Lecture 17:

- Semantic Analysis (intro)

Announcements:

- HW-3 out
- Quiz 4 on Friday (AST creation)

Intro to Semantic Analysis – Important Terminology

Denotable Objects

- Items that can be "**named**" in a programming language
- By the programmer (e.g., variables, functions, classes)
- By the language itself (e.g., primitive types, built-in functions)

Blocks

- A block is a **textual region** of a program (e.g., function body, loop body)
- A block uses syntax to define start and end
- Declarations (e.g., of user-defined denotable objects) occur within "blocks"

Bindings

- The association between between names and objects
- Type bindings connect names to their types
- Location bindings connect names to their locations in memory
- Value bindings connect names to their corresponding values

Environments (aka **Contexts**)

- The current set of bindings of a program, statement, expression
- Typing environments give names and their types "visible" at a program location
- Similarly with locations and values

Scope Rules (aka "**visibility**" rules)

- Define what names are visible in which blocks
- An object is **local** to the block it is declared in
- In general, an object is visible in its local and nested blocks
- To find the declaration, look in the current block and the containing blocks

Static vs Dynamic

- Static generally implies decisions made at **compile time** (before runtime)
- Dynamic generally implies decisions made at **runtime**

Static Scope (aka "**lexical**" scope)

- The visibility of names determined at compile time
- Based on the text of the source code
- What we normally think of as scope (visibility)

Dynamic Scope

- The visibility of names determined at runtime
- Based on last association created for the name

Most (modern) PLs primarily adopt static scoping rules

- some tricky cases though ...
- e.g., with nested functions, passing code blocks to functions, closures

The goal of static analysis is to:

- Detect errors due to type issues, e.g.:

```
x = 0 + "1";            // int + string not allowed
if (42 <= true) {       // int <= bool not allowed
  x = 1;
}
```

- Detect "use before def" errors, e.g.:

```
int x = 42 + y;         // y not defined
if (x > 42) {
   int y = x + 1;
}
else {
   x = y;               // y not defined in this block
}
```

- Detect function call errors, e.g.:

```
int add(int x, int y) {
  return x + y;
}
void main() {
  int r1 = add(1, 2, 3);  // wrong number of args
  int r2 = add(3.14, 1);  // wrong argument types
  bool r3 = add(1, 2);    // wrong return type
}
```

- plus more ...

Summary – Things to Know

1. Definitions of Denotable Objects, Blocks, Bindings, Environments, Scope Rules, Static vs Dynamic, Static Scope, and Dynamic Scope

2. Basic examples of use-before-def and type errors in MyPL