(6 days) [illegible] done [illegible] 7. OOPS) Rem

Date
Page No.

5 (a) (Arrays)

→ Array

→ An array is a collection of elements (values) of the ~~date~~ same data type, stored in contigous memory locations. It's a way to group multiple values under a ~~single~~ single variable name.

- Why we need ~~or~~ Array or why use Array.

(i) Store multiple values of the same datatype.
(ii) Access values ~~[illegible]~~ using an index.
(iii) Organize data efficiently for processing.

Example:- (Not Array)

- Insted of creating 5 variables like this: ✗
  int a = 10, b = 20, c = 30, ~~d~~ = 40, e = 50;

- we can use array like this. (Array) ✓

- int[] arr = {10, 20, 30, 40, 50};

1. # Declaring an Array (Syntax):

dataType[] arrayName;

Ex → int[] numbers; // (numbers is getting defined in the stack)

2. # Creating an Array

→ size of an array to create.

dataType ~~int~~ [] arryName = new dataType[size];

Ex → int[] number = new int[5];
// (initialization: actuall here object is created in the Heap).

Date
Page No.

3. # Declaring + Creating ~~to~~ + Initializing.

(1) Static Initialization (when values are known):

```
int[] arr = {10, 20, 30, 40, 50};
```

(2) Dynamic Initialization (values given later):

```
int[] arr = new int[5];
arr[0] = 10;
arr[1] = 20;       => 0 & 1 are index.
arr[2] = 30;
// --- and so on.
```

4. # Array Operations

(1) Accessing Array Elements. (Access)
=> Use indexing, which starts from 0.
Ex -> System.out.print(arr[0]); //prints first element.

(2) Changing values & (Modify)
Ex ~~arr[1]~~ => arr[1] = 99; // changes the second element to 99.

(3) Array Length (Length)
Ex -> System.out.println(arr.length); -> To get an array size.

P.T.O for more
$\longrightarrow$

(4) For-Each Loop with Arrays

-> use to read values (not change them directly):

ex->

```
int[] arr= {10, 20, 30};
// for every element in array, print the element.
> for(int num : arr){   -> //here num represents
                              element of the array
system.out.println(num);
}
```

(5) For Loop with Arrays

```
int[] arr= {10,20,30};

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

(5) Important Concepts.

~~#~~(1) Default Values in Arrays

Data Type	Default values.
1. int	0
2. double	0.00.
3. boolean	false
4. String	null.

(14min) | chatgpt.

Date
Page No.

(2) Null Arrays
→ Array variable itself can be null.

Ex →
```
int[] arr = null;
if(arr != null){
System.out.println(arr.length);
}
```

(3) Array of objects (Non-primitive)
→ can only be assigned at Non-primitive data Type.

Example →
```
String[] names = new String[3];
names[0] = "Alice"; // elements can be assigned
                       later.
```

(4) Reading Input into Arrays (from Scanner)

```
Scanner sc = new Scanner(System.in);
int[] inputArr = new int[5];                    → taking input
for(int i = 5; i< inputArr.length; i++){
   inputArr[i] = sc.nextInt();
}.
```

(5) printing the Reading Values.
```
for(int i = 5; i< inputArr.length; i++){   → printing the taken inputs
System.out.println(~~[illegible]~~(InputArr[i]);

}
```

6 Array Methods (java.util.Arrays)

- Import Arrays utility

  => import java.util.Arrays;

(i) Copying Arrays ~~A.copyof(~~ (Arrays.copyof(original, newLength)

=> Create a new array copying existing elements (can specify new size).

```
ex -> int[] a = {5,6};
      int[] b = Arrays.copyof(a, 4);
```

(ii) Arrays.toString(array)

-> use to convert array to readable String (helpful for logging, debugging).

```
ex -> int a = {1,2,3};
   System.out.println(Arrays.toString(a));   //output: [1,2,3]
```

(iii) ~~Arrays.copyof(original, ne~~

(iii) Arrays.fill(array, value)

-> Fill an array with a single value (eg. default Status Code, Placeholder).

```
ex -> int[] a = new int[5];
      Arrays.fill(a,-1);    // [-1,-1,-1,-1,-1]
```

(iv) Arrays. equals (arr1, arr2)

-> Check if two arrays have same length and values (element-wise comparision).

Ex -> int a[] = {1,2,3};
      int b[] = {1,2,3};
      System.out.println(Arrays.equals(a,b)); //true

(V) Arrays . sort (array)

-> Sort array in ascending order (primitive types or comparable objects).

Ex -> int[] a = {5, 1, 3};
      Arrays.sort(a); // [1,3,5]

~~(vi) Arrays. Copy of~~

(Vi) Arrays. copyofRange (original, from, to)

-> Copy a portion (slice) of an array.

Ex -> int[] a = {10, 20, 30, 40};
      int [] sub = Arrays.copyofRange(a, 1,3); //[20, 30]

Date
Page No.

7. Arrays with Methods in Java

(i) Passing Array to Method.

→ Sending an array to a method so it can use or print the elements

Ex →

// Method that takes an array & prints its elements →

```
public class Main {
    public static void printNumbers(int[] numbers) {
        for(int num: numbers) {
            System.out.println (num);
        }
    }

    public static void main(String[] args {
        int[] arr = {10,20,30};
        printNumbers(arr);  //passing array to the method.
    }
}
```

(ii) Modifying Array Inside a Method

→ If we change an array inside a method, it also change the original array.

Ex →

// Method that change the first element of the array. ←

```
public class Main {
    public static void changeFirst(int[] numbers) {
        numbers[0] = 99;
    }
    public static void main(String[] args) {
        int [] arr = {1,2,3}
      changeFirst(arr);      // Array is modified inside the method.
    System.out.println(arr[0]);      // output : 99
          }
    }
```

(3) Returning an Array from a Method

→ A method can create and return an arry to the main method.

Ex→

```
                  public class Main {
// method that →      public static int[] getMarks(){
   returns an             return new int[]{60,70,80};
   array                }
                  public static void main(String[] args){
                      int[] marks = getMarks(); // Get array from method.
                  System.out.println(marks[0]);   // output: 60
                    }
                  }
```

8. Searching (Manual)

→ Checks if an element exists.

Ex →
```
public class Main{
  public static void main(String[] args){
    int[] arr = {10, 20, 30, 40, 50};
     int target = 30;
    boolean found = false;

   //using Regular for Loop Search.
   for (int i=0; i< arr.length; i++){
      if(arr[i] == target) {
         found = true;
          break;
      }
   }

   //using for-each Search.
    for(int num : arr){
    if(num == target){
       found = true;
        break;
     }
   }
```

For Loop →

For-Each →

In Both
```
      if (found){
      System.out.println("found");
      } else {
      System.out.println("Not Found");
      }
    }
```