58, 41, 52}; // Simplified form
//  testScoresV = {47, 55, 58, 41, 52};       // Compile time error!

    System.out.println(testScoresV[NO_OF_TESTS]); // Out-of-bounds error
  }
}
```

Program Output

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
at ArrayInitialisation.main(ArrayInitialisation.java:37)
```

Arrays

Creating and using arrays

Array aliases

- As arrays are objects, we can create aliases to arrays:

  intI[ ] messageCounters = noOfTextMessages;
  String[ ] trackTitles = cdTrackNames;

- **messageCounters** and **noOfTextMessages** are aliases (the same for trackTitles and cdTrackNames)

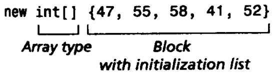
- Any alias to an array can be used to manipulate the array

Alternate notation for array declaration

- There two forms for declaring arrays:
  - **Form 1:** **int[ ] arrayA, arrayB, arrayC;**
  - **Form 2:** **int arrayA[ ], arrayB[ ], arrayC[ ];**
- **Attention:** **int arrayA[ ], arrayB, arrayC[ ];**
- The standard convention is Form 1

Arrays
Initializing arrays

- At the creation of an array, its length field is initialized
- For more value in the elements when we create the array, we must explicitly specify the values
- It is called (explicit) array initialization

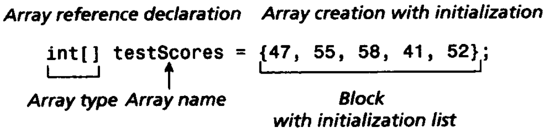


```
new int[] {47, 55, 58, 41, 52}
```

(a) Creating an anonymous array

- It creates an **anonymous array**
- A typical use for an anonymous array: as parameter in a method call

Arrays
Initializing arrays


Array reference declaration
Array creation with initialization
int[] testScores = {47, 55, 58, 41, 52};
Array type
Array name
Block
with initialization list

(b) Array declaration, creation and initialization

- The reference value returned by the array creation expression can be assigned to an array variable

  int[ ] testScores;

  testScores = new int[ ] {47, 55, 58, 41, 52};

- OR

  int[ ] testScores = new int[ ] {47, 55, 58, 41, 52};

Initializing arrays

- Simplified form:
  int[ ] testScores = {47, 55, 58, 41, 52};

Examples

- **boolean[ ] booleanArray = new boolean[ ] {true, false, false, true};**
- **char[ ] charArray = {'J', 'a', 'v', 'a'};**
- **double fpArray= new double[ ] {25.0, 3.14, 1. 5};**
- **String[ ] pets = {"crocodiles", "elephants", "crocophants" , "elediles"};**
- **pets = new String[ ] {"cat", null, "dog"};** // pets[1] does not refer to an object

Initializing arrays

```
// Array initialisation
public class ArrayInitialisation {
  public static void main(String[] args) {

    final int NO_OF_TESTS = 5;

    // Array declaration:
    int[] testScores;  // (1) Only declaration, no array creation
    // Array creation, default initialisation and assignment:
    testScores = new int[NO_OF_TESTS];        // (2) Array length
        specified
    assert(testScores != null);
    assert(testScores.length == NO_OF_TESTS);
    assert(testScores[0] == 0);               // First value
    assert(testScores[NO_OF_TESTS - 1] == 0); // Last value
    // and the other elements are also initialised to the default value
        0.

    // Combined (1) and (2).
    // Array declration, creation, default initialisation and assignment
        :
    int[] testScoresII = new int[NO_OF_TESTS];

    // Array declaration:
    int[] testScoresIII;  // (3) Only declaration, no array creation
    // Array creation, explicit initialisation and assignment:
    testScoresIII = new int[] {47, 55, 58, 41, 52}; // (4) Anonymous
        array
    assert(testScoresIII.length == NO_OF_TESTS);
    assert(testScoresIII[0] == 47);              // First value
    assert(testScoresIII[NO_OF_TESTS -1] == 52); // Last value
    // and the other elements are also explicitly initialised
        accordingly.

    // Combined (3) and (4)
    // Array declaration, creation, explicit initialisation
    // and assignment:
    int[] testScoresIV = new int[] {47, 55, 58, 41, 52};
    int[] testScoresV = {47, 55, 58, 41, 52}; // Simplified form
//  testScoresV = {47, 55, 58, 41, 52};       // Compile time error!

    System.out.println(testScoresV[NO_OF_TESTS]); // Out-of-bounds error
  }
}
```

