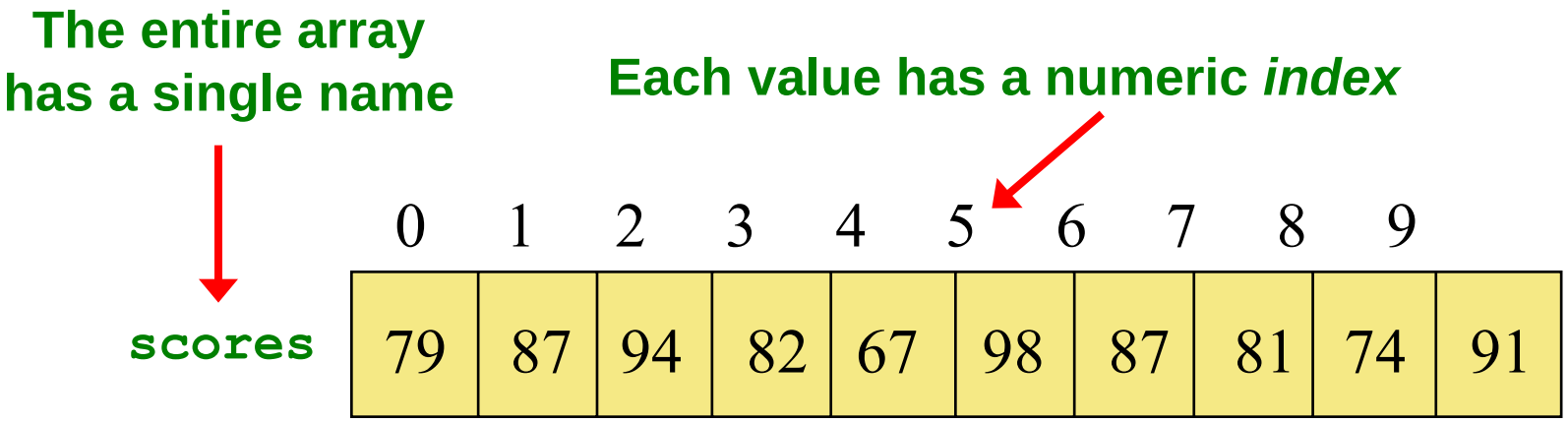
Principles of Computer Science

Java Arrays

Arrays

- An *array* is an ordered list of values



An array of size N is indexed from zero to N-1

This array holds 10 values that are indexed from 0 to 9

Arrays

- A particular value in an array is referenced using the array name followed by the index in brackets
- For example, the expression

```
scores[2]
```

  refers to the value 94 (the 3rd value in the array)

- That expression represents a place to store a single integer and can be used wherever an integer variable can be used

Arrays

- For example, an array element can be assigned a value, printed, or used in a calculation

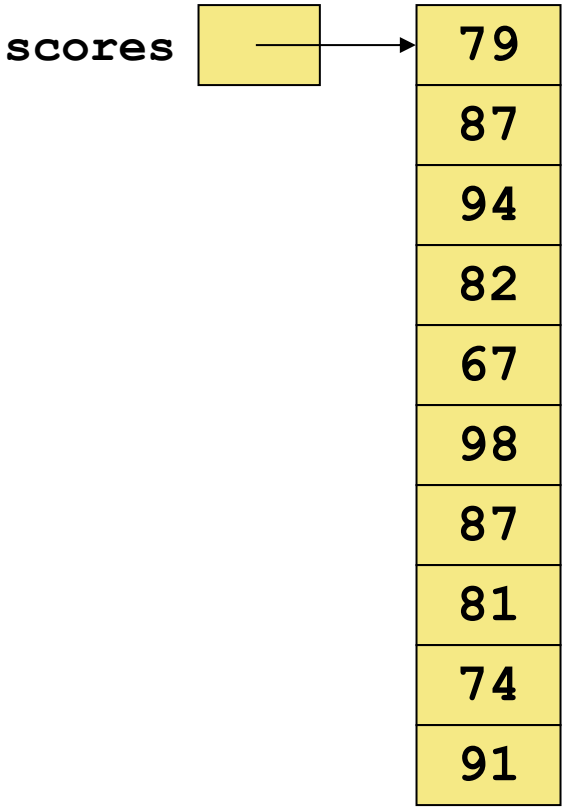
```
scores[2] = 89;

scores[first] = scores[first] + 2;

mean = (scores[0] + scores[1])/2;

System.out.println ("Top = " + scores[5]);
```

