Einführung in die Nutzung von *eclipse*

1 Einleitung

eclipse ist ein mächtiges und weit verbreitetes Entwicklungswerkzeug zur Java-Programmentwicklung. *eclipse* kann aber auch für andere Entwicklungsaufgaben genutzt werden kann, wie z.B. für die C++-Programmentwicklung. Die nachfolgende Anleitung bezieht sich primär auf die Nutzung von *eclipse* unter Linux. Die Nutzungsanleitung ist aber im Wesentlichen auch auf Microsoft Windows, Apple macOS etc. übertragbar.
Wichtig: Solange Sie nicht exakt wissen was Sie tun, welche Konsequenzen dies hat und wie Sie etwas vollständig rückgängig machen können, folgen Sie genau dieser Anleitung und machen bei der Arbeit mit *eclipse* nicht mehr und nicht weniger, als hier beschrieben ist. Verschieben Sie also **alle** Konfigurations-, Layout-, Farbänderungen etc. in *eclipse* um mindesten noch ein Jahr. Schliessen Sie nicht irgendwelche Fenster, von denen Sie nicht wissen, was sie bedeuten. All dies erspart Ihnen und uns erfahrungsgemäß immens viel Arbeit, Zeit und Ärger!

2 Installieren von *eclipse*

Falls Sie mit einem Pool-Rechner arbeiten, brauchen Sie an dieser Stelle nichts weiter zu lesen. Auf allen Pool-Rechnern des Fachbereichs ist *eclipse* bereits unter Linux und Windows installiert.
Links zum Download der nachfolgend angegebenen Software-Pakete finden Sie auf unserer Java-Seite

https://berrendorf.inf.h-brs.de/Java/Java.html

Laden Sie eclipse über den entsprechenden Link auf unserer Java-Seite. Achten Sie darauf, dass Sie die richtige Version für ihr Betriebssystem auswählen (Windows 32/64 Bit, Linux 32/64 Bit, macOS)!
Wenn Ihnen im Installationsprozess mehrere eclipse-Versionen angeboten werden, so wählen Sie die Version **`Eclipse IDE for Java Developers`**.
Für Windows und macOS gibt es fertige Installationspakete, die viele der nachfolgend beschrieben Schritte automatisch machen. Lassen Sie alle Voreinstellungen (z.B. Pfade) bei der Installation so, wies es Ihnen vorgeschlagen wird.
Hier die Schritte, die für Linux getan werden müssen:
Laden Sie die geeignete Version für ihr System (im Folgenden **`eclipse.tar.gz`** genannt) auf ihren Linux-Rechner in ein Verzeichnis ihrer Wahl. Geeignete Verzeich-

nisse sind ihr Home-Verzeichnis oder als root das Verzeichnis **`/usr/local`**. Im Folgenden nehmen wir als Beispiel an, dass ihr Home-Verzeichnis den Namen hat **`/home/meinname`**
Öffnen Sie ein Shell-Fenster und wechseln in das Verzeichnis, in dem die eclipse-Datei liegt. Beispiel:

```
cd /home/meinname
```

Entpacken Sie dort die eclipse-Datei mit:

```
tar xzf eclipse.tar.gz
```

Es wird dadurch ein Unterverzeichnis **`eclipse`** angelegt.
Wenn Sie in Zukunft einfach durch die Eingabe von **`eclipse`** das Entwicklungswerkzeug starten wollen, sollten Sie den Suchpfad für Kommandos entsprechend erweitern. Ändern Sie bzw. fügen Sie in der Datei **`.profile`** (das ist ein . gefolgt von profile) in ihrem Home-Verzeichnis die Zeile ein:

```
export PATH=/home/meinname/eclipse:$PATH
```

3 *eclipse* starten

Wenn Sie *eclipse* auf einem **eigenen Rechner** installiert haben, so gibt es mehrere Möglichkeiten zum Start (hier für Linux angegeben).

