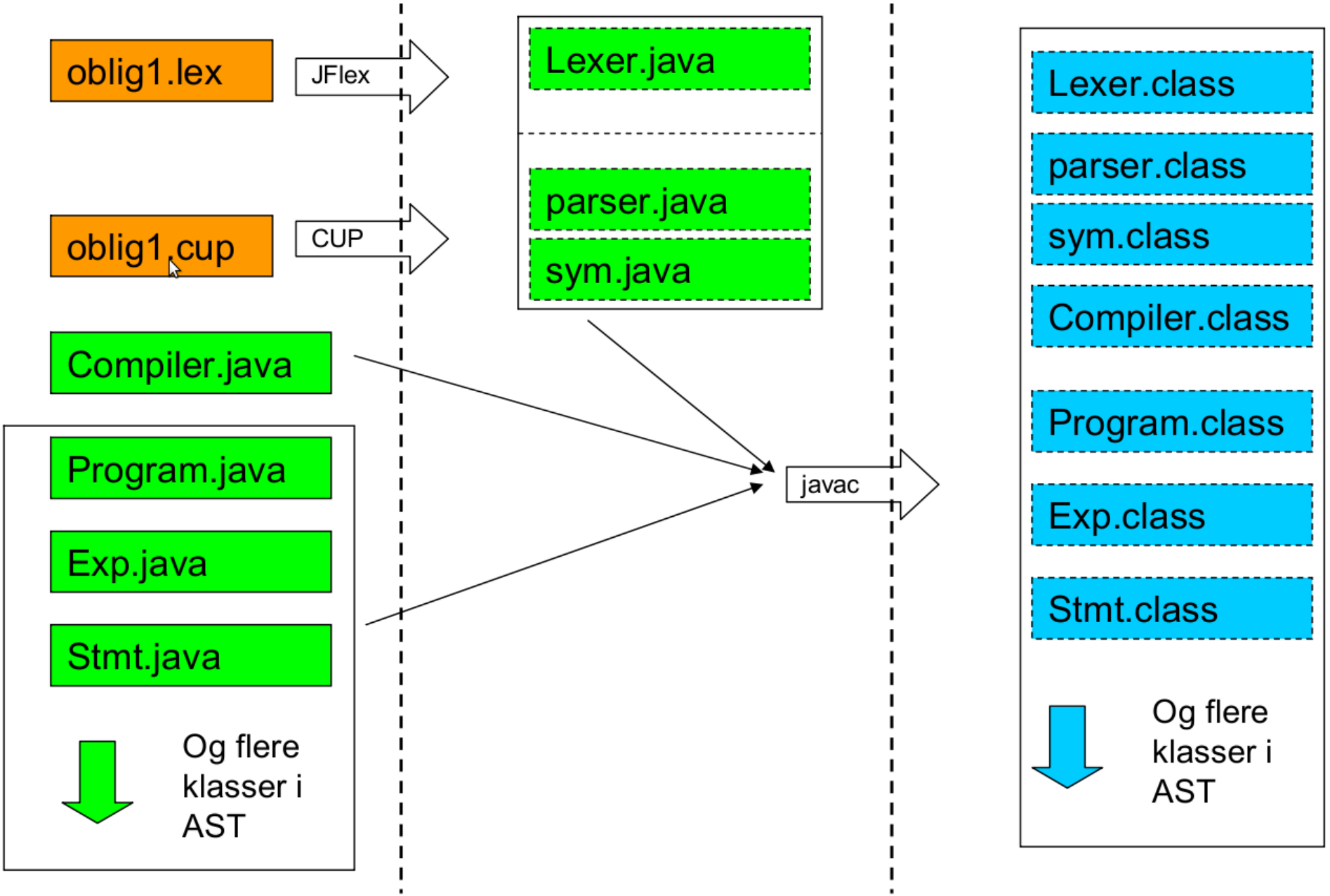
JFlex og Cup

En introduksjon for kurset INF5110
våren 2011

Mål for i dag

- JFlex
  - Regulære uttrykk
  - Cup interaksjon, `sym` klassen
  - Tilstander
  - http://jflex.de/manual.html
- CUP
  - Terminaler og ikke-terminaler
  - Aksjonskode
  - Feilhåndtering
  - http://www2.cs.tum.edu/projects/cup/manual.html
- Se litt på prekoden for Oblig 1

Filer generert av JFLex og CUP



JFlex - bruker en tredelt spesifikasjonsfil

1. User Code:
   - Blir kopiert til den genererte klassen (før klassedefinisjonen).
   - Inneholder typisk `package` og `import` statements.
2. Options and Macros:
   - Options er sett av predefinerte egenskaper som kan inkluderes.
   - Makroer gjør det mulig å gi kompliserte regulære uttrykk navn.
3. Lexical Rules:
   - Inneholder regulære uttrykk med tilhørende aksjonskode.
   - Kan definere forskjellige tilstander som hver har et sett med regulære uttrykk og aksjonskoder.

