Array & String in Java

MODULE NAME: OBJECT ORIENTED PROGRAMMING WITH JAVA

MODULE CODE: CST 05102

INSTRUCTORS: JAMILA L.

Array definition

An array is a group of memory locations related by the fact that they all have the same name and the same type. **Java array** is an object which contains elements of a similar data type.

An array is a normal java variable like any other, but it works like a container; you can put other variables inside it.

- Arrays are **objects**, so they are considered **reference types**.
- Each variable inside an array is called an **element** .
- We can store primitive values or objects in an array in Java.
- Like C/C++, we can also create single dimensional or multidimensional arrays in Java.

*** Simply, Arrays are **data structures consisting of related data items of the same type**.

Array definition cont…

The array definition e.g.

- **int n[5] = { 32, 27, 64, 18, 95, 14 };**

This causes a syntax error because there are **six initializers** and only **five array elements**.

If the array size is omitted from a definition with an initializer list, the number of elements in the array will be the number of elements in the initializer list.

For example,

- **int n[] = { 1, 2, 3, 4, 5 };**

would create a five-element array.

Element of array

- The first element in every array is the zeroth element.
- The second element of array *arrayName* is referred to as *arrayName*[1],
- The seventh element of array *arrayName* is referred to as *arrayName*[6],
- In general, the *n*th element of array *arrayName* is referred to as *arrayName*[1 - n].
- Array names, like other variable names, can contain only letters, digits and underscores.
- Array names cannot begin with a digit.
- The position number contained within square brackets is more formally called a subscript (or index).
- A subscript must be an integer or an integer expression.

Setting up an array

- To set up an array with 5 names in it do:

```
char names[0] = 'John';
char names[1] = 'Paul';
char names[2] = 'Steven';
char names[3] = 'George';
char names[4] = 'David';
```

- To add a value to an array you must specify the location in the array by putting a number in [ ].

To tell the computer to reserve 12 elements for integer array c, use the definition

- **int c[12];**

Common Programming Error 6.1

It's important to note the difference between the "seventh element of the array" and "array element seven." Because array subscripts begin at 0, the "seventh element of the array" has a subscript of 6, while "array element seven" has a subscript of 7 and is actually the eighth element of the array. This is a source of "off-by-one" errors.

Types of arrays

1. Single (One) dimensional array
2. Multidimensional array

Declaring and Creating Single dimensional Arrays

Like other objects, arrays are created with keyword **new**.

To create an array object, the programmer specifies the type of the array elements and the number of elements as part of an array-creation expression that uses keyword new.

Such an expression returns a reference that can be stored in an array variable.

The following declaration and array-creation expression create an array object containing 12 integers elements and store the array's reference in variable c:

```
int c[] = new int[ 12 ]
```

Example

```
public class ArrayExample {

   public static void main(String[] args) {

      int array[]; // declare array named array
      array = new int[ 10 ]; // create the space for array
      System.out.printf( "%s%8s\n", "Index", "Value" );
      // output each array element's value
      for ( int counter = 0; counter < array.length; counter++ )
         System.out.printf( " %2d  %6d\n", counter, array[ counter ]
        );
   }

}
```

The Output

Index	Value
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

Example

```
// Initializing the elements of an array with an array initializer.

public class InitArray {
  public static void main( String args[] ) {
   // initializer list specifies the value for each element
     int array[] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };

    System.out.printf( "%s%8s\n", "Index", "Value" ); // column headings

  // output each array element's value
    for ( int counter = 0; counter < array.length; counter++ )
          System.out.printf( "%5d%8d\n", counter, array[ counter ] );
  } // end main
} // end class InitArray
```

The Output

Index	Value
0	32
1	27
2	64
3	18
4	95
5	14
6	90
7	70
8	60
9	37

Array with Constant variable

The following program uses the modifier **final** to declare the constant variable ARRAY_LENGTH with the value 10.

***Constant variables (also known as final variables) must be initialized before they are used and cannot be modified thereafter.*

If you attempt to modify a final variable after it is initialized in its declaration, the compiler issues the error message cannot assign a value to final variable variableName.

Array with Constant variable example

```
public class ArrayConst {

     public static void main(String[] args) {

          final int ARRAY_LENGTH = 10; // declare constant

          int array[] = new int[ ARRAY_LENGTH ]; // create array
          // calculate value for each array element
           for ( int counter = 0; counter < array.length; counter++ )
                   array[ counter ] = 2 + 2 * counter;

                   System.out.printf( "%s%8s\n", "Index", "Value" ); // column headings
                   // output each array element's value
           for ( int counter = 0; counter < array.length; counter++ )
           System.out.printf( "%2d%6d\n", counter, array[ counter ] );
     }
}
```

