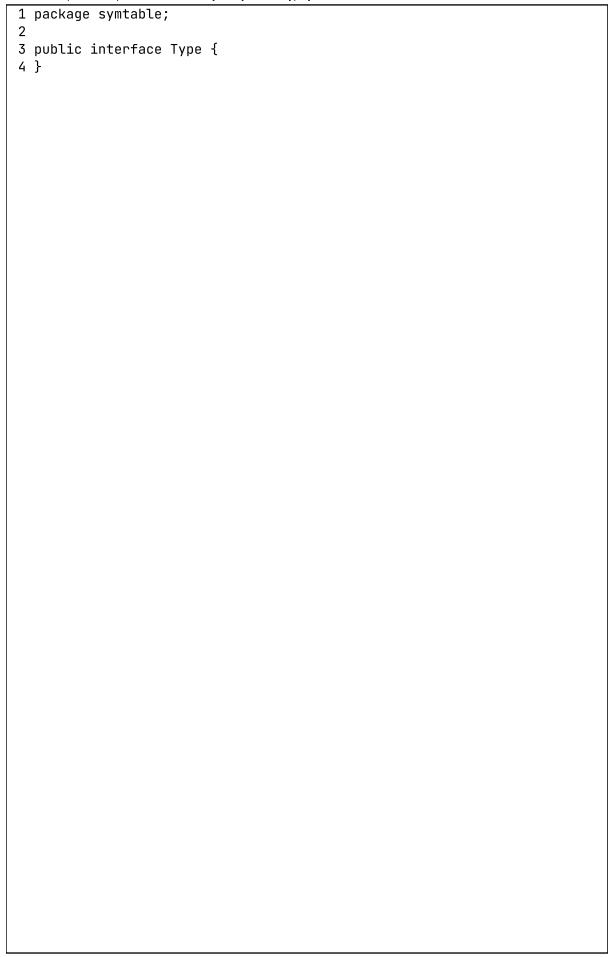
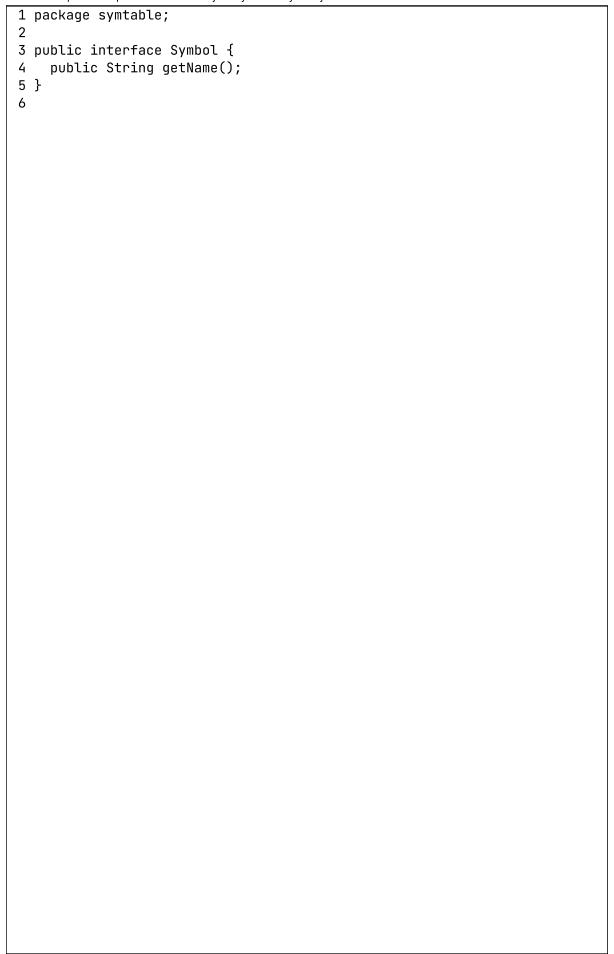
```
1 package symtable;
 3 import com.google.common.base.MoreObjects;
 5 import java.util.LinkedHashMap;
 6 import java.util.Map;
8 public class BaseScope implements Scope {
     private final Scope enclosingScope;
10
     private final Map<String, Symbol> symbols = new LinkedHashMap<>();
11
     private String name;
12
13
     public BaseScope(String name, Scope enclosingScope) {
14
       this.name = name;
15
       this.enclosingScope = enclosingScope;
16
     }
17
18
     @Override
19
     public String getName() {
20
       return this.name;
21
     }
22
23
     @Override
24
     public void setName(String name) {
25
       this.name = name;
26
     }
27
28
     @Override
29
     public Scope getEnclosingScope() {
30
       return this.enclosingScope;
31
     }
32
33
     public Map<String, Symbol> getSymbols() {
34
       return this.symbols;
35
     }
36
37
     @Override
38
     public void define(Symbol symbol) {
39
       symbols.put(symbol.getName(), symbol);
40
       System.out.println("+" + symbol.toString());
41
     }
42
43
     @Override
44
     public Symbol resolve(String name) {
45
       Symbol symbol = symbols.get(name);
46
       if (symbol != null) {
47
         System.out.println("*" + symbol.toString());
48
         return symbol;
49
       }
50
51
       if (enclosingScope != null) {
52
         return enclosingScope.resolve(name);
53
       }
```

```
54
55
       System.err.println("Cannot find " + name);
       return null;
56
57
    }
58
59
    @Override
    public String toString() {
60
61
       return MoreObjects.toStringHelper(this)
           .add("name", name)
62
           .add("symbols", symbols.values().toString())
63
64
           .toString();
65
    }
66 }
67
```

```
1 package symtable;
 3 import java.util.Map;
 5 public interface Scope {
    public String getName();
 7
 8
    public void setName(String name);
 9
10
    public Scope getEnclosingScope();
11
12
    public Map<String, Symbol> getSymbols();
13
14
     public void define(Symbol symbol);
15
16
     public Symbol resolve(String name);
17 }
```





```
1 package symtable;
3 public class VariableSymbol extends BaseSymbol {
