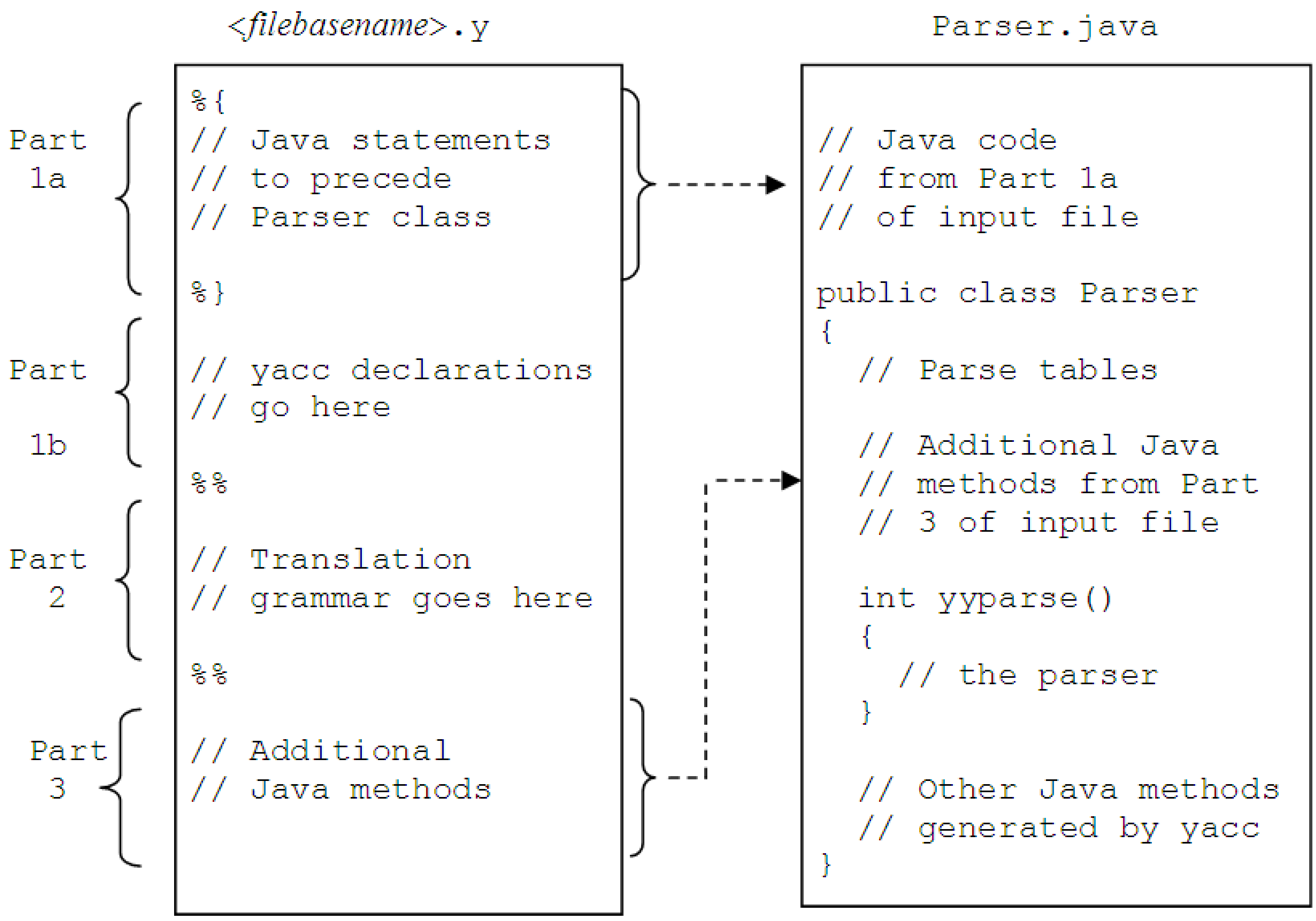
# Chapter 23

Yacc

# Yacc input file



```
%{
// Java statements
// to precede
// Parser class

%}

// yacc declarations
// go here

%%

// Translation
// grammar goes here

%%

// Additional
// Java methods
```

```
// Java code
// from Part 1a
// of input file

public class Parser
{
  // Parse tables

  // Additional Java
  // methods from Part
  // 3 of input file

  int yyparse()
  {
    // the parser
  }

  // Other Java methods
  // generated by yacc
}
```

