

Übungen Teil 1

1. Aufgabe

Lernziel

Erläutern können, wie Java-Programme im Vergleich zu traditioneller Programmiersprachen übersetzt und ausgeführt werden.

Aufgabenstellung

Eine Firma arbeitet auf zwei verschiedenen Plattformen. Ein Vertreter wird eingeladen, um auf jeder Plattform je ein Java- und ein C++-Programm zu demonstrieren.

- Was muss der Vertreter für seine Vorführung tun und beachten?
- Gehen Sie auf die prinzipiellen Unterschiede der beiden Programmiersprachen ein.

2. Aufgabe

Lernziele

Überprüfung und Korrektur der Syntax. Programme durch Strukturierung/geeignetes Einrücken lesbarer machen.

Aufgabenstellung

Markieren und verbessern Sie die Fehler in folgendem Java-Quellprogramm, so dass dieses Programm übersetzt und in der Java-Laufzeitumgebung (JRE) ausgeführt werden kann. Übernehmen Sie in diesem Zusammenhang das Programm nach Eclipse.

```

/* Unsere erste Java-Anwendung
 * (in diesem Zusammenhang gleich ein mehrzeiliger Kommentar)
 */
//>> damit beginnt ein Zeilenkommentar
Public class Beispiell //Beispiell ist ein Klassenname
//der Dateiname muss Beispiell.java heißen
{
    public static void main(String[] args)//Dies ist eine Operation
    //bzw. Methode; es folgen mehrere Ausgabeanweisungen
    {
        System.out.println("*****");
        system.out.println("*           Java           *");
        System.out.println("* eine Einführung in die Grundlagen *");
        System.out.println("*           FH Ansbach           *");
        System.out.println("*****");
    }
}

```

Üben Sie in diesem Zusammenhang das Arbeiten im Eingabefenster – Verwenden der Befehle **javac** und **java** – und das Arbeiten in der Eclipse-Umgebung. Legen Sie sich für Eclipse eine geeignete Projektumgebung für die weiteren Beispiele und Übungen an.

3. Aufgabe

Lernziele

Java-Programme erfassen, übersetzen und ausführen können. Anhand einer gegebenen Syntax ein richtiges Programm oder Programmstück schreiben können. Programme entsprechend den Richtlinien durch Einrücken geeignet strukturieren können. Eine Programmierungsumgebung für Java bedienen können.

Aufgabenstellung

- Modifizieren Sie den vorhandenen Quell-Code des Beispiels "Beispiel1" so, dass Sie daraus ein eigenes Programm entwickeln.
Ändern Sie den Ausgabertext von "Beispiel1" in einen eigenen Ausgabertext. Fügen Sie ein oder mehrere weitere Ausgabetexte hinzu.
- Testen Sie das Programm.

4. Aufgabe

Lernziele

Mit Eclipse vertraut werden

Aufgabenstellung

- Richten Sie das Package JavaPack ein. Integrieren Sie dazu das Java-Archiv `jpWS1516.jar` in Ihren Workspace
- Lesen Sie eine ganze Zahl ein und geben Sie diese wieder aus.
- Übung zur Online-Dokumentation: Suchen Sie in der Online-Dokumentation (API) nach der Beschreibung der Methode `println` und des Objekts `System.out`. (Falls notwendig, richten Sie die Hilfefunktion unter Eclipse ein.)
- Setzen Sie das Netto-/Bruttopreisprogramm aus den Vorlesungsfolien um (siehe unten).

```
System.out.print("Menge: ");
int menge = Einlesen.liesInt();
System.out.print("Nettopreis: ");
double nettoPreis = Einlesen.liesDouble();
System.out.print("MWST: ");
double mwst = Einlesen.liesDouble();
double wareNetto = nettoPreis*menge;
double wareBrutto = wareNetto*(mwst + 100)/100;
System.out.println(" Warenpreis Brutto: " + wareBrutto);
```

