```
 1 package cymbol.callgraph;
 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.Files;
13 import java.nio.file.Path;
14
15 import cymbol.CymbolLexer;
16 import cymbol.CymbolParser;
17
18 public class CallGraph {
19   public static void main(String[] args) throws IOException {
20     InputStream is = new FileInputStream(Path.of("src/main/antlr/
   cymbol/cymbol-functioncall.txt").toFile());
21     CharStream cs = CharStreams.fromStream(is);
22     CymbolLexer lexer = new CymbolLexer(cs);
23     CommonTokenStream tokens = new CommonTokenStream(lexer);
24
25     CymbolParser parser = new CymbolParser(tokens);
26     ParseTree tree = parser.prog();
27
28     ParseTreeWalker walker = new ParseTreeWalker();
29     FunctionCallListener fc = new FunctionCallListener();
30     walker.walk(fc, tree);
31
32     Path fileName = Path.of("src/main/antlr/cymbol/functioncall.dot");
33     Files.writeString(fileName, fc.graph.toDot());
34   }
35 }
36
```

```
 1 package cymbol.callgraph;
 2
 3 import cymbol.CymbolBaseListener;
 4 import cymbol.CymbolParser;
 5
 6 public class FunctionCallListener extends CymbolBaseListener {
 7   Graph graph = new Graph();
 8   String currentFunctionName = null;
 9
10   @Override
11   public void enterFunctionDecl(CymbolParser.FunctionDeclContext ctx
   ) {
12     currentFunctionName = ctx.ID().getText();
13     graph.nodes.add(currentFunctionName);
14   }
15
16   @Override
17   public void enterCall(CymbolParser.CallContext ctx) {
18     String funcName = ctx.ID().getText();
19     graph.edge(currentFunctionName, funcName);
20   }
21 }
22
```

```
 1 package cymbol.callgraph;
 2
 3 import org.antlr.v4.runtime.misc.MultiMap;
 4 import org.antlr.v4.runtime.misc.OrderedHashSet;
 5
 6 import java.util.Set;
 7
 8 public class Graph {
 9   Set<String> nodes = new OrderedHashSet<>();
10   MultiMap<String, String> edges = new MultiMap<>();
11
12   public void edge(String source, String target) {
13     edges.map(source, target);
14   }
15
16   public String toDot() {
17     StringBuilder buf = new StringBuilder();
18
19     buf.append("digraph G {\n")
20         .append("  ranksep = 0.25;\n")
21         .append("  edge [arrowsize = 0.5]\n")
22         .append(" node [shape = circle, fontname = \"ArialNarrow\",
   fontsize = 12, fixedsize = true, height = 0.45];\n");
23
24     nodes.forEach(node -> buf.append(node).append("; "));
25     buf.append("\n");
26
27     edges.getPairs().forEach(edge ->
28         buf.append(" ")
29             .append(edge.a)
30             .append(" -> ")
31             .append(edge.b)
32             .append(";\n"));
33     buf.append("}\n");
34
35     return buf.toString();
36   }
37 }
```