Freie Universität Berlin

Kundenpräsentation: Lexer und Symboltabelle
SWP Übersetzerbau Sommer 2010

Institut für Informatik
FU Berlin

- Inhaltsübersicht:
  - Übernommene Aufgaben und Programmierstil
  - Grundlegende Datentypen
  - Lexen der XML-Eingaben
  - Lexen der Ausdrücke
  - Die Symboltabelle
  - Fazit

- Jeweils:
  - Einführung in die Thematik
  - Klassenstruktur
  - Entstandene Probleme
  - Zusammenfassung der Umsetzung

Übernommene Aufgaben und Programmierstil

Übernommene Aufgaben

- Allgemeine Programmieraufgaben
  - Nicht auf Quell- oder Zielsprache spezialisiert
  - Augenmerk auf hohe Wiederverwendbarkeit
  - Stark modularisiert
  - Strikte Trennung zwischen Interface und Implementierung
  - Somit:
    - Einzelteile einfach ersetzbar
    - Fehler einfach auffindbar

Programmierstil

- Allgemeine Programmieraufgaben

- Augenmerk auf Verwendung von Entwurfsmustern
  - Verringern der Einlesezeit
  - Factory-Pattern
    - Nur ein Einstiegspunkt, jener aber wohldefiniert
    - Statisches Interface
    - Dadurch konkrete Implementierung ersetzbar
  - Iterator-Pattern
    - Eingabe wird zu einem „Strom an Daten“
    - Nur zwei Methoden müssen verwendet werden:
      - `hasNext()` – existieren weitere Daten?
      - `next()` – ein Datum einlesen
  - Beide Entwurfsmuster in der OOP weit verbreitet

Grundlegende Datentypen

Grundlegende Datentypen

- Grundlagentypen ziehen sich durch das gesamte Projekt hindurch

- Die „Stream-Position“
  - Konstrukte (z.B. Schleifen und Bezeichner) kennen jeweils Ort, an dem sie in der Eingabe standen
  - Realisiert durch:
    - *StreamPosition:* allgemeines Interface
    - *PositionBean:* Implementierung der *StreamPosition*
    - *PositionString:* String der seine Position „kennt“