- Verkürzen Sie das Netto-/Bruttopreisprogramm, indem Sie die `Einlesen.Lies*`-Funktionen mit Prompt-Parameter verwenden

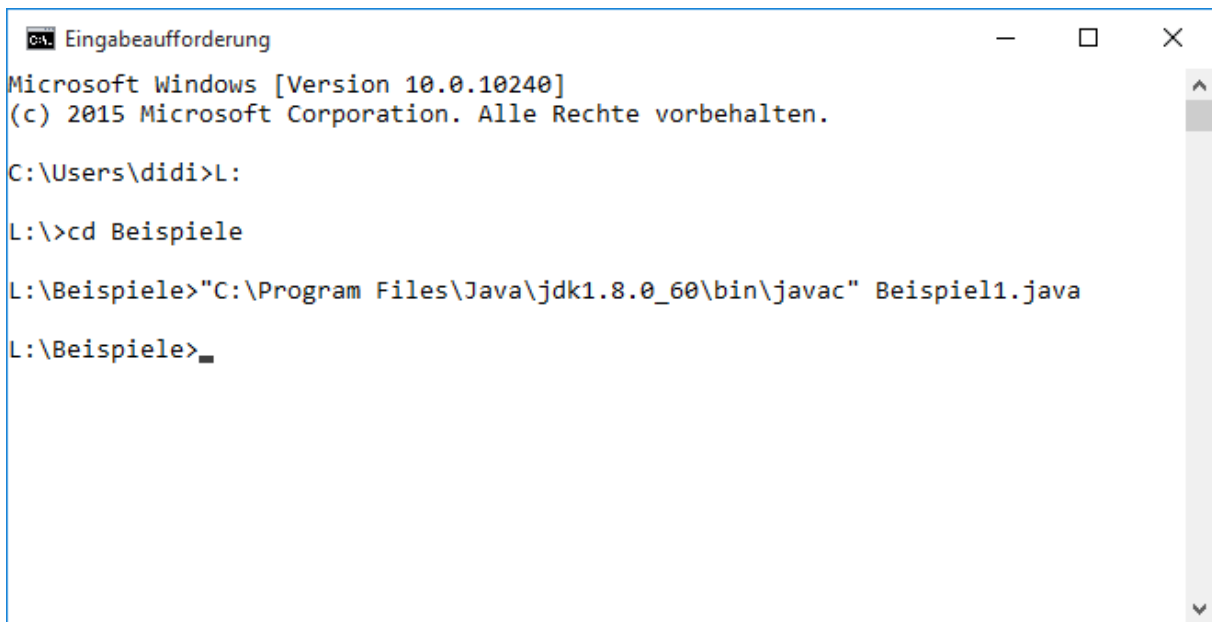
Anhang 1: manuelles kompilieren und starten eines einfachen Java-Programms

Öffnen Sie ein Konsolenfenster (auch *Eingabeaufforderung* genannt), und wechseln Sie in das Verzeichnis mit dem neu erstellten Quellprogramm **Beispiel1.java**.

Lassen Sie das Programm vom JDK-Compiler **javac** übersetzen:

