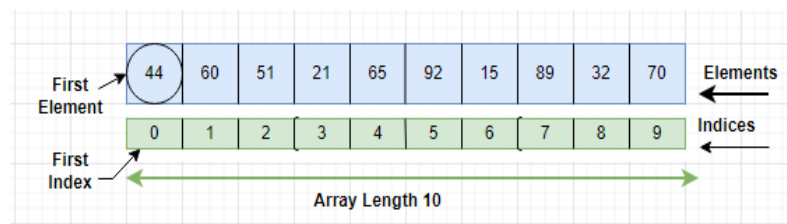


CHAPTER 10: ARRAYS

10.1 PROPERTIES

Information storage is an important task for programmers to perform in today's technologically advanced world. For later use, each and every item of information must be stored in the memory location. We already know what a variable is and how it stores data. The same way, data is stored using an array. However, Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value. It is a data storage where we store a fixed set of similar elements. For example, if we want to store the names of 40 students then we can create an array of the string type that can store 40 names.

Any data type, including primitive types like integer, double, and Boolean as well as object types like String, can be stored in a Java array. Arrays can also be multi-dimensional, meaning that they can have multiple rows and columns. An array's length is assigned when it initially gets constructed. After creation, its length is fixed and cannot be changed. Arrays in Java are index-based, the first element of the array is stored at the 0th index, the 2nd element is stored on the 1st index and so on.



10.2 DECLARATION, CREATION AND INITIALIZATION

Making an array in a Java program involves three distinct steps:

- Declare the array name.
- Create the array.
- Initialize the array values.

To create an array, an array variable of the desired data type must be declared first. The following is the general form for declaring an array:

```
data_type variable_name [ ] ;
```

Here, the data type determines what type of elements will be stored in the array. For example:

```
int quiz_marks [ ] ;
```

In the above declaration `quiz_marks` is the name of the array. And this array will store only the integer types of elements. Although this declaration establishes the fact that `quiz_marks` is an array variable, no array actually exists.

To link `quiz_marks` with an actual, physical array of integers, we must allocate one actual array using the “new” keyword and assign it to `quiz_marks`. “new” is a special operator that allocates memory for arrays.

```
quiz_marks = new int [5] ;
```

Now, we have finally created the actual array. Here, 5 defines the length of the array. That means this array can store five elements.

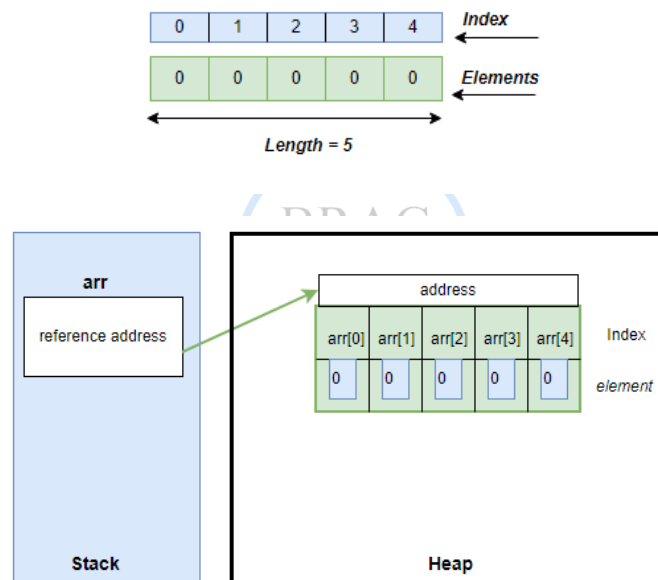
^{data type of each element}
`int []`
^{name of the array}
`total_marks;` } Declaration

`total_marks = new int [5];` } Creation
^{Array length/ total number of elements}

This was the descriptive process for declaration and creation of an array. There is another short way for declaring and creating an array. Instead of using two separate lines of codes we can simply use one line for declaration and creation of an array.

```
int [ ] total_marks = new int [5];
```

Here, we have created an integer array, which can store 5 elements. Initially when we create an integer array, all the elements of the array remain 0.



Stack is a temporary memory. after using new, now the array will be inserted into an actual data storage memory called heap. Also, the name of the array only refers to the address where the actual array is stored.

