

Übungsblatt 1

1 Allgemeines

In diesem Semester werden Sie in Rahmen des Faches Informatik II Übungen haben: zwei Stunden Laborübungen pro zwei Wochen und eine Stunde Seminarübung pro Woche. Die Laborübungen werden in Computersäle der FDIBA je nach dem Stundenplan durchgeführt. Leitern der Laborübungen sind Doz. Dr.-Ing. Dimitar Jetchev (Zimmer 12248, Tel. 2139, E-Mail: jetch@tu-sofia.bg) und Tutor Mihajlo Anđelković (E-Mail: michael.angelkovich@gmail.com).

Sie werden in Java programmieren. Dafür werden sie vor allem JDK 6.0¹ brauchen und dann noch einen Texteditor oder eine integrierte Entwicklungsumgebung wie JOE oder Eclipse². Es ist vollkommen egal was Sie zum Programmieren benutzen, solange die Programmen laufen. Dennoch sollte man in den Computersäle der FDIBA nichts ausser JDK, Eclipse und Notepad erwarten.

Als Referenz für verschiedene Paketen und Klassen können Sie die Dokumentation von `java.sun.com` benutzen. Ein leichter Weg zur gewünschten Webseite ist in Google Search das Wort „java“ und den Namen der Klasse einzutragen, deren Information Sie brauchen. Bevor Sie mit Programmieren beginnen, wäre es schön über Coding Conventions³ in Java zu lesen.

1.1 Eclipse

Eclipse als Werkzeug wird in den Laborübungen betrachtet. Wichtig ist, dass Sie an der FDIBA beim Hochfahren der Eclipse als Ihr Arbeitsraum (workspace) ein Datenverzeichnis (directory) auf Ihrem Z:\ nennen, sodass Ihre Programme von jedem Computer im Netzwerk erreichbar werden.

1.2 Hausarbeit

Hausarbeit ist Art der Seminararbeit, die aus zwei Teile besteht. Es ist wichtig den Stoff aus den Übungen und Vorlesungen zu verarbeiten und zu verstehen, da er eng mit der Hausarbeit verbunden wird. Wenn man die Bonusaufgaben aus der Hausarbeit selbständig löst und das durch seine Antworten auf gestellte Fragen bestätigt, kriegt man Bonuspunkte für die Prüfung.

¹<http://java.sun.com/javase/downloads/>

²Eclipse Classic: <http://www.eclipse.org/downloads/>

³<http://java.sun.com/docs/codeconv/>

2 Erste Aufgaben

2.1 Hallo Welt

Nun werden Sie Ihre erste Übungsaufgabe machen. Schreiben Sie den folgenden Code in Ihrem Editor ab, und kompilieren Sie ihn.

```
public class HalloWelt {
    public static void main(String[] args) {
        System.out.println("Hallo Welt");
    }
}
```

Name der Datei, in der diese Klasse gespeichert ist, soll gleich wie der Name der Klasse sein (auch in Größe der Schrift) und Dateinamenserweiterung `*.java` haben — `HalloWelt.java` in diesem Fall. Nach dem Kompilieren der Datei mit `javac.exe`, bekommt man eine `*.class` Datei, die man mit `java.exe` rennen kann. Die Eclipse versteckt diesen Teil im Ganzen vom Programmierer.

Wenn man das Programm startet, wird in der Befehlszeile (oder Konsole) der Text „Hallo Welt“ erscheinen. Wenn Sie dies zum ersten Mal machen, versuchen Sie nun den Text zu ändern und beobachten Sie die Resultaten. Methode `println` aus `System.out` können Sie ab nun zum Schreiben auf `stdout` benutzen. Der Name `stdout` steht für „Standard Output“ und ist ein der Standard-Datenströme (standard data streams) in meist der heutigen Betriebssystemen. Ähnlich steht `stdin` für „Standard Input“.

Eine andere Methode zum Schreiben auf `stdout` in Java ist `printf`. Sie ist auch von `System.out` zu rufen und ist der schon bekannten `printf`-Funktion aus C sehr ähnlich. Die Dokumentation können Sie sich auf `java.sun.com` schauen (Klasse `Formatter`⁴).

2.2 Ein Applet

Bevor wir mit den Übungen weitermachen, werden wir kurz noch ein Hello-Welt-Programm schreiben, indem man sich noch eine Art von Java-Anwendungen kennenlernt: Java Applet. Java Applets sind stand-alone⁵ Anwendungen, die in Kontext eines anderen Programms rennen: meistens in einem Webbrowser. Lokal kann man sie mit `appletviewer.exe` rennen. Schreiben wir also einen Hallo-Welt-Applet.

```
import java.applet.*;
import java.awt.*;

public class HalloWeltApplet extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hallo Welt", 50, 50);
    }
}
```

