SableCC

Etienne Gagnon
Sable Research Group
McGill University

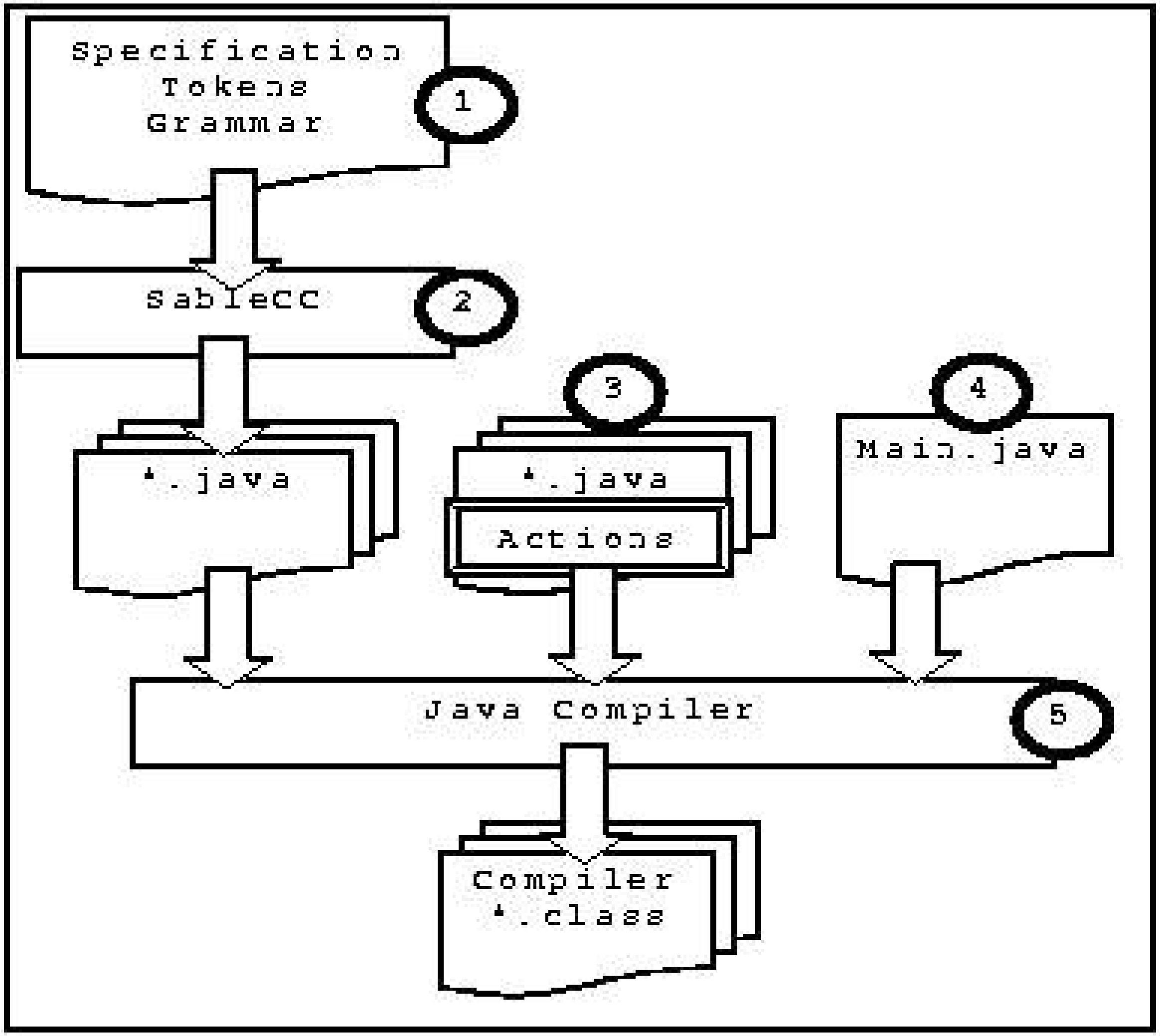
What is SableCC?