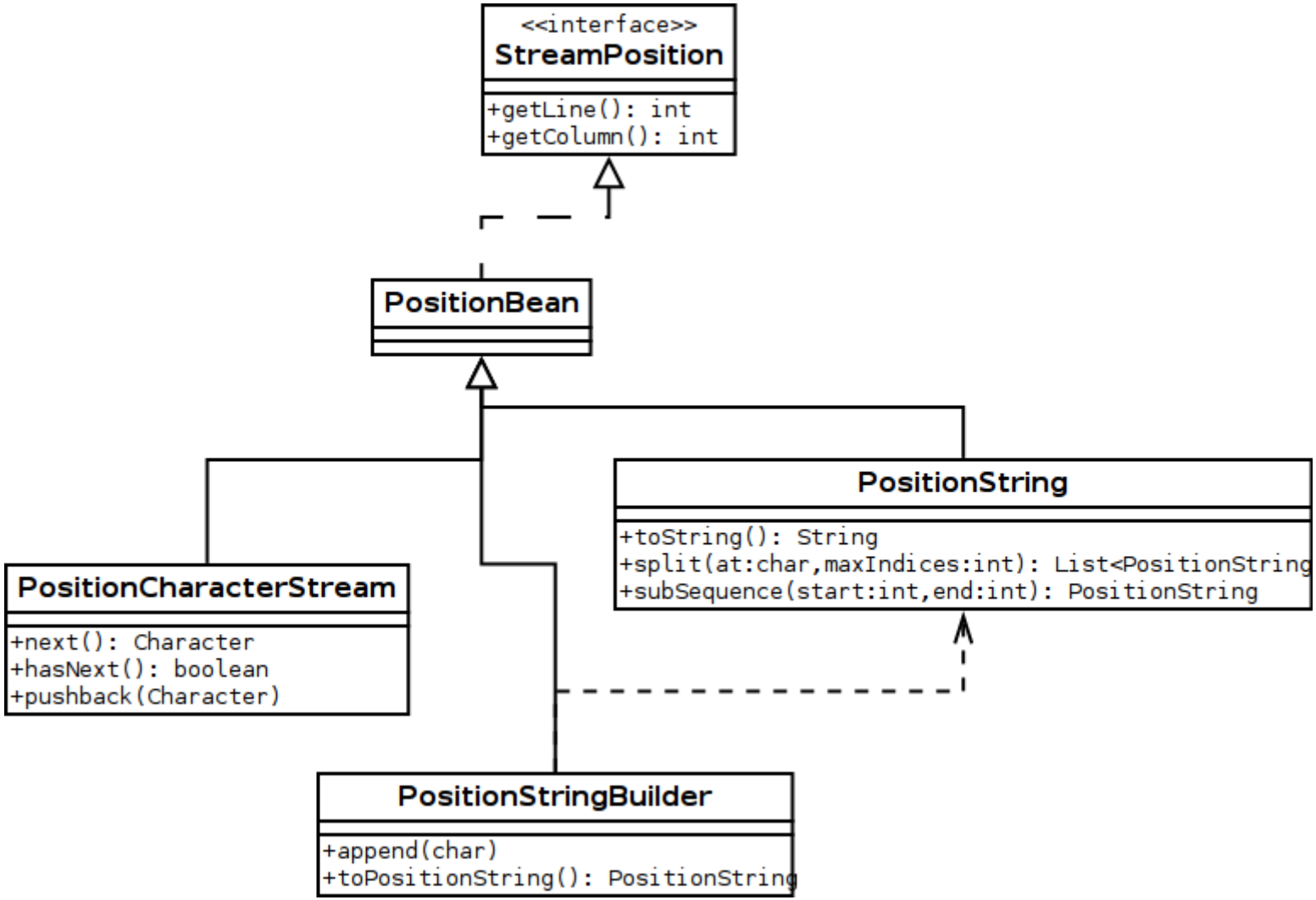
Grundlegende Datentypen

- Grundlagentypen ziehen sich durch das gesamte Projekt hindurch

- Die „Stream-Position“
  - Konstrukte (z.B. Schleifen und Bezeichner) kennen jeweils Ort, an dem sie in der Eingabe standen
  - Realisiert durch:
    - *StreamPosition:* allgemeines Interface
    - *PositionBean:* Implementierung der *StreamPosition*
    - *PositionString:* String der seine Position „kennt“

    - So wie weitere Helferklassen
    - Und alle Klassen, die das Interface benutzen
      - Hervorstehend: Die Fehlerausgabe

- Klassenstruktur



Grundlegende Datentypen

- Entstandene Probleme
  - StreamPosition noch nicht allgemein genug
    - Nicht für mehrere Eingabedateien geeignet
- Stand der Umsetzung
  - Noch nicht von allen Klassen benutzt

Lexen der XML-Eingaben

Lexen der XML-Eingaben

- Der Begriff *lexen*
  - „Lexer“, kurz für „lexikalischer Scanner“
  - Das Zerteilen eines Datenstromes in syntaktische Teilstücke (Tokens)
  - Aufbereitung der Daten für die semantische Analyse