⁴<http://java.sun.com/j2se/1.5.0/docs/api/java/util/Formatter.html>

⁵autonom, selbstständig, unabhängig

Hierbei erscheinen die neue Schlüsselwörter: `import` und `extends`, während die neue Klassen `Applet` und `Graphics` sind. Von den werden wir nun nur `import` besprechen, das sich sehr grob als Äquivalent von `include` aus C und C++ darstellen lässt. Dieses Schlüsselwort ermöglicht, dass man schon geschriebene Klassen und Paketen in einer Klasse lokal verfügbar macht. Obwohl dieses Schlüsselwort gar nicht notwendig ist, um irgendein Programm zu schreiben, macht es das Programmieren bequemer und übersichtlicher. Zum Beispiel würde unser Applet ohne `import` so lauten:

```
public class HalloWeltApplet extends java.applet.Applet {
    public void paint(java.awt.Graphics g) {
        g.drawString("Hallo Welt", 50, 50);
    }
}
```

2.3 Quadratgleichung

Nun ist die Zeit gekommen, ein längeres Programm zu schreiben. Nämlich handelt es sich von der Lösung einer Quadratgleichung. Sie sollen zwei Lösungsvarianten entwickeln:

1. Variante, die nur zwei reellen Lösungen drückt auch wenn sie gleich sind. (`Quadratgleichung1.java`)
2. Variante, die auch alle andere Fällen behandelt: keine Lösung, eine Lösung, zwei Lösungen, unendlich viel Lösungen. (`Quadratgleichung2.java`) Versuchen Sie das Programm möglichst kurz zu machen während es immer noch übersichtbar bleibt und den erwähnten Coding Konventionen entspricht. Imaginäre Werte werden Sie nicht speziell verarbeiten müssen. Falls sie erscheinen (Diskriminante kleiner null), drucken Sie einfach den Text „Imaginäre Werte!“ und brechen Sie die Methode mit einer `return`; Anweisung.

Da Java im Prinzip für Sie etwas neues ist, **wird diesmal auch eine Musterlösung der zweiten Variante im Aufgabebblatt vorgelegt werden**. Merken Sie sich die Klasse `Scanner` aus dem Paket `java.util.*` und wie sie benutzt wird (hierbei zum lesen von `stdin` d.h. `System.in`). Es ist angenommen, dass man versteht, wie der Algorithmus zur Lösung einer Quadratgleichung läuft und somit werden eher die technische Aspekte der Lösung besprochen, die man bei den weiteren Übungen brauchen wird.

```
import java.util.*;

public class Quadratgleichung2 {
    public static Scanner stdin = new Scanner(System.in);
```

Hier ist ein `stdin`-Objekt als publisches und statisch definiert. Das heißt, dass es nun von jeder Methode dieser Klasse sowie von jeder anderen Klasse benutzt sein kann.

```
    public static void main(String[] args) {
        double a, b, c; // Variablen für  $ax^2 + bx + c = 0$ 

        System.out.println("Geben sie die Werte a,b,c fuer  $ax^2 + bx + c = 0$  ein:");
        a = stdin.nextDouble();
        b = stdin.nextDouble();
        c = stdin.nextDouble();
```

Hier sieht man wie die Klasse `Scanner` zur Eingabe der reellen Zahlen von der Tastatur benutzt wird.

```
if(Math.abs(a) > 1e-4) {
```

Dies ist etwas wichtiges. Normalerweise würde hier eine $a \neq 0$ stehen. Doch wird $|a| > 10^{-4}$ benutzt. Wieso? Obschon der Typ `double` eine sehr breite Skala der Werte darstellen kann, ist er immer noch diskret und kleine Fehler können bei jeder Rechnung oder Eingabe entstehen. Wegen der akkumulativen Natur dieser Fehler, ist es empfehlenswert **nie** zwei reelle Werte mit Operatoren `==` und `!=` zu vergleichen, sondern immer eine Umgebung ϵ (in diesem Fall 10^{-4}) zu benutzen. Zum Beispiel, statt $a == 5$ würde so $|a - 5| < \epsilon$ stehen. Diese ϵ können Sie natürlich als eine `public static final double` Variable irgendwo definieren, aber später darüber.

```
double diskriminante = b*b - 4*a*c;
if(Math.abs(diskriminante) > 1e-4) { // diskriminante != 0
    if(diskriminante < 0) {
        System.out.println("Imaginäre Werte!");
        return;
    }
    double wurzel = Math.sqrt(diskriminante);
```

Die Neuigkeit hier ist die Klasse `Math`. Sie enthält viele Methoden, die bei den Rechnungen nutzbar sind. Sehen Sie ihre Beschreibung auf java.sun.com⁶.

```
        System.out.printf(
            "Zwei Loesungen: %.2f und %.2f",
            (-b - wurzel) / (2*a),
            (-b + wurzel) / (2*a)
        );
    } else {
        System.out.printf(
            "Eine Loesung: %.2f",
            -b / (2*a)
        );
    }
} else {
    if(Math.abs(b) > 1e-4) { // b != 0
        System.out.printf("Lösung: %.2f", - c/b);
    } else if(Math.abs(c) > 1e-4) { // c != 0
        System.out.println("Keine Loesungen.");
    } else {
        System.out.println("Unendlich viele Loesungen.");
    }
}
}
}
```

Die Lösung der ersten Variante werden Sie durch Modifikationen der zweiten Variante kriegen.

2.4 Zusatzlinks

JOE	http://www.javaeditor.de/
Google Search	http://www.google.com/

⁶<http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Math.html>