Iterating over an array

- A common task in programming is accessing elements in an array successively
- Accessing elements of an array successively is called iteration over the array
- How to iterate on an array with *n* elements?
- A counter-controlled **for(; ;)** loop is convenient for iterating over arrays

```
// Code pattern for iterating over an array.
 for (int index = 0; index < array.length; index++) {
        // ... current element given by array[index] ...
  }
```

Iterating over an array

```
/*
 * FindingElementInArray
 * To illustrate comparing all values in an array with a given value
 */

public class FindingElementInArray {
  public static void main(String[] args) {
    int MY_VALUE = 23, position = -1;
    boolean found = false;
    int[] myArray = new int[] {2, 45, 34, 35, 3, 5, 6, 10, 23, 17};
    for (int index = 0; index < myArray.length && !found; index++) {
      if (myArray[index] == MY_VALUE) {
        found = true;
      }
      position++;
    }
    if (found) {
      System.out.printf("The value %3d is found in the array at the
            index %d %n", MY_VALUE, position);
    } else {
      System.out.printf("The value %3d is NOT found in the array%n",
            MY_VALUE);
    }
  }
}
```

Program Output

```
The value  23 is found in the array at the index 8
```

Iterating over an array

```java
/*
 * FindingElementInArray
 * To illustrate comparing all values in an array with a given value
 */

public class SumElementsOfArray {
  public static void main(String[] args) {
    int sum = 0;

    int[] myArray = new int[] {2, 45, 34, 35, 3, 5, 6, 10, 23, 17};

    for (int index = 0; index < myArray.length; index++) {
      sum += myArray[index];
  }

    System.out.printf("The sum of the elements in the array is %4d %n",
         sum);
  }
}
```

Program Output

```
The sum of the elements in the array is  180
```

Arrays

Iterating over an array

```java
import java.util.Scanner;
public class ArrayIteration1 {

  public static void main(String[] args) {
    // Array with the names of week days
    String[] daysOfTheWeek = {"Monday", "Tuesday", "Wednesday",
                              "Thursday", "Friday", "Saturday", "Sunday"
                                  };
    // Array with the no. of text messages sent during a day.
    int[] noOfTextMessages = new int[7];

    // Explicit initialisation
    noOfTextMessages[0] = 20;                          // Monday
    noOfTextMessages[1] = 12;                          // Tuesday
    noOfTextMessages[2] = 13;                          // Wednesday
    noOfTextMessages[3] = noOfTextMessages[1];         // Thursday
    noOfTextMessages[4] = 10;                          // Friday
    noOfTextMessages[5] = noOfTextMessages[0];         // Saturday
    noOfTextMessages[6] = noOfTextMessages[2] + 4;     // Sunday

    // Setup to read from the terminal window.
    Scanner keyboard = new Scanner(System.in);

    // Problem (1) Find how many days have their number of text messages
    // equal to or greater than a specified lower bound.
    System.out.print("Enter the lower bound for " +
                     "the no. of text messages: ");
    int lowerBound = keyboard.nextInt();
    int noOfDays = 0;
    for (int index = 0; index < noOfTextMessages.length; index++) {
      if (noOfTextMessages[index] >= lowerBound) {
        noOfDays++;
      }
    }
    System.out.println("No. of days with more than " + lowerBound +
                       " text messages: " + noOfDays);

    // Problem (2) Find the lowest number of messages sent during the
         week.
    int lowestNoOfTextMessages = noOfTextMessages[0];
    for (int index = 1; index < noOfTextMessages.length; index++) {
      if (lowestNoOfTextMessages > noOfTextMessages[index]) {
        lowestNoOfTextMessages = noOfTextMessages[index];
      }
    }
    System.out.println("Lowest no. of text messages: " +
                       lowestNoOfTextMessages);

    // Problem (3) Find the highest no. of text messages sent during
    // the week and the days on which that number of messages were sent.
    // Find the highest no. of text messages sent during a day.
    int highestNoOfTextMessages = 0;
    for (int index = 0; index < noOfTextMessages.length; index++) {
      if (highestNoOfTextMessages < noOfTextMessages[index]) {
        highestNoOfTextMessages = noOfTextMessages[index];
      }
    }
    System.out.println("Highest no. of text messages: " +
                       highestNoOfTextMessages);
    // Print all days with the highest no.of text messages sent.
    System.out.println("Days with the highest no. of text messages:");
    for (int index = 0; index < noOfTextMessages.length; index++) {
      if (highestNoOfTextMessages == noOfTextMessages[index]) {
        System.out.println(daysOfTheWeek[index]);
      }
```

