TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

「近」未来のプログラミング言語

千葉　滋
東京工業大学
Seasar Conference Autumn on 2008 September 6

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

言語 v.s. フレームワーク

- フレームワークとは…
  - Seasar2 とか
  - 一般的に必要な機能をあらかじめ提供
  - 開発者を制約

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

プログラミング言語とは…

- 目的
  - 一般的に必要な機能をあらかじめ提供
    - 機械語で書かなくてもよいように
      - for 文、ライブラリ関数、…
  - 開発者を制約
    - Goto 文を使わせない
    - オブジェクト指向を強制する

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

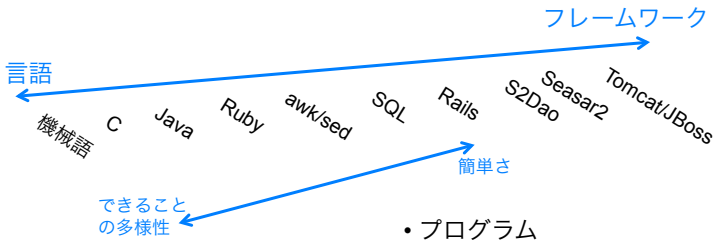
Cobol はフレームワークか？

- 当時としては Yes?
  - ビジネスアプリケーションに特化した言語
  - OS は書けない
  - 簡易言語？　今で言うスクリプト言語？

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

言語からフレームワーク

- 境界はあいまい



- プログラム
- クラス・関数ライブラリ
- 設定ファイル

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

設定ファイル

- フレームワーク上で動くプログラム（スクリプト）？
  - 独自言語
  - Wizard 形式（対話的な入力）
  - Shell Script
  - XML
  - Java Annotations
    - プログラミング言語と設定ファイルの融合

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

専用言語のススメ

- 設定ファイルを何言語で書くか
  - Annotations では不十分
  - XML hell はイヤ
  - Java ... だけど、少し専用の文法拡張が入っている
    - 実装の手間が膨大で、非現実的？

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

例えば

```
@S2Dao(bean = Book.clss)
public interface BookShelf {
  @Sql("SELECT count(*) FROM books")
  public int getCount();
      :
}
```

```
@S2Dao(Book)
public interface BookShelf {
  @Sql public int getCount() {
    return SELECT count(*) FROM books;
  }
      :
}
```

もちろんコンパイラ・チェック、
Eclipse plug-in 有り

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

コンパイラの授業

- 字句解析　　lex, Jflex, …
  - 文字列から token（単語）列に。空白は除去。
- 構文解析　　yacc, bison, Beaver, …
  - token 列から抽象構文木を作成。
- 意味解析
  - 型検査など、各種コンパイル・エラーの検出
- コード生成

参考 http://www.csg.is.titech.ac.jp/~chiba/lecture/os/

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

字句解析

- Token（単語）の切り出し
  - “int num = 0;”
    → “int”, “num”, “=”, “0”, “;”
- Jflex での定義

Java コード

```
{[A-z][A-z0-9]*}   { return new Symbol(Terminals.IDENTIFIER); }
“int”              { return new Symbol(Terminals.INT); }
                          :
```

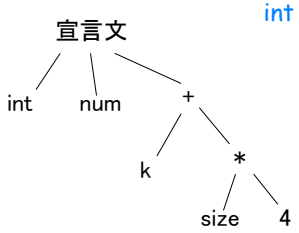
正規表現

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

構文解析

- 抽象構文木 (Abstract Syntax Tree) を作る
  - 式や文単位に token をまとめる
  - 括弧や ; など不要な token を捨てる

int num = k + size * 4;



TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

Parser generator

- BNF による文法定義
  - token が構文のどこに対応するかを自動判定
  - 木の作り方は Java コードで書く

```
Expr add_expr =
    mul_expr.e                          {: return e; :}
  | add_expr.e1 PLUS mul_expr.e2        {: return new AddExpr(e1, e2); :}
  | add_expr.e1 MINUS mul_expr.e2       {: return new SubExpr(e1, e2); :}
```

k + p * 2 - n
e1 e2
e1 e2

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

意味解析、コード生成

- 決定版といえるツールはまだない
- 手でプログラムを書く
  - 構文木を visitor パターンで何度もたどりながら
    - 名前解決 （これは変数名か、フィールド名か、…）
    - 型検査
    - Visibility の検査（public, private など）
      :
    - コード生成
    - 最適化　　Jimple などを利用可

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

JastAdd

http://jastadd.org/

- コンパイラ・コンパイラ
  - 属性文法による意味解析（型検査）
  - （少し）アスペクト指向
  - 字句解析と構文解析は好きなツールを利用可能
    - AST ノード・クラスの定義言語は JastAdd が提供
- JastAddJ
  - JastAdd で作られた Java 1.4 & 1.5 コンパイラ
  - Java 言語の拡張版を作るときのベース

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

ASTノード・クラスの定義

- 定義から自動的に Java クラスを生成

```
abstract Expr;
AssignExpr : Expr  ::=  Left : Expr    Right : Expr
```

```
abstract class Expr {}
class AssignExpr extends Expr {
   private Expr Left;
   private Expr Right;
   Expr getLeft() { return Left; }
   Expr getRight() { return Right; }
}
```

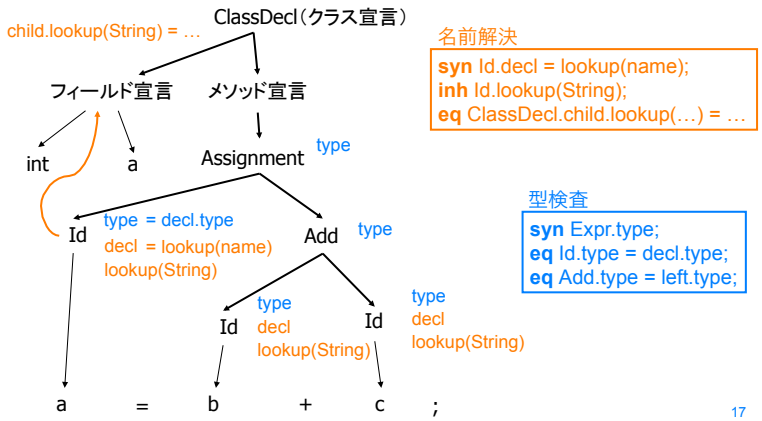

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

属性文法

- Attribute Grammar
  - TeX を作った Knuth の 1960 年代の仕事のひとつ
- 利点
  - 構文木のノードの属性を宣言的に定義
    - 属性の例： 型、ある変数が宣言された場所、定数値、…
    - JastAdd の属性は、ノードクラスの Java メソッド
  - 属性の値を自動計算
    - Visitor パターン hell から逃れられる
    - 属性間に依存関係があると自動的に計算順序を調整

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

合成属性と継承属性



TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

アスペクト指向

- 差分だけをまとめて別ファイルに定義できる
  - 各種定義に規則を追加、上書き
    - Token 定義、構文定義、AST ノード・クラス定義
  - (自動生成される) Java クラスに追加、上書き
    - メソッドやフィールド

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

構文拡張は plug-in

- Java 言語の新しい構文拡張
  - コンパイラ plug-in (= アスペクト) として実装
    - 1個のファイルにまとめられる
  - 元のコンパイラのソースは無変更
- JastAddJ
  - Java 1.4 Frontend 10.4K LOC
  - Java 1.4 Backend 4.6K LOC
  - Java 1.5 Frontend 4.9K LOC
  - Java 1.5 Backend 1.1K LOC

Plug-inとして実装

TOKYO TECH TOKYO INSTITUTE OF TECHNOLOGY

まとめ

- プログラミング言語 v.s. フレームワーク
- 専用言語のススメ
- JastAddJ
  - 最新のコンパイラ・コンパイラ
    - 属性文法ふたたび登場
  - Java 言語の拡張が容易に
    - 構文拡張はアスペクト指向により
      コンパイラ plug-in として実装可能