# Example part 2

```
// Fig2302.y

// no part 1

%%
S : B C   {System.out.println("Prod 1");}
  ;
B : 'b' B {System.out.println("Prod 2");}
  | 'b'   {System.out.println("Prod 3");}
  ;
C : 'c'   {System.out.println("Prod 4 " + yytext);}
  ;
%%
```

## Example part 3

```
// parser expects 0 on end of file
private static final int EOF = 0;
private String input;
private int inputIndex = 0;
private char currentChar;
//------------------------------------------
public static void main(String[] args)
{
  Parser parser = new Parser();
  parser.input = args[0];
  parser.yyparse();    // call yacc-generated parser
}
//------------------------------------------
private int yylex()  // lexical analyzer
{
  if (inputIndex >= input.length())
    return EOF;
  else
  {
    currentChar = input.charAt(inputIndex++);
    yytext = Character.toString(currentChar);
    return currentChar;
  }
}
//------------------------------------------
private void yyerror(String s) // error handler
{
  System.err.println(s);
  System.exit(1);
}
```

## Using Yacc

```
yacc -J Fig2302.y
javac Parser.java
java Parser bbc
```

## Actions don't have to be rightmost

```
S : B
    {System.out.println("hello");}
    C
    {System.out.println("goodbye");}
    ;
B : 'b' B   {System.out.println("Production 2");}
  | 'b'     {System.out.println("Production 3");}
  ;
C : 'c'     {System.out.println("Production 4");}
  ;
```

# Passing values using the value stack

```
// Fig2305.y

%token  UNSIGNED

%%
S     : expr {System.out.println($1.ival);}
      ;
expr :  expr '-' UNSIGNED  {$$.ival = $1.ival-$3.ival;}
      |  UNSIGNED
      ;
```

# ParseVal class

```java
public class ParserVal
{
  public int ival;
  public double dval;
  public String sval;
  public Object obj;
  public ParserVal()
  {
  }
  public ParserVal(int val)
  {
    ival=val;
  }
  public ParserVal(double val)
  {
    dval=val;
  }
  public ParserVal(String val)
  {
    sval=val;
  }
  public ParserVal(Object val)
  {
    obj=val;
  }
}
```

# Using ambiguous grammar

```
// Fig2309.y

%token  UNSIGNED


%%
S     : expr {System.out.println($1.ival);}
      ;
expr : expr '+' expr  {$$.ival = $1.ival + $3.ival;}
     | expr '-' expr  {$$.ival = $1.ival - $3.ival;}
     | expr '*' expr  {$$.ival = $1.ival * $3.ival;}
     | expr '/' expr  {$$.ival = $1.ival / $3.ival;}
     | UNSIGNED
     ;
%%
    // same part 3 as in Fig. 23.5
```

## Disambiguate by inserting in part 1

```
%left '+' '-'
%left '*' '/'
```

## Handling unary minus

```
%left '+' '-'
%left '*' '/'
%right UNARYMINUS
```

```
expr : expr '+' expr  {$$.ival = $1.ival + $3.ival;}
    | expr '-' expr  {$$.ival = $1.ival - $3.ival;}
    | expr '*' expr  {$$.ival = $1.ival * $3.ival;}
    | expr '/' expr  {$$.ival = $1.ival / $3.ival;}
    | '-' expr %prec UNARYMINUS {$$.ival = $2.ival;}
    | UNSIGNED
    ;
```

# Passing values down the parse tree

```
S : B C D
  ;
B : UNSIGNED
  ;
C : UNSIGNED
  ;
D : E
  ;
E : UNSIGNED UNSIGNED {   }
  ;
```

Action here should display the sum of the unsigned integers generated by B, C, and E.

