INF5110
Obligatorisk Oppgave 1
del 2

Andreas Svendsen
SINTEF

26. Februar 2009

Oversikt

- Oppgaven
- Mer om JFlex
- Mer om CUP
- Eksempel

Rettet opp filer

- Canonical.d
- inf5110-oblig1.zip

3

Oppgaven

- Skrive om grammatikken til BNF:
  – Uten metasymbolene { } og [ ].

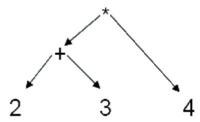
```
PROGRAM    ::= DECL_LIST
           ;
DECL_LIST  ::= DECL
           | DECL DECL_LIST
           ;
DECL       ::= VAR_DECL
           | FUNC_DECL
           | STRUCT_DECL
           ;
```

4

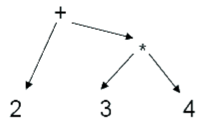
Oppgaven (2)

- Kontrollere assosiativitet og presendens
  - Skrive om grammatikken
  - Funksjonalitet i CUP

EXP -> EXP ARIT_OP EXP



2 + 3 * 4

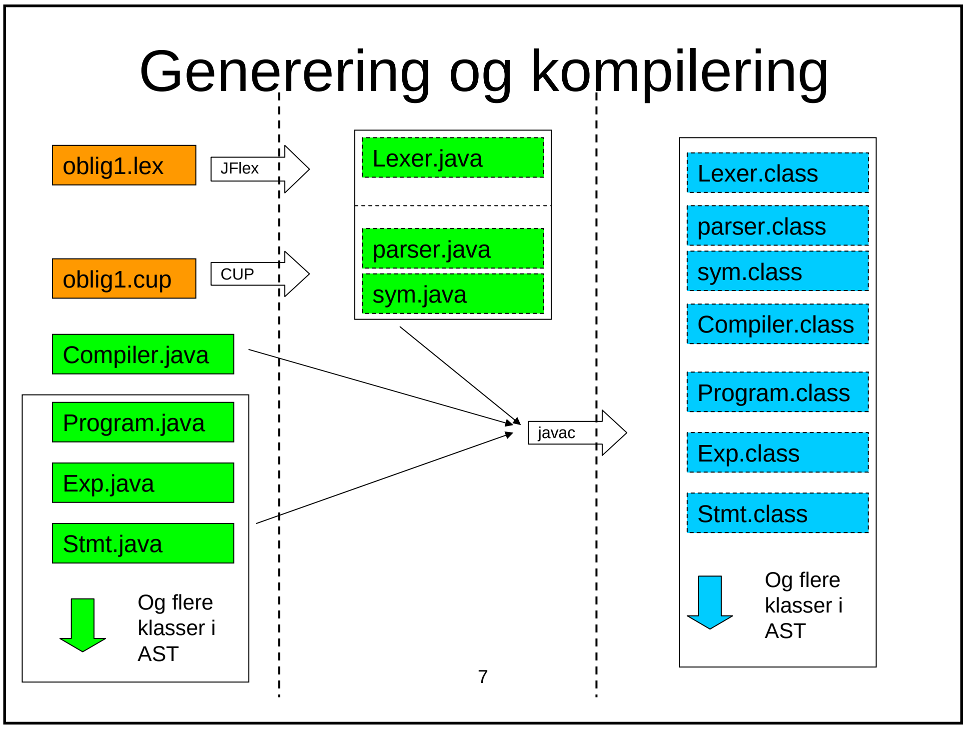


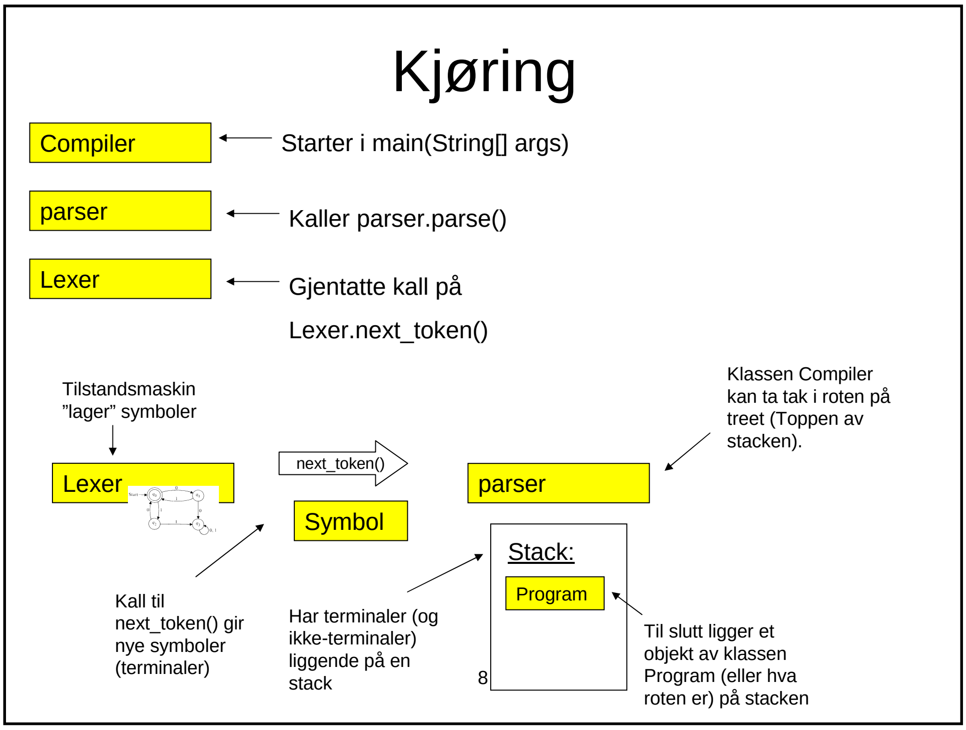
Oppgaven (3)

