Bemerkung 0.1 – Kompilierung

Dieses Dokument wurde kompiliert durch:

```
pdflatex test-lst.tex
```

1 Einfache Benutzung zum Code-Highlighting, etlicher Sprachen

Java:

```java
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("Hallo Welt");
}
```

C++:

```cpp
#include <iostream>
public static int main(int argc, char** argv) {
    std::cout << "Hallo Welt" << std::endl;
    std::cout << "Die Äntwort ist: " << 21*2+14-(2*7)
        << std::endl;
}
```

Die Existenz einer Sprache kann dank LILLYxLIST leicht geprüft werden:

```
\isLanguageLoaded{java/lJava} % → 'TRUE' (\true)
\isLanguageLoaded{super/lSuper} % → 'FALSE' (\false)
```

Bemerkung 1.1 – Notation

Soll dies auf nur '`java`' geändert werden?

Alle geladenen Sprachen finden sich in der Liste `RegisteredLanguages`.

2 Besonderheiten

Durch die Syntax '`:<Keyword>:`' können Sonderzeichen und andere tolle Features in listings eingeführt werden, so wurden die oberen Pfeile durch '`:yields:`' erzeugt. Weiter interessant sind vielleicht noch '`:lan:`' (left angle: '⟨') und '`:ran:`' (right angle: '⟩')

3 Befehle

Für jede Sprache wie '`Java`' existiert ein entsprechend kleingeschriebenes Environment[1] was das Highlighting übernimmt (hier: '`java`'), soll die Sprache ohne Firlefanz gesetzt werden, so existiert `lstplain`:

```
\begin{lstplain}[language=lJava]
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("Hallo Welt");
}
\end{lstplain}
```

Ergibt:

```
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("Hallo Welt");
}
```

Für das inline-Code-Highlighting, existieren für jede Sprache `c<Sprache>` (code) und `b<Sprache>` (blank):

```
\cpython{print('hallo')}, \bpython{print('hallo')}
```

`print("hallo")`, `print('hallo')`

Ersterer kann aufgrund der Implementation des Hintegrund nicht umgebrochen werden, Listings versucht sich allerdings in einem normalen Maße der badparagraph (überlauf Problematik) anzunehmen. Letzterer kann ganz normal umgebrochen werden. Hier ein Beispiel:

```
print('hallo welt, na wie geht es dir? dies ist sehr langer Text, ein bisschen unangenehm oder? aber irgendwie muss ich die Zeile sprengen tut mir wirklich leid.')
```

Dahingegegen:

```
print('hallo welt, na wie geht es dir? dies ist sehr langer Text, ein bisschen unangenehm oder? aber irgendwie muss ich die Zeile sprengen tut mir wirklich leid.')
```

Weiter existiert noch '`i<Sprache>`', diesem Befehl kann eine Datei übergeben werden der diese dann in der jeweiligen Sprache hervorhebt. Beispiel mit diesem Dokument:

```
\ilatex{test-lst.tex}
```

Da es sich hierbei auch um das Dokument handelt, findet sich das Ergebnis auf der nächsten Seite.

[1] C++ ist 'cpp'

4 Dokumentcode

```
\documentclass{article}

\usepackage{LILLYxLISTINGS}
\usepackage{LILLYxCONTROLLERxBOX}

\begin{document}

\begin{bemerkung}[Kompilierung]
   Dieses Dokument wurde kompiliert durch: \begin{center}
      \bbash{pdflatex test-lst.tex}
   \end{center}
\end{bemerkung}

\section{Einfache Benutzung zum Code-Highlighting, etlicher Sprachen}
Java:
\begin{java}
public static void main(String[] args) {
   int i = 42*3;
   System.out.println("Hallo Welt");
}
\end{java}
C++:
\begin{cpp}
#include <iostream>
public static int main(int argc, char** argv) {
   std::cout << "Hallo Welt" << std::endl;
   std::cout << "Die Äntwört ist: " << 21*2+14-(2*7) << std::endl;
}
\end{cpp}
Die Existenz einer Sprache kann dank \LILLYxNOTExLibrary{LILLYxLIST} leicht geprüft werden:
\begin{latex}
\isLanguageLoaded{java/lJava} % → 'TRUE' (\true)
\isLanguageLoaded{super/lSuper} % → 'FALSE' (\false)
\end{latex}
\begin{bemerkung}[Notation]
   Soll dies auf nur '\verb|java|' geändert werden?
\end{bemerkung}
Alle geladenen Sprachen finden sich in der Liste \verb|RegisteredLanguages|.
\section{Besonderheiten}
```

```
Durch die Syntax '\verb|:<Keyword>:|' können Sonderzeichen und andere tolle Features in listings eingeführt werden, so wurden die oberen Pfeile durch '\verb|→|' erzeugt. Weiter interessant sind vielleicht noch '\verb|⟨|' (left angle: '\bjava{⟨}') und '\verb|⟩|' (right angle: '\bjava{⟩}')

\section{Befehle}
Für jede Sprache wie '\verb|Java|' existiert ein entsprechend kleingeschriebenes Environment\footnote{C++ ist 'cpp'} was das Highlighting übernimmt (hier: '\verb|java|'), soll die Sprache ohne Firlefanz gesetzt werden, so existiert \verb|lstplain|: \begin{latex}
\begin{lstplain}[language=lJava]
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("Hallo Welt");
}
\end{lstplain}
\end{latex}
Ergibt:
\begin{lstplain}[language=lJava]
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("Hallo Welt");
}
\end{lstplain}
Für das inline-Code-Highlighting, existieren für jede Sprache \verb|c<Sprache>| (code) und \verb|b<Sprache>| (blank):
\begin{latex}
\cpython{print('hallo')}, \bpython{print('hallo')}
\end{latex}
\cpython{print("hallo")}, \bpython{print('hallo')}\newline
Ersterer kann aufgrund der Implementation des Hintegrund nicht umgebrochen werden, Listings versucht sich allerdings in einem normalen Maße der bad-paragraph (überlauf Problematik) anzunehmen. Letzterer kann ganz normal umgebrochen werden. Hier ein Beispiel:\newline
\cpython{print('hallo welt, na wie geht es dir? dies ist sehr langer Text, ein bisschen unangenehm oder? aber irgendwie muss ich die Zeile sprengen tut mir wirklich leid.')}\newline
Dahingegegen:\newline
\bpython{print('hallo welt, na wie geht es dir? dies ist sehr langer Text, ein bisschen unangenehm oder? aber
```

```latex
    irgendwie muss ich die Zeile sprengen tut mir wirklich leid.')}\medskip\newline
Weiter existiert noch '\verb|i<Sprache>|', diesem Befehl kann eine Datei übergeben werden der diese dann in der jeweiligen Sprache hervorhebt. Beispiel mit diesem Dokument:
\begin{latex}
\ilatex{test-lst.tex}
\end{latex}
Da es sich hierbei auch um das Dokument handelt, findet sich das Ergebnis auf der nächsten Seite.
\clearpage
\section{Dokumentcode}
\ilatex{test-lst.tex}
\end{document}
```