Array

```java
// Basic initialization
int[] intArray = new int[20];
int[] intArray = new int[]{ 1,2,3,4,5,6,7,8,9,10 };

//Two dimensional array:
int[][] twoD_arr = new int[10][10];

// fill the array
Arrays.fill(arrName, 10);

// Sort
Arrays.sort(arr);
Arrays.sort(arr, Collections.reverseOrder());

// reverse the array
Collections.reverse(Arrays.asList(a));
// reverse array using string builder
String[] arr = {"Hello", "World"};
StringBuilder reversed = new StringBuilder();
for (int i = arr.length; i > 0; i--) {
    reversed.append(arr[i - 1]).append(" ");
};
String[] reversedArray = reversed.toString().split(" ");
```

Array List

```java
// Basic initialization
List<Integer> list= new ArrayList<>();
List<Integer> numbers= new ArrayList<Integer>(Arrays.asList(60, 25, 12));
// Or
Integer[] arr = {2,3,4,4};
List<Integer> list = new ArrayList<Integer>(Arrays.asList(arr));

// Two dimentional list
List<List<String>> countries = new ArrayList<>();
for(int i = 0; i < 100; i++){
  countries.add(new ArrayList<>());
}
// Or
```

```
List<String>[] countries = new ArrayList[100];
for(int i = 0; i < 100; i++){
  list[i] = new ArrayList<>();
}

// Desfine the array size
list.size();

// Add new item
list.add(10);
list.add(20);
list.add(int index, E element) // insert data in specific position
list.addAll(numbers);

// Get new item
list.get(int index)

// Remove item by index
list.remove(int index)
list.clear() // clear all data;

list.contains(120)); // returns true/false
list.indexOf(Object o) // returns first index of finding or -1
list.isEmpty() // returns true/false

list.toString(); // make the array string
list.toArray() // Returns an array containing all of the elements

// Sorting and changed the data source for both cases
list.sort(Comparator.naturalOrder()); // ASC
list.sort(Comparator.reverseOrder()); // DSC
// Or
Collections.sort(list); // ASC
Collections.sort(list, Collections.reverseOrder()); // DSC
```

LinkedList

```
// Basic initialization
LinkedList<String> linkedList= new LinkedList<String>();

// Desfine the array size
linkedList.size();
```

```
// Add item
linkedList.add() // appends the value in the linked list
linkedList.add(int index, E element) // Inserts element at specified index
linkedList.offer(E e) // Adds the specified in the end of the list
linkedList. offerFirst(E e)
linkedList.offerLast(E e)

// Get item
linkedList.get(int index)
linkedList.getFirst() // Returns the first element in this list.
linkedList.getLast() // Returns the last element in this list.
linkedList.peek() // Retrieves, does not remove, first element of list.
linkedList.peekFirst()
linkedList.peekLast()

// Remove item
linkedList.remove() // Retrieves and removes the head of this list.
linkedList.remove(int index) // Removes item at specified position in list.
linkedList.removeFirst()
linkedList.removeLast()
linkedList.poll() // Retrieves and removes the head of this list.
linkedList.pollFirst()
linkedList.pollLast()


linkedList.size();
linkedList.toArray();
```

Stack

```
// Basic initialize
Stack<Integer> stack = new Stack<>();

stack.isEmpty() // returns true/false

stack.peek() // returns the top value but will not remove

stack.pop() // retruns the top and remove it from the stack

stack.push()

stack.size() // returns the size of the stack
stack.toArray()
stack.toString()
```

Queue

```
// Basic initialization
Queue<Integer> queue = new LinkedList<>();

queue.isEmpty() // returns true/false

queue.peek() // returns the first element of the queue and don't remove

// Add item
queue.offer(10); // add end of the queue. don't throw exception
queue.add(10);   // add end of the queue throw exception

// Remove
queue.poll() // return the first element and remove the first of the queue
queue.remove() // same as poll but capable to throws an exception

queue.size() // returns the size of the queue

queue.toArray()
```

Priority Queue

```
// Basic initialization
Queue<Integer> pQueue = new PriorityQueu<>() // By default ASC
// DSC
PriorityQueue<Integer> pQueue =
      new PriorityQueue<Integer>(Collections.reverseOrder());
PriorityQueue<Integer> pQueue =
      new PriorityQueue<Integer>((a, b) -> b - a);


pQueue.isEmpty() // returns true/false

pQueue.peek() // returns the first element of the queue and don't remove

// Add item
pQueue.offer(10); // add end of the priority queue. don't throw exception
pQueue.add(10);   // add end of the priority queue throw exception

// Remove
pQueue.poll() // return first item and remove first item of priority queue
pQueue.remove() // same as poll but capable to throws an exception

pQueue.size() // returns the size of the queue

pQueue.toArray()
```

