Assignment - 7

Task1: implement a concrete HeapPQueue (Java) class (a min-Heap based priority queue) that implements the PQueue interface (provided).  The template of HeapPQueue class is given below:

public interface PQueue {  
 int size(); // return the number of keys stored  
 boolean isEmpty(); // return whether or not this heapPQueue is empty  
 void insert(int key); // insert a provided key to this heapPQueue  
 int min(); // return the key of the highest priority  
 int removeMin(); // remove and return the key of the highest priority  
}

public class HeapPQueue implements PQueue {

private int[] list;  
 private int size;

//constructor  
 public HeapPQueue(int[] items) {  
 // your implementation  
 }

public int size() {// your implementation}

public boolean isEmpty() {// your implementation}

public void insert(int key) {// your implementation}

public int removeMin() {// your implementation }

public int min() {// your impelementation }  
 private void bottom\_up(int[] items) {  
 // your impelementation   
 // invoke this method when you construct a heap given an array of keys  
 }  
 private void heapdown() {  
 // your impelementation   
 // invoke this method when you remove the key of the highest priority  
 }  
 private void heapup(int i) {  
 // your impelementation   
 // invoke this method when you insert a new key into this heap  
 }  
 }  
  
Task2:   
a) given the min-heap class you implement in task1, then implement a heapSort() method that sorts a given key array by using heapSort.   
b) implement a quickSort() method (based on quicksort algorithm and Lomuto\_partition algorithm discussed in the lectures) that performs the quicksort for the given key array.  
  
Task3:  
Compare the sorting performance between your implemented heapSort() and quickSort() methods. A template is given below.  
  
#import java.util.Random;  
public class SortingTest {  
 public static void main(String[] args) {  
 long startTime, endTime;  
 Random rand = new Random();  
 int[] keys = new int[10000];  
 int[] output1 = new int[10000];  
 int[] output2 = new int[10000];  
 // generate 5,000 - 50,000 random key values and assign them to keys[]  
 for (int i = 0; i < 10000; i++) {  
 keys[i] = rand.nextInt();  
 output1[i] = keys[i];  
 output2[i] = keys[i];  
 }  
  
 startTime = System.currentTimeMillis();  
 heapSort(output1);  
 endTime = System.currentTimeMillis();  
 System.out.println("Heapsort takes: " + (endTime - startTime) + " ms");  
  
 startTime = System.currentTimeMillis();  
 quickSort(output2);  
 endTime = System.currentTimeMillis();  
 System.out.println("Quicksort takes: " + (endTime - startTime) + " ms");  
  
 }  
  
 private static void heapSort(int[] items) {  
 /\* your implementation \*/  
 }  
  
 private static void quickSort(int[] items) {  
 /\* your implementation \*/  
 }  
}