Introduction to the 1st Obligatory Exercise

Scanning, parsing, constructing the abstract syntax tree

Department of Informatics
University of Oslo

Compiler technique, Friday 22 2013

Scanner and Parser Generators

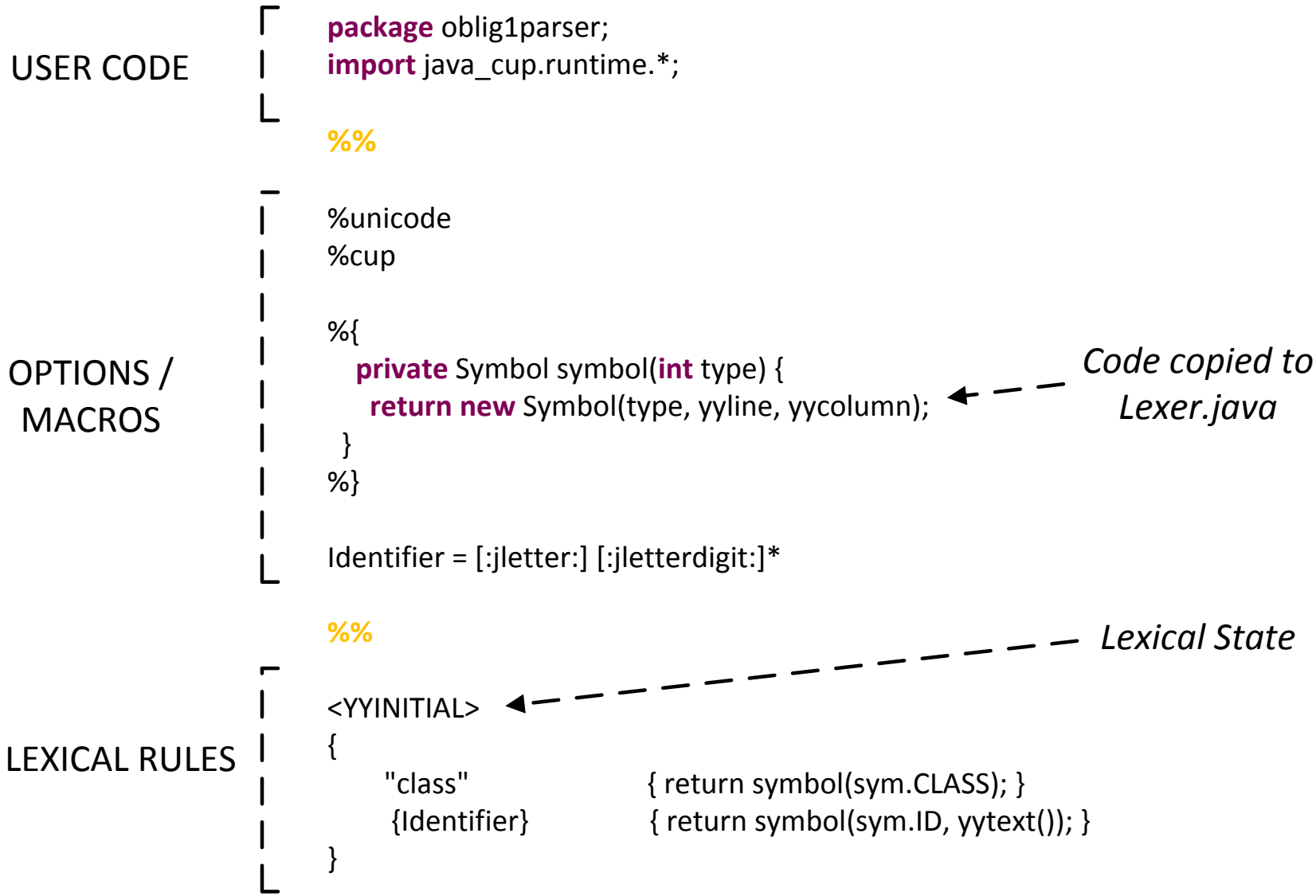
JFlex

JFlex

- Is specified in .lex file
- Specification consists of three parts:
  - User code
  - Options and macros
  - Lexical rules