    public VariableSymbol(String name, Type type) {
5
      super(name, type);
   }
6
7 }
```

```
1 package symtable;
3 public class BasicTypeSymbol extends BaseSymbol implements Type {
    public BasicTypeSymbol(String name) {
 5
       super(name, null);
    }
 6
 7
 8
   @Override
9
    public String toString() {
10
       return name;
11
12 }
13
```

```
1 package symtable;
3 public class LocalScope extends BaseScope {
    public LocalScope(Scope enclosingScope) {
      super("LocalScope", enclosingScope);
5
    }
6
7 }
8
```

```
1 package symtable;
3 public class GlobalScope extends BaseScope {
    public GlobalScope(Scope enclosingScope) {
      super("GlobalScope", enclosingScope);
5
      define(new BasicTypeSymbol("int"));
6
7
    }
8 }
```

```
1 package symtable;
3 import com.google.common.base.MoreObjects;
5 public class BaseSymbol implements Symbol {
    final String name;
7
    final Type type;
8
9
     public BaseSymbol(String name, Type type) {
10
       this.name = name;
11
       this.type = type;
12
     }
13
14
     public String getName() {
15
     return name;
16
17
18
     public Type getType() {
19
       return type;
20
     }
21
22
     public String toString() {
23
       return MoreObjects.toStringHelper(this)
           .add("name", name)
24
25
           .add("type", type)
           .toString();
26
27
    }
28 }
```

```
1 package symtable;
3 public class FunctionSymbol extends BaseScope implements Symbol {
    public FunctionSymbol(String name, Scope enclosingScope) {
5
      super(name, enclosingScope);
6
    }
7 }
```

```
1 # symtable
 2
3 ## monolithic
 4
5 - Type
 6 - Symbol
7
   - VariableSymbol
8 - BasicTypeSymbol
 9
      - int
10 - Scope
11 - BaseScope
12
     - GlobalScope
13 - DefPhaseListener
14 - enterProg
15 - ~~exitProg~~
16
    - exitVarDecl
17
   - exitID
18
19 ## nested
20
21 - Symbol
22 - FunctionSymbol
23 - Scope
24 - BaseScope
25
      - LocalScope
26 - FunctionSymbol
27 - SymbolTableListener
28 - enterFunctionDecl
29 - exitFunctionDecl
30 - enterBlock
31
      - counter for uniqueness
32
    - exitBlock
    - exitFormalParameter
34 - SymbolTableTreeGraph
35
36 ## forward reference
37
38 ## type checking
```