- Bygge syntakstre og skrive det ut
  - Lage nodene i syntakstreet
    - Program, Struct, Variable, etc.
  - CUP bygger treet

```
package syntaxtree;

public class Program {
        public String printAst(){
                return "(PROGRAM)";
        }
        //metoder for å håndtere semantisk sjekk, kodegenerering osv.
}
```


Generering og kompilering
oblig1.lex
JFlex
oblig1.cup
CUP
Lexer.java
parser.java
sym.java
Compiler.java
Program.java
Exp.java
Stmt.java
Og flere klasser i AST
javac
Lexer.class
parser.class
sym.class
Compiler.class
Program.class
Exp.class
Stmt.class
Og flere klasser i AST
7


Kjøring
Compiler
Starter i main(String[] args)
parser
Kaller parser.parse()
Lexer
Gjentatte kall på Lexer.next_token()
Tilstandsmaskin ”lager” symboler
Lexer
next_token()
Symbol
parser
Klassen Compiler kan ta tak i roten på treet (Toppen av stacken).
Stack:
Program
Kall til next_token() gir nye symboler (terminaler)
Har terminaler (og ikke-terminaler) liggende på en stack
Til slutt ligger et objekt av klassen Program (eller hva roten er) på stacken
8

JFlex

- Regular expressions for å gjenkjenne og bygge tokens

```
FloatLiteral = [0-9]+ \. [0-9]+
```

- Nytt Symbol-objekt for alle tokens:

```
"struct"                { return new Symbol(sym.STRUCT); }
```

9

JFlex (2)

- Konfigurering av JFlex
  - %class ”classname”
  - %implements ”interface1”[,”interface2”]
  - %extends ”classname”
  - %public
  - %{ …
    %}
  - %init{ … // Konstruktør-kode
    %}
  - %eofval{ … // Se CUP/sym
    %eofval}
  - %function ”name” // Navn til skanner-metoden
  - %eofclose // Lukke input-stream ved EOF
  - %debug // Informasjon om tokens
  - %standalone // Skriver ut det som ikke matcher

10

JFlex (3)

- %cup
  - %implements java_cup.runtime.Scanner
  - %function next_token
  - %type java_cup.runtime.Symbol
  - %eofval{
    return new
    java_cup.runtime.Symbol(<CUPSYM>.EOF);
    %eofval}
  - %eofclose

11

JFlex (4)

- %line
  - int yyline
- %column
  - int yycolumn

- Se JFlex-manualen for full oversikt over alle muligheter

12

CUP

- Legger tokens på stacken (shift) helt til vi har en produksjon vi kan redusere til en ikke-terminal
  - Denne legges på stacken
  - Redusere/shifte videre

CUP (2)

- Konfigurering av CUP (runtime-options)
  - -parser *name*
    - Navn på klassen
  - -symbols *name*
  - -expect *number*
    - Må angi antall ”conflicts”
  - -dump
    - dump_grammar
    - dump_states
    - dump_tables

CUP (3)

- Automatisk ”konfliktløsning”
  - shift/reduce
    - Velger skift, men kun et gitt antall (Default:0)
  - reduce/reduce
    - Velger den som står først.

- Se CUP-manualen for mer informasjon

Eksempel

```
EXPR  ::=
      EXPR:e1 PLUS TERM:e2            /*  1  */
       {: RESULT = new Integer(e1.intValue() + e2.intValue()); :}
    | EXPR:e1 MINUS TERM:e2           /*  2  */
       {: RESULT = new Integer(e1.intValue() - e2.intValue()); :}
    | TERM:e                          /*  3  */
       {: RESULT = e; :}
    ;

TERM  ::=
      TERM:e1 MULT FACTOR:e2          /*  4  */
       {: RESULT = new Integer(e1.intValue() * e2.intValue()); :}
    | TERM:e1 DIV FACTOR:e2           /*  5  */
       {: RESULT = new Integer(e1.intValue() / e2.intValue()); :}
    | FACTOR:e                        /*  6  */
       {: RESULT = e; :}
    ;

FACTOR ::=
      LPAREN EXPR:e RPAREN            /*  7  */
       {: RESULT = e; :}
    | NUM:n                           /*  8  */
       {: RESULT = n; :}
    ;
```

Stack	Input	Action
	1 + 1 * 2	Shift 1 symbol
1	+ 1 * 2	Reduce (regel 8)
FACTOR	+ 1 * 2	Reduce (regel 6)
TERM	+ 1 * 2	Reduce (regel 3)
EXPR	+ 1 * 2	Shift 1 symbol
EXPR +	1 * 2	Shift 1 symbol
EXPR + 1	* 2	Reduce (regel 8)
EXPR + FACTOR	* 2	Reduce (regel 6)
EXPR + TERM	* 2	Shift 1 symbol
EXPR + TERM *	2	Shift 1 symbol
EXPR + TERM * 2		Reduce (regel 8)
EXPR + TERM * FACTOR		Reduce (regel 4)
EXPR + TERM		Reduce (regel 1)
EXPR		accept

Oppsummering

- Frist
  - Fredag 20. mars
- Send samlet svar for hele gruppa
- Se oppgaveteksten for detaljer om krav til innlevering
- Husk å levere to grammatikker
- Send spørsmål så snart noe er uklart

17