FACULTY OF ENGINEERING
Department of Telecommunications and Electronics Engineering

ARRAYS

Basic Java Course

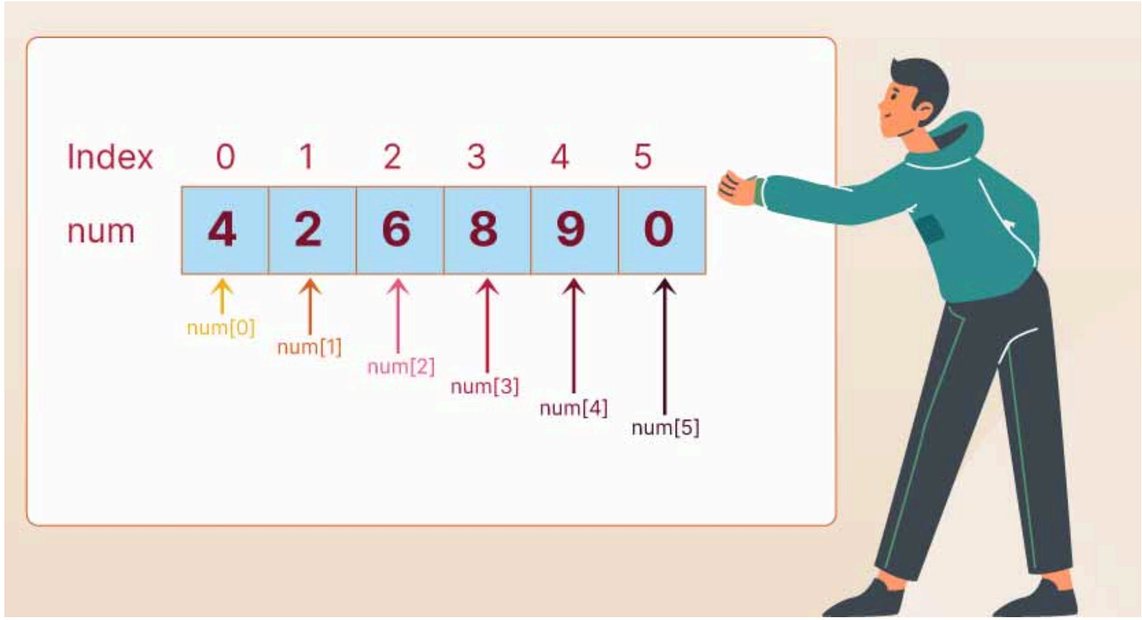
Advisor: Prof. Thap Thareoun

Present By : Ngouch HongCheng
: Saing LymChhun

CONTENT

INTRODUCTION

- In Java, an array is a data structure that can store a fixed-size sequence of elements of the same data type.
- An array is an object in Java, which means it can be assigned to a variable, passed as a parameter to a method, and returned as a value from a method.



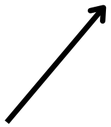
TYPES OF ARRAYS

A. SINGLE DIMENSION ARRAYS

- **A 1D (one-dimensional) array** is a basic array structure in Java, which stores a collection of elements of the same type in a linear format.
- Each element can be accessed using its index, starting from 0.

Syntax

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



dataType[] arrayName;

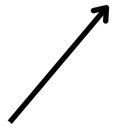
TYPES OF ARRAYS

B. 2D ARRAYS

- **A 2D (two-dimensional) array** is an array of arrays, which can be visualized as a table with rows and columns.
- Each element in a 2D array is accessed using two indices, one for the row and one for the column.

Syntax

```
int[][] matrix = {{1, 2, 3},{4, 5, 6},{7, 8, 9}};
```



dataType[][] arrayName;

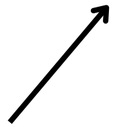
TYPES OF ARRAYS

C. 3D ARRAYS

- **A 3D (three-dimensional) array** is an array of arrays of arrays, which can be visualized as a collection of 2D arrays stacked on top of each other.
- Each element in a 3D array is accessed using three indices: one for the depth, one for the row, and one for the column.

Syntax

```
int[][][] cube = {{{1, 2, 3},{4, 5, 6}},{{7, 8, 9},{10, 11, 12}},{{13, 14, 15},{16, 17, 18}}};
```



dataType[][][] arrayName;

ACCESSING AND MODIFYING ARRAYS

A. 1D ARRAYS

- To access an element in a 1D array, use the array name followed by the index of the element in square brackets. The index starts at 0.

B. 2D ARRAYS

- To access an element in a 2D array, use two indices: the first for the row and the second for the column.

C. 3D ARRAYS

- To access an element in a 3D array, use three indices: the first for the depth, the second for the row, and the third for the column.

ACCESSING AND MODIFYING ARRAYS

NOTE

- **Bounds Checking:** Always ensure that you do not access an index outside the array bounds to avoid *ArrayIndexOutOfBoundsException*.

- **Initialization**: Make sure the array is properly initialized before accessing or modifying its elements.

- **Enhanced For Loop**: For readability, use enhanced for loops when you only need to traverse and not modify the elements directly.

COMMON OPERATIONS

A. ASSIGNMENT OPERATOR “ = “

- The assignment operator is used to assign values to the elements of an array and to initialize arrays.

B. ARRAY LENGTH OPERATOR

- You can use the .length property to get the length (number of elements) of an array.

C. INDEX OPERATOR ([])

- The index operator is used to access or modify elements at a specific index.

COMMON OPERATIONS

D. ENHANCE FOR LOOP

- The enhanced for loop is a syntactic sugar for iterating through arrays. It simplifies the code and eliminates the need for explicit index management.

D. COMPARISON OPERATORS (==, !=)

Comparison operators can be used to compare array references, not their contents.

COMMON OPERATIONS

F. THE ARRAYS CLASS METHODS

- The *java.util.Arrays* class provides various utility methods for array manipulation.
- ***Arrays.toString()***: converts the array to a string representation.
- ***Arrays.sort()***: sorts the array in ascending order.
- **Arrays.fill()**: fills the array with a specified value.
- **Arrays.equals()**: compares two arrays for equality.
- **Arrays.copyOf()**: copies an array to a new array.
- **Arrays.binarySearch()**: searches for a specific value in a sorted array using binary search algorithm.

COMMON OPERATIONS

SUMMARY

- **Assignment (=)**: Assign values to array elements and initialize arrays.
- **Length (.length)**: Get the length of an array.
- **Indexing ([])**: Access and modify elements at specific indices.
- **Enhanced For Loop**: Iterate through array elements easily.
- **Arrays Class Methods**: Provide utility functions like toString(), sort(), fill(), equals(), copyOf(), and binarySearch().
- **Comparison Operators (==, !=)**: Compare array references.

EXTRA PRACTICE

PRACTICE

1. Write a Java program that creates an array of integers and calculates the sum of all elements in the array.

2. Write a Java program that finds the largest element in an array of integers.

END