- Wechseln Sie in einem Shell-Fenster in das eclipse-Verzeichnis und geben ein:

  ```
  ./eclipse
  ```

- Geben Sie in einem Shell-Fenster den absoluten Pfad an zum eclipse-Programm:

  ```
  /home/meinname/eclipse/eclipse
  ```

- Wenn Sie die PATH-Variable wie oben beschrieben erweitert haben, können Sie in einem beliebigen Shell-Fenster einfach eingeben:

  ```
  eclipse
  ```

Auf Windows und macOS klicken Sie auf das Desktop eclipse-Symbol. Sollte keins angelegt worden sein, so richten Sie zweckmäßigerweise ein solches Desktopsymbol ein.
Auf den **Pool-Rechnern des Fachbereichs** starten Sie eclipse, indem Sie unter Linux in einem Shell-Fenster eingeben

```
eclipse
```

oder im Menu unter Entwicklungswerkzeugen eclipse auswählen.
Nach dem Start erscheint ein Fenster, in dem Sie nach dem Namen eines Workspaces (=Verzeichnis) gefragt werden. Ein Workspace dient *eclipse* zur Verwaltung von Entwicklungsprojekten, z.B. werden dort alle Dateien zu einem Projekt abgelegt. Zu jedem Projekt wird ein Unterordner in dem Workspace angelegt und darin wiederum evtl. weitere Unterordner. Es ist (vorerst) sinnvoll, für alle ihre Projekte den gleichen Workspace anzugeben, im Normalfall der vorgeschlagene Pfad. Beispielsweise

```
/home/meinname/workspace
```

Nachdem Sie auf

Launch

geklickt haben, sollte ein eclipse-Fenster erscheinen, das wiederum im Innern ein Begrüssungsfenster enthält und am oberen Rand eine Menüleiste. Sollten Sie stattdessen eine Fehlermeldung bekommen, dass das Kommando **eclipse** nicht gefunden wurde, überprüfen Sie auf korrekte Schreibweise und ob ihr Kommandosuchpfad das eclipse-Verzeichnis enthält (**echo** $**PATH**; siehe Installation).
Das Begrüßungsfenster können Sie löschen (Den Pfeil oben rechts mit **Hide** anklicken). Ebenso das danach erscheinende große Fenster mit **Donate** durch anklicken des Kreuzes oberhalb dieses Fensters. Sie sehen danach den sogenannten Workbench (Arbeitsblatt).

4 Wichtige Hinweise vorab

Es erspart Ihnen und uns viel Ärger, wenn Sie **nirgendwo** Umlaute oder Leerzeichen in Namen verwenden. Nicht in Dateinamen, nicht in Projektnamen, nicht in Java-Programmen, **wirklich nirgendwo**. Achten Sie weiterhin darauf, dass überall (Dateinamen, Projektnamen, in Java-Programmen, überall) zwischen Groß- und Kleinschreibung unterschieden wird.
Weiterer Hinweis: Der Praktonat / der Java-Compiler erwartet Dateien im sogenannten UTF8-Format für die Zeichenkodierung (dies wird später in der Vorlesung auch behandelt) **ohne Byte-Order-Mark**. Sie können den eclipse-Editor so einstellen, dass er grundsätzlich immer mit dieser Zeichenkodierung ihr Programm speichert:

Window -- Preferences

dann geben Sie oben in der Suchmaske **encoding** ein. Ihnen wird dann unter der Suchergebnissen u.a.

General - Workspace

angeboten. Wenn Sie dort auf

Workspace

klicken, finden Sie unten auf der Einstellungsseite zu Workspace unter

Text file encoding

die Auswahl

other

die Sie wählen, und dort

UTF-8

auswählen.
Sollten Sie aus welchen Gründen auch immer mit einem anderen Editor Änderungen am Programm durchführen, so achten Sie darauf, dass dieser Editor ebenfalls in der geforderten Zeichenkodierung speichert.

