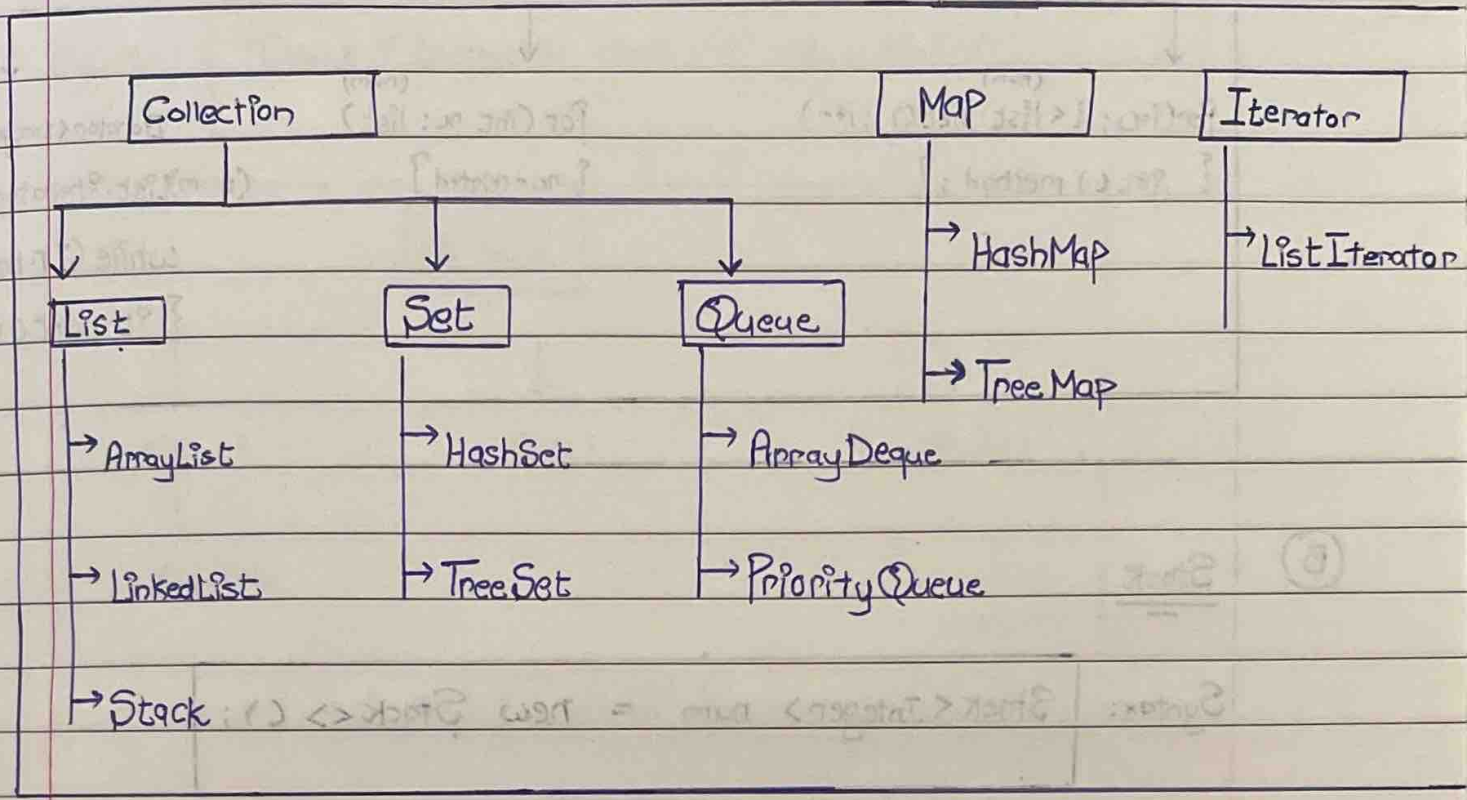
Date
Page

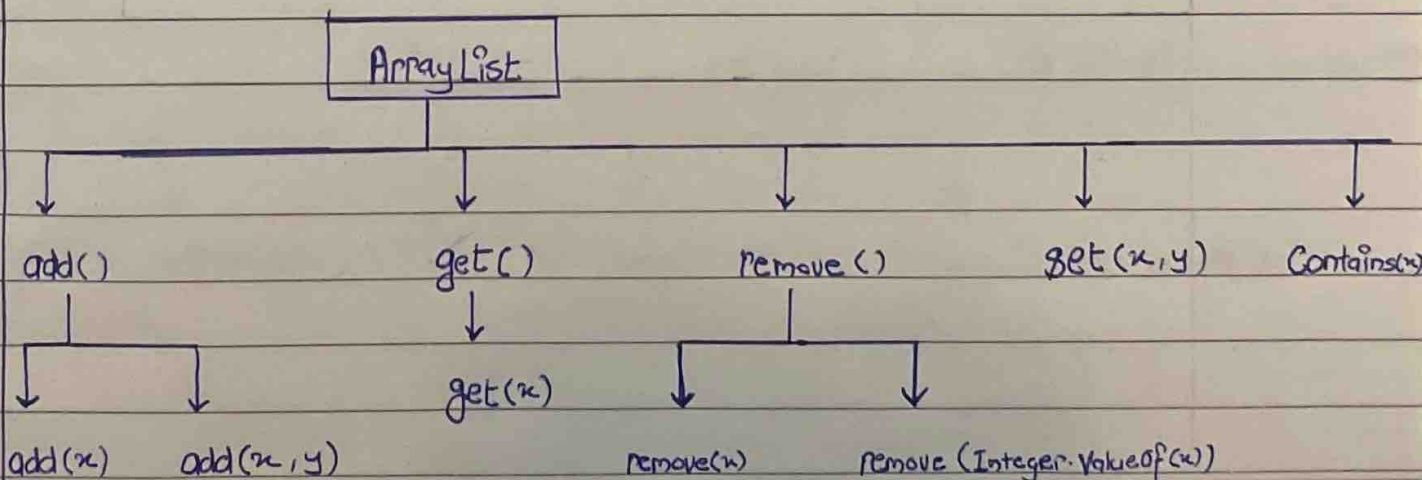
Java Collection Framework : It provides a set of interfaces & classes to implement various data structures and algorithms.