Scanner and Parser Generators

JFlex



Scanner and Parser Generators

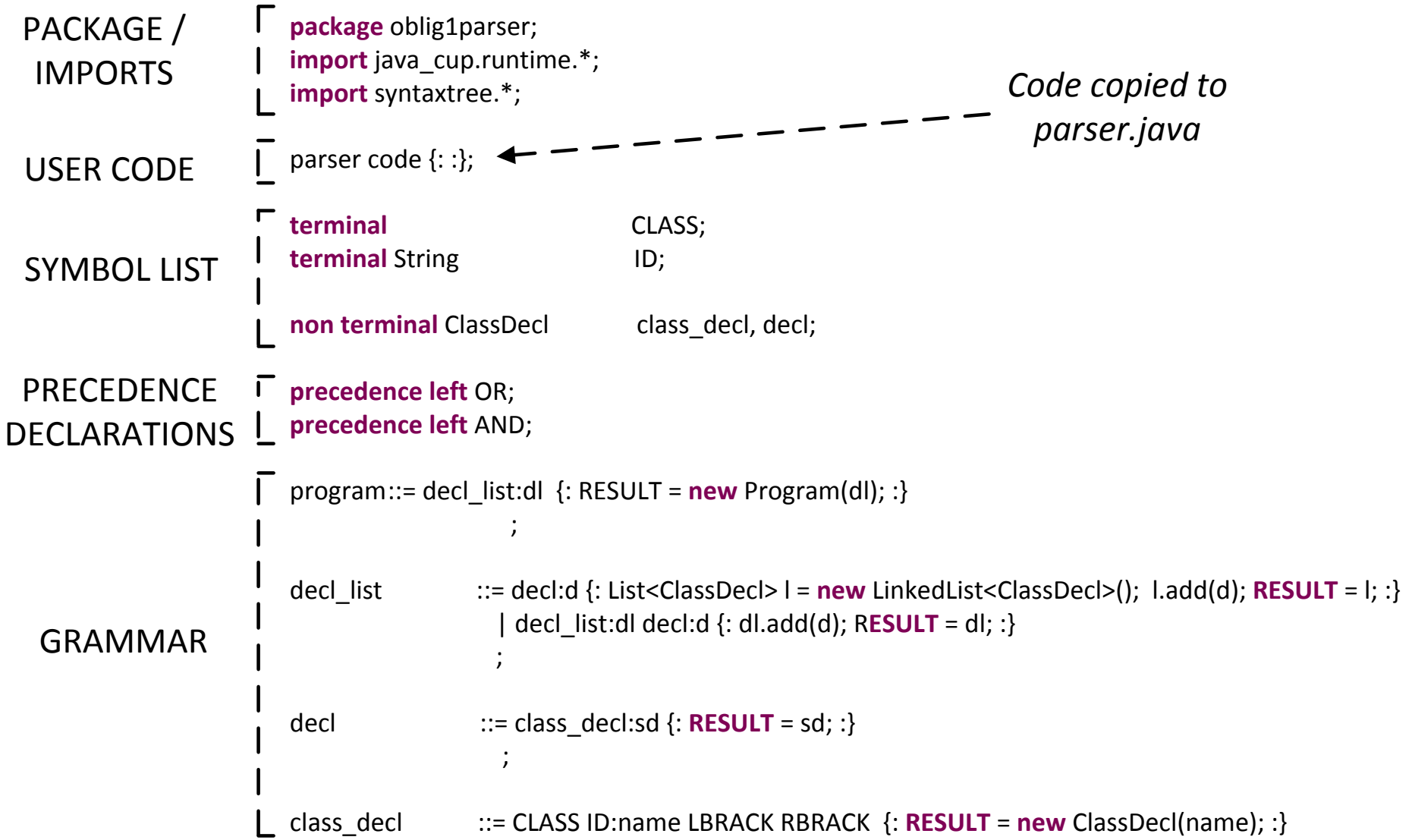
CUP

Java Based Constructor of Useful Parsers (CUP)

- Is specified in .cup file
- Specification consists of five parts:
  - Package and import specifications
  - User code components
  - Symbol list
  - Prcedence declarations
  - Grammar

Scanner and Parser Generators

CUP



Scanner and Parser Generators

CUP

PACKAGE / IMPORTS

```
package oblig1parser;
import java_cup.runtime.*;
import syntaxtree.*;
```

