von Ulrich Grude

Inhaltsverzeichnis

Die Übungen für die Lehrveranstaltung MB1-PR1 (Programmieren 1) finden im Labor für Software-Engineering (**SWE-Labor**) statt. Am Anfang des Semesters sollten Sie sich auf einem Rechner dieses Labors einloggen und Ihr dortiges Konto *einrichten*. Wenn Sie für Programmieren 1 auch einen eigenen Rechner benutzen wollen, sollten Sie auch den dafür *einrichten*. Dieses Papier enthält Hinweise für diese Einrichtungen (ein paar geeignete Möbel finden Sie im Archiv `pr1_Moebel.zip`).

Konvention: In diesem Papier gilt: **+ABC** bedeutet: Sie sollen auf das Pluszeichen **+** vor **ABC** klicken, *nicht* auf das Wort **ABC**.
Konkrete Beispiele für diese Notation: **+Extras**, **+Java**, **+Dokumentenklasse** ... etc.

1. Einrichten im SWE-Labor

Auf den Rechnern im SWE-Labor ist (unter anderem) ein Windows Betriebssystem und folgende Software bereits installiert:

Ein aktuelles **Java**-System (Java 8)
Der Editor **TextPad** (Version 7)
Die Office-Software **LibreOffice**

Am Anfang des Semesters sollten Sie diese Software noch ein bisschen "einrichten und anpassen", selbst wenn Sie vorhaben, meistens auf Ihrem eigenen Laptop zu arbeiten (unbenutzte Konten werden eventuell wieder gelöscht und von den Laborrechnern aus können Sie auch drucken).

Grundregel: Im SWE-Labor ist das Laufwerk `Z:` Ihr Heimatordner, in dem Sie "alles machen dürfen" (neue Ordner und Dateien anlegen, lesen, schreiben, löschen etc.). Sie sollten keine Daten auf das Laufwerk `C:` schreiben.

Nur in ganz seltenen Fällen sollen Sie (auf eine ganz bestimmte Weise) von dieser **Grundregel** abweichen (siehe unten).

Es folgen hier die einzelnen Schritte die Sie unternehmen sollen, um Ihr Konto im SWE-Labor einzurichten. Beim Lesen sollten Sie den Benutzernamen **`s123456`** durch Ihren eigenen Benutzernamen (der vermutliche eine ganz ähnliche Form hat) ersetzen.

Schritt 0: Loggen Sie sich im SWE-Labor an einem der Computer ein. Ihr Benutzername besteht aus einem `s` gefolgt von Ihrer Matrikel-Nr. (so ähnlich wie: **`s123456`**). Beim ersten Einloggen ist Ihr Passwort gleich **`swe`** (3 kleine Buchstaben). Wenn Sie dann aufgefordert werden, ein neues Passwort einzugeben, können Sie z.B. das gleiche Passwort angeben, wie für Ihr Konto beim HRZ (Hochschulrechenzentrum), oder irgendein anderes.

Schritt 1: Legen Sie einen Ordner mit dem Pfadnamen `Z:\Klassen` an. In diesem Ordner sollen im Laufe des Semesters alle Java-Klassen-Dateien (`.class`-Dateien) abgelegt werden.

Schritt 2: Schreiben Sie den Pfadnamen `Z:\Klassen` in die Umgebungsvariable `CLASSPATH` (siehe unten den Abschnitt **3.2. Den CLASSPATH ändern im SWE-Labor**).

Überprüfen Sie diesen Schritt, indem Sie eine **Eingabeaufforderung** öffnen (d.h. das Programm `cmd.exe` starten) und das Kommando `echu c` eingeben (nicht `echo c`!).

Schritt 3: Starten Sie das Programm TextPad (Linksklick auf den Windows-**Start-Knopf**, **Alle Programme**, **TextPad 7.4**) und beenden Sie es nach ein paar Sekunden wieder. Bei diesem ersten Start legt der TextPad ein paar Ordner an, die im nächsten Schritt gebraucht werden.

Schritt 4: Besorgen Sie sich das Archiv `pr1_Moebel.zip`. Es sollte (unter anderem) die drei Dateien `java.tcl`, `Comment.tpm` und `Uncomment.tpm` enthalten. Kopieren Sie diese drei Dateien in den Ordner

`C:\Users\`**`s123456`**`\AppData\Roaming\Helios\TextPad\7`

(natürlich nur, wenn Ihr Benutzername gleich **`s123456`** ist :-))

Anmerkung: Dies ist wahrscheinlich das einzige Mal, dass Sie im SWE-Labor Daten auf das Laufwerk `C:` schreiben sollen.

Schritt 5: Überprüfen Sie, ob der vorige Schritt die erhoffte Wirkung hat.
Starten Sie dazu den TextPad. Falls er schon lief, sollten Sie ihn beenden und erneut starten. Nur beim Start sucht der TextPad nach **Textbausteinen** (`.tcl`-Dateien) und **Makros** (`.tpm`-Dateien).

Schritt 5.1: Klicken Sie ein- oder zweimal auf **Menü Ansicht**, **Textbausteine**,
bis links oben im TextPad-Fenster das Wort `Textbausteine` erscheint und darunter der Name einer Textbausteine-Sammlung, z.B. `ANSI-Characters` oder `HTML-Tags` oder so ähnlich.

Links-klicken Sie auf diesen Namen. Dadurch sollte eine Liste aufgehen, die unter anderem den Namen `Java` enthält (das ist der Name der Textbausteine-Sammlung in der Datei `java.tcl`, die Sie im Schritt 4 kopiert haben). Links-klicken Sie auf den Namen `Java`.

Falls der Name `Java` nicht in der Liste vorkommt, ist etwas schief gegangen. Überprüfen Sie, ob der **Schritt 4** korrekt durchgeführt wurde. Wenn das nicht hilft, dann wenden Sie sich an den nächsten fachkundigen Menschen, den Sie erreichen können.

Schritt 5.2: Die beiden TextPad-Makros **Comment** und **Uncomment** müssen noch aktiviert werden, bevor man sie benutzen kann. Das geht so: Den TextPad starten. Dann:
Menü Konfiguration, **Einstellungen**, **Makros**.

Zwei Fensterchen mit 6 Knöpfen dazwischen sollten erscheinen. Im linken Fensterchen (**Verfügbare Makros:**) sollten unter anderem die Namen **Comment** und **Uncomment** stehen. Klicken Sie auf **Comment** und dann auf **Hinzufügen>>** (und dann ebenso mit **Uncomment**). Danach sollten die beiden Makros im rechten Fensterchen (**Makros im Menü:**) stehen. Klicken Sie auf **Übernehmen** und **OK** um das **Einstellungen**-Fenster zu schließen.

Danach sollten im TextPad-Menü **Makros** die beiden Makros **Comment** und **Uncomment** zur Auswahl stehen. Falls das nicht der Fall ist: Überprüfen Sie, ob der **Schritt 4** korrekt durchgeführt wurde. Wenn das nicht hilft, dann wenden Sie sich an den nächsten fachkundigen Menschen, den Sie erreichen können.

Schritt 6: Jetzt müssen Sie noch den TextPad dazu bewegen, alle neuen Java-Klassen-Dateien (`.class`-Dateien) in den dafür vorgesehenen Ordner `Z:\Klassen` zu schreiben (statt sie im aktuellen Arbeitsverzeichnis abzulegen). Starten Sie dazu den TextPad. Dann **Menü Konfiguration**, **Einstellungen**, **+Extras** (d.h. Sie sollen auf das Pluszeichen + vor **Extras** klicken), **Java Compilieren**. Jetzt sollte rechts oben in einem Text-Eingabefeld (rechts neben **Parameter:**) der Text `$File` stehen. Ersetzen Sie diesen Text durch `-d Z:\Klassen $File` und klicken Sie auf **Übernehmen** und **OK**. Das `d` in `-d` soll an `directory` (Verzeichnis, Ordner) erinnern.

Schritt 7: Als vorläufigen Abschluss der Einrichtungs-Arbeiten sollten Sie noch ein kleines Java-Programm erstellen, dem Ausführer übergeben und ausführen lassen. Wie das geht wird unten im Abschnitt **2.5 Ein kleines Java-Programm zusammen klicken** beschrieben.

2. Den eigenen Rechner einrichten

Falls Sie keinen Windows-Rechner haben, sollten Sie versuchen, sich einen zu leihen oder auf andere Weise zu besorgen (gebrauchte, aber noch gut brauchbare Laptops gibt es schon ab etwa 200 Euro). Als Alternative zu einem eigenen Windows-Rechner gibt es noch die Möglichkeit, sich von einem Linux- oder MacOS-Rechner aus mit einem **X2Go**-Klienten mit den Windows-Servern im SWE-Labor zu verbinden. Wenden Sie sich vertrauensvoll an die Ingenieure des SWE-Labors (an Herrn **Brüntrup** und Herrn **Shahbaz**) wenn diese Möglichkeit Sie interessiert. Auf jeden Fall müssen Sie in der Lage sein, bestimmte Übungen mit dem Editor TextPad durchzuführen (den es leider nur für Windows gibt).

