```
package blatt09;

import util.ArrayUtil;

public class Heap {

	private int[] heap;
	private int elements;

	public Heap(int size){
		this.heap = new int[size + 1];
		this.elements = 0;
	}

	/**
	 * Fügt das Element <code>item</code> in den Heap ein, wenn noch Platz ist
	 */
	public void insert(int item){
		if(this.elements + 1 < this.heap.length){
			// Das item in den Heap einfügen
			this.heap[this.elements + 1] = item;
			// Das item den Heap aufsteigen lassen
			this.ascent(this.elements + 1);
			// Den Element-Zähler um 1 erhöhen
			this.elements++;
		}
	}

	/**
	 * Lässt den Knoten am <code>index</code> aufsteigen, wenn es kleiner ist als sein Elternknoten
	 */
	private void ascent(int index){
```

```
		// Eltern Position berechnen:
		int parent = index / 2; // 0 ersetzen durch Berechnung
		if(parent == 0) return;
		if(this.heap[parent] > this.heap[index]){
			// Aufgabe: Elemente tauschen und weiter aufsteigen lassen
			ArrayUtil.swap(this.heap, index, parent);
			ascent(parent);
		}
}

/**
 * Entfernt das minimum und gibt es zurück
 */
public int removeMin(){
		int ret = ArrayUtil.remove(this.heap, 1);
		// Das Element an Position 1 entfernen

		// Das Element an der letzten Position des Feldes mit der 1. tauschen
		ArrayUtil.swap(this.heap, 1, this.elements);

		// Das neue 1. Element im Baum nach unten sickern lassen
		this.descent(1);

		// Den Element-Zähler um 1 reduzieren
		this.elements--;

		return ret;
}

/**
 * Schiebt den Knoten am index hinab, wenn er größer ist als seine Kindknoten
 */
```

```
	private void descent(int index){
		int left = index * 2;			// Index des linken Kindknoten berechnen
		int right = index * 2 + 1;	// Index des rechten Kindknoten berechnen
		if(left <= this.elements && right > this.elements){		// Es gibt nur einen linksknoten
			// Wenn linkes Kind kleiner ist als aktueller Knoten: Tauschen
			if(this.heap[left] < this.heap[index]){
				ArrayUtil.swap(this.heap, left, index);
			}
		} else if (right <= this.elements){ // Es existieren sowohl linkes als auch rechtes Kind
			// Kleineren der beiden Kinder selektieren
			int select = this.heap[left] < this.heap[right] ? left : right;
			// Wenn kleinerer Knoten kleiner ist als
			//  aktueller Knoten: Tauschen und weiter hinabsteigen
			if(this.heap[select] < this.heap[index]){
				ArrayUtil.swap(this.heap, select, index);
				this.descent(select);
			}
		}
	}
}
```