

Object Oriented Programming

Prepared by Dr. Yasser Abdelhamid

IT223

Object Oriented Programming

Lecture 09



OUTLINE

- Arrays
- 2D arrays

ARRAYS

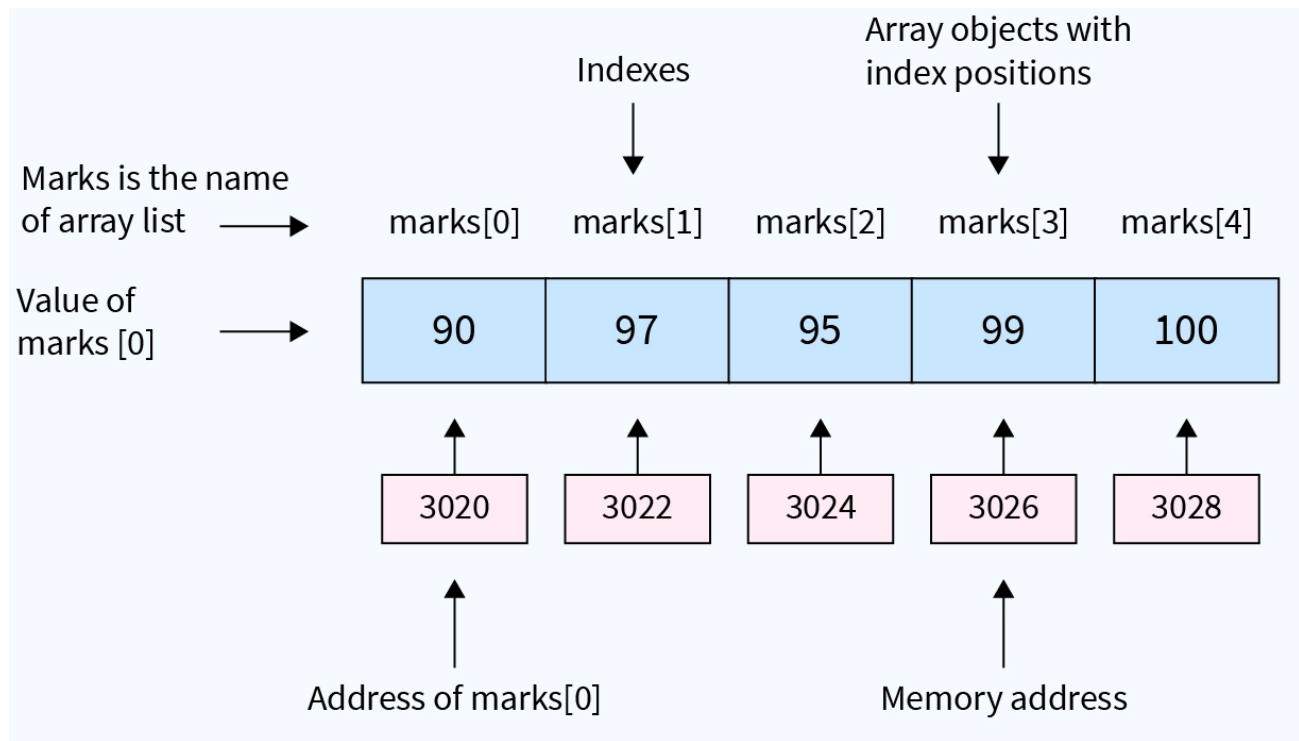
ARRAY

- ❖ An array is a **collection of elements** of **similar type** stored in **contiguous** memory locations.
- ❖ The element type can be a **primitive** data type or an **object reference**.