USER CODE

```
parser code {: :};
```

SYMBOL LIST

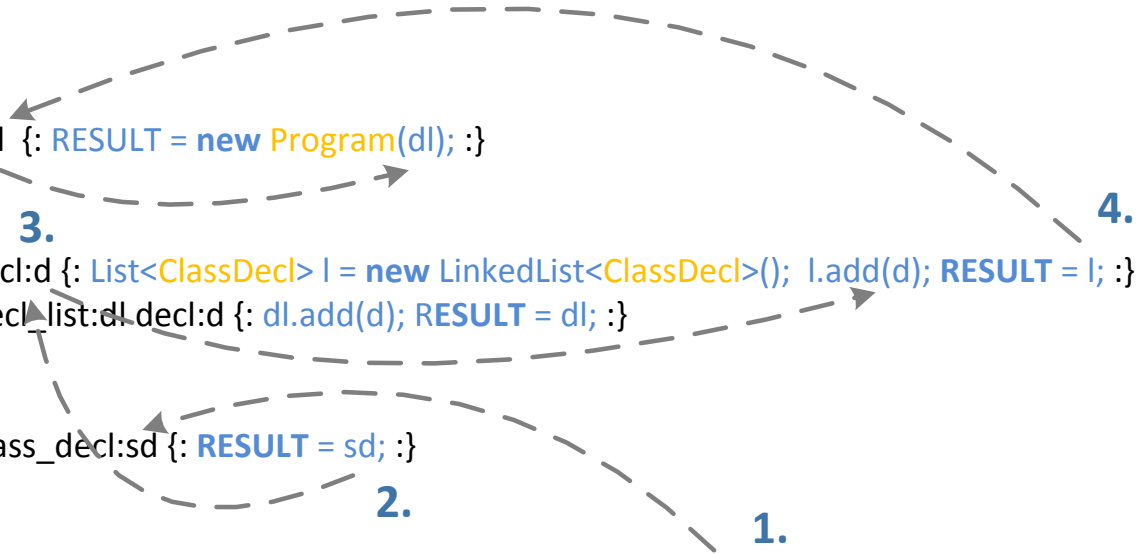
```
terminal                    CLASS;
terminal String             ID;

non terminal ClassDecl      class_decl, decl;
```

PRECEDENCE DECLARATIONS

```
precedence left OR;
precedence left AND;
```

GRAMMAR

```
program::= decl_list:dl  {: RESULT = new Program(dl); :}
           ;

decl_list       ::= decl:d {: List<ClassDecl> l = new LinkedList<ClassDecl>();  l.add(d); RESULT = l; :}
                  | decl_list:dl decl:d {: dl.add(d); RESULT = dl; :}
                  ;

decl            ::= class_decl:sd {: RESULT = sd; :}
                  ;

class_decl      ::= CLASS ID:name LBRACK RBRACK  {: RESULT = new ClassDecl(name); :} ;
```



Scanner and Parser Generators

CUP

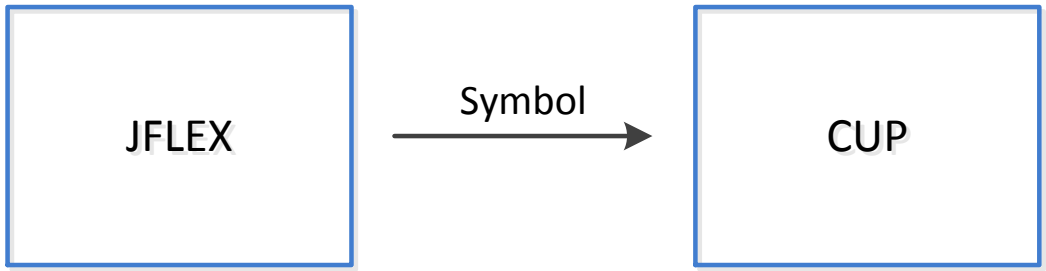
Conflicts

Two types of conflicts may occur:

- Shift-Reduce in which CUP chooses to reduce
- Reduce-Reduce in which CUP chooses first rule

Scanner and Parser Generators

Integration




```
"class" { return symbol(sym.CLASS); }
```

```
public interface sym
{
  public static final int NOT = 36;
  public static final int AND = 21;
  public static final int RPAR = 42;
  public static final int COMMA = 37;
  public static final int CLASS = 2;
  public static final int REFERENCE = 5;
  ...
}
```

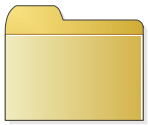
Abstract syntax tree model

Using Java

Each required node in the tree model is represented by a Java class

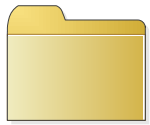
```
package syntaxtree.nonterminals;
import java.util.List;
import utils.Utils;

public class ClassDecl extends Decl {
   private String name;
   private List<VarDecl> varDecls;

public ClassDecl( String name, List<VarDecl> varDecls )  {
   this.name = name;
   this.varDecls = varDecls;
}

public void extractAST( StringBuilder sb, String indent )  {
   sb.append( "\n" + indent + "(CLASS_DECL (NAME " + name + ")" );
      ...
  }
}
```

Overview of provided code



build

Class files for compiler, lexer, parser, syntaxtree, etc.