Auf Ihrem eigenen (oder geliehenen) Windows-Rechner sollte (für die LV Programmieren 1) folgende Software zur Verfügung stehen:

1. Eine Java-Entwicklungs-Umgebung (JDK) für **Java 8** der Firma Sun/Oracle
(siehe dazu: http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html).

2. Eine aktuelle Version von **OpenOffice** oder **LibreOffice**
(siehe dazu: http://www.openoffice.de/ bzw. https://de.libreoffice.org/).

3. Der Editor **TextPad**, am besten die Version 7 (wie im SWE-Labor, nicht die neuste Version 8)
(siehe dazu: http://www.textpad.com/).

Der TextPad ist ein ziemlich ausgereifter Editor, den man auch als relativ einfache (d.h. weitgehend durchschaubare und verstehbare) Entwicklungsumgebung für Java-Programme benutzen kann. Er "spielt in der gleichen Liga" wie z.B. die Editoren Notepad++ und Ultraedit. Eine Einzelplatz-Lizenz kostet 16,50 £ (das sind etwa 22 Euro, solange England zur EU gehört). Bevor man eine (bezahlte) Lizenz-Nr eingibt, wird man regelmäßig (aber nicht übermäßig oft) zum Kauf aufgefordert, kann den TextPad aber *ohne Einschränkungen* und *ohne zeitliche Begrenzung* benutzen.

2.1. Java vor dem TextPad installieren

Wenn Sie den Java Development Kit (JDK) von Sun/Oracle *vor* dem TextPad installieren, dann "merkt der TextPad das und verbindet sich mit dem JDK". Andernfalls müssen Sie die Verbindung wie folgt selbst herstellen:

Den TextPad starten. Dann **Menü Konfiguration**, **Einstellungen**, **Extras.**

Rechts oben auf **Hinzufügen** klicken und **Java SDK Befehle** wählen.

2.2. Das echu-Kommando installieren

Besorgen Sie sich das Archiv `pr1_Moebel.zip`. Es sollte (unter anderem) eine Datei namens `echu.exe` enthalten. Kopieren Sie diese Datei in einen Ordner, dessen Pfadname in der Umgebungsvariablen `PATH` steht (z.B. in den Ordner `C:\Windows`).

So können Sie prüfen, ob das `echu`-Kommando funktioniert:
Starten Sie eine **Eingabeaufforderung** (d.h. das Programm `cmd.exe`). Geben Sie das Kommando `echu` (und Return) ein.
Danach ebenso die Kommandos `echu c`, `echu p` und `echu computername`.
Die Ausgabe sollte jedes mal lesbar sein (und nicht unlesbar wie z.B. die Ausgabe der Kommandos `echo %path%` oder `echo %classpath%`). Der Name "echu" soll an "echo für **U**mgebungsvariablen" erinnern.

2.3. Einen Ordner für alle Java-Klassen-Dateien anlegen

Auf Ihrem eigenen Rechner können Sie den Ordner für alle Java-Klassen-Dateien an einer beliebigen Stelle in Ihrem Dateisystems anlegen. Empfohlen wird ein Ort mit einem kurzen Pfadnamen. Hier wird angenommen, dass Ihr `Klassen`-Ordner den Pfadnamen `D:\Klassen` hat.

Schreiben Sie den Pfadnamen `D:\Klassen` in die Umgebungsvariable `CLASSPATH` (siehe unten den Abschnitt **3.3. Den CLASSPATH ändern auf Ihrem eigenen Rechner**).

Überprüfen Sie diesen Schritt, indem Sie das Kommando `echu c` in eine **Eingabeaufforderung** eingeben.

2.4. Ordner für TextPad Textbausteine und Makros

Das Archiv `pr1_Moebel.zip` sollte (unter anderem) die drei Dateien `java.tcl`, `Comment.tpm` und `Uncomment.tpm` enthalten. Um herauszufinden, in welchen Ordner Sie diese Dateien kopieren müssen (damit der TextPad sie auch findet) gehen Sie so vor:

TextPad starten. **Menü Konfiguration**, **Einstellungen**, **Ordner**.

Eine Liste mit zwei Spalten, **Dateiart:** und **Ordner:** sollte sichtbar werden.
Kopieren Sie die `.tpm`-Dateien in den für die Dateiart **Makros** angegebenen Ordner.
Kopieren Sie die `.tcl`-Datei in den für die Dateiart **Textbausteine** angegebenen Ordner.

Anmerkung: Sehr wahrscheinlich gilt:
Alle drei Dateien müssen in *denselben* Ordner kopiert werden, der Name des Ordners ist `Samples` und er enthält schon mehrere `.tcl`-Dateien und mehrere `.tpm`-Dateien.

2.5. Ein kleines Java-Programm zusammen klicken

Starten Sie den TextPad. Links vom Hauptfenster sollte ein schmales Fenster mit den **Textbausteinen** für **Java** sichtbar sein (sonst sollten Sie noch mal auf **Menü Ansicht, Textbausteine** klicken).

1. Doppelklicken Sie auf den Textbaustein namens **Klasse (mit main, pln)**. Dadurch sollten etwa 19 Zeilen Text in das Dokument im Hauptfenster kopiert werden.
2. Wählen Sie in der obersten Zeile dieses Textes die Zeichenkette `XX` mit der Maus aus und drücken Sie dann auf F8.
3. Das **Ersetzen**-Fenster öffnet sich. Tragen Sie in die Texteingabe rechts von **Ersetzen durch:** den Namen `Hallo37` (oder einen noch schöneren Namen) ein und klicken Sie auf **Alle ersetzen**.
4. Wählen Sie in der obersten Zeile des Textes die Zeichenkette `Hallo37.java` mit der Maus aus und kopieren Sie sie in die Ablage (mit Strg-C).
5. Klicken Sie auf **Menü Datei, Speichern** und drücken Sie dann Str-V (Einfügen). Speichern Sie die Datei `Hallo37.java` in irgend einen geeigneten Ordner.
6. Klicken Sie auf **Menü Extras, Benutzer-Programme, Java compilieren**.
7. Klicken Sie auf **Menü Extras, Benutzer-Programme, Java-Programm starten**.

Anmerkung: Wenn Sie den Instruktionen 1. bis 7. genau gefolgt sind, dann haben Sie nur einmal eine Zeichenkette (nämlich `Hallo37`) eingetippt und das mehrfache Kopieren dieser Zeichenkette mit dem Befehl **Alle ersetzen** und copy-and-paste (Str-C, Str-V) erledigt. Dadurch haben Sie bestimmte Fehler sehr unwahrscheinlich gemacht (z.B. dass die Klasse `Hallo37` aus Versehen anders heißt als die Datei `Hallo37.java`). Folgen Sie jedes mal, wenn Sie ein neues Java-Programm entwickeln, genau den Instruktionen 1. bis 7. (und ersparen Sie sich damit unnötigen Ärger und Zeitverlust).

2.6. JUnit installieren

Sie werden im Laufe des Semesters häufig **JUnit-Testprogramme** compilieren und ausführen lassen. Dazu müssen Sie auf Ihrem Rechner die **Version 3.8** von JUnit (nicht eine neuere Version wie 4. 3 oder 4.12 etc.) installieren wie folgt:

1. Laden Sie von der Adresse https://sourceforge.net/projects/junit/files/junit/3.8.2/
das Archiv `junit3.8.2.zip` herunter (ca. 500 KB).
2. Dieses Archiv enthält einen Ordner `junit3.8.2` und dieser Ordner enthält eine Datei namens `junit.jar`. Kopieren Sie diese Datei an einen beliebige Ort in Ihrem Dateisystems. Empfohlen wird ein Ordner mit einem kurzen Pfadnamen. Hier wird angenommen, dass dieser Ordner den Pfadnamen `D:\jars\junit.jar` hat (dazu müssen Sie also den Ordner `D:\jars` erstellen und die Datei `junit.jar` hineinkopieren).
3. Schreiben Sie den Pfadnamen `D:\jars\junit.jar` in die Umgebungsvariable `CLASSPATH`.

Überprüfen Sie den Teilschritt 3., indem Sie das Kommando `echu c` in eine **Eingabeaufforderung** eingeben.

Überprüfen Sie dann Ihre JUnit-Installation wie folgt:

Besorgen Sie sich die Datei **pr1_Aufgaben.odt** und kopieren Sie daraus den Text des Programms `StringBuilder00Jut` in eine Datei namens `StringBuilder00Jut.java`. Öffnen Sie diese Datei mit dem TextPad und versuchen Sie, sie zu compilieren und auszuführen (wie im Abschnitt **2.5.** bzw. im Abschnitt 1, **Schritt 6** beschrieben). Wenn das problemlos klappt, ist Ihre JUnit-Installation sehr wahrscheinlich in Ordnung (und sonst ist sie nicht in Ordnung).