- Object-oriented compiler framework written in Java
- Compiler generator classified between front-end compiler generators (Flex/Bison) and end-to-end compiler generators (Amadeus)
- More concretely?

SableCC is an integrated:

- Lexer generator (DFA/Unicode)
- Parser generator (Pure LALR(1))
- Object-oriented framework generator:
  - Strictly-typed Abstract Syntax Tree (AST)
  - Tree-walker classes
  - Uses inheritance to implement actions (e.g. functionality)

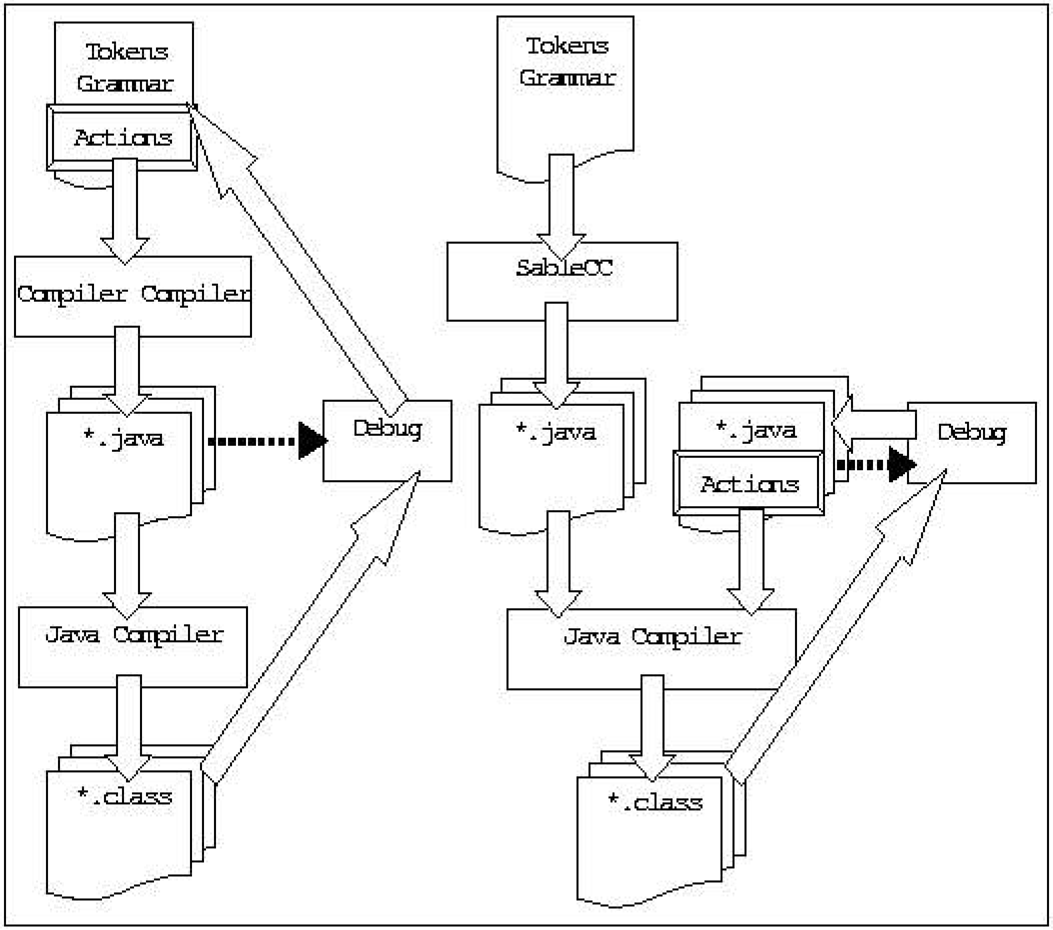
Using SableCC



But, what’s the point?

