TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Computer Vision Group
Einführung in die Informatik I für Ingenieure (MSE)
Prof. Dr. Daniel Cremers, Mohamed Souiai

WS 2012
Übungsblatt 5
24.11.2012

Aufgabe 5.1 (Ü) Strings

Schreiben Sie eine Methode, welche einen Dateinamen (als String) als Argument erhält und die mittelere Länge (als float) der Wörter des in der Datei enthaltenen Textes brechnet und zurückgibt, in dem Sie den Code auf der Übungshomepage erweitern. Mit der vorgegebenen Funktion `readFile` kann eine Textdatei eingelesen werden und der darin enthaltene Text in einem Sting umgewandelt werden. Sie können einen String aufsplitten in dem Sie die String-Methode `split()` wie folgt verwenden:

```
String [] einzelneWoerter= textAlsString.split("Trennzeichen");
```

wobei das Trennzeichen die Stellen spezifiziert, an dem der Text getrennt wird.

Lösungsvorschlag 5.1

```
class TxtAvgWordLength extends MiniJava {

    public static String readFile(String filename) {
        try {
            // initialization
            java.io.BufferedReader reader = new java.io.BufferedReader(
               new java.io.FileReader(filename));
            StringBuffer buffer = new StringBuffer();
            // read line after line
            String line;
            while ((line = reader.readLine()) != null) {
                buffer.append(line);
            }
            reader.close();

            // return file contents as string
            return buffer.toString();

        } catch (Exception e) {
            // error handling
            System.err.println(e);
            return null;
        }
    }

    public static float avgWordLength(String dateiname) {
        String string = readFile(dateiname);
        // split text into an array of strings
        String[] textliste = string.split("␣");
        float word_length_sum = 0;
```

```
        //calculate average wordlength
        for (int i = 0; i < textliste.length; i++) {
            System.out.println(textliste[i]);
            word_length_sum = word_length_sum + textliste[i].length();
        }

        return word_length_sum / textliste.length;
    }

    public static void main(String[] args) {

        // input filename
        String dateiname = readString();
        write(avgWordLength(dateiname));


    }
}
```

Aufgabe 5.2 (Ü) Fakultät

Die Fakultätsfunktion ist für positive Zahlen wie folgt definiert:

$$n! = n \cdot (n-1) \cdot \ldots \cdot 1 \qquad (n \geq 1)$$
$$0! = 1$$

Schreiben Sie ein Programm `Fak.java`, das die Fakultätsfunktion auf zwei Varianten berechnet:

a) *iterativ* in der Funktion `static int fakIt(int n)` und
b) *rekursiv* in der Funktion `static int fakRec(int n)`.

Lösungsvorschlag 5.2

```
public class Fak extends MiniJava{
        public static int fakIt(int k){
                int result = 1;
                while(k>0){
                        result = result * k;
                        k = k - 1;
                }
                return result;
        }
        public static int fakRec(int n){
                if(n == 0){
                        return 1;
                }else{
                        return n*fakRec(n-1);
                }
        }
        public static void main(String[] args) {
                write("Iterativ:"+fakIt(5));
                write("Rekursiv:"+fakRec(5));
        }
}
```

Aufgabe 5.3 (Ü) Permutationen eines Strings

Schreiben Sie eine rekursive Methode, welche als Parameter einen String der Länge n akzeptiert und dessen $n!$ Permutationen ausgibt. Überlegen Sie sich dazu wie Sie etwa für die Zeichenfolge abc die Permutationen

```
abc acb bac bca cab cba
```

ausrechnen würden, und verallgemeinern Sie Ihre Überlegung auf n stellige Zeichenketten. Sie können für Ihren Algorithmus die folgenden String-Methoden verwenden:

- substring(i,j): Gibt den Teilstring von Stelle i bis Stelle j zurück.
- charAt(i): Gibt das i-te Zeichen eines Strings zurück.

