

```
1 package cymbol.calc;
2
3 import org.antlr.v4.runtime.CharStream;
4 import org.antlr.v4.runtime.CharStreams;
5 import org.antlr.v4.runtime.CommonTokenStream;
6 import org.antlr.v4.runtime.tree.ParseTree;
7 import org.antlr.v4.runtime.tree.ParseTreeWalker;
8
9 import java.io.FileInputStream;
10 import java.io.IOException;
11 import java.io.InputStream;
12 import java.nio.file.Path;
13
14 import cymbol.CymbolLexer;
15 import cymbol.CymbolParser;
16 import cymbol.calc.listener.CalcListenerWithProps;
17 import cymbol.calc.visitor.CalcVisitor;
18
19 public class Calculator {
20     public static void main(String[] args) throws IOException {
21         InputStream is = new FileInputStream(Path.of("src/main/antlr/
cymbol/cymbol-calculator.txt").toFile());
22         CharStream cs = CharStreams.fromStream(is);
23         CymbolLexer lexer = new CymbolLexer(cs);
24         CommonTokenStream tokens = new CommonTokenStream(lexer);
25
26         CymbolParser parser = new CymbolParser(tokens);
27         // use expr() as the root
28         ParseTree tree = parser.expr();
29
30         // System.out.println(tree.toStringTree());
31
32         // for CalcListenerWithProps
33         ParseTreeWalker walker = new ParseTreeWalker();
34         CalcListenerWithProps calcListener = new CalcListenerWithProps();
35         walker.walk(calcListener, tree);
36         System.out.println("Result = " + calcListener.getValues().get(tree
));
37
38         // for CalcVistor
39         CalcVisitor cacLVisitor = new CalcVisitor();
40         // FIXME: NullPointerException
41         // Should override all visitxxx methods.
42         int result = cacLVisitor.visit(tree);
43         System.out.println("Result = " + result);
44     }
45 }
46
```

```
1 package cymbol.calc.visitor;
2
3 import static cymbol.CymbolParser.ADD;
4 import static cymbol.CymbolParser.MUL;
5
6 import cymbol.CymbolBaseVisitor;
7 import cymbol.CymbolParser;
8
9 public class CalcVisitor extends CymbolBaseVisitor<Integer> {
10     @Override
11     public Integer visitNegate(CymbolParser.NegateContext ctx) {
12         return -visit(ctx.expr());
13     }
14
15     @Override
16     public Integer visitMultDiv(CymbolParser.MultDivContext ctx) {
17         int lvalue = visit(ctx.lhs);
18         int rvalue = visit(ctx.rhs);
19
20         return ctx.op.getType() == MUL ?
21             lvalue * rvalue : lvalue / rvalue;
22     }
23
24     @Override
25     public Integer visitAddSub(CymbolParser.AddSubContext ctx) {
26         int lvalue = visit(ctx.lhs);
27         int rvalue = visit(ctx.rhs);
28
29         return ctx.op.getType() == ADD ?
30             lvalue + rvalue : lvalue - rvalue;
31     }
32
33     @Override
34     public Integer visitInt(CymbolParser.IntContext ctx) {
35         return Integer.valueOf(ctx.INT().getText());
36     }
37
38     @Override
39     public Integer visitParens(CymbolParser.ParensContext ctx) {
40         return visit(ctx.expr());
41     }
42 }
```

```
1 package cymbol.calc.listener;
2
3 import static cymbol.CymbolParser.ADD;
4 import static cymbol.CymbolParser.MUL;
5
6 import org.antlr.v4.runtime.tree.ParseTreeProperty;
7
8 import cymbol.CymbolBaseListener;
9 import cymbol.CymbolParser;
10
11 public class CalcListenerWithProps extends CymbolBaseListener {
12     private ParseTreeProperty<Integer> values = new ParseTreeProperty
13         <>();
14
15     public ParseTreeProperty<Integer> getValues() {
16         return values;
17     }
18
19     @Override
20     public void exitNegate(CymbolParser.NegateContext ctx) {
21         values.put(ctx, -values.get(ctx.expr()));
22     }
23
24     @Override
25     public void exitMultDiv(CymbolParser.MultDivContext ctx) {
26         int lvalue = values.get(ctx.lhs);
27         int rvalue = values.get(ctx.rhs);
28
29         if (ctx.op.getType() == MUL) {
30             values.put(ctx, lvalue * rvalue);
31         } else {
32             values.put(ctx, lvalue / rvalue);
33         }
34     }
35
36     @Override
37     public void exitAddSub(CymbolParser.AddSubContext ctx) {
38         int lvalue = values.get(ctx.lhs);
39         int rvalue = values.get(ctx.rhs);
40
41         if (ctx.op.getType() == ADD) {
42             values.put(ctx, lvalue + rvalue);
43         } else {
44             values.put(ctx, lvalue - rvalue);
45         }
46     }
47
48     @Override
49     public void exitInt(CymbolParser.IntContext ctx) {
50         values.put(ctx, Integer.valueOf(ctx.INT().getText()));
51     }
52
53     @Override
```

```
53     public void exitParens(CymbolParser.ParensContext ctx) {  
54         values.put(ctx, values.get(ctx.expr()));  
55     }  
56 }
```