5 Klasse anlegen

eclipse arbeitet mit Projekten. Wenn wir ein Programm schreiben wollen, so werden wir dies jedesmal als ein eigenes Projekt anlegen. Innerhalb eines Projekts kann es ein oder auch mehrere Klassen geben. Legen Sie für jede Übungsaufgabe (oder wenn Sie

etwas testen wollen) ein eigenes Projekt an. Vorgehensweise:

File -- New -- Java Project

Es erscheint ein Dialogfenster, in dem man verschiedene Angaben machen kann. Ändern Sie bzw. tragen ein **nur** den Projektnamen. Tragen Sie dort einen sinnvollen Namen ein (ohne Umlaute / Leerzeichen!). Beispielsweise für die 1. Aufgabe in der 8. Übung geben Sie als Projektnamen

U8A1

an. Ändern Sie ansonsten nichts. Klicken Sie auf

Finish

Je nach von *eclipse* verwendeter Java-Version werden Sie in einem Popup-Fenster ggfs. gefragt, ob Sie ein **module-info.java** erzeugen wollen. Wählen Sie hier

Dont Create

Nach dem Click auf **Finish** erscheint dann im sogenannten Package Explorer (links im eclipse-Gesamtfenster) ein neuer Eintrag mit dem Namen des Projekts.
Wenn Sie das Projekt angelegt haben, so legen Sie anschließend in diesem Projekt eine neue Java-Klasse an. Vorgehensweise:

File -- New -- Class

Es erscheint wieder ein Dialogfenster. Ändern Sie nichts an den vielen Default-Einstellungen und tragen lediglich in

Name:

den Namen der neuen Klasse ein. *eclipse* legt automatisch eine Datei mit dem entsprechenden Namen an. Beispiel für einen Klassennamen: **Programmgeruest** Klicken Sie dann auf

Finish

Hinweis: Achten Sie darauf, dass das Feld **Package** beim Anlegen einer Klasse frei bleibt (dies ist die Voreinstellung).
Im mittleren Editor-Fenster erscheint eine minimal vorausgefüllte Java-Klasse mit dem gewählten Namen:

```
public class Programmgeruest {

}
```

Sie können jetzt nach Wünschen diese Klasse ergänzen. Sie werden sehen, dass *eclipse* Ihnen an vielen Stellen bei der Eingabe hilft und so z.B. Ergänzungsvorschläge macht, eine Klammerung schon komplett einsetzt oder Programmteile automatisch einrückt. Beispiel für eine Klasse, die Sie ergänzt haben:

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

Denken Sie daran, dass Sie in diesem Semester für jede Übungsaufgabe (=Klasse) auch ein neues Projekt anlegen.

6 Java-Programm übersetzen und starten

Wenn Sie eine Klasse fertig programmiert haben, starten Sie die Programmausführung: über das Menü

Run -- Run

oder einfach in der unteren Menüleiste das Icon mit dem weißen Pfeil im grünen Kreis anklicken. Wenn Sie gefragt werden, ob das Programm gestartet werden soll als Java Applet oder als Java Application, so antworten Sie mit Java Application.
Wenn Sie die Datei noch nicht gespeichert hatten bzw. noch Änderungen vorhanden sind, die nicht gespeichert sind, werden Sie ggfs. zuerst gefragt, ob und was Sie vor der Ausführung speichern wollen. Antworten Sie in diesem Fall mit

OK

(also speichern).
eclipse übersetzt ihr Programm vor der eigentlichen Ausführung. Sollte die Übersetzung erfolgreich sein (d.h. keine Syntax- und Semantikfehler), so erscheint im unteren Fenster mit dem Reiter **Console** die Ausgabe vom Programm.
Sollten bei der Programmübersetzung vom Java Compiler Fehler erkannt werden, so werden diese mit einem weißen Kreuz auf einem runden Kreis links am Editor-Fenster angezeigt. Wenn Sie mit der Maus dort drüberfahren, so werden Ihnen Hinweise zu dem Fehler gegeben.

