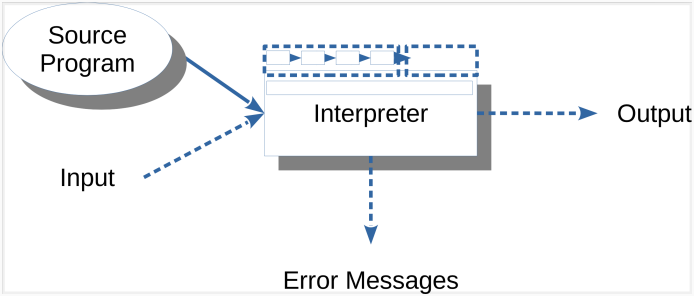
Syntaxgesteuerte Interpreter

Carsten Gips (FH Bielefeld)

Überblick Interpreter



- Syntaxgesteuerte Interpreter
- AST-basierte Interpreter
- Stack-basierte Interpreter
- Register-basierte Interpreter

Syntaxgesteuerte Interpreter: Attributierte Grammatiken

```
s     : expr                    {System.err.println($expr.v);} ;

expr returns [int v]
      : e1=expr '*' e2=expr     {$v = $e1.v * $e2.v;}
      | e1=expr '+' e2=expr     {$v = $e1.v + $e2.v;}
      | DIGIT                   {$v = $DIGIT.int;}
      ;

DIGIT : [0-9] ;
```

Eingebettete Aktionen in ANTLR I

```
rulename[args] returns [retvals] locals [localvars] : ... ;
```

```
add[int x] returns [int r] : '+=' INT {$r = $x + $INT.int;} ;
```

Eingebettete Aktionen in ANTLR II

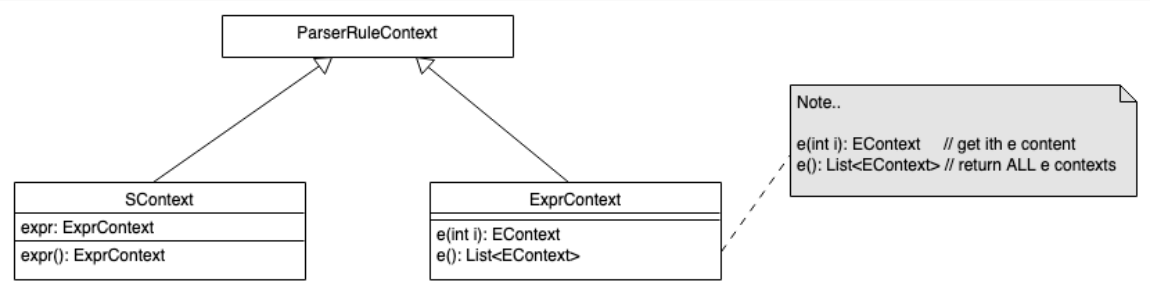
```
@members {
    int count = 0;
}

expr returns [int v]
      @after {System.out.println(count);}
      : e1=expr '*' e2=expr     {$v = $e1.v * $e2.v; count++;}
      | e1=expr '+' e2=expr     {$v = $e1.v + $e2.v; count++;}
      | DIGIT                   {$v = $DIGIT.int;}
      ;

DIGIT : [0-9] ;
```

ANTLR: Kontext-Objekte für Parser-Regeln

```
s    : expr          {List<EContext> x = $expr.ctx.e();} ;
expr : e '*' e ;
```



ANTLR: Arbeiten mit dem Listener-Pattern

```
expr : e1=expr '*' e2=expr      # MULT
     | e1=expr '+' e2=expr      # ADD
     | DIGIT                    # ZAHL
     ;
```

```
public static class MyListener extends calcBaseListener {
    Stack<Integer> stack = new Stack<Integer>();

    public void exitMULT(calcParser.MULTContext ctx) {
        int right = stack.pop();
        int left = stack.pop();
        stack.push(left * right);   // {$v = $e1.v * $e2.v;}
    }
    public void exitADD(calcParser.ADDContext ctx) {
        int right = stack.pop();
        int left = stack.pop();
        stack.push(left + right);   // {$v = $e1.v + $e2.v;}
    }
    public void exitZAHL(calcParser.ZAHLContext ctx) {
        stack.push(Integer.valueOf(ctx.DIGIT().getText()));
    }
```

ANTLR: Arbeiten mit dem Visitor-Pattern

```
expr : e1=expr '*' e2=expr      # MULT
     | e1=expr '+' e2=expr      # ADD
     | DIGIT                    # ZAHL
     ;
```

```
public static class MyVisitor extends calcBaseVisitor<Integer> {
    public Integer visitMULT(calcParser.MULTContext ctx) {
        return visit(ctx.e1) * visit(ctx.e2);   // {$v = $e1.v * $e2.v;}
    }
    public Integer visitADD(calcParser.ADDContext ctx) {
        return visit(ctx.e1) + visit(ctx.e2);   // {$v = $e1.v + $e2.v;}
    }
    public Integer visitZAHL(calcParser.ZAHLContext ctx) {
        return Integer.valueOf(ctx.DIGIT().getText());
    }
}
```

Wrap-Up

- Interpreter simulieren die Programmausführung
- Syntaxgesteuerter Interpreter (attributierte Grammatiken)
- Beispiel ANTLR: Eingebettete Aktionen, Kontextobjekte, Visitors/Listeners (AST-Traversierung)

LICENSE