ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert

19. Juni 2018
Blatt 7

Digitale Typografie (SS 2018)

Abgabe bis zum 26. Juni 2018, 10:00 Uhr

Lernziele:

- Umgang mit den Datenstrukturen der Knuth'schen Schachteln

Aufgabe 9: Generierung einer Tabelle

Im Rahmen dieser Aufgabe soll ein Java-Programm entwickelt werden, mit Hilfe dessen aus einer komma-separierten Eingabe eine Tabelle in einem PostScript-Dokument generiert wird. Verwenden Sie als Grundlage die auf der Vorlesungsseite bereitgestellten typografischen Java-Klassen.

Bei der Umsetzung sind folgende Punkte zu beachten:

- Die Tabelleninhalte sind aus der Standardeingabe einzulesen. Die Ausgabe des PostScript-Texts erfolgt über die Standardausgabe.
- Als Eingabeparameter soll ein Schriftschnitt, die zu verwendente Schriftgröße, die Zeilenhöhe sowie die maximale Breite der Tabelle in Punkten übergeben werden.
- Eine mögliche Aufrufdefinition wäre also beispielsweise

  ```
  TableGenerator font.afm size baselineskip tabwidth alignment
  ```

  Ein beispielhafter Programmaufruf für eine CSV-Datei mit fünf Spalten sähe dann wie folgt aus:

  ```
  TableGenerator ptmr8a.afm 14 17 450 LLRZR <test.csv >out.ps
  ```

  (Hinweis: Vergessen Sie bitte nicht, *baselineskip* und *tabwidth* an passender Stelle in Ihrem Programm mit 1000 zu multiplizieren, so dass dies mit den Einheiten der Vorlesungsbibliothek übereinstimmt.)

- Sie dürfen voraussetzen, dass alle Spalten die gleiche Breite zugewiesen bekommen. Ein zusätzlicher Eingabeparameter zur Definition der einzelnen Spaltenbreiten ist somit nicht notwendig.

Hinweise:

Es steht Ihnen frei, welche konkreten Klassen bzw. Datenstrukturen Sie für die Knuth'schen Schachteln verwenden. Sie können aber gerne die in der Vorlesung vorgestellten Klassen benutzen. Auf der Vorlesungsseite finden Sie die Vorlesungsbibliothek zusammen mit einer Javadoc-basierten Dokumentation, einem JAR aller Bibliotheksklassen und einem tar.gz-Archiv mit den Quellen. (Außer den Java-Standardklassen werden keine weiteren Bibliotheken benötigt.)

Die Vorlesungsbibliothek unterstützt bislang nur Schriftschnitte, für die eine Metrik im AFM-Format vorliegt.[1] Ähnlich wie bei der FOP-Bibliothek gibt es eine *FontMetrics*-Schnittstelle. AFM-Dateien können Sie mit Hilfe der *AdobeFontMetrics*-Klasse eröffnen:[2]

```
FontMetrics fm = new AdobeFontMetrics(args[0]);
```

Eine Folge zusammenhängender Zeichen in der Eingabe (hier konkret ein einzelnes durch Kommata getrenntes Feld) kann in der Vorlesungsbibliothek am elegantesten als *HorizontalSequence* unter Verwendung der Klasse *Sequencer* erstellt werden. Der *Sequencer* unterstützt auch vollautomatisch das Kerning. Wenn eine Sequenz abgeschlossen ist, kann sie direkt zur Konstruktion einer *HorizontalBox* verwendet werden. Schematisch könnte das dann so aussehen:

```
InputStream in = new BufferedInputStream(System.in);
HorizontalSequence hseq = new SimpleHorizontalSequence();
Sequencer s = new Sequencer(hseq, fm, fontSize);
int ch;
while ((ch = in.read()) >= 0 && /* ... */) {
    s.add(ch);
}
s.finish();
HorizontalBox hbox = new HorizontalBox(hseq);
```

