

Übungsblatt 2: Eclipse

Aufgabe 1 (Eclipse)

Bevor Sie Eclipse starten, sollten Sie auf Ihrem Homeverzeichnis innerhalb von `.nt` ein Verzeichnis, zum Beispiel `java` einrichten, in dem alle Programme gespeichert werden. Speichern Sie die Programme *nie* lokal auf dem Windowsrechner! Alternativ können Sie Ihre Programme auch auf einem USB-Stick dauerhaft speichern.

- Starten Sie Eclipse
- Überprüfen Sie, ob Eclipse standardmäßig ein JDK 1.5 (oder 1.6) verwendet und nicht lediglich ein JRE. Öffnen Sie dazu das Menü `Windows -> Preferences`. Suchen Sie den Eintrag `Java -> Installed JREs`. Wenn dort nur JRE 1.4 vorhanden ist, müssen Sie ein aktuelles JDK hinzufügen. Sie finden diese unter `\\ads\dfs\rz\Apps\Sun`. Stellen Sie eines der neueren JDK/SDKs als Standard ein.¹
- Stellen Sie den Editor so ein, dass er Tabulatoren automatisch in Leerzeichen wandelt und eine vertikale Linie bei 80-100 Zeichen in einer Zeile anzeigt. Mehr Zeichen pro Zeile sollten Sie nicht eingeben. Verwenden Sie die Hilfe von Eclipse, um herauszufinden, wie das geht.

Aufgabe 2 (1. Java-Programm, Grundfunktionen von Eclipse)

- Erstellen Sie in Eclipse ein **Java**-Projekt mit dem Namen `info1`
- Erstellen Sie folgende Klasse `Konto` und führen Sie es aus.

```
public class Konto {
    public static void main(String[] args) {
        double guthaben = 0.0;
        System.out.println("Guthaben = " + guthaben);
        guthaben = guthaben + 100.0;
        System.out.println("Guthaben = " + guthaben);
        guthaben = guthaben - 50.0;
        System.out.println("Guthaben = " + guthaben);
    }
}
```

- Erweitern Sie die Klasse `Konto` um eine lokale Variable `zinssatz` (Typ `double`, in Prozent) sowie der Berechnung und Ausgabe für den Zinsbetrag, der für das Guthaben am Ende des Jahres dem Konto gutgeschrieben wird.

¹Im Poolraum E203 existieren auch lokale Versionen von Java.

- Setzen Sie einen Haltepunkt auf die erste Zeile in der Methode und führen Sie das Programm mit dem Debugger Zeile für Zeile aus. Betrachten Sie dabei die Änderungen der Werte im **Variables** Fenster. Achten Sie darauf, dass nach Ende der Fehlersuche, der Debugger beendet ist (rotes Rechteck anwählen).
- Kommentieren Sie die Klasse und die Methode mit einem Javadoc Kommentar `/** ... */` und sinnvollem Text. Erzeugen Sie eine Java-Dokumentation der Klasse. (In Eclipse unter **Project->Generate Javadoc**). Wenn dort das Kommando `javadoc.exe` fehlt, dann haben Sie kein JDK, sondern nur ein JRE als Standard eingestellt.
- Wenn Sie Variablen, Klassen oder ähnliches umbenennen, dann machen Sie dies immer mit der Refactor-Funktion. Wählen Sie die Klasse mit Rechtsklick an und benennen Sie sie mit **Refactor->Rename** in **Bankkonto** um. Auch der Dateiname wird korrekt geändert. Wählen Sie **guthaben** und ändern Sie den Namen analog in **guthabenInEuro** um.
- Markieren Sie Codeteile oder den den Text der ganzen Klasse und formatieren Sie ihn mit **Source->Format**.

Erläuterungen zum Programm (Details in der Vorlesung)

Das Schlüsselwort `public` definiert, dass die Klasse `Konto` von außen zugänglich ist. Ansonsten wäre die Klasse nur innerhalb der in der Datei definierten Programmteile nutzbar. Eine Java Programmdatei muss also immer eine Klasse enthalten, die als `public` deklariert ist.

Innerhalb der geschweiften Klammern hinter dem Klassennamen `Konto` ist eine ganz spezielle Methode aufgeführt: die `main`-Methode. Sie ist auch `public`, damit sie von außen, etwa vom Java Interpreter, aufgerufen werden kann. Der "Modifier" `static` zeigt an, dass diese Methode direkt auf der Klasse aufgerufen werden kann (`Konto.main()`). Das Schlüsselwort `void` definiert, dass die `main`-Methode keinen Wert an das aufrufende Programm zurückgibt. Hinter dem Methodennamen `main` ist ein Parameter definiert. Die `main`-Methode muss immer `main` heißen und mit `public static void` sowie genau einen Parameter von Typ `String[]` deklariert werden, ansonsten kann Eclipse die Methode nicht ausführen.

Innerhalb der geschweiften Klammer der `main`-Methode wird zuerst eine lokale Variable namens `guthaben` deklariert, die Gleitkommawerte mit 64bit Genauigkeit annehmen kann (`double`). Mit `= 0.0` werden die Variablen bei der Deklaration mit dem Wert 0 initialisiert.

Die Anweisung `System.out.println("Guthaben =" + guthaben)` gibt den in Hochkommas eingeschlossenen Text und `(+)` den aktuellen Inhalt der Variablen `guthaben` aus. Die Anweisung `guthaben = guthaben + 100.0` weist der lokalen Variablen `guthaben` den Wert der rechten Seite vom Zuweisungsoperator `=` zu. Der Wert wird bei Ausführung der Anweisung berechnet: Wert zum Ausführungszeitpunkt von `guthaben` (Wert 0) addiert zum Wert 100, ergibt 100.

Bevor eine lokale Variable verwendet werden kann, muss sie wie oben deklariert werden. Es können beliebig viele lokale Variablen mit unterschiedlichen Namen deklariert werden. Anweisungen, Zuweisungen und Deklarationen können in beliebiger Reihenfolge innerhalb einer Methode auftreten. Zur besseren Übersicht sollten aber alle lokalen Variablen wie oben am Anfang der Methode deklariert werden.