Single variable		1				
Array:	Indexes	0	1	2	3	4
	Values	1	3	8	23	99

EXAMPLE

❖ `int [] marks = new int[5];`



EXAMPLE

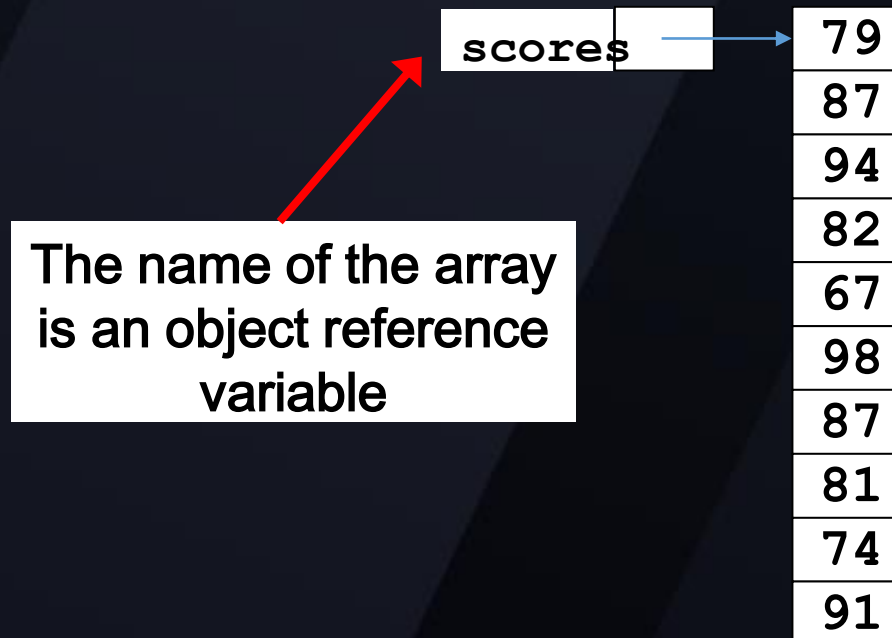
- ❖ Based on the following array, what is the value for each statement?
- ❖ a. `height[1]`
- ❖ b. `height[2] + height[5]`
- ❖ c. `height[2 + 5]`
- ❖ d. the value stored at index 8
- ❖ e. the fourth value
- ❖ f. `height.length`

height	
0	69
1	61
2	70
3	74
4	62
5	69
6	66
7	73
8	79
9	62
10	70

An array called `height` containing integer values

ARRAYS IN JAVA

- ❖ In Java, the array itself is an object that must be instantiated
- ❖ 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

ARRAY DECLARATION EXAMPLES

❖ Some other examples of array declarations:

```
int[] weights = new int[2000];
```

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

```
boolean[] flags;  
flags = new boolean[20];
```

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

USING ARRAYS

- ❖ The for-each 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 starting at index 0
- ❖ It can't be used to set the array values

```
//*****
//  BasicArray.java          Author: Lewis/Loftus
//
//  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

```
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 + " ");
    }
}
```

BASICARRAY EXAMPLE

The array is created
with 15 elements,
indexed from 0 to 14

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

After three
iterations of the
first loop

0	0
1	10
2	20
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

After completing
the first loop

0	0
1	10
2	20
3	30
4	40
5	50
6	60
7	70
8	80
9	90
10	100
11	110
12	120
13	130
14	140

After changing
the value of
list[5]

0	0
1	10
2	20
3	30
4	40
5	999
6	60
7	70
8	80
9	90
10	100
11	110
12	120
13	130
14	140

EXERCISES

Write an array declaration to represent the ages of 100 children.

Write code that prints each value in an array of integers named `values`.

ANSWER

Write an array declaration to represent the ages of 100 children.

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

Write code that prints each value in an array of integers named `values`.

```
for (int value : values)  
    System.out.println(value);
```

BOUNDS CHECKING

- ❖ Once an array is created, it has a fixed size
- ❖ An index used in an array reference must specify a valid element
- ❖ That is, the index value must be in range 0 to N-1
- ❖ The Java interpreter throws an `ArrayIndexOutOfBoundsException` if an array index is out of bounds
- ❖ This is called automatic *bounds checking*

BOUNDS CHECKING

- ❖ For example, if the array `codes` can hold 100 values, it can be indexed from 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:

```
for (int index=0; index <= 100; index++)  
    codes[index] = index*50 + epsilon;
```

problem

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
- ❖ See `ReverseOrder.java`
- ❖ See `LetterCount.java`

