```java
/*
 * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
 * Version: Sun Oct  6 16:28:17 CEST 2024
 */

package uebung04.as.aufgabe02;


/**
 * A heap-based (array-implementation) Priority-Queue with fixed length.
 */
public class PriorityQueue<K extends Comparable<? super K>, V> {


  protected PQEntry<K, V>[] heapArray;

  /** Points to the last element in the heap. */
  protected int last = 0;

  public static class PQEntry<K extends Comparable<? super K>, V> implements
      Entry<K, V>, Comparable<PQEntry<K, V>> {

    protected K key;
    protected V value;

    protected PQEntry(K key, V value) {
      this.key = key;
      this.value = value;
    }

    @Override
    public K getKey() {
      return key;
    }

    @Override
    public V getValue() {
      return value;
    }

    @Override
    public int compareTo(PQEntry<K, V> other) {
      return this.key.compareTo(other.key);
    }

    @Override
    public String toString() {
      return "("+key+","+value+")";
    }

  }


  @SuppressWarnings("unchecked")
  public PriorityQueue(int maxSize) {
    heapArray = new PQEntry[maxSize + 1];
  }

  public Entry<K, V> insert(K key, V value) throws FullPriorityQueueException {

    // TODO: Implement here...

    last++;
    PQEntry<K, V> e = new PQEntry<>(key, value);
    heapArray[last] = e;

    // TODO: Implement here...

    return e;
  }
```



```java

  public Entry<K, V> min() {

    // TODO: Implement here...

    return null;
  }

  public Entry<K, V> removeMin() {

    // TODO: Implement here...

    return null;
  }

  protected void upheap(int currentIndex) {

    // TODO: Implement here...

  }

  protected void downheap(int currentIndex) {

    // TODO: Implement here...

  }

  /**
   * Swaps a child-node with its parent-node.
   * @param parentIndex Index of the parent-node.
   * @param childIndex Index of the child-node.
   */
  protected void swap(int parentIndex, int childIndex) {

    // TODO: Implement here...

  }

  public boolean isEmpty() {

    // TODO: Implement here...

    return true;
  }

  public int size() {

    // TODO: Implement here...

    return -1;
  }
```

```
122
123   @Override
124   public String toString() {
125     StringBuilder sb = new StringBuilder();
126     sb.append("[");
127     for (int i = 0; i < heapArray.length; i++) {
128       PQEntry<K, V> e = heapArray[i];
129       if (e != null) {
130         sb.append('[');
131         sb.append(e);
132         sb.append(',');
133         sb.append(i);
134         sb.append(']');
135       } else {
136         sb.append("null");
137       }
138       if (i < heapArray.length-1) {
139         sb.append(", ");
140       }
141     }
142     sb.append("]");
143     return sb.toString();
144   }
145
146   public void print() {
147     System.out.println(toString());
148   }
149
150 }
151
152
```



```
1  /*
2   * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3   * Version: Sun Oct  6 16:28:17 CEST 2024
4   */
5
6  package uebung04.as.aufgabe02;
7
8  public interface Entry<K, V> {
9
10   K getKey();
11
12   V getValue();
13
14 }
```

6.10.2024 16:28:17 **FullPriorityQueueException.java** Page 1/1

```java
/*
 * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
 * Version: Sun Oct  6 16:28:17 CEST 2024
 */

package uebung04.as.aufgabe02;


/**
 * Thrown at insert()-operation when PriorityQueue is already full.
 */
public class FullPriorityQueueException extends Exception {

  private static final long serialVersionUID = 1L;

  public FullPriorityQueueException() {
    super();
  }
}
```

6.10.2024 16:28:17 **PriorityQueueTest.java** Page 1/3

```java
/*
 * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
 * Version: Sun Oct  6 16:28:17 CEST 2024
 */

package uebung04.as.aufgabe02;

import java.util.Arrays;
import java.util.Random;

public class PriorityQueueTest {

  private static void stressTest() throws FullPriorityQueueException {
    System.out.print("\nStress-Test: ... ");
    final int NUMBER_OF_TESTS = 1_000_000;
    final int LENGTH_RANGE = 10;
    final int DATA_RANGE = 10;
    Random random = new Random(1);
    for (int testNr = 0; testNr < NUMBER_OF_TESTS; testNr++) {
      int length = (int) (random.nextDouble() * LENGTH_RANGE + 1);
      int[] array = new int[length];
      for (int i = 0; i < length; i++) {
        int number = (int) (random.nextDouble() * DATA_RANGE + 1);
        array[i] = number;
      }
      PriorityQueue<Integer, String> ourPQ = new PriorityQueue<>(length);
      java.util.PriorityQueue<Integer> javaPQ = new java.util.PriorityQueue<>();
      for (int i : array) {
        ourPQ.insert(i, "Value_"+i);
        javaPQ.add(i);
      }
      for (int i = 0; i < array.length; i++) {
        if (ourPQ.size() != javaPQ.size()) {
          System.out.println("ERROR: wrong size!");
          System.out.println("Array: " + Arrays.toString(array));
          System.exit(1);
        }
        if (!ourPQ.removeMin().getKey().equals(javaPQ.poll())) {
          System.out.println("ERROR: wrong removeMin()!");
          System.out.println("Array: " + Arrays.toString(array));
          System.exit(1);
        }
      }
      if (ourPQ.removeMin() != null) {
        System.out.println("ERROR: removeMin() != null");
        System.out.println("Array: " + Arrays.toString(array));
        System.exit(1);
      }
    }
    System.out.println("o.k.");
  }
```

```
52
53    public static void main(String[] args) throws FullPriorityQueueException {
54
55      PriorityQueue<Integer, String> pq =
56          new PriorityQueue<>(7);
57          //new PriorityQueueADV<>(7, "Uebung 4:PQ", 2, 2);
58
59      System.out.println("insert()'s: ");
60      pq.print();
61      pq.insert(4, "D");
62      pq.print();
63      pq.insert(5, "E");
64      pq.print();
65      pq.insert(3, "C");
66      pq.print();
67      pq.insert(2, "B");
68      pq.print();
69      pq.insert(1, "A");
70      pq.print();
71      System.out.println("\nmin(): " + pq.min());
72      while (pq.size() > 1) {
73        System.out.println("removeMin(): " + pq.removeMin());
74        pq.print();
75      }
76
77      stressTest();
78
79    }
80
81  }
```



```
82
83  /* Session-Log:
84
85  insert()'s:
86  [null, null, null, null, null, null, null, null]
87  [null, [(4,D),1], null, null, null, null, null, null]
88  [null, [(4,D),1], [(5,E),2], null, null, null, null, null]
89  [null, [(3,C),1], [(5,E),2], [(4,D),3], null, null, null, null]
90  [null, [(2,B),1], [(3,C),2], [(4,D),3], [(5,E),4], null, null, null]
91  [null, [(1,A),1], [(2,B),2], [(4,D),3], [(5,E),4], [(3,C),5], null, null]
92
93  min(): (1,A)
94  removeMin(): (1,A)
95  [null, [(2,B),1], [(3,C),2], [(4,D),3], [(5,E),4], null, null, null]
96  removeMin(): (2,B)
97  [null, [(3,C),1], [(5,E),2], [(4,D),3], null, null, null, null]
98  removeMin(): (3,C)
99  [null, [(4,D),1], [(5,E),2], null, null, null, null, null]
100 removeMin(): (4,D)
101 [null, [(5,E),1], null, null, null, null, null, null]
102
103 Stress-Test: ... o.k.
104
105 */
```