Plan for Today

More PA5 Details

PA6 Overview
  – Goals
  – Grammar
  – Debugging grammar teaser

Symbol Table
  – “pass info from declarations to uses”

Semantic analysis for ...
  – variables
  – integer and byte expressions
  – assignment statements
  – method calls

PA6 Overview

Goals
  – Code generation for objects and assignment statements
  – Create a symbol table that is used by later passes/visitors in the compiler
  – Perform semantic analysis to find ALL redeclared variables and to report the first type error.

New pieces of grammar
  – Variable declarations
  – Assignment statements

Will probably need to debug the grammar
Symbol Table

Information maintained in a symbol table, which is a kind of environment
– For each identifier: type, scope (includes lifetime), visibility, and run-time location
– For named scopes, the set of identifiers it contains.
– While processing the program, the current

Example scopes
– global scope
– file scope
– named space
– package
– unnamed scopes

Scoping in MeggyJava

SymTable and STE classes

Suggested SymTable interface
– STE lookup(String sym)
– void insert(STE ste)
– Type getExpType(node)
– int outputDot(java.io.PrintStream out, …)
Using the SymTable interface

Error message for symbols redeclared within same scope

```
Class ID ... ClassDecl
public Type ID ... MethodDecl
Type ID; VarDecl
[LINENUM,POSNUM] Redefined symbol VARNAME
// different in that ALL of these must be printed
```
SymTable, Scope, and STE classes

More sophisticated type representation
### Implementing type checking for PA6 MeggyJava

Visitor over AST will check for type errors at each AST node

<table>
<thead>
<tr>
<th>Syntax</th>
<th>AST node</th>
</tr>
</thead>
<tbody>
<tr>
<td>id = Exp ;</td>
<td>AssignStatement(id, Exp)</td>
</tr>
<tr>
<td></td>
<td>[LINENUM,POSNUM] Undeclared variable VARNAME</td>
</tr>
<tr>
<td></td>
<td>[LINENUM,POSNUM] Invalid expression type assigned to variable VARNAME</td>
</tr>
<tr>
<td>public Type name(...) { ...return Exp; }</td>
<td>MethodDecl(name, Stms, Exp)</td>
</tr>
<tr>
<td></td>
<td>[LINENUM,POSNUM] Invalid type returned from method METHODNAME</td>
</tr>
<tr>
<td>Exp . name ( Args )</td>
<td>CallExp(name, Args)</td>
</tr>
<tr>
<td></td>
<td>[LINENUM,POSNUM] Receiver of method call must be a class type</td>
</tr>
<tr>
<td></td>
<td>[LINENUM,POSNUM] Method METHODNAME does not exist</td>
</tr>
<tr>
<td></td>
<td>[LINENUM,POSNUM] Method METHODNAME requires exactly NUM arguments</td>
</tr>
<tr>
<td></td>
<td>[LINENUM,POSNUM] Invalid argument type for method METHODNAME</td>
</tr>
</tbody>
</table>