3. Windows einrichten

3.1. Einführung Umgebungsvariablen

Die Umgebungsvariable **CLASSPATH** muss eine Folge von Pfadnamen enthalten, die durch Semikolons `;` voneinander getrennt sind. Jeder Pfadname sollte entweder ein `.jar`-Datei bezeichnen (z.B. `D:\jars\junit.jar`) oder einen Ordner, in dem sich `.class`-Dateien befinden (z.B. `D:\Klassen`). Wenn der Java-Ausführer (repräsentiert durch den Compiler `javac.exe` oder den Interpreter `java.exe` oder ...) eine `.class`-Datei benötigt, sucht er sie in allen `.jar`-Dateien und Ordnern, die im `CLASSPATH` eingetragen sind. Die `.jar`-Dateien und Ordner werden in der Reihenfolge durchsucht, in der sie im `CLASSPATH` stehen.

Der Inhalt der Umgebungsvariablen `CLASSPATH` kann z.B. so aussehen:

```
D:\jars\junit.jar;D:\Klassen;C:\KarlHeinz\Klassen
```

Die Umgebungsvariable **PATH** muss eine Folge von Pfadnamen enthalten, die durch Semikolons `;` voneinander getrennt sind. Jeder Pfadname sollte einen Ordner bezeichnen, in dem sich *ausführbare Dateien* befinden. Als *ausführbar* gelten Dateien mit Erweiterungen wie `.exe`, `.cmd`, `.bat` (und weitere). Wenn ein Windows-Betriebssystem eine ausführbare Datei benötigt, sucht es sie in allen Ordnern, die in der `PATH`-Variablen eingetragen sind. Die Ordner werden in der Reihenfolge durchsucht, in der sie im `PATH` stehen.

Außer `PATH` und `CLASSPATH` gibt es noch viele weitere Umgebungsvariablen, die hier aber nicht von Interesse sind. Wenn man in einer **Eingabeaufforderung** das Kommando `set` eingibt, werden einem die Namen und Inhalte *aller* Umgebungsvariablen angezeigt.

Es ist üblich, die Namen von Umgebungsvariablen mit Großbuchstaben zu schreiben. Man kann aber z.B. statt `PATH` auch `Path`, `path` oder `pAtH` schreiben.

Anmerkung: Umgebungsvariablen (engl. environment variables) gibt es nicht nur unter Windows, sondern in ähnlicher Form auch unter Unix-Betriebssystemen (z.B. Linux) und unter dem MacOS.

3.2. Den CLASSPATH ändern im SWE-Labor

Um z.B. den Pfadnamen `Z:\Klassen` in die Umgebungsvariable `CLASSPATH` einzutragen, können Sie so vorgehen:

Links-Mausklick auf **Start**, **Systemsteuerung**.

Rechts oben in die Texteingabe eingeben: `Umgebungsvar` und Return.

Auf **Umgebungsvariablen für dieses Konto bearbeiten** klicken.

Ein Fensterchen **Umgebungsvariablen** sollte aufgehen. Es besteht aus einer oberen und einer unteren Hälfte. Die untere (für **Systemvariablen**) ist nur für Administratoren benutzbar. Die obere Hälfte hat die Überschrift **Benutzervariablen für s123456** (falls Ihr Benutzername gleich s123456 ist).

Wenn in der Spalte **Variable** noch keine Variable namens `CLASSPATH` angezeigt wird, klicken Sie auf **Neu...** und legen eine Variable mit dem Namen `CLASSPATH` und dem Wert
`Z:\Klassen`
an. Falls es schon eine Variable `CLASSPATH` gibt, wählen Sie sie aus, klicken Sie auf **Bearbeiten...** und schreiben Sie hinter den schon vorhandenen Inhalt die 9 Zeichen: `;\Klassen`

3.3. Den CLASSPATH ändern auf Ihrem eigenen Rechner

Sie können ganz ähnlich vorgehen, wie im vorigen Abschnitt **3.2.** beschrieben. Da Sie auf Ihrem eigenen Rechner die Rechte eines Administrators haben (oder sich verschaffen können), können Sie (statt

einer **Benutzervariablen**) auch den Inhalt einer **Systemvariablen** festlegen, die dann einheitlich für alle Benutzer Ihres Rechners gilt (das ist häufig empfehlenswert).

3.4. Den PATH ändern

Die Umgebungsvariable `PATH` zu ändern geht ganz entsprechend wie beim `CLASSPATH` (siehe die beiden vorangehenden Abschnitte **3.2.** bzw. **3.2.**). Aber seien Sie vorsichtig: Wenn man den `CLASSPATH` falsch ändert sind "nur" Java-Programme betroffen. Wenn man den `PATH` falsch verändert läuft eventuell kein Programm mehr (auch keines, mit dem man den `PATH` wieder reparieren könnte!).

3.5. Eine Eingabeaufforderung anpassen

Eine **Eingabeaufforderung** können Sie auf verschiedene Weisen öffnen:

Linksklick auf den Startknopf. **Alle Programme, Zubehör, Eingabeaufforderung.**

Linksklick auf den Startknopf. Unten in die Texteingabe `cmd.exe` eingeben. Return.

Windows-Taste `+ R`. Ein **Ausführen**-Fensterchen öffnet sich. Dort `cmd.exe` eingeben. Return.

Den Explorer öffnen und einen Ordner, in dem Sie arbeiten wollen, anzeigen lassen.
Mit gedrückter Umschalt-Taste ein Rechtsklick auf den Ordner.
Ein Kontextmenü öffnet sich. Darin ein Linksklick auf **Eingabeaufforderung hier öffnen**.

Standardmäßig ist eine Eingabeaufforderung ein Fenster mit schwarzem Hintergrund, schmutzig-weißem Text und einer kleinen Schrift. Als MedieninformatikerIn sollte man das nicht einfach hinnehmen (und zur Verbreitung eines schlechten Geschmacks beitragen :-)).

Empfehlung: Legen Sie einen Link auf die Eingabeaufforderung auf Ihren Desktop wie folgt:

Linksklick auf den Startknopf. **Alle Programme, Zubehör**. Mit gedrückter rechter Maustaste die **Eingabeaufforderung** auf den Desktop ziehen und (die Maustaste) loslassen. Ein Menü geht auf. **Verknüpfungen hier erstellen** wählen (auch wenn es nur um *eine* Verknüpfung geht :-)).

Die Eingabeaufforderung, die mit diesem Link gestartet wird, können Sie wie folgt anpassen:

Rechtsklick auf den Link. Ein Menü öffnet sich. Wählen Sie (ganz unten) **Eigenschaften**.
Ein **Eigenschaften für ...** Fenster öffnet sich mit etwa 10 Reitern darin (**Kompatibilität**, **Sicherheit**, **Details**, ...).

Reiter **Verknüpfung**: Geben Sie neben **Ausführen in:** den Pfadnamen des Ordners an, in dem Sie mit der Eingabeaufforderung (häufig/meistens/immer) arbeiten wollen.

Reiter **Allgemein**: Ersetzen Sie (ganz oben) den Text **Eingabeaufforderung** durch einen nützlicheren Text, z.B. durch den Namen des Ordners, in dem sie (häufig/meistens/immer) arbeiten werden.

Reiter **Schriftart**: Stellen Sie unter **Schriftgrad** eine gut lesbare Größe ein, z.B. `12x16`. Denken Sie dabei bitte auch an den Betreuer Ihrer Übungsgruppe, der wahrscheinlich Brillenträger ist!

Reiter **Layout**: Vergrößern Sie die **Fensterpuffergröße** (z.B. auf **Breite:** 100, **Höhe:** 1000)
und vergrößern Sie die **Fenstergröße** (z.B. auf **Breite:** 100, **Höhe:** 50).

Reiter **Farben**: Wählen Sie für den **Fensterhintergrund** die Farbe Weiß ("das Farbkästchen ganz rechts") und für den **Fenstertext** die Farbe Schwarz ("das Farbkästchen ganz links").

Schließen Sie das **Eigenschaften für ...** Fenster mit einem Klick auf **OK**.

Ein Links-Doppelklick auf den Link startet jetzt die "angepasste" Eingabeaufforderung. Ändern Sie alle Einstellungen, die Ihnen nicht gefallen.

Empfehlung: Häufig ist es günstig, sich *mehrere* solche Links-auf-die-Eingabeaufforderung auf den Desktop zu legen, und in jedem Link einen anderen (häufig benutzten) Ordner anzugeben (unter den Rei-

tern **Verknüpfung** und **Allgemein**). Man kann dann mit einem Links-Doppelklick auf den richtigen Link direkt "in den richtigen Ordner springen" (ohne zeitaufwendiges Navigieren).

3.6. Unix-Befehle unter Windows benutzen

Im SWE-Labor gilt: In einer Eingabeaufforderung kann man nicht nur die üblichen Windows-Eingabeaufforderungs-Kommandos (`cd`, `dir`, `SET`, `SETLOCAL`, `ENDLOCAL`, `CALL` etc.) ausführen lassen, sondern auch viele Kommandos der (in Unix-Systemen verbreiteten) **bash**-Shell (z.B. `ls`, `basename`, `cmp`, `grep`, `where` etc.). Wenn Sie in einer Eingabeaufforderung das Kommando `bash` eingeben, wird das Fenster zu einem bash-Shell-Fenster (in dem Sie nur noch bash-Kommandos ausführen lassen können). Mit dem Kommando `exit` kehren Sie in die Eingabeaufforderung zurück (in der Sie cmd-Kommandos und bash-Kommandos ausführen lassen können).