`"C:\Program Files\Java\jdk1.8.0_1441\bin\javac" Beispiel1.java`

Falls beim Übersetzen keine Probleme auftreten, meldet sich der Rechner nach kurzer Bedenkzeit mit einer neuen Kommandoaufforderung zurück, und die Quellcodedatei **Beispiel1.java** erhält Gesellschaft durch die Bytecode-Datei **Beispiel1.class**, z.B.:



```

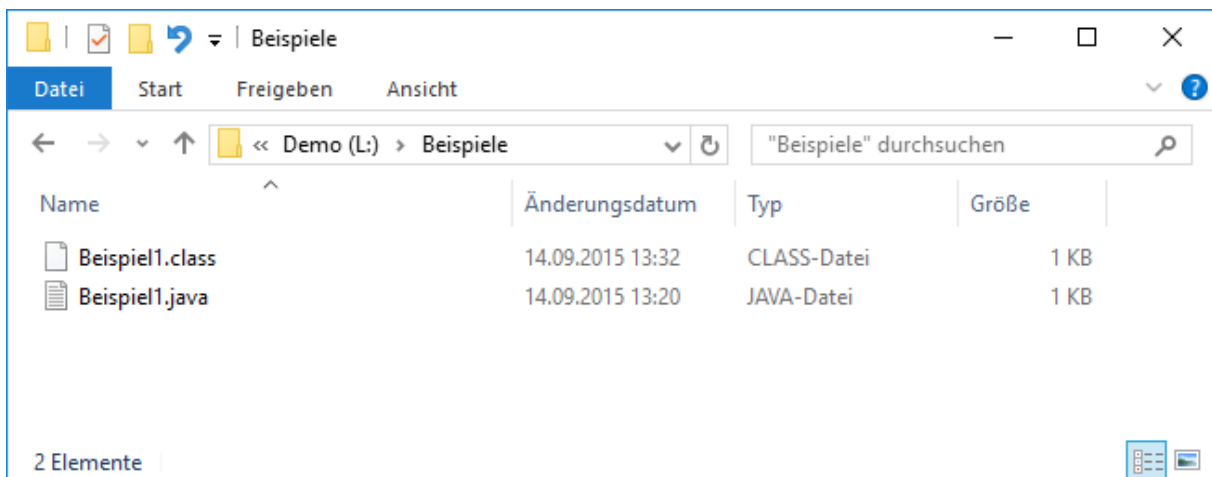
C:\> Eingabeaufforderung
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\didid>L:

L:\>cd Beispiele

L:\Beispiele>"C:\Program Files\Java\jdk1.8.0_60\bin\javac" Beispiel1.java

L:\Beispiele>_
    
```



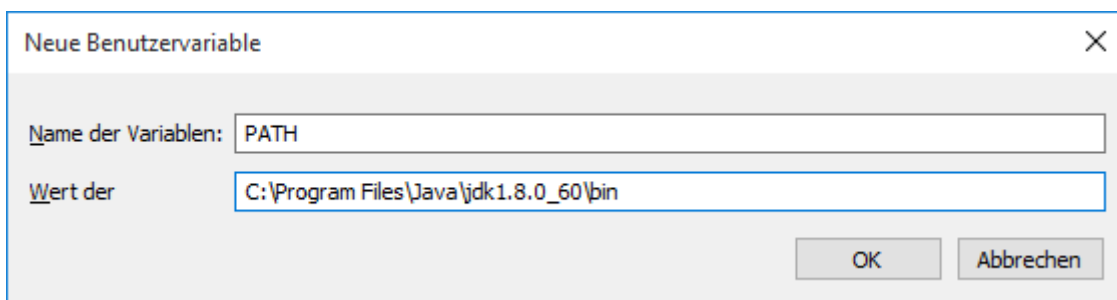
Damit der Compiler ohne Pfadangabe von jedem Verzeichnis aus gestartet werden kann, **javac Beispiel1.java**

muss das **bin**-Unterverzeichnis der JDK-Installation (mit dem Compiler **javac.exe**) in die Definition der Umgebungsvariablen **PATH** aufgenommen werden. Bei der Software-Entwicklung mit Eclipse spielt der Compiler **javac.exe** keine Rolle, und ein Verzicht auf den **PATH**-Eintrag wirkt sich in der Übung kaum aus.

