Dr. Annette Bieniusa
Mathias Weber, M. Sc.
Peter Zeller, M. Sc.

TU Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Übungsblatt 1: Software-Entwicklung 1 (WS 2017/18)

Ausgabe: 24. Oktober 2017
Abgabe: keine

Das erste Übungsblatt soll Sie mit grundlegenden Funktionalitäten vertraut machen, die für die Bearbeitung der nächsten Übungsblätter benötigt werden.
Beachten Sie bitte folgende Punkte, bevor Sie mit der Bearbeitung des Übungsblatts beginnen:

1. Sofern noch nicht geschehen, beantragen Sie bitte einen Account beim SCI (Gebäude 48, Erdgeschoss). Ein solcher Account wird für die Übungsstunden im Rahmen der Vorlesung *SE-1* benötigt.
2. Sofern noch nicht geschehen, melden Sie sich bitte bei einer der Übungsgruppen zur Vorlesung an. Alle Details dazu sind auf der Internetpräsenz der AG unter https://softech.cs.uni-kl.de/se1/ws17 zu finden.

Zur Beantwortung von Fragen und Hilfe bei Problemen stehen Ihnen die Tutoren in der ersten Woche *zu den Zeiten der zehn Übungsgruppen* im Terminalraum von Gebäude 48 (48-211) zur Verfügung. Nehmen Sie diese Gelegenheit war, um sich mit Ihrer Arbeitsumgebung und den Werkzeugen vertraut zu machen!

Aufgabe 0 Hinweise zu den Übungen und Plagiaten

Programmieren lässt sich nur durch praktische Anwendung erlernen. Daher legen wir großen Wert darauf, dass Sie die Übungen selbst bearbeiten. Insbesondere werden Plagiate an der Universität nicht akzeptiert.

1. Die Übungen werden in Teams von je etwa 5 Studenten bearbeitet. Wir erwarten, dass jeder im Team an der Lösung der Aufgaben mitarbeitet und dass Lösungen im Team diskutiert werden, so dass jeder im Team alle Aufgaben erklären kann.

   Zu jedem Übungsblatt gibt es eine sogenannte Abnahme, bei dem sich das Team mit der Tutorin oder dem Tutor trifft und Fragen zur Abgabe beantwortet werden müssen. Wenn in der Abnahme klar wird, dass ein Mitglied des Teams eine Lösung nicht erklären kann, werden individuell Punkte abgezogen.

2. Wenn Sie sekundäre Quellen, wie Bücher oder das Internet verwenden, müssen Sie immer die Quelle angeben. Das einfache Kopieren aus anderen Quellen ist für die Übungen nicht gestattet. Wenn Sie andere Quellen benutzen, versuchen sie diese erst zu verstehen und dann die Idee selbstständig umzusetzen. Den größten Lerneffekt erhalten Sie aber natürlich durch komplett eigenständiges Lösen der Aufgaben.

   Wenn wir in einer Übungs-Abgabe kopierten Code finden, wird die gesamte Abgabe mit 0 Punkten bewertet.

3. Sie können Übungsaufgaben gerne mit den Mitgliedern anderer Teams diskutieren. Sie sollten jedoch Ihren Code nie Mitgliedern von anderen Teams zeigen.

4. Wenn Code von anderen Teams kopiert wurde, werden die Abgaben **von beiden Teams** mit 0 Punkten bewertet.

5. Wir behalten uns vor Punkte auch nachträglich abzuziehen, wenn ein Verstoß erst später bemerkt wird.

Weitere Hinweise zum Übungs-System und den Voraussetzungen zur Klausurzulassung finden Sie unter https://softech.cs.uni-kl.de/se1/ws17/offiziell/uebungen.

Aufgabe 1 Die Shell

Diese Aufgabe bezieht sich auf UNIX (-artige) Betriebssysteme, wie zum Beispiel Linux oder Mac OS. Wenn Sie auf Ihren eigen Rechnern Windows verwenden, können Sie sich zusätzlich über die entsprechenden Windows-Befehle informieren.

Neben den heute üblichen grafischen Benutzeroberflächen kann ein Anwender auch mittels der *Shell*[1] Programme ausführen und so mit dem Rechner interagieren. Dies geschieht durch Eingabe des Programmnamens, dem gegebenenfalls noch weitere Parameter folgen, gefolgt vom Drücken der `RETURN`-Taste.

Beispiel: Der Aufruf des Programms `ls` listet die Dateien im aktuellen Verzeichnis auf.

a) Machen Sie sich mit den Prinzipien und dem Aufbau vertraut, der hinter Dateiverzeichnissen in UNIX (-artigen) Betriebssystemen steht, insbesondere mit dem Kommando `cd` und der Bedeutung von '.' und '..' im Kontext von Verzeichnissen.

b) Was ist der Unterschied zwischen *absoluten* und *relativen Dateipfaden*?

c) Machen Sie sich mit der Funktionalität der Kommandos `pwd`, `ls`, `mkdir`, `cp`, `mv` und `rm` durch Lesen der sogenannten *Man-Pages*[2] vertraut (die Man-Page zu einem bestimmten Kommando wird mittels

`man` <*Name des Kommandos*>

angezeigt). Beschreiben Sie die Funktionalität jedes Kommandos *kurz* mit eigenen Worten. Was bedeuten die Kürzel der Kommandos?

Tipp: Es lohnt sich neben dem eigentlichen Kommando auch gängige Parameter zu kennen. Besonders nützlich für das Kommando `ls` sind beispielsweise `-l`, `-h` und `-a`. Sie lassen sich auch einfach kombinieren: `ls -alh`. Schauen Sie sich in der Man-Page zu `rm` auch den Parameter `-r` an, für `mkdir` den Parameter `-p` und finden Sie heraus was `cd` ohne jeglichen Parameter macht.

