**ARRAYS**

**What is Array?**

An array refers to a data structure that contains homogeneous elements(elements of same datatype)

**Features of Array:**

* Dynamic allocation
* Elements stored under a single name
* Occupies contiguous location

**Advantages of Array:**

It enables us to access any element randomly with the help of index.

Easy to store and manipulate large data sets

**Disadvantages of Array:**

Fixed size-array size cannot be increased or decreased once it is declared.

It cannot store heterogeneous(elements of different data type) data.

**Declaring an Array:**

Syntax: Datatype[] arrayname;

**Creating an Array:**

Syntax:

Datatype[] arrayname = new datatype[arraysize];

(Or)

Datatype[] arrayname = {value0,value1,….valuek};

Note: Assigning different type of data to an array leads to Syntax error

Assigning values to Array more than length of array leads to Runtime time error

Accessing array elements beyond specified length leads to ArrayIndexOutOfBound Exception

**Types of Array:**

* Single dimensional array
* Multi dimensional array

**Syntax to create Multi dimensional array**

Datatype[][] arrayname = new datatype[row\_size][column\_size];

**Operations on Array:**

Consider, int[] demo = {1,2,3,4,5};

**To find Array length:**

Syntax: arrayname.length;

**Print an array using for loop:**

Syntax: for(int i=0;i<demo.length-1;i++)

Sytem.out.println(demo[i]);

**Print an array using for-each loop:**

Synatx:

for(int refvar : demo)

Sytem.out.println(refvar);

**Copy an array to string:**

Syntax:

String str = Arrays.toString(arrayname)

Example:

String[] strarr ={“god”, “is”, “great”};

String strvar = Arrays.toString(strarr); #[god, is, great]

**Check if array contains certain value:**

Syntax:

Boolean val = Arrays.asList(arrayname).contains(“valuename”);

**Copy an array to another array**

Syntax:

Datatype[] newarr\_name = Arrays.copyOf(oldarrname, arr\_len);

(Note: It is always good practice to convert an array to string and then proceed)

Program:

Int[] arr1 = {10,20,30,40};

String strarr = Arrays.toString(arr1);

System.out.println(strarr);

Int[] arr2 = Arrays.copyOf(arr1,4);

String strarr2 = Arrays.toString(arr2);

System.out.printn(strarr2);

Sort arrays in java:

Syntax: Arrays.sort(arrayname)

Program:

Int[] arr1 = {10,20,30,40};

Arrays.toString(arr1);

Arrays.sort(arr1);

System.out.println(Arrays.toString(arr1));

Reverse an Array:

Hint: Convert an array to list-- Arrays.asList(arrayname)

* Convert an array to list
* use the reverse method in Collections interface

**static** **void** reverse(Integer myArray[])

    {

        Collections.reverse(Arrays.asList(myArray));

        System.out.println("Reversed Array:" + Arrays.asList(myArray));

    }

**public** **static** **void** main(String[] args)

    {

        Integer [] myArray = {1,3,5,7,9};

        System.out.println("Original Array:" + Arrays.asList(myArray));

        reverse(myArray);

    }

How to return an array

**static** **int**[] addsub(**int** val1,**int** val2)

{

**int**[] result = **new** **int**[2];

result[0] = val1+val2;

result[1] = val1-val2;

**return** result;

}

**public** **static** **void** main(String[] args)

{

**int**[] result = *addsub*(10,20);

System.***out***.println(result[0]);

System.***out***.println(result[1]);

}

}

**How do you Reverse an Array in Java?**

* Using a for loop to traverse the array and copy the elements in another array in reverse order.
* Using the reverse method of the Collections interface that works on lists.

**How do you Reverse a List in Java?**

* You can use the reverse method provided by the Collections interface of Java.

**ARRAY LIST**

**WHY ARRAY LIST?**

* We need to declare the size of an array before we can use it. Once the size of an array is declared, it's hard to change it.
* To handle this issue, we can use the **Array List** class. It allows us to create resizable arrays.
* Unlike arrays, **Array List** can automatically adjust their capacity when we add or remove elements from them. Hence, **Array List** are also known as **dynamic arrays**.

