**COLLECTION**

The collections framework in Java is a way to store and manage groups of objects (data) in an organized manner. It includes different types of structures (called collections) that you can use depending on your needs.

**1. Different types of container**

* Lists
* Sets
* Maps
* Queues
* Deques

**2. About  List**

 List is an ordered collection that allows duplicate elements and provides indexed access to its elements. Common implementations include ArrayList, which uses a resizable array for fast random access, and LinkedList, which uses a doubly-linked list for efficient insertions and deletions. Lists are versatile and used extensively for storing and managing ordered sequences of elements.

**EXAMPLE**

public class list {

    public static void main(String[] args) {

        List<Integer>  = new LinkedList<>();

      list.add(4);

      list.add(7);

      list.add(5);

      list.add(2);

      list.add(3);

      list.add(6);

      list.add(null);

      list.add(null);

        ListIterator<Integer> it = set.listIterator(set.size());

        while (it.hasPrevious()) {

            Integer value = it.previous();

            System.out.println(value);

        }

    }

}

**Functions**

* set() - To replace a existing value in a collection.
* get() - To get a specific value in a collection.
* indexOf() - To extract a particular index value (mentioned inside the parenthesis) in the collection.
* sort() - For sorting in ascending order .

**3. About Set**

           A Set is an unordered collection that does not allow duplicate elements. Common implementations include HashSet, which is backed by a hash table, and TreeSet, which is backed by a tree structure and maintains elements in sorted order.

**EXAMPLE**

Set<Integer> set = new HashSet<>();

set.add(4);

set.add(7);

 set.add(5);

 set.add(2);

set.add(3);

set.add(6);

set.add(7);

 for (Integer value : set)

{

System.out.println(value);

 }

**Functions**

* add(element) - To add an element to the set.
* remove(element) - To remove an element from the set.
* contains(element) - To check if an element is in the set.
* size() - To get the number of elements in the set.

**4. About Queue**

 A **Queue** is a collection used to hold multiple elements prior to processing, typically in FIFO (first-in-first-out) order. Common implementations include LinkedList and PriorityQueue.

**Example**

        Queue<Integer> number = new LinkedList();

        number.offer(1);

        number.offer(2);

        number.offer(3);

        System.out.println(number);

        int removeNumber = number.poll();

       System.out.println(removeNumber);

**Function**

* add(element) - To add an element to the queue.
* poll() - To retrieve and remove the head of the queue.
* peek() - To retrieve the head of the queue without removing it.
* isEmpty() - To check if the queue is empty.

**5. About Deque**

A Deque is a linear collection that supports element insertion and removal at both ends. It stands for "double-ended queue". Implementations like LinkedList  provide efficient operations for adding, removing, and inspecting elements.

**Functions:**

* addLast(element) - To add an element to the end of the deque.
* removeFirst() - To remove and retrieve the first element of the deque.
* removeLast() - To remove and retrieve the last element of the deque.

**7. About Map**

       Map is a collection that maps keys to values, with no duplicate keys allowed. Common implementations include HashMap, which is backed by a hash table, and TreeMap, which is backed by a tree structure and maintains keys in sorted order.

**Example**

 Map<Integer, String> studentMap = new HashMap<>();

        studentMap.put(101, "Senthil");

        studentMap.put(102, "Kumar");

        studentMap.put(103, "Cheran");

        studentMap.put(104, "Manoj");

        System.out.println("Size"+studentMap.size())

        if (studentMap.containsKey(102)) {

            System.out.println(" present in the map");

        } else {

            System.out.println("Not present in the map");

        }

        String studentName = studentMap.get(103);

        System.out.println( studentName);

        studentMap.remove(101);

**Function**

* size() - it is used to find the size of map elements
* isEmpty() - checks if the map is empty
* put(k,v) - To add elements to the map as key and value pair
* get(k) - To get the respective value of the given key
* remove(k) - To remove the key and value