TECHNISCHE HOCHSCHULE AUGSBURG – FAKULTÄT FÜR INFORMATIK

PRAKTIKUM EINFÜHRUNG IN DIE PROGRAMMIERUNG

BLATT 0

Wintersemester 2025/2026

AUFGABE 1 - STARTEN VON ECLIPSE

Sie können diese Aufgabe am eigenen Rechner oder an den Pool-Rechnern im Labor bearbeiten.

Eigener Rechner:

Installieren sie *Eclipse for Java Developers*[1] und *git*[2] auf ihrem Rechner.

Pool-Rechner der Informatik:

- Booten Sie in die Umgebung "Debian/Linux"
- Loggen Sie sich mit Ihrem RZ-Account ein
- Starten Sie ein Terminal
- Starten Sie Eclipse mit dem Befehl */opt/eclipse-java/eclipse*

AUFGABE 2 - ERSTE SCHRITTE IN ECLIPSE

a. Legen sie ein neues ”Java Project” in Eclipse an und erstellen sie darin ein *Hello World*-Programm. Führen sie das Programm aus, um ihre Installation zu testen.

b. Laden sie sich im Anschluss den *Source Code* zum Kapitel *Einführung* aus dem Moodle-Kurs herunter und importieren sie Code in Eclipse.

   Sie können dies über das Menu *Files -> Import -> Existing Projects into Workspace -> Select Archive File* verwenden.

   Testen sie mit Hilfe des Codes aus der Vorlesung, dass ihre Installation funktioniert

c. *Eclipse* ist eine sehr umfangreiche IDE, mit vielen Funktionen. Eclipse kann durch Plugins um viele weitere Funktionen erweitert werden. Auch die Darstellung einzelner Fenster und Funktionen kann an die eigenen Bedürfnisse angepasst werden.

   Eclipse hat verschiedene *Perspectives*, die in unterschiedlichen Betriebsmodi unterschiedliche Informationen anzeigen. Die standardmäßige *Perspective* ist die *Java Perspective*. Sollten sie in Eclipse einmal in der Situation sein, dass alles anders aussieht, dann sind sie vermutlich in einer anderen *Perspective*. Um wieder zurück zu kommen, können sie über das Menu *Window -> Perspective -> Open Perspective* wieder die Java-View öffnen. In der rechten oberen Ecke befindet sich dazu auch ein Shortcut.

[1] `https://www.eclipse.org/downloads/`
[2] `https://git-scm.com/downloads`

Sie können die Java-Perspective auch umkonfigurieren, so dass andere Informationen angezeigt werden. Sollten sie ausversehen die Java-Perspective so umkonfiguriert haben, dass sie nicht mehr zurecht kommen, können sie diese unter dem Menu *Window -> Perspective -> Reset Perspective* wieder herstellen.

Sie können ebenfalls einzelne *Views*, wie z.B. die Console oder den Package Explorer an anderer Stelle platzieren oder schließen. Um geschlossene *Views* wieder herzustellen, können sie diese über das Menu *Window -> Show View* wieder herstellen.

Suchen sie den View *Console*, also den View in dem z.B. der Text *Hello World* angezeigt wurde. Schließen sie den View und bringen sie ihn über das Menu wieder zurück.

d. Öffnen sie die Klasse *TestDebugger.java*. Setzen sie einen Break-Point in die Zeile 6 durch Doppelklick links neben der Zeilennummer. Wenn der Breakpoint erfolgreich gesetzt wurde, dann erscheint ein Punkt neben der Zeilennummer.

   Führen sie nun das Programm im Debug-Modus aus. Die Ausführung sollte nun an dem Break-Point halten. Experimentieren sie mit den Funktionen *Step Over*, *Step Into*, *Step Out* und beobachten sie was genau bei den einzelnen Schritten passiert.

   Erschließen sie sich die Bedeutung des speziellen Zeichens '\' anhand ihrer Beobachtungen.