Arrays

- The values held in an array are called *array elements*
- An array stores multiple values of the same type – the *element type*
- The element type can be a primitive type or an object reference
- Therefore, we can create an array of integers, an array of characters, an array of String objects, an array of Coin objects, etc.
- In Java, the array itself is an object that must be instantiated

Arrays

- Another way to depict the scores array



Declaring Arrays

- The scores array could be declared as follows

```
int[] scores = new int[10];
```

- The type of the variable scores is int[] (an array of integers)
- Note that the array type does not specify its size, but each object of that type has a specific size
- The reference variable scores is set to a new array object that can hold 10 integers

Declaring Arrays

- Some other examples of array declarations

```
float[] prices = new float[500];

boolean[] flags;

flags = new boolean[20];

char[] codes = new char[1750];
```

Using Arrays

- The iterator version of the for loop can be used when processing array elements

```
for (int score : scores)
   System.out.println (score);
```

- This is only appropriate when processing all array elements from top (lowest indext) to bottom (highest index).

```
//********************************************************************
//  BasicArray.java
//  Demonstrates basic array declaration and use.
//********************************************************************

public class BasicArray
{
  //-----------------------------------------------------------
  //  Creates an array, fills it with various integer values,
  //  modifies one value, then prints them out.
  //-----------------------------------------------------------
  public static void main (String[] args)
  {
    final int LIMIT = 15, MULTIPLE = 10;

    int[] list = new int[LIMIT];

    //  Initialize the array values
    for (int index = 0; index < LIMIT; index++)
      list[index] = index * MULTIPLE;
                list[5] = 999;  // change one array value

    //  Print the array values
    for (int value : list)
      System.out.print (value + "  ");
  }
}
```

Output

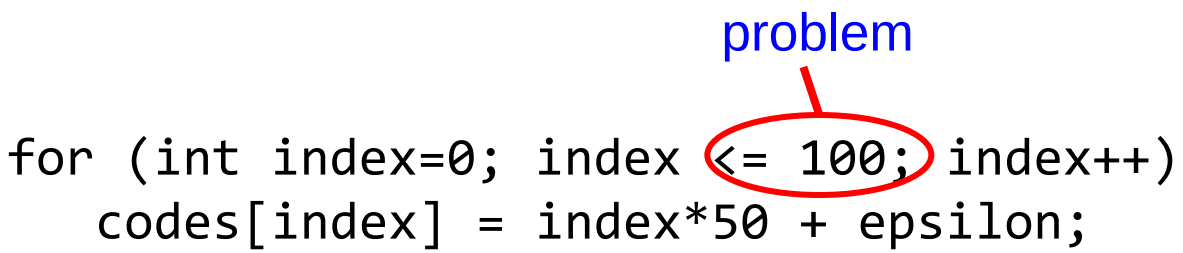
```
0  10  20  30  40  999 60 70 80 90 100 110 120 130 140
```

Bounds Checking

- For example, if the array codes can hold 100 values, it can be indexed using only the numbers 0 to 99
- If the value of count is 100, then the following reference will cause an exception to be thrown

```
System.out.println (codes[count]);
```

- It's common to introduce *off-by-one errors* when using arrays



Bounds Checking

- Each array object has a public constant called length that stores the size of the array
- It is referenced using the array name

```
scores.length
```

- Note that length holds the number of elements, not the largest index

Alternate Array Syntax

- The brackets of the array type can be associated with the element type or with the name of the array
- Therefore the following two declarations are equivalent

```
float[] prices;
float prices[];
```

