

Plan for Today

Attributes

- Inherited
- Synthesized

Calculating line and position for each AST node

Scope

- environment
- static versus dynamic scope
- scope examples
- scoping for MiniJava

Symbol Table

- info maintained and basic operation
- the SymTable class you will be implementing

Calculating Lines and Positions



Example of calculating lines and positions



Scope

Terms

- environment
- scope
- visibility

Example scopes

- global scope
- file scope
- named space
- package
- unnamed scopes

Scoping in MiniJava

Static versus Dynamic Scope

Static Scope

- also called *lexical scope* because can determine scoping by analyzing the program
- each use of a variable is bound to a location statically

Dynamic Scope

- each use of a variable is bound to the most recently visible defined value for that same variable name

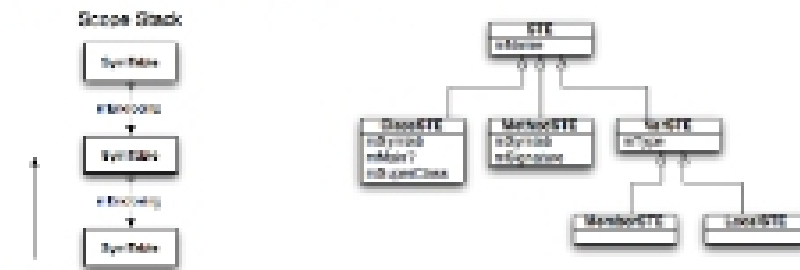
```
int x = 0;
int f() { return x; }
int g() { int x = 1; return f(); }
```

CS453 Lecture

Symbol Table

3

SymTable and STE classes



SymTable interface (DO NOT change the SymTable members or interface)

- SymTable pushClassScope(String classname)
- SymTable pushMethodScope(String methodname)
- SymTable popScope()
- STE lookup(String sym)
- void insert(STE ste)
- int outputDot(java.io.PrintStream out, int nodeCount)

CS453 Lecture

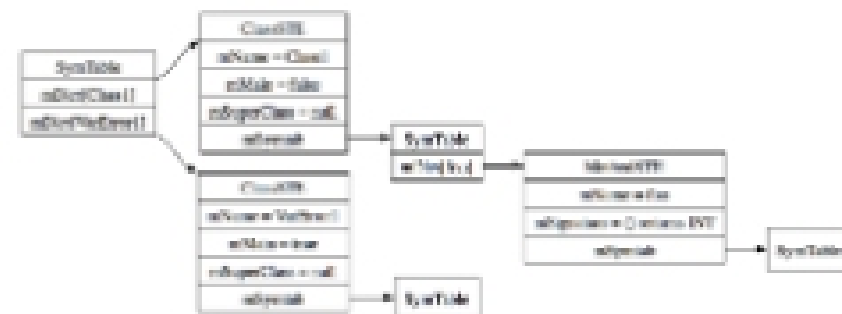
Symbol Table

4

Example SymTable dot output

```
class VarError1{
    public static void main(String[] a){ System.out.println(4); }
}

class Class1 {
    public int foo() {
        if (a) {} else {}
        return 8;
    }
}
```



CS453 Lecture

Symbol Table

5

Using the SymTable interface