Lösungsvorschlag 5.3

```
public class Anagramm extends MiniJava{

    public static void main(String[] args) {
        permute("", "abcde");
    }

    public static void permute(String fixed, String variable) {
        if (variable.length() == 0) {
            // sind wir am Ende angelangt, geben wir den String aus
            System.out.print(fixed + ", ");
        } else {
            for (int i = 0; i < variable.length(); i++) {
                // fuer die naechste Position also eins nach dem fixed
                   String
                // wird jeder verbleibende Buchstabe durchprobiert
                String newfixed = fixed + variable.charAt(i);
                // und aus dem variablen String entfernt
                String newvariable = variable.substring(0, i) +
                   variable.substring(i + 1);
                // fuer dieses neue Paar wird wieder dieselbe Methode
                   angewendet
                permute(newfixed, newvariable);
            }
        }
    }
}
```

Aufgabe 5.4 (H) Matrix-Multiplikation

Vektoren und Matrizen lassen sich durch Arrays darstellen. Eine Matrix ist hier durch einen Vektor von Zeilen gegeben. D.h. der Eintrag `m[2][0]` steht in der dritten Zeile an erster Stelle. Im folgenden sollen Methoden zur Multiplikation von Matrizen und Vektoren erstellt werden.

a) Die Multiplikation zweier Vektoren ist folgendermaßen definiert:

$$\begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \sum_{j=1}^{n} a_j \cdot b_j$$

Schreiben Sie eine Methode `static int vecvecmul(int[] a, int[] b)`, welche die Multiplikation zweier Vektoren implementiert.

b) Die Multiplikation einer Matrix mit einem Vektor ist folgendermaßen definiert:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{pmatrix}$$

wobei die Einträge c_i gegeben sind durch

$$c_i = \sum_{j=1}^{n} a_{ij} \cdot b_j.$$

Schreiben Sie eine Methode `static int[] matvecmul(int[][] a, int[] b)`, welche die Multiplikation einer Matrix und eines Vektors implementiert.

c) Die Transposition einer Matrix ist folgendermaßen definiert:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}^{\top} = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{pmatrix}$$

Schreiben Sie eine Methode `static int[][] transpose(int[][] a)`, welche die transponierte zur Matrix a zurückgibt.

d) Die Multiplikation zweier Matrizen ist folgendermaßen definiert:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1k} \\ b_{21} & b_{22} & \dots & b_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nk} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1k} \\ c_{21} & c_{22} & \dots & c_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ c_{m1} & c_{n2} & \dots & c_{mk} \end{pmatrix}$$

wobei die Einträge c_{ij} gegeben sind durch

$$c_{ij} = \sum_{l=1}^{n} a_{il} \cdot b_{lj}.$$

Schreiben Sie eine Methode `static int[][] matmatmult(int[][] a, int[][] b)`, welche das Produkt der Matrizen a und b zurückgibt.

e) Schreiben Sie eine Mehtode `main()` um Ihre Methoden zu testen.

Lösungsvorschlag 5.4

```
public class MatmultOhneExceptions {


    public static int vecvecmul(int[] v1, int[] v2) {
        int res = 0;
        for (int i = 0; i < v1.length; i++)
            res += v1[i] * v2[i];
        return res;
    }

    public static int[] matvecmul(int[][] m, int[] v) {
        int[] resvec = new int[m.length];
        for (int i = 0; i < resvec.length; i++)
            resvec[i] = vecvecmul(m[i], v);
        return resvec;
    }
    public static int[][] transpose(int[][] m) {
        int[][] resmat = new int[m[0].length][m.length];
        for (int i = 0; i < resmat.length; i++)
            for (int j = 0; j < resmat[0].length; j++)
                resmat[i][j] = m[j][i];
        return resmat;
    }
    public static int[][] matmatmul(int[][] a,int b[][]){
        int [][] resmat = new int[a.length][];
        int [][] transposed = transpose(b);
        for (int i = 0; i < resmat.length; i++)
            resmat[i]=matvecmul(a,transposed[i]);
        return transpose(resmat);
    }
    public static void printmat(int[][] resmat){
        for (int i = 0; i < resmat.length; i++) {
            System.out.print("| ");
            for (int j = 0; j < resmat[0].length; j++)
                System.out.print(resmat[i][j]+" ");
            System.out.println("|");
        }
    }

    public static void main(String [] args){
        int[][] matrix1 = {{1,0,0},{1,0,0},{0,1,0},{0,0,1}};
        int[][] matrix2 = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
        int[][] resmat = matmatmul(matrix1,matrix2);
        printmat(matrix1);
        System.out.println("*");
        printmat(matrix2);
        System.out.println("=");
        printmat(resmat);
    }

}
```