Wenn Sie auf Ihrem eigenen Rechner etwas Entsprechendes erreichen wollen, können Sie z.B. von der Adresse http://win-bash.sourceforge.net/ eine "bash-Shell für Windows" herunterladen und installieren.

4. TextPad für die erste Woche

4.1. Zeilen-Nummern einstellen

Im TextPad kann man zwei verschiedene "Arten von Zeilen-Nummern" unabhängig voneinander ein- und ausschalten, wie im Folgenden erläutert wird.

Art 1: Zeilen-Nummern auf dem Bildschirm

Die Nummern dieser Art erscheinen nur *auf dem Bildschirm*, werden aber nicht in die editierte Datei geschrieben und werden auch *nicht gedruckt*. Diese Art 1 kann man nur *pauschal für alle Klassen von Dokumenten* an- und ausschalten wie folgt:

Menü Konfiguration, **Einstellungen...**, **Ansicht**,
ganz unten vor **Zeilennummern** ein Häkchen anbringen (oder wieder entfernen).

Art 2: Zeilen-Nummern in Ausdrucken einstellen

Die Nummern dieser Art erscheinen *nur beim Ausdrucken* (**Menü Datei**, **Drucken**...), werden aber nicht in die editierte Datei geschrieben und werden auch nicht auf dem Bildschirm angezeigt. Diese Einstellung kann (und muss) man für jede *Dokumentenklasse* (z.B. für Java-Dateien, Text-Dateien, XML-Dateien etc.) einzeln an- bzw. ausschalten wie folgt:

Menü Konfiguration, **Einstellungen**, **+Dokumentenklasse** (d.h. auf das Pluszeichen **+** vor **Dokumentenklasse** klicken), **+Java**, **Drucken**
Vor **Zeilennummern** ein Häkchen anbringen (oder wieder entfernen).

4.2. Einrücktiefe und Tab-Zeichen einstellen

Die Einrückung von Programmtexten kann man durch *Tabulator-Zeichen* (kurz: Tab-Zeichen) oder durch *Leerzeichen* realisieren. Leerzeichen haben den Vorteil, dass sie von allen Editoren gleich dargestellt werden. Einige Programmierer halten Tab-Zeichen in Programmtexten deshalb für unhöflich. Stellen Sie Ihren TextPad bitte höflich ein wie folgt:

Menü Konfiguration, **Einstellungen**, **+Dokumentenklasse** (d.h. auf das Pluszeichen **+** vor **Dokumentenklasse** klicken), **+Java**, **Tabulator**

Nach **Standard-Tabstopps:** und nach **Einzugsgröße:**
je eine `3` eintragen.

Vor **Neue Tabulatoren in Leerzeichen umwandeln** und
vor **Vorhandene Tabulatoren beim Speichern in Leerzeichen umwandeln**
je ein Häkchen anbringen.

Anmerkung: Alle für die LV Programmieren 1 vorgegebenen Programme sind mit den hier vorgeschlagenen Einstellungen erstellt. Das Benutzen und Bearbeiten der Programme wird besonders einfach, wenn Sie die *gleichen* Einstellungen verwenden.

4.3. Den Partner einer Klammer finden (Tipp)

Wenn Sie (in irgendeinem TextPad-Fenster) den Cursor vor oder hinter einer *Klammer* positionieren (oder die Klammer auswählen) und `Strg-M` eingeben (M wie match), springt der Cursor automatisch zur "Partner-Klammer". Falls der Cursor sich nicht bewegt, gibt es keine Partner-Klammer.

Als Klammern gelten dabei die folgenden 8 Zeichen: `( [ { < > } ] )`

Dieser `Strg-M`-Befehl kann einem das Entfernen einer überzähligen oder das Einfügen einer fehlenden Klammer erheblich erleichtern. Das gilt besonders für *geschweifte Klammern*, die in Java-Programmen ziemlich tief geschachtelt vorkommen können. Eine einzige falsche geschweifte Klammer kann den Java-Ausführer (repräsentiert durch den Compiler `javac.exe`) ganz durcheinander bringen und zur Ausgabe zahlreicher (manchmal verwirrender) Fehlermeldungen veranlassen.

4.4. Java-Texte richtig einrücken

Wichtige Anmerkung: Die *Einrückung* eines Quelltextes ist *keine optionale Verzierung*, sondern ein wichtiges *Arbeitsmittel* (welches das Lesen des Textes und das Finden von Fehlern erheblich erleichtern kann).

Grundregel zum richtigen Einrücken: Wenn Sie ein Java-Programm mit dem TextPad editieren und eine *neue Zeile einfügen* wollen, dann sollten Sie immer
1. Den Cursor an des Ende der vorherigen Zeile positionieren
2. Auf Return drücken
3. Den Text der neuen Zeile eingeben

Solange Sie konsequent darauf verzichten, die Cursor-Tasten (←, ↑, → und ↓) zu benutzen oder den Cursor mit der Maus zu positionieren (anders als in der **Grundregel** beschrieben), rückt der TextPad den Text während der Eingabe automatisch korrekt ein, indem er sich an die folgenden 3 Regeln hält:

Regel E1: Wenn eine Zeile mit einer *öffnenden* geschweiften Klammer `{` *endet*, wird die nächste Zeile um eine Stufe *mehr* eingerückt als ihre Vorgängerin.

Regel E2: Wenn eine Zeile mit einem anderen Zeichen (ungleich `{`) *endet*, wird die nächste Zeile *genau so weit* eingerückt wie ihre Vorgängerin.

Regel E3: Wenn eine Zeile mit einer *schließenden* geschweiften Klammer `}` *beginnt*, wird sie um eine Stufe *weniger* eingerückt als ihre Vorgängerin.

Die Regeln E1 bis E3 beschreiben vollständig, wie der TextPad Java-Quelltexte richtig einrückt. Gewöhnen Sie sich an, ihn dabei möglichst wenig zu stören :-).

Anmerkung: Der TextPad kann Text nur *während der Eingabe* richtig einrücken. Wenn man einen Text chaotisch eingegeben hat (was mit Hilfe der Cursortasten ←, ↑, → und ↓ und der Maus ziemlich einfach ist :-), hilft der TextPad einem *nicht* mehr beim Formatieren. Man muss den Text dann (mühsam) von Hand formatieren oder mit einem geeigneten Programm (z.B. mit NetBeans oder Eclipse oder ...). Programmtexte gleich richtig eingerückt einzugeben ist deutlich eleganter und empfehlenswerter, als chaotisch eingeben und nachträglich reparieren.

5. TextPad für Fortschreiter

5.1. Das aktive Dokument

Mit dem TextPad kann man viele Dokumente (d.h. Dateien) öffnen und quasi gleichzeitig bearbeiten. In jedem Moment ist genau eines dieser Dokumente *das aktive Dokument* und das Fenster, in dem es angezeigt wird, ist *das aktive Fenster*. Vom aktiven Fenster sagt man auch, dass es *den Fokus hat*. Ein Fenster bekommt den Fokus unter anderem dann, wenn man irgendwo in das Fenster klickt oder auf seinen Reiter (engl. tab) oder auf seinen Namen in der Dateiliste klickt (Die Dateiliste ist ein schmales Fenster am linken Rand des TextPad-Fensters, das man mit **Menü Ansicht**, **Dateiliste** erscheinen oder verschwinden lassen kann).

Viele Befehle des TextPad beziehen sich immer auf *das aktive Dokument* (z.B. die Befehle **Java kompilieren** und **Java-Programm starten** im **Menü Extras, Benutzer-Programme**).

5.2. Der Befehl 'Java compilieren'

Diesen Befehl aufrufen: **Menü Extras**, **Benutzer-Programme**, **Java compilieren**.

Oder mit Tastenkürzel: **Strg-1**

Mit diesem Befehl übergibt man das aktive Dokument dem Java-Ausführer (genauer: dem Java-Compiler `javac.exe`). Falls das aktive Dokument kein korrektes Java-Quellprogramm (sondern z.B. ein

Kriminalroman oder ein fehlerhaftes Java Quellprogramm) ist, reagiert der Java-Ausführer mit Fehlermeldungen.

Den Befehl **Java compilieren** kann man verändern und anpassen. Dazu muss man eingeben:

Menü Konfiguration, Einstellungen,
+Extras (d.h. auf das Pluszeichen + vor **Extras** klicken), **Java kompilieren**.

Im Eingabefeld **Ausgangsordner:** sollte (im SWE-Labor und auf Ihrem Rechner) `$FileDir` stehen.

Im Eingabefeld **Parameter:** sollte (im SWE-Labor) `-d Z:\Klassen $File` stehen.

`$File` bezeichnet das aktive Dokument und `$FileDir` den Ordner, in dem es steht.

`$File` und `$FileDir` sind spezielle TextPad-Makros (die nichts mit den Makros im Menü **Makros** zu tun haben). Nähere Einzelheiten zu diesen speziellen Makros findet man in der TextPad-Hilfe im Abschnitt "Makros für Werkzeug-Parameter".

