Competitive Programming Java Data Structures

Arrays

Declaration Syntax: int[] arr = new int[10];

Notes: For fixed-size sequences

Methods & Explanations:

Arrays.sort(arr): Sort array

Arrays.binarySearch(arr, key): Binary search

LinkedList

Declaration Syntax: LinkedList<Integer> list = new LinkedList<>();

Notes: Doubly linked list

Methods & Explanations:

list.add(1): Adds to end

list.get(index): Get element

list.remove(index): Remove element

Queue

Declaration Syntax: Queue<Integer> queue = new LinkedList<>();

Notes: FIFO structure

Methods & Explanations:

queue.add(1): Add to queue

queue.poll(): Removes and returns head

queue.peek(): Retrieves but does not remove head

Stack

Declaration Syntax: Stack<Integer> stack = new Stack<>();

Notes: LIFO structure

Methods & Explanations:

stack.push(1): Push onto stack

stack.pop(): Pop from stack

stack.peek(): View top of stack

Deque

Declaration Syntax: Deque<Integer> deque = new LinkedList<>();

Notes: Double-ended queue

Methods & Explanations:

deque.addFirst(1): Add to front

deque.addLast(1): Add to back

deque.pollFirst(): Remove from front

PriorityQueue

Declaration Syntax: PriorityQueue<Integer> pq = new PriorityQueue<>();

Notes: Min-heap by default

Methods & Explanations:

pq.offer(1): Add element

pq.poll(): Remove and return smallest element

pq.peek(): Get smallest element

Set

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

Notes: No duplicates

Methods & Explanations:

set.add(1): Add element

set.contains(1): Check if element exists

set.remove(1): Remove element

SortedSet

Declaration Syntax: SortedSet<Integer> sset = new TreeSet<>();

Notes: Sorted set of unique elements

Methods & Explanations:

sset.add(1): Add element

sset.first(): Get smallest element

sset.last(): Get largest element

Tree

Declaration Syntax: class TreeNode { int val; TreeNode left, right; }

Notes: Binary tree structure

Methods & Explanations:

Inorder traversal, Preorder traversal, Postorder traversal

TreeSet

Declaration Syntax: TreeSet<Integer> tset = new TreeSet<>();

Notes: Implements NavigableSet

Methods & Explanations:

tset.add(1): Add element

tset.first(): Smallest element

tset.last(): Largest element

HashMap

Declaration Syntax: HashMap<Integer, String> map = new HashMap<>();

Notes: Key-value pairs, no order

Methods & Explanations:

map.put(1, 'A'): Insert key-value

map.get(1): Get value

map.containsKey(1): Check key existence

Hashtable

Declaration Syntax: Hashtable<Integer, String> ht = new Hashtable<>();

Notes: Synchronized, slower than HashMap

Methods & Explanations:

ht.put(1, 'A'): Insert key-value

ht.get(1): Get value

ht.containsKey(1): Check key existence

StringBuilder

Declaration Syntax: StringBuilder sb = new StringBuilder();

Notes: Mutable sequence of characters

Methods & Explanations:

sb.append('A'): Append string

sb.toString(): Convert to string

sb.reverse(): Reverse string

Comparator

Declaration Syntax: Comparator<Integer> comp = (a, b) -> a - b;

Notes: Used for custom sorting

Methods & Explanations:

compare(a, b): Compare two elements for sorting

Collections

Declaration Syntax: Collections

Notes: Utility class for common algorithms

Methods & Explanations:

Collections.sort(list): Sorts list

Collections.binarySearch(list, key): Binary search