Lexen der XML-Eingaben

- Der Begriff *lexen*

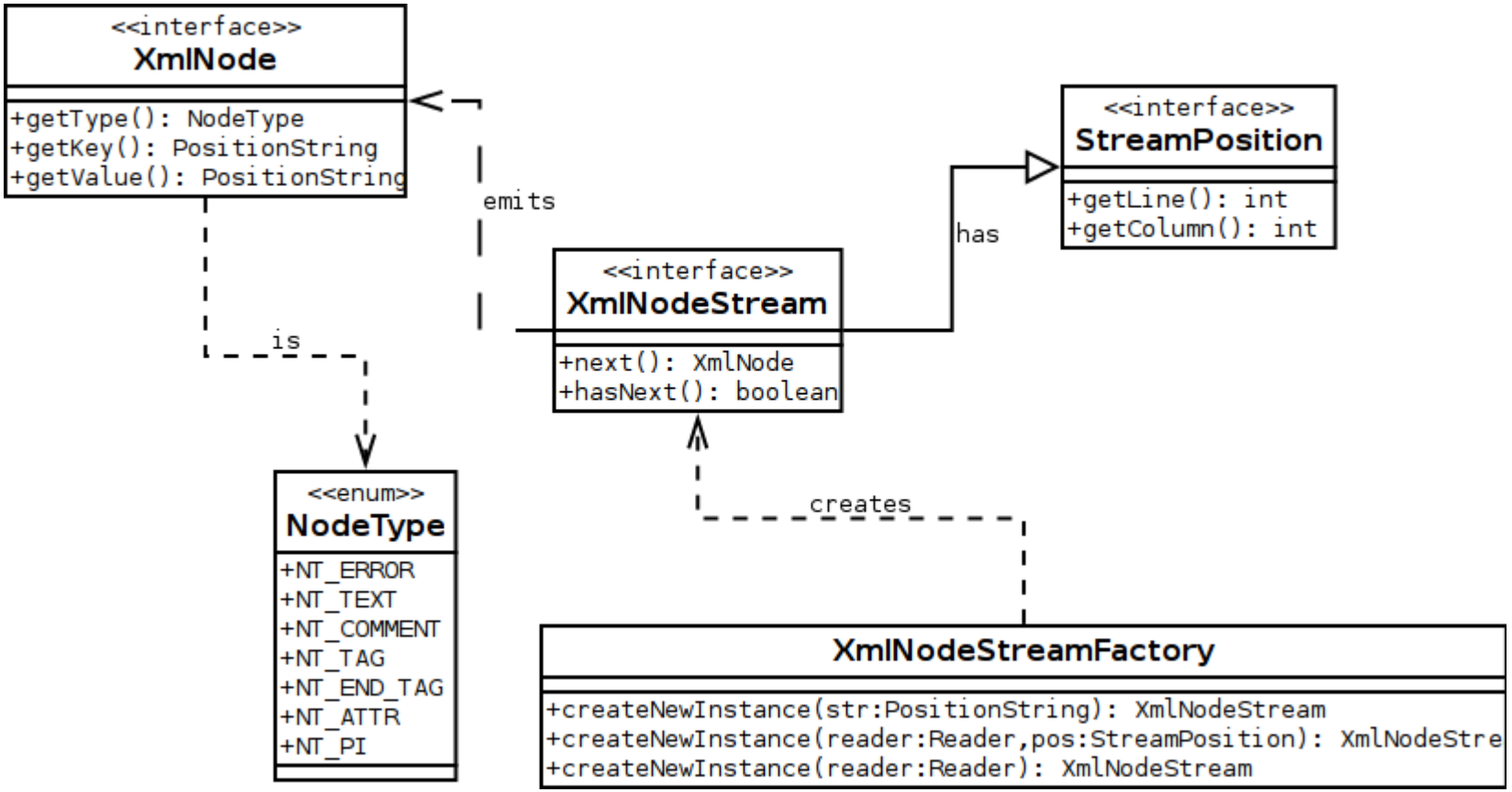
- Die Struktur der Programmiersprache hat einen XML-artigen Aufbau
  - mit geringfügigen Vereinfachungen für den Anwender
    - U.a. „<“ und „>“ in Attributen erlaubt
  - Eingabe als Rohtext im Format
    - ```
      <module name="Wert">
          abc
      </module>
      ```
  - Verstanden als
    - Öffnendes Tag „`module`“
    - Attribute „`name`“ = „`Wert`“
    - Text „`abc`“
    - Schließendes Tag „`module`“
      ```