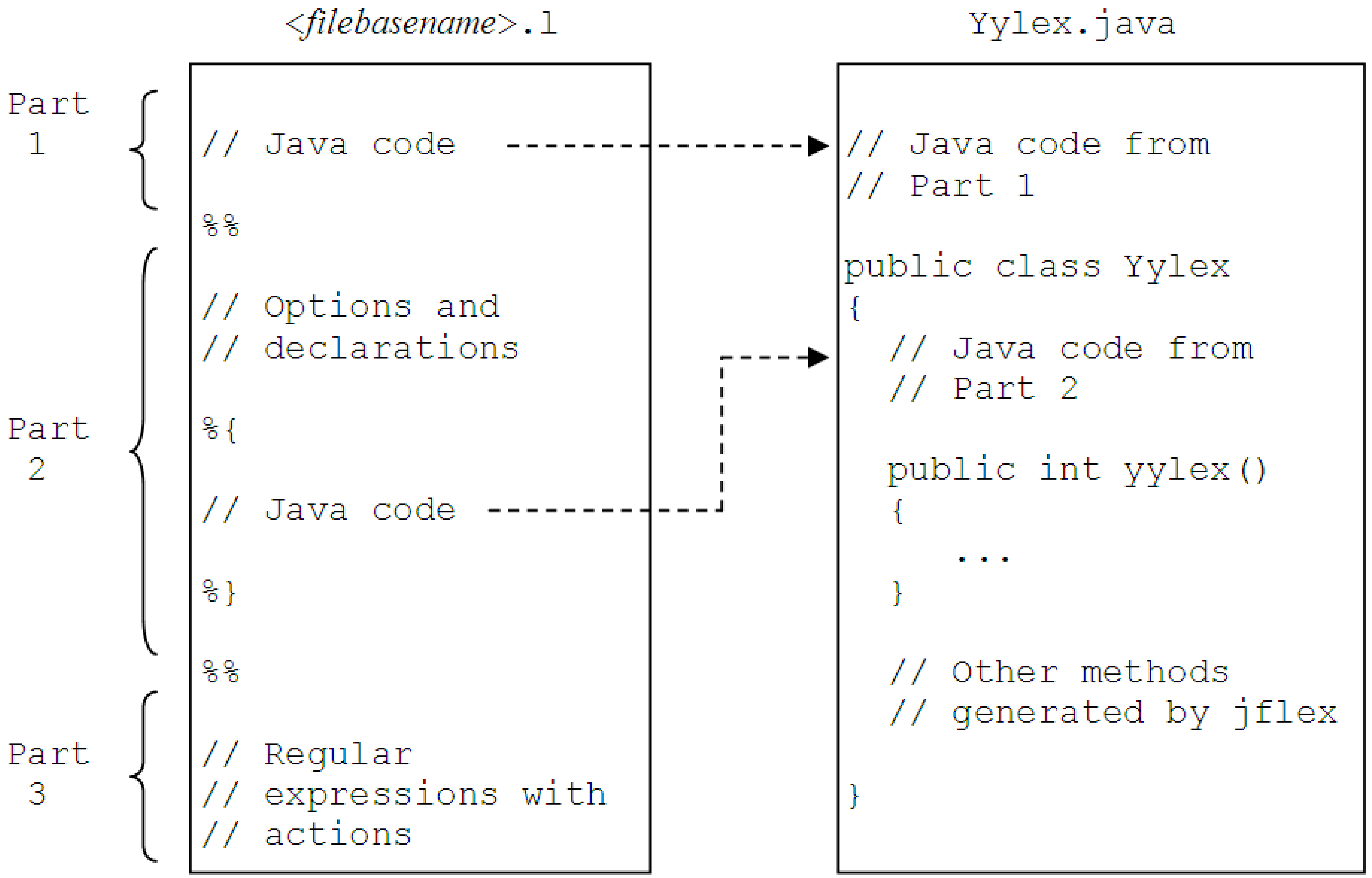
# $0.ival, $-1.ival  below E

```
E  : UNSIGNED UNSIGNED {System.out.println($2.ival +
                              $1.ival + $0.ival + $-1.ival);}
```