Aufgabe 2 Text-Editoren und der Java-Compiler

Für die Eingabe von Texten im Allgemeinen und Programmcode im Speziellen wird ein sogenannter *Text-Editor* verwendet. Es gibt eine Vielzahl verschiedener Editoren, welche zum Bearbeiten von Programmen benutzt werden können. Programme wie Word sind hingegen nicht geeignet, da Sie den eingegeben Text zusammen mit Informationen zum Aussehen und Layout des Textes in einem anderen Format speichern. Einfache Editoren wie "Notepad" können zwar theoretisch verwendet werden, sind aber nicht für das Programmieren ausgelegt. Für diese Übung verwenden wir den Editor Visual Studio Code[3], welcher bereits auf den SCI-Rechnern installiert ist.

- Öffnen Sie ein neues Terminal und führen Sie folgendes Kommando aus:

  ```
  code Hello.java
  ```

  Es öffnet sich der Editor Geany, und Sie können die Datei `Hello.java` editieren. Hat diese Datei noch nicht existiert, können Sie sie nun durch Speichern in Geany (*shortcut:* `STRG-S`) anlegen.

- Geben Sie folgenden Text im Editor ein und speichern Sie die Datei:

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

[1] Manchmal wird, insbesondere im Kontext grafischer Benutzeroberflächen, auch vom sogenannten *Terminal* gesprochen.
[2] Die abkürzende Bezeichnung *Man-Page* steht für *Manual-Page*, also quasi die Anleitung zum jeweiligen Kommando.
[3] Download von Visual Studio Code unter https://code.visualstudio.com/

- Übersetzen Sie die Datei im Terminal mit dem Java-Compiler durch Eingabe folgenden Kommandos:

  ```
  javac Hello.java
  ```

  Ist ein Fehler in der Datei, dann sehen Sie eine Fehlermeldung. Zum Beispiel bekommen Sie die folgende Meldung, wenn Sie in Zeile 4 das Wort "println" mit einem Großbuchstaben am Anfang schreiben.

  ```
  Hello.java:4: cannot find symbol
  symbol  : method Println(java.lang.String)
  location: class java.io.PrintStream
          System.out.Println("Hello World!");
                     ^
  1 error
  Exit 1
  ```

  In einer Fehlermeldung finden Sie immer den Dateinamen, gefolgt von einem Doppelpunkt und der Zeilennummer. Danach steht dann eine Beschreibung des Fehlers.

  Wenn Sie alles richtig gemacht haben, sollten Sie keine solche Fehlermeldung sehen. Die Ausgabe ist dann entweder leer oder zeigt eventuell gesetzte Optionen an, wie zum Beispiel die folgende:

  ```
  Picked up _JAVA_OPTIONS: -Xms512m -Xmx512m
  ```

- Nach dem Ausführen des Java-Compilers sollte sich nun eine Datei namens "Hello.class" im aktuellen Verzeichnis befinden. Verwenden Sie den Befehl "`ls`", um dies zu überprüfen.

  Sie können das neu erstellte Programm `Hello` nun im Terminal ausführen, durch Eingabe von:

  ```
  java Hello
  ```

  Es sollte dann die Ausgabe "Hello World!" angezeigt werden.

Tipps zur Arbeit mit Visual Studio Code und dem Terminal:

- Der Befehl `code` zum starten von Visual Studio Code wartet nicht, bis das Programm beendet wird. Bei manchen Editoren ist dies anders. Zum Beispiel wartet der Editor `geany` bis das Programm beendet wurde und blockiert so lange das Terminal. Durch das Anfügen von `&` an einen Befehl können Sie das Terminal direkt weiter benutzen!
- Haben Sie bereits ein Programm gestartet, welches nun das Terminal blockiert, können Sie es mit `STRG-Z` kurz unterbrechen und mit dem Befehl `bg` im Hintergrund weiter laufen lassen. Damit erhalten Sie wieder die Kontrolle über das Terminal. (Das duale Kommando `fg` holt es wieder in den Vordergrund.)
- Reagiert ein Programm im Terminal nicht mehr auf Eingaben, oder haben sie zum Beispiel ein selbst geschriebenes Programm ausgeführt, das nicht mehr terminiert, können Sie es mit `STRG-C` beenden!
- Beim Entwickeln eines Programms lohnt es sich sowohl den Editor als auch das Terminal ständig offen zu halten. Änderungen am Programm können Sie schnell mit `STRG-S` speichern. Mit `ALT-TAB` können Sie zwischen Fenstern wechseln und so zum Terminal gelangen. Visual Studio Code enthält auch ein integriertes Terminal, welches Sie verwenden können um mehrere Fenster zu vermeiden.
- Die meisten Terminals besitzen eine Historie, die Sie mit den Pfeil-Hoch bzw. Pfeil-Runter Tasten durchlaufen können. Damit können Sie zum Beispiel einen Aufruf des Compilers, nach einer Änderung an Ihrem Programm, schnell erneut ausführen.
- Andere beliebte Editoren neben Visual Studio Code sind zum Beispiel: Vim, Emacs, Atom, Geany, Sublime Text, Notepad++ oder Gedit. Außerdem gibt es noch sogenannte integrierte Entwicklungsumgebungen (IDEs), wie IntelliJ IDEA, Netbeans oder Eclipse.
- Da Sie beim Programmieren viel mit Ihrem Editor arbeiten, sollten Sie sich mit dessen Features und Shortcuts vertraut machen. Eine Übersicht zu Visual Studio Code finden Sie zum Beispiel unter https://code.visualstudio.com/docs/getstarted/tips-and-tricks.