Index	Value
0	2
1	4
2	6
3	8
4	10
5	12
6	14
7	16
8	18
9	20

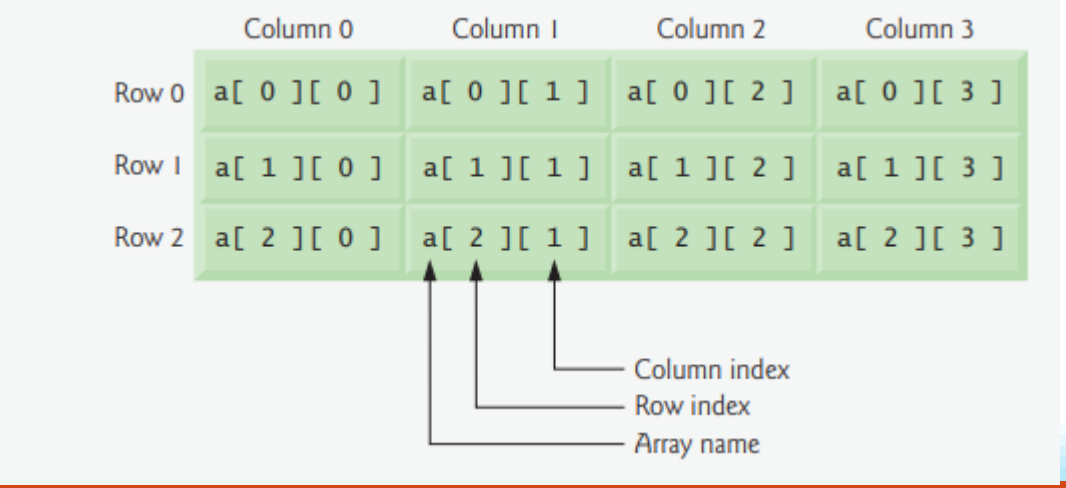
Two dimensional array

Multidimensional arrays with two dimensions are often used to represent tables of values consisting of information arranged in rows and columns. (Multidimensional arrays can have more than two dimensions.)

Like one-dimensional arrays, multidimensional arrays can be initialized with array initializers in declarations.

A two-dimensional array b with two rows and two columns could be declared and initialized with nested array initializers as follows:

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



Two dimensional array cont…

The manner in which multidimensional arrays are represented makes them quite flexible. In fact, the lengths of the rows in array b are not required to be the same. For example,

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

Example:

```
public class TwoDimensionalArray {
public static void main(String[] args) {
// Initializing two-dimensional arrays.
int array1[][] = { { 1, 2, 3 }, { 4, 5, 6 } };
int array2[][] = { { 1, 2 }, { 3 }, { 4, 5, 6 } };
System.out.println( "Values in array1 by row are" );
outputArray( array1 ); // displays array1 by row
System.out.println( "\nValues in array2 by row are" );
outputArray( array2 ); // displays array2 by row
}
```

Example continue:

```
private static void outputArray(int[][] array) {
// loop through array's rows
for ( int row = 0; row < array.length; row++ )
{
 // loop through columns of current row
for ( int column = 0; column < array[ row ].length; column++ )
 System.out.printf( "%d ", array[ row ][ column ] );

 System.out.println(); // start new line of output
} // end outer f

}
}
```

```
Values in array1 by row are
1  2  3
4  5  6

Values in array2 by row are
1  2
3
4  5  6
```

String Variable

Introduction

A string is a sequence of characters, e.g., "Hello".

- In Java strings are enclosed in quotation marks, which are not themselves part of the string.
- Next to numbers, strings are the most important data type that most programs use
- You can declare variables that hold strings, e.g. **String name = "John";**
- Use assignment to place a different string into the variable e.g., **name = "Java";**

String data type

- The number of characters in a string is called the length of the string
- E.g., the length of "Hello, World!" is 13.
- Compute the length of the string with the length method.
- int n = name.length();

Unlike numbers, strings are objects.

- See that **String** is a class because it starts with an uppercase letter
- The basic data types **int** and **double** start with a lowercase letter
- Thus, can call methods on strings, e.g., **name.length()**

Substrings

- Can extract substrings, and can glue smaller strings together to form a larger ones
- To extract substring, use the **substring** method: **s.substring(start, pastEnd)**
- Returns a string that is made up from the characters in the string **s**, starting at character **start**, and containing all characters up to, but not including, the character **pastEnd**.

Substrings cont…

- E.g.,
  - **String greeting = "Hello, World!";**
  - **String sub = greeting.substring(0,4); //sub is "Hell"**

H	e	l	l	o	,		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11	12

Concatenation

Given two strings, such as "Snuffer" and "Dog-Dog", you can concatenate them to one long string:

- String fname = "Snuffer";
- String lname = "Dog-Dog";
- String name = fname + lname;

The + operator concatenates two strings

- The resulting string is "SnufferDog-Dog"
- We would like the first and last name separated by a space.
  - String name = fname + " " + lname;

Concatenation cont…

- If one of the expressions, either to the left or the right of a + operator, is a string, then the other one is automatically forced to become a string as well, and both strings are concatenated.
- The final result of concatenation is a string

E.g.,

```
String a = "Agent";
int n = 7;
String bond = a + n;
```

Since a is a string, n is converted from the integer 7 to the string "7".

The two strings "Agent" and "7" are concatenated to form the string "Agent7"

Concatenation cont…

- Can reduce the number of System.out.print instructions
- E.g. System.out.print(“The total is”); System.out.println(total);
- To the single call
- System.out.println("The total is " + total);

String methods

- The toUpperCase and toLowerCase methods make strings with only upper or lowercase characters.
  - String greeting = “Hello”;
  - System.out.println(greeting.toUpperCase() + " " + greeting.toLowerCase());

  Display HELLO hello.

- The toUpperCase and toLowerCase do not change the original String object greeting.
- They return new String objects that contain the uppercased and lowercased versions of the original string.

Converting between numbers and Strings

- All command line arguments are passed to a Java application as String objects.
- This is why you must convert String values obtained at the keyboard to respective data type before using them in expressions.
- In general to convert a number to a string, you can concatenate with the empty string:

```
int age = 19;
String ageString = " " + age;
```

- Some programmers prefer to use the toString methods of the Integer and Double classes, because it is more explicit:

```
String ageString = Integer.toString(age);
```

- To convert a string containing just digits to its integer value use the static parseInt method of the Integer class.
- To convert a string containing floating-point digits to its floating-point value, use the static parseDouble method of the Double class.

Thank you!

for listen