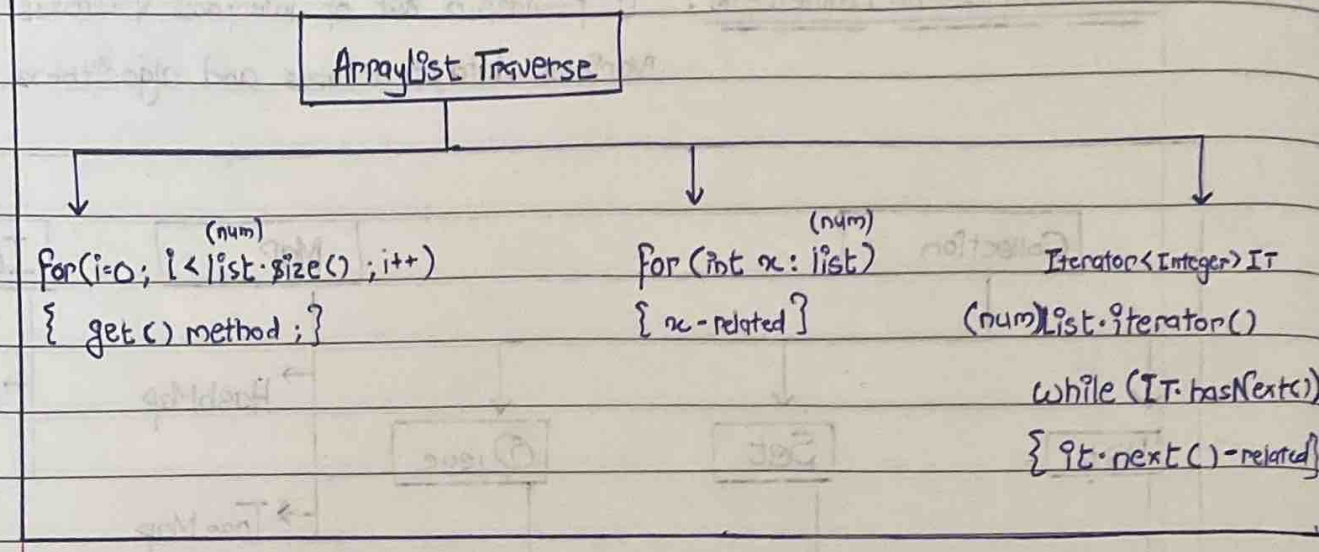


Collection FrameWork.

(A) ArrayList

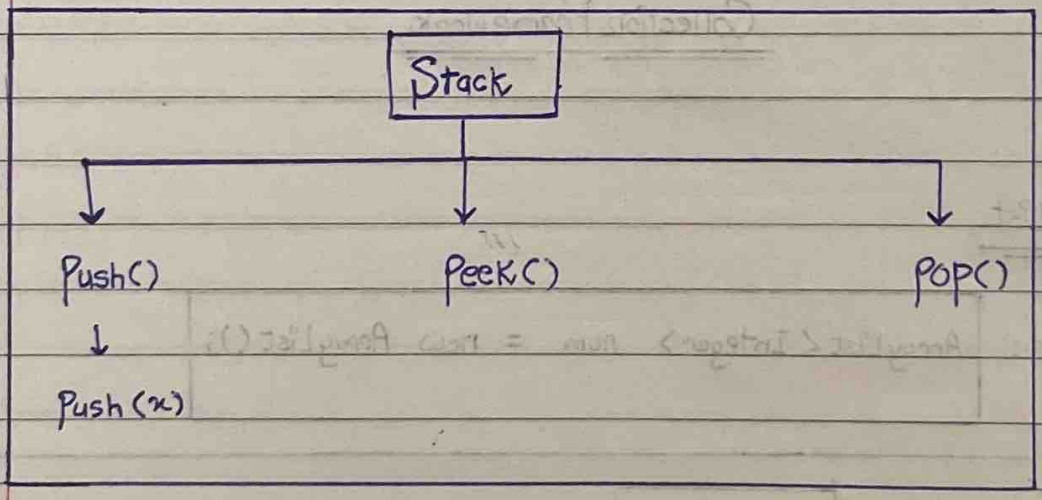
Syntax : `ArrayList<Integer> num = new ArrayList();`





(B) Stack

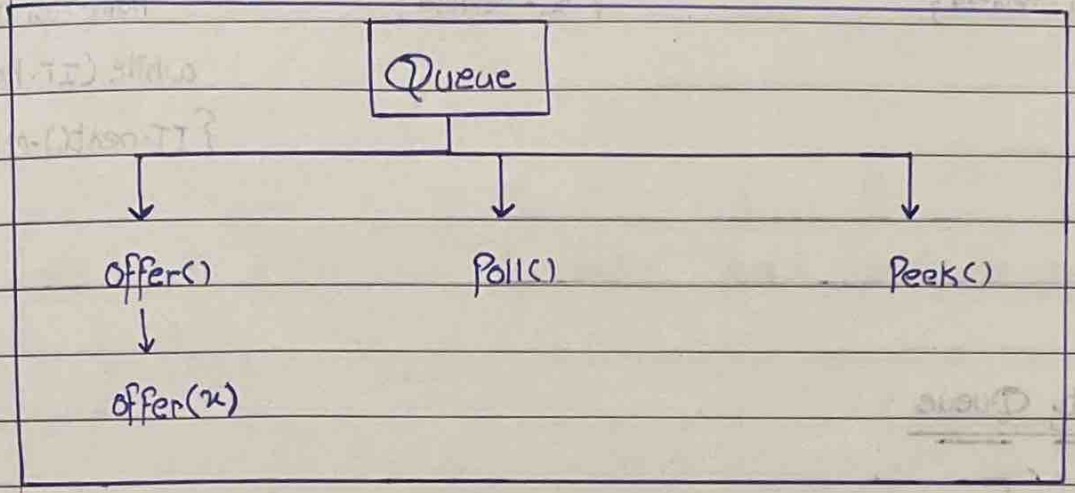
Syntax: `Stack<Integer> num = new Stack<>();`