- The first format generally is more readable and should be used

Initializer Lists

- An *initializer list* can be used to instantiate and fill an array in one step
- The values are delimited by braces and separated by commas
- Examples

```
int[] units = {147, 323, 89, 933, 540,
               269, 97, 114, 298, 476};

char[] letterGrades = {'A', 'B', 'C', 'D', ’F'};
```

Initializer Lists

- Note that when an initializer list is used
  - the new operator is not used
  - no size value is specified
- The size of the array is determined by the number of items in the initializer list
- An initializer list can be used only in the array declaration

Arrays as Parameters

- An entire array can be passed as a parameter to a method
- Like any other object, the reference to the array is passed, making the formal and actual parameters aliases of each other
- Therefore, changing an array element within the method changes the original
- An individual array element can be passed to a method as well, in which case the type of the formal parameter is the same as the element type

Arrays of Objects

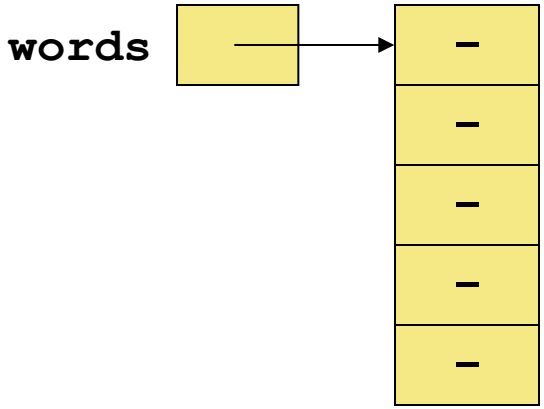
- The elements of an array can be object references
- The following declaration reserves space to store 5 references to String objects

```
String[] words = new String[5];
```

- It does not create the String objects themselves
- Initially an array of objects holds null references
- Each object stored in an array must be instantiated separately

Arrays of Objects

- The words array when initially declared

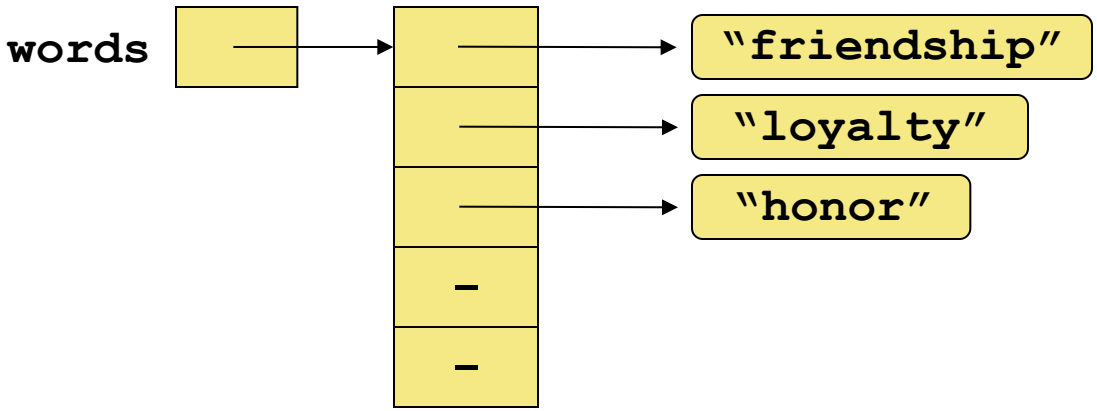


- At this point, the following reference would throw a NullPointerException

```
System.out.println (words[0]);
```

Arrays of Objects

- After some String objects are created and stored in the array



Arrays of Objects

- Keep in mind that String objects can be created using literals
- The following declaration creates an array object called verbs and fills it with four String objects created using string literals

```
String[] verbs = {"play", "work", "eat", "sleep"};
```

- The following example creates an array of Grade objects, each with a string representation and a numeric lower bound