Initial elements of the array change according to the data type. According to different data type array initial elements:

Data Type	Default Values
Byte	0
Short	0
Int	0
Long	0
Float	0.0
Double	0.0
Boolean	false
Char	'\u0000' which is the Unicode for null
String	null

Object	null
--------	------

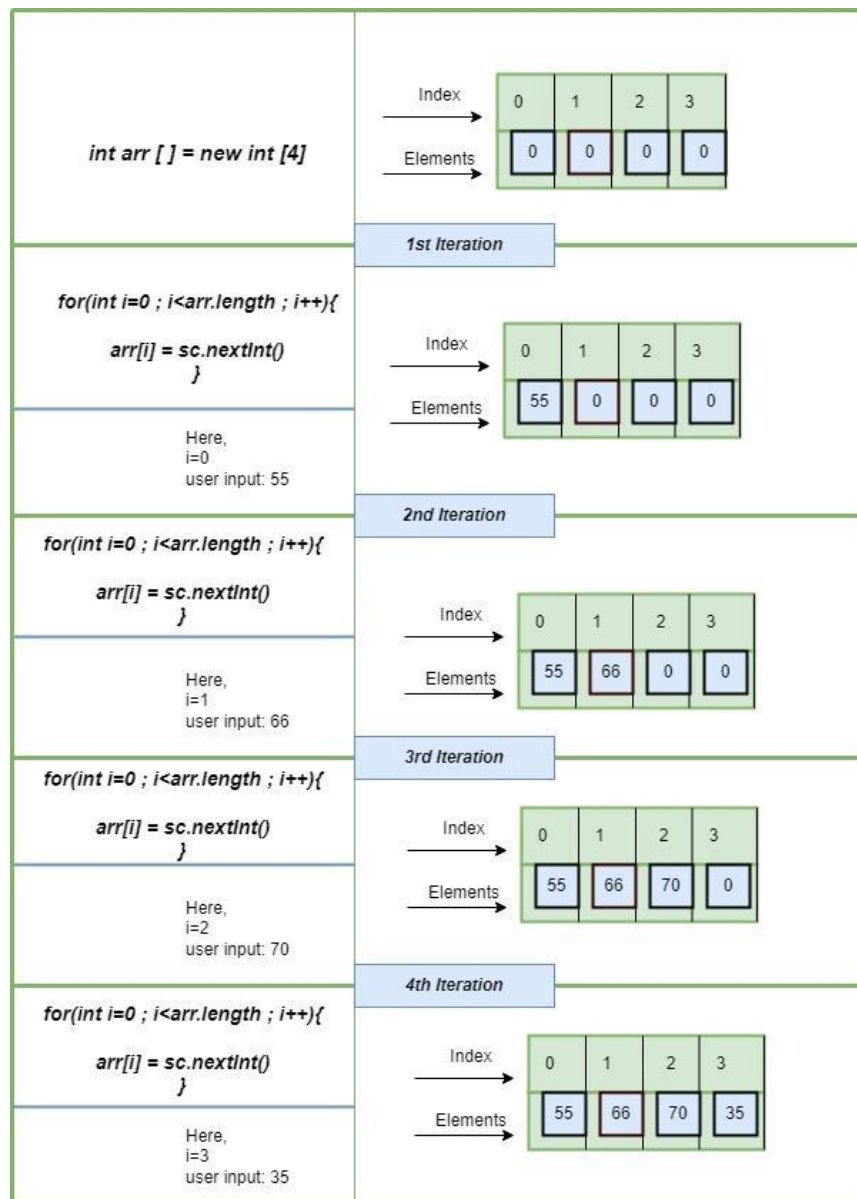
Array declaration and creation examples with different data types:

<code>int [] a1 = new int [5];</code>	
<code>double [] a1 = new double [5];</code>	
<code>float [] a1 = new float [5];</code>	
<code>String [] a1 = new String [5];</code>	

To sum up, there are two steps involved in constructing an array. First, you must create a variable with the specified data type. Second, you must use the “new” keyword to allocate the memory required to store the array and then assign that memory to the array variable. After that default elements will be assigned automatically to the array according to the data type.