Date
Page

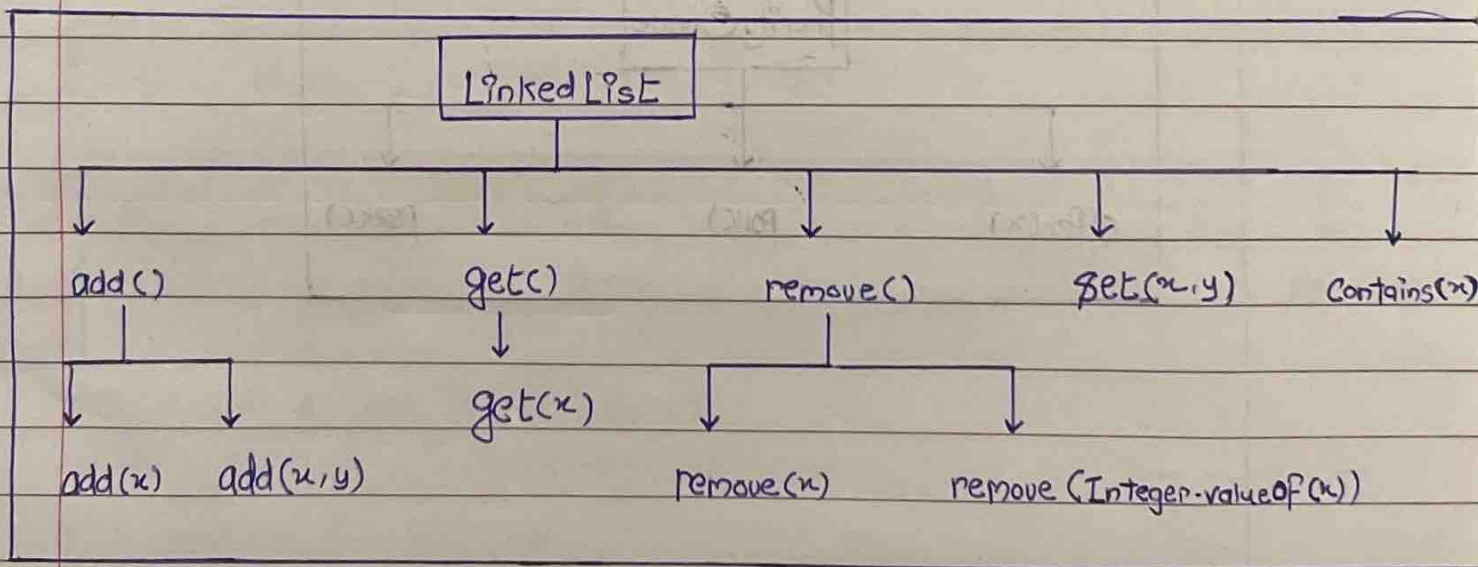
ⓒ Queue [Implements LinkedList]