Lexen der XML-Eingaben

- Der Begriff *lexen*
- Die Struktur der Programmiersprache hat einen XML-artigen Aufbau
- Einlesen erfolgt in Anlehnung an *StAX*
  - *Streaming API for XML*
  - Moderner Quasistandard zum Parsen von XML-Eingaben in Java
  - Schrittweises Erkennen einzelner Lexeme
    - Öffnende Tags, Attribute, Texte, schließende Tags, u.ä.
  - Vereinfacht Aufbau eines Baumes

Lexen der XML-Eingaben

- Klassenstruktur



Lexen der XML-Eingaben

- Entstandene Probleme
  - „Gewissensfragen“: Nähe zu XML-Standard oder mehr Vereinfachungen?
    - U.a. `<set name="cmp" value="a<b" />` wird nicht beanstandet
    - Nur eine Processing Instruction (quasi ein Pragma für den XML-Parser) pro Datei wird erlaubt

Lexen der XML-Eingaben

- Entstandene Probleme

- Zusammenfassung der Umsetzung
  - Kann allgemeine XML-Daten lesen
    - Selbstanspruch: Wiederverwendbarkeit
  - Implementierung des Iterator-Interfaces
    - Anwender brauchen nur `hasNext()` und `next()` aufrufen
  - Nichtüberprüfen auf Wohlgeformtheit
    - „<a>…“ ist kein Fehler
    - Somit auch als Lexer für durchschnittliche HTML-Seiten verwendbar

