```java
package cymbol.calc;

import org.antlr.v4.runtime.CharStream;
import org.antlr.v4.runtime.CharStreams;
import org.antlr.v4.runtime.CommonTokenStream;
import org.antlr.v4.runtime.tree.ParseTree;
import org.antlr.v4.runtime.tree.ParseTreeWalker;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Path;

import cymbol.CymbolLexer;
import cymbol.CymbolParser;
import cymbol.calc.listener.CalcListenerWithProps;
import cymbol.calc.visitor.CalcVisitor;

public class Calculator {
  public static void main(String[] args) throws IOException {
    InputStream is = new FileInputStream(Path.of("src/main/antlr/cymbol/cymbol-calculator.txt").toFile());
    CharStream cs = CharStreams.fromStream(is);
    CymbolLexer lexer = new CymbolLexer(cs);
    CommonTokenStream tokens = new CommonTokenStream(lexer);

    CymbolParser parser = new CymbolParser(tokens);
    // use expr() as the root
    ParseTree tree = parser.expr();

//    System.out.println(tree.toStringTree());

    // for CalcListenerWithProps
    ParseTreeWalker walker = new ParseTreeWalker();
    CalcListenerWithProps calcListener = new CalcListenerWithProps();
    walker.walk(calcListener, tree);
    System.out.println("Result = " + calcListener.getValues().get(tree));

    // for CalcVistor
    CalcVisitor caclVisitor = new CalcVisitor();
    // FIXME: NullPointerException
    // Should override all visitxxx methods.
    int result = caclVisitor.visit(tree);
    System.out.println("Result = " + result);
  }
}

```

```java
package cymbol.calc.visitor;

import static cymbol.CymbolParser.ADD;
import static cymbol.CymbolParser.MUL;

import cymbol.CymbolBaseVisitor;
import cymbol.CymbolParser;

public class CalcVisitor extends CymbolBaseVisitor<Integer> {
  @Override
  public Integer visitNegate(CymbolParser.NegateContext ctx) {
    return -visit(ctx.expr());
  }

  @Override
  public Integer visitMultDiv(CymbolParser.MultDivContext ctx) {
    int lvalue = visit(ctx.lhs);
    int rvalue = visit(ctx.rhs);

    return ctx.op.getType() == MUL ?
        lvalue * rvalue : lvalue / rvalue;
  }

  @Override
  public Integer visitAddSub(CymbolParser.AddSubContext ctx) {
    int lvalue = visit(ctx.lhs);
    int rvalue = visit(ctx.rhs);

    return ctx.op.getType() == ADD ?
        lvalue + rvalue : lvalue - rvalue;
  }

  @Override
  public Integer visitInt(CymbolParser.IntContext ctx) {
    return Integer.valueOf(ctx.INT().getText());
  }

  @Override
  public Integer visitParens(CymbolParser.ParensContext ctx) {
    return visit(ctx.expr());
  }
}
```

```java
package cymbol.calc.listener;

import static cymbol.CymbolParser.ADD;
import static cymbol.CymbolParser.MUL;

import org.antlr.v4.runtime.tree.ParseTreeProperty;

import cymbol.CymbolBaseListener;
import cymbol.CymbolParser;

public class CalcListenerWithProps extends CymbolBaseListener {
  private ParseTreeProperty<Integer> values = new ParseTreeProperty
<>();

  public ParseTreeProperty<Integer> getValues() {
    return values;
  }

  @Override
  public void exitNegate(CymbolParser.NegateContext ctx) {
    values.put(ctx, -values.get(ctx.expr()));
  }

  @Override
  public void exitMultDiv(CymbolParser.MultDivContext ctx) {
    int lvalue = values.get(ctx.lhs);
    int rvalue = values.get(ctx.rhs);

    if (ctx.op.getType() == MUL) {
      values.put(ctx, lvalue * rvalue);
    } else {
      values.put(ctx, lvalue / rvalue);
    }
  }

  @Override
  public void exitAddSub(CymbolParser.AddSubContext ctx) {
    int lvalue = values.get(ctx.lhs);
    int rvalue = values.get(ctx.rhs);

    if (ctx.op.getType() == ADD) {
      values.put(ctx, lvalue + rvalue);
    } else {
      values.put(ctx, lvalue - rvalue);
    }
  }

  @Override
  public void exitInt(CymbolParser.IntContext ctx) {
    values.put(ctx, Integer.valueOf(ctx.INT().getText()));
  }

  @Override
  public void exitParens(CymbolParser.ParensContext ctx) {
    values.put(ctx, values.get(ctx.expr()));
  }
}
```