Now, we need to store our actual elements. There are multiple ways to assign values into an empty array. The general code for inserting any element into any specific index of an array is:

array_name [index_number] = element



Here, at first, we declared the array with length 4. We declared the data type integer that is why the initial array is loaded with all 0's. After that we initiated the for loop, where we stated $i=0$. Because the first index of an array is 0. The loop will stop when we reach the last index of the array. For that reason we have to know the length of the array. In Java, the array length is the number of elements that an array can hold. There is no predefined method to obtain the length of an array. We can find the array length in Java by using the array attribute length. We use this attribute with the array name. In the above example if we write:

```
System.out.println(arr.length);
```

Our program will give us 4 as an output. So, the last index of an array is $\text{array.length}-1$.

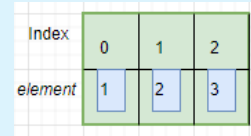
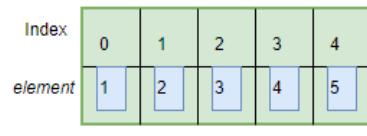
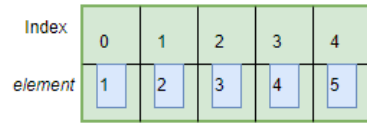
In the above diagram, the value of "i" becomes: 0,1,2,3.

So, the first index is 0 and the last index is 3 which is the length - 1. Inside the loop we are taking inputs from the user and inserting that input into the array index.

```
arr[i] = sc.nextInt();
```

In this line, arr is the name of the array, i is the index number. `sc.nextInt()` is taking inputs from the user. Here, `nextInt` only takes the integer inputs

In this way we can take inputs from the user and create an array. There are various ways to initialize the array values. Some of them are given below:

Process	Description	Code
Dynamically allocating one by one	Here, a is the name of the array. a[0] means the first index of that array. so in the first index we insert the element 1 and so on	<pre>int a [] = new int [3]; a[0] = 1; a[1] = 2; a[2] = 3;</pre> <p>Created array:</p> 
At the time of declaration	We can also insert the elements at the time of array creation. here at first we declared the data type which is int, then a is the name of the array. And then on the right side of the equal sign inside curly braces we write the elements we want to insert.	<pre>int a[] = { 1, 2, 3, 4, 5 };</pre> <p>Created array:</p> 
Using loop	Using for or while loop we can also insert values into an array. Here, the loop starts from 0, and it will continue until it doesn't reach the last index of the array. The length of the array is 5. the value of i will be 0,1,2,3,4.	<pre>int a [] = new int [5]; for (int i = 0; i < a.length; i++) { a[i] = i + 1; }</pre> <p>Created array:</p> 
Taking User Input	We can insert elements into an array by taking user inputs. a[i] = sc.nextInt(); In this line, a is the name of the array, i is the index number. sc.nextInt() is taking inputs from the user. Here, nextInt only takes the integer inputs.	<pre>import java.util.Scanner; Scanner sc= new Scanner(System.in); int a[] = new int [5]; for (int i = 0; i < 5 ; i++) { a[i] = sc.nextInt(); }</pre>
Random value generator	Java has a special class called Random. First we need to import that class. After that we create an object of that class just like a scanner. rd.nextInt(10). Here, rd is the object name, nextInt will generate only integer values and 10 defines the range. Any random numbers between 0 to 10 will be inserted into the array.	<pre>import java.util.Random; Random rd = new Random(); int a[] = new int [5]; for (int i = 0; i < 5 ; i++) { a[i] = rd.nextInt(10); }</pre>

In this way, we can declare, create and initialize an array in java. For better efficiency practice different approaches with different data types for array initialization.

10.3 ARRAY ITERATION

Arrays are used to store homogeneous elements, meaning the same type of elements. Till now we have already learned what an array is and how to declare, initialize an array. Throughout this process we stored our necessary elements into the array. Our task is not only to store the values but also we need to work with the values. Here

comes the iteration part where we need to access the elements from the array. Iterating over an array means accessing each element of the array one by one.

```
String student_name [ ] = {"David", "Jon", "Sam", "Elsa", "Anne"};
System.out.println(name[0]);
```

Output:
David

Suppose we have an array named `student_name`. Where we stored the name of those students who were the top scorers in the midterm examination. Now you guys want to know the name of the student who got the highest marks and became first.