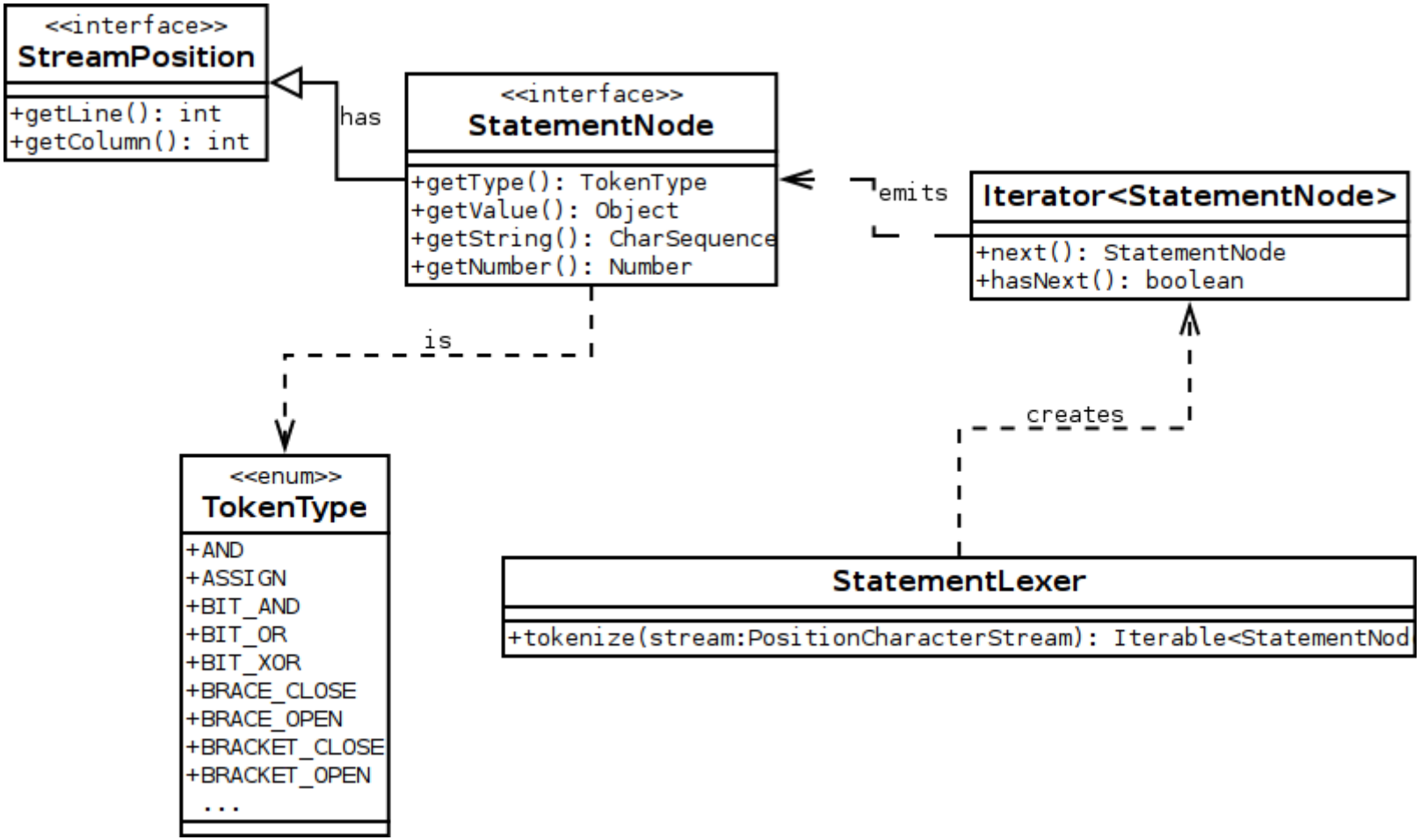
Lexen der Ausdrücke

Lexen der Ausdrücke

- Syntaktisches Verstehen von Java-Ausdrücken:
  - `(1+2) * 3`
    - öffnende Klammer
    - Zahl „1“
    - Plus
    - Zahl „2“
    - …
- Ähnliche Aufgabe wie das Lexen der XML-Daten
  - Nur mit mehr Lexem-Typen

Lexen der Ausdrücke

- Klassenstruktur



Lexen der Ausdrücke

- Entstandene Probleme
  - Designfragen:
  - Abweichung von Java-Syntax, wenn sinnvoll
    - `1e3` bezeichnet Ganze Zahl 1000, wäre aber Float in Java
  - Schwergewichtige Objekte zur Vereinheitlichung
    - Integers und Floats durch BigDecimal gekapselt

- Zusammenfassung der Umsetzung
  - Anpassungen an XML noch nicht vollständig
    - Zeichenreferenzen wie `>` werden nicht verstanden

Die Symboltabelle

Die Symboltabelle

- Schnittstelle: Plattform- und Compiler-spezifische Datentypen
  - Für Plattform Java werden die Klassen Javas geladen
  - Interface ist Plattform-unabhängig
    - Strikte Trennung von Implementierung
    - Verwendung des Factory-Pattern
  - Java-ähnlicher Aufbau der Plattform jedoch zwingend

