Freie Universität Berlin

SWP Übersetzerbau Sommer 2010
DOM + Builder

Stefan Meißner
Institut für Informatik
FU Berlin

Agenda

I. Aufgabenübersicht

II. DOM-Tree
   I. Aufbau
   II. Umsetzung & Anwendung

III. Builder
   I. Aufbau
   II. Umsetzung & Anwendung

Aufgaben I

Erstellung des DOMs (Document Object Model)

Ziel:

- Quellcode soll leicht verarbeitet werden können
- DOM repräsentiert den Quellcode hierarchisch

=> Einfache Navigation beim Erstellen des Syntaxbaums möglich

Eingliederung in den Entwicklungsprozess:

... -> Lexer -> **DOM** -> Syntaxbaum -> ...

Aufgaben II

Erzeugung des Zwischencodes

Ziel:

- möglichst architekturunabhängiger Code
- muss nicht ausführbar sein
- muss nicht optimiert sein

Jedoch: unser Compiler generiert Java Source als Zwischencode

Eingliederung in den Entwicklungsprozess:

... -> Syntaxbaum -> **Zwischencode**

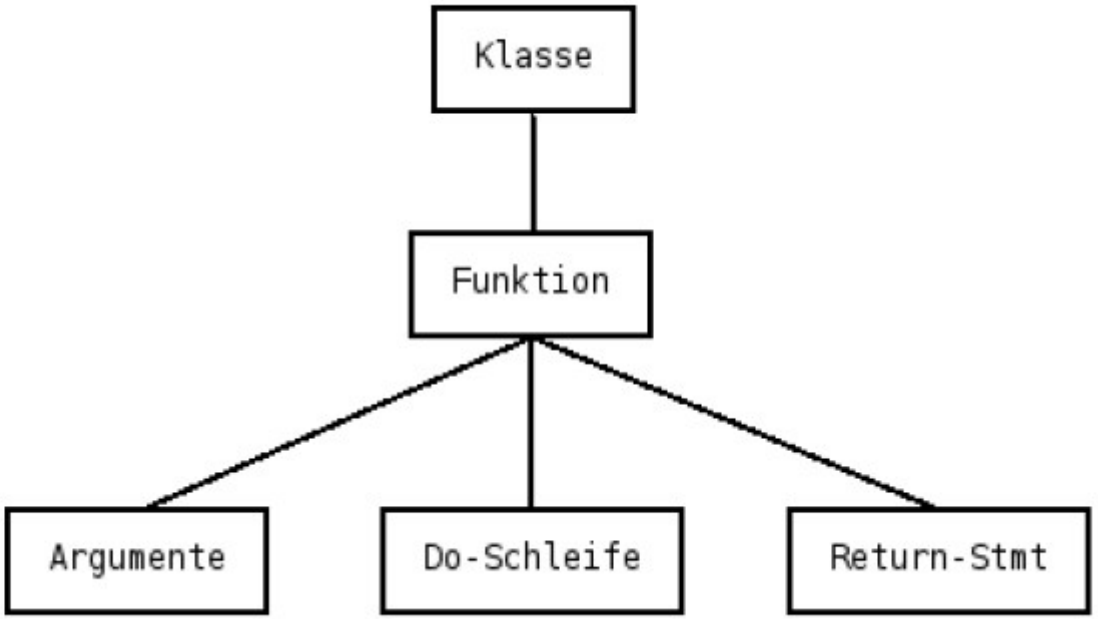
- Letzter Teilprozess dieses Projektes

DomTree

Aufbau

- DOM bildet eine Hierarchie
- jeder Knoten kann als eigener unabhängiger DOM betrachtet werden
- Knoten sind attributiert

```
<class>
  <function>
      <arguments>
          <decl>
          <decl>
      </arguments>
      <do>
          <decl>
          <set>
          <set>
      </do>
      <return>
  </function>
</class>
```



DomTree

Umsetzung

- klassische Baumdatenstruktur
- Knoten: DomNode.java
- Attribute: DomAttribute.java
- Erzeugung: DomCreator.java
- Zusammenarbeit mit Lexer-Team tadellos