oblila-code

Oblila source code

grammars

Three pairs of .lex/.cup files

src

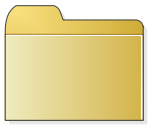
Java source code for compiler, syntax tree, etc.

input-examples

Test file for example parser

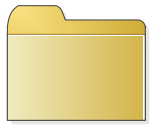
src-examples

Java source code example syntax tree



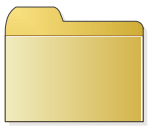
lib

JFlex and CUP libs



src-gen

Generated Java source code for lexer and parser

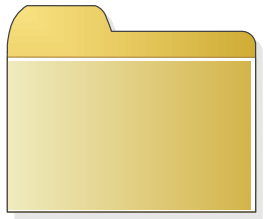


oblila-ast

Generated abstract syntax tree

Overview of provided code

grammars



grammars

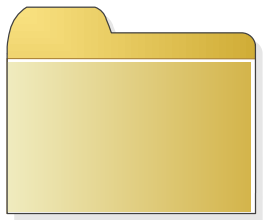
Three pairs of .lex/.cup files

Three grammar pairs

- expression-eval (example expression language)
- expression-par (example language handling parentheses)
- oblig1 (oblig grammars)

Overview of provided code

oblila-ast



oblila-ast

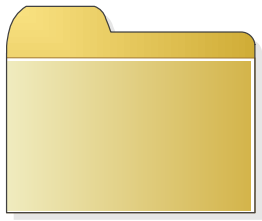
Generated abstract syntax tree

One abstract syntax tree

- Oblila.ast

Overview of provided code

oblila-code



oblila-code

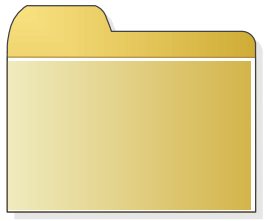
Oblila source code

One Oblila source file

- Oblila.obl

Overview of provided code

src



src

Java source code for compiler, syntax tree, etc.

Two folders

- compiler
- syntaxtree

Using build script

Ant

Ant targets

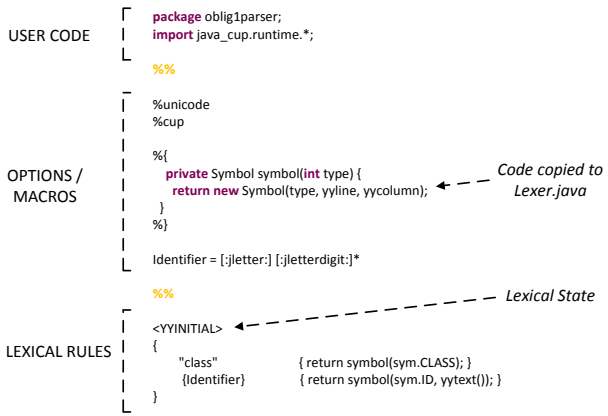
build.xml contains an Ant-based build script with targets and task definitions

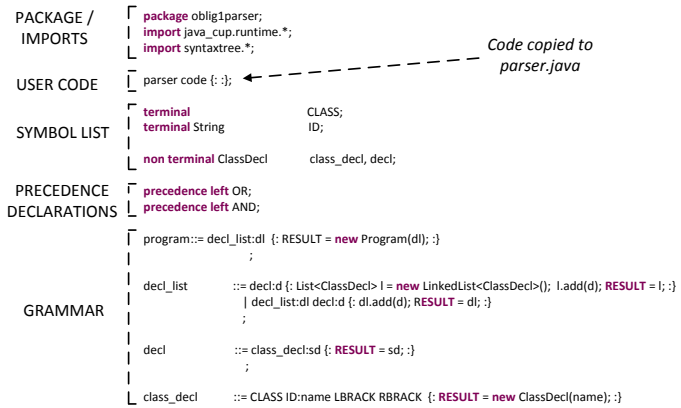
- $ ant clean
- $ ant build
- $ ant run
- ...

The tasks of this assignment

Three main parts

- Create JFlex specification
- Create CUP specification (two versions)
- Define abstract syntax tree model using Java






```
package syntaxtree.nonterminals;
import java.util.List;
import utils.Utils;

public class ClassDecl extends Decl {
  private String name;
  private List<VarDecl> varDecls;

public ClassDecl( String name, List<VarDecl> varDecls ) {
  this.name = name;
  this.varDecls = varDecls;
}

public void extractAST( StringBuilder sb, String indent ) {
  sb.append( "\n" + indent + "(CLASS_DECL (NAME " + name + ")" );
    ...
 }
}
```

Additional information

More information

- Exercise and Oblila descriptions
- JFlex manual
- CUP manual