**CREATING AN ARRAY LIST**

**Syntax: ArrayList<datatype> arrylistname = new ArrayList<>();**

**<>refers to Generics.**

**Here, datatype used must be its corresponding wrapper class.**

**Operations in ArrayList**

**Add Elements**

To add elements of the arraylist, we use the add() method of the ArrayList class

**add(class ele);**

**add(int index, class ele);**

//Create ArrayList

ArrayList<String> languages = new ArrayList<>();

// add() method is used to add Elements

languages.add("Java");

languages.add("C");

languages.add("Python");

System.out.println(languages);

Output: [Java, C, Python]

// add JavaScript at index 1

languages.add(1, "JavaScript");

**Access Elements**

To access elements of the arraylist, we use the get() method of the ArrayList class

**get(int index);**

//Create ArrayList

ArrayList<String> languages = new ArrayList<>();

// get() method is used to access Elements

languages.get(1);

languages.get(2);

**Change Elements**

To change elements of the arraylist, we use the set() method of the ArrayList class

**set(int index, class ele);**

// set() method is used to change Elements

languages.set(2, "JavaScript");

**Remove Elements**

To remove an element from the arraylist, we can use the remove() method of the ArrayList class.

remove(int index);

remove(Class ele);

// remove() method is used to remove Elements

languages.remove(2);

**Remove All Elements**

To remove all elements from the arraylist, we can use the clear() method of the ArrayList class.

//Syntax:

arraylist.clear()

// clear() method is used to remove all Elements

languages.clear();

OTHER COMMON METHODS USED:

|  |  |
| --- | --- |
| Methods | Descriptions |
| [size()](https://www.programiz.com/java-programming/library/arraylist/size) | Returns the length of the arraylist. |
| [sort()](https://www.programiz.com/java-programming/library/arraylist/sort) | Sort the arraylist elements. |
| [contains()](https://www.programiz.com/java-programming/library/arraylist/contains) | Searches the arraylist for the specified element and returns a boolean result. |
| [isEmpty()](https://www.programiz.com/java-programming/library/arraylist/isempty) | Checks if the arraylist is empty. |
| [indexOf()](https://www.programiz.com/java-programming/library/arraylist/indexof) | Searches a specified element in an arraylist and returns the index of the element. |

## Iterate through an ArrayList

// creating an array list

ArrayList<String> animals = new ArrayList<>();

animals.add("Cow");

animals.add("Cat");

animals.add("Dog");

System.out.println("ArrayList: " + animals);

// iterate using for-each loop

System.out.println("Accessing individual elements: ");

for (String language : animals) {

System.out.print(language);

## To convert ArrayList into Array

public static void main(String[] args) {

ArrayList<String> languages = new ArrayList<>();

// add elements in the array list

languages.add("Java");

languages.add("Python");

// create a new array of String type

String[] arr = new String[languages.size()];

// convert ArrayList into an array

languages.toArray(arr);

// access elements of the array

for (String item : arr) {

System.out.print(item + ", ");

## To convert Array into ArrayList

To convert array into arraylist, we can use the Arrays.asList() method of the Arrays class.

String[] arr = { "Java", "Python", "C++" };

// create an ArrayList from an array

ArrayList<String> languages = new ArrayList<>(Arrays.asList(arr));

System.out.println(languages);

## To convert ArrayList into String

To convert ArrayList into String, we can use the toString() method of the ArrayList class.

ArrayList<String> languages = new ArrayList<>();

// add elements in the ArrayList

languages.add("Java");

languages.add("Python");

// convert ArrayList into a String

String str = languages.toString();

System.out.println(str);

## To create ArrayList using List Interface

## It’s because ArrayList class implements List Interface

// create arraylist using List

List<String> languages = new ArrayList<>();

languages.add("Java");

languages.add("Python");

System.out.println(languages);

### **How to Copy Specific Array Values into String**

**public** **static** **void** **main(**String**[]** args**)**

**{**

Char[] ch = **{**'S','o','f','t','w','a','r','e','T','e','s','t','i','n','g','B','l','o','g'**}**;

//We can copy a char array to a string by using \* copyValueOf() method.

String chStr = String.copyValueOf**(**ch**)**;

System.out.println**(**chStr**)**;

//We can also copy only range of charactors in a \* char array by copyValueOf() method

String subStr = String.copyValueOf**(**ch,8,7**)**;

System.out.println**(**subStr**)**;

**}**

**Output:**

SoftwareTestingBlog

Testing

**Difference between List and Set:**

| **List** | **Set** |
| --- | --- |
| Implements List interface. | Implements Set interface. |
| Contains a Legacy class, Vector. | No legacy classes. |
| ArrayList, LinkedList is List Interface implementations. | HashSet, TreeSet, LinkedHashSet are Set implementations. |
| An ordered sequence of elements. | An unordered collection of distinct elements. |
| Allows duplicates. | No duplicates are allowed. |
| Able to access elements as per the position of the element. | No positional access. |
| Null values are allowed. | Only one null value allowed. |
| New methods defined in a List interface. | No new methods defined in the Set interface. Collection interface methods are to be used with Set subclasses. |
| Can be traversed in forward and backward direction using ListIterator. | It can be traversed only in the forward direction with Iterator. |

**When not to use List?**

## When you keep adding the items in between

## When you keep removing the items in between

## DEFAULT VALUES OF VARIOUS ARRAY TYPES:

## String array: null

## Integer array: 0

## Double array:0.0

## Boolean array:false

## Reference array:null