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

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

Besprechung: 14. Februar 2002

Beispiellösung

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));
    }
}
```

Lösung:

MyVector.java

```
public class MyVector {
    private double[] v;
    public MyVector(int n) {
        if (n <= 0)
            throw new IllegalArgumentException("n must be positive");

        v = new double[n];
    }
    public MyVector(double[] v) {
        if (v.length < 1)
            throw new IllegalArgumentException("length of v must be positive");

        this.v = v;
    }
    public int dimension() {
        return v.length;
    }
    public double norm() {
        double res = 0.0;
        for (int i = 0; i < v.length; i++)
            res += v[i] * v[i];
        return Math.sqrt(res);
    }
    public MyVector add(MyVector w) {
        if (v.length != w.v.length)
            throw new IllegalArgumentException("w must have the same" +
                                                    " dimension as this vector");

        MyVector x = new MyVector(v.length);
        for (int i = 0; i < v.length; i++)
            x.v[i] = v[i] + w.v[i];
        return x;
    }
    public MyVector sub(MyVector w) {
        if (v.length != w.v.length)
            throw new IllegalArgumentException("w must have the same " +
                                                    "dimension as this vector");

        MyVector x = new MyVector(v.length);
        for (int i = 0; i < v.length; i++)
            x.v[i] = v[i] - w.v[i];
        return x;
    }
    public double scalarProduct(MyVector w) {
        if (v.length != w.v.length)
            throw new IllegalArgumentException("w must have the same " +
                                                    "dimension as this vector");

        double res = 0.0;
```

```
        for (int i = 0; i < v.length; i++)
            res += v[i] * w.v[i];
        return res;
    }
    public String toString() {
        String s = "(";
        for (int i = 0; i < v.length; i++) {
            if (i > 0)
                s += ", ";
            s += v[i];
        }
        return s + ")";
    }

    // kleiner Tester fuer die Klasse 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);
}
    }
}
```

Lösung:

Stack.java

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

public class Stack {
    private Vector v;
    public Stack() {
        v = new Vector();
    }
    public int size() {
        return v.size();
    }
    public boolean isEmpty() {
        return v.isEmpty();
    }
    public void clear() {
        v.clear();
    }
    public void push(Object o) {
        v.add(o);
    }
    public Object pop() {
        if (v.size() > 0) {
            Object o = v.lastElement();
            v.remove(v.size()-1);
            return o;
        }
        else
            return null;
    }

    // Testroutine fuer die Stack-Klasse
    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);
        }
    }
}
```

Queue.java

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

public class Queue {
    private Vector v;
    public Queue() {
        v = new Vector();
    }
    public int size() {
        return v.size();
    }
    public boolean isEmpty() {
        return v.isEmpty();
    }
    public void clear() {
        v.clear();
    }
    public void append(Object o) {
        v.add(o);
    }
    public Object front() {
        if (!v.isEmpty()) {
            Object o = v.firstElement();
            v.remove(0);
            return o;
        }
        else
            return null;
    }

    // Testroutine fuer die Queue-Klasse
    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());
    }
}
```

Lösung:

WordCount.java

```
import java.io.*;
```

```
/**
* Diese Klasse kann aus einem <code>Reader</code> die
* Worte, Zeilen und Zeichen z&auml;hlen.
* <p>
* In diesem Quelltext sehen sie ein Beispiel f&uuml;r eine in
* den Quelltext eingebettete sogennante javadoc-Dokumentation.
* Mit dem Kommando <code>javadoc</code> kann dann aus der Dokumenation
* in diesem Quelltext eine HTML-Seite generiert werden. Folglich
* k&ouml;nnen in den Kommentaren auch HTML-Kommandos enthalten sein.
* Zur Erzeugung der Dokumentation muss man folgende Kommandozeile
* eingeben (das Kommando <code>javadoc</code> und den/die Klassen-
* bzw. Paketnamen):<br>
* <pre>
* thales$ javadoc WordCount
* </pre>
*
* @author  Johannes Mayer
* @version 1.0, 2002-02-06
*/
public class WordCount {
    /** Das Ergebnis des Z&auml;hlens: Die Anzahl der Zeilen, Worte und Zeichen. */
    private int numOfLines = 0;
    private int numOfWords = 0;
    private int numOfChars = 0;

    /**
    * Erzeugt ein neues <code>WordCount</code>-Objekt und z&auml;hlt
    * sogleich die Eingabe von <code>r</code>.
    *
    * @param    r       der <code>Reader</code>, dessen Eingabe gez&auml;hlt
    *                   werden soll.
    * @throws   IOException
    *                   falls es zu einem Fehler beim Lesen aus <code>r</code>
    *                   kommen sollte.
    */
    public WordCount(Reader r) throws IOException {
        count(r);
    }
    /**
    * Liest die Eingabe aus <code>r</code> und z&auml;hlt sie. Das
    * Ergebnis steht danach in den Objekt-Variablen <code>numOfChars</code>,
    * <code>numOfWords</code> und <code>numOfLines</code>.
    *
    * @param    r       das <code>Reader</code>-Objekt, dessen Eingabe
    *                   gez&auml;hlt werden soll.
    * @throws   IOException
    *                   falls es beim Lesen von <code>r</code> zu einem
    *                   derartigen Fehler kommt.
    */
```

```
private void count(Reader r) throws IOException {
    // Das letzte gelesene Zeichen wird zum erkennen von Worten und
    // Zeilen benoetigt.
    char lastChar = '\n';

    int c;
    while ((c = r.read()) >= 0) {
        // neues Zeichen
        numOfChars++;
        // wenn Uebergang von Whitespace zu Nicht-Whitespace, dann
        // neues Wort zaehlen
        if (!Character.isWhitespace((char) c) && Character.isWhitespace(lastChar))
            numOfWords++;
        // wenn letztes Zeichen ein Zeilenumbruch war und jetzt
        // noch ein Zeichen kam, dann weitere Zeile zaehlen
        if (lastChar == '\n')
            numOfLines++;
        lastChar = (char) c;
    }
}
/**
* Liefert die Anzahl der Zeichen in der Eingabe.
*
* @return   die Anzahl der Zeichen in der Eingabe.
*/
public int getNumOfChars() {
    return numOfChars;
}
/**
* Liefert die Anzahl der Worte in der Eingabe.
*
* @return   die Anzahl der Worte in der Eingabe.
*/
public int getNumOfWords() {
    return numOfWords;
}
/**
* Liefert die Anzahl der Worte in der Eingabe.
*
* @return   die Anzahl der Worte in der Eingabe.
*/
public int getNumOfLines() {
    return numOfLines;
}
/**
* Eine kleine Testroutine f&uuml;r die Klasse <code>WordCount</code>,
* die von der Standardeingabe (<code>System.in</code>) liest und
* das Ergebnis des Z&auml;hlens nach <code>System.out</code> ausgibt.
*
```

```
     * @param    args    die Kommandozeilenargumente.
     * @throws   IOException
     *                   falls es beim Lesen der Standardeingabe zu einer
     *                   derartigen Ausnahme kommen sollte.
     */
    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());
    }
}
```