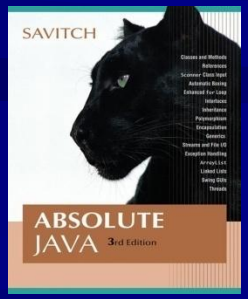
Comp 248
Introduction to Programming
Chapter 6 *Arrays*

Part A

Dr. Aiman Hanna

Department of Computer Science & Software Engineering
Concordia University, Montreal, Canada

These slides has been extracted, modified and updated from original slides of Absolute Java 3rd Edition by Savitch; which has originally been prepared by Rose Williams of Binghamton University. Absolute Java is published by Pearson Education / Addison-Wesley.

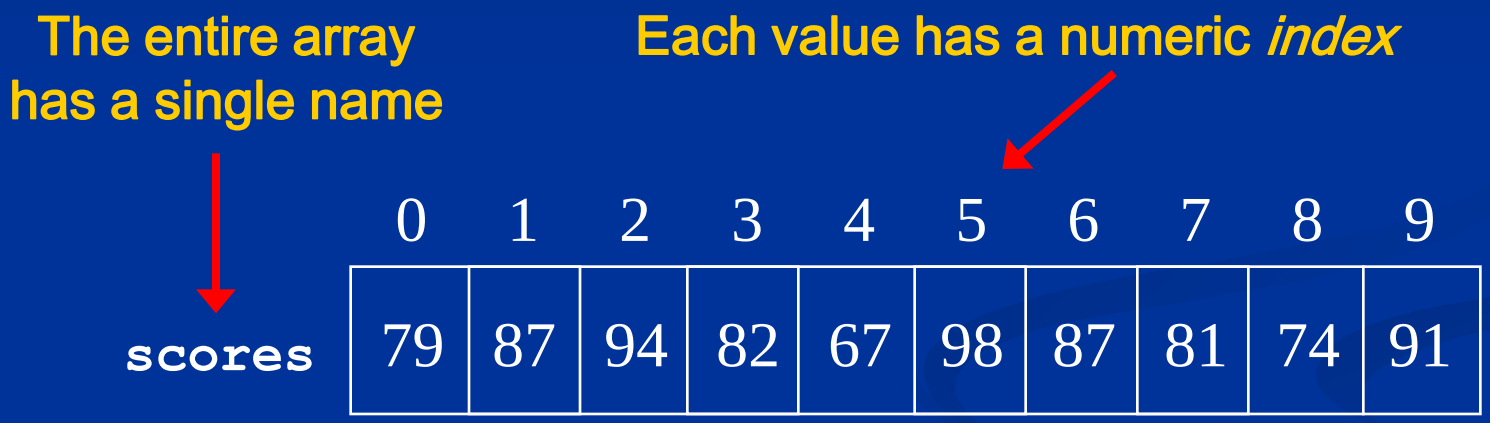


UNIVERSITÉ Concordia UNIVERSITY

PEARSON Addison Wesley

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 of size N is indexed from zero to N-1

The above array holds 10 values that are indexed from 0 to 9

Creating and Accessing Arrays

- An array is declared and created in almost the same way that objects are declared and created
- An array of **`double`** values can be created using one statement as follows:

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

- Or using two statements:

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

  - The first statement declares the variable **`score`** to be of the array type **`double[]`**
  - The second statement creates an array with five numbered variables of type **`double`** and makes the variable **`score`** a name for the array

Creating and Accessing Arrays

- ArrayOperations1.java (MS-Word file)
- ArrayOperations2.java (MS-Word file)
- ArrayOperations3.java (MS-Word file)