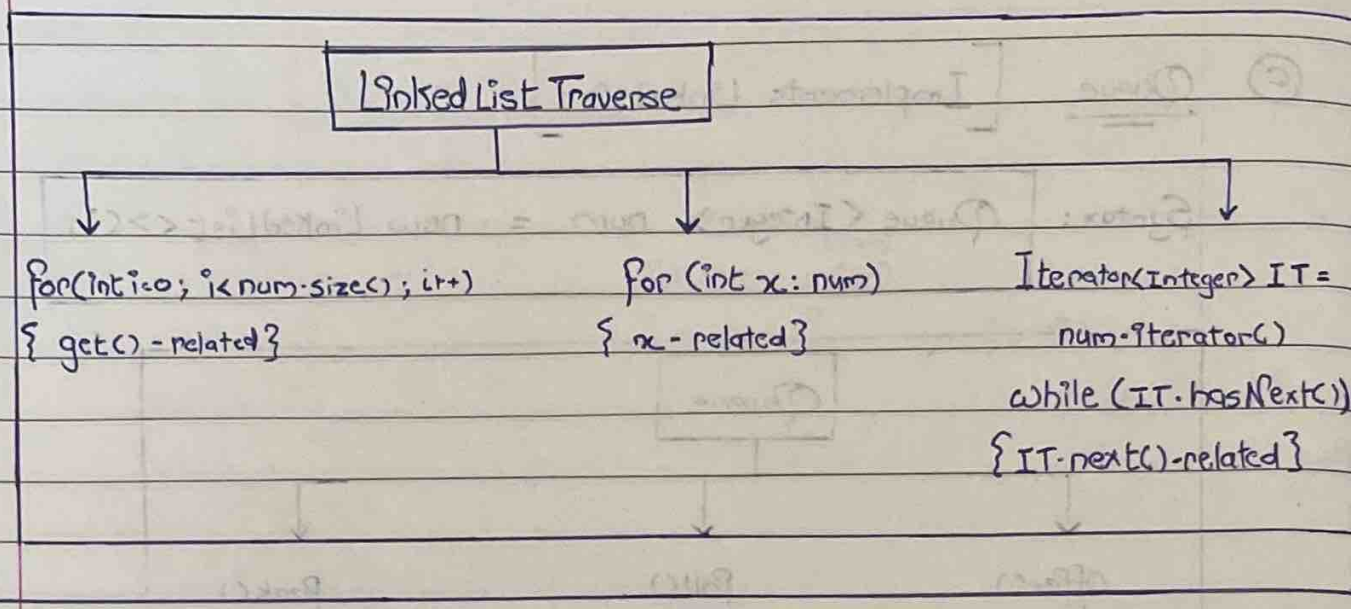
Syntax: `Queue <Integer> num = new LinkedList<>();`



ⓓ LinkedList

Syntax: `List <Integer> num = new LinkedList<>();`



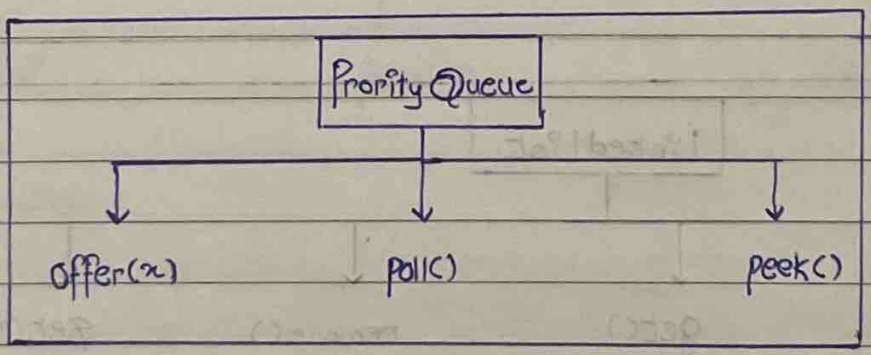


(E) **Priority Queue**

Syntax : `Queue<Integer> pq = new PriorityQueue<>();` (Min Heap)

