



## SWP Übersetzerbau im SS 13

Einführung und Organisatorisches

Till Zoppke   Maximilian Konzack   Yves Müller  
Freie Universität Berlin

Auftaktveranstaltung am 13. April 2013

Projektidee

Einteilung in Gruppen

Organisatorisches

- Treffen

- Bürozeiten der Betreuer

- Bewertung

- Repositories

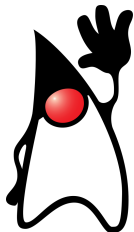
Git Primer

Ende



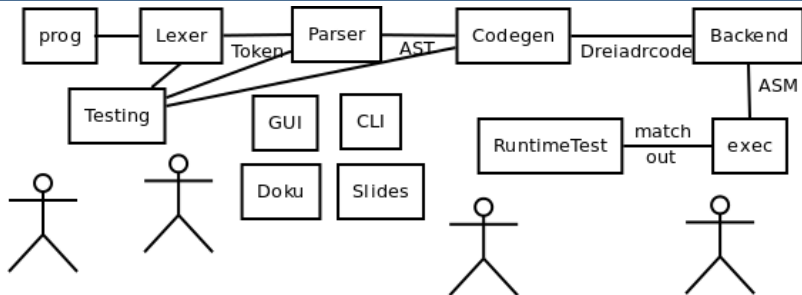
## Idee

- ▶ Implementierung eines Übersetzers
- ▶ soll im Rahmen der Übersetzerbau genutzt werden können
- ▶ Quellsprache ist imperativ und statisch typisiert



## Ziele

- modular** Abgrenzung gegenüber anderen Modulen
- einfach** Verwendbarkeit mit anderen Komponenten
- getestet** Black- und Whiteboxtests
- dokumentiert** im Quellcode und als Text



- ▶ bekannte Aufteilung in Front- und Backend
- ▶ Projektmanagement bezogene Artefakte
  1. Präsentationen
  2. Dokumentation
- ▶ Visualisierung als GUI und/oder Commandline Interface von
  1. abstrakter Syntax (AST) als ASCII, xml, SVG, ...
  2. drei Adresscode als Text, Tripel-, Quadrupeldarstellung, ...
  3. anderen Ergebnissen: Interpreter, Debugger, ...

# Aufteilung in Gruppen

## Gruppengröße

- ▶  $\approx 25$  Teilnehmer im KVV
- ▶ 2 Gruppen mit  $\approx 12,5$  Teilnehmer

## Organisation

- ▶ grundsätzlich frei gestellt
- ▶ zwei unterschiedlichen Zielcode:
  1. Java Bytecode
  2. LLVM, GNU Assembler, ...
- ▶ jedoch legen wir Wert auf:
  1. Inkrementelle Software Entwicklung
  2. Implementierung
  3. Interface Spezifikation
  4. Automatisierung der verschiedenen Tests
  5. Visualisierung der Ergebnisse über Commandline, GUI, ...

## Treffen aller Teilnehmer

- ▶ alle zwei Wochen
- ▶ donnerstags von 14 bis 16 Uhr c.t. im SR 049
- ▶ bei zu vielen Fehlterminen wird Anwesenheitspflicht eingeführt!
- ▶ von jedem Teilnehmer wird erwartet mindestens 1x zu präsentieren

⇒ andere Woche für Statusbericht und projektinterne Treffen vormerken!

## Zweck der Projekttreffen

1. Arbeitsfortschritt
2. Wissensaustausch
3. Projektmanagement
4. Visualisierung der Ergebnisse
5. Probleme, Fragen, Diskussion, ...

# Vorgaben zu den Präsentationen

## Allgemeines

- ▶ Dauer eines Vortrags  $\approx$  15 Minuten
  - ▶ 10 Minuten davon für Präsentation
  - ▶ 5 Minute für Fragen vorsehen
- ▶ im Zweifel weniger Folien sind besser!

## Präsentation zum Meilenstein

Fokus auf Projektalltag:

1. Projektstruktur
2. Status zum Meilenstein
3. Aufgabenverteilung
4. Projektmanagement
5. Demonstration des Compilers
6. ...

## Fachvortrag

Fokus auf Wissensaustausch.