Iterating over an array of objects

```java
// Array iteration
public class ArrayIteration2 {
  public static void main(String[] args) {
    String[] cdTrackNames = {
      "Symphony No. 1 in C major",
      "Symphony No. 2 in D major",
      "Symphony No. 3 in E-flat major",
      "Symphony No. 4 in B-flat major",
      "Symphony No. 5 in C minor",
      null
    };
    cdTrackNames[5] = cdTrackNames[0] + " (CBC orchestra)";

    // Print all track names with the word "Java" in them.
    for (int trackNumber = 0;                                   // (1)
         trackNumber < cdTrackNames.length;
         trackNumber++) {
      if(cdTrackNames[trackNumber].indexOf("in C") != -1) {
        System.out.println(cdTrackNames[trackNumber]);
      }
    }
  }
}
```

Program Output

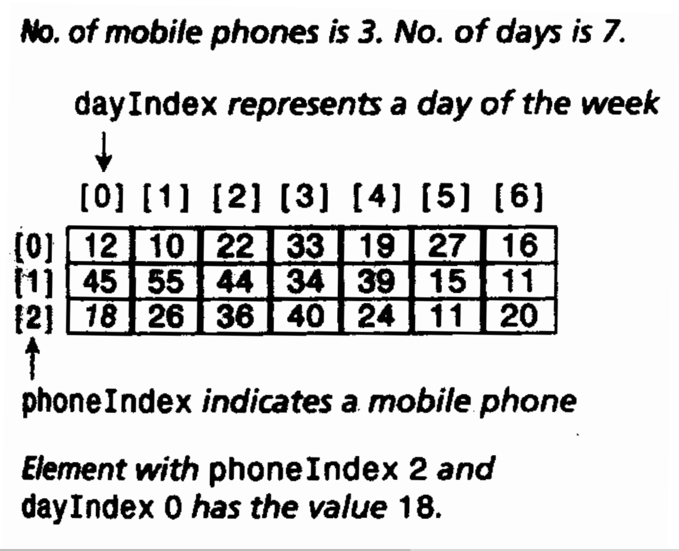
```
Symphony No. 1 in C major
Symphony No. 5 in C minor
Symphony No. 1 in C major (CBC orchestra)
```

Multidimensional arrays

- So far, we used simple or one-dimensional arrays
- To navigate within one-dimensional array, one index is needed
- Instead of using several same size simple arrays, we use multidimensional arrays
- How we can declare, create, initialize and use multidimensional arrays?
- Multidimensional arrays can be implemented in Java by creating arrays of arrays
- The number of indices indicates the dimension of the array

Arrays

Multidimensional arrays



No. of mobile phones is 3. No. of days is 7.

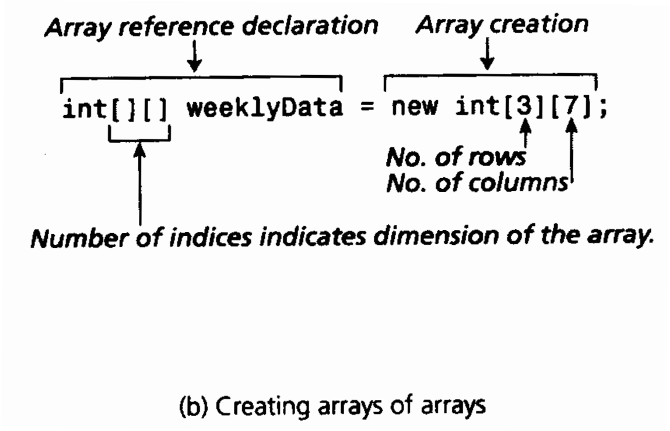
dayIndex *represents a day of the week*

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
[0]	12	10	22	33	19	27	16
[1]	45	55	44	34	39	15	11
[2]	18	26	36	40	24	11	20

phoneIndex *indicates a mobile phone*

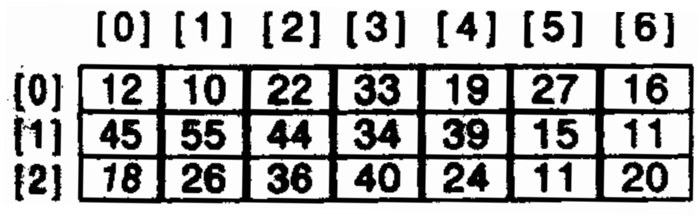
Element with phoneIndex 2 ***and*** dayIndex 0 ***has the value*** 18.

