Arbeitsgruppe Datenbanksysteme
M. Gogolla, R. Kollmann, A. Lattner, S. Prelle,
M. Richters, H. Shi, A. Werner

2. April 2001

Vorbereitungswoche Softwarepraktikum 2001

Übungen zu Java (Teil 1)

Der Quell-Code zu einigen Beispielen findet sich unter:
`/home/dbs/lehre/swp2001/vw/java1/beispiele`

Die Folien vom Vormittag sind über den WWW-Server der AG Datenbanksysteme verfügbar:
`http://db.informatik.uni-bremen.de/swp2001`

Aufgabe 1:

Schreiben Sie das "Hello World" Programm. Übersetzen Sie es mit dem JDK 1.3.

Aufgabe 2:

Schreiben Sie ein Programm, welches Liter in pint umrechnet. Dabei gilt: 1 Liter = 1.7607 pint. Das Programm soll Benutzereingaben von der Konsole lesen und dann die Umwandlung vornehmen. Das folgende Beispiel demonstriert, wie man Benutzereingaben mit Ausnahmebehandlung programmieren kann.

```java
import java.io.*;
public class ReadNumber {
    public static void main(String args[]) {
        try {
            BufferedReader reader
                = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Please enter number: ");
            System.out.flush();
            String line = reader.readLine();
            int num = Integer.parseInt(line);
            System.out.println("Your input: " + num);
        } catch (NumberFormatException e) {
            System.out.println("Invalid number format: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Error: " + e.toString());
        }
    }
}
```

Aufgabe 3:

Schreiben Sie ein Programm zur Verwaltung von Personen (die über Nachnamen, Vornamen, Geschlecht und Geburtsjahr verfügen sollen). Folgende Funktionen sollen bereitgestellt werden:

- Einfügen einer neuen Person
- Ausgeben einer Liste aller Personen
- Suche nach einer bestimmten Person über ihren Nachnamen

Überlegen Sie zuerst, welche Klassen Sie benötigen und welche Methoden diese haben müssen. Beginnen Sie erst dann mit der Implementierung.

Aufgabe 4:

Erweitern Sie das Programm aus Aufgabe 3 so, daß dargestellt werden kann, wer die Eltern einer Person sind. Es soll dazu folgende neue Funktionen geben.

- Einfügen einer Elternbeziehung zwischen zwei bereits vorhandenen Personen, wobei die eine Vater oder Mutter der anderen ist.
- Ausgabe des kompletten Stammbaums (alle Personen, deren Eltern, Großeltern, usw.)

Beachten Sie dabei, daß eine Person immer nur direkter Nachfahre von zwei anderen Personen sein darf.

Aufgabe 5:

Java zwingt Programmierer dazu, Fehlerbehandlung durchzuführen. Dies geschieht zum einen dadurch, daß bei kritischen Methoden explizit angegeben wird, welche Fehler auftreten können und zum anderen dadurch, daß man bei Verwendung solcher kritischen Methoden immer angeben muß, was man beim Auftreten eines Fehlers oder eines speziellen Fehlers tun möchte.

```java
public class Division1 {

    public static int intDiv(int a, int b) {
        return (a / b);
    }

    public static void main(String args[]) {
        System.out.println("6 DIV 3 = " + intDiv(6,3));
        System.out.println("6 DIV 0 = " + intDiv(6,0));
        System.out.println("6 DIV 2 = " + intDiv(6,2));
    }
}
```

Die Ausführung des obigen Programmes führt zu folgender Ausgabe:

```
6 DIV 3 = 2
java.lang.ArithmeticException: / by zero
        at Division1.intDiv(Division1.java:4)
        at Division1.main(Division1.java:9)
```

Die Programmausführung bricht mit der Division durch Null ab. Ein Programmierer hat nun drei Möglichkeiten, diesen Fehler abzufangen:

a. Die Divisionsoperation in `intDiv` kann mit `try` und `catch` umgeben werden, um eine Exception abzufangen. Der Fehler könnte innerhalb der Divisionsmethode behandelt werden. Für die aufrufende Methode bliebe der Fehler allerdings unbemerkt.

