Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. D. Sabel, Dr. P. Wendler

WS 2018/19
Übungsblatt 1

Übungen zu Einführung in die Informatik: Programmierung und Software-Entwicklung:

Lösungsvorschlag

Aufgabe 1-1 Arbeiten mit Java ohne IDE *Präsenz*

In dieser Aufgabe sollen Sie ausprobieren, wie Java-Programme ohne Unterstützung durch eine IDE (Integrated Development Environment) erstellt, kompiliert und ausgeführt werden. Um diese Aufgabe bearbeiten zu können, müssen Sie Java installiert haben, wie in der Installationsanweisung für Ihre Plattform (Windows oder Mac) besprochen wird.

a) Verändern Sie das Programm `Begruessung` aus der Installationsanleitung Windows/Mac so, dass Sie selbst begrüßt werden. Speichern Sie die Datei ab. Führen Sie das Programm aus, **ohne** es zu kompilieren. Was wird in der Eingabeaufforderung ausgegeben und warum?

Es wird immer noch `Hallo Welt!` ausgegeben. Das Programm wurde nicht nochmal kompiliert, so dass der kompilierte Byte-Code in der Datei `Begruessung.class` immer noch der alte ist. Dieser alte Byte-Code wird beim Ausführen benutzt und gibt daher auch die alte Begrüßung aus.

b) Kompilieren Sie Ihr verändertes Programm und führen Sie es aus. Was wird jetzt in der Eingabeaufforderung ausgegeben und warum?

Nun wird die entsprechende Person begrüßt. Das Programm wurde neu kompiliert, so dass der alte Byte-Code in der Datei `Begruessung.class` mit dem neuen Byte-Code überschrieben wurde. Jetzt kann der neue Byte-Code beim Ausführen benutzt werden und gibt daher auch die neue Begrüßung aus.

c) Stellen Sie wieder den Ursprungszustand Ihres Programms her, d.h. das Programm soll nun wieder `Hallo Welt!` ausgeben. Verändern Sie Ihr Programm, indem Sie die Anführungszeichen weglassen, d.h. Ihr Programm sollte nun folgendermaßen aussehen:

```
1 public class Begruessung {
2    public static void main(String[] args) {
3       System.out.println(Hallo Welt!);
4    }
5 }
```

Warum können Sie Ihr Programm jetzt nicht mehr kompilieren? Wie nennt man diese Art von Fehler?

Das Programm kann nicht mehr kompiliert werden, da es einen **syntaktischen** Fehler enthält. Strings müssen in Anführungszeichen eingeschlossen werden. Der Compiler javac erkennt dies und gibt eine Fehlermeldung aus. Der Byte-Code in der Datei `Begruessung.class` wird nicht neu erzeugt.

d) Fügen Sie die Anführungszeichen wieder in das Programm ein, ersetzen Sie jetzt aber das Wort `Welt` durch `Wlt`, d.h. Ihr Programm sollte folgendermaßen aussehen:

```
1 public class Begruessung {
2    public static void main(String[] args) {
3       System.out.println("Hallo Wlt!");
4    }
5 }
```

Können Sie Ihr Programm jetzt kompilieren? Warum enthält dieses Programm trotzdem einen Fehler und wie nennt man diesen Fehler?

Das Programm kann jetzt ordnungsgemäß kompiliert werden, da es syntaktisch korrekt ist, d.h. dieses Programm enthält nur gültigen Java-Code. Das Programm enthält allerdings trotzdem einen Fehler, da es nicht mehr die Welt begrüßt, sondern hier ein sinnloser Name `Wlt` steht. Dies ist ein semantischer Fehler.

Aufgabe 1-2 **Arbeiten mit Java mit IDE (Eclipse)** *Präsenz*

In dieser Aufgabe sollen Sie die Vorteile einer IDE (Integrated Development Environment) wie Eclipse gegenüber der Benutzung eines einfachen Editors erfahren. Um diese Aufgabe bearbeiten zu können, müssen Sie Java und Eclipse installiert haben, wie in der Installationsanweisung für Ihre Plattform (Windows oder Mac) besprochen wird.

a) Verändern Sie das Programm `Begruessung` aus der Installationsanleitung Windows/Mac in Eclipse so, dass Sie selbst begrüßt werden. Speichern Sie die Datei ab. Führen Sie das Programm erneut in Eclipse aus. Was wird in der Console ausgegeben und warum?

Hinweis: Vergleichen Sie die Ausgabe mit der Ausgabe von Aufgabe 1-1a.