Im Eingabefeld **Regulärer Ausdruck für Sprungziele in Werkzeug-Ausgabe:** steht (das wird jetzt kaum überraschen) ein regulärer Ausdruck, z.B. `^(..[^:]+):(\d+):`. Der dient dazu, aus einer Fehlermeldung des Compilers `javac.exe` den Namen der fehlerhaften Datei und die Zeilen-Nr, in der der Fehler gefunden wurde, "heraus zu filtern". Dadurch wird folgendes möglich: Angenommen, wir haben eine Java-Datei `Hallo37.java` compiliert und im Fenster **Programmausgabe** (meistens unten, ziemlich breit, aber nicht hoch) sind Fehlermeldungen des Compilers aufgetaucht. Wenn wir jetzt einen *Links-Doppelklick* auf *die erste Zeile einer Fehlermeldung* ausführen, springt der Cursor automatisch in die betreffende Zeile der Datei `Hallo37.java`.

Merke: Der Datei-Name und die Zeilen-Nr., auf die eine Fehlermeldung sich bezieht, stehen nur in der *ersten Zeile* der Meldung. Deshalb muss man auf die *erste Zeile* doppelklicken, wenn "der Cursor springen soll". Ein Doppelklick auf eine andere Zeile einer Fehlermeldung hat keine Wirkung (auch wenn man ihn mehrmals und immer härter klickend wiederholt :-).

Nähere Einzelheiten zu den hier erwähnten *regulären Ausdrücken* findet man in der TextPad-Hilfe im Abschnitt "Reguläre Ausdrücke zum Interpretieren von Compiler-Fehlermeldungen".

5.3. Der Befehl 'Java-Programm starten'

Diesen Befehl aufrufen: **Menü Extras, Benutzer-Programme, Java-Programm starten**.

Oder mit Tastenkürzel: **Strg-2**

Dieser Befehl bewirkt Folgendes: Der TextPad nimmt den Namen des aktiven Dokuments, entfernt die Erweiterung und ruft mit dem Ergebnis den Java-Ausführer (genauer: den Java-Interpreter `java.exe`) auf. Der Java-Interpreter versucht dann, das betreffende Java-Programm zu interpretieren (d.h. auszuführen).

Beispiel-01: Name des aktiven Dokuments: `Hallo37.java`. Ohne Erweiterung: `Hallo37`. Aufruf des Java-Interpreters: `java Hallo37`

Daraufhin sucht der Interpreter (in allen Ordnern, die im `CLASSPATH` stehen) eine Datei namens `Hallo37.class`. Wenn er keine findet, gibt er eine Fehlermeldung aus. Sonst versucht er, die `.class`-Datei zu interpretieren (d.h. auszuführen).

Beispiel-01: Name des aktiven Dokuments: `Schneewittchen.txt`. Ohne Erweiterung: `Schneewittchen`. Aufruf des Java-Interpreters: `java Schneewittchen`

Daraufhin sucht der Interpreter (in allen Ordnern, die im `CLASSPATH` stehen) eine Datei namens `Schneewittchen.class`. Wenn er keine findet, gibt er eine Fehlermeldung aus. Sonst versucht er, die `.class`-Datei zu interpretieren (d.h. auszuführen).

Auch den Befehl **Java-Programm starten** kann man verändern und anpassen. Dazu muss man eingeben:

Menü Konfiguration, Einstellungen,
+Extras (d.h. auf das Pluszeichen + vor **Extras** klicken), **Java-Programm starten**.

Im Eingabefeld **Ausgangsordner:** sollte (im SWE-Labor und auf Ihrem Rechner) `$FileDir` stehen.

Im Eingabefeld **Parameter:** sollte (im SWE-Labor) `$BaseName` stehen.

Nähere Einzelheiten zu den speziellen Makros `$BaseName` und `$FileDir` findet man in der Text-Pad-Hilfe im Abschnitt "Makros für Werkzeug-Parameter".

Zu den Optionen ("mit/ohne einem Häkchen davor") für den Befehl Java-Programm starten:

1. Sehr empfehlenswert ist ein Häkchen vor **Dateien vor dem Start speichern**.

2. Ein Häkchen vor **Ausgabe Erfassen** hat Vor- und Nachteile:

Vorteil: Die Ausgaben (zur Standardausgabe) des ausgeführten Programms erscheinen nicht in einer **Eingabeaufforderung**, sondern im **Programmausgabe**-Fenster des TextPad und können dort direkt bearbeitet werden (z.B. mit der Maus und `Strg-C`).

Nachteil: Falls das ausgeführte Programm versucht, von der Standardeingabe zu *lesen*, wird ein Fehler auftreten (denn aus dem Programmausgabe-Fenster kann man nicht lesen).

5.4. Tastenkürzel des TextPad

Standardmäßig kennt der TextPad mehr als 100 Tastenkürzel (oder: Tastenkombinationen, engl. short-cuts). Ein vollständige Liste findet man in der TextPad-Hilfe im Abschnitt "Die Tastatur".

In den TextPad-Menüs steht hinter jedem Befehl das zugehörige Tastenkürzel (wenn es eines gibt). Somit kann man schrittweise (und so langsam oder schnell wie man möchte) von der Benutzung der Menüs übergehen zu den (schnelleren, professionelleren) Tastenkürzeln.

5.5. Tastenkürzel selbst einrichten

Anhand von zwei Beispielen wird beschrieben, wie man eigene Tastenkürzel einrichten kann. Die beiden konkreten Kürzel finde ich sehr nützlich. Aber noch wichtiger ist: Wenn man länger als eine Woche mit einem Editor arbeitet, sollte man lernen, wie man Tastenkürzel vereinbart.

Beispiel-01: Um die aktuelle Text-Zeile ("die, in der der Cursor gerade steht") schnell in die Ablage kopieren zu können (ohne einen Dreifach-Klick und `Strg-C`), wollen wir dafür das Tastenkürzel `Alt-C` einrichten. Das geht so:

Menü Konfiguration, Einstellungen, Tastatur.

Im Fensterchen **Kategorien:** wählen wir **Bearbeiten**.
Im Fensterchen **Befehle:** wählen wir **EditCopyLine**.
Dann setzen wir den Cursor in das Eingabefeld **Neue Tastenkombination** und geben ein `Alt-C`.
Ein Links-Klick auf **Zuweisen** (rechts oben), **Übernehmen**, **OK**.

Beispiel-02: Um die aktuelle Text-Zeile schnell löschen (und dabei in die Ablage kopieren) zu können, wollen wir dafür das Tastenkürzel `Alt-Y` einrichten. Das geht so:

Menü Konfiguration, Einstellungen, Tastatur.

Im Fensterchen **Kategorien:** wählen wir **Bearbeiten**.
Im Fensterchen **Befehle:** wählen wir **EditCutLine**.
Dann setzen wir den Cursor in das Eingabefeld **Neue Tastenkombination** und geben ein `Alt-Y`.
Ein Links-Klick auf **Zuweisen** (rechts oben), **Übernehmen**, **OK**.

5.6. In mehreren Dateien nach einer Zeichenkette suchen

Außer zum Entwickeln von Java-Programmen kann man den TextPad auch dazu benutzen, schnell in vielen Text-Dateien nach einer Zeichenkette zu suchen. Als Beispiel sei angenommen:
In einem Ordner namens `Z:\Java\BspJaSp` (und in seinen Unterordnern) stehen zahlreiche Java-Programme. Sie möchten wissen, in welchen davon der Befehl `continue;` vorkommt. Mit dem TextPad können Sie das etwa so herausfinden:

Menü Suchen, **In Dateien suchen...**

Ein **Suchen in Dateien**-Fenster geht auf. Geben Sie Folgendes ein:

In das Eingabefeld **Suchen nach:** `continue;`
In das Eingabefeld **In Dateien:** `*.java`
In das Eingabefeld **Im Ordner:** `Z:\Java\BspJa`

Zum Ordner, in dem gesucht werden soll, kann man auch hin navigieren nachdem man auf **Durchsuchen...** geklickt hat.

Empfehlung: Verwechseln Sie möglichst selten die beiden Knöpfe **Durchsuchen...** und **Suchen** (obwohl ihre Beschriftungen sich ähneln).

Unter **Details anzeigen** wähle ich praktisch immer **Nur Anzahl in Dateien** (damit ich pro Datei nur eine Suchergebniszeile bekomme, auch wenn eine Datei viele `continue;` - Befehle enthält).

Bringen Sie vor **Untergeordnete Ordner einbeziehen** ein Häkchen an.

Starten Sie die Suche durch einen Links-Klick auf **Suchen**.

Wie lange die Suche dauert hängt von vielen Einzelheiten ab (Hardware, Anzahl der Dateien, Größe der Dateien, Größe des Suchbegriffs, Anzahl der Unterordner etc.). Hier nur ein "grobes Beispiel" ohne Einzelheiten: Auf meinem PC durchsuche ich häufig eine Sammlung von ca. 2800 Java-Programmen. Das dauert ungefähr 20 Sekunden. Wenn ich dann (ohne den TextPad zwischendurch zu schließen) in derselben Sammlung noch mal (etwas anderes) suche, dauert das nur noch ca. 2 sec.

