

Plan for Today

MiniJava types and type rules

- representation and interpretation will be discussed while doing IRT generation
- how type information is represented with SymTable and Type data structures
- what type errors can occur in some of the AST nodes

Type implementation in the MiniJava compiler

```
public class Type {
    public static final Type ARRAY = new Type();

    public static final Type BOOL = new Type();

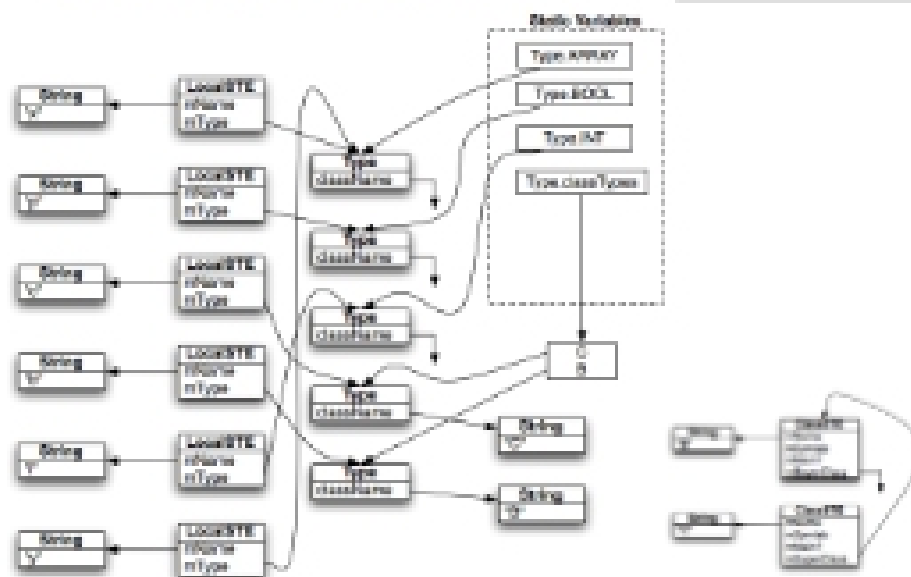
    public static final Type INT = new Type();

    // class type map (key: class name, value: type)
    private static final HashMap<String, Type> classTypes
        = new HashMap<String, Type>();
}
```

Only one instance of the type object per atomic type and class type

- to determine if types are equal just compare references
- does the Type class know about inheritance?

MiniJava Types for Example



Implementing type checking for MiniJava (Slide 1)

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

Syntax	AST production	AST node
<code>id = Exp ;</code>	<code>statement = {assign} id exp</code>	[LINENUM, POSNUM] Undeclared variable VARNAME [LINENUM, POSNUM] Invalid expression type assigned to variable VARNAME
<code>id [Exp] = Exp ;</code>	<code>statement = {array_assign} id [index]:exp exp</code>	[LINENUM, POSNUM] Undeclared variable VARNAME [LINENUM, POSNUM] Array reference to non-array type [LINENUM, POSNUM] Invalid index expression type for array reference [LINENUM, POSNUM] Invalid expression type assigned into array
<code>Exp op Exp</code>	<code>exp = {op} [l_exp]:exp [r_exp]:exp</code>	[LINENUM, POSNUM] Invalid left operand type for operator OP [LINENUM, POSNUM] Invalid right operand type for operator OP

Implementing type checking for MiniJava (Slide 2)

Syntax	AST production	AST node
Errors ! Exp	exp = {not} exp	[LINEUM, POSNUM] Invalid operand type for operator !
new int [Exp]	exp = {new_array} exp	[LINEUM, POSNUM] Invalid operand type for new array operator
Exp [Exp]	exp = {array} exp [index]:exp	[LINEUM, POSNUM] Array reference to non-array type [LINEUM, POSNUM] Invalid index expression type for array reference
Exp . length	exp = {length} exp	[LINEUM, POSNUM] Operator length called on non-array type

CS453 Lecture

Semantic Analysis

5

Implementing type checking for MiniJava (Slide 3)

Syntax	AST production	AST node
Errors new id ()	exp = {new} id	[LINEUM, POSNUM] Class CLASSNAME does not exist
Exp . id (ExpList)	exp = {call} exp id [args]:exp*	[LINEUM, POSNUM] Receiver of method call must be a class type [LINEUM, POSNUM] Method METHODNAME does not exist in class type CLASSNAME [LINEUM, POSNUM] Method METHODNAME requires exactly NLM arguments [LINEUM, POSNUM] Invalid argument type for method METHODNAME

CS453 Lecture

Semantic Analysis

6

Implementation Plan

Test-driven approach

- Write test cases for
 - one AST node at a time
 - one type check at a time
 - one possible type at a time (start with atomic types)
- Set up a regression testing script
 - capture your compiler output on test case to a temp file
 - compare output to a handwritten output for test case
- Implement
 - one AST node at a time
 - one type check at a time
 - one possible type at a time (start with atomic types)

Advantages

- turn in your program at any point to get partial credit
- separate two most difficult pieces: understanding MiniJava typing and implementing the typecheck with the provided data structures

CS453 Lecture

Semantic Analysis

7