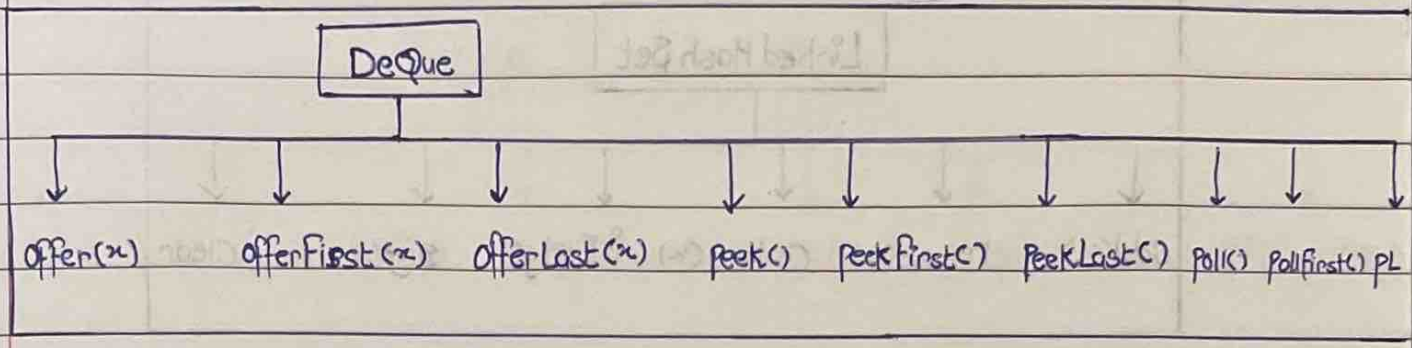
OR

(Max Heap) `Queue<Integer> pq = new PriorityQueue<>(Comparator.reverseOrder());`



(F) **Deque (Double Ended Queue):**

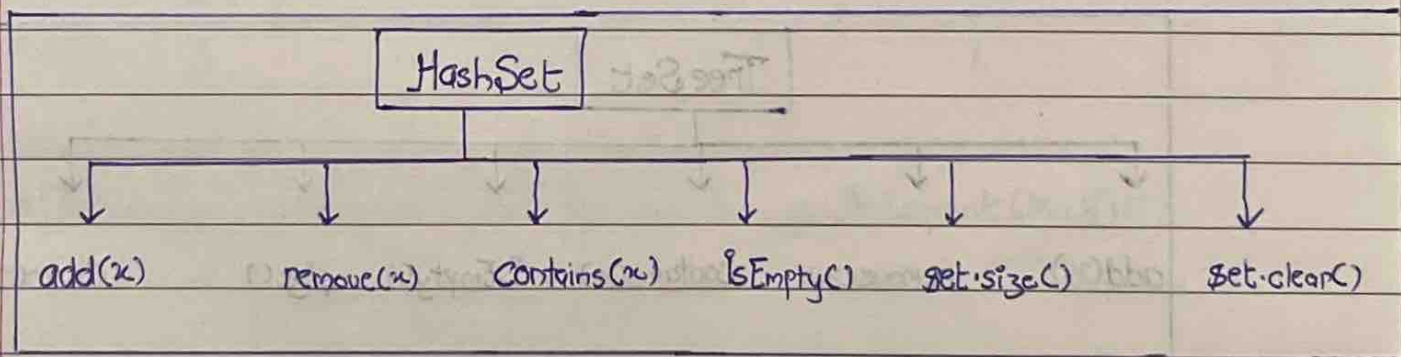
Syntax: `ArrayDeque<Integer> adq = new ArrayDeque<>();`



