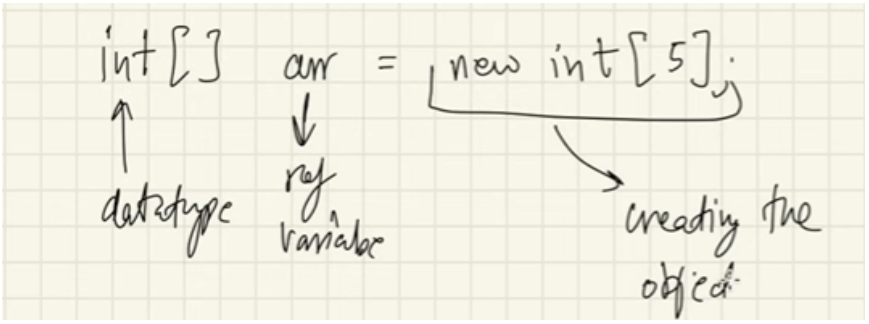
Arrays

- Collection of Homogenous Data types

Consider a situtaion to store roll numbers of students in school. Generally for a single student, we create a variable and store a value. A school will have ‘n’ students. creating ‘ n’ variables and storing a value doesnt work and high consumption of both time n space. → Here, we use ARRAY

Syntax

```
datatype[] variable_name=new datatype[size]
```



- **datatype → datatype of the array**. This need to be specified as all the elements in the array should have same datatype.

  Hetrogeneous Elements cannot be stored in a ARRAY

- ***datatype[] variable_name :*** **Declaration** of ARRAY. *‘variable_name’ is* getting defined in a **STACK** → **‘COMPILE TIME’**
- ***variable_name=new datatype[size] :*** Object is being created in the **HEAP** memory [**Initialisation of the variable**] → **‘RUNTIME’**
- ***new :*** Create an object

📌 ***HEAP*** objects are not Continous acc to JLS (Java Lang Specification)

Storing 5 Roll numbers of students in a class

```
int[] roll_num=new int[5]
// or, it can also be written as
int[] roll_num={1, 2, 3, 4, 5}
```

Indexing of an Array

- Position of an array
- Consider, int[] arr={1, 2, 3, 4, 5}

arr[0] gives me the element inthe 0th index → 1

arr[3]= 99 → this updates the current element in the array (Mutability)

NULL in java

```
String[] arr=new String[5];
System.out.print(arr[0]);
```

This give an output as 'NULL' → special Literal (which cannot be assigned/ declared as null type)

```
String str=null;
int num=null; // This throws an error
```

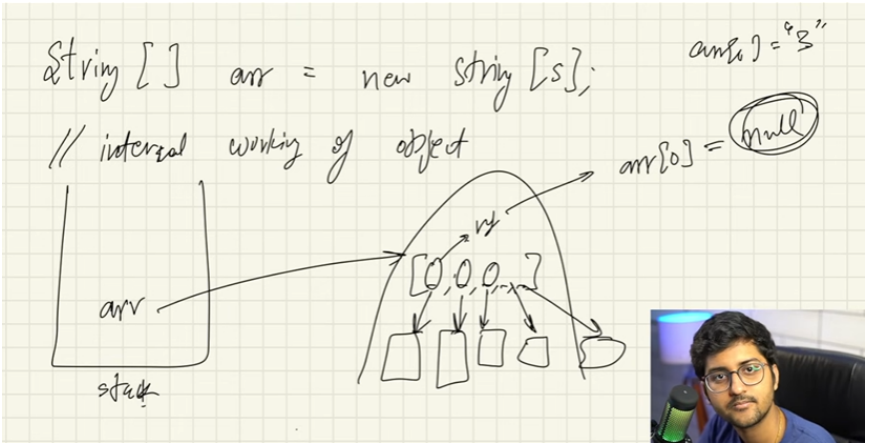
Null type cannot be assigned to PRIMITIVES

📌 Any referece varialble by default is a null , before initialising.

Consider, an array of Objects (not primitives)

```
String[] arr=new String[5];
```

📌 WKT, Primitives are stored in stack memory only, while non-primitves are stored in heap memory.



Arrays : Primitives

```
public class nullInput {
    public static void main(String[] args) {
        int[] arr=new int[5];
        arr[0]=23;
        arr[1]=89;
        arr[2]=80;
        arr[3]=56;
        arr[4]=43;

        System.out.println(arr[3]);
```

```
        //Internally stored as [23,89,80,56,43]
    }
}
```

```
//Using FOR LOOP:
import java.util.Scanner;
public class nullInput {
    public static void main(String[] args) {
        int[] arr=new int[5];
        Scanner in=new Scanner(System.in);
        for(int i=0;i<(arr.length);i++){
            System.out.print("Enter the value at the index"+"
            int input=in.nextInt();
            arr[i]=input;
        }

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

This can be also written using **ENHANCED FOR LOOP**

```
import java.util.Scanner;
public class nullInput {
    public static void main(String[] args) {
        int[] arr=new int[5];
        Scanner in=new Scanner(System.in);
        for(int i=0;i<(arr.length);i++){
            System.out.print("Enter the value at the index"+"
            int input=in.nextInt();
            arr[i]=input;
        }
```

