Tipps zum TextPad

Inhaltsverzeichnis

In diesem Papier wird vorausgesetzt, dass Sie auf einem PC unter Windows den **Java Development Kit** (JDK) und den Editor **TextPad** bereits installiert haben (wie im Papier **JdkUndTextPad.pdf** beschrieben). Die folgenden Tipps sollen Sie dabei unterstützen, den TextPad so einzustellen, dass Sie damit noch bequemer Java-Programme entwickeln können.

Notation: In diesem Papier bedeutet z.B.
Dokumentenklasse
dass Sie auf das Wort Dokumentenklasse klicken sollen. Dagegen bedeutet
+ Dokumentenklasse
dass Sie auf das Pluszeichen + vor Dokumentenklasse klicken sollen.

Tipp 1: Zeilen-Nummern auf dem Bildschirm

Starten Sie den TextPad. Dann:
Konfiguration, Einstellungen, Ansicht.
Dann ganz unten neben **Zeilennummern** ein Häkchen anbringen.

Danach zeigt der TextPad alle Dateien auf dem Bildschirm mit Zeilennummern an.

Tipp 2: Zeilen-Nummern beim Ausdrucken

Starten Sie den TextPad. Dann:
Konfiguration, Einstellungen, + Dokumentenklasse, + Java, Drucken, Zeilennummern

Wenn man danach eine `.java`-Datei mit **Datei, Drucken...** (oder mit **Strg+P**) ausdruckt, werden auch die Zeilen-Nummern ausgedruckt.

Tipp 3: Die Hammer-Leiste sichtbar machen

Starten Sie den TextPad. Dann:
Ansicht, Symbolleisten.
Bringen Sie neben **Extras** ein Häkchen an.

Danach zeigt der TextPad eine zusätzliche Symbolleiste an, die zahlreiche Hammer-Symbole enthält. Durch einen Klick auf einen dieses Hämmer kann man das entsprechende Kommando aus dem Menü **Extras, Benutzerprogramme** aufrufen:
Hammer 1: Java kompilieren
Hammer 2: Java-Programm starten
Hammer 3: Java-Applet starten

Tipp 4: Eine .java-Datei dem Ausführer übergeben (kompilieren)

Öffnen Sie eine .java-Datei, z.B. die Datei `Z:\BspSaSp\Hallo01.java` mit dem TextPad.
Möglichkeit 1: **Extras, Benutzerprogramme, Java kompilieren**
Möglichkeit 2: Auf den **Hammer 1** klicken
Möglichkeit 3: **Strg+1** eingeben

Daraufhin wird der TextPad den Java-Compiler `javac.exe` auffordern, die Datei `Hallo01.java` zu kompilieren. Der Compiler wird prüfen, ob die Datei formale Fehler ("Java-Rechtschreibungsfehler") enthält. Wenn ja, dann gibt er entsprechende Fehlermeldungen aus. Sonst (wenn die Datei ein formal korrektes Java-Programm enthält) wird der Compiler eine Datei namens `Hallo01.class` erzeugen.

Tipp 5: Ein Java-Programm starten

Ein Java-Programm besteht aus einer *Hauptklasse* (die eine `main`-Methode enthalten muss) und beliebig vielen *Nebenklassen*. (die main-Methoden enthalten dürfen, aber nicht müssen). Das Programm heißt so wie seine Hauptklasse.

Um ein Programm namens `Hallo01` zu starten, sollten Sie mit dem TextPad irgendeine Datei mit dem Namen `Hallo01` und irgendeiner Erweiterung betrachten (z.B. `Hallo01.java` oder `Hallo01.txt` oder `Hallo01.tmp` oder `Hallo01` etc.)
Möglichkeit 1: **Extras, Benutzerprogramme, Java-Programm starten**
Möglichkeit 2: Auf den **Hammer 2** klicken
Möglichkeit 3: **Strg+2** eingeben

Wenn man so z.B. das Programm `Hallo01` startet, wird die `main`-Methode der Hauptklasse `Hallo01.class` ausgeführt.

Tipp 6: Die Einrücktiefe für Java-Quelldateien festlegen

Alle Java-Quelldateien (`.java`-Dateien) sollten sauber eingerückt sein. Alle Beispielprogramme und andere vorgegebene `.java`-Dateien sind mit einer Einrücktiefe von 3 Zeichen (pro Einrückstufe) formatiert. Deshalb sollten Sie Ihren TextPad ebenfalls so einstellen, dass der eine Einrücktiefe von 3 Zeichen unterstützt. Das geht so:

Konfiguration, Einstellungen, + Dokumentenklasse, + 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 in Häkchen anbringen. **Übernehmen, OK**.

