**Java数据结构**

Java 提供了丰富的数据结构来处理和组织数据。

Java 的 java.util 包中提供了许多这些数据结构的实现，可以根据需要选择合适的类。

以下是一些常见的 Java 数据结构：

数组（Arrays）

数组（Arrays）是一种基本的数据结构，可以存储固定大小的相同类型的元素。

int[] array = new int[5];

特点： 固定大小，存储相同类型的元素。

优点： 随机访问元素效率高。

缺点： 大小固定，插入和删除元素相对较慢。

列表（Lists）

Java 提供了多种列表实现，如 ArrayList 和 LinkedList。

List<String> arrayList = new ArrayList<>();

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

ArrayList:

特点： 动态数组，可变大小。

优点： 高效的随机访问和快速尾部插入。

缺点： 中间插入和删除相对较慢。

LinkedList:

特点： 双向链表，元素之间通过指针连接。

优点： 插入和删除元素高效，迭代器性能好。

缺点： 随机访问相对较慢。

集合（Sets）

集合（Sets）用于存储不重复的元素，常见的实现有 HashSet 和 TreeSet。

Set<String> hashSet = new HashSet<>();

Set<Integer> treeSet = new TreeSet<>();

HashSet:

特点： 无序集合，基于HashMap实现。

优点： 高效的查找和插入操作。

缺点： 不保证顺序。

映射（Maps）

映射（Maps）用于存储键值对，常见的实现有 HashMap 和 TreeMap。

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

Map<String, Integer> treeMap = new TreeMap<>();

HashMap:

特点： 基于哈希表实现的键值对存储结构。

优点： 高效的查找、插入和删除操作。

缺点： 无序，不保证顺序。

TreeMap:

特点： 基于红黑树实现的有序键值对存储结构。

优点： 有序，支持按照键的顺序遍历。

缺点： 插入和删除相对较慢。

栈（Stack）

栈（Stack）遵循先进后出（FILO）原则。

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

Stack 类:

特点： 代表一个栈，通常按照后进先出（LIFO）的顺序操作元素。

队列（Queue）

队列（Queue）遵循先进先出（FIFO）原则，常见的实现有 LinkedList 和 PriorityQueue。

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

Queue 接口:

特点： 代表一个队列，通常按照先进先出（FIFO）的顺序操作元素。

实现类： LinkedList, PriorityQueue, ArrayDeque。

堆（Heap）

堆（Heap）优先队列的基础，可以实现最大堆和最小堆。

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

PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Collections.reverseOrder());

树（Trees）

Java 提供了 TreeNode 类型，可以用于构建二叉树等数据结构。

class TreeNode {

int val;

TreeNode left;

TreeNode right;

TreeNode(int x) { val = x; }

}

图（Graphs）

图的表示通常需要自定义数据结构或使用图库，Java 没有内建的图类。

以上介绍的只是 Java 中一些常见的数据结构，实际上还有很多其他的数据结构和算法可以根据具体问题选择使用。