Compiler Design

HW3
Code Presentation

AbstractTeamFactory
Niko & Ary

Overview

3 Visitors:

- AstEnricher
- AstTypeChecker
- AstSemanticChecker

AstEnricher

```
class AstEnricher extends
          AstVisitor<Symbol, Map<String,Symbol.VariableSymbol>> {…}
```

- Returns a `Symbol`
- Arg is the symbol table for the enclosing scope (either class or method)
- Goes through the AST and sets the symbols of the AST nodes

AstEnricher

In SemanticAnalyzer:

```
// Populate symbol table for all top-level classes.
AstEnricher astEnricher = new AstEnricher();
classDecls.stream().forEach(classDecl -> {
    Symbol.ClassSymbol clsSymbol = (Symbol.ClassSymbol)
astEnricher.visit(classDecl, null);

    if (globalSymbols.containsKey(clsSymbol.name)) {
        String errorFmt = "Found two classes named %s.";
        throw new
SemanticFailure(SemanticFailure.Cause.DOUBLE_DECLARATION,
errorFmt, clsSymbol.name);
    } else {
        globalSymbols.put(clsSymbol.name, clsSymbol);
    }
});
```

AstTypeChecker

```
class AstTypeChecker extends
                         AstVisitor <Symbol.TypeSymbol,Symbol> {…}
```

- Returns a `TypeSymbol`, to simplify recursive visit() calls
- Arg is a `Symbol` so the enclosing `MethodSymbol` can be accessed in order to check for the correct return type
- Goes through the AST and ensures type safety.

AstTypeChecker

In SemanticAnalyzer:

```
// Run type checks.
AstTypeChecker astTypeChecker = new AstTypeChecker(globalSymbols);

classDecls.stream()
          .forEach(
                   classDecl -> astTypeChecker.visit(classDecl, null)
                  );
```

AstSemanticChecker

```
class AstSemanticChecker extends
                  AstVisitor<Void,Symbol> {…}
```

- Returns nothing really
- Arg is a **`Symbol`** so the enclosing scope can be accessed
- Goes through the AST and checks for semantic error which require a type-checked AST

AstSemanticChecker

In SemanticAnalyzer:

```
// Run all other global semantic checks.
AstSemanticChecker astSemanticChecker = new
AstSemanticChecker(globalSymbols);

classDecls.stream()
          .forEach(
                   classDecl -> astSemanticChecker.visit(classDecl, null)
                  );
```