

Übungsserie Nr. 0

Ausgabe: 23. Februar 2011

Keine Abgabe

Keine Bewertung

Übungen

Bitte melden Sie sich über die Webseite der Vorlesung^a zu den Übungen an. Dort finden sich auch regelmässig aktuelle Informationen zur Vorlesung, den Übungen, dem Testat und der Prüfung. Beachten Sie auch die Informationen zum Einführungskurs in Java.

^a<http://www.vs.inf.ethz.ch/edu/I2>

Unterlagen

Für diese Serie benötigen Sie das Archiv

<http://www.vs.inf.ethz.ch/edu/I2/downloads/u0.zip>

1. Aufgabe: HelloWorld.java

(1a) Falls Sie die Übungen auf Ihrem privaten System bearbeiten wollen, installieren^{1,2} Sie sich das Java Developer Kit (JDK) für Java SE 6³. In den Pools des D-ITET ist diese Version bereits installiert.

(1b) Entpacken Sie das Archiv dieser Serie und bringen Sie das darin enthaltene Programm *HelloWorld* auf der Kommandozeile zur Ausführung. Führen Sie dazu folgende Schritte aus:

```
$ javac u0a1/HelloWorld.java  
$ java u0a1.HelloWorld
```

¹<http://java.sun.com/javase/6/webnotes/install/index.html>

²<http://java.sun.com/javase/downloads/widget/jdk6.jsp>

³<http://java.sun.com/javase/6/docs/>

Der erste Schritt kompiliert das Programm und erzeugt die Datei *HelloWorld.class*. Diese enthält das übersetzte Programm in ausführbarer Form, in sogenanntem *Bytecode*. Mit dem zweiten Schritt wird die *Java Virtual Machine* gestartet und das Programm ausgeführt, indem die Methode `main` der Klasse `HelloWorld` aufgerufen wird. Beachten Sie, dass die Methode `main` als `public` und `static` deklariert sein muss. Beachten Sie ausserdem, dass die Klasse im package `u0a1` liegt. Das spiegelt sich sowohl durch das gleichnamige Verzeichnis als auch durch die entsprechende Deklaration in *HelloWorld.java* wieder.

(1c) Obwohl ein Texteditor und die oben genannten Programme hinreichend für die Entwicklung von Java-Programmen sind, bietet der Einsatz einer integrierten Entwicklungsumgebung viele Vorteile. In der Java-Welt gibt es z.B. *Eclipse*, *IntelliJIDEA*, *BlueJ*, *JBuilder*, *NetBeans* und *JCreator*. Sie können frei wählen, in welcher Umgebung Sie die Übungen bearbeiten wollen. Beachten Sie allerdings, dass Ihnen Ihr Tutor nur für Eclipse Hilfe anbieten kann. Machen Sie sich mit Ihrer Entwicklungsumgebung vertraut. Nutzen Sie dazu auch die zahlreichen Anleitungen und Foren, die es im Internet gibt. Erstellen Sie ein Projekt für das *Hello World* Beispiel und bringen Sie es in Ihrer Entwicklungsumgebung zur Ausführung.

2. Aufgabe: Das erste Java Program

(2a) Bringen Sie das im Archiv dieser Serie enthaltene Programm `u0a2.Main` in Ihrer Entwicklungsumgebung zur Ausführung.

(2b) Das Programm besteht aus zwei Teilen, einer Klasse `Main` und einer Klasse `Signum`. Beachten Sie, dass die `.java` Dateien so heissen müssen wie die jeweils darin enthaltene Klasse. Die Methode `signum` aus der Klasse `Signum` implementiert die *Signum*-Funktion

$$\text{sgn}(x) := \begin{cases} -1, & \text{if } x < 0 \\ 0, & \text{if } x = 0 \\ +1, & \text{if } x > 0 \end{cases}$$

Die Methode `main` aus der Klasse `Main` ruft `signum` für verschiedene Werte auf und gibt die Ergebnisse auf der Standardausgabe aus. Dadurch kann man überprüfen, ob die Implementierung von `signum` korrekt ist.

Ergänzen Sie die `main` um einen Aufruf von `signum` mit einer positiven Zahl und überprüfen Sie die Ausgabe Ihres Programms.

