Basic idea to understand :
Map -> V get(k1)

- get : method,
- k1 : accept key as an value,
- V : return datatype

1. Map // collection
1. Definition ->Map<K, V> map = new HashMap<>();
2. insert / update -> V put(k1, v1); // TC: O(1)
3. delete -> V remove(k1); // TC: O(1)
4. get -> V get(k1); // TC: O(1)
5. size -> int size(); // TC: O(1)
6. check for Empty -> boolean isEmpty(); // TC: O(1)
7. value present -> boolean containsKey(k1); // TC: O(1)
8. remove all map values -> clear(); // TC: O(2n + 1) -> O(n) *(n-key, n-value, 1 for map itself)*

2. ArrayList // Collection
1. Definition -> ArrayList list = new ArrayList<>();
2. insert -> boolean add(t) **[TC: O(1)]** / add(int index, T) **[TC: O(n)]**
3. delete -> T remove(int index); // TC: O(n) as you have to shuffle the elements above that point
4. set/update index value -> T set(int index, T); // TC: O(1)
5. get index-> T get(int index); // TC: O(1)
6. size -> int list.size(); // TC: O(1)
7. clear elements -> void clear(); // TC: O(n) & removeAll : O(n^2).
8. check for Empty -> boolean isEmpty(); // TC: O(1)
9. value contain check -> boolean contains(t); // TC: O(n)
10. get Index of value -> int indexOf(t); // TC: O(n), checking each element one by one
11. non premitive to premitive list -> toArray(); // TC: O(n)
12. Sorting for List ->

- Collections.sort(list, (a, b) -> a - b); // ascending , TC: O(nlogn)
- Collections.sort(list, (a, b) -> b - a); // descnding , TC: O(nlogn)

3. Array
1. Definition ->T arr [ ]= new T[N]; // N: static size , T : datatype
2. insert -> arr[index] = v1; // TC: O(1)
3. update -> arr[index] = v2; // TC: O(1)
4. get -> T arr[index] // TC: O(1)
5. size -> int arr.length // TC: O(1)

6. Arrays.fill(arr, 0); // filled array with value=0, TC: O(n)
7. Sorting -> TC: O(nlogn)

- premitive (int[] ..)
  - Arrays.sort(arr); // default ascending,
- non-premetive (Integer[] ..)
  - Arrays.sort(arr); // default ascending
  - Arrays.sort(arr, (a,b) -> b-a); // descening

4. Stack // Collection