technische universität
dortmund

Bearbeitet von:
Sebastian Hauer
Sergej Karsten

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Iman Kamehkhosh, Marcel Preuß, Henning Timm,
Alexander Lochmann, Bartho Rudak, Thomas Schmitz

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2014/15

Übungsblatt 5

Bearbeitungszeit:
24.11.14 -
28.11.14

Aufgabe 5.1 – Standardaus- und eingabe

In den bisherigen Übungsblättern im Praktikum haben Sie zum Ausgeben von Text eine Methode namens `System.out.println();` verwendet. Das von der Java-Standard-Bibliothek bereitgestellte Modul `System` bietet Zugang zu sogenannten „Streams“ — Objekten, mit denen Daten übertragen werden können. Der Standard-Ausgabe-Stream wird mit `System.out` erreicht und man kann mit der Methode `println()` Text in die Standard-Ausgabe auf der Konsole schreiben.

Wenn wir Daten einlesen wollen, bietet der Standard-Eingabe-Stream `System.in` Möglichkeiten, Daten einzulesen. Die eingelesenen Daten — hauptsächlich Zahlen oder Zeichen, die Sie auf der Konsole eingeben — müssen jedoch erst verarbeitet werden. Damit Sie diese Arbeit nicht von selbst erledigen müssen, bietet die Java-Standard-Bibliothek eine Programmkomponente namens `Scanner`, welche einen Datenstrom analysiert und eingelesene Daten damit für Sie in einen passenden Wert umwandelt.

Die Funktionalität des `Scanner`s können Sie mit dem Schlüsselwort `import` zwischen Ihrer `package` und Ihrer `class` Anweisung im Quellcode importieren. Die genaue Codezeile lautet:

```
import java.util.Scanner;
```

Um nun einen `Scanner` zu verwenden, müssen Sie in ihrem Quellcode eine Variable vom Typ `Scanner` anlegen und mit der Programmanweisung `new Scanner(System.in)` initialisieren und ihm dabei mitteilen, dass er die Standard-Eingabe verarbeiten soll. Die genaue Codezeile für die Verwendung eines Scanners lautet also:

```
Scanner scanner = new Scanner(System.in);
```

Sie können im Anschluss mit der Variablen `scanner` (Groß-, Kleinschreibung beachten) Daten aus der Konsole einlesen, indem Sie die Methoden `nextInt()`, `nextBoolean()`, `nextLine()` und `nextDouble()` verwenden. Der Aufruf einer solchen Methode funktioniert durch eine Codezeile, wie zum Beispiel:

```
int eingabe = scanner.nextInt();
```

Der Programmfluss wird an dieser Stelle halten, bis Sie in der Konsole einen Wert eingegeben haben und mit der `Enter`-Taste in den Stream geschickt haben. Der Scanner wertet Ihre Eingabe aus und versucht diese zu einem Integer-Wert um zu formen.

Schreiben Sie eine Klasse mit dem Namen „EchoTest“, in deren **main**-Methode Sie eine Variable mit dem Typ Scanner anlegen und einen Integer einlesen lassen, welchen Sie anschließend ausgeben (im Fachjargon „echo“-en genannt).

Sorgen Sie anschließend dafür, dass Sie den Benutzer nach Programmstart mit einer höflichen Willkommensnachricht begrüßen und ihn zu einer Eingabe auffordern, bevor Sie mit dem Scanner

seine Eingabe abfragen. Formatieren Sie anschließend die Ausgabe entsprechend mit einer Nachricht wie „Sie haben folgende Zahl eingegeben: “.

Geben Sie anschließend während der Ausführung des Programms keine ganze Zahl ein, sondern ein Zeichen oder eine Fließkommazahl und beschreiben Sie, was passiert:

Diskutieren Sie mit der Praktikumsgruppe diese Fehlermeldung.

Aufgabe 5.2 – Programmflusskontrolle mit `for`

Aus der Vorlesung sollten Ihnen nun auch `for`-Schleifen bekannt sein.

Als Hilfestellung ist hier noch einmal das syntaktische Grundgerüst gegeben:

```
for(Initialisierung; Bedingung; Fortschritt){
    ...
}
```

Erstellen Sie eine neue Klasse mit dem Namen „HumbleSum“ und schreiben Sie eine main-Methode, den Benutzer begrüßt und in einer `for`-Schleife den Nutzer zehn mal nach einer Zahl fragt und die Eingabe des Benutzers aufaddiert. Nach Austritt aus der `for`-Schleife wird die Summe ausgegeben.