```
//*****
// ReverseOrder.java      Author: Lewis/Loftus
//
// Demonstrates array index processing.
//*****

import java.util.Scanner;

public class ReverseOrder
{
    //-----
    // Reads a list of numbers from the user, storing them in an
    // array, then prints them in the opposite order.
    //-----
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);

        double[] numbers = new double[10];

        System.out.println("The size of the array: " + numbers.length);

        // continue next page
    }
}
```

```
for (int index = 0; index < numbers.length; index++)
{
    System.out.print("Enter number " + (index+1) + ": ");
    numbers[index] = scan.nextDouble();
}

System.out.println("The numbers in reverse order:");

for (int index = numbers.length-1; index >= 0; index--)
    System.out.print(numbers[index] + " ");
}
```

Sample Run

The size of the array: 10

Enter number 1: 18.36

Enter number 2: 48.9

Enter number 3: 53.5

Enter number 4: 29.06

Enter number 5: 72.404

Enter number 6: 34.8

Enter number 7: 63.41

Enter number 8: 45.55

Enter number 9: 69.0

Enter number 10: 99.18

The numbers in reverse order:

99.18 69.0 45.55 63.41 34.8 72.404 29.06 53.5 48.9
18.36

```
//*****
// LetterCount.java      Author: Lewis/Loftus
//
// Demonstrates the relationship between arrays and strings.
//*****

import java.util.Scanner;

public class LetterCount
{
    //-----
    // Reads a sentence from the user and counts the number of
    // uppercase and lowercase letters contained in it.
    //-----
    public static void main(String[] args)
    {
        final int NUMCHARS = 26;

        Scanner scan = new Scanner(System.in) ;

        int[] upper = new int[NUMCHARS];
        int[] lower = new int[NUMCHARS];

        char current;    // the current character being processed
        int other = 0;   // counter for non-alphabets
    }
}
```

// continue next page

```
// Read a line
System.out.println("Enter a sentence:");
String line = scan.nextLine();

// Count the number of each letter occurrence
for (int ch = 0; ch < line.length(); ch++)
{
    current = line.charAt(ch);
    if (current >= 'A' && current <= 'Z')
        upper[current-'A']++;
    else
        if (current >= 'a' && current <= 'z')
            lower[current-'a']++;
        else
            other++;
}
```

// continue next page

```
// Print the results
System.out.println();
for (int letter=0; letter < upper.length; letter++)
{
    System.out.print( (char) (letter + 'A') );
    System.out.print(": " + upper[letter]);
    System.out.print("\t\t" + (char) (letter + 'a') );
    System.out.println(": " + lower[letter]);
}

System.out.println();
System.out.println("Non-alphabetic characters: " + other);
}
}
```

Sample Run

Enter a sentence:

In Casablanca, Humphrey Bogart never says "Play it again, Sam."

A: 0	a: 10
B: 1	b: 1
C: 1	c: 1
D: 0	d: 0
E: 0	e: 3
F: 0	f: 0
G: 0	g: 2
H: 1	h: 1
I: 1	i: 2
J: 0	j: 0
K: 0	k: 0
L: 0	l: 2
M: 0	m: 2
N: 0	n: 4
O: 0	o: 1
P: 1	p: 1
Q: 0	q: 0

continue

Sample Run (continued)

R: 0	r: 3
S: 1	s: 3
T: 0	t: 2
U: 0	u: 1
V: 0	v: 1
W: 0	w: 0
X: 0	x: 0
Y: 0	y: 3
Z: 0	z: 0

Non-alphabetic characters: 14

EXERCISES

1. Write a Java program that accepts an array of integers and returns the sum of all the even numbers in the array. For example, if the input array is {1, 2, 3, 4, 5, 6}, the program should return 12 (which is the sum of 2, 4, and 6).
2. Write a Java program that accepts an array of strings and returns the length of the longest string in the array. For example, if the input array is {"cat", "dog", "elephant", "bird"}, the program should return 8 (which is the length of "elephant").
3. Write a Java program that accepts two arrays of integers and returns a new array that contains only the common elements between the two arrays. For example, if the first array is {1, 2, 3, 4, 5} and the second array is {3, 4, 5, 6, 7}, the program should return an array with {3, 4, 5}.

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:

