Blatt 0

1 Aufgabe 1 - How to Java

Die erste Aufgabe soll zeigen, wie man ein Java Programm auch ohne IDE kompilieren und ausführen kann. Hierfür werden die beiden Programme `java` und `javac` benötigt. Ersteres wird mit der Java JRE standardmäßig mitgeliefert. Für zweiteres muss man die Java JDK herunterladen. Beides ist unter dem Linuxaccount vom ZID installiert.

In einem Texteditor können wir nun folgendes Programm erstellen:

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

Dieses speichern wir in der Datei **MyFirst.java** ab.

Nun können wir das Programm auf der Kommandozeile mit dem Befehl `javac MyFirst.java` kompilieren. Hierbei erhalten wir nun eine Datei namens **MyFirst.class**, welche den Java Bytecode enthält. Diesen können wir mit Hilfe des Befehls `java MyFirst` ausführen.

Wichtig ist dabei, dass die Klasse und die Datei den gleichen Namen haben. Parameter, die wir über die Kommandozeile übergeben, werden in der String-Array `args` gespeichert. Diese können wie gewohnt mit `args[n]`, wobei n ein Integer Wert ist, ausgelesen werden. Wie in **C/C++** ist der Einstiegspunkt in das Programm die `main` Methode. In Java muss diese Methode `public` und `static` sein (Was diese Bezeichner bedeuten, wird im Laufe der Lehrveranstaltung noch erklärt).

2 Aufgabe 2 - BlueJ

Im Zuge dieser Lehrveranstaltung werden wir mit 2 verschiedenen Entwicklungsumgebungen arbeiten. Eine davon ist BlueJ. Hier können wir die Jar Datei herunterladen und diese mit java -jar bluej.jar installieren. Das Programm danach einfach mit BlueJ ausführen.

Nachdem wir das Programm gestartet haben, legen wir als erstes ein neues Projekt an. Das Projekt ist bis auf eine `readme` Datei leer. Jetzt legen wir eine neue Klasse an. Diese nennen wir wieder `MyFirst`. Mit einem Doppelklick können wir die Klasse öffnen. Die Klasse enthält nun einen Beispielcode, bis auf die Klassendefinition löschen wir diesen Code heraus, und ergänzen die Klasse um unsere `main` Methode aus der Aufgabe 1.

Nachdem wir die Klasse kompiliert haben, können wir sie im Diagramm mit einem Klick mit der rechten Maustaste auf die Klasse mit der Auswahl der `main` Methode das Programm ausführen. Nun sollte ein Konsolenfenster aufgehen, dass die Ausgabe anzeigt. Nun erweitern wir unsere `main` Methode um folgenden Code:

```
int x, y, z;
x = 2;
y = 3;
z = x * y;
System.out.println("Ergebnis: "+ x +" * "+ y +" = "+ z);
```

in der Zeile `x = 2` setzen wir nun mit `Strg + [B]` einen *Breakpoint*. Wenn wir die Klasse nun kompilieren und ausführen, öffnet sich der Debugger. Hier können wir Schrittweise die Anweisungen verfolgen, sowie die Werte der Variablen nachvollziehen.

3 Aufgabe 3 - Eclipse

Die zweite Entwicklungsumgebung die wir verwenden ist eclipse. Diese sollte bereits vom ZID sowohl unter Windows und wie auch Linux installiert sein.

In eclipse legen wir auch ein neues Projekt an und wie gewohnt die Klasse MyFirst. Im Erstellungsdialog der Klasse können wir gleich auswählen, dass sie eine `main` Methode beinhalten soll. Auch hier fügen wir den Code unserer bereits bekannten main Methode ein. Mit der rechten Maustaste können wir nun in die Methode klicken, dort sagen **Run As** > **Java Application** und damit das Programm ausführen.

Unseren Breakpoint können wir setzen, indem wir am linken Bildschirmrand bei der Zeile, bei der wie ihn einfügen wollen, mit der rechten Maustaste klicken und sagen **Toggle Breakpoint**. Debuggen können wir ähnlich wie das Programm ausführen, nur anstatt **Run as** sagen wir in diesem Fall **Debug as** > **Java Application**. Die Debug View öffnet sich, in der wir wie gewohnt unser Programm schrittweise ab dem Breakpoint ausführen können, und die Werte der Variablen überprüfen.