(G) **HashSet**

Syntax: `Set<Integer> set = new HashSet<>();`

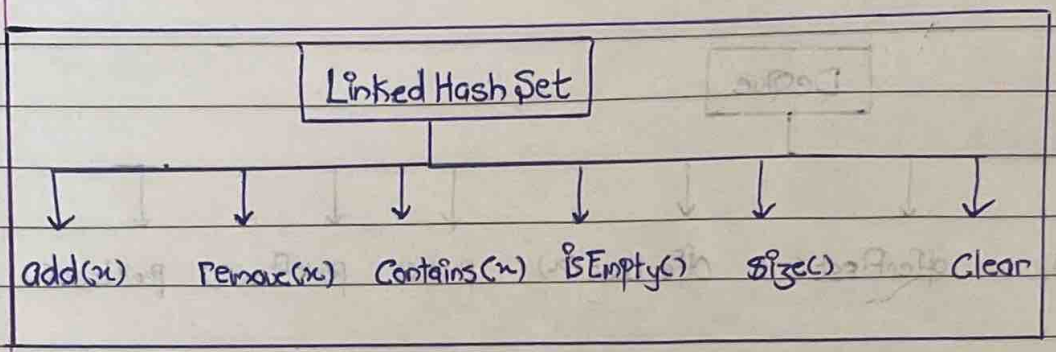
- No Duplicates allowed.
- Order of elements inside is random.



* Built in functions in O(1)

(H) LinkedHashSet:

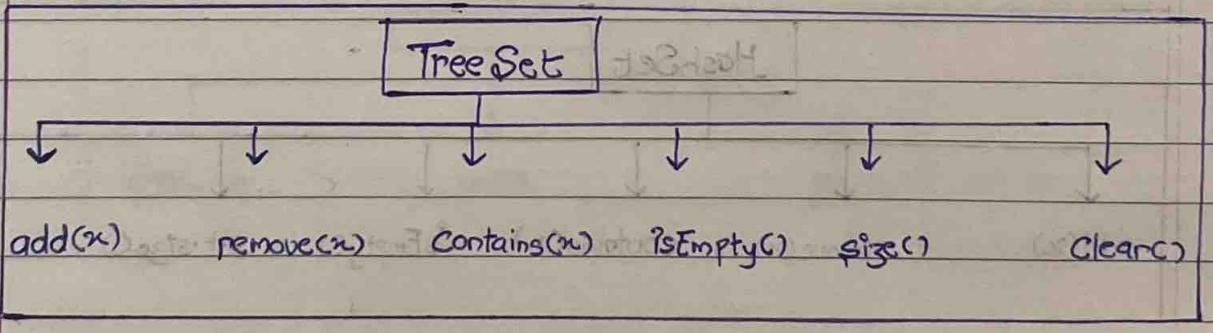
Syntax: `Set<Integer> set = new LinkedHashSet<>();`



* No Duplicates
* Order same as user putting elements.
* Built-in Functions in O(1)

(I) TreeSet:

