**Task 3: Implementing Heap Operations**

**Code a min-heap in java with methods for insertion, deletion, and fetching the minimum element. Ensure that the heap property is maintained after each operation."**

**import** java.util.ArrayList;

**public** **class** MinHeap {

**private** ArrayList<Integer> heap;

**public** MinHeap() {

heap = **new** ArrayList<>();

}

// Method to insert a new element into the heap

**public** **void** insert(**int** element) {

heap.add(element);

heapifyUp(heap.size() - 1);

}

// Method to delete the minimum element (root) from the heap

**public** **int** deleteMin() {

**if** (heap.isEmpty()) {

**throw** **new** IllegalStateException("Heap is empty");

}

**int** minElement = heap.get(0);

**int** lastElement = heap.remove(heap.size() - 1);

**if** (!heap.isEmpty()) {

heap.set(0, lastElement);

heapifyDown(0);

}

**return** minElement;

}

// Method to fetch the minimum element (root) without removing it

**public** **int** getMin() {

**if** (heap.isEmpty()) {

**throw** **new** IllegalStateException("Heap is empty");

}

**return** heap.get(0);

}

// Helper method to maintain the heap property after insertion

**private** **void** heapifyUp(**int** index) {

**int** parentIndex = (index - 1) / 2;

**if** (index > 0 && heap.get(index) < heap.get(parentIndex)) {

swap(index, parentIndex);

heapifyUp(parentIndex);

}

}

// Helper method to maintain the heap property after deletion

**private** **void** heapifyDown(**int** index) {

**int** leftChildIndex = 2 \* index + 1;

**int** rightChildIndex = 2 \* index + 2;

**int** smallestIndex = index;

**if** (leftChildIndex < heap.size() && heap.get(leftChildIndex) < heap.get(smallestIndex)) {

smallestIndex = leftChildIndex;

}

**if** (rightChildIndex < heap.size() && heap.get(rightChildIndex) < heap.get(smallestIndex)) {

smallestIndex = rightChildIndex;

}

**if** (smallestIndex != index) {

swap(index, smallestIndex);

heapifyDown(smallestIndex);

}

}

// Helper method to swap elements in the heap

**private** **void** swap(**int** index1, **int** index2) {

**int** temp = heap.get(index1);

heap.set(index1, heap.get(index2));

heap.set(index2, temp);

}

// Method to display the heap (for debugging purposes)

**public** **void** printHeap() {

System.***out***.println(heap);

}

**public** **static** **void** main(String[] args) {

MinHeap minHeap = **new** MinHeap();

// Inserting elements into the heap

minHeap.insert(10);

minHeap.insert(15);

minHeap.insert(20);

minHeap.insert(17);

minHeap.insert(25);

minHeap.printHeap(); // Output: [10, 15, 20, 17, 25]

// Fetching the minimum element

System.***out***.println("Minimum element: " + minHeap.getMin()); // Output: 10

// Deleting the minimum element

System.***out***.println("Deleted minimum element: " + minHeap.deleteMin()); // Output: 10

minHeap.printHeap(); // Output: [15, 17, 20, 25]

// Fetching the new minimum element

System.***out***.println("Minimum element: " + minHeap.getMin()); // Output: 15

}

}