¹ Bitte immer das aktuelle JDK nehmen

Unter Windows 7/8 (im Prinzip auch 10) lässt sich der **PATH**-Eintrag z.B. so realisieren:

- Öffnen Sie über das Startmenü die **Systemsteuerung**.
- Wählen Sie als **Anzeige** über das Bedienelement oben rechts die Option **Kleine Symbole**.
- Wählen Sie im renovierten Fenster per Mausklick die Option **Benutzerkonten**.
- Klicken Sie im Seitenmenü des nächsten Dialogs auf den Link **Eigene Umgebungsvariablen ändern**.
- Nun können Sie in der Dialogbox **Umgebungsvariablen** die **Benutzervariable PATH** anlegen oder erweitern. Mit Administratorrechten lässt sich auch die Definition des regelmäßig vorhandenen **Systemvariablen** gleichen Namens erweitern.
- Im folgenden Dialog wird eine **neue Benutzervariable** angelegt:



Funktioniert erst nach Neuöffnung der Eingabeaufforderung. Beim Erweitern einer **PATH**-Definition trennt man zwei Einträge durch ein Semikolon.

Zum *Ausführen* von Java-Programmen wird *nicht* das JDK (mit Entwicklerwerkzeugen, Dokumentation etc.) benötigt, sondern lediglich die **Java Runtime Environment** (JRE) mit dem Interpreter **java.exe** und der Standardklassenbibliothek.

Lassen Sie das Programm (bzw. die Klasse) **Beispiel1.class** von der JVM ausführen. Der Aufruf

java Beispiel1

sollte also funktionieren.

```

C:\> Eingabeaufforderung
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\didi>L:

L:\>cd Beispiele

L:\Beispiele>"C:\Program Files\Java\jdk1.8.0_60\bin\javac" Beispiel1.java

L:\Beispiele>java Beispiel1
*****
*                Java                *
* eine Einführung in die Grundlagen *
*                FH Ansbach          *
*****

L:\Beispiele>_
    
```

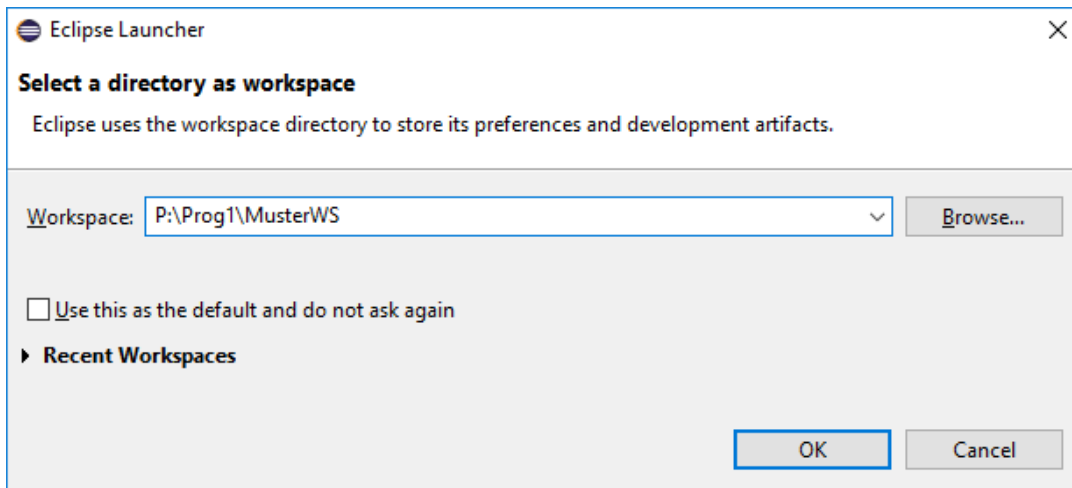
In der Regel müssen Sie keinen **PATH**-Eintrag vornehmen, um **java.exe** bequem starten zu können, weil sich das JRE-Installationsprogramm, darum kümmert.

Beim Programmstart ist zu beachten:

- Die Namenserverweiterung **.class** wird **nicht** angegeben.
Wer es doch tut, erhält eine Fehlermeldung.
- Beim Aufruf des Interpreters wird der Name der auszuführenden Klasse als Argument angegeben. Weil es sich dabei um einen Java-Bezeichner handelt, muss die Groß-/Kleinschreibung mit der Klassendeklaration (in der Datei **Beispiel1.java**) übereinstimmen (auch unter Windows!). Java-Klassennamen beginnen meist mit großem Anfangsbuchstaben, und genau so müssen die Namen auch beim Programmstart geschrieben werden.