```
        //System.out.println(Arrays.toString(arr));
        /*Using the in-built method*/

        System.out.print("[");
        // for(int i=0;i<(arr.length);i++){
        //     System.out.print(arr[i]+",");
        // }
        for(int num:arr){ //For every element in array
            System.out.print(num + ", "+ " "); //here 'i' rep
        }
        System.out.print("]");
    }
}
```

Arrays : Primitives

```
import java.util.Arrays;
import java.util.Scanner;;
public class npArrays {
    public static void main(String[] args) {
        Scanner in=new Scanner(System.in);
        String[] str=new String[5];
        for (int i = 0; i < str.length; i++) {
            System.out.print("Enter the value at the index"+"
            String input=in.next();
            str[i]=input;
        }

        // System.out.println(Arrays.toString(str));

        System.out.print("[" + " ");
        for (int i = 0; i < str.length; i++) {
            System.out.print(str[i]+ ",");
        }
        System.out.print("]");
```

```
    }
}
```

Passing Arrays of Objects in Functions/Methods

```
import java.util.Arrays;

public class multiDimArr {
    public static void main(String[] args) {
        int[] num={15,20,40,62,34};
        System.out.println(Arrays.toString(num));
        change(num);
        System.out.println(Arrays.toString(num));
    }
    public static void change(int[] arr){
        arr[3]=100;

    }
}
/*Arrays are muttable in nature. while Strings are not*/
```

Multi Dimensional Arrays

2D Arrays

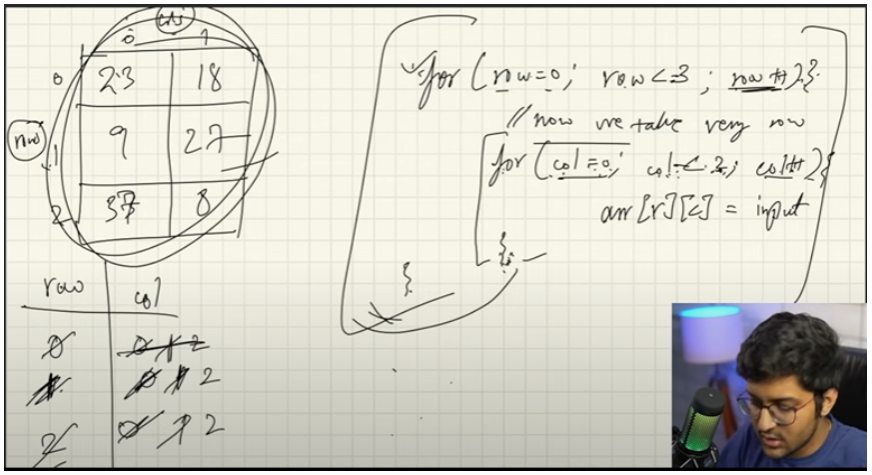
1. **Syntax**

```
//Defining 2D array
datatype[][] array_name=new datatype[size][];
```

```
//Defining 2D array
datatype[][] array_name={ {__,_,_},
```

```
                                                {_,_,_},
                                                {__,_,_}}
```

2. **Taking input**




```
import java.util.Arrays;
import java.util.Scanner;

public class multiDimArrays {
    public static void main(String[] args) {
        //int[][] arr=new int[3][];

        /*

        int[][] arr={
        {1, 2, 3},      // 0th index
        {4, 5},     //1st index
        {6, 7, 8, 9}  //2nd index
        }

        */

        /* Now, to fill the array or to take an input,
         for loop and iterate to every single index
         and then fill the input
        */
```

```
Scanner in=new Scanner(System.in);
int[][] arr=new int[3][2];

//input:

for (int row = 0; row < (arr.length); row ++) {
    System.out.println("This is"+" "+row+ " "+ "th"+
    //for each col in a row
    for(int col=0; col<(arr[row].length); col++){
        System.out.print("Enter the value at"+" "+ co
        arr[row][col]=in.nextInt();
    }
}
// System.out.println("Array content: " + Arrays.deep

//output:
// System.out.print("["+ " ");
// for (int row = 0; row < (arr.length); row ++) {
//     System.out.print("["+ " ");
//     //for each col in a row
//     for(int col=0; col<(arr[row].length); col++){
//         System.out.print(arr[row][col]+ ","+ "");
//     }
//     System.out.print("]"+ ","+ " ");
//     System.out.println();
// }
// System.out.print("]");

System.out.print("[");
// for (int row = 0; row < arr.length; row++) {
//     System.out.println(Arrays.toString(arr[row])+
// }
for (int[] a : arr) {
    System.out.println(Arrays.toString(a)+ ","+" ");
}
System.out.print("]");
```

```
    }
}
```

3. **When Number of Columns are not fixed**

```
public class colNoFixed {
    public static void main(String[] args) {
        int[][] arr={
            {1, 2, 3, 4},
            {5, 6},
            {7, 8, 9}};

