Struktogrammgenerator

Peter Lenser
Peter.Lenser@gmail.com

Stand: 19. Februar 2013

Das Programm dient dazu Methoden, die in Java geschrieben wurden, so zu übersetzen, dass mit Hilfe des LaTeX- package struktex von Jobst Hoffmann eine Darstellung des Codes in Struktogrammform nach Nassi-Shneiderman erfolgt. Die Übersetzung kann z.B. mit ©Miktex erfolgen.
Ziel ist es, Methoden in Struktogramme zu wandeln um diese als Folie für einen Beamer zu verwenden oder in anderen Latexfiles einzubinden.
Das Programm wurde mit Eclipse 3.n und Eclipse Juno mit dem Betriebssystem ©Windows 7 programmiert und getestet. Die Methoden sollten mit Eclipse formatiert werden. (Standardeinstellung). Andere Einstellungen und frei formatierter Code führen möglicherweise zu Fehlern. Eclipse benützt mit Windows das Text file encoding Cp1252 zum Speichern einer Datei, mit Linux UTF 8. Wenn das Programm mit Linux benützt wird, müssen im Quellcode Korrekturen an den Umlauten vorgenommen werden, da das Programm aus Variablennamen Umlaute automatisch entfernt. Ich empfehle immer ohne Umlaute, auch in den Kommentaren, zu programmieren.

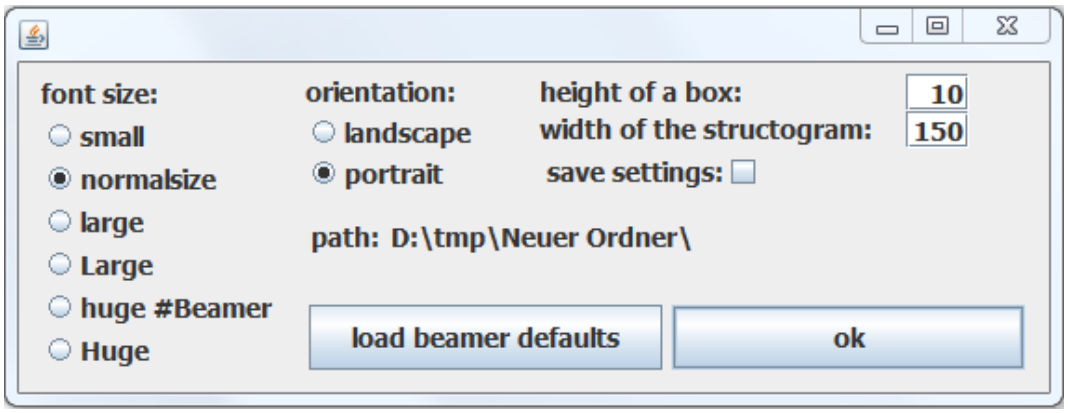
Bedienung:

Kopieren Sie eine Methode aus dem Quellcode in Eclipse und speichern Sie diese Methode mit einem Texteditor. Beispiel:

```
private String until_semicolon(StringBuilder sB) {
    int i = 0;
    int length = sB.length();
    String word;
    while ((i < length) && (sB.charAt(i) != ';')) {
        i++;
    }
    word = format_text(new StringBuilder(sB.substring(0, i)));
    sB = sB.delete(0, i + 1);
    return word;
}
```

Wird gespeichert unter until_semicolon.txt. Nur die Endung txt ist relevant.

Das Programm startet mit einer kleinen Benutzeroberfläche:



Hier kann man die Einstellungen anpassen und die Anpassungen gegebenenfalls speichern. Für dieses Beispiel wurde Hochformat und Breite 150 gewählt. Der ok – Button öffnet ein Fenster um eine Datei zu öffnen. Nach Umwandlung wird ein Fenster geöffnet um die erzeugte Datei zu speichern. Die erzeugte Datei heißt in diesem Beispiel until_semicolon.tex und kann sofort mit TeXworks aus Miktex geöffnet werden. Hier der erzeugte Dateiinhalt:

```
1  \documentclass[a4paper,11pt]{letter}
2  \usepackage[utf8]{inputenc}
3  \usepackage[T1]{fontenc}
4  \usepackage{anysize}
5  \usepackage{lmodern}
6  \usepackage{german}
7  \marginsize{1 cm}{1 cm}{0 cm}{2 cm}
8  \usepackage[pict2e,german]{struktex}
9  \pFonts{\large\sffamily}{\large\sffamily\bfseries}{\large\sffamily\slshape}
10 \begin{document}
11 \normalsize
12 \begin{struktogramm}(150,80)
13     [private\ String\ until\_semicolon(StringBuilder\ sB)]
14 \assign[10]{\(int\ \ i\ \gets\ 0\)}
15 \assign[10]{\(int\ \ length\ \gets\ sB.length()\)}
16 \assign[10]{\(String\ \ word\)}
17 \while[10]{\(while\ \((i\ <\ length)\ \&\&\ (sB.charAt(i)\ !=\ ';'))\)}
18 \assign[10]{\(i++\ \)}
19 \whileend
20 \assign[10]{\(word\ \ \gets
21     format\_text(new\ StringBuilder(sB.substring(0,\ i)))\)}
22 \assign[10]{\(sB\ \ \gets\ sB.delete(0,\ i\ +\ 1)\)}
23 \assign[10]{\(\gets\ \\ word\)}
24 \end{struktogramm}\\
25 \end{document}
```

Die erste Zeile wird nach der Auswahl der Orientierung generiert. Der Inhalt der Zeilen 2 bis 9 sind in der Datei firstlines.txt gespeichert und können dort angepasst werden.

Ergebnis:

private String until_semicolon(StringBuilder sB)

$int \;\; i \;\; \leftarrow 0$
$int \;\; length \;\; \leftarrow sB.length()$
$String \;\; word$
$while \; ((i \; < \; length) \; \&\& \; (sB.charAt(i) \; != \; ';'))$
$i++$
$word \;\; \leftarrow format\_text(new \; StringBuilder(sB.substring(0, \; i)))$
$sB \;\; \leftarrow sB.delete(0, \; i \; + \; 1)$
$\leftarrow \qquad\qquad word$

Anmerkungen, Probleme

Das Programm wandelt das switch – statement in Java in das case – Konstrukt nach Nassi – Shneiderman um. Dies ist (leider) nicht korrekt. Korrekt wäre das switch – statement, wenn am Ende jedes Falls kein break steht, in eine Folge von Verzweigungen zu übersetzen. Das Programm lässt bei switch den im Struktogramm unnötigen Befehl break im Struktogramm trotzdem erscheinen. Für meine Anwendungen in der Lehre ist diese Lösung pragmatisch und richtig.
Was das Programm nicht kann:

- Lange Zeilen automatisch umbrechen und die Höhe der Box verändern
- Variablendeklaration besser setzen oder entfernen
- Lange Programme auf mehrere Seiten verteilen
- Parameter setzen und beschreiben
- Die Struktogrammhöhe bei geschachtelten Struktogrammen richtig berechnen
- Und vieles mehr ...

Da die meisten Details leicht aus dem tex File selbst angepasst werden können und in der Regel individuell gemacht werden, plane ich hier auch nicht nachzupflegen.

Lizenzvereinbarung

Peter Lenser
Göppingen, Germany
E-Mail: plenser (at) gmail.com