Wenn Sie die Einträge für eine Spalte an die jeweilige Breite anpassen möchten, können Sie das tun, indem Sie passende Leerzeichen einfügen, die bis zur gewünschten Spaltenbreite auffüllen. Alternativ können Sie auch mit unendlich dehnbaren Fugen arbeiten (*Glue* mit *Item.INFINITY* bei *stretchability*) und dann die einen Spalteneintrag repräsentierende *HorizontalSequence* mit dem *HorizontalFitter* auf die gewünschte Spaltenbreite ziehen. Wenn auszudehnen ist, sieht der *HorizontalFitter* zunächst nach, ob unendlich dehnbare Fugen dabei

[1] Es ist ohnehin nur sinnvoll, die Schriftschnitte zu verwenden, die PostScript direkt unterstützt. Ansonsten müssten die benötigten Schriften als Binärobjekte in den zu generierenden PostScript-Text eingebettet werden.

[2] Bei den herunterladbaren Materialien zu diesem Übungsblatt finden Sie auch die Datei *ptmr8a.afm* für den Schriftschnitt *Times-Roman* von Adobe. Die Metriken zu alternativen Schriftschnitten im AFM-Format finden Sie auf der Theon unterhalb des Verzeichnisses */opt/ulm/ballinrobe/texlive/texmf-dist/fonts* oder unter */usr/share/texlive/texmf-dist/fonts/afm* auf einem Debian-System.

sind und zieht diese gleichmäßig auseinander. Andernfalls wird gewichtet auseinandergezogen bzw. zusammengepresst.

Die PostScript-Ausgabe erfolgt erst ganz am Schluss, wenn Ihr Schachtelsystem fertig gebaut ist:

```
Item box = /* ... */; // die Box, die alles enthält
PostScriptContext context = new PostScriptContext();
System.out.print(EPSWrapper.gen(context, box));
```

Der *EPSWrapper* erzeugt dann eine gültige EPS-Datei mit einer korrekt berechneten Bounding-Box.

Wenn Sie zu Testzwecken die Schachteln selbst visualisiert haben möchten, können Sie die dafür vorgesehenen Wrapper verwenden. Zunächst sollten Sie dem *Sequencer* mitteilen, dass alle erzeugten Schachteln einzurahmen sind:

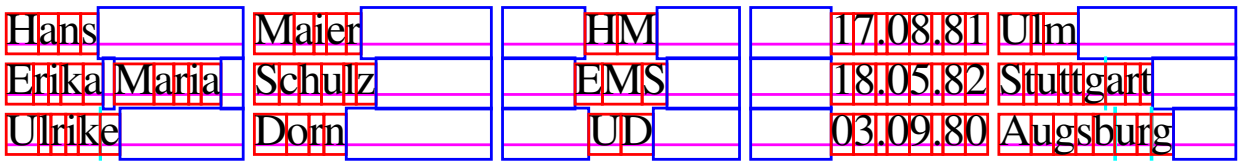
```
Sequencer s = new Sequencer(hseq, fm, fontSize); // wie oben
// alle Schachteln mit Rahmen versehen
s.setWrapper(new FrameWrapper());
```

Schachteln, die nicht vom *Sequencer* verarbeitet werden, können auch explizit eingeschachtelt werden:

```
box = new FramedItem(box);
```

Wenn eine vollständige Zeile abgeschlossen und in eine *HorizontalBox* verpackt ist, lohnt sich ggf. auch der Einsatz von *LinedItem*.

So könnte dann eine beispielhafte Ausgabe mit visualisierten Schachteln aussehen:



Diese Ausgabe wurde mit dem folgenden Kommando erzeugt:

```
TableGenerator ptmr8a.afm 16 18 440 LLZRL <in >out.eps
```

Wenn Sie mit Ihrer Lösung fertig sind, sollten Sie alle notwendigen Dateien mitsamt den Quellen in ein ausführbares Java-Archiv *table.jar* packen und dieses mit dem Kommando

```
submit typo 9 table.jar
```

auf der Theon oder Thales einreichen.

Viel Erfolg!