Arrays

Arrays

⇒ In JAVA arrays are static in size. The size cannot be added or removed during run time.

Syntax

```
int[] arr = new int[5]
arr[0] = 10; // to insert value 10.
```

By using Arrays. ____ function we can perform some operations in JAVA.

① Arrays.toString(arr); // Print the whole array ([1,2,3])

② Arrays.sort(arr); // Sort the array.

③ Arrays.asList(arr); // Convert from array of fixed size to a list such as ArrayList

eg: ArrayList<Integer> list = Arrays.asList(arr);

NOTE

This works only if we declare int[] arr as Integer[] arr (ie wrapper class)

④ Arrays.binarySearch(arr, key) ⇒ Here the datatype can be any primitive type

key ⇒ value to be searched

⑤ To perform binary search first sort the array Arrays.sort(arr);

int arr-1 = {1, 2, 3}

int arr-2 = {1, 2, 3}

[arr1 == arr2] => (always returns false)

⑤ Arrays.equals(arr-1, arr-2) → eg: {1, 2, 3} {1, 2, 3} } true

{1, 2, 3} {1, 2, 3, 4} } false

{1, 2, 3} {3, 2, 1} } false.

⑥ Arrays.compare(arr-1, arr-2)

//returns 0, -1, 1

-1 → less than 0

1 → greater than 0.

if 0 => both array are equal.

if 1 => arr1 is lexographically greater than arr2

if -1 => arr1 is lexographically less than arr2

[Lexographically => dictionary order or alphabetical order.

Notes for array

① Initialing all the value to zero:

=> int[] arr = new int[5];

Internally all values are initialized to zero

② Other value fill:

=> Arrays.fill(arr, 0, 3, 1) (o/p) [1, 1, 1, 0, 0]

0 → start index, 3 → end+1 index, 1 → value

② Collections

① `ArrayList<Integer> list = new ArrayList<>();`

we can mention the size of arraylist also

Some important methods

① `list.add(value);`
② `list.add(index, value);`
③ `list.remove(index);`
④ `list.set(index, value);` // change the value at the index to the new value
⑤ `list.get(index);`
⑥ `list.size();`

---

② `HashSet<Integer> set = new HashSet<>();`

Some important methods: (unique values)

① `set.add(value);`
② `Set.clear();`
③ `Set.isEmpty()` // if no element returns true
④ `Iterator<Integer> it = Set.iterator();`

```
while(i.hasNext()){
  System.out.println(i.next());
}
```

⑤ `set.size()`
⑥ `set.remove(value);`

3) HashMap <Integer, Integer> map = new HashMap<>();

Some methods in HashMap:

1. map.Put (Key, Value);
2. map.get (Key);
3. map.isEmpty();
4. map.Size();
5. map.clear();
6. map.remove(Key); — removes the value
7. (map.ContainsKey (Key)) => returns boolean
8. (map.ContainsValue(value)) => returns boolean based on the value
9. map.keySet() => gets all key Set in form of array
10. map.values() => gets all values in form of array.

---

4) TreeMap <Integer, Integer> tree-map = new TreeMap<>();

(TreeMap is nothing but a Sorted map)

⇓

Sorting is based on keys

```
for (int i : map.KeySet()) {
    System.out.Println (i); // Key
    System.out.Println (map.get(i)); // Values
}
```

Collection methods

=> Same like Arrays.__( ) we have collections also some methods. Some of them are ↓

① Collections.sort (___);

② Collections.rotate (___, value); it moves the last elements to front till the value.

③ Collections.reverse (___);

④ Collections.binarySearch (___, "value"); //returns the index of the value.

⑤ Collections.copy (___ copied to ___);