        System.out.print("[");
        for (int row = 0; row < arr.length; row++) {
            //columns:
            for (int col = 0; col < arr[row].length; col++) {
                System.out.print(arr[row][col]+ " ");
            }
            System.out.println();
        }
        System.out.print("]");

    }

}
```

Array List

> Similar to VECTORS in Cpp

- when we have **no idea** about the **SIZE OF AN ARRAY**

1. **Syntax**

```
ArrayList<datatype> list=new ArrayList<datatype>();
```

2. ArrayList operations [In built methods that can be used]

```
import java.util.Scanner;
import java.util.ArrayList;

public class arrayListExample {
    public static void main(String[] args) {
        ArrayList<Integer> list=new ArrayList<>(10);

        /*only WRAPPER CLASSES can be used but not
        the primitive Data types.
        */

        /* https://www.geeksforgeeks.org/wrapper-classes-java

        list.add(67);
        list.add(74);
        list.add(77);
        list.add(90);
        list.add(37);
        list.add(28);
        list.add(89);
        list.add(81);
        list.add(83);
        list.add(76);
        list.add(15);
        list.add(90);
        list.add(57);

        System.out.println(list);
        System.out.println(list.contains(99));
        list.set(1,1000);
        System.out.println(list);
        list.remove(5);
```

```
        System.out.println(list);

    }

}
```

3. Iterate using for loop and creating an ArrayList

```
import java.util.Scanner;
import java.util.ArrayList;

public class arrayListExample {
    public static void main(String[] args) {
        Scanner in=new Scanner(System.in);
        ArrayList<Integer> list=new ArrayList<>(10);

        for (int i = 0; i < 5; i++) {
            list.add(in.nextInt());
        }
        System.out.println(list);
```

3. Get item at any index of the ArrayList

```
import java.util.Scanner;
import java.util.ArrayList;

public class arrayListExample {
    public static void main(String[] args) {
        Scanner in=new Scanner(System.in);
        ArrayList<Integer> list=new ArrayList<>(10);
//Get any item in the list
        for (int i = 0; i < 5; i++) {
            System.out.println(list.get(i)); //passing index
        }
```

```
/*using list[index] isnt the way*/
```

Why size in ArrayList can accept as many values we give ?

- Size is Fixed for ArrayList internally.
- **When the ArrayList is filled by some amount, It will create a new array list of may be double the size, Old elements gets copied to a new list and the old ArrayList gets dltd**
- **Annotised Time Complexity (Constant) → O(1)**

Multi Dimensional Array List

1. **Syntax**

```
ArrayList<ArrayList<datatype>> list=new ArrayList<datatype>(
```

```
import java.util.ArrayList;
import java.util.Scanner;

public class MultiDimArrayListEx {
    public static void main(String[] args) {
        Scanner in=new Scanner(System.in);
        ArrayList<ArrayList<Integer>> list=new ArrayList<>();

        //initialisation of Multi Dim array
        for (int i = 0; i < 2; i++) {
            list.add(new ArrayList<>());
        }

        //Adding an element
        for (int i = 0; i<2; i++) {
            for (int j = 0; j < 5; j++) {
                list.get(i).add(in.nextInt());
```

```
            }
        }
        System.out.println(list);
    }

}
```

Examples

▼ Swap the elements in Array

```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Arrays;

public class SwapEleInArr {
    public static void main(String[] args) {
        int[] arr={1, 2, 23, 9, 18,};

        swap(arr, 1, 2);
        System.out.println(Arrays.toString(arr));
    }

    static void swap(int[] arr, int index1, int index2){
        int temp=arr[index1];
        arr[index1]=arr[index2];
        arr[index2]=temp;
    }

}
```

https://chatgpt.com/share/66fd4fe0-949c-800f-9ba9-24d1fef328f7

▼ Max value in Array

```
import java.util.Scanner;

public class MaxValInArray {
    public static void main(String[] args) {
        int[] arr={1, 45, 78, 96, 35};
        int max=arr[0];
        for (int i =0; i < arr.length; i++) {
            if(arr[i]>max){
                max=arr[i];
            }
        }
        System.out.println(max);
    }

}
```

▼ Max value in the range of an Array

- Conditions in the FOR loop changes acc to the range

▼ Reversing an Array

```
import java.util.Arrays;

public class ReverseArray {
    public static void main(String[] args) {
        int[] arr = {1, 2, 23, 9, 18};
        System.out.println("Original array: " + Arrays.toS
        arrRev(arr);  // Pass the array to the method
        System.out.println("Reversed array: " + Arrays.toS
    }

    static void swap(int[] arr, int index1, int index2) {
        int temp = arr[index1];
        arr[index1] = arr[index2];
        arr[index2] = temp;
    }

    static void arrRev(int[] arr) {  // Add array paramete
```

```
        int start = 0;  // Use index for start
        int end = arr.length - 1;  // Use index for end

        // Swap elements from start to end until the array
        while (start < end) {
            swap(arr, start, end);
            start++;
            end--;
        }
    }
}
```