AUFGABE 3 - VORBEREITUNG - ABGABE VON AUFGABEN

a. Generieren Sie wie im Praktikum beschrieben ein SSH-Schlüsselpaar mit ssh-keygen. Stellen sie sicher, dass ihr Schlüsselpaar in dem Verzeichnis *.ssh* unterhalb ihres Home-Verzeichnisses liegt.

b. Öffnen sie einen Browser und besuchen sie die Seite `https://prog.informatik.tha.de`. Loggen sie sich mit ihrem RZ-Account ein und laden sie ihren Public Key, also den Inhalt der Datei mit der Dateiendung .pub in das System.

   Hinweis: Wenn sie mit mehreren Rechnern, z.B. einem der Laborrechner und ihrem Desktop zu Hause, auf den Code haben möchten, dann generieren sie sich auf dem zweiten System einen neuen SSH-Key und laden sie ihn ebenfalls als zweiten Key in das System.

c. Verifizieren sie dass der Schlüssel korrekt konfiguriert wurde. Öffnen sie dazu eine Shell und tippen sie den folgenden Befehl ein:

   ```
   ssh -p 3333 git@prog.informatik.tha.de
   ```

   Wenn die Konfiguration erfolgreich war, dann sollten sie einen Output haben, der in etwa wie folgt aussieht:

   ```
   PTY allocation request failed on channel 0
   hello <username>, this is git@74eb43179f0e running gitolite3 3.6.12-4 (Debian) on git 2.47.3

   ...
   ```

   Wenn sie einen ähnlichen Output sehen, dann haben sie alles korrekt konfiguriert.

   Diese Fehlermeldung zeigt an, dass noch kein Source Code vorhanden ist und daher alle Tests fehlgeschlagen sind. Sobald sie

d. Im Anschluss können sie sich das Eclipse-Projekt zu Blatt0 vom Server ziehen. In dieses Projekt können sie ihre Lösung einprogrammieren. Nutzen sie dazu den Import-Dialog von Eclipse

   File -> Projects from Git (with smart import) -> Clone URI .

Im erscheinenden Dialog im Feld *URI* geben sie folgende URI ein (Achtung: Diese müssen sie personalisieren):

`ssh://git@prog.informatik.tha.de:3333/prog1/<username>/blatt0`

Ersetzen sie dabei <username> durch den Nutzernamen ihres RZ-Accounts (lassen sie dabei die eckigen Klammern weg).

Folgen Sie den darauf folgenden Dialogen. In ihrem Eclipse sollte dann ein Projekt mit dem Namen *blatt0* erscheinen.

Dieses Projekt ist noch leer. Im folgenden sollen sie ihren Code in dieses Projekt einbauen.

e. Testen sie den Upload auf den Server und die Durchführung von Tests auf dem Server. Öffnen sie dazu ein Terminal mit Rechtsklick auf das Projekt. Wählen sie dann *Show in -> Terminal* aus.

In dem Terminal führen sie den Befehl *git push* aus. Beobachten sie die Ausgabe. Wenn sie eine Meldung darüber bekommen, dass die Tests fehlgeschlagen sind, dann sind sie startklar.

AUFGABE 4 - ERSTE AUFGABEN UND VALIDIERUNG DER LÖSUNG

a. Legen sie im Default-Paket eine Klasse mit dem Namen *HelloWorld* an. Schreiben sie eine Main-Methode, die den String *Hello World!* ausgibt.

Testen sie ihre Main-Methode, bis sie zufrieden mit der Ausgabe sind.

b. Nutzen sie nun das Versionskontrollsystem *git*, um den aktuellen Stand ihres Codes zu festzuschreiben. Führen sie wie zuvor gezeigt einen *Commit* durch:

Rechtsklick auf das Projekt -> Team -> Commit

Überführen sie alle Dateien in die *Staged Changes* und schreiben sie eine *Commit-Message*.