7 Programmausgabe

Alle Textausgaben ihres Programms (also beispielsweise von **System.out.println()**) erscheinen in dem unteren Fenter unter dem Reiter **Console**.

8 Wo finde ich die Java-Programmdatei?

Sie finden ihre Programmdatei mit Endung **.java** (die Sie für den Praktomat einreichen müssen) im Workspace-Verzeichnis von *eclipse* , dort im Unterordner zu ihrem Projekt und dort im Unterordner **src**.

9 Datenübergabe an ein Programm

Oft sollen in Übungsaufgaben Daten an ein Programm übergeben werden, mit denen dann im Programm gearbeitet / gerechnet werden kann. Dazu gibt es verschiedene Möglichkeiten, von denen einige einfache Möglichkeiten sind:

1. Eingabe über die Kommandozeile. Beispielprogramm in Java:

```
public class Kommandozeile {
        public static void main(String[] args) {
                System.out.println("Hello World!");
                for(int i=0; i<args.length; i++)
```

```
                        System.out.println(args[i]);
        }
}
```

Bei diesem Programm kann man über die "'Kommandozeile"' Daten übergeben. In *eclipse* können solche Daten eingegeben werden über:

`Run -- Run Configurations -- (x)=Arguments`

Geben Sie in dem erscheinenden Teilfenster mit der Beschriftung **`Program Arguments`** die Argumente für das Programm in das entsprechende Teilfenster ein. Sie können diese hintereinander in eine Zeile schreiben, können sie aber auch über mehrere Zeilen verteilen. Solche Argumente können Zahlen, Buchstabenfolgen und anderes sein. Beispieleingabe:

```
Dies ist mein zweites Programm
```

Klicken Sie dann auf **`Run`**. Das obige Programm würde bei den angegebenen Argumenten folgende Ausgabe produzieren (jedes Wort in einer eigenen Zeile):

```
Hallo
Dies
ist
mein
zweites
Programm
```

2. Einlesen zur Laufzeit von der Konsole. Beispielprogramm in Java:

```
import java.util.*;
public class Einlesen {
    public static void main(String[] args) {
        // Scanner von der Tastatur anlegen
        Scanner sc = new Scanner(System.in);
        // einen ganzzahligen Wert einlesen
        int i = sc.nextInt();
        // und wieder ausgeben
        System.out.println(i);
    }
}
```

Wenn Sie dieses Programm starten, erwartet das Programm die Eingabe eines ganzzahligen Wertes von der Tastatur. Klicken Sie dazu auf das untere Fenster mit dem Reiter **`Console`** und geben dort (in dem Fenster) einen ganzzahligen Wert ein. Beispiel 4711. Sobald Sie auf die Enter-Taste drücken, erscheint in dem gleichen Fenster unter der Zahl in einer neuen Zeile diese Zahl nochmals, diesmal als Ausgabe von Ihrem Programm (**`System.out.println(i)`**).

3. Einlesen aus einer Datei. Umfangreiche Daten will man normalerweise nicht eintippen, sondern diese liegen z.B. in Dateien vor. Das folgende Programm liest von einer Datei **`test.txt`** einen ganzzahliegen Wert ein und gibt den Wert wieder (auf der Konsole) aus.

```
import java.util.*;
import java.io.*;
public class Dateieingabe {
    public static void main(String[] args) {
        Scanner sc = null;
        // Scanner von der Datei "test.txt" anlegen
        try {
            sc = new Scanner(new File("test.txt"));
        } catch(FileNotFoundException e) { System.out.println("Problem"); }
        int i = sc.nextInt();
        System.out.println(i);
    }
}
```

Legen Sie dazu die Datei **`test.txt`** im Verzeichnis zu dem Projekt ab, also z.B. in **`~/workspace/projektname`**