In der Console wird der eigene Name ausgegeben. Eclipse kompiliert das Programm im Hintergrund automatisch selbst, so dass im Gegensatz zu Aufgabe 1-2a die Änderung sofort kompiliert wird und dadurch beim Ausführen auch angezeigt wird.

b) Stellen Sie wieder den Ursprungszustand Ihres Programms her, d.h. das Programm soll nun wieder `Hallo Welt!` ausgeben. Verändern Sie Ihr Programm, indem Sie die Anführungszeichen weglassen, d.h. Ihr Programm sollte nun folgendermaßen aussehen:

```
1 public class Begruessung {
2    public static void main(String[] args) {
3       System.out.println(Hallo Welt!);
4    }
5 }
```

Wie und wo stellt Eclipse überall dar, dass in Ihrem Programm ein Fehler enthalten ist?

Dieser syntaktische Fehler wird in der Standard-Ansicht an vier Stellen angezeigt: im Package Explorer, im Editor von Eclipse, in der Outline und in der Problems-View. Besonders hilfreich sind die Fehlermeldungen im Editor, da dort die entsprechende Stelle „unterringelt“ wird, sowie an der Seite angezeigt wird. Hält man die Maus über diese Stelle wird eine ausführlichere Fehlermeldung angezeigt. In der Problems-View werden diese Fehlermeldungen ebenfalls dargestellt.

c) Fügen Sie die Anführungszeichen wieder in das Programm ein, ersetzen Sie jetzt aber das Wort `Welt` durch `Wlt`, d.h. Ihr Programm sollte folgendermaßen aussehen:

```
1 public class Begruessung {
2    public static void main(String[] args) {
3       System.out.println("Hallo Wlt!");
4    }
5 }
```

Kann Eclipse hier einen Fehler finden? Erklären Sie Ihre Antwort.

Eclipse kann diesen semantischen oder logischen Fehler nicht finden, da Eclipse lediglich syntaktische Fehler erkennen kann.

d) Stellen Sie wieder den Ursprungszustand Ihres Programms her, d.h. das Programm soll nun wieder `Hallo Welt!` ausgeben. Fügen Sie anschließend nach jedem Wort (außer nach `"Hallo`) einen Zeilenumbruch ein und speichern Sie die Datei ab. Ihr Programm sollte dann folgendermaßen aussehen:

```
1 public
2 class
3 Begruessung {
4 public
5 static
6 void
7 main(
8 String[]
9 args) {
10 System.
11 out.
12 println(
13 "Hallo Welt!"
14 );
15 }
16 }
```

Können Sie Ihr Programm jetzt noch kompilieren und ausführen? Erklären Sie Ihre Antwort!

Der Java-Compiler behandelt Zeilenumbrüche ebenso wie Leerzeichen nur als Trennzeichen zwischen Source-Code-Elementen. Für den Java-Compiler ist es also egal, ob Sie ein Leerzeichen oder einen Zeilenumbruch zwischen Source-Code-Elementen verwenden. Der Java-Compiler kann den Code trotzdem übersetzen. Ein Sonderfall ist ein String, der in Anführungszeichen eingeschlossen werden muss. Das beginnende Anführungszeichen und das beendende Anführungszeichen eines String müssen immer auf derselben Zeile stehen. Daher ist folgendes nicht möglich:

```
1 "Hallo
2 Welt!"
```

Falls ein String für eine Zeile zu lang ist, muss er in zwei Strings aufgeteilt werden und diese zwei Strings anschließend konkateniert werden:

```
1 "Hallo " +
2 "Welt!"
```

e) Klicken Sie nun im Package Explorer rechts auf die Datei `Begruessung.java` und wählen Sie im Kontextmenü `Source -> Format`. Was passiert mit Ihrem Source Code und warum ist das sinnvoll?

Der Source-Code wird formatiert, so dass er leichter lesbar ist. Für Java gibt es Coding Conventions, die angeben wie „guter“ Code formatiert sein sollte, damit er schnell und leicht lesbar ist. Diese Coding Conventions sind von Unternehmen zu Unternehmen unterschiedlich, sollten aber prinzipiell einheitlich festgehalten werden. Beispielsweise können diese Coding Conventions sein, dass nach der öffnenden Klammer nach einer Klassendeklaration oder Methodendeklaration stets eine neue Zeile begonnen wird. Ein anderes Beispiel ist, dass

eine Code-Zeile nie länger als 80 Zeichen sein sollte, um zu gewährleisten, dass sie auf eine Bildschirmseite passt.

Hinweis: Schneller kann der Source-Code in Eclipse formatiert werden, indem `Strg+Shift+F` gedrückt wird. Eclipse kann außerdem auch so konfiguriert werden, dass der Source-Code automatisch beim Speichern einer Datei formatiert wird. Wählen Sie dazu im Menü `Window -> Preferences -> Java -> Editor -> Save Actions`. Dort muss die Checkbox für `Perform the selected actions on save` und die Checkbox für `Format source code` angehakt werden. Nach Betätigung des `Ok`-Buttons wird der Source-Code immer automatisch beim Speichern formatiert.