Das Ergebnis der Suche wird in einem **Suchergebnis-Fenster** angezeigt, je eine Zeile für jede gefundene Datei. Wenn man auf eine dieser Zeilen links-doppelklickt, wird die betreffende Datei geöffnet (sehr praktisch!).

5.7. In mehreren Dateien etwas ersetzen

Um in vielen Dateien z.B. `"Hallo"` durch `"Hellooo"` zu ersetzen, bietet die Firma Helios (Eigentümerin des TextPad) ein separates Programm "mit allen Schikanen" an (**WildEdit**, Preis einer Einzellizenz: ca. 10 Euro). Deshalb sollte man sich nicht wundern, dass der TextPad auf diesem Gebiet ein bisschen schwächelt. Es folgt ein Beispiel "für das Wenige", was er kann.

Angenommen, Sie haben 10 Text-Dateien (z.B. `.java`-Dateien). Um in all diesen Dateien `"Hallo"` durch `"Hellooo"` zu ersetzen, können Sie so vorgehen:

Sie öffnen all diese Dateien im TextPad (das ist der Schwachpunkt 1 dieses Verfahrens!).
Dann **Menü Suchen, Ersetzen** (oder kürzer: **F8**). Das **Ersetzen**-Fenster öffnet sich. Tragen Sie ein:

In das Eingabefeld **Suchen nach:** `Hallo`
In das Eingabefeld **Ersetzen Durch:** `Hellooo`

Unter **Bereich** wählen Sie (nicht **Aktives Dokument** oder **Textauswahl** sondern) **Alle Dokumente** (damit sind alle zur Zeit im TextPad geöffneten Dateien gemeint).

Diese **Bereich**s-Auswahl bewirkt, dass die Knöpfe **Nächstes suchen**, **Ersetzen** und **Nächstes ersetzen** bleich (d.h. deaktiviert) werden. Das ist der Schwachpunkt 2 dieses Verfahrens).

Holen Sie tief Luft und klicken Sie auf **Alle Ersetzen**. Ein Dialog-Fensterchen mit der Frage **"In allen Dokumenten ersetzen?"** öffnet sich. Holen Sie noch mal tief Luft und klicken Sie auf **OK**.

Wenn Ihnen jetzt einfallen sollte, dass Sie doch nicht alle Dokumente ändern wollten, dann haben Sie noch eine Chance: Die geänderten Dateien sind noch nicht abgespeichert. Sie können alle Dateien schließen (z.B. mit **Menü Datei, Alle schließen**) und dabei eine Speicherung ablehnen.

Ansonsten können Sie mit `Str-Umschalt-S` alle geänderten Dateien abspeichern.

Trotz der beiden Schwachpunkte benutze ich diesen TextPad-Befehl ab und zu.

5.8. Im Blockmodus Textblöcke löschen, kopieren und einfügen

Der TextPad kennt zwei Modi (oder: Betriebsarten): Den *Normalmodus* und den *Blockmodus*. Im Blockmodus kann man in einem Text *rechteckige Textblöcke* auswählen und dann *löschen* oder ausschneiden oder in die Ablage *kopieren* oder aus der Ablage an beliebige Stellen des Textes *einfügen*.

Anmerkung: Es ist üblich, Programmtexte in einer *nicht-proportionalen Schriftart* ("in einer Mono-Schrift" wie z.B. Courier) zu schreiben und zu bearbeiten. Deshalb ist es naheliegend, Programmtexte manchmal im Blockmodus zu bearbeiten. Alle guten Programm-Editoren haben einen Blockmodus.

5.8.1. Ein (hoffentlich) motivierendes Beispiel

Angenommen, Sie wollen in einem Java-Programm die Ergebnisse bestimmter Funktionsaufrufe "gut kommentiert" ausgeben, etwa mit Befehlen wie den folgenden:

```
1    printf("Math.sin(0.5 * Math.PI): %6.2f%n", Math.sin(0.5 * Math.PI));
2    printf("Math.cos(0.5 * Math.PI): %6.2f%n", Math.cos(0.5 * Math.PI));
3    printf("Math.sin(1.0 * Math.PI): %6.2f%n", Math.sin(1.0 * Math.PI));
4    printf("Math.cos(1.0 * Math.PI): %6.2f%n", Math.cos(1.0 * Math.PI));
5    printf("Math.sin(1.5 * Math.PI): %6.2f%n", Math.sin(1.5 * Math.PI));
6    printf("Math.cos(1.5 * Math.PI): %6.2f%n", Math.cos(1.5 * Math.PI));
```

In diesem Text wurden zwei rechteckige Blöcke **fett** hervorgehoben um leichter erkennbar zu machen, dass sie *genau übereinstimmen*. Wenn man diese Zeilen einfach so abtippt, muss man also an vielen Stellen zweimal genau das Gleiche tippen (und kann dabei natürlich Fehler machen).

Mit dem TextPad kann man statt dessen so vorgehen: Man tippt die 6 Zeilen ein, lässt aber auf jeder Zeile die Daten, die zum rechten Block gehören (und die man gerade schon mal eingegeben hat) weg. Wenn man damit fertig ist, kopiert man den linken Block an die Stelle des rechten Blocks. Das ist (nur) im Blockmodus möglich und geht (nach ein bisschen Übung) ziemlich einfach und schnell.

In diesem Beispiel hat das Arbeiten im Blockmodus noch einen weiteren Vorteil: Der linke Block besteht ganz aus Teilen von String-Literalen. Falls er Fehler enthält, könnte man die nur durch sehr sorgfältiges Korrekturlesen entdecken. Der rechte Block wird vom Compiler geprüft. Wenn der Fehler meldet, kann man die korrigieren und sollte dann den linken Block durch eine Kopie des rechten ersetzen.

5.8.2. Blockmodus-Grundlagen

In den *Blockmodus* versetzen kann man den TextPad auf zwei unterschiedliche Weisen:

1. Flüchtiger Blockmodus: Man drückt auf die Alt-Taste und hält sie gedrückt. Sobald man die Taste loslässt kehrt der TextPad aus dem *Blockmodus* wieder in den *Normalmodus* zurück.

2. Fester Blockmodus: Man gibt den Befehl **Menü Konfiguration**, **Blockauswahl-Modus** (oder **Strg-Q B**) ein. In diesem Fall kehrt der TextPad erst dann wieder in den Normalmodus zurück, wenn man den gleichen Befehl noch einmal eingibt.

Während der TextPad sich im Blockmodus befindet, kann man auf zwei unterschiedliche Weisen rechteckige Textblöcke *auswählen* (oder: *markieren*):

1. Auswählen mit der Maus: Bei gedrückter linker Maustaste den gewünschten Block von links oben nach rechts unten (oder umgekehrt von rechts unten nach links oben) "überstreichen".

2. Auswählen mit Tasten: Den Cursor links oben "vor das *erste* Zeichen des gewünschten Blocks" (oder umgekehrt: unten rechts "hinter das *letzte* Zeichen des gewünschten Blocks") positionieren. Spätestens jetzt den Blockmodus einschalten und mit den Cursortasten ←, ↑, → und ↓ den gewünschten Block auswählen.

5.8.3. Textblöcke bearbeiten: Ein paar Beispiele

Einen ausgewählten Block kann man ähnlich wie eine andere Auswahl bearbeiten: Man kann ihn

löschen (mit der **Entf**-Taste oder der **Back**-Taste)
ausschneiden (mit **Strg-X**)
kopieren (mit **Strg-C**)
einfügen (mit **Strg-V**) .

Die folgenden Beispiele sind ein Versuch, die Wirkung wichtiger Blockbefehle zu veranschaulichen.

Beispiel-01: Textblöcke löschen

```
++++++++++++++++++++        ++++++++++++++++++++        ++++++++++++++++++++
++AAA+++++++++++++++        ++AAA+++++++++++++++        ++++++++++++++++
++AAA++++BBBBBB+++++        ++AAA+++++++++              ++++++++++
++AAA++++BBBBBB+++++        ++AAA+++++++++              ++++++++++
++AAA++++BBBBBB+++++        ++AAA+++++++++              ++++++++++
+++++++++BBBBBB+++++        ++++++++++++++              ++++++++++++++
+++++++++BBBBBB+++++        ++++++++++++++              ++++++++++++++
```

Text 1: Mit A- und B-Block **Text 2**: Ohne den B-Block ... **Text 3**: ... und ohne A-Block

Der Text 1 besteht aus 7 gleich langen Zeilen, die zu einem großen Teil Pluszeichen enthalten. Nur zwei rechteckig Blöcke enthalten anstelle der Pluszeichen As bzw. Bs.

Wenn man den B-Block (auswählt und) *löscht*, erhält man den **Text 2**. Wenn man darin den A-Block (auswählt und) *löscht* erhält man **Text 3**. Der **Text 1** hat einen glatten rechten Rand. Der **Text 3** hat rechts einen sogenannten *Flatterrand*.

