Arrays 1/4
Outline

- Introduction
- Creating and Accessing Arrays
- Declaring and Creating an Array
- Referring to Arrays and Array Elements
- Three Ways to Use Square Brackets [] with an Array Name
- The length Instance Variable
- Initializing Arrays
- Introduction to Arrays

- An array is a data structure used to process a collection of data that is all of the same type
  - An array behaves like a numbered list of variables with a uniform naming mechanism
  - It has a part that does not change: the name of the array
  - It has a part that can change: an integer in square brackets
  - For example, given five scores:

    score[0], score[1], score[2], score[3], score[4]
An array that behaves like this collection of variables, all of type \texttt{double}, can be created using one statement as follows:

\begin{verbatim}
    double[] score = new double[5];
\end{verbatim}

Or using two statements:

\begin{verbatim}
    double[] score;
    score = new double[5];
\end{verbatim}

- The first statement declares the variable \texttt{score} to be of the array type \texttt{double[]}.
- The second statement creates an array with five numbered variables of type \texttt{double} and makes the variable \texttt{score} a name for the array.
The individual variables that together make up the array are called *indexed variables*. They can also be called *subscripted variables* or *elements of the array*. The number in square brackets is called an *index* or *subscript*. In Java, *indices must be numbered starting with 0, and nothing else*.

score[0], score[1], score[2], score[3], score[4]
The number of indexed variables in an array is called the *length* or *size* of the array.

When an array is created, the length of the array is given in square brackets after the array type.

The indexed variables are then numbered starting with 0, and ending with the integer that is *one less than the length of the array*.

```
score[0], score[1], score[2], score[3], score[4]
```
... - Creating and Accessing Arrays

```java
double[] score = new double[5];
```

- A variable may be used in place of the integer (i.e., in place of the integer 5 above)
  - The value of this variable can then be read from the keyboard
  - This enables the size of the array to be determined when the program is run

```java
double[] score = new double[count];
```

- An array can have indexed variables of any type, including any class type
- All of the indexed variables in a single array must be of the same type, called the *base type* of the array
Declaring and Creating an Array

An array is declared and created in almost the same way that objects are declared and created:

```java
BaseType[] ArrayName = new BaseType[size];
```

- The `size` may be given as an expression that evaluates to a nonnegative integer, for example, an `int` variable

```java
char[] line = new char[80];
double[] reading = new double[count];
Person[] specimen = new Person[100];
```
- Referring to Arrays and Array Elements

- Each array element can be used just like any other single variable by referring to it using an indexed expression: \texttt{score[0]}

- The array itself (i.e., the entire collection of indexed variables) can be referred to using the array name (without any square brackets): \texttt{score}

- An array index can be computed when a program is run
  - It may be represented by a variable: \texttt{score[index]}
  - It may be represented by an expression that evaluates to a suitable integer: \texttt{score[next + 1]}
The **for** loop is ideally suited for performing array manipulations:

```java
for (index = 0; index < 5; index++)
    System.out.println(score[index] + 
        " differs from max by " + 
        (max - score[index]) );
```

---

**Using the `score` Array in a Program**

<table>
<thead>
<tr>
<th>Indices</th>
<th>The array <code>score</code></th>
<th><code>score[3]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>80</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>99.9</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>85.5</td>
<td></td>
</tr>
</tbody>
</table>
Three Ways to Use Square Brackets [] with an Array Name

- Square brackets can be used to create a type name:

  ```java
  double[] score;
  ```

- Square brackets can be used with an integer value as part of the special syntax Java uses to create a new array:

  ```java
  score = new double[5];
  ```

- Square brackets can be used to name an indexed variable of an array:

  ```java
  max = score[0];
  ```
- The `length` Instance Variable

- An array is considered to be an object
- Since other objects can have instance variables, so can arrays
- Every array has exactly one instance variable named `length`
  - When an array is created, the instance variable `length` is automatically set equal to its size
  - The value of `length` cannot be changed (other than by creating an entirely new array with `new`)

```
double[] score = new double[5];
```

- Given `score` above, `score.length` has a value of 5
Pitfall: Array Index Out of Bounds

- Array indices always start with 0, and always end with the integer that is one less than the size of the array

  - The most common programming error made when using arrays is attempting to use a nonexistent array index

- When an index expression evaluates to some value other than those allowed by the array declaration, the index is said to be out of bounds

  - An out of bounds index will cause a program to terminate with a run-time error message

- Array indices get out of bounds most commonly at the first or last iteration of a loop that processes the array: Be sure to test for this!
- Initializing Arrays …

- An array can be initialized when it is declared
  - Values for the indexed variables are enclosed in braces, and separated by commas
  - The array size is automatically set to the number of values in the braces
    
    ```
    int[] age = {2, 12, 1};
    ```

- Given `age` above, `age.length` has a value of 3
Another way of initializing an array is by using a `for` loop

```java
double[] reading = new double[100];
int index;
for (index = 0; index < reading.length; index++)
    reading[index] = 42.0;
```

If the elements of an array are not initialized explicitly, they will automatically be initialized to the default value for their base type.
Pitfall: An Array of Characters Is Not a String

- An array of characters is conceptually a list of characters, and so is conceptually like a string.

- However, an array of characters is not an object of the class `String`.

```java
char[] a = {'A', 'B', 'C'};
String s = a; //Illegal!
```

- An array of characters can be converted to an object of type `String`, however.
Pitfall: An Array of Characters Is Not a String

- The class `String` has a constructor that has a single parameter of type `char[]`

  ```java
  String s = new String(a);
  ```

  The object `s` will have the same sequence of characters as the entire array `a` ("ABC"), but is an independent copy.

- Another `String` constructor uses a subrange of a character array instead

  ```java
  String s2 = new String(a,0,2);
  ```

  Given `a` as before, the new string object is "AB"
Pitfall: An Array of Characters Is Not a String

- An array of characters does have some things in common with `String` objects
  - For example, an array of characters can be output using `println`
    
    ```java
    System.out.println(a);
    ```
  - Given `a` as before, this would produce the output `ABC`
THE END