Übung 01: Einführung in Java und IntelliJ

Aufgabe 1: Einrichten von IntelliJ

Für die Vor- und Nachbereitungen der Übungen sowie für die PStAt wird empfohlen, IntelliJ auf dem eigenen PC zu installieren.

a) Installieren Sie zunächst eine aktuelle Version des Java Development Kits. Auf den meisten aktuellen PCs können Sie die 64-Bit Version verwenden:
https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

b) Finden Sie heraus, in welchem Verzeichnis der Java Compiler (`javac.exe`) bzw. der Java Interpreter (`java.exe`) liegt. Sorgen Sie dafür, dass die Pfad-Variable richtig gesetzt ist
https://javatutorial.net/set-java-home-windows-10
http://www.sajeconsultants.com/how-to-set-java_home-on-mac-os-x/
https://docs.opsgenie.com/docs/setting-java_home

c) Installieren Sie die Entwicklungsumgebung IntelliJ. Verwenden Sie die Community Edition in der aktuellen Version. Sie können auch die Ultimate Edition bekommen, wenn Sie sich dafür mit Ihrer TH Rosenheim Emailadresse registrieren (Nicht nötig für diese Veranstaltung).
https://www.jetbrains.com/idea/

Aufgabe 2: „Hallo Welt“

a) Starten Sie IntelliJ und legen Sie ein neues, leeres Java Projekt in einem Zielordner Ihrer Wahl an.
- *Hinweis:* Achten Sie darauf, dass unter „Project SDK“ ein korrekter Pfad für ein Java Development Kit definiert ist.
- Welche Unterordner werden in Ihrem Zielordner angelegt?

b) Erzeugen Sie eine leere Java-Klasse `HelloWorld.java` im Unterordner `src`!

c) IntelliJ kann häufig verwendete Codepassagen selbstständig erzeugen. Erzeugen Sie eine `main`-Methode unter Verwendung eines sogenannten ***Live Templates****!*
Hinweis: Entweder *„Code-Insert Live Template“* oder direkt *„Strg+J“* und anschließend *„psvm“*. (für was steht wohl „psvm“?)

d) Als nächstes verwenden Sie ***Auto Completion****,* um effizient Code zu erzeugen, der „Hallo Welt“ auf der Konsole ausgibt! Dazu ist ein Java die folgende Anweisung notwendig:
`System.out.println(„Hallo Welt“);`
Beginnen Sie diese Anweisung innerhalb der in c) erzeugten Methode zu tippen und beobachten Sie, welche Vorschläge Ihnen IntelliJ macht. Mit `Enter` wird ein Vorschlag übernommen, mit `Strg+“.“` wird ein Vorschlag übernommen und noch ein Punkt eingefügt.

e) Führen Sie nun Ihr Programm aus!
Hinweis: Entweder Rechtsklick auf die Datei im Projektfenster und „`Run`“ auswählen oder „`Alt + Umschalt + F10`“

Aufgabe 3: Debuggen in IntelliJ

a) Gegeben sei der rechte Java-Code. Geben Sie für ***jede*** Zeile den Wert der Variable(n) an, die sich in der jeweiligen Zeile ändert bzw. ändern!
Lösen Sie diese Aufgabe zunächst auf dem Papier und geben Sie an, welche Belegung das Array nach Ausführung von Zeile 8 hat.

```
1:      int input[] = {3, 4, 48, 0};
2:      int a = input[2] - input[1];
3:      int b = input[1] * input[2];
4:      int c = b / a;
5:      int d = b % a;
6:      b = ++c * a++ + d--;
7:      int[] output = new int[4];
8:      output[2] = b;
```

b) Debugger erlauben die schrittweise Ausführung eines Programmteils und zeigen u.a. die Belegung der Variablen zum aktuellen Zeitpunkt der Ausführung an. Kopieren Sie den Code in eine `main`-Methode einer neu zu erstellen Klasse `Debugging.java`!
Setzen Sie einen *Breakpoint* am Anfang der `main`-Methode (links neben die betreffende Codezeile klicken), starten Sie Ihr Programm im Debug Modus (siehe „Run“-Menü) und gehen Sie schrittweise durch den Code (F8).
Verifizieren Sie Ihr Ergebnis aus Aufgabe a).

c) Welchen Wert hat die Variable `b` nach Ausführung des folgenden Codes? Überlegen Sie sich das Ergebnis manuell! Falls notwendig, verifizieren Sie es über ein Programm!

```
boolean b;
int a = 7, c = 22, d;
d = (c / a) * 2;
b = (( c % a) <= (c / a)) && (d == 6);
```

Aufgabe 4: Ein- und Ausgabe auf der Konsole

a) Kopieren Sie die Datei `Benzin.java` aus der Community in den `src`-Ordner Ihres Projektes.

b) Bereinigen Sie den schlampig formatierten Code in `Benzin.java` mit „`Code-Reformat Code`“.

c) Ein Programmierer hat in der Datei `Benzin.java` versehentlich den Bezeichner `kmh` anstatt `km` verwendet. Korrigieren Sie dies durch ***Refactoring*** (Tipp: „`Code-Refactor`“).

d) Versuchen Sie das Programm grob zu verstehen und lassen Sie das Programm laufen.

e) Schreiben Sie nun ein eigenes, ähnliches Programm `Sekunden` in der Datei `Sekunden.java`, das Stunden und Minuten von der Tastatureingabe einliest, anschließend diese in Sekunden umrechnet und das Ergebnis ausgibt.

Link: **Default Keymaps von IntelliJ**
https://resources.jetbrains.com/storage/products/intellij-idea/docs/IntelliJIDEA_ReferenceCard.pdf