b. In der `main`-Methode könnte jede einzelne Zeile mit `try` und `catch` umgeben werden. Dies würde viel Programmieraufwand bedeuten, ließe aber zu, daß das Programm ungehindert weiterläuft und ein Fehler von der aufrufenden Funktion behandelt werden kann.

c. Es ist auch denkbar, die drei Aufrufe von `intDiv` in der Methode `main` mit einem try / catch zu umgeben. Dann könnte zwar der Fehler behandelt werden, aber Befehle, die nach dem fehlerhaften Aufruf noch abzuarbeiten gewesen wären, würden nicht mehr ausgeführt werden.

Hier nun ein Beispiel für Möglichkeit 1:

```
public class Division2 {

    public static int intDiv(int a, int b) {
        try {
            return (a / b);
        } catch (ArithmeticException e) {
            System.err.println("Caught exception ‘" + e.getMessage() +
                               "’ in operation ‘" + a + " DIV " + b + "’");
            return -1;
        }
    }

    public static void main(String args[]) {
        System.out.println("6 DIV 3 = " + intDiv(6,3));
        System.out.println("6 DIV 0 = " + intDiv(6,0));
        System.out.println("6 DIV 2 = " + intDiv(6,2));
    }
}
```

Das Ergebnis nun:

```
6 DIV 3 = 2
Caught exception ‘/ by zero’ in operation ‘6 DIV 0’
6 DIV 0 = -1
6 DIV 2 = 3
```

Schreiben Sie ein Programm für einen Taschenrechner mit den Grundrechenoperationen, Wurzelziehen und Quadrieren. Ein- und Ausgabe sollen über die Standardein- und ausgabe erfolgen.

Schreiben Sie das Programm fehlertolerant durch die Verwendung von Exceptions. Für diese Aufgabe soll nicht die Eingabe überprüft, sondern lediglich auf auftretende Fehler reagiert werden.

Aufgabe 6:

Die Klasse ArrayList (`java.util.ArrayList`) implementiert ein dynamisches Array von Objekten. Es empfiehlt sich somit immer dann, wenn ein Array benötigt wird, dessen Größe sich zur Laufzeit verändert.

Die Klasse StringTokenizer (`java.util.StringTokenizer`) zerlegt eine Zeichenkette unter Verwendung von angegebenen Trennzeichen. Es ist somit z.B. möglich, einen String in seine Worte zu zerlegen:

```
String str = "Aus den Augen, aus dem Sinn.";
StingTokenizer tokens = new StringTokenizer(str, " ,");
while ( tokens.hasMoreTokens() )
    System.out.println(tokens.nextToken());
```

In diesem Beispiel dienen das Leerzeichen und das Komma als Trennzeichen. Die Ausgabe wäre demnach:

```
Aus
den
Augen
aus
dem
Sinn.
```

Schreiben Sie ein Programm, welches die Funktion des Unix-Kommandos `wc` nachempfindet. Es soll also eine Textdatei eingelesen werden, die Anzahl der Zeichen, Zeilen und Wörter gezählt und das Resultat am Ende ausgegeben werden. Benutzen Sie dabei statt StringTokenizer die Klasse `java.io.StreamTokenizer` und für das Einlesen von Dateien die Klasse `java.io.FileReader`.

Aufgabe 7:

Entwerfen Sie ein interface für den Datentyp Stack. Schreiben Sie Klassen `ArrayStack` und `ListStack`, die das Stack-interface implementieren. Die Klasse `ArrayStack` soll eine `ArrayList` als Datenstruktur für die Stack-Elemente verwenden, während in `ListStack` eine `LinkedList` eingesetzt werden soll. Überlegen Sie Vor- und Nachteile beider Implementierungen. In welchen Situationen würden Sie eher einen `ArrayStack` verwenden, in welchen eher einen `ListStack`?