7 Java Arrays

Objectives

At the end of the lesson, the student should be able to:

- Declare and create arrays
- Access array elements
- Determine the number of elements in an array
- Declare and create multidimensional arrays

Introduction to Arrays

- Suppose we have here three variables of type int with different identifiers for each variable.

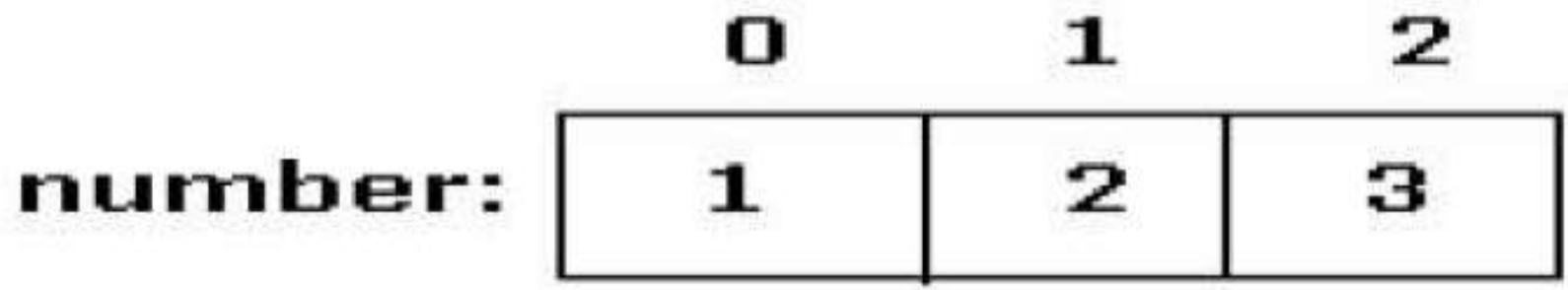
```
int number1;
int number2;
int number3;

number1 = 1;
number2 = 2;
number3 = 3;
```

As you can see, it seems like a tedious task in order to just initialize and use the variables especially if they are used for the same purpose.

Introduction to Arrays

- In Java and other programming languages, there is one capability wherein we can use one variable to store a list of data and manipulate them more efficiently. This type of variable is called an array.

- An array stores multiple data items of the same data type, in a contiguous block of memory, divided into a number of slots



Declaring Arrays

- To declare an array, write the data type, followed by a set of square brackets[], followed by the identifier name.

- For example,

```
int []ages;
```

or

```
int ages[];
```

Array Instantiation

- After declaring, we must create the array and specify its length with a constructor statement.

- Definitions:
  - Instantiation
    - In Java, this means creation
  - Constructor
    - In order to instantiate an object, we need to use a constructor for this. A constructor is a method that is called to create a certain object.
    - We will cover more about instantiating objects and constructors later.

Array Instantiation

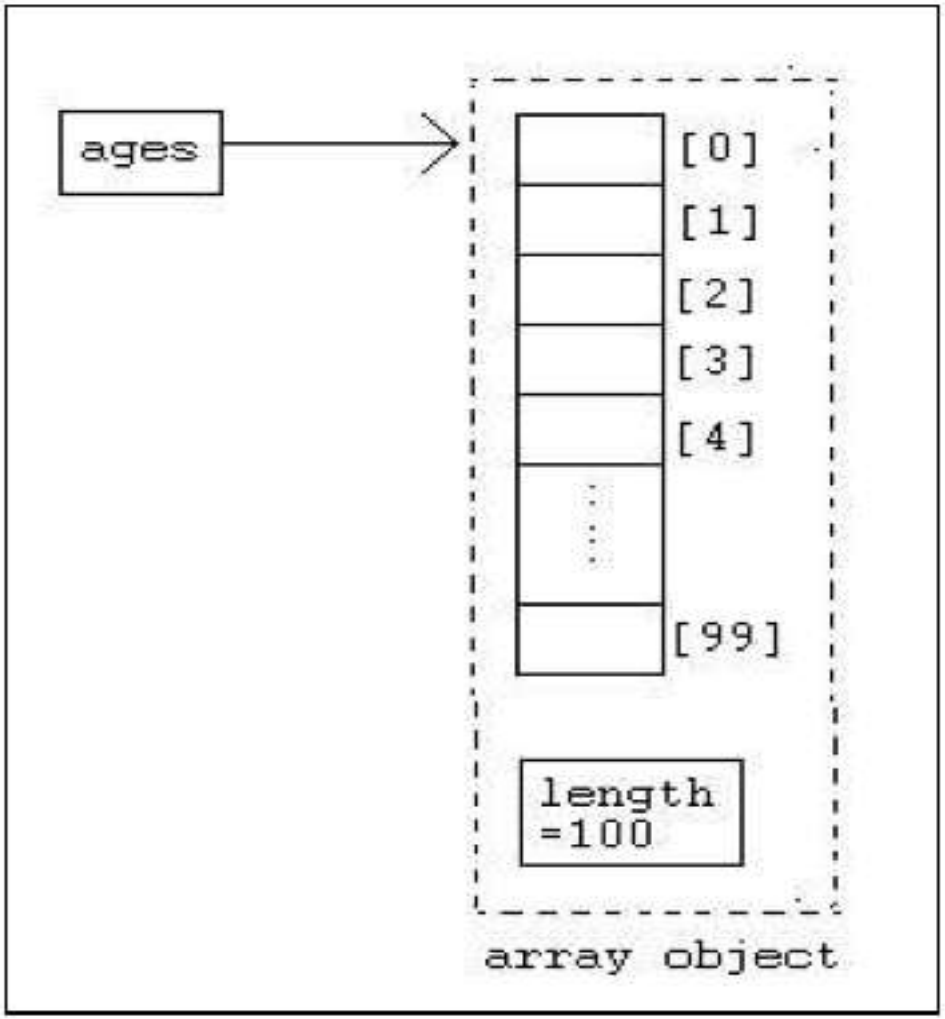
- To instantiate (or create) an array, write the `new` keyword, followed by the square brackets containing the number of elements you want the array to have.
- For example,

```
//declaration
 int ages[];

//instantiate object
ages = new int[100];
```