Creating and Accessing Arrays

```
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

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

- An array can have indexed variables of any type, including any class type

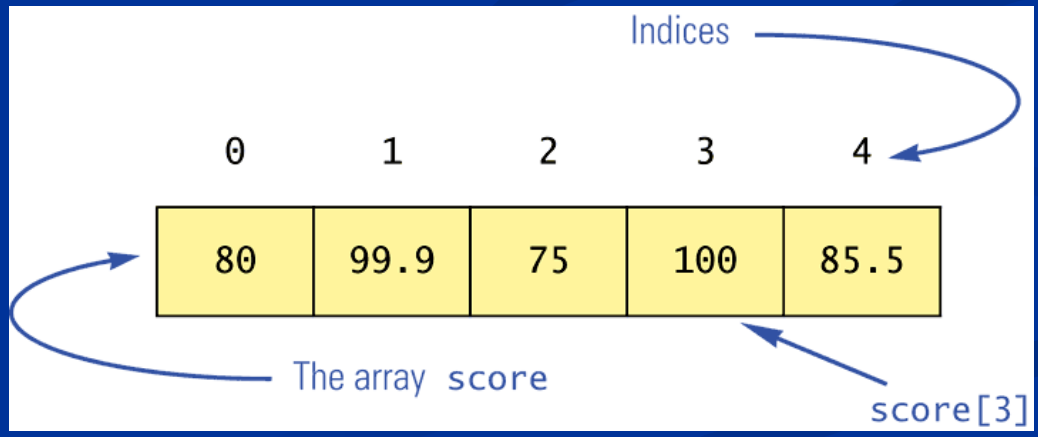
```
Car[] arr1 = new Car[10]; // array of 10 cars
// MUST BE CAREFUL HERE. DO WE REALLY HAVE 10 Cars?
```

- All of the indexed variables in a single array must be of the same type, called the *base type* of the array

Example: Using the `score` Array in a Program

- The **`for`** loop is ideally suited for performing array manipulations:

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



The `length` Instance Variable

- An array is considered to be an object
- Since other objects can have instance variables (attributes), so can arrays
- Every array has 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

- ArrayOperations4.java (MS-Word file)

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

Initializing Arrays

- Another way of initializing an array is by using a **`for`** loop

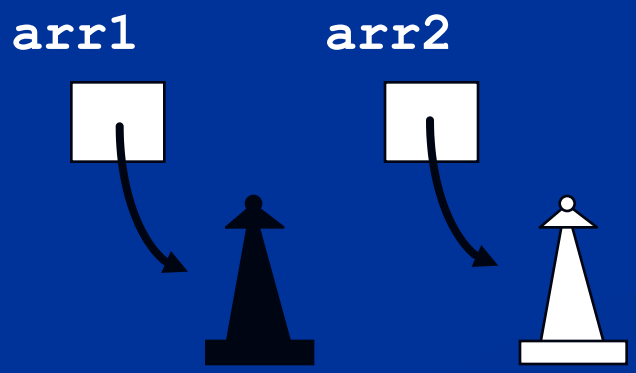
```
double[] reading = new double[100];
int index;
for (index = 0;
     index < reading.length; index++)
  reading[index] = 42.0;
```

- If the elements of an array are not initialized explicitly, they will automatically be initialized to the default value for their base type

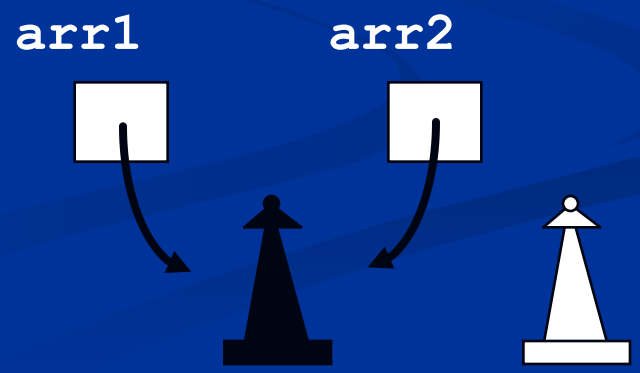
Copying Arrays

- ArrayOperations5.java (MS-Word file)
- ArrayOperations6.java (MS-Word file)
- Arrays are object. Hence, "=" cannot be used to copy one array to another
  For example, arr2 = arr1; would result in the following

Before



After



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!

Pitfall: Array Index Out of Bounds

- ArrayOperations7.java (MS-Word file)
- ArrayOperations8.java (MS-Word file)