import java.util.\*;

class MinimumPriorityQueue {

    public static int heapSize = 0;

    public static int treeArraySize = 50;

    public static int INF = 100000;

    // function to get right child of a node of a tree

    public static int getRightChild(int A[], int index) {

        if ((((2 \* index) + 1) < A.length && (index >= 1)))

            return (2 \* index) + 1;

        return -1;

    }

    // function to get left child of a node of a tree

    public static int getLeftChild(int A[], int index) {

        if (((2 \* index) < A.length && (index >= 1)))

            return 2 \* index;

        return -1;

    }

    // function to get the parent of a node of a tree

    public static int getParent(int A[], int index) {

        if ((index > 1) && (index < A.length)) {

            return index / 2;

        }

        return -1;

    }

    public static void minHeapify(int A[], int index) {

        int leftChildIndex = getLeftChild(A, index);

        int rightChildIndex = getRightChild(A, index);

        // finding smallest among index, left child and right child

        int smallest = index;

        if ((leftChildIndex <= heapSize) && (leftChildIndex > 0)) {

            if (A[leftChildIndex] < A[smallest]) {

                smallest = leftChildIndex;

            }

        }

        if ((rightChildIndex <= heapSize && (rightChildIndex > 0))) {

            if (A[rightChildIndex] < A[smallest]) {

                smallest = rightChildIndex;

            }

        }

        // smallest is not the node, node is not a heap

        if (smallest != index) {

            int temp;

            // swapping

            temp = A[smallest];

            A[smallest] = A[index];

            A[index] = temp;

            minHeapify(A, smallest);

        }

    }

    public static void buildMinHeap(int A[]) {

        for (int i = heapSize / 2; i >= 1; i--) {

            minHeapify(A, i);

        }

    }

    public static int minimum(int A[]) {

        return A[1];

    }

    public static int extractMin(int A[]) {

        int minm = A[1];

        A[1] = A[heapSize];

        heapSize--;

        minHeapify(A, 1);

        return minm;

    }

    public static void decreaseKey(int A[], int index, int key) {

        A[index] = key;

        while ((index > 1) && (A[getParent(A, index)] > A[index])) {

            int temp;

            temp = A[getParent(A, index)];

            A[getParent(A, index)] = A[index];

            A[index] = temp;

            index = getParent(A, index);

        }

    }

    public static void increaseKey(int A[], int index, int key) {

        A[index] = key;

        minHeapify(A, index);

    }

    public static void insert(int A[], int key) {

        heapSize++;

        A[heapSize] = INF;

        decreaseKey(A, heapSize, key);

    }

    public static void printHeap(int A[]) {

        for (int i = 1; i <= heapSize; i++) {

            System.out.print(" " + A[i]);

        }

        System.out.println("");

    }

    public static void main(String[] args) {

        int heap[] = new int[treeArraySize];

        buildMinHeap(heap);

        // menu

        System.out.println("1 - Insert an element into Min Heap");

        System.out.println("2 - Change priority of an element");

        System.out.println("3 - Delete element from Min Heap");

        System.out.println("4 - Display Min/Root/Top Element in Heap");

        System.out.println("5 - Display Min Heap");

        System.out.println("6 - Exit");

        int choice;

        Scanner scanner = new Scanner(System.in);

        while (true) {

            System.out.print("Enter your choice : ");

            choice = scanner.nextInt();

            switch (choice) {

                case 1:

                    System.out.println("Inserting elements into Queue");

                    System.out.print("Enter number of elements: ");

                    int n = scanner.nextInt();

                    System.out.println("Insert elements");

                    for (int i = 0; i < n; i++)

                        insert(heap, scanner.nextInt());

                    System.out.println("Elements inserted successfully");

                    break;

                case 2:

                    System.out.println("Changing priority of element");

                    System.out.print("index: ");

                    int change\_index = scanner.nextInt();

                    System.out.print("priority: ");

                    int priority = scanner.nextInt();

                    increaseKey(heap, change\_index, priority);

                    System.out.println("Priority changed successfully");

                    break;

                case 3:

                    System.out.println("Deleted element " + extractMin(heap) + " with minimum priority");

                    break;

                case 4:

                    System.out.println("Min element in Heap is " + minimum(heap));

                    break;

                case 5:

                    System.out.print("Min Heap: ");

                    printHeap(heap);

                    break;

                case 6:

                    System.exit(0);

                default:

                    System.out.println("Choice is incorrect, Enter a correct choice");

            }

        }

    }

}

Output

PS Y:\development\ds-java\prority-queue> java .\main.java

1 - Insert an element into Min Heap

2 - Change priority of an element

3 - Delete element from Min Heap

4 - Display Min/Root/Top Element in Heap

5 - Display Min Heap

6 - Exit

Enter your choice : 1

Inserting elements into Queue

Enter number of elements: 10

Insert elements

20

15

8

10

5

7

6

2

9

1

Elements inserted successfully

Enter your choice : 5

Min Heap: 1 2 6 8 5 15 7 20 9 10

Enter your choice : 2

Changing priority of element

index: 5

priority: 22

Priority changed successfully

Enter your choice : 5

Min Heap: 1 2 6 8 10 15 7 20 9 22

Enter your choice : 4

Min element in Heap is 1

Enter your choice : 3

Deleted element 1 with minimum priority

Enter your choice : 5

Min Heap: 2 8 6 9 10 15 7 20 22

Enter your choice : 4

Min element in Heap is 2

Enter your choice : 3

Deleted element 2 with minimum priority

Enter your choice : 5

Min Heap: 6 8 7 9 10 15 22 20

Enter your choice : 4

Min element in Heap is 6

Enter your choice : 3

Deleted element 6 with minimum priority

Enter your choice : 5

Min Heap: 7 8 15 9 10 20 22

Enter your choice : 6