Anhang 2: Organisation des Workspace, der Übungen, der Aufgaben

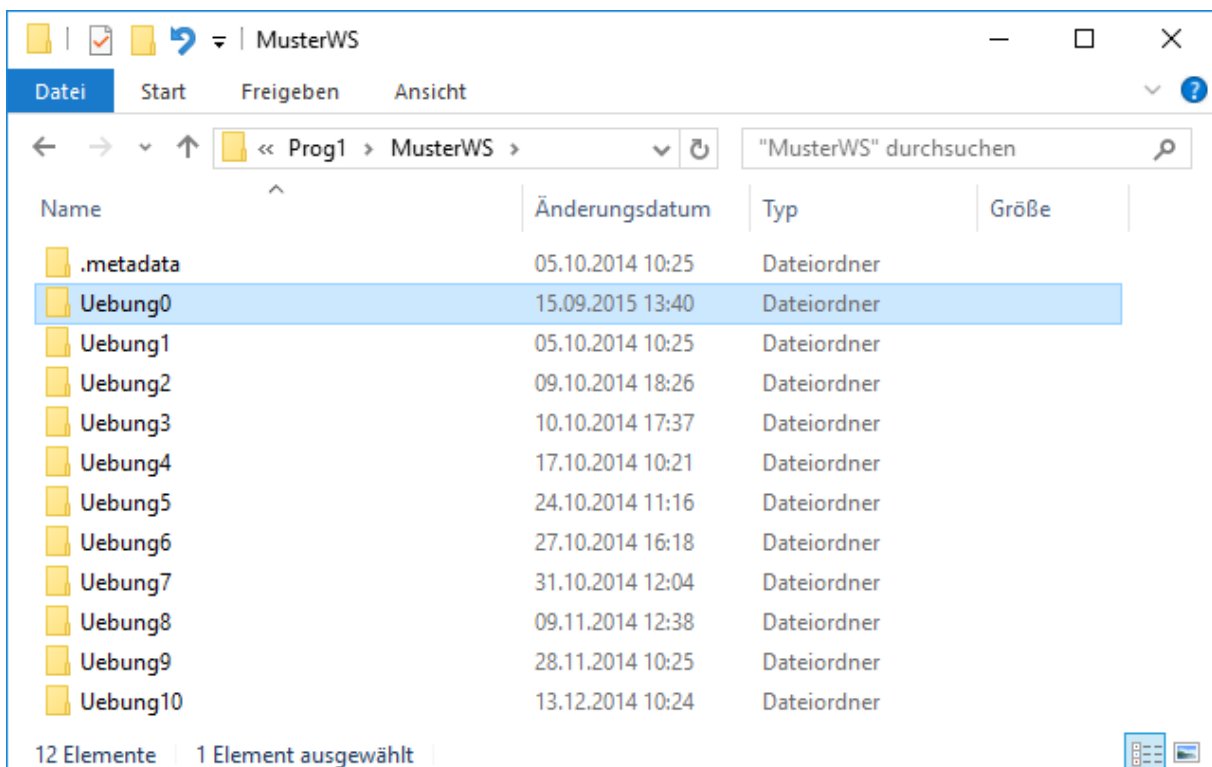
Eclipse legt für jedes Projekt einen Ordner an, der Quellcode-, Bytecode-, Konfigurations- und Hilfsdateien aufnimmt. Zusammengehörige Projekte bilden einen Arbeitsbereich (Workspace), und weil Eclipse zur Bearbeitung eines solchen Projekt-Ensembles konzipiert ist, verlangt es bei jedem Start den Arbeitsbereichsordner zur Sitzung. Aufruf des Workspace:



