

Arrays

What is array?

- A group of contiguous memory locations that all have the same name and the same type. Each element in the array is referred by its subscript (position number/index)

What is array?

Name of array (Note that all elements of this array have the same name, c)

Position number (index of subscript) of the element within array c

c[0]	-45
c[1]	6
c[2]	0
c[3]	72
c[4]	1543
c[5]	-89
c[6]	0
c[7]	62
c[8]	-3
c[9]	1
c[10]	6453
c[11]	78

Fig. 7.1 A 12-element array.

Java syntax for array

- Declaration
 - `int c[];`
 - `int[] c;`
- Allocation
 - `c = new int[12];`
 - or we could combine both as
 - `int c[] = new int [12];`
- Explicit declaration
 - `int fibo[] = { 1, 1, 2, 3, 5, 8, 13, 21 };`

Arrays (cont.)

- Subscript
 - Also called an *index*
 - Position number in square brackets
 - Must be integer or integer expression

```
a = 5;  
b = 6;  
c[ a + b ] += 2;
```

- Adds 2 to `c[11]`
- Subscripted array name is an *lvalue*

Java syntax for array

- Using array
 - `int prime[] = { 1, 2, 3, 5, 7, 11, 13, 17, 19 };`
 - How many elements in this array?
 - `a = prime[5];`
 - What is a? What should be its data type?
 - `b = prime[a];`
 - `c = prime[a - prime[3]];`

Features

- `Array.length` attribute
- Argument passing into methods and return type

Using an Initializer List to Initialize Elements of an Array

- Initialize array elements
 - Use *initializer list*
 - Items enclosed in braces ({})
 - Items in list separated by commas
- ```
int n[] = { 10, 20, 30, 40, 50 };
```
- Creates a five-element array
  - Subscripts of 0, 1, 2, 3, 4
- Do not need operator `new`

## Arrays and Loops

```
int intArray = { 3, 4, 5, 6, 7, 8, 9 };
```

```
for(int i = 0; i < intArray.length; i++) {
 System.out.println(intArray[i]);
}
```

```
// As long as we're in the array's range, were
// fine...
```

## In class example with FiboArray

## References and Reference Parameters

- Two ways to pass arguments to methods
  - Pass-by-value
    - Copy of argument's value is passed to called method
    - In Java, every primitive is pass-by-value
  - Pass-by-reference
    - Caller gives called method direct access to caller's data
    - Called method can manipulate this data
    - Improved performance over pass-by-value
    - In Java, every object is pass-by-reference
      - In Java, arrays are objects
        - Therefore, arrays are passed to methods by reference

## Passing Arrays to Methods

- To pass array argument to a method
    - Specify array name without brackets
      - Array `hourlyTemperatures` is declared as
- ```
int hourlyTemperatures = new int[ 24 ];
```
- The method call
- ```
modifyArray(hourlyTemperatures);
```
- Passes array `hourlyTemperatures` to method `modifyArray`

# Sorting Arrays

- **Sorting data**
  - Attracted intense research in computer-science field
  - **Bubble sort**
    - Smaller values “bubble” their way to top of array
    - Larger values “sink” to bottom of array
    - Use nested loops to make several passes through array
      - Each pass compares successive pairs of elements
        - Pairs are left alone if increasing order (or equal)
        - Pairs are swapped if decreasing order

```

3
4 // Java core packages
5 import java.awt.*;
6
7 // Java extension packages
8 import javax.swing.*;
9
10 public class BubbleSort extends JApplet {
11 // Line 19
12 // Declare 10-int array with initializer list
13 public void init() {
14 {
15 JTextArea outputArea = new JTextArea();
16 Container container = getContentPane();
17 container.add(outputArea);
18
19 int array[] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
20
21 String output = "Data items in original order\n";
22
23 // append original array values to String output
24 for (int counter = 0; counter < array.length; counter++)
25 output += " " + array[counter];
26
27 bubbleSort(array); // sort array
28
29 output += "\n\nData items in ascending order\n";
30
31 // append sorted array values to String output
32 for (int counter = 0; counter < array.length; counter++)
33 output += " " + array[counter];

