Übung 6: Heap

Abgabetermin: 03.05.2016

Name: _____ **Matrikelnummer:** _____

Gruppe: ☐ **G1** Di 10:15-11:00 ☐ **G2** Di 11:00-11:45 ☐ **G3** Di 12:45-13:30

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

Aufgabe 1: Prioritätswarteschlange für Objekte mit Comparator (24 Punkte)

Implementieren Sie eine Prioritätswarteschlange für Java-Objekte mit einem Heap. Zum Vergleich der Objekte wird im Konstruktor ein *Comparator* übergeben. Die Methode *compare(obj1, obj2)* von *Comparator* gibt einen negativen Wert zurück, wenn *obj1* kleiner als *obj2* ist, oder einen positiven Wert, wenn *obj1* größer als *obj2* ist. Ein Wert von 0 bedeutet, dass die Objekte gleich sind. Die Schnittstelle der Warteschlange ist durch die abstrakte Klasse *PriorityQueue* gegeben (Methodenbeschreibungen in Java-Dokumentation der Vorgabedatei).

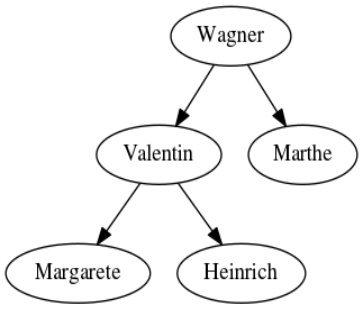
```
package at.jku.ssw.pi.heap;
public abstract class PriorityQueue {
  public abstract void offer(Object value);
  public abstract Object poll();
  public abstract int size();
  public abstract Iterator iterator();
  public abstract void clear();
  public abstract Object peek();
  public abstract boolean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

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

Implementieren Sie die Klassen *ArrayPriorityQueue* und *ArrayPriorityQueueIterator* im Paket *at.jku.ssw.pi.heap.ue06*.

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

```
Comparator cmp = String.CASE_INSENSITIVE_ORDER;
PriorityQueue pq
  = new ArrayPriorityQueue(cmp);
pq.offer("Margarete");
pq.offer("Valentin");
pq.offer("Marthe");
pq.offer("Wagner");
pq.offer("Heinrich");
Out.open("Test.dot");
Out.print(
 ((ArrayPriorityQueue)
   pq).makeDot());
Out.close();
Out.print(pq.size()
          + ":");
while (pq.size() > 0) {
 Out.print(" " + pq.poll());
} // Ausgabe 5: Wagner Valentin Marthe Margarete Heinrich
```



Implementierungshinweise:

Verwenden Sie ein Array um den Heap zu implementieren. Lassen Sie das Array am Index 0 leer. Wenn der Platz beim Einfügen nicht mehr ausreicht, erzeugen Sie ein neues, Array mit der doppelten Größe und kopieren Sie die Elemente.

Der Iterator soll die Objekte sortiert nach Priorität liefern, absteigend vom größten zum kleinsten Objekt. Beim Iterieren dürfen aber keine Objekte aus der ursprünglichen Warteschlange entfernt werden.

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 Heap zu erstellen.

Testen Sie Ihre Implementierung mit der Vorgabedatei *PriorityQueueTest.java*.

Abzugeben ist: Java-Programm, Testfälle (inkl. Ausgabe von *PriorityQueueTest.java*)