Die Häkchen bewirken, dass Ihre fertigen Java-Quelldateien keine Tab-Zeichen enthalten, sondern nur entsprechend viele Leerzeichen (Blanks, Spaces). Das hat den Vorteil, dass auch andere Editoren Ihre Java-Quelldateien richtig eingerückt anzeigen werden.

Tipp 7: .java-Dateien automatisch mit dem TextPad öffnen

Rechtsklick auf den Windows-**Startknopf**, **Windows-Explorer öffnen**.
Navigieren Sie mit dem Explorer zu irgendeiner `.java`-Datei, z.B. zur Datei
`Z:\BspJaSp\Abbildungen01.java`.
Linksdoppelklick auf den Dateinamen `Abbildungen01.java`.
Ein Windows Fenster öffnet sich. Wählen Sie darin
`Programm aus einer Liste installierter Programme auwählen`
OK.
Ein Fenster mit installierten Programmen öffnet sich. Wählen Sie darin den TextPad.

Ab jetzt sollte ein Linksdoppelklick auf eine `.java`-Datei diese Datei im TextPad öffnen.

Tipp 8: Nur eine Instanz des TextPad zulassen

Wenn Sie den vorigen Tipp realisiert haben, und nacheinander auf zwei verschiedene `.java`-Dateien einen Linksdoppelklick ausführen (z.B. auf `EM.java` und auf `EMTst.java`), werden *zwei* Instanzen des TextPad gestartet, eine mit `EM.java` und eine mit `EMTst.java`. Häufig wäre es nützlicher, wenn alle angeklickten `.java`-Dateien in ein-und-derselben Instanz des TextPad geöffnet würden. Dieses Verhalten des TextPads können Sie wie folgt einstellen:

Starten Sie den TextPad. Dann:
Konfiguration, Einstellungen, Allgemein.
Entfernen Sie das Häkchen neben **Mehrere Instanzen zulassen**.

Tipp 9: Den Partner einer Klammer finden

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 tief geschachtelt vorkommen können.

Tipp 10: Eigene Tastenkombinationen festlegen

Für viele Befehle an den TextPad sind bereits Tastenkombinationen festgelegt. Und den anderen Befehlen kann der Benutzer eigene Tastenkombinationen zuordnen, wenn ihm das sinnvoll erscheint.

Als Beispiel soll dem Befehle **Lösche die aktuelle Zeile** (das ist die Zeile, in der sich der Cursor gerade befindet) die Tastenkombination **Alt+Y** zugeordnet werden:

Konfiguration, Einstellungen, Tastatur.
Unter **Kategorien:** wählen wir **Bearbeiten**.
Unter **Befehle:** wählen wir **EditDelLine**.
In das Textfeld unter **Neue Tastenkombination:** geben wir **Alt+Y** ein
(d.h. wir halten die Alt-Taste gedrückt während wir zusätzlich auf die Y-Taste drücken)
und klicken auf **Zuweisen**, **Übernehmen**, **OK**.

Diese Tastenkombinationen Alt-Y ist besonders gut dazu geeignet, mehrere untereinander stehende Zeilen zu löschen: Man bringt den Cursor in die oberste der Zeilen, hält dann die Alt-Taste gedrückt, während man mehrmals auf die Y-Taste drückt. Probieren Sie es gleich mal aus.

Eine Liste aller Tastenkombinationen, die der TextPad standardmäßig kennt finden Sie so:

Hilfe, Hilfethemen, Index.
Als **Zu suchendes Schlüsselwort** das Wart **Tastatur** eingeben.
In der Liste der gefundenen Stellen auf **Tastatur** klicken. Daraufhin werden mehrere Alternativen angeboten. Wählen Sie **Die Tastatur**.

Tipp 11: Zwei Makros und eine TCL einrichten

Mit dem TextPad-Makro **C-Comment** kann man beliebig viele Zeilen zu `//`-Kommentaren machen.
Mit dem Makro **C-Uncomment** kann man die Kommentarzeichen `//` wieder entfernen.

Die TextPad Clip Library (TCL) `java.tcl` enthält eine Reihe von Bausteinen (engl. clips), die man in einem Java-Programm häufig einsetzen kann, z.B. Kommentarzeilen wie

```
// ------------------------------------------------------------------
```

oder

```
/* ------------------------------------------------------------------
------------------------------------------------------------------ */
```

oder Vereinbarungen von Methoden wie `p`, `pln` und `printf` etc.