```

```

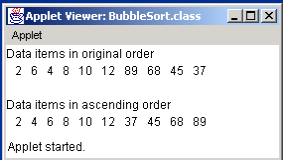
35 outputArea.setText(output);
36 }
37
38 // sort elements of array with bubble sort
39 public void bubbleSort(int array2[])
40 {
41 // loop to control number of passes
42 for (int pass = 1; pass < array2.length; pass++) {
43 // loop to control number of comparisons
44 for (int element = 0; element < array2.length - 1; element++) {
45 // compare side-by-side elements and swap them if
46 // first element is greater than second element
47 if (array2[element] > array2[element + 1])
48 swap(array2, element, element + 1);
49 } // end loop to control comparisons
50 } // end loop to control passes
51 } // end method bubbleSort
52
53 // swap two elements of an array
54 public void swap(int array3[], int first, int second)
55 {
56 int hold; // temporary holding area for swap
57
58 hold = array3[first];
59 array3[first] = array3[second];
60 array3[second] = hold;
61 }

```

```

BubbleSort.java
Line 39
Method bubbleSort
receives array
reference as parameter
Lines 42-47
Use loop and nested
loop to make passes
through array
Lines 51-52
If pairs are in decreasing
order, invoke method
swap to swap pairs
Lines 61-68
Method swap swaps
two values in array
reference

```



# Searching Arrays: Linear Search and Binary Search

- **Searching**
  - Finding elements in large amounts of data
    - Determine whether array contains value matching *key value*
  - Linear searching
  - Binary searching

# Searching an Array with Linear Search

- **Linear search**
  - Compare each array element with *search key*
    - If search key found, return element subscript
    - If search key not found, return **-1** (invalid subscript)
  - Works best for small or unsorted arrays
  - Inefficient for larger arrays

```

5 import java.awt.*;
6 import java.awt.event.*;
7 import java.text.*;
8
9 // Java extension packages
10 import javax.swing.*;
11
12 public class BinarySearch extends JApplet
13 implements ActionListener {
14
15 JLabel enterLabel, resultLabel;
16 JTextField enterField, resultField;
17 JTextArea output;
18
19 int array[];
20 String display = "";
21
22 // set up applet's GUI
23 public void init()
24 {
25 // get content pane and set its layout to FlowLayout
26 Container container = getContentPane();
27 container.setLayout(new FlowLayout());
28
29 // set up JLabel and JTextField for user input
30 enterLabel = new JLabel("Enter integer search key");
31 container.add(enterLabel);
32
33 enterField = new JTextField(10);
34 container.add(enterField);
35

```

BinarySearch.java  
Line 19  
Declare array of ints

Declare array of ints

```

36 // register this applet as enterField's action listener
37 enterField.addActionListener(this);
38
39 // set up JLabel and JTextField for displaying results
40 resultLabel = new JLabel("Result");
41 container.add(resultLabel);
42
43 resultField = new JTextField(20);
44 resultField.setEditable(false);
45 container.add(resultField);
46
47 // set up JTextArea for displaying comparison
48 output = new JTextArea(6, 60);
49 output.setFont(new Font("Monospace",
50 container.add(output);
51
52 // create array and fill with even integers 0 to 28
53 array = new int[15];
54
55 for (int counter = 0; counter < array.length; counter++)
56 array[counter] = 2 * counter;
57
58 } // end method init
59
60 // obtain user input and call method binarySearch
61 public void actionPerformed(ActionEvent actionEvent)
62 {
63 // input also can be obtained with enterField.getText()
64 String searchKey = actionEvent.getActionCommand();
65
66 // initialize display string for new search
67 display = "Portions of array searched:\n";
68

```

BinarySearch.java  
Lines 53-56  
Allocate 15 ints for array and populate array with even ints  
Line 61  
Invoked when user presses Enter

Allocate 15 ints for array and populate array with even ints

Invoked when user presses Enter

```

70 int element =
71 binarySearch(array, Integer.parseInt(searchKey));
72
73 output.setText(display);
74
75 // display search result
76 if (element != -1)
77 resultField.setText(
78 "Found value in element " + element);
79 else
80 resultField.setText("Value not found");
81
82 } // end method actionPerformed
83
84 // method to perform binary search of an array
85 public int binarySearch(int array[], int key)
86 {
87 int low = 0; // low element subscript
88 int high = array.length - 1; // high element subscript
89 int middle; // middle element subscript
90
91 // loop until low subscript is greater than high subscript
92 while (low <= high) {
93
94 // determine middle element subscript
95 middle = (low + high) / 2;
96
97 // display subset of array elements used in this
98 // iteration of binary search loop
99 buildOutput(array, low, middle, high);
100

```

BinarySearch.java  
Lines 70-71  
Invoke method  
binarySearch, using array and search key as arguments  
Lines 102-103  
If search key matches middle array element, return element subscript