Übertragen sie mit *Commit and Push* die Daten an den Server.

c. Führen sie einen Rechtsklick auf das Projekt durch und wählen Sie *Show in -> Terminal* aus. Dadurch öffnet sich ein Terminal-Fenster, in dass sie den Befehl *git push* eintippen.

Dadurch werden die Tests auf dem Server ausgeführt. Die Ausgabe des Befehls zeigt das Ergebnis des Tests der Server an. Stellen sie sicher, dass die Tests für diese Aufgabe funktionieren.

AUFGABE 5 - ASCII-ARTS

a) Schreiben Sie eine Klasse *Cow* im *default package*, die folgendes ASCII-Art auf der Konsole ausgibt:

```
        (__)
        (oo)
  /------\/
 / |    ||
*  /\---/\
   ~~   ~~
...\"Have you mooed today?\"...
```

Bei zwei Zeichen in diesem Bild kann es Kompiler-Fehler geben. Um welche Zeichen handelt es sich?

Versuchen Sie anhand des Kompiler-Fehlers herauszufinden, wie sie das Zeichen schreiben müssen, damit der Fehler verschwindet und die Kuh korrekt gezeichnet wird.

b) Erzeugen Sie eine Kopie der Klasse *Cow* unter dem Namen *Cow2*. Diese soll die gleiche Kuh ausgeben, aber diesmal nur eine einzige System.out.println()-Anweisung verwenden. Nutzen Sie dazu dass aus der Vorlesung bekannte Zeichen für einen Zeilenumbruch.

c) Führen sie wieder einen Commit ihres Codes durch und laden sie den Code, wie in der vorherigen Aufgabe, auf den Server. Testen sie ob ihr Code ebenfalls auf dem Server korrekt getestet wurde. Das Vorgehen ist wiederum gleich wie in der vorherigen Aufgabe.

AUFGABE 6 - VORBEREITUNG NÄCHSTE VORLESUNG

In der Vorlesung hatten wir Literale und Operatoren kennen gelernt. Schreiben Sie je Teilaufgabe eine Klasse, die folgende Ausgaben produziert:

a) Erinnern Sie sich aus der Vorlesung daran, wie ein Literal für eine Zeichenkette und eine Zahl aussieht. Schreiben Sie ein Programm, welches folgende Dinge für eine beliebige Zeichenkette und Zahl ausgibt:

- Zeichenkette + Zahl + Zahl
- Zahl + Zahl + Zeichenkette

Fällt ihnen ein Unterschied in der Ausführung der Operatoren auf? In welcher Reihenfolge werden die Operationen ausgeführt?

Probieren Sie aus, ob sich die Ausgabe verändert wenn Sie Klammern setzen (wie in mathematischen Berechnungen). Können Sie die ausgegebenen Texte durch geeignete Klammern verändern?

b) In der Vorlesung wurden Literale für Ganzzahlen und Kommazahlen behandelt. Schreiben Sie ein Programm, dass folgende Ausgaben produziert (jeweils mit Ganzzahlen und mit Kommazahlen):

- Teilen einer ungeraden Zahl durch 2
- Berechnung von $\frac{1}{3}$
- Eine sehr lange Zahl (probieren Sie wie lange die Zahl sein darf bevor ein Fehler auftritt)
- Multiplizieren Sie die längste Zahl, die sie zuvor finden konnten mit sich selbst
- Teilen sie eine positive/negative Zahl durch Null
- Berechnen Sie Null geteilt durch Null

Notieren Sie ihre Beobachtungen. Welche Effekte fallen Ihnen bei Kommazahlen und bei Ganzzahlen auf?

Nutzen Sie ihr Wissen aus der Vorlesung, um die Effekte zu erklären (Hinweis: es kann sein, dass wir noch nicht so weit in der Vorlesung gekommen sind. In diesem Fall: Schreiben Sie auf was sie nicht verstanden haben und bringen Sie es in die nächste Vorlesung mit).