```java
package decaf;

import java.io.*;
import java.util.List;
import java.util.Arrays;
import org.antlr.v4.runtime.Token;
import org.antlr.v4.runtime.Parser;
import org.antlr.v4.runtime.ParserRuleContext;
import org.antlr.v4.runtime.ANTLRInputStream;
import org.antlr.v4.runtime.CommonTokenStream;
import org.antlr.v4.runtime.tree.*;
import org.antlr.v4.runtime.misc.Utils;

public class TreePrinterListener implements ParseTreeListener {
    private final List<String> ruleNames;
    private final StringBuilder builder = new StringBuilder();
    private int level = 0;

    public TreePrinterListener(Parser parser) {
        ruleNames = Arrays.asList(parser.getRuleNames());
    }

    public TreePrinterListener(List<String> ruleNames) {
        this.ruleNames = ruleNames;
    }

    @Override
    public void visitTerminal(TerminalNode node) {
        if (builder.length() > 0) {
            builder.append(' ');
        }

        builder.append(Utils.escapeWhitespace(Trees.getNodeText(node, ruleNames)
            , false));
    }

    @Override
    public void visitErrorNode(ErrorNode node) {
        if (builder.length() > 0) {
            builder.append(' ');
        }

        builder.append(Utils.escapeWhitespace(Trees.getNodeText(node, ruleNames)
            , false));
    }

    @Override
    public void enterEveryRule(ParserRuleContext ctx) {
        String indent = "";
        for (int i = 0; i < level; i++)
                indent = indent + " ";

        if (ctx.getChildCount() > 0) {
                builder.append("\n" + indent + "(\n");
            level++;
            indent = indent + " ";
        }
```

```
        builder.append(indent);

        int ruleIndex = ctx.getRuleIndex();
        String ruleName;
        if (ruleIndex >= 0 && ruleIndex < ruleNames.size()) {
            ruleName = ruleNames.get(ruleIndex);
        }
        else {
            ruleName = Integer.toString(ruleIndex);
        }

        builder.append(ruleName);
    }

    @Override
    public void exitEveryRule(ParserRuleContext ctx) {
        if (ctx.getChildCount() > 0) {
            level--;
                builder.append('\n');
                String indent = "";
                for (int i = 0; i < level; i++)
                        indent = indent + " ";

                builder.append(indent);
                builder.append(")\n");
                builder.append(indent);
        }
    }

    @Override
    public String toString() {
        return builder.toString();
    }
}
```