Um diese Makros und Java-Bausteine **auf einem Rechner im SWE-Labor** einzurichten, müssen Sie folgende Schritte ausführen:

1. Laden Sie das Archiv http://public.beuth-hochschule.de/~grude/FuerTextPad.zip in Ihren Bereich.

2. Öffnen Sie das Archiv mit dem Windows-Explorer und kopieren Sie alle darin enthaltenen Dateien (`C-Comment.tpm, C-Uncomment.tpm, ..., java.tcl`) in den Ordner

`C:\Users\IHR-NAME\AppData\Roaming\Helios\TextPad\7`

Anstelle von `IHR-NAME` sollen Sie natürlich ihren Namen angeben, mit dem Sie sich im SWE-Labor einloggen, z.B. `s123456`.

3. Damit die beiden Makros im Menü **Makros** des TextPad erscheinen, ist noch Folgendes nötig

Konfiguration, Einstellungen, Makros.

Unter **Verfügbare Makros** sollten unter anderen die Makros **C-Comment** und **C-Uncomment** aufgelistet sein. Wählen Sie die beiden Makros und klicken Sie auf **Hinzufügen**>>. Danach sollten die Makros unter **Makros im Menü** stehen. **Übernehmen**, **OK**.

Danach sollten die beiden Makros im Menü **Makros** des TextPad angezeigt werden.

Um die Makros auszuprobieren: Öffnen (oder erstellen) Sie eine kleine Textdatei.

Wählen Sie ein paar Zeilen mit der Maus und dann **Makros, C-Comment**. Alle ausgewählten Zeilen sollten jetzt auskommentiert sein, d.h. mit `//` beginnen.

Wählen Sie ein paar der auskommentierten Zeilen mit der Maus und dann **Makros, C-Uncomment**. Die Kommentarstriche `//` am Anfang der ausgewählten Zeilen sollten wieder verschwunden sein.

4. Und so können Sie auf die Bausteine in der Datei `java.tcl` zugreifen:

4.1. Öffnen Sie mit dem TextPad eine neue (leere) Datei.
4.2. Verbreitern Sie das linke, senkrechte Fenster des TextPads, so dass Sie die Texte auf den drei Reitern an der Unterkante des Fensters lesen können.
4.3. Klicken Sie auf den Reiter **Textbausteine** (vielleicht sehen Sie auch nur **Textb...** oder nur **Te...**).
4.4. Jetzt ein Linksklick ganz oben in die Textzeile unter der Überschrift **Textbausteine** (in dieser Textzeile kann z.B. **ANSI-Characters** oder **HTML-Tags** oder etwas anderes stehen). Durch Ihren Linksklick öffnet sich ein Menü. Wählen Sie darin **Java**. Jetzt sollten in dem senkrechten Fenster die Namen der verfügbaren Java-Bausteine sichtbar sein (**Mehrzeilenkommentar**, **Minuszeile**, **Pluszeile**, ...).
4.5. Doppelklicken Sie auf den Baustein-Namen **Klasse (mit main, pln)**.
In der anfangs geöffneten leeren Datei sollte jetzt eine Java-Klassenvereinbarung erscheinen. Die Klasse heißt (vorläufig) `XXX`. Sie enthält eine `main`-Methode und die Vereinbarung einer Methode namens `pln` (als Abkürzung für `System.out.println`). Sie brauchen bloß noch alle `XXX` durch einen besseren Namen (z.B. `Hallo15`) zu ersetzen. Drücken Sie dazu auf die Taste F8.

Die anderen Bausteine funktionieren ähnlich.

Auf Ihrem eigenen Rechner:

Wenn Sie den TextPad auf Ihrem Rechner z.B. in den Ordner `C:\Programme\TextPad7\` installiert haben, dann sollten Sie die Dateien aus dem Archiv `FuerTextPad.zip` in den Ordner `C:\Programme\TextPad7\Samples` kopieren und dann die oben beschriebenen Schritte entsprechend ausführen.

Tipp 12: Im Blockmodus Textblöcke bearbeiten

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*.

Ein (hoffentlich) motivierendes Beispiel

Angenommen, Sie wollen in einem Java-Programm (z.B. in einem Testprogramm) 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, würde man also an vielen Stellen zweimal das Gleiche tippen.

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.

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 **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.

Textblöcke bearbeiten: Ein paar Beispiele

Einen ausgewählten Block kann man genau 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 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, erhält 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 ma-

chen (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 :-).