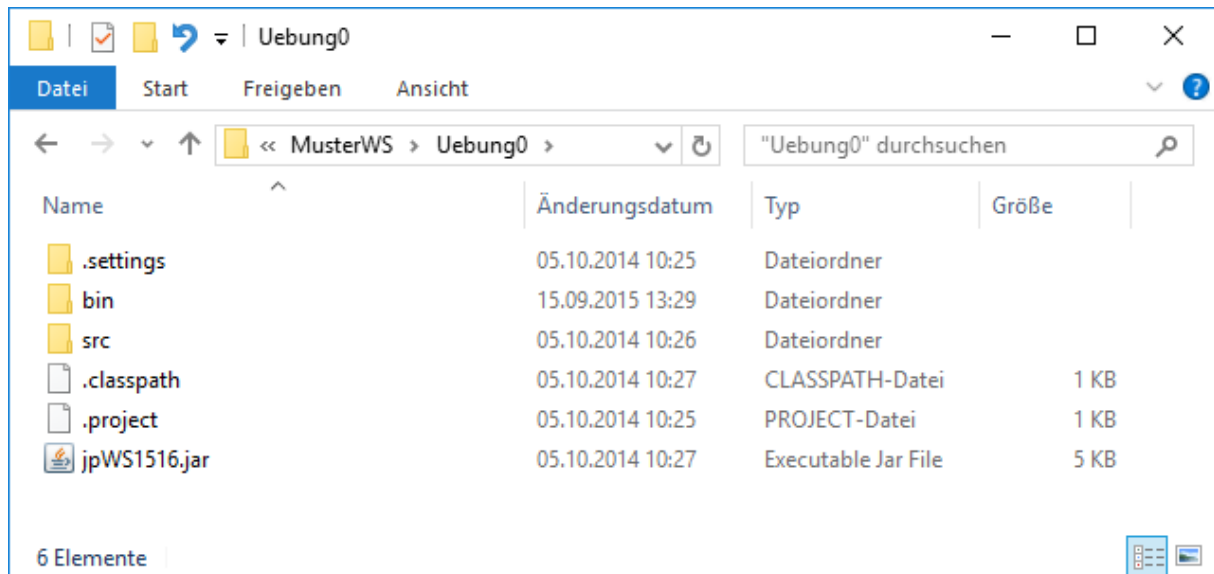
Ein Arbeitsbereichsordner enthält:

- Konfigurationsunterordner (z.B. .metadata)
- und die Ordner der „internen“ Projekte des Arbeitsbereichs.

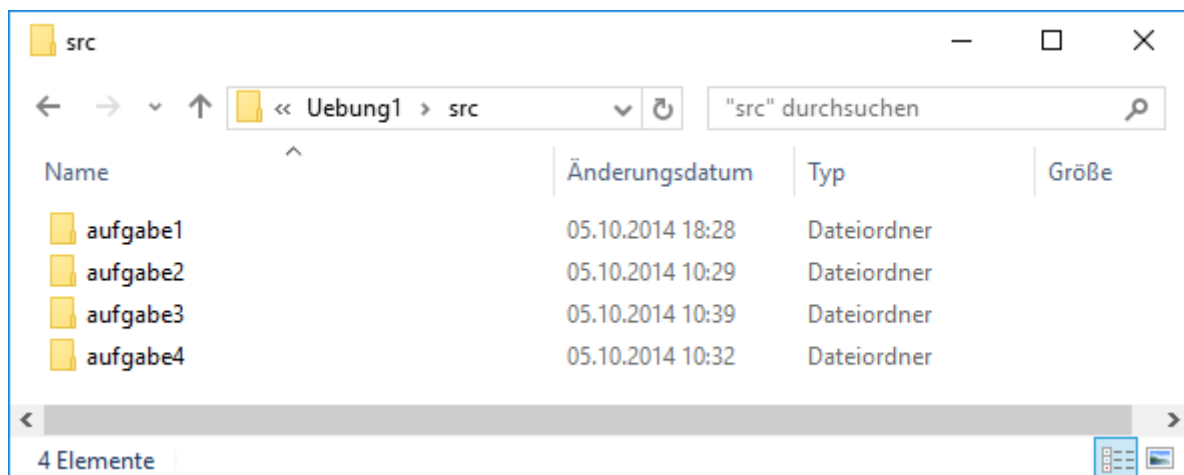
Hier ist ein Arbeitsbereichsordner mit elf internen Projekten (*Uebung0* bis *Uebung10*) zu sehen:



