Übung 6: Heap

Abgabetermin: 29.04.2014

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: Prioritätswarteschlange für vergleichbare Objekte (24 Punkte)

Implementieren Sie eine Prioritätswarteschlange für Java-Objekte mit einem Heap. Die Objekte sind vom Typ *Comparable,* der unter anderem von *String* implementiert wird. Die Methode *obj.compareTo(other)* von *Comparable* gibt einen negativen Wert zurück, wenn *obj* kleiner als *other* ist, oder einen positiven Wert, wenn *obj* größer als *other* ist (bei 0 sind die Objekte gleich). Die Schnittstelle ist durch die abstrakte Klasse *PriorityQueue* gegeben (für Methodenbeschreibungen siehe Java-Dokumentation in der Vorgabedatei).

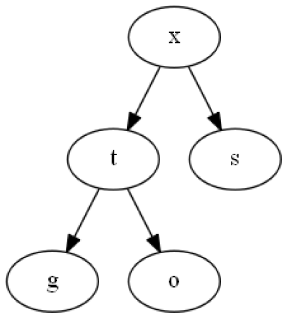
```
package at.jku.ssw;
public abstract class PriorityQueue {
  public abstract void offer(Comparable value);
  public abstract Comparable poll();
  public abstract int size();
  public abstract Iterator iterator();
  public abstract void clear();
  public abstract Comparable peek();
  public abstract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

```
public abstract class Iterator {
  public abstract boolean hasNext();
  public abstract Object next();
}
```

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

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

```
PriorityQueue pq
  = new ArrayPriorityQueue();
pq.offer("t");
pq.offer("g");
pq.offer("s");
pq.offer("x");
pq.offer("o");
Out.open("Test.dot");
((ArrayPriorityQueue) pq)
   .makeDot();
Out.close();
Out.print(pq.size() +
":");
while (pq.size() > 0) {
 Out.print(" " + pq.poll());
}  // Ausgabe 5: x t s o g
```



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.
- Der Iterator soll die Objekte sortiert nach Priorität liefern, absteigend vom größten zum kleinsten Objekt.

- 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