Übung 6: Heap

Abgabetermin: 30.04.2013

Name: _____ **Matrikelnummer:** _____

Gruppe: □ **G1** Di 10:15 □ **G2** Di 11:00 □ **G3** Di 12:45

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Punkte
Aufgabe 1	24	□	Java-Programm Testfälle und Ergebnisse	Java-Programm	□	

Aufgabe 1: Pioritätswarteschlange für Zeichen (24 Punkte)

Implementieren Sie eine Prioritätswarteschlange für Buchstaben mit einem Heap. Erlaubt sind nur die Kleinbuchstaben 'a' bis 'z'. Größere Zeichen sollen höhere Priorität als kleinere haben, d. h. 'z' kommt vor 'a'. Die Schnittstelle ist durch die abstrakte Klasse *PriorityQueue* gegeben (für Methodenbeschreibungen siehe Java-Dokumentation in der Vorgabedatei).

```
package at.jku.ssw;
public abstract class PriorityQueue {
  public abstract void offer(char value);
  public abstract char poll();
  public abstract int size();
  public abstract CharIterator iterator();
  public abstract void clear();
  public abstract char peek();
  public abstract boolean remove(char value);
}
```

```
public abstract class CharIterator {
  public abstract boolean hasNext();
  public abstract char next();
}
```

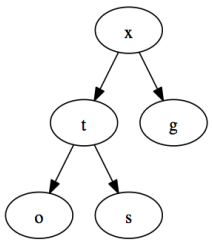
Implementieren Sie die Klassen *ArrayPriorityQueue* und *ArrayPriorityQueueIterator* im Paket *at.jku.students*.

```
package at.jku.students;
public class ArrayPriorityQueue
    extends PriorityQueue {
  char[] values = new char[1];
  int count = 1;
  public String makeDot() {
    return DotMaker.makeDotForHeap(
      Arrays.copyOf(values, count));
  }
  ...
}
public class ArrayPriorityQueueIterator
    extends CharIterator {
  ...
}
```

```
PriorityQueue pq
  = new ArrayPriorityQueue();
pq.offer('T');
pq.offer('O');
pq.offer('G');
pq.offer('X');
pq.offer('S');
makeDot((ArrayPriorityQueue) pq, "Test.dot");
Out.print(pq.size() + ": ");
while (pq.size() > 0) {
  Out.print(" " + pq.poll());
} // Ausgabe 5: xtsog
```

Implementierungshinweise:

- Verwenden Sie ein Array um den Heap zu implementieren. Lassen Sie das Array am Index 0 für ein Dummy-Element leer. Lassen Sie das Array dynamisch wachsen, indem Sie bei Bedarf die Länge verdoppeln.
- Verwenden Sie die Vorgabedateien *ssw-pi2.jar* und die zugehörige Java-Dokumentation von der LVA-Website.
- Definieren Sie für alle Klassen, Methoden und Felder die geeignete Sichtbarkeit (private, protected, package, public).
- Verwenden Sie die Methode *DotMaker.makeDotForHeap* um GraphViz-Bilder Ihres Heaps zu erstellen.
- Testen Sie Ihre Implementierung mit der Vorgabedatei *PriorityQueueTest.java* und vergleichen Sie Ihre Ergebnisse mit *PriorityQueueTest.Output.txt*.



Abzugeben ist: Java-Programm, Testfälle