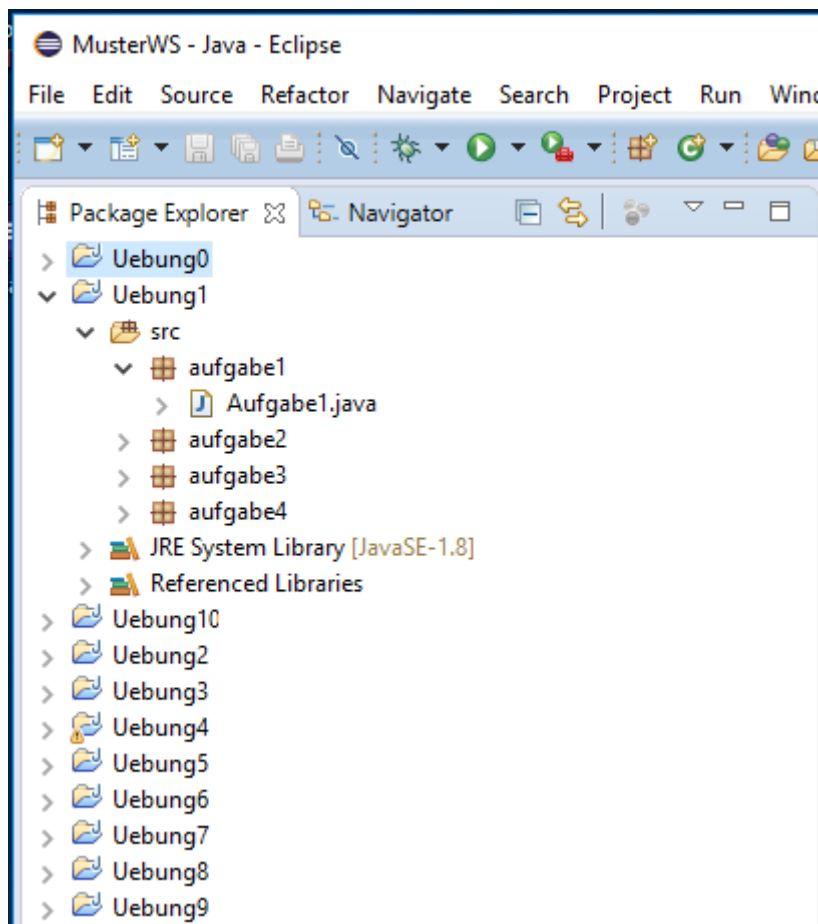
Dazu sehen Sie beispielhaft den Projektordner *Uebung0*:



Die Quellfiles (.java) werden unter „src“, die kompilierten Files (.class) unter „bin“ abgelegt. An den beiden Abbildungen erkennt man die gewünschte Systematik unserer Übungen. Es soll nur einen Workspace geben. Innerhalb dieses **Workspace** sind die einzelnen Übungen jeweils als Projekt angelegt. Eine Übung besteht aus einer Sammlung von Aufgaben (wie diese Übung 1). Für jede Aufgabe wird innerhalb eines **Projekts** wieder ein eigener Ordner (Paket) angelegt – parallel unter „src“ und „bin“. Dort sind dann die .java und .class Files abgelegt. Siehe dazu auch die folgende **Abbildung**:



Darstellung unter Eclipse (Sicht Package Explorer): eine reine Entwicklersicht



Darstellung unter Eclipse (Sicht „Navigator“): zeigt alle vorhandenen Files innerhalb eines Workspaces