- Traditional tools have problems
- Amadeus:
  - Unfamiliar language (longer learning curve)
- Flex/Bison (Jlex/Cup):
  - Not integrated
  - No support for building ASTs
  - Action debugging difficult

Jlex/Cup vs SableCC



Advantages of SableCC

- Automatic AST builder for multiple-pass compilers
- Compiler generator out of development cycle when grammar stable
- Easier action debugging
- Access sub-nodes by name, not position
- Clear separation of user and machine generated code
- Automatic AST pretty-printer.

Bison Example

```
... // standard declarations
program: exp  { theexpression = $1;};
exp :  tINTCONST {$$ = makeEXPintconst($1);}
    | exp exp '*' {$$ = makeEXPtimes($1,$2);}
    | exp exp '+' {$$ = makeEXPplus($1,$2);}
...
```

```
EXP *makeEXPintconst(int intconst)
{ EXP *e;
  e = NEW(EXP);
  e->lineno = lineno;
  e->kind = intconstK;
  e->val.intconstE = intconst;
  return e;
}
```

```
EXP *makeEXPtimes(EXP *left, EXP *right)
{ EXP *e;
  e = NEW(EXP);
  e->lineno = lineno;
  e->kind = timesK;
  e->val.timesE.left = left;
  e->val.timesE.right = right;
  return e;
}
```

Bison Example (continued)

```
void prettyEXP(EXP *e)
{ switch (e->kind) {
   case intconstK:
         printf("%i",e->val.intconstE);
         break;
    case timesK:
         prettyEXP(e->val.timesE.left);
         prettyEXP(e->val.timesE.right);
         printf("*");
         break;
...
```

SableCC Example

Tokens

```
lf = 10 | 13 | 10 13;
num = ['0' .. '9']+;
plus = '+';
minus = '-';
mult = '*';
div = '/';
pow = '^';
neg = 'n';
blank = ' '+;
```

Ignored Tokens

```
blank;
```

Productions

```
exp =  {num} num |
       {plus}    [left]:exp [right]:exp plus |
       {minus}   [left]:exp [right]:exp minus |
       {mult}    [left]:exp [right]:exp mult |
       {div}     [left]:exp [right]:exp div |
       {pow}     [left]:exp [right]:exp pow |
       {neg}     exp neg;
```

Pretty-printer

```
System.out.println(ast);
```

SableCC Example (continued)

```
import analysis.*; import node.*;

class Calculator extends DepthFirstAdapter
{
  public void caseTNum (TNum node)
  { setResult(node, Integer.parseInt(node.getText());
  }
  public void outAPlusExp(APlusExp node)
 {setResult(node,
      getResult(node.getLeft()) + getResult(node.getRight()));
 }
…
  public void outStart(Start node)
  {System.out.println(getResult(node.getExp()));
  }
}
```

1998 Wig Projects

- C (Flex/Bison) vs Java (SableCC)
  - Grammar: easier with Bison (precedence)
    - This year, the grammar will be provided.
  - AST/Pretty printer: Much faster with SableCC
  - All Java projects were
    - On time
    - complete with additional features
  - Java source code much shorter (~ 1/5 C code)
  - Java source code more readable.
  - Java code robust

Where can I find more information?

- SableCC web page:
  - http://www.sable.mcgill.ca/sablecc/
- Mailing-list:
  - mailto:sablecc-list-request@sable.mcgill.ca
  - body:subscribe sablecc-list <your-email>
- Your T.A. :-)

SableCC in the world…

- Over 100 mailing-list subscribers from countries around the world
- Listed in the Java FAQ
- Listed on many free Java sites
- Included on the “Just Java 1.2” CD-ROM

Future Development

- Lexers:
  - Difference of regular expressions
  - Parser triggered state transitions
- Parsers:
  - Linear approx. LALR(K)
  - Error Recovery
- Framework
  - Concrete Syntax Tree → Abstract Syntax Tree