## S1y compiler

S1y.txt

# jflex input file



```
// Java code

%%

// Options and
// declarations

%{

// Java code

%}

%%

// Regular
// expressions with
// actions
```

```
// Java code from
// Part 1

public class Yylex
{
  // Java code from
  // Part 2

  public int yylex()
  {
     ...
  }

  // Other methods
  // generated by jflex

}
```

# Regular expressions

| JavaCC | jflex | Meaning |
|---|---|---|
| "b" | b | one b |
| ("b")*` | b* | zero or more b's |
| "b""*" | b"*" or b* | b followed by ordinary asterisk |
| ("b")+` | b+ | one or more b's |
| ("b")?` | b? | optional b |
| "b" "c" | bc | b followed by c |
| ["b', "c"] | [bc] | b or c |
| "b"\|"c" | b\|c | b or c |
| ~["b", "c"] | [^bc] | any character except b or c |
| ["A"-"Z"] | [A-Z] | A through Z |
| "b" "\|" "c" | b "\|" c or b \\| c | b followed by \| and c |
| ["b", "\|", "c"] | [b\|c] | b or \| or c |
| ["-","b"] | [-b] | - or b |
| ~[] | .\|\n | any character |
| ("b"){2,5} | b{2,5} | two to five b's |
|  | ^ | beginning of a line |
|  | $ | end of a line |
|  | . | any character except newline |
|  | b/c | b if followed by c |

## jflex example

```
// Fig2314.l
import java.io.*;

%%

%byaccj   // byacc/j compatibility mode

%{
private int wordCnt = 0;
public static void main(String[] args) throws IOException
{
  FileReader r = new FileReader(args[0]);

  // create lexical analyzer
  Yylex counter = new Yylex(r);

  // call lexical analyzer
  counter.yylex();

  System.out.println("Word count = " + counter.wordCnt);
}
%}

%%

[^ \r\n\t]+  {wordCnt++;}        // match entire line
.|\n         {/* do nothing */} // match any single char
```

## Using jflex

```
jflex Fig2314.l
javac Yylex.java
java Yylex f.txt
```

# Another example

```
// Fig2315.l
import java.io.*;

%%

%byaccj     // byacc/j compatibility mode

%{
private int lineno = 1;
PrintWriter w;
public static void main(String[] args) throws IOException
{
  FileReader r = new FileReader(args[0]);
  PrintWriter w = new PrintWriter(args[1]);
  Ylex numberFile = new Ylex(r);

  // initialize instance variable in numberFile
  numberFile.w = w;

  // call lexical analyzer
  numberFile.yylex();

  w.close();
}
%}

%%

[^\r\n]+    {w.printf("%4d %s%n", lineno++, yytext());}
.|\n        {/* do nothing */}
```

# jflex file for S1

```
// S11.l
%%

%byaccj

%{
private Parser parser;
public Yylex(java.io.Reader inFile, Parser parser)
{
  this(inFile);
  this.parser = parser;
}
%}

ID = [A-Za-z][A-Za-z0-9]*

%%

[ \t\n\r] { /* do nothing */ }  // discard whitespace
println   {
            parser.yylval = new ParserVal(yytext());
            return parser.PRINTLN;
          }
[0-9]+    {
            parser.yylval = new ParserVal(yytext());
            return parser.UNSIGNED;
          }
{ID}      {
            parser.yylval = new ParserVal(yytext());
            return parser.ID;
          }
.         { // <-- period at the start of this line
            parser.yylval = new ParserVal(yytext());
            return yytext().charAt(0);
          }
```

# yacc parser calls yylex

```
private int yylex()
{
  int yyl_return = -1;
  try
  {
    yyl_return = lexer.yylex();
  }
  catch (IOException e)
  {
    System.err.println(e);
  }
  return yyl_return;
}
```