Beispiel-02: Textblöcke einfügen

```
++++++++++          ++++++++++          ++++++++++
++CC+++++++         +CCCCC++++++++      +CCCCC++++++++
++++++++++          ++CCC+++++++++      ++CCC+++++++++
+++++++DD++         ++CCC+++DD++++      ++CCC+++DDDD++++
++++++++++          ++++++++++          +++++++++DD++
++++++++++          ++++++++++          +++++++++DD++
++++++++++          ++++++++++          +++++++++DD++
```

Text 4: Ausgangstext **Text 5**: Mit einem C-Block **Text 6**: Mit C- und D-Block

Wenn man im **Text 4** den Cursor zwischen die beiden Ces positioniert und dann einen3x3-Block von Ces einfügt, erhät man **Text 5**. Wenn man darin den Cursor zwischen die beiden Des positioniert und einen 2x4-Block mit Des einfügt, erhält man **Text 6**. Der **Text 4** hat einen glatten rechten Rand. Der **Text 6** hat rechts einen Flatterrand.

Beispiel-03: Rechteckige und andere Blöcke

```
    +    E             +E                +    F         +F
   ++    E            ++E                ++   F         ++F
  +++    E           +++E                +++  F         +++F
 ++++    E          ++++E                ++++ F         ++++F
+++++    E         +++++E                +++++F         +++++F
```

Text 7 **Text 8** **Text 9** **Text 10** **Text 11** **Text 12**

Wenn man den **Text 7** vor den **Text 8** kopiert, erhält man **Text 9**. Offenbar ist **Text 7** ein rechteckiger Block, der oberhalb der Diagonalen Leerzeichen enthält. Das ist kaum überraschend.

Wenn man den **Text 10** vor den **Text 11** kopiert erhält man **Text 12**. Daraus kann man schließen: **Text 10** ist *kein rechteckiger* Block, der oberhalb der Diagonalen Leerzeichen enthält, sondern ein *dreieckiger* Block, der oben kürzer ist als unten. Das ist möglicherweise überraschend (weil die Markierung eines Blocks *immer rechteckig* aussieht, auch wenn der Block in Wirklichkeit dreieckig ist oder einen rechten Flatterrand hat).

Wenn man nur **Text 10** und **Text 11** sieht, kann man nicht mit Sicherheit voraussagen, wie **Text 12** aussehen wird. Denn man könnte den **Text 10** oberhalb der Diagonalen auch mit Leerzeichen zu einem Rechteck ergänzen oder "flattern lassen". Das würde sein Aussehen nicht ändern, wohl aber das von **Text 12** (dort würden die eFs dann untereinander stehen oder ebenfalls flattern, statt eine regelmäßige Treppe zu bilden).

Wenn das Ergebnis eines Block-Einfüge-Befehls nicht wie erwartet aussieht und einem falsch vorkommt, sollte man sich an dieses Beispiel erinnern oder es noch einmal ansehen. In solchen Fällen ist es meist auch hilfreich, die (normalerweise unsichtbaren) *Leerzeichen* und *Zeilenwechsel* sichtbar zu machen (z.B. indem man **Strg-Q I** eingibt oder auf den entsprechenden kleinen Knopf mit dem Paragraphen-Symbol ¶ darauf klickt).

Merke: Im Blockmodus kann man *rechteckige Textblöcke* bearbeiten, aber einige dieser Blöcke sind dreieckiger oder flatteriger als andere :-).

5.9. Suchen und Ersetzen mit regulären Ausdrücken

Sicherlich wissen Sie, dass man (beim Bearbeiten eines Textes mit dem TextPad) mit dem Befehl **Menü Bearbeiten, Suchen** (oder kürzer: **F5**) sehr schnell alle Vorkommen einer Zeichenkette finden kann und dass man mit dem Befehl **Menü Bearbeiten, Ersetzen** (oder kürzer: **F8**) sehr schnell z.B. alle Vorkommen der Zeichenkette `Hallo` durch die Zeichenkette `Hellooo` ersetzen kann.

Aber wissen Sie auch, wie man sehr schnell (nicht nur eine bestimmte sondern) alle Zahlen (Folgen der Ziffern 0 bis 9, z.B. `3` oder `123` oder `7263546`) finden kann? Oder wie man sehr schnell vor jede solche Zahl ein Pluszeichen oder dahinter ein Minuszeichen einfügen kann? Oder wie man aus jedem Namen der Form NACHNAME, VORNAME (z.B. `Müller, Fritz` oder `Özdemir, Ertan`) einen Namen der Form VORNAME NACHNAME (z.B. `Fritz Müller` oder `Ertan Özdemir`) machen kann? Was würden Sie machen, wenn Sie 100 Namen auf diese Weise umformen müssten? Oder 10 Tausend?

Mit sogenannten *regulären Ausdrücken* kann man diese Probleme ziemlich leicht lösen. Und wenn man sich mit regulären Ausdrücken vertraut gemacht hat (was, zugegeben, einige Anstrengung erfordert), dann merkt man plötzlich, dass die Welt der Informatik voller Probleme ist, die man mit solchen Ausdrücken elegant und schnell lösen kann, die aber ohne sie nur mühsam oder gar nicht lösbar sind.

Im Internet gibt es unüberschaubar viele Seiten zum Thema *Reguläre Ausdrücke*. Ein besonders umfangreiches Tutorium (auf Englisch) ist http://www.regular-expressions.info/. Dort gibt es auch eine Liste von Varianten (von regulären Ausdrücken in verschiedenen Programmiersprachen und anderen Umgebungen). In der TextPad-Hilfe gibt es einen Abschnitt "Regular Expression Syntax" (auch in der deutschen Hilfe auf Englisch).

Ein regulärer Ausdruck wird häufig auch als **RegEx** bezeichnet (Plural: **RegExen**).

Im Folgenden wird versucht, anhand von ein paar Beispielen anzudeuten, wie mächtig und nützlich RegExen beim Bearbeiten von Texten sein können.

5.9.1. Suchen mit regulären Ausdrücken

Beispiel-01: Sie haben einen sehr langen Text mit ein paar (ganzen) Zahlen darin. Die Zahlen liegen zum Teil weit auseinander (mit vielen Zeilen oder sogar mehreren Seiten ohne Zahlen dazwischen). Wie können Sie möglichst schnell all diese Zahlen noch mal ansehen und überprüfen?

Öffnen Sie das **Suchen nach**-Fenster mit **F5**.
Tragen Sie in das Eingabefeld **Suchen nach** den regulären Ausdruck `[0-9]+` ein.
Bringen Sie vor **Regulärer Ausdruck** ein Häkchen an.

Wenn Sie danach auf **Nächstes suchen** links-klicken, springt der Cursor im aktuellen Dokument zur nächsten (ganzen) Dezimalzahl (falls nötig, über viele Textseiten hinweg).

Anstelle von `[0-9]+` kann man auch einfacher `\d+` ("d" wie "digit") schreiben.

2-er-Zahlen (Binärzahlen) können Sie ganz ähnlich mit dem regulären Ausdruck `[01]+` suchen.

16-er-Zahlen (Hexadezimalzahlen), die keine Kleinbuchstaben enthalten, können Sie mit dem regulären Ausdruck `[0-9A-F]+` suchen.

Allerdings: Wenn in dem durchsuchten Text z.B. die Zeichenkette `ADAC` vorkommt, wird sie als 16-er-Zahl behandelt, obwohl Sie vielleicht als Akronym eines Automobil-Clubs gemeint ist.

Beispiel-02: Mit dem RegEx `(?<=Euro)\d+` findet man alle Ganzzahlen, *vor* denen das Wort `Euro` steht. Wichtig ist, dass das Wort `Euro` da sein muss, aber *nicht zu dem gefundenen String zählt*. Ein RegEx wie `(?<=Euro)` wird auch als **lookbehind** bezeichnet

Mit dem RegEx `Euro(?=\d)` findet man alle Vorkommen von `Euro` *nach* denen eine Ganzzahl steht. Hier muss die Ganzzahl (zumindest eine Ziffer) da sein, *gehört aber nicht zu dem gefundenen String*. Das ist wichtig, wenn man den gefunden String z.B. durch `Dollar` ersetzen will. Ein RegEx wie `Euro(?=\d)` wird auch als als **lookahead** bezeichnet.

Beispiel-03: Mit dem RegEx `(ta){4}` kann man nach `tatatata` suchen.
Mit dem Ausdruck `(ta){2, 4}` findet man `tata` oder `tatata` oder `tatatata` (mindestens 2 und höchstens 4 `ta`'s). Ohne die runden Klammern um das `ta` würden sich die Häufigkeitsangaben in den geschweiften Klammern nur auf das `a` beziehen.

5.9.2. Ersetzen mit regulären Ausdrücken

Im vorigen Abschnitt haben wir mit RegExen der Form `(?  )` gesucht. In diesem Abschnitt werden wir RegExen der Form `(  )` (*ohne* ein Fragezeichen `?` nach der öffnenden Klammer) als **Fanggruppen** (engl. capturing groups) verwenden. Diese Fanggruppen werden automatisch (mit `1` beginnend) nummeriert und das, was die `n`-te Fanggruppe "fängt" wird in eine Variable namens `\n` geschrieben. Diese Variablen `\1`, `\2`, `\3`, ... darf man dann im **Ersetzen durch**-Ausdruck verwenden.