Invoke method binarySearch, using array and search key as arguments

If search key matches middle array element, return element subscript

If search key matches middle array element, return element subscript

```

105 // if key less than middle element, set new high element
106 else if (key < array[middle])
107 high = middle - 1;
108
109 // key greater than middle element, repeat search on first array half
110 else
111 low = middle + 1;
112
113
114 return -1; // key not found
115
116 } // end method binarySearch
117
118 // build row of output showing subset of array elements
119 // currently being processed
120 void buildOutput(int array[],
121 int low, int middle, int high)
122 {
123 // create 2-digit integer number format
124 DecimalFormat twoDigits = new DecimalFormat("00");
125
126 // loop through array elements
127 for (int counter = 0; counter < array.length;
128 counter++) {
129
130 // if counter outside current array subset, skip
131 // padding spaces to String display
132 if (counter < low || counter > high)
133 display += " ";
134

```

BinarySearch.java  
Lines 106-107  
If search key is less than middle array element, repeat search on first array half  
Lines 110-111  
If search key is greater than middle array element, repeat search on second array half  
Lines 120-121  
Method buildOutput displays array contents being searched

If search key is less than middle array element, repeat search on first array half

If search key is greater than middle array element, repeat search on second array half

Method buildOutput displays array contents being searched

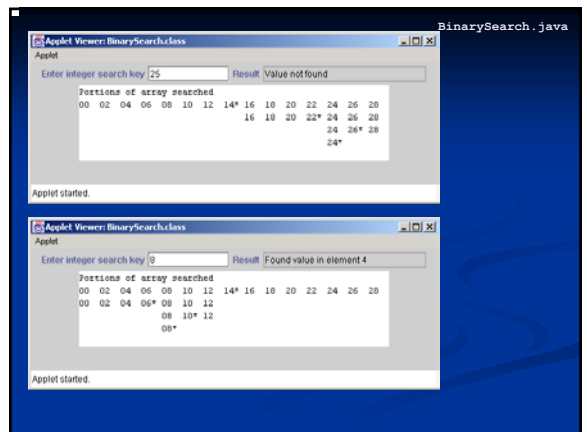
```

135 // if middle element, append element to String display
136 // followed by asterisk (*) to indicate middle element
137 else if (counter == middle)
138 display +=
139 twoDigits.format(array[counter]) + "* ";
140
141 // append element to String display
142 else
143 display +=
144 twoDigits.format(array[counter]) + " ";
145
146 } // end for structure
147
148 display += "\n";
149
150 } // end method buildOutput
151
152 } // end class BinarySearch

```

BinarySearch.java  
Line 139  
Display an asterisk next to middle element

Display an asterisk next to middle element



## Multidimensional Array

|       | Column 0    | Column 1    | Column 2    | Column 3    |
|-------|-------------|-------------|-------------|-------------|
| Row 0 | a[ 0 ][ 0 ] | a[ 0 ][ 1 ] | a[ 0 ][ 2 ] | a[ 0 ][ 3 ] |
| Row 1 | a[ 1 ][ 0 ] | a[ 1 ][ 1 ] | a[ 1 ][ 2 ] | a[ 1 ][ 3 ] |
| Row 2 | a[ 2 ][ 0 ] | a[ 2 ][ 1 ] | a[ 2 ][ 2 ] | a[ 2 ][ 3 ] |

Diagram illustrating the structure of a multidimensional array. The array is represented as a grid of elements. The first row is labeled 'Row 0' and the first column is labeled 'Column 0'. The elements are shown as `a[ row ][ column ]`. Labels with arrows point to the components: 'Array name' points to 'a', 'Row Subscript (or index)' points to the first bracketed value, and 'Column subscript (or index)' points to the second bracketed value.

## Multidimensional Array

- Multidimensional arrays are declared like this:

```
int twoDim[][] = new int[4][];
twoDim[0] = new int[5];
twoDim[1] = new int[5];
twoDim[2] = new int[5];
twoDim[3] = new int[5];
```

- This is illegal:

```
int twoDim[][] = new int[][4];
```

- This is ok:

```
int twoDim[][] = new int[4][5];
```

- Two dimensional arrays don't have to be rectangular.

## Other Notes on Array

- Can't resize arrays, but can use the same reference variable to refer to a new array.
- To copy arrays, use `System.arraycopy()` like this:

```
int elements[] = new int[] {1,2,3};
int hold[] = new int[] {1,2,3,4,5,6};
System.arraycopy(elements, 0, hold, 3,
elements.length);
```