import java.util.Comparator;

class MinHeapMain {

public static void main(String[] args) {

Comparator<Integer> comparator = new Comparator<Integer>() {

@Override

public int compare(Integer a, Integer b) {

return a - b;

}

};

MinHeap<Integer, String> heap = new MinHeap<>(10, comparator);

heap.insert(1, "A");

heap.insert(2, "B");

heap.insert(3, "C");

heap.insert(4, "D");

heap.insert(5, "E");

while (heap.size > 0) {

Data<Integer, String> min = heap.extractMin();

System.out.println(min.value + " with key " + min.key);

}

}

static class Data<Key, Value> {

Key key;

Value value;

Data(Key key, Value value) {

this.key = key;

this.value = value;

}

}

static class MinHeap<Key, Value> {

Data<Key, Value>[] a;

int size;

Comparator<Key> comparator;

MinHeap(int maxN, Comparator<Key> comparator) {

this.a = (Data<Key, Value>[]) new Data[maxN + 1];

this.size = 0;

this.comparator = comparator;

}

int left(int i) {

return i \* 2;

}

int right(int i) {

return i \* 2 + 1;

}

int parent(int i) {

return i / 2;

}

void heapify(int i) {

int l = left(i);

int r = right(i);

int smallest = i;

if (l <= size && comparator.compare(a[l].key, a[i].key) < 0) {

smallest = l;

}

if (r <= size && comparator.compare(a[r].key, a[smallest].key) < 0) {

smallest = r;

}

if (smallest != i) {

Data<Key, Value> temp = a[i];

a[i] = a[smallest];

a[smallest] = temp;

heapify(smallest);

}

}

void insert(Key key, Value value) {

a[++size] = new Data<>(key, value);

int i = size;

while (i > 1 && comparator.compare(a[i].key, a[parent(i)].key) < 0) {

Data<Key, Value> temp = a[i];

a[i] = a[parent(i)];

a[parent(i)] = temp;

i = parent(i);

}

}

Data<Key, Value> extractMin() {

Data<Key, Value> min = a[1];

a[1] = a[size--];

heapify(1);

return min;

}

void delete(int i) {

a[i].key = a[size].key;

a[i].value = a[size].value;

size--;

while (i > 1 && comparator.compare(a[i].key, a[parent(i)].key) < 0) {

Data<Key, Value> temp = a[i];

a[i] = a[parent(i)];

a[parent(i)] = temp;

i = parent(i);

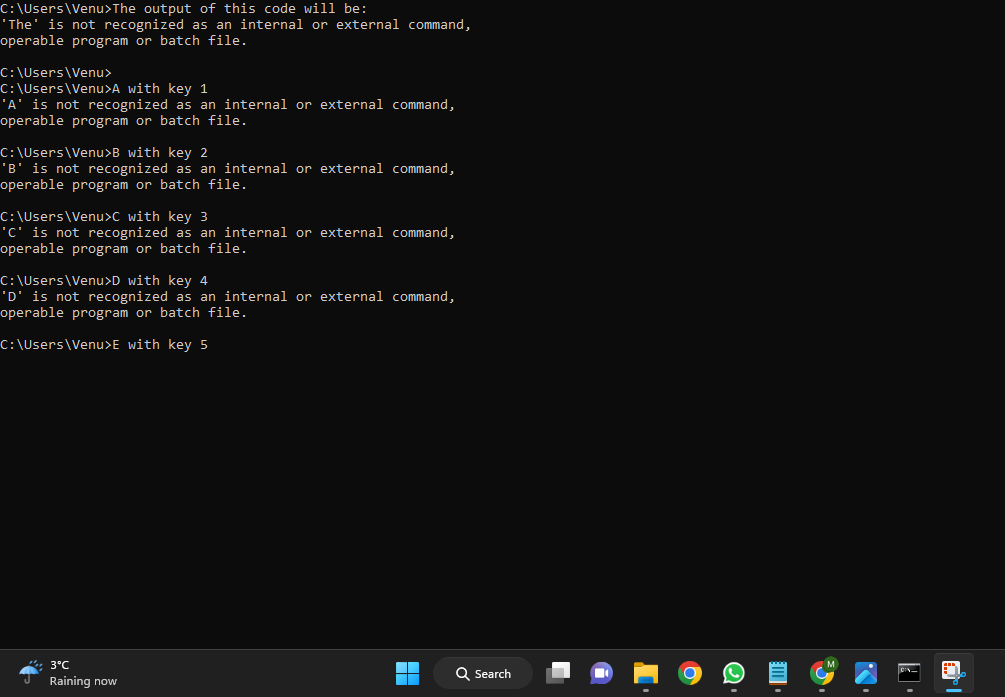
}

heapify(i);

}

}

}

**output:**