CSE 401 - Compilers Section 1

1/16/2013
12:30 - MEB 238
1:30 - EE 037

Who Am I?

Zach Stein

- 5th Year Masters Student
- Took 401 in Autumn 2011
- Email: steinz@cs
- Office Hours
  - Friday 1:30-2:30 in CSE 218
  - By appointment

Who Are You?

- Name

At Least One Favorite:

- Class
- Research or Personal Project
- Hobby
- Movie, TV Show, or Internet Meme
- [a-zA-Z]*

Anything specific you want to learn in this class?

Tip: Discussion Board Notifications

Discussion Board
> Profile
> Notifications
> Immediate / Daily Digest

Project Tools

Space: `/projects/instr/13wi/cse401/X`

Use `groups` to figure out what letter is yours

Alternative private repositories:

- bitbucket
- github

Tool Surveys:

- Eclipse? Command Line? Other IDE?
- ant?

Project Intro/Resources

Scanner due January 28

Anybody started? Start soon!

Homepage Grammar

READMEs

- /README
- /README.eclipse
- Scanner/README
- Parser/README

Scanner and Parser Generation

- JFlex compiles Scanner/minijava.jflex
- CUP compiles Parser/minijava.cup

- `Symbol` class shared by scanner and parser
- Tokens defined in CUP file

- ant tasks defined in build.xml generate scanner and parser from .jflex and .cup files

`Symbol` and `sym` Classes

```
class Symbol { // represents tokens
  int sym; // which token
  Object value; // extra data
  ...
}
class sym {
  static int LPAREN = 1;
  static int IDENTIFIER = 2;
  ...
}
```

`Symbol` and `sym` Classes

- Not very Java like...
  - Why isn't `Symbol` an enum?
  - `symbolToString(Symbol s)` in minijava.flex (not `Symbol.toString()`)
  - Historical reasons (LEX/YACC ports)
  - Feel free to make these more Java like if you're bored (and we'll use the code next time)

- Any MiniJava program is a Java program, so we need (but won't properly implement):
  - `public, static, void, main, String`
  - `System.out.println`

Token Declarations in CUP

```
/* operators: */
terminal PLUS, BECOMES;

/* delimiters: */
terminal LPAREN, RPAREN, SEMICOLON;

/* tokens with values: */
terminal String IDENTIFIER;
```

Tokens and REs in JFlex

Token definition: `regex { Java-statement }`

`regex` can be

- String literal - "class", "+"
- A helper in braces - {letter}
- Range (or negated range) - [a-z], [^\r\n]
- . (matches any single character)
- *rs*, *r*|*s*, *r**, *r*+, *r*?, (*r*) - where *r* and *s* are REs

```
/* Helper definitions */
letter = [a-zA-Z]
```

Tokens and REs in JFlex

Token definition: `regex { Java-statement }`

Do useful work with `Java-statement`

For us this will be things like:

- `return Symbol(sym.PLUS);`
- `return Symbol(sym.IDENTIFIER, yytext());`
- `/* ignore */`

Project Questions?

Homework Questions?

Anybody started?
It's due Friday night.

Other Questions?

Lecture?
Scanning?
Parsing?
Regular Expressions? DFAs? NFAs?

How's the Lecture Speed?

Too fast? Too slow? Just right?

How are the Readings?

Anybody done them?
Anybody looking for more?

An Amusing C++ Problem

`vector<vector<int>>`

Problem: `>>` is one lexical token (for shift right)

- C++03: Had to use `> >` instead
- But some compilers did the right thing anyway
  - Turns out it is never ambiguous
- C++11 (C++0x): Added support

Compiler implementors and language specifiers have to deal with these kinds of issues a lot.

Also: `int* a; int *a;` A PhD Thesis on Parsing C++

Regex Exercise

English -> Regular Expression -> NFA -> DFA

strings of 0s and 1s such that every sequence of two 1s must be preceded by at least two consecutive 0s and followed by at least three

(feel free to work with your neighbors)