Multidimensional arrays

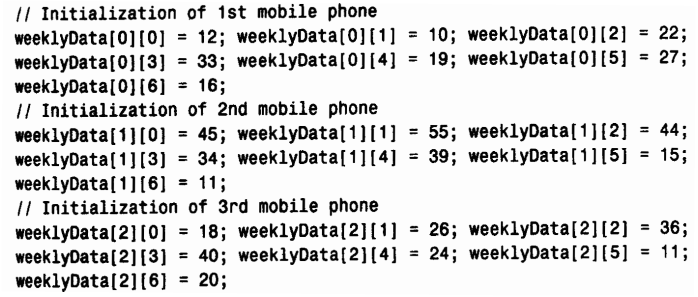

Array reference declaration
Array creation
int[][] weeklyData = new int[3][7];
No. of rows
No. of columns
Number of indices indicates dimension of the array.

(b) Creating arrays of arrays

Arrays
Multidimensional arrays

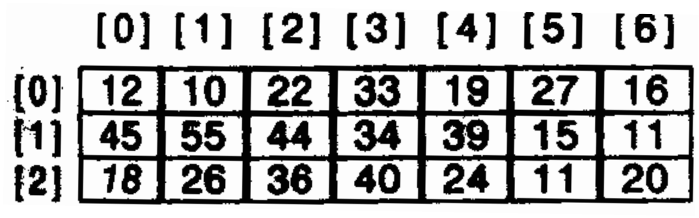


	[0]	[1]	[2]	[3]	[4]	[5]	[6]
[0]	12	10	22	33	19	27	16
[1]	45	55	44	34	39	15	11
[2]	18	26	36	40	24	11	20



```
// Initialization of 1st mobile phone
weeklyData[0][0] = 12; weeklyData[0][1] = 10; weeklyData[0][2] = 22;
weeklyData[0][3] = 33; weeklyData[0][4] = 19; weeklyData[0][5] = 27;
weeklyData[0][6] = 16;
// Initialization of 2nd mobile phone
weeklyData[1][0] = 45; weeklyData[1][1] = 55; weeklyData[1][2] = 44;
weeklyData[1][3] = 34; weeklyData[1][4] = 39; weeklyData[1][5] = 15;
weeklyData[1][6] = 11;
// Initialization of 3rd mobile phone
weeklyData[2][0] = 18; weeklyData[2][1] = 26; weeklyData[2][2] = 36;
weeklyData[2][3] = 40; weeklyData[2][4] = 24; weeklyData[2][5] = 11;
weeklyData[2][6] = 20;
```

Multidimensional arrays

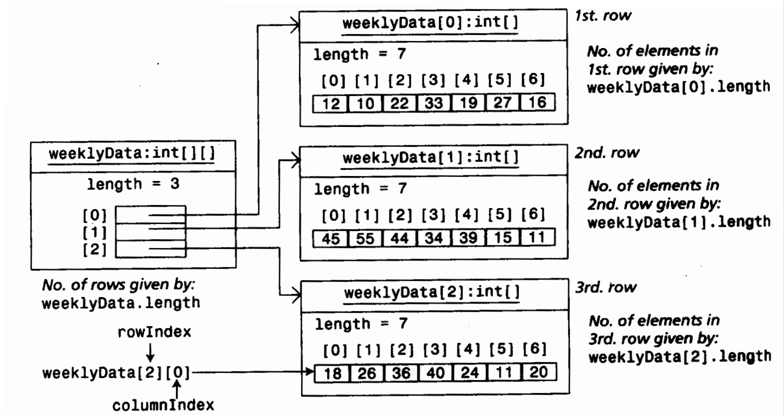


	[0]	[1]	[2]	[3]	[4]	[5]	[6]
[0]	12	10	22	33	19	27	16
[1]	45	55	44	34	39	15	11
[2]	18	26	36	40	24	11	20