Die Symboltabelle

- Schnittstelle: Plattform- und Compiler-spezifische Datentypen

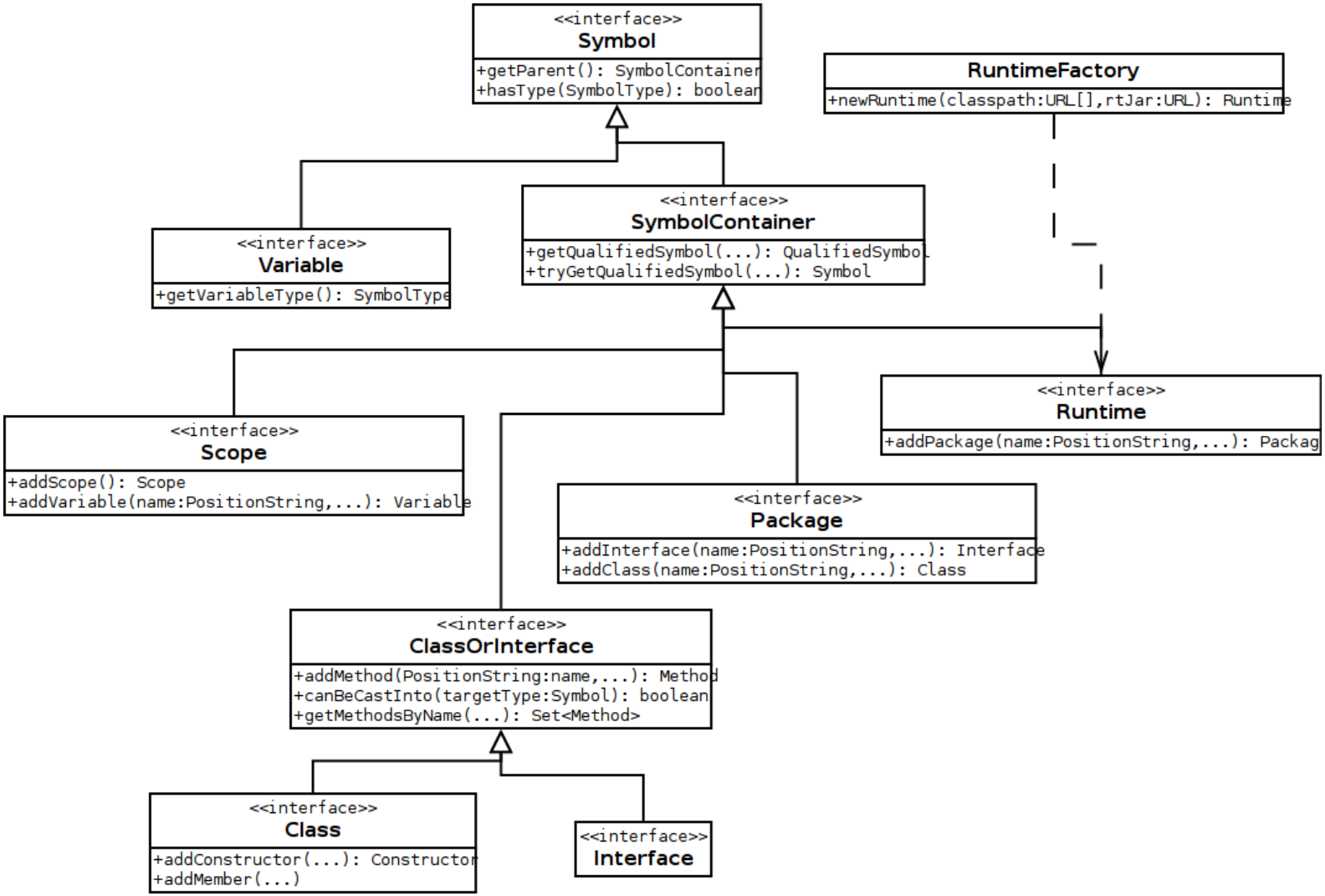
- Namensauflösung
  - Tabelle wird durch den Annotator mit Benutzerdaten befüllt
  - Verschachtelung und Sichtbarkeit durch Composite-Pattern
    - D.h. bspw., dass Methoden in Klassen und Klassen in Packages liegen
    - Interface zur Suche nach Bezeichnern stets gleich
      - egal ob auf Package, Klasse oder Methode angewandt
  - Speicherung der Referenzierungen von Symbolen
    - Erkennung von zirkulären Abhängigkeiten
    - Hilfreiche Fehlermeldungen für Anwender

Die Symboltabelle

- Schnittstelle: Plattform- und Compiler-spezifische Datentypen
- Namensauflösung
- Gültigkeit von Bezeichnern
  - Abweisen von Schlüsselwörtern als Bezeichner
    - Name „int“ ist ungültig
  - Überprüfung von Bezeichnern auf Gültigkeit
    - Bspw. ist „123abc“ ungültig
  - Dekoration von ungültigen Bezeichnern
    - Ermöglichen von landessprachigen Bezeichnern