JFlex - Options

- Blir skrudd på med `%<option_name>`
- Vanlige options er:
  - `%cup` - Integrasjon med CUP
  - `%unicode` - Bruk Unicode
  - `%class <class_name>` - Navnet på generert klasse.
  - `%line` - JFlex vil holde rede på linjenummer og kolonne
  - `%column`
  - `%state <state_name>` - definerer en tilstand
  - `%{ <hjelpekode> %}` - Kopieres inn i klassen.

JFlex - Macros

- Har formen:
  - `macroidentifier = regular expression`
- Vanlig å definere whitespace og identifier som macro:
  - `LineTerminator = \r|\n|\r\n`
  - `WhiteSpace = {LineTerminator} | [ \t\f]`
  - `Identifier = [:jletter:][:jletterdigit:]*`

JFlex - Lexical Specifications

- Bruker regulære uttrykk til å kjenne igjen symboler.
  1. Bruker lengste treff.
  2. Mønster definert først gjelder hvis flere ekvivalent treff.
- Aksjonskoden i blokken etterfulgt av et regulært uttrykk er hva som blir eksekvert om uttrykket matcher.

```
"foo"          {...}
"foobar"       {...}
{Letters}      {...}
```

```
fo          // pattern 3 [of 3]
foo         // pattern 1 [of 1,3]
foobar      // pattern 2 [of 1,2,3]
```

Expression-par eksempel

Demo 1 – Jflex

`grammars/expression-par.lex`

CUP

Generer en LALR parser utifra BNF grammatikk og aksjonskode.

Nodene i parseringstreet til parseren er objekter av egendefinerte klasser.

Parseren vil legge symboler fra Scanneren på stacken helt til en produksjon kan reduseres til en ikke-terminal.

- Aksjonskoden for produksjonen vil da utføres og resultatet av den vil så legges på stacken.
- Redusere/shifte videre

CUP - feilsituasjoner

To vanlige typer feil:

- Shift-Reduce Conflict:
  - I denne situasjonen kan parseren både fortsette å shifte, eller gjøre en reduksjon.
  - CUP feilhåndtering: velger shift.
- Reduce-Reduce Conflict:
  - Kan gjøre to forskjellige reduksjoner på et gitt tidspunkt.
  - Som regel følge av feil med grammatikken.
  - CUP feilhåndtering: velger regelen som er først i filen.
- Mer info:
  - CUP Manualen
  - http://www.gobosoft.com/eiffel/gobo/geyacc/algorithm.html

CUP - Spesifikasjonsfilen

Fem deler:

1. package and import specifications
2. user code components
3. symbol (terminal and non-terminal) lists
4. precedence declarations
5. the grammar

- Del 1 og 2 ganske likt som i JFlex.
- Del 3 er definisjoner over alle terminaler og ikke-terminaler. Her defineres også typen til terminalene.
- Del 4 gjør det mulige å løse presedenskonflikter om dette ikke er gjort entydig gjennom BNF.
- Del 5 er selve grammatikken med aksjonskoden.

Expression-par eksempel

Demo 2 – CUP

`grammars/expression-par.cup`

Apache Ant

- Fleksibelt bygg-verktøy for Java
- build.xml

- http://ant.apache.org/
- http://ant.apache.org/manual/index.html

Apache Subversion

- Sentralisert versjonskontroll
- Gratis hosting på universitetet (med tilgangskontroll)
- Anbefales om dere skal jobbe i team

- http://subversion.apache.org/
- http://svnbook.red-bean.com/en/1.5/index.html
- https://wwws.ifi.uio.no/system/svn
- http://www.uio.no/studier/emner/matnat/ifi/INF5750/h10/undervisningsmateriale/revision_control_and_subversion.pdf