Beispiel-01: Angenommen, wir haben eine Liste mit Namen der Form NACHNAME, VORNAME (z.B. `Müller, Fritz` oder `Özdemir, Ertan` etc.) und wollen diese Namen in die Form VORNAME NACHNAME (z.B. `Fritz Müller` oder `Ertan Özdemir` etc.) umwandeln. Dann können wir etwa folgende RegExen eingeben:

Im Eingabefeld **Suchen nach:** `(\p{L}+), (\p{L}+)`
Im Eingabefeld **Ersetzen durch:** `\2 \1`

Der RegEx `\p{L}` bedeutet: A character with the **p**roperty to be a **L**etter (auch ü und Ö etc. haben diese property).

Der RegEx im **Suchen nach**-Feld besteht aus 2 Fanggruppen, mit einem Komma und einem Blank dazwischen. Die Fanggruppe 1 fängt den Nachnamen (z.B. `Müller`) und die Fanggruppe 2 den Vornamen (z.B. `Fritz`). Der RegEx im **Ersetzen durch**-Feld bezeichnet den Vornamen gefolgt von einem Blank und dem Nachnamen.

Beispiel-02: Angenommen, wir haben eine Liste von Matrikel-Nrn, jede auf einer neuen Zeile, z.B.

```
s123456
s234567
s345678
s456789
s567890
s678901
s789012
...
```

Um Zeilen zu sparen wollen wir die Liste so umformen, dass je 3 Nrn auf einer Zeile stehen, etwa so:

```
s123456 s234567 s345678
s456789 s567890 s678901
s789012 ...
```

Das können wir mit den folgenden RegExen erreichen:

Im Eingabefeld **Suchen nach:** `^(.*)\n(.*)\n(.*)$`
Im Eingabefeld **Ersetzen durch:** `\1 \2 \3`

Die Zeichen `^` und `$` bezeichnen den Anfang bzw. das Ende einer Zeile. `\n` bezeichnet einen Zeilenwechsel (unabhängig davon, ob der aus einem CR-Zeichen, aus einem LF-Zeichen oder aus den zwei Zeichen CR, LF besteht).

In diesem Beispiel war es nicht nötig, beim Suchen genau zu beschreiben, dass eine Matrikel-Nr aus einem `s` und 6 Dezimalziffern bestehen muss.

Anmerkung: Dieses Beispiel stammt aus dem Alltag eines Dozenten der Beuth Hochschule :-).

Beispiel-03: Angenommen, wir haben eine Textdatei erstellt und wollen jetzt (nachträglich) am Anfang jeder Zeile eine Zeilen-Nr anbringen. Das können wir z.B. so erreichen:

Im Eingabefeld **Suchen nach:** `^`
Im Eingabefeld **Ersetzen durch:** `\i`

`\i` ist der Name einer ganz speziellen TextPad-Variablen. Sie funktioniert nur dann richtig, wenn wir die Ersetzung durch einen Links-Klick auf **Alle ersetzen** vornehmen (mehrere Klicks auf **Nächstes ersetzen** fügen vor jeder betroffenen Zeile eine 1 ein, was als Nummerierung unbefriedigend ist).

Es empfiehlt sich, nach dem `\i` noch mindestens ein weiteres Zeichen als Trenner zwischen Zeilen-Nr und Text anzugeben, z.B. ein Blank, oder einen Doppelpunkt `:` und ein Blank oder ... .

Im hier beschriebenen einfachsten Fall beginnt die Nummerierung bei 1 und der Abstand zwischen zwei Nummern ist +1. Man kann aber auch bei anderen Zahlen beginnen und andere Abstände (positive und negative) angeben. Wie das geht, steht in der TestPad-Hilfe im Abschnitt "Sequenznummern einfügen".

Anmerkung: Die Variable `\i` ist wohl sehr TextPad-spezifisch. Man sollte nicht erwarten, dass sie auch in allen anderen RegEx-Systemen vorhanden ist.

5.9.3. Reguläre Ausdrücke vernünftig editieren

Selbst auf einem 30-Zoll-Plasmabildschirm sind die Fenster der Befehle **Suchen nach...** und **Ersetzen...** relativ klein und die Eingabefelder **Suchen nach:** und **Ersetzen durch:** in diesen Fenstern sind noch kleiner. Statt einen regulären Ausdruck direkt in einem dieser engen Eingabefelder zu editieren, kann man auch ein neues Dokument anlegen (z.B. mit dem Tastenkürzel **Strg-N**), den regulären Ausdruck in einer Zeile dieses Dokuments editieren und den Ausdruck erst dann in das entsprechende Eingabefeld kopieren (mit **Strg-C** und **Strg-V**).

Reguläre Ausdrücke, die man später wiederverwenden will, kann man auch z.B. in eine `.txt`-Datei schreiben. Und wenn man kleine Schriften nur mit Mühe lesen kann, sollte man `.txt`-Dateien mit einem etwas größeren Font anzeigen lassen, etwa so:

Menü Konfiguration, **Einstellungen**, **+Dokumentenklassen** (d.h. auf das Pluszeichen + vor **Dokumentenklasse** klicken), **+Text**, **Schriftart**
Unter **Schriftgrad** eine geeignete Schriftgröße auswählen.

5.10. Eigene Textbausteine schreiben

In die Datei `java.tcl` (siehe oben den Abschnitt **2.4. Ordner für TextPad Textbausteine und Makros**) können Sie eigene Textbausteine einfügen oder Sie können eine weitere, ähnliche Datei (z.B. eine namens `meinJava.tcl`) erstellen und im Ordner `Samples` ablegen. Den Namen der Datei können Sie frei wählen, aber die Erweiterung `.tcl` muss sein, damit der TextPad die Datei als *TextPad Clip Library* erkennt.

Eine Java-Baustein-Sammlung sollte mit Zeilen ähnlich den folgenden (aber ohne die Zeilen-Nrn) beginnen:

```
1 !TCL=123, von Serkan Müller, April 2016
2 !TITLE=Java
3 !SORT=N
4 !CHARSET=ANSI
```

Die Zahl hinter `!TCL` kann ziemlich frei gewählt werden, sollte aber zwischen 1 und 999 liegen und *eindeutig* sein (d.h. alle gleichzeitig benutzen Sammlungen sollten *unterschiedliche* Nummern haben). Hinter `!TITLE=` gibt man den *Namen* der Baustein-Sammlung an. Diesen Namen kann man später im **Textbausteine**-Fenster wählen (siehe Abschnitt **2.5. Ein kleines Java-Programm zusammen klicken**). Weitere Erläuterungen zu diesen Zeilen findet man in der TextPad-Hilfe im Abschnitt **Textbausteine und Sammlungen direkt bearbeiten**.

Ein typischer Textbaustein in einer Sammlung sieht z.B. so aus:

```
5 !TEXT=stat-pub-XXX-Methode
6    static public XXX (PP) {
7    } //
8
9 !
```

Die erste Zeile muss mit `!TEXT=` beginnen. Dahinter gibt man einen *Namen* für den Baustein an. Auf diesen Namen kann der Benutzer später doppelklicken, um den Baustein in sein Dokument einzufügen. Die letzte Zeile darf nur ein Ausrufezeichen `!` enthalten (wie hier in Zeile 9). Die Zeilen dazwischen (im Beispiel: Zeile 6 bis 8) enthalten den *Text* des Textbausteins.

Beachten Sie, dass der Text um drei Zeichen eingerückt ist, denn der TextPad kopiert ihn "zeichengetreu" ohne seine Einrückung zu verändern.

Die (leere) Zeile 8 bewirkt, dass nach einem Einfügen des Bausteins der Cursor *unter* dem eingefügten Text steht, und nicht *am Ende der letzten Zeile*. Wenn Sie das unbequem finden, können Sie die leere Zeile 8 auch weglassen.

Es folgt ein Textbaustein für Fortgeschrittene (Baustein-*Schreiber* und -*Anwender*):

```
10 !TEXT=stat-pub-void-Methode^
11    static public void \^ (PP) {
12    } //
13 !
```

Der Name dieses Bausteins (`stat-pub-void-Methode^`) endet mit dem Zeichen `^` (einem Zirkumflex). Damit wird dem Anwender signalisiert, dass er ein geeignetes Wort mit der Maus auswählen (markieren) sollte, bevor er den Baustein in sein Dokument einfügt. Die Zeichen `\^` im Text des Bausteins (siehe Zeile 11) werden dann durch das ausgewählte Wort ersetzt.

Anmerkung: Leider wird nur das *erste Vorkommen* von \^ ersetzt. Besser wäre es, wenn *alle* Vorkommen von \^ ersetzt würden. Auch der TextPad ist noch nicht perfekt :-).

Empfehlung: Wenn man die gleichen Zeilen (oder sehr ähnliche Zeilen) mehr als 3 Mal eingetippt hat, sollte man erwägen, aus diesen Zeilen einen Textbaustein zu machen.

Zitat (habe leider vergessen, von wem): Warum sollte man ein paar Zeilen in wenigen Sekunden eintippen, wenn man mehrere aufregende Stunden damit verbringen kann, einen guten Textbaustein daraus zu machen?