- We can also declare, create and initialize the two-dimensional array *weeklyData* as follows:

```
int [ ] [ ] weeklyData = {
      {12, 10, 22, 33, 19, 27, 16},
      {45, 55, 44, 34, 39, 15, 11},
      {18, 26, 36, 40, 24, 11, 20}
}
```

Arrays
Multidimensional arrays



Printing a two-dimensional array

```java
// Multi-dimensional Array Iteration using for(;;) loop
public class MultidimensionalArrayIteration1 {

  public static void main(String[] args) {

    int[][] weeklyData = { // Declaration, creation and initialisation.
        {12, 10, 22, 33, 19, 27, 16},  // 1st mobile phone
        {45, 55, 44, 34, 39, 15, 11},  // 2nd mobile phone
        {18, 26, 36, 40, 24, 11, 20}   // 3rd mobile phone
      };

    // Problem (1) Print the data in tabular form
    for (int phoneIndex = 0;
         phoneIndex < weeklyData.length;
         phoneIndex++) {
      System.out.print("Phone index " + phoneIndex + ": ");
      for (int dayIndex = 0;
           dayIndex < weeklyData[phoneIndex].length;
           dayIndex++) {
        System.out.printf("%4d", weeklyData[phoneIndex][dayIndex]);
      }
      System.out.println();
    }

  }
}
```

Program Output

```
Phone index 0:   12  10  22  33  19  27  16
Phone index 1:   45  55  44  34  39  15  11
Phone index 2:   18  26  36  40  24  11  20
```

Iterating over a specific row

```java
// Multi-dimensional Array Iteration using for(;;) loop
public class MultidimensionalArrayIteration2 {

  public static void main(String[] args) {

    int[][] weeklyData = { // Declaration, creation and initialisation.
        {12, 10, 22, 33, 19, 27, 16}, // 1st mobile phone
        {45, 55, 44, 34, 39, 15, 11}, // 2nd mobile phone
        {18, 26, 36, 40, 24, 11, 20}  // 3rd mobile phone
      };


    // Problem (2) Find the total number of text messages sent from
    // the mobile phone indicated by index 1
    int sumWeek = 0;
    for (int dayIndex = 0; dayIndex < weeklyData[1].length; dayIndex++)
        {
      sumWeek += weeklyData[1][dayIndex];
    }
    System.out.println(
      "Total no. of text messages sent from the mobile phone given" +
      " by index 1: " + sumWeek);
  }
}
```

Program Output

```
Total no. of text messages sent from the mobile phone given by index 1: 243
```

Arrays
Multidimensional arrays
Iterating over all the columns

```java
// Multi-dimensional Array Iteration using for(;;) loop
public class MultidimensionalArrayIteration3 {

  public static void main(String[] args) {

    int[][] weeklyData = { // Declaration, creation and initialisation.
        {12, 10, 22, 33, 19, 27, 16},  // 1st mobile phone
        {45, 55, 44, 34, 39, 15, 11},  // 2nd mobile phone
        {18, 26, 36, 40, 24, 11, 20}   // 3rd mobile phone
      };

    // Problem (3) Find the total number of text messages sent from all
    // mobile phones on Wednesday (day index 2)
    int sumMessages = 0;
    for (int phoneIndex = 0;
         phoneIndex < weeklyData.length;
         phoneIndex++) {
      sumMessages += weeklyData[phoneIndex][2];
    }
    System.out.println(
      "Total no. of text messages sent from all mobile phones on" +
      " Wednesday: " + sumMessages);
  }
}
```

Program Output

```
Total no. of text messages sent from all mobile phones on Wednesday: 102
```