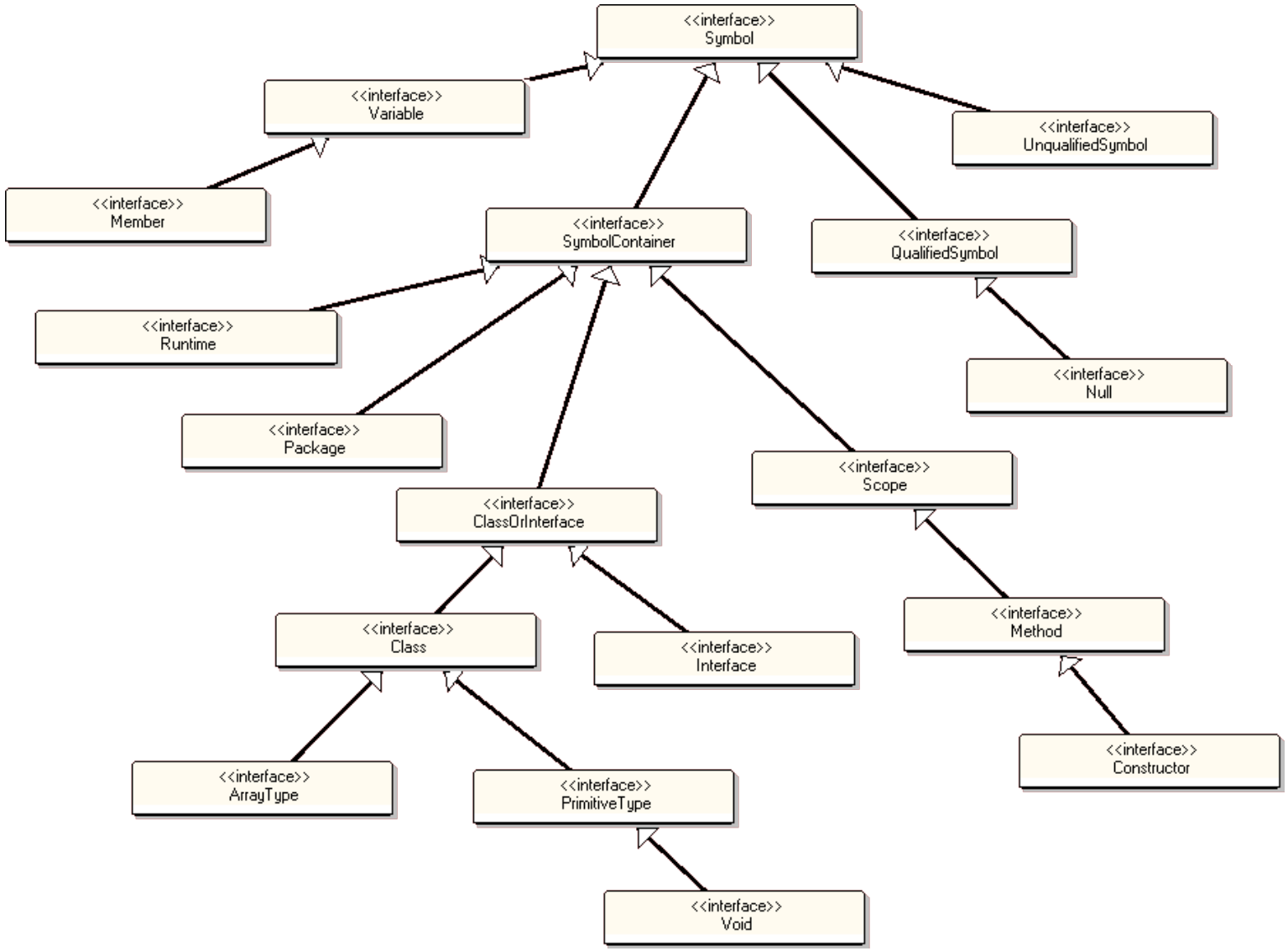
Die Symboltabelle

- Klassenstruktur (stark vereinfacht)



Die Symboltabelle

- Klassenstruktur (Vererbungshierarchie)



Die Symboltabelle

- Entstandene Probleme
  - TODO
- Zusammenfassung der Umsetzung
  - TODO

Fazit

Fazit

- TODO

Vielen Dank!