Lecture 18:

- Semantic analysis (cont)

Announcements:

- HW-4 out
- Proj. Part 1 due Fri

Semantic Checker

Check more complex statements and expressions (from last time ...)

```
public void visit(BasicExpr node) {
  node.rvalue.accept(this);
}

public void visit(UnaryExpr node) {
  node.expr.accept(this);
  if (node.unaryOp.lexeme.equals("not"))  // not is the only choice
    if (!currType.type.lexeme.equals("bool") || currType.isArray)
      error("expecting boolean expression", currType.type);
}

public void visit(BinaryExpr node) {
  // set up the lhs
  node.lhs.accept(this);
  DataType lhsType = currType;
  // set up the rhs
  node.rhs.accept(this);
  DataType rhsType = currType;
  // get the operator
  String op = node.binaryOp.lexeme;
  // check the cases ...
}
```

Type Inference Rules

Purpose

- like grammar rules, give rules for infering types
- the "legal" inferences (from which implies type errors)
- not all semantic errors captured (e.g., shadowing, use-before-def, etc.)

Basics

- "$e : t$" states that expression e has type t ... e.g., `42 : int`
- Γ denotes the typing context (the entire environment)
- $\vdash$ stands for "implies"
- $\Gamma \vdash e : t$ means it is implied from the given typing context that e has type t

Type Inference Rules

An example typing rule (not from `MyPL`) ...

$$\frac{\Gamma \vdash e_1 : t \quad \Gamma \vdash e_2 : t}{\Gamma \vdash e_1 + e_2 : t}$$

"If expressions e_1 and e_2 have type t in the current context, then expression $e_1 + e_2$ has the type t as well"

- typing rules allow us to infer the types of complex expressions
- which help us to assign types to (var) names
- and (indirectly) type check statements

Check In: Look over the typing rules and find 2 rules that are confusing to you