3. Aufgabe: Automatisiertes Testen

(3a) In der letzten Aufgabe haben Sie visuell überprüft, ob die Ergebnisse der `signum` Funktion korrekt sind. Dieser manuelle Ansatz ist bei grösseren Programmen mit vielen Tests allerdings sehr aufwändig und fehleranfällig. Daher ist eine automatisierte Verifikation von Testergebnissen zu bevorzugen. JUnit4⁴ ist eine Java-Bibliothek für diesen Zweck. Installieren Sie JUnit4 auf Ihrem System.

⁴<http://www.junit.org/>

(3b) Führen Sie die Tests aus der Klasse `Tests` aus. Geben Sie dazu folgende Befehle auf der Konsole ein:

```
$ export CLASSPATH=/usr/share/java/junit4.jar:.  
$ javac u0a3/*.java  
$ java org.junit.runner.JUnitCore u0a3.Tests
```

Beachten Sie, dass Sie dazu im Verzeichnis sein müssen, in dem das Verzeichnis `u0a3` liegt. Ausserdem müssen Sie ggf. den Pfad zur `junit4.jar`, der JUnit4 Bibliothek, entsprechend der Installation auf Ihrem System ändern.

Hinweis: unter Windows werden die Verzeichnisse im `CLASSPATH` durch Semikolon anstatt durch Doppelpunkt getrennt.

(3c) Finden Sie durch geeignete Internetsuche heraus, wie man die Tests in Ihrer Umgebung ausführen kann. Hinweis: Eclipse bringt die JUnit-Bibliothek bereits mit. Achten Sie allerdings darauf, dass sie die Version 4 von JUnit verwenden.

(3d) Ergänzen Sie die Tests um einen Fall mit einer positiven Zahl und führen Sie die Tests in Ihrer Entwicklungsumgebung aus.

(3e) Verändern Sie die Implementierung von `signum`, so dass manche Tests fehlschlagen. Führen Sie die Tests in Ihrer Entwicklungsumgebung aus und analysieren Sie die Ausgaben. Machen Sie sich damit vertraut, wie Sie innerhalb Ihrer Entwicklungsumgebung schnell von den Ausgaben fehlgeschlagener Tests zu deren Ursache gelangen können.

4. Aufgabe: Gerichtete Graphen

Ein gerichteter Graph besteht aus einer Menge von Kanten und einer Menge von Knoten, wobei jede Kante als Pfeil dargestellt wird, welcher einen Ausgangsknoten mit einem Zielknoten verbindet. Knoten stellt man i.A. in Form von Kreisen dar. Betrachten Sie die folgende Problemstellung:

Sie besitzen drei Kannen; die erste fasst 8 Liter und ist bis zum Rand mit Wein gefüllt, die zweite fasst 5 Liter und ist leer, die dritte fasst 2 Liter und ist ebenfalls leer. Dies ist der Startzustand. Eine Umschüttung von Wein aus einer Kanne in eine andere führt von einem Zustand in einen anderen, wobei entweder eine Kanne vollständig geleert oder vollständig gefüllt werden muss. Ein Zustand wird durch das Tripel (a, b, c) kodiert, wobei a der Inhalt der 8-Liter-Kanne, b der Inhalt der 5-Liter-Kanne und c der Inhalt der 2-Liter-Kanne ist.

(4a) Zeichnen Sie den Graphen, der aus allen möglichen Zuständen (= Knoten) und möglichen Umschüttungen (= gerichteten Kanten) besteht. Markieren Sie die Knoten des Graphen mit den Zuständen, die durch die Folge der Umschüttungen auftreten. Achten Sie darauf, dass Knoten mit gleichem Tripel nicht mehrfach im Graphen auftreten.

(4b) Versehen Sie die Knoten des Graphen mit der minimalen Anzahl von benötigten Umschüttungen, um den entsprechenden Zustand vom Startzustand ausgehend zu erreichen. Wie gehen Sie dabei vor?

(4c) Ausgehend vom Startzustand, was ist die maximale Anzahl von Umschüttungen die benötigt wird, um ein beliebigen Zustand zu erreichen? Was ist die mittlere Anzahl?