```
double[] prices;  
double 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[] grades = {'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 list
- ❖ An initializer list can be used only in the array declaration
- ❖ See `Primes.java`

```
//*****
//  Primes.java      Author: Lewis/Loftus
//
//  Demonstrates the use of an initializer list for an array.
//*****

public class Primes
{
    //-----
    //  Stores some prime numbers in an array and prints them.
    //-----
    public static void main(String[] args)
    {
        int[] primeNums = {2, 3, 5, 7, 11, 13, 17, 19};

        System.out.println("Array length: " + primeNums.length);

        System.out.println("The first few prime numbers are:");

        for (int prime : primeNums)
            System.out.print(prime + "  ");
    }
}
```

Output

```
Array length: 8
The first few prime numbers are:
2 3 5 7 11 13 17 19
```

```
//*****
// Primes.java
//
// Demonstrate
//*****
```

```
*****
array.
*****
```

```
public class Primes
{
    //-----
    // Stores some prime numbers in an array and prints them.
    //-----
    public static void main(String[] args)
    {
        int[] primeNums = {2, 3, 5, 7, 11, 13, 17, 19};

        System.out.println("Array length: " + primeNums.length);

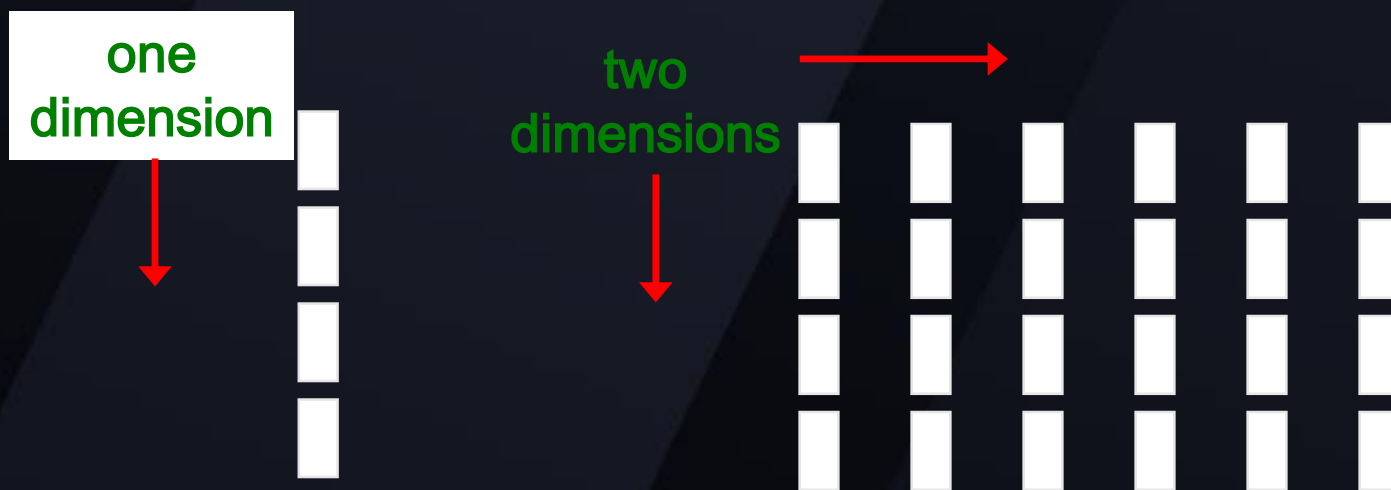
        System.out.println("The first few prime numbers are:");

        for (int prime : primeNums)
            System.out.print(prime + " ");
    }
}
```

2D ARRAYS

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

- ❖ An array element is referenced using two index values:

```
value = table[3][6]
```

