```java
package symtable;

import com.google.common.base.MoreObjects;

import java.util.LinkedHashMap;
import java.util.Map;

public class BaseScope implements Scope {
  private final Scope enclosingScope;
  private final Map<String, Symbol> symbols = new LinkedHashMap<>();
  private String name;

  public BaseScope(String name, Scope enclosingScope) {
    this.name = name;
    this.enclosingScope = enclosingScope;
  }

  @Override
  public String getName() {
    return this.name;
  }

  @Override
  public void setName(String name) {
    this.name = name;
  }

  @Override
  public Scope getEnclosingScope() {
    return this.enclosingScope;
  }

  public Map<String, Symbol> getSymbols() {
    return this.symbols;
  }

  @Override
  public void define(Symbol symbol) {
    symbols.put(symbol.getName(), symbol);
    System.out.println("+" + symbol.toString());
  }

  @Override
  public Symbol resolve(String name) {
    Symbol symbol = symbols.get(name);
    if (symbol != null) {
      System.out.println("*" + symbol.toString());
      return symbol;
    }

    if (enclosingScope != null) {
      return enclosingScope.resolve(name);
    }
```

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

```java
package symtable;

import java.util.Map;

public interface Scope {
  public String getName();

  public void setName(String name);

  public Scope getEnclosingScope();

  public Map<String, Symbol> getSymbols();

  public void define(Symbol symbol);

  public Symbol resolve(String name);
}
```

```
1 package symtable;
2
3 public interface Type {
4 }
```

```
1 package symtable;
2
3 public interface Symbol {
4   public String getName();
5 }
6
```

File - D:\compilers\compilers-antlr\src\main\java\symtable\VariableSymbol.java

```
1 package symtable;
2
3 public class VariableSymbol extends BaseSymbol {
4   public VariableSymbol(String name, Type type) {
5     super(name, type);
6   }
7 }
8
```

```
 1 package symtable;
 2
 3 public class BasicTypeSymbol extends BaseSymbol implements Type {
 4   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;
2
3 public class LocalScope extends BaseScope {
4   public LocalScope(Scope enclosingScope) {
5     super("LocalScope", enclosingScope);
6   }
7 }
8
```

File - D:\compilers\compilers-antlr\src\main\java\symtable\GlobalScope.java

```
1 package symtable;
2
3 public class GlobalScope extends BaseScope {
4   public GlobalScope(Scope enclosingScope) {
5     super("GlobalScope", enclosingScope);
6     define(new BasicTypeSymbol("int"));
7   }
8 }
9
```

```java
package symtable;

import com.google.common.base.MoreObjects;

public class BaseSymbol implements Symbol {
  final String name;
  final Type type;

  public BaseSymbol(String name, Type type) {
    this.name = name;
    this.type = type;
  }

  public String getName() {
    return name;
  }

  public Type getType() {
    return type;
  }

  public String toString() {
    return MoreObjects.toStringHelper(this)
        .add("name", name)
        .add("type", type)
        .toString();
  }
}
```

File - D:\compilers\compilers-antlr\src\main\java\symtable\FunctionSymbol.java

```
1 package symtable;
2
3 public class FunctionSymbol extends BaseScope implements Symbol {
4   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
33   - exitFormalParameter
34 - SymbolTableTreeGraph
35
36 ## forward reference
37
38 ## type checking
```