Übung 2: Stack, Queue

Abgabetermin: 22.03.2011

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	12	☐	Java-Programm Testfälle und Ergebnisse	Java-Programm	☐	
Aufgabe 2	12	☐			☐	

Aufgabe 1: Stack für Integer-Zahlen (12 Punkte)

Implementieren Sie einen Kellerspeicher, einmal mit einem Array in der Klasse *ArrayStack* und einmal als verkettete Liste in der Klasse *LinkedListStack*. Der Kellerspeicher hat folgende Methoden: *push* kellert eine Zahl ein, *pop* kellert eine Zahl aus, *size* liefert die Anzahl der Zahlen und *iterator* liefert einen Iterator mit dem der Kellerspeicher von oben nach unten durchlaufen werden kann.

Implementierungshinweise: Beim *ArrayStack* initialisieren Sie das Array mit Länge 1 und verdoppeln Sie die Länge wenn das Array voll ist. Beim *LinkedListStack* verwenden Sie intern die *LinkedList* aus Übung 1 um die Zahlen zu speichern.

```
class ArrayStack {
  int[] stack = new int[1]; int count = 0;
  public void push(int value) { ... }
  public int pop() { ... }
  public int size() { ... }
  public Iterator iterator() { ... }
}
class LinkedListStack {
  LinkedList list = ... // aus Uebung 1
  ... // Methoden wie ArrayStack
}
public abstract class Iterator {
  public abstract boolean hasNext();
  public abstract int next();
}
```

```
ArrayStack s = new ArrayStack();
s.push(5);
s.push(9);
s.push(7);
Iterator it = s.iterator();
while (it.hasNext()) {
  Out.print(" " + it.next());
}
// Ausgabe: 7 9 5
s.push(s.pop() + s.pop());
Out.println(s.pop() * s.pop());
// Ausgabe 80
```

Aufgabe 2: Queue für Integer-Zahlen (12 Punkte)

Implementieren Sie eine FIFO-Warteschlange (First-in-first-out), einmal mit einem Array in der Klasse *ArrayQueue* und einmal als verkettete Liste in der Klasse *LinkedListQueue*. Die Warteschlange hat folgende Methoden: *put* fügt eine Zahl ein, *get* entnimmt eine Zahl, *size* liefert die Anzahl der Zahlen und *iterator* liefert einen Iterator mit dem man die Warteschlange in FIFO-Reihenfolge durchlaufen kann.

Die Implementierungshinweise aus Aufgabe 1 gelten sinngemäß für Aufgabe 2.

```
public class ArrayQueue {
  int[] queue = new int[1]; int count = 0;
  public void put(int value) { ... }
  public int get() { ... }
  public int size() { ... }
  public Iterator iterator() { ... }
}
public class LinkedListQueue {
  LinkedList list = ... // aus Uebung 1
  ... // Methoden wie ArrayQueue
}
```

```
ArrayQueue q = new ArrayQueue();
q.put(3); q.put(9); q.put(5);
Out.print(q.get());
// Ausgabe: 3
q.put(1);
Out.print(l.size() + ": ");
Iterator it = l.iterator();
while (it.hasNext()) {
  Out.print(" " + it.next());
}
// Ausgabe 3: 9 5 1
```

Abzugeben ist: Java-Programm, Testfälle und Ergebnisse