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

Index 0 contains the highest scorer. So inside the print statement if we write the array name with index 0. then the output will show the highest scorer's name. Here, we just printed the name. We can also store elements as separate variables from the array.

```
String student_name [ ] = {"David", "Jon", "Sam", "Elsa", "Anne"};
top_scorer = student_name [0]
System.out.println(top_scorer);
```

Output:
David

Here, we created a new variable called `top_scorer` and copied an element from the array inside the new variable. The element will remain the same in the array, we just copied it and stored that copy inside a new variable.

Print the element	<code>System.out.println (array_name [index_number]);</code>
Copy and print the element	<code>Variable_name = array_name [index_number];</code> <code>System.out.println (Variable_name);</code>

Sometimes, we need to iterate the whole array instead of a specific index. There are multiple ways to iterate over a whole array. Mostly we use a loop for that process. The first index of an array is 0 so we initialize for loop variable `i = 0`. Then `i` will continue incrementing by 1 value until it doesn't reach the last index which is `array length - 1`. Inside the loop we write the print statement for printing the array.

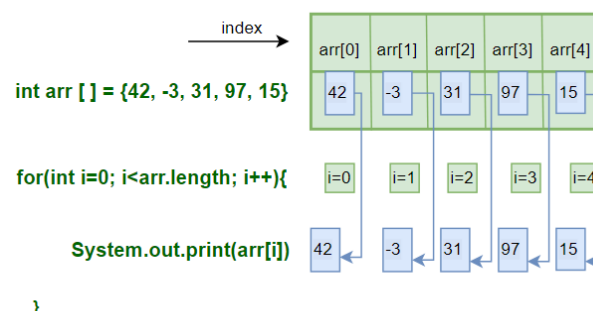
```
int arr [ ] = {42,-3,31,97,15};

for(int i =0; i< arr.length;i++) {
    System.out.println(arr[i])
}
```

Output:

42
-3
31
97
15

Here, loop variable `i` works as the index value and `i` iterate over the array.



We can iterate an array using not only for loop but also while loop and do while loops. As per our convenience and requirements we can use any of the iteration processes.

For Loop	loop starts from 0, runs until it reaches the last index which is <code>array length-1</code> . <code>i</code> increments by 1. iterate over the whole array.	<code>int a [] = {5,6,9,8,6};</code>
----------	---	---------------------------------------

	inside the loop, print statement prints the elements.	for (int i = 0; i < a.length; i++) { System.out.println(a[i]); }
While Loop	loop starts from 0, runs until it reaches the last index which is array length-1. inside the loop, print statement prints the elements and i increments by 1. Iterate over the whole array.	int a [] = {5,6,9,8,6}; int i = 0; while (i < a.length) { System.out.println(a[i]); i++; }
Do While Loop	The initial variable i starts from 0. inside the do block we write the statements to be executed which is printing elements of the array. Then we increment i by 1. Then using while we set the value to stop the loop.	int a [] = {5,6,9,8,6}; int i = 0; do { System.out.println(a[i]); i++; } while (i < a.length);

10.4 OPERATIONS ON AN ARRAY

Several different operations can be performed on an array. Some examples are given below. For all the examples, we shall use this array:

```
int arr [] = {55, 66, 7, -2, 9};
```

Operation	Description	Code	Output
Length	Counts total number of elements in an array.	System.out.println(arr.length);	5
Traverse	Traversing through all the elements in the array one by one. Can be used to perform tasks that require visiting all the elements such as printing all the elements.	for (int i = 0; i < arr.length; i++) { System.out.println(arr[i]); }	55 66 7 -2 9
Insertion	Adds an element at the given index.	int x [] = new int [5]; x [1] = 10; System.out.println(x[1]); for (int i = 0; i < x.length; i++) { System.out.println(x[i]); }	0 10 0 0 0
Deletion	Deletes an element at the given index.	arr [1]= 0; for (int i = 0; i < arr.length; i++) { System.out.println(arr[i]); }	55 0 7 -2 9
Update	Updates an element at the given index.	arr [1]= 65; for (int i = 0; i < arr.length; i++) { System.out.println(arr[i]); }	55 65 7 -2 9

Copying an array	Create another array with the same length and datatype, and copy all the elements to the new array one by one. The memory addresses of both arrays are different and changing one array does not affect the another.	<pre>int arr2 [] = new int [arr.length]; for (int i = 0; i < arr.length; i++){ arr2[i]= arr[i]; } for (int i = 0; i < arr2.length; i++) { System.out.println(arr2[i]); }</pre>	55 0 7 -2 9
------------------	---	--	-------------------------