or, can also be written as,

```
//declare and instantiate object
int ages[] = new int[100];
```

Array Instantiation



Array Instantiation

- You can also instantiate an array by directly initializing it with data.

- For example,

```
int arr[] = {1, 2, 3, 4, 5};
```

  This statement declares and instantiates an array of integers with five elements (initialized to the values 1, 2, 3, 4, and 5).

Sample Program

```
1   //creates an array of boolean variables with identifier
2   //results. This array contains 4 elements that are
3   //initialized to values {true, false, true, false}
4
5   boolean results[] = { true, false, true, false };
6
7   //creates an array of 4 double variables initialized
8   //to the values  {100, 90, 80, 75};
9
10  double []grades = {100, 90, 80, 75};
11
12  //creates an array of Strings with identifier days and
13  //initialized. This array contains 7 elements
14
15  String days[] = { “Mon”, “Tue”, “Wed”, “Thu”, “Fri”, “Sat”,
    “Sun”};
```

Accessing an Array Element

- To access an array element, or a part of the array, you use a number called an index or a subscript.

- index number or subscript
  - assigned to each member of the array, to allow the program to access an individual member of the array.
  - begins with zero and progress sequentially by whole numbers to the end of the array.
  - NOTE: Elements inside your array are from 0 to (sizeOfArray-1).

Accessing an Array Element

- For example, given the array we declared a while ago, we have

```
//assigns 10 to the first element in the array
ages[0] = 10;

//prints the last element in the array
System.out.print(ages[99]);
```

Accessing an Array Element

- NOTE:
  - once an array is declared and constructed, the stored value of each member of the array will be initialized to zero for number data.
  - for reference data types such as Strings, they are NOT initialized to blanks or an empty string “”. Therefore, you must populate the String arrays explicitly.

Accessing an Array Element

- The following is a sample code on how to print all the elements in the array. This uses a for loop, so our code is shorter.

```
1  public class ArraySample{
2     public static void main( String[] args ){
3        int[] ages = new int[100];
4        for( int i=0; i<100; i++ ){
5           System.out.print( ages[i] );
6        }
7     }
8  }
```

Coding Guidelines

1. It is usually better to initialize or instantiate the array right away after you declare it. For example, the declaration,

```
int []arr = new int[100];
```

is preferred over,

```
int []arr;
arr = new int[100];
```

Coding Guidelines

2. The elements of an n-element array have indexes from 0 to n-1. Note that there is no array element arr[n]! This will result in an array-index-out-of-bounds exception.

3. Remember: You cannot resize an array.

Array Length

- In order to get the number of elements in an array, you can use the length field of an array.

- The length field of an array returns the size of the array. It can be used by writing,

```
arrayName.length
```

Array Length

```
1  public class ArraySample {
2     public static void main( String[] args ){
3        int[] ages = new int[100];
4
5        for( int i=0; i<ages.length; i++ ){
6           System.out.print( ages[i] );
7        }
8     }
9  }
```

Coding Guidelines

1. When creating for loops to process the elements of an array, use the array object's length field in the condition statement of the for loop. This will allow the loop to adjust automatically for different-sized arrays.

2. Declare the sizes of arrays in a Java program using named constants to make them easy to change. For example,

```
final int ARRAY_SIZE = 1000; //declare a constant
. . .
int[] ages = new int[ARRAY_SIZE];
```

Multidimensional Arrays

- Multidimensional arrays are implemented as arrays of arrays.
- Multidimensional arrays are declared by appending the appropriate number of bracket pairs after the array name.

Multidimensional Arrays

- For example,

```
// integer array 512 x 128 elements
int[][] twoD = new int[512][128];

// character array 8 x 16 x 24
char[][][] threeD = new char[8][16][24];

// String array 4 rows x 2 columns
String[][] dogs = {{ "terry", "brown" },
                   { "Kristin", "white" },
                   { "toby", "gray"},
                   { "fido", "black"}
                  };
```

Multidimensional Arrays

- To access an element in a multidimensional array is just the same as accessing the elements in a one dimensional array.

- For example, to access the first element in the first row of the array dogs, we write,

```
System.out.print( dogs[0][0] );
```

This will print the String "terry" on the screen.

Summary

- Arrays
  - Definition
  - Declaration
  - Instantiation and constructors (brief overview – to be discussed more later)
  - Accessing an element
  - The length field
  - Multidimensional Arrays