Syntax: `Set<Integer> set = new TreeSet<>();`

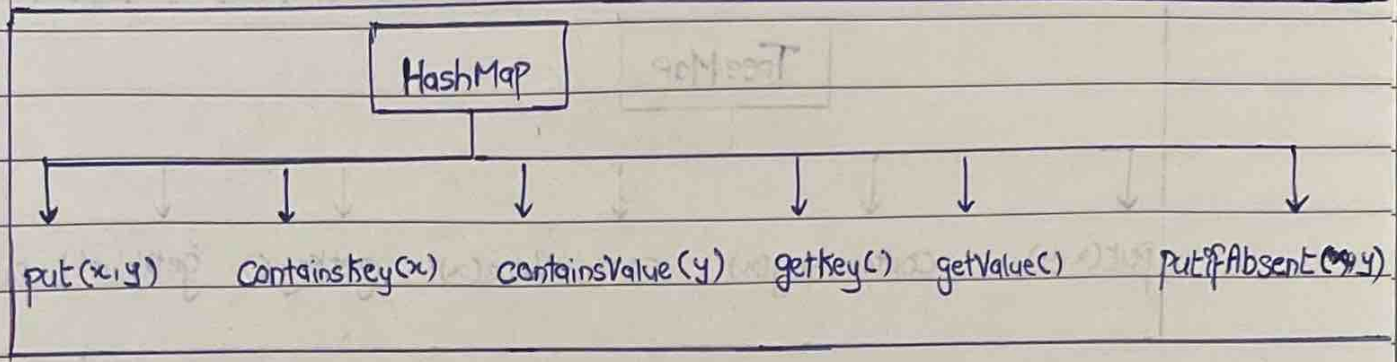


* No Duplicates.
* Elements in Sorted form.
* Built in function in O(log n) due to binary search.

Date
Page

(J) HashMap :

Syntax : `Map <String , Integer> numbers = new HashMap <>();`



HashMap Traverse

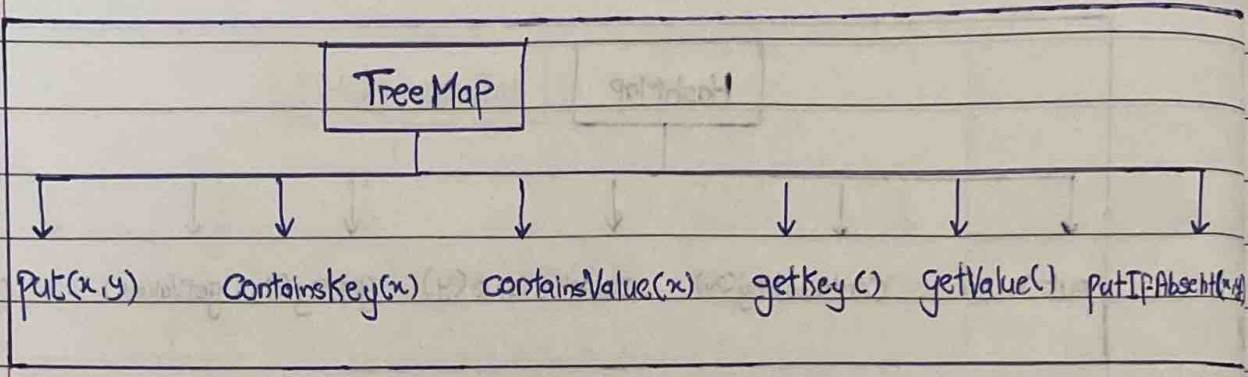
```
for (Map.Entry <String , Integer> e : numbers.entrySet())
{
    Sysout (e);
    Sysout (e.getKey());
    Sysout (e.getValue());
}
```

* To Remember additional builtin function ≡ getOrDefault(x,y)

* Order of elements is random
* Duplicates not allowed, but values can override

(K) TreeMap :

Syntax : `Map<String, Integer> numbers = new TreeMap<>();`



TreeMap Traverse

```
for(Map.Entry<String, Integer> e : numbers.entrySet())
{
  SysOut(e);
  Systout(e.getkey());
  Sysout(e.getValue());
}
```

* Elements are Sorted.
* So, built in functions in O(log N)

(L) Arrays Class

```
int[] arr = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int index = Arrays.binarySearch(arr, 4); // for binarySearch

int[] arr = {56, 23, 65, 8, 100};
Arrays.sort(arr); // to sort the array

Arrays.fill(arr, 0); // To make all elements of array to zero or any other value
```

(M) Collections Class

```
List<Integer> mylist = new ArrayList<>();
mylist.add(28);
mylist.add(36);
mylist.add(4);
mylist.add(2);
mylist.add(44);
mylist.add(1);
```

Collections.min(mylist) → gives Minimum Number in the list
Collections.max(mylist) → gives Maximum Number in the list
Collections.frequency(mylist, 44) → gives number of times element occurs
Collections.sort(mylist) → sorts in ascending order
Collections.sort(mylist, Comparator.reverseOrder()); → sort in descending order