Arrays
Multidimensional arrays
Iterating over all the columns

```java
// Multi-dimensional Array Iteration using for(;;) loop
public class MultidimensionalArrayIteration4 {

  public static void main(String[] args) {

    int[][] weeklyData = { // Declaration, creation and initialisation.
        {12, 10, 22, 33, 19, 27, 16},  // 1st mobile phone
        {45, 55, 44, 34, 39, 15, 11},  // 2nd mobile phone
        {18, 26, 36, 40, 24, 11, 20}   // 3rd mobile phone
      };


    // Problem (4) Find which days the total no. of text messages sent
    // from all mobile phones is greater than 100.
    for (int dayIndex = 0; dayIndex < weeklyData[0].length; dayIndex++)
         {
      int sumDays = 0;
      for (int phoneIndex = 0;
           phoneIndex < weeklyData.length;
           phoneIndex++) {
        sumDays += weeklyData[phoneIndex][dayIndex];
      }
      if (sumDays > 100) {
        System.out.println("The day with index " + dayIndex +
                           " has over 100 text messages registered.");
      }
    }
  }
}
```

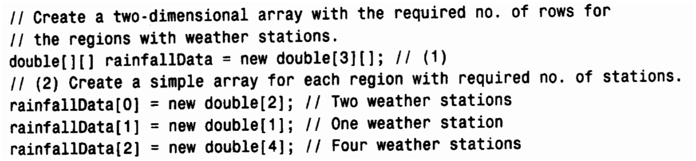
Program Output

```
The day with index 2 has over 100 text messages registered.
The day with index 3 has over 100 text messages registered.
```

Arrays

Ragged arrays

- Ragged arrays:
  - Each row in a two-dimensional array is a simple array
  - HOWEVER, inner simple arrays need not have the same length



```
// Create a two-dimensional array with the required no. of rows for
// the regions with weather stations.
double[][] rainfallData = new double[3][]; // (1)
// (2) Create a simple array for each region with required no. of stations.
rainfallData[0] = new double[2]; // Two weather stations
rainfallData[1] = new double[1]; // One weather station
rainfallData[2] = new double[4]; // Four weather stations
```

Arrays

Ragged arrays

```java
// Using ragged arrays
public class RaggedArrays {

  public static void main(String[] args) {

    // (1) Create and initialise the two-dimensional array
    // with rainfall data
    double[][] rainfallData = {
      {56.6, 30.2},            // Two weather stations
      {20.5},                  // One weather station
      {15.8, 7.0, 45.8, 0.6} // Four weather stations
    };

    // Print rainfall data.
    for (int regionIndex = 0;
         regionIndex < rainfallData.length;
         regionIndex++) {
      System.out.printf("Rainfall for region%2d: ", regionIndex);
      for (int stationIndex = 0;                                  // (2)
           stationIndex < rainfallData[regionIndex].length;
           stationIndex++) {
        System.out.printf("%10.2f",
                              rainfallData[regionIndex][stationIndex]);
      }
      System.out.println();
    }
  }
}
```

Program Output

```
Rainfall for region 0:      56.60     30.20
Rainfall for region 1:      20.50
Rainfall for region 2:      15.80      7.00     45.80      0.60
```

Enhanced for loop

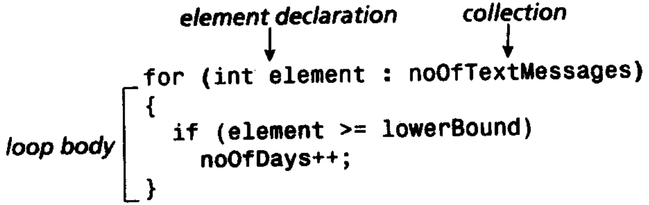
- We often want to iterate over a collection of elements, such as an array, modifying the elements
  (for (; ;) loop)
- We need a for loop that is tailored to successively reading all the values in a collection
  (Enhanced for loop: for(:))
- In each iteration of this loop the current element can be accessed
- The body of the for(:) loop is executed for each value in the collection

Arrays

Enhanced for loop

```
...
for (int index = 0; index < noOfTextMessages.length; index++) {
  if (noOfTextMessages[index] >= lowerBound)
    noOfDays++;
}
```

The for(; ;) loop above rewritten using the enhanced for loop



```
for (int element : noOfTextMessages)
{
  if (element >= lowerBound)
    noOfDays++;
}
```

Enhanced for loop

- The type of the element variable is the element type of the collection
- The for (:) loop iterates over the specified collection
- For each iteration of the loop, the element variable is assigned a new value from the collection
- The element variable is declared in the header
- THEREFORE, it is a local variable
- With the for (:) loop we also avoid out-of-bounds errors

Cases where for(; ;) is preferable to the for(:)

- Requiring the index to access particular element(s) or change element value(s)
- Iteration over more than one collection simultaneously
- Iteration needs to be in increments other than one
- The direction of the iteration is in reverse order