Mögliche Themen:

1. Aufbau und Struktur von Java Bytecode
2. git (Branch und Merge Strategien, Issues, ...)
3. Design Patterns (Visitor, Interpreter, Factory, ...)
4. JUnit



## Statusbericht über Meilenstein

- ▶ kurzer Status Strukturierung des Meilensteins
- ▶ wenn *kein Projekttreffen* ist

### Was wollen wir sehen?

mind. 1 Punkt aus folgenden:

- ▶ Offene und abgeschlossene Tickets/Issues
- ▶ To-Do-Liste
- ▶ Projektverlaufsplan
- ▶ Mündlicher Bericht
- ▶ ...

### Was *nicht*?

alles was im Vortrag zum Meilenstein gehört

## Bürozeiten der Betreuer

Betreuer des Softwareprojekts sind

1. Till Zoppke Email: [zoppke@zedat.fu-berlin.de](mailto:zoppke@zedat.fu-berlin.de)
2. Maximilian Konzack Email: [maximilian.konzack@fu-berlin.de](mailto:maximilian.konzack@fu-berlin.de)
3. Yves Müller Email: [yves.mueller@fu-berlin.de](mailto:yves.mueller@fu-berlin.de)

### Wo und wann?

Donnerstags 16 bis 18 Uhr im SR 158

### Wann zu nutzen?

bei

- ▶ Problemen im Team
- ▶ Unklarheiten
- ▶ Fragen
- ▶ Anregungen
- ▶ ...

umfasst

1. Quellcode
2. Dokumentation
3. Präsentationen
4. Abschluss

bezüglich der Meilensteine

## Meilensteine

M1 Arithmetik

M2 print Anweisung und Verzweigungen

M3 Schleifen und Arrays

# Repositories



1. Ein *allgemeines Repository* für projektübergreifende
  - ▶ Dokumentation
  - ▶ Beispiele
  - ▶ Tests
  - ▶ Interfaces
2. *jede* Gruppe erhält eigenes Repository für ihre Implementierung



## Was ist git?

1. Versionsverwaltung von Dateien, insbesondere für Quellcode
2. freie Software
3. geeignet für kleine bis große Projekte
4. kann nur lokal (auf einem Rechner) oder stark verteilt genutzt werden
5. viele große Open Source Projekte nutzen git

## Wichtige Befehle für die Arbeit mit git

1. Lokale Kopie vom Repository anlegen:

```
$ git clone <repo>
```

2. Revision ansehen:

```
$ git show <rev number>
```

3. Lokalen Veränderungen (noch ohne commit) ansehen:

```
$ git status
```

4. Commit auf lokalem Repo:

```
$ git commit -m "message" -a|<file>|<dir>
```

5. Veränderungen vom remote Repo ziehen:

```
$ git pull origin <branch>
```

6. Eigene Commits auf remote Repo hoch laden:

```
$ git push origin <branch>
```

1. Erstelle einen neuen Branch (und wechsele zu ihm):

```
$ git checkout -b <branch>
```

2. Zeige alle verfügbaren Branches:

```
$ git branch -a
```

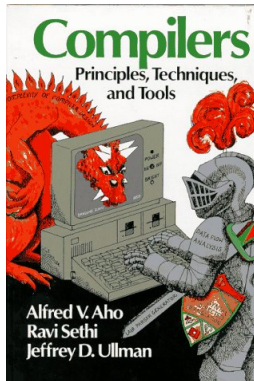
3. Wechsele vom aktuellen Branch zu angegeben:

```
$ git checkout <branch>
```

4. Merge von aktuellen Branch mit angegeben:







```
$ git merge <branch>
```

# Danke für die Aufmerksamkeit.





- FU Berlin, SWP Übersetzerbau im SS 13, Auftaktveranstaltung am 13. April 2013

-  Alfred V. Aho, Jeffrey Ullman, and Ravi Sethi.  
*Compiler: Prinzipien, Techniken und Werkzeuge.*  
Pearson Studium, 2. edition, 2008.
-  FindBugs – Find Bugs in Java Programs.  
<http://findbugs.sourceforge.net/>.
-  GitHub  
<https://github.com/>.
-  Git Reference  
<http://gitref.org/>.
-  Some Notes on Git  
<http://java.dzone.com/articles/some-notes-git>.
-  Michael Lee Scott.  
*Programming language pragmatics.*  
Morgan Kaufmann Publishers, 3. edition, 2009.