Schreiben Sie anschließend die Schleife so um, dass sie genau dann terminiert, wenn der Nutzer eine `0` eingibt und ansonsten immer weiter nach Zahlen fragt und diese aufsummiert. Die Ausgabe der Summe soll dabei weiterhin erst nach Verlassen der Schleife ausgegeben werden. Bietet sich für diese Umformung eventuell eine andere Schleifenform an?

Aufgabe 5.3 – Umformulierungen

Notieren Sie die 4 Möglichkeiten, den Wert einer ganzzahligen Variable `x` um eins zu erhöhen.

Schreiben Sie ein Programmfragment, welches eine Variable `x` danach überprüft, ob sie größer als 10 ist. Falls ja, soll der Wert dieser Variablen um 5 erhöht, falls nein, soll der Wert dieser Variablen um 5 verringert werden.

Sie können jede `while` und `do ... while`-Schleife auch als `for`-Schleife schreiben.

Schreiben Sie folgendes Programm so um, dass es anstatt einer `while`- eine `for`-Schleife verwendet und versuchen Sie, die Syntax der `for`-Schleife so zu verbessern, dass deutlicher wird, was berechnet werden soll. Notieren Sie sich dafür zuerst, in **einem** Satz, was das Programm berechnet:

```java
public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    int n = scanner.nextInt();
    int s = 0;
    while(true){
        if(2 > n){
            break;
        }
        if (n % 2 == 0){
            s += n;
        }
        n--;
    }
    System.out.println(s);
}
```

Aufgabe 5.4 – Mehr Fallunterscheidungen

Die `switch-case` Programmstruktur ist Ihnen aus der Vorlesung bekannt.

Schreiben Sie ein Programm in einer Klasse namens „Months“, welches eine Zahl aus der Standard-Eingabe einlesen lässt und im Falle, dass die Zahl zwischen 1 und 12 liegt, den entsprechenden Monatsnamen, der mit dieser Zahl assoziiert wird, ausgibt. Falls keine passende Zahl eingegeben wurde, soll eine Fehlermeldung ausgegeben werden.

Aufgabe 5.5 – Syntaxelemente

Wiederholen Sie, wie die Syntax eines Java-Programms strukturiert ist:

Wie deklarieren Sie eine Klasse?

An welcher Stelle beginnt das Programm mit seiner Ausführung und wie wird diese Stelle deklariert?

Wie wird eine Variable deklariert?

Wie werden Variablen Werte zugewiesen?

Wie können zwei numerische Werte miteinander verglichen werden?

Wie können zwei bool’sche Werte miteinander verknüpft werden?

Welche Schlüsselwörter leiten Programmstrukturen ein, die den Programmfluss steuern?

Mit welchem Schlüsselwort können Sie vor der Klassendeklaration andere Programmbibliotheken in ihrem Programm nutzbar machen?

Aufgabe 5.6 – Fibonacci

Die Fibonacci-Folge ist eine Folge von Zahlen, deren erste beiden Elemente den Wert 1 haben und deren weiteren Elemente jeweils die Summe der beiden vorherigen Elemente als Wert haben. So sind die ersten 6 Zahlen der Fibonacci-Folge: `1,1,2,3,5,8,...`

Schreiben Sie ein Programm, welches eine Zahl `n` über die Standard-Eingabe einliest und die ersten `n` Fibonacci-Zahlen über eine `for`-Schleife ausgibt.

Aufgabe 5.7 – 99 Bottles of Beer

Eine traditionelle Übungsaufgabe für diverse Programmiersprachen ist es, das traditionelle englische Volkslied „99 Bottles of Beer“ über eine Schleife ausgeben zu lassen.

Der Songtext lautet:

```
99 bottles of beer on the wall, 99 bottles of beer,
Take one down, pass it around, 98 bottles of beer on the wall.
98 bottles of beer on the wall, 98 bottles of beer, ...
..., 0 bottles of beer on the wall.
Go to the store, buy some more,
99 bottles of beer on the wall.
```

Schreiben Sie mithilfe einer `for`-Schleife ein Programm, welches den Songtext in seiner Vollständigkeit ausgibt. Beachten Sie dabei den Spezialfall, dass bei einer verbleibenden Flasche kein Plural von *bottle* verwendet wird.