More miscellaneous operations on arrays

```
// Misc. Array Operations
public class MiscArrayOperations1 {
  public static void main(String[] args) {
    // Problem (1) Copying an array of primitive values
    int[] intValuesI = {1, 3, 1949};                // Copy from this
         array
    int[] intValuesII = new int[intValuesI.length];// to this array.
    for (int i = 0; i < intValuesI.length; i++) {
      intValuesII[i] = intValuesI[i];
    }
  }
}
```

More miscellaneous operations on arrays

```
// Misc. Array Operations
public class MiscArrayOperations2 {
  public static void main(String[] args) {
    // Problem (2) Copying an array of objects (Reference value copying)
    String[] refValuesI = {"1.", "March", "1949"}; // Copy from this
         array
    String[] refValuesII = new String[refValuesI.length];// to this
         array.
    for (int i = 0; i < refValuesI.length; i++) {
      refValuesII[i] = refValuesI[i];
    }
   }
}
```

More miscellaneous operations on arrays

```
// Misc. Array Operations
public class MiscArrayOperations3 {
  public static void main(String[] args) {
    // Problem (1) Copying an array of primitive values
    int[] intValuesI = {1, 3, 1949};               // Copy from this
         array
    int[] intValuesII = new int[intValuesI.length];// to this array.
    // Problem (3) Comparing arrays of primitive values
    boolean equalValues = true;
    for (int i = 0; equalValues && i < intValuesI.length; i++){
      if (intValuesI[i] != intValuesII[i]) {
        equalValues = false;
      }
    System.out.printf("IntValueI[%d] is equal to %d and IntValueII[%d]
         is equal to %d %n", i, intValuesI[i], i, intValuesII[i]);
    }
    String notStr = "not ";
    if (equalValues) {
      notStr ="";
    }
    System.out.println("Arrays intValuesI and intValuesII are " + notStr
         +
                         "equal");
  }
}
```

Program Output

```
IntValueI[0] is equal to 1 and IntValueII[0] is equal to 0
Arrays intValuesI and intValuesII are not equal
```

More miscellaneous operations on arrays

```
// Misc. Array Operations
public class MiscArrayOperations4 {
  public static void main(String[] args) {
    String[] refValuesI = {"1.", "March", "1949"}; // Copy from this
         array
    String[] refValuesII = new String[refValuesI.length];// to this
         array.
    for (int i = 0; i < refValuesI.length; i++) {
      refValuesII[i] = refValuesI[i];
    }

    // Problem (4) Comparing arrays of objects for reference value
         equality
    String[] refValuesIII = {"1949", "March", "1."};
    boolean equalRefValues = true;
    for (int i = 0; equalRefValues && i < refValuesIII.length; i++) {
      if (refValuesIII[i] != refValuesII[i]) {
        equalRefValues = false;
      }
    }
    String notStr = "not ";
    if (equalRefValues) {
      notStr ="";
    }
    System.out.println("Arrays refValuesIII and refValuesII are " +
                       notStr + "equal");
  }
}
```

Program Output

```
Arrays refValuesIII and refValuesII are not equal
```

Arrays

Working with partially-filled arrays

```
import java.util.Scanner;

public class PartiallyFilledArrays {
  public static void main(String[] args) {

    // Setup to read from the terminal window
    Scanner keyboard = new Scanner(System.in);

    // Create the array to hold maximum 50 words
    String[] sentence = new String[50];

    System.out.print("Enter a sentence (terminate with \"EOL\"): ");

    int wordIndex = -1;
    String word = keyboard.next(); // Read the first word.
    while (!word.equals("EOL") && wordIndex < sentence.length) { // (1)
      wordIndex++;                   // Index is incremented before
          storing
      sentence[wordIndex] = word;
      word = keyboard.next();        // Read the next word.
    }
    int wordCount = wordIndex + 1;
    System.out.println("No. of words: " + wordCount);

    // Print the words in reverse.
    for (int i = wordCount - 1; i >= 0; i--) {                  // (2)
      System.out.printf("%s ", sentence[i]);
    }
    System.out.println();
  }
}
```

Program Output

```
Enter a sentence (terminate with "EOL"): Don't worry, be happy. EOL
No. of words: 4
happy. be worry, Don't
```