Nebenutzen bei der Erzeugung

- XML-Struktur wird validiert (Syntax)
- Semantik wird im Syntaxbaum geprüft

Verbesserungsmöglichkeiten

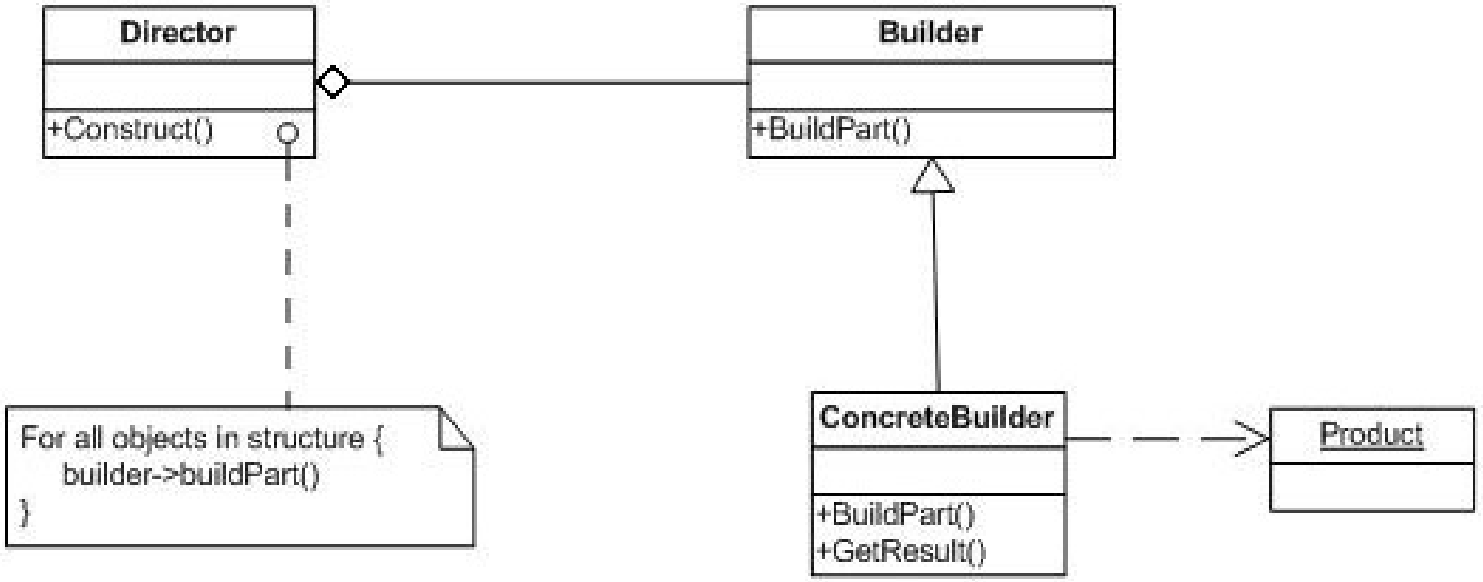
- Faktory zum Erzeugen eines DOMs pro File

Umsetzung

- Builder Pattern:

  Trennung komplexer Aufbau des Syntaxbaums und dessen Darstellung als spezifischer Zwischencode

=> unabhängige Erzeugung Codeerzeugung in ausgewählten Sprachen



Umsetzung

- Builder kümmert sich um die komplette

  (Zwischencode-)Erzeugung

- jedes Programmierelement hat eigene Implementation

- konkreter JavaBuilder erzeugt für jedes Element

  JavaCode (Product)und delegiert ggf. das Erzeugen

  von Unterelemente an andere Builderfunktionen

Umsetzung

- Director kümmert sich um die Reihenfolge der zu bauenden Teile
- Director besitzt nur Funktion build, in der die Reihenfolge der Codeerzeugung festgelegt ist
- Director legt für jede Klasse und für jedes Interface eine neue Datei an (jedoch nur Java Datein, sollte abstrakter sein)

Vielen Dank!