COMP 110
More arrays

Darrell Bethea
June 2, 2011
Announcements

- Program 4 due next Friday
  - Last day of class

- Final exam
  - 6/13, 8–11 AM
  - SN014
Questions?
Today in COMP 110

- More about arrays
Review: Arrays

- An array is a collection of items of the same type
- Like a list of variables, but with a nice, compact way to name them
- A special kind of object in Java
Review: Creating an array

```java
int[] scores = new int[5];
```

- This is like declaring 5 strangely named variables of type `int`:
  - `scores[0]`
  - `scores[1]`
  - `scores[2]`
  - `scores[3]`
  - `scores[4]`
Variables such as scores[0] and scores[1] that have an integer expression in square brackets are known as:
- indexed variables, subscripted variables, array elements, or simply elements

An index or subscript is an integer expression inside the square brackets that indicates an array element.
Where have we seen the word index before?
- String’s indexOf method

Index numbers start with 0. They do NOT start with 1 or any other number.
The number inside square brackets can be any integer expression
- An integer: \[\text{scores}[3]\]
- Variable of type int: \[\text{scores}[\text{index}]\]
- Expression that evaluates to int: \[\text{scores}[\text{index} \times 3]\]

Can use these strangely named variables just like any other variables:
- \[\text{scores}[3] = 68;\]
- \[\text{scores}[4] = \text{scores}[4] + 3; \quad // \text{just made a 3-pointer!}\]
- \[\text{System.out.println(}\text{scores}[1]);\]
Review: Array

- The array itself is referred to by the name scores (in this particular case)
The array itself is referred to by the name `scores` (in this particular case).

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>68</td>
<td>73</td>
<td>57</td>
<td>102</td>
<td>94</td>
</tr>
</tbody>
</table>

the array scores
The array itself is referred to by the name scores (in this particular case)

Indices

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>68</td>
<td>73</td>
<td>57</td>
<td>102</td>
<td>94</td>
</tr>
</tbody>
</table>

the array scores
Review: Array

- The array itself is referred to by the name scores (in this particular case)

```
Indices

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>68</td>
<td>73</td>
<td>57</td>
<td>102</td>
<td>94</td>
</tr>
</tbody>
</table>
```

the array scores

scores[3]
System.out.println("Enter 5 basketball scores:");

int[] scores = new int[5];
int scoreSum = 0;
for (int i = 0; i < scores.length; i++)
{
    scores[i] = keyboard.nextInt();
    scoreSum += scores[i];
}
double average = (double) scoreSum / scores.length;
System.out.println("Average score: " + average);

for (int i = 0; i < scores.length; i++)
{
    if (scores[i] > average)
        System.out.println(scores[i] + ": above average");
    else if (scores[i] < average)
        System.out.println(scores[i] + ": below average");
    else
        System.out.println(scores[i] + ": equal to the average");
}
You can also use another form of the `for` loop with collections (such as arrays)

```java
for (int s : scores)
{
    if (s > average)
        System.out.println(s + " : above average");
    else if (s < average)
        System.out.println(s + " : below average");
    else
        System.out.println(s + " : equal to the average");
}
```

`s` takes on the value of each element of the array `score`, but you cannot change an element’s value this way.
Arrays as instance variables

```java
public class Weather {
    private double[] temperature;
    private double[] pressure;

    public void initializeTemperature(int len) {
        temperature = new double[len];
    }
}
```
Arrays of objects

Smiley[] smilies = new Smiley[3];
Arrays as arguments

public void changeArray(int[] arr)
{
    int len = arr.length;
    arr[len - 1] = 25;
}

23  47  52  14  7
Arrays as arguments

```java
public void changeArray(int[] arr)
{
    int len = arr.length;
    arr[len - 1] = 25;
}
```

23  47  52  14  25
Arrays as return types

```java
public double[] buildArray(int len) {
    double[] retArray = new double[len];
    for (int i = 0; i < retArray.length; i++) {
        retArray[i] = i * 1.5;
    }
    return retArray;
}
```
Introduction to sorting

Given an array of numbers, sort the numbers into ascending order

- Input array:
  
  | 4 | 7 | 3 | 9 | 6 | 2 | 8 |
  
- Sorted array:

  | 2 | 3 | 4 | 6 | 7 | 8 | 9 |
Selection sort pseudocode

for (index = 0; index < length; index++)
{
    Find index of smallest value of array between index and end of array
    Swap values of current index and the index with the smallest value
}
Selection sort

| 4 | 7 | 3 | 9 | 6 | 2 | 8 |
Selection sort

4  7  3  9  6  2  8
Selection sort

4 7 3 9 6 2 8
Selection sort

\[
\begin{array}{cccccc}
4 & 7 & 3 & 9 & 6 & 2 \\
\end{array}
\]

\[
\begin{array}{cccccc}
2 & 7 & 3 & 9 & 6 & 4 \\
\end{array}
\]
Selection sort

<table>
<thead>
<tr>
<th>4</th>
<th>7</th>
<th>3</th>
<th>9</th>
<th>6</th>
<th>2</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td>3</td>
<td>9</td>
<td>6</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>
## Selection sort

<table>
<thead>
<tr>
<th>4</th>
<th>7</th>
<th>3</th>
<th>9</th>
<th>6</th>
<th>2</th>
<th>8</th>
</tr>
</thead>
</table>

| 2 | 7 | 3 | 9 | 6 | 4 | 8 |

The selection sort algorithm works by repeatedly selecting the smallest element from the unsorted part of the list and moving it to the beginning of the list.
Selection sort

\[
\begin{array}{ccccccc}
4 & 7 & 3 & 9 & 6 & \boxed{2} & 8 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
2 & 7 & \boxed{3} & 9 & 6 & 4 & 8 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
2 & 3 & 7 & 9 & 6 & 4 & 8 \\
\end{array}
\]
# Selection sort

<table>
<thead>
<tr>
<th>4</th>
<th>7</th>
<th>3</th>
<th>9</th>
<th>6</th>
<th>2</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td>3</td>
<td>9</td>
<td>6</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>6</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>
Selection sort

```
4 7 3 9 6 2 8
```

```
2 7 3 9 6 4 8
```

```
2 3 7 9 6 4 8
```
Bubble sort

4 7 5 9 2 3 8
Bubble sort

4 7 5 9 2 3 8
Bubble sort
Bubble sort

4  5  7  9  2  3  8
Bubble sort

4 5 7 9 2 3 8
Bubble sort

4 5 7 9 2 3 8
Bubble sort

4 5 7 2 9 3 8
Bubble sort

4  5  7  2  9  3  8
Bubble sort

4 5 7 2 3 9 8
Bubble sort
Bubble sort

4 5 7 2 3 8 9
Bubble sort

- Repeat until array is sorted

4 5 7 2 3 8 9
Next

- Lab 8 – using arrays and writing a bubble sort algorithm
- Last lab of the semester!