Dr. M. Grabert   Abteilung Angewandte Informationsverarbeitung   15. Februar 2002
Johannes Mayer   Blatt 10

Objektorientierte Programmierung mit C++ (WS 2001/2002)

Besprechung: 14. Februar 2002

Dieses Blatt ist ein **rein optionales** Übungsblatt zum **Java-Teil** der Vorlesung. Dieses Übungsblatt hat keine Auswirkungen auf den Erhalt des Scheins. Es dient nur der Vertiefung der Einführung in Java. Die Lösung wird **nicht in den Tutorien**, sondern **in den Übungen** besprochen.

14 Vektoren

Schreiben Sie eine eigene Klasse `MyVector`, die die wichtigsten Vektor-Operationen unterstützt. Als Anhaltspunkt ist folgendes Hauptprogramm vorgegeben:

```
public class MyVector {
    /* ... */
    public static void main(String[] args) {
MyVector u = new MyVector(new double[] {1, 2, 3});
MyVector v = new MyVector(new double[] {-3, 5, 9});

System.out.println("u = " + u);
System.out.println("v = " + v);
System.out.println("Dimension von u: " + u.dimension());
System.out.println("Norm von u:      " + u.norm());
System.out.println("u + v = " + u.add(v));
System.out.println("u - v = " + u.sub(v));
System.out.println("<u,v> = " + u.scalarProduct(v));
    }
}
```

15 Elementare Datenstrukturen

Schreiben Sie basierend auf die Klasse `Vector` (siehe Skript S. 50 f. und *API-Dokumentation zur Klasse Vector*) eine Klasse `Stack`, die einen Stack von Elementen vom Typ `Object` implementiert. Gegeben ist das folgende Grundgerüst:

```
import java.util.Vector;
import java.io.*;

public class Stack {
    /* ... */
    public static void main(String[] args) throws IOException {
Stack st = new Stack();

// aus stdin einen BufferedReader (zum Lesen von Zeilen) machen ...
InputStreamReader ir = new InputStreamReader(System.in);
BufferedReader r = new BufferedReader(ir);

// zeilenweise von stdin lesen und auf den Stack legen ...
while (true) {
    String s = r.readLine();
    if (s == null)
break;
    st.push(s);
}

// ... und zum Schluss den Stackinhalt ausgeben
while (!st.isEmpty()) {
    String s = (String) st.pop();
    System.out.println(s);
}
    }
}
```

Wichtig bei Stack sind die Operationen `push(.)`, mit der man ein Element auf den Stack legen kann, und `pop()`, mit der man das oberste Element erhält und zugleich entfernt. Des Weiteren soll es noch die Operationen `size()`, `isEmpty()` und `clear()` geben.
Schreiben Sie außerdem noch eine Klasse `Queue`, die eine Warteschlange mit den Operationen `append(.)`, welche ein Element ans Ende anhängt, und `front()`, die das erste Element entfernt und liefert, (ebenfalls mittels `Vector`) implementiert.

```
import java.util.Vector;
import java.io.*;

public class Queue {
    /* ... */
    public static void main(String[] args) throws IOException {
Queue q = new Queue();

// aus stdin einen BufferedReader (zum Lesen von Zeilen) machen ...
InputStreamReader ir = new InputStreamReader(System.in);
BufferedReader r = new BufferedReader(ir);
```

```
// zeilenweise von stdin lesen und in die Queue tun ...
while (true) {
    String s = r.readLine();
    if (s == null)
break;
    q.append(s);
}

// ... und zum Schluss den Queueinhalt ausgeben
while (!q.isEmpty()) {
    String s = (String) q.front();
    System.out.println(s);
}
    }
}
```

16 Fröhliches Zählen

Schreiben Sie nun noch eine Klasse `WordCount`, die im Konstruktor einen `Reader` enthält und alle Zeichen von dieser Eingabe sofort liest und zählt. Diese Klasse soll dann Methoden haben, mit denen das Ergebnis des Zählens abgefragt werden kann (Anzahl der Zeichen, Worte und Zeilen). Das folgende Gerüst ist schon gegeben:

```
import java.io.*;

public class WordCount {
    /* ... */
    public static void main(String[] args) throws IOException {
Reader r = new BufferedReader(new InputStreamReader(System.in));
WordCount wc = new WordCount(r); // aus stdin lesen und zaehlen

// Ergebnis des Zaehlens ausgeben
System.out.println(" " + wc.getNumOfLines() + " " +
    wc.getNumOfWords() + " " + wc.getNumOfChars());
    }
}
```

Viel Spass und Erfolg!