Hash Map

```
// Basic initialization
Map<K, V> map = new HashMap<>();
Map<Integer, List<String>> map = new HashMap<>();

map.size()
map.isEmpty()
map.containsKey(Object key) // Returns true if this map contains key.

// Get item
map.get(Object key)
map.getOrDefault(Object key, V defaultValue);

// Put item
map.put(K key, V value)

// Remove
map.remove(Object key)

// Looping through the HashMap
for (Map.Entry<String,Integer> element : map.entrySet()) {
    String key = element.getKey();
    int value = element.getValue();
}



// List of keys
map.keySet()
List<String> list = new ArrayList<>(map.keySet()); // make list

// List of values
map.values()
List<Value> list = new ArrayList<Value>(map.values());

// how to check the key is available or not
if(map.get(key) != null){
  // to do
}

// array of map
Map<String, String>[] map = new HashMap[26];
for(int i = 0; i < 26; i++){
  map[i] = new HashMap<>();
}
for(int i = 0; i < 26; i++){
  for(int j = 0; j<10; j++){
    map[i].put("key"+j, "value"+j);
  }
}
```

<u>**Tree Map**</u>

```
// Basic initialization
Map<String, Integer> treeMap = new TreeMap<>(); // by default ASC
Map<K, V> treeMap = new TreeMap<>(Collections.reverseOrder());

treeMap.get(key)
treeMap.getOrDefault(Object key, V defaultValue);

// Put item
treeMap.put(key, vlaue)
```

```
// Remove
treeMap.remove(Object key)

// Entry access
Map.Entry<K,V> entry = treeMap.firstEntry();


Map.Entry<K,V> entry = treeMap.lastEntry();

// Looping through the HashMap
for (Map.Entry<String,Integer> element : treeMap.entrySet()) {
    String key = element.getKey();
    int value = element.getValue();
}

// List of keys
treeMap.keySet()
List<String> list = new ArrayList<>(treeMap.keySet()); // makes list

// List of valu
treeMap.values()
List<Value> list = new ArrayList<Value>(treeMap.values());
```

Hash Set

```
// Basic Initialization
Set<String> set= new HashSet<>();

set.size();
set.add()
set.contains()
set.remove(Object o) // Removes the element from this set if it's present.

set.clear();
set.isEmpty();

// Looping through
for (String s : set) {
    System.out.println(s);
}
```

Tree Set

```
Set<Integer> treeSet = new TreeSet<Integer>() // default ASC
// reverse order
TreeSet<Integer> treeSet = new TreeSet<Integer>(
  new Comparator<Integer>(){
      public int compare(Integer i1,Integer i2){
        return i2.compareTo(i1);
      }
  });


treeSet.size();
treeSet.add()
treeSet.contains()
treeSet.remove(Object o) //Removes the element from this set if present.

treeSet.clear();
treeSet.isEmpty();

treeSet.first() // Returns the first element currently in this set.
treeSet.last()

// Retrieves and removes the first element, or
// returns null if this set is empty
treeSet.pollFirst()
treeSet.pollLast()
```

String & StringBuilder

```
// Basic initialization
StringBuilder sb = new StringBuilder();
StringBuilder sb = new StringBuilder("Hello world"); // accept string

sb.length()
sb.reverse();
sb.toString();

// Add
sb.append('c'); // adding char
sb.append("Stirng") // adding string
sb.append(int num) // adding number, float

sb.charAt(int index) // Returns the char value at that index
sb.setCharAt(index, 'a'); // set the char
sb.indexOf(String str) // return index or -1
sb.toString().equals(sbGoal.toString()); // equals

//split
String str = sb.toString();
String strArr[] = str.split(":");

//Int to String
int i=200;
String s=String.valueOf(i);

// string to int
String s="200";
int value = Integer.parseInt(s);

// Char array to string;
char ch_arr[] = {'H','e','l','l','o',' ','W','o','r','l','d','!'};
String str1 = new String(ch_arr);
System.out.println("String: "+ str1);

// char array to string builder
char ch_arr[] = {'J','a','v','a',' ','P','r','o','g','r','a','m'};
StringBuilder sb_obj = new StringBuilder();
for(int i = 0; i < ch_arr.length; i++){
    sb_obj.append(ch_arr[i]);
}


int[] frequency = new int[26];
for (char ch : s.toCharArray()) {
    frequency[ch - 'a'] += 1;
    if (frequency[ch - 'a'] == 2) {
        return true;
    }
}
```