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

```

//*****
//  TwoDArray.java          Author: Lewis/Loftus
//
//  Demonstrates the use of a two-dimensional array.
//*****

public class TwoDArray
{
    //-----
    //  Creates a 2D array of integers, fills it with increasing
    //  integer values, then prints them out.
    //-----

    public static void main(String[] args)
    {
        int[][] table = new int[5][10];

        // Load the table with values
        for (int row=0; row < table.length; row++)
            for (int col=0; col < table[row].length; col++)
                table[row][col] = row * 10 + col;

        // Print the table
        for (int row=0; row < table.length; row++)
        {
            for (int col=0; col < table[row].length; col++)
                System.out.print(table[row][col] + "\t");
            System.out.println();
        }
    }
}

```

```
//*****  
//  TwoDArray.java          Author: Lewis/Loftus
```

Output

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49

```
//  
public static void main(String[] args)  
{  
    int[][] table = new int[5][10];  
  
    // Load the table with values  
    for (int row=0; row < table.length; row++)  
        for (int col=0; col < table[row].length; col++)  
            table[row][col] = row * 10 + col;  
  
    // Print the table  
    for (int row=0; row < table.length; row++)  
    {  
        for (int col=0; col < table[row].length; col++)  
            System.out.print(table[row][col] + "\t");  
        System.out.println();  
    }  
}
```

EXAMPLE

❖ Which of the following are valid declarations? Which instantiate an array object? Explain your answers.

- `int primes = {2, 3, 4, 5, 7, 11};`

Invalid; an int cannot be declared and initialized using an initializer list. The brackets are missing.

- `float elapsedTimes[] = {11.47, 12.04, 11.72, 13.88};`

Valid; the brackets can be placed either after the element type or after the reference variable. However, this is not the preferred technique. This declaration creates an array object.

- `int[] scores = int[30];`

Invalid; the right hand side of the assignment operator must contain either an initializer list or a new operation.

- `int[] primes = new {2,3,5,7,11};`

Invalid; “new” on the right hand side of the assignment operator is neither necessary nor acceptable.

EXAMPLE

- `int[] scores = new int[30];`

Valid; the assignment is correct Java syntax. This declaration creates an array object.

- `char grades[] = {'a', 'b', 'c', 'd', 'f'};`

Valid; the brackets can be placed either after the element type or after the reference variable. However, this is not the preferred technique. This declaration creates an array object.

- `char[] grades = new char[];`

Invalid; the size of the array must be indicated when the array is instantiated.

EXAMPLE

❖ Write an array declaration and any necessary supporting classes to represent the following statements:

a. students' names for a class of 25 students

```
String[] students = new String[25];
```

b. students' test grades for a class of 40 students

```
int[] grades = new int[40];
```

or, for simple letter grades:

```
char[] grades = new char[40];
```

or, for letter grades that include pluses and minuses

```
String[] grades = new String[40];
```

c. students' names for a class and homework grades for each student

```
Student[] myClass = new Student[MAX];
```

```
public class Student
```

```
{
```

```
    private String name;
```

```
    private int[] grades;
```

```
}
```

d. for each employee of the L&L International Corporation: the employee number, hire date, and the amount of the last five raises

```
Employee[] employees = new Employee[MAX];  
public class Employee  
{  
    private int employeeNumber;  
    private String hireDate;  
    private double raise[] = new double[5];  
}
```

EXAMPLE

❖ Describe what problem occurs in the following code. What modifications should be made to it to eliminate the problem?

```
int[] numbers = {3, 2, 3, 6, 9, 10, 12, 32, 3, 12, 6};  
for (int count = 1; count <= numbers.length; count++)  
    System.out.println(numbers[count]);
```

The for loop fails to print the 0th element of the array, and attempts to print the nonexistent 11th element of the array. As a consequence, an `ArrayIndexOutOfBoundsException` is thrown. The problem can be eliminated by providing a for loop which initializes count to 0 (rather than 1) and tests if count is less than (rather than less than or equal to) numbers.length.

SUMMARY

- Arrays
- 2D Arrays



Thank You

www.midocean.ae

