Java Cheat Sheet for LeetCode & Competitive Programming

1. Arrays

- Declaration & Initialization:

  int[] arr = new int[5];

  int[] arr = {1, 2, 3, 4, 5};

- 2D Arrays:

  int[][] matrix = new int[3][3];

  int[][] matrix = {{1, 2}, {3, 4}, {5, 6}};

- Basic Operations:

  - Length: arr.length
  - Sort: Arrays.sort(arr);
  - Binary Search: int index = Arrays.binarySearch(arr, target);
  - Copying Arrays: int[] newArr = Arrays.copyOf(arr, arr.length);

2. ArrayList (Dynamic Array)

- Declaration:

  List<Integer> list = new ArrayList<>();

- Common Operations:

  - Add Element: list.add(5);
  - Remove Element: list.remove(index);
  - Get Element: list.get(index);
  - Size: list.size();

- Contains: list.contains(element);

3. Nodes for Trees, Graphs, and Linked Lists

- Binary Tree Node:

```
class TreeNode {
    int val;
    TreeNode left, right;
    TreeNode(int x) { val = x; }
}
```

- Graph Node:

```
class GraphNode {
    int val;
    List<GraphNode> neighbors;
    GraphNode(int x) { val = x; neighbors = new ArrayList<>(); }
}
```

- Linked List Node:

```
class ListNode {
    int val;
    ListNode next;
    ListNode(int x) { val = x; }
}
```

4. Custom Comparators

- Array Sorting with Comparator:

Arrays.sort(arr, (a, b) -> a - b);

- ArrayList Sorting:

list.sort((a, b) -> b - a); // Descending order

5. Stack, Queue, and Deque (Double-ended Queue)

- Stack:

Stack<Integer> stack = new Stack<>();

stack.push(5); stack.pop(); stack.peek();

- Queue (LinkedList):

Queue<Integer> queue = new LinkedList<>();

queue.add(5); queue.remove(); queue.peek();

- Deque:

Deque<Integer> deque = new ArrayDeque<>();

deque.addFirst(1); deque.addLast(2); deque.removeFirst();

6. HashMap and HashSet

- HashMap:

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

map.put(1, "One");

- HashSet:

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

set.add(5); set.contains(5);

7. Priority Queue (Min-Heap with Custom Comparator)

- Min-Heap (default):

  PriorityQueue<Integer> minHeap = new PriorityQueue<>();

- Custom Comparator for Priority Queue:

  PriorityQueue<Integer> customHeap = new PriorityQueue<>((a, b) -> b - a); // Max-Heap

8. Math Functions

- Absolute: Math.abs(x);
- Max/Min: Math.max(a, b); Math.min(a, b);