Command-Line Arguments

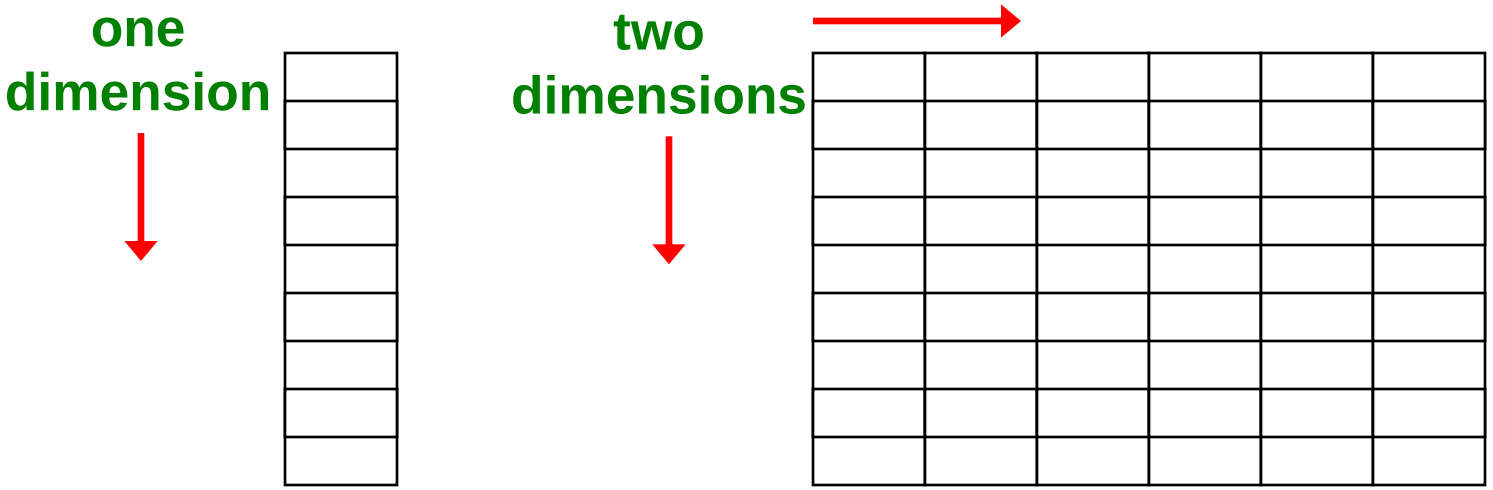
- The signature of the main method indicates that it takes an array of String objects as a parameter
- These values come from *command-line arguments* that are provided when the interpreter is invoked
- For example, the following invocation of the interpreter passes three String objects into main

```
> java StateEval pennsylvania texas arizona
```

- These strings are stored at indexes 0-2 of the array parameter of the main method

Two-Dimensional Arrays

- A *one-dimensional array* stores a list of elements
- A *two-dimensional array* can be thought of as a table of elements, with rows and columns



Two-Dimensional Arrays

- To be precise, in Java a two-dimensional array is an array of arrays
- A two-dimensional array is declared by specifying the size of each dimension separately

```
int[][] scores = new int[12][50];
```

- A array element is referenced using two index values

```
value = scores[3][6]
```

- The array stored in one row can be specified using one index

Two-Dimensional Arrays

Expression	Type	Description
`table`	`int[][]`	2D array of integers, or array of integer arrays
`table[5]`	`int[]`	array of integers
`table[5][12]`	`int`	integer

Multidimensional Arrays

- An array can have many dimensions – if it has more than one dimension, it is called a *multidimensional array*
- Each dimension subdivides the previous one into the specified number of elements
- Each dimension has its own length constant
- Because each dimension is an array of array references, the arrays within one dimension can be of different lengths
  - these are sometimes called *ragged arrays*

Today's Topics

Today's lecture focuses on

- array declaration and use
- bounds checking and capacity
- arrays that store object references
- variable length parameter lists
- multidimensional arrays

READING MATERIAL

- TEXT
  - Chapter 2
    The very basics of Java
  - Chapter 3
    Classes and objects
  - Chapter 4
    Basic syntax of procedure programming using Java
  - Chapter 5
    Writing Java classes and simple design principles of writing Java classes.
  - Chapter 7
    Arrays
- Online
  - http://java.sun.com/docs/books/tutorial/java/index.html
    - Language Basics
    - Classes and Objects
    - Numbers and Strings