10.5 MULTIDIMENSIONAL ARRAYS

The arrays we have looked at so far were all one-dimensional arrays (values are only in a row). When we want to represent data in a tabular form (rows and columns), we use multidimensional arrays. These are basically arrays of arrays, i.e. an array inside another array. The basic format for creating such arrays is-

data_type[1st dimension][2nd dimension][...][Nth dimension] **array_name** = new **data_type**[size1][size2]...[sizeN];

For example:

```
int[][][] newArray = new int[10][20][30]; //This is a multidimensional array.
```

```
int[][] numbers = { {1, 2, 3, 4}, {5, 6, 7} }; //This is also a multidimensional array.
```

To access or modify values of a multidimensional array, we can use array indexing, just like previous examples, but instead of just one index, we use multiple indexes to point to the location of a certain value. For example **numbers[1][2]** represents 7, as [1] means the index of the outer array, and [2] is the index of the inner array.

Example Code	Output
<pre>//To access values. public class Example1 { public static void main(String[] args) { int[][] numbers = { {1, 2, 3, 4}, {5, 6, 7} }; System.out.println(numbers[0][1]); } }</pre>	2
<pre>//To modify values. public class Example2 { public static void main(String[] args) { int[][] numbers = { {1, 2, 3, 4}, {5, 6, 7} }; System.out.println(numbers[0][0]); //Prints 1. numbers[0][0] = 9; System.out.println(numbers[0][0]); //Prints 9. } }</pre>	1 9

10.6 WORKSHEET

- A. Answer the following theoretical conceptual questions:
1. Multi-Dimensional Arrays: Discuss the concept of multi-dimensional arrays, such as 2D arrays or matrices. Explain how they are implemented and provide examples of their applications.
 2. Array Indexing: Explain the concept of array indexing and how it is used to access elements in an array. Discuss the starting index convention (0-based or 1-based) in different programming languages and its implications.
 3. Array Memory Allocation: Describe how arrays are stored in memory. Discuss the contiguous allocation of elements in memory and the impact of this allocation on array access and performance.
 4. Static vs. Dynamic Arrays: Compare static and dynamic arrays. Explain the difference between fixed-size arrays and arrays with dynamic memory allocation. Discuss the advantages and limitations of each approach.
 5. Array Resizing: Discuss the challenges associated with resizing arrays, especially when they are implemented with static memory allocation. Explain how dynamic arrays address this issue by dynamically resizing the array as needed.
 6. Array Copying and References: Explain how arrays are copied or referenced in different programming languages. Discuss the implications of shallow copying versus deep copying for multidimensional arrays or arrays containing mutable objects.

- B. Given an array of integers, write a program to find the maximum element in the array.

Given Array 1
 {5, 2, 8, 3, 9}
 Sample Output 2
 9



- C. Write a function that takes an array of integers as input and returns the sum of all the elements.

Given Array 1
 {5, 2, 8, 3, 9}
 Sample Output 1
 27

Inspiring Excellence

- D. Write a program that takes an array and a target element as input and prints the index of the target element in the array. If not found in the array, print "Not Found".

Given Array 1
 {5, 2, 8, 3, 9}
 8
 Sample Output 1
 Target element is at index: 2

Given Array 2
 {5, 2, 8, 3, 9}
 18
 Sample Output 2
 Not Found

- E. Given an array of integers, write a program that prints the occurrence of each element in the array.

Given Array 1
 {5, 3, 5}
 Sample Output 1
 5 is repeated 2 time(s)
 3 is repeated 1 time(s)

- F. Given two arrays, write a program that prints an array containing the common elements in both arrays.

Given Arrays

{5, 2, 8, 3, 9}

{6, 3, 5, 11, 10}

Sample Output 1

{3, 5}

- G. Given an array with duplicate elements, write a program that removes the duplicates and prints a new array with unique elements only.

Given Array 1

{5, 2, 8, 3, 2}

Sample Output 1

{5, 2, 8, 3}

- H. Write a program that takes an array as input and returns a new array with the elements reversed.

Given Array 1

{1, 2, 3, 4, 5}

Sample Output 1

{5, 4, 3, 2, 1}



Inspiring Excellence