1. What does it mean for an attribute to be *statically* bound (emphasis on the word **statically**)?

For an attribute to be statically bound means that the binding time occurs when the program is compiled (before it is executed). In terms of storage, the memory address would be bound before program execution (during compile time) and would remain unchanged (or static) throughout the lifetime of the program.

1. Explain what an implicit declaration is. Additionally, give an example, and state whether it is static or dynamic.

Implicit declaration refers to the act of inferring type by surrounding context. This is as opposed to explicit declaration, where it would be explicitly stated in source code what the type of a variable is.

Example (C++):

int x = 55;

auto y = x; 🡨 the data type of y is implicitly declared to be of type *int* since **x** is an int.

This is done statically (or at compile time) as the observation can be made simply by examining

the source code.

1. What is the difference between **Scope** and **Lifetime** of a variable or identifier?

Scope denotes where, in a program, a particular variable can be referenced, vs Lifetime, which denotes when a particular variable is bound to a memory cell or address.

Example (not required):

int scope\_same\_as\_lifetime\_example() {

int x = 10; 🡨 scope is this block. Lifetime also begins and ends in this block

}

int scope\_not\_equal\_lifetime\_example() {

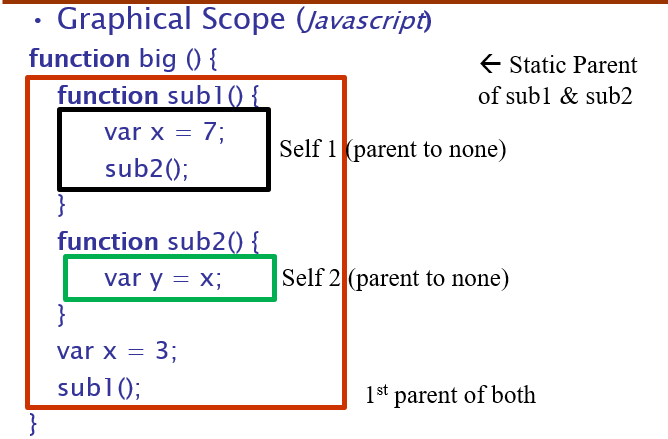
static int x = 10; 🡨 scope is this block, lifetime is beginning and end of program execution (or the entire lifetime of the program)

}

1. What is static scoping? Additionally, give an example.

Static scoping is the idea that scopes are tied to blocks of code. It is a spatial type of scoping where the scope is determined by where blocks of code appear in a file.

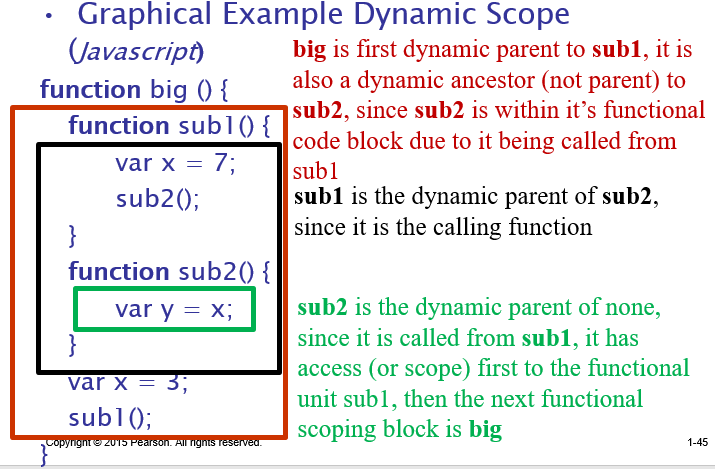
Example:



1. What is dynamic scoping? Additionally, give an example.

Dynamic scoping is the idea that scopes are tied to functional units of code. It is a temporal type of scoping where the scope is determined by the order of functions that are called, or more specifically, what function(s) another function calls.

Example:



1. What is the referencing environment for the block of code containing the variable ***myf2\_var*** in bold below using static scoping?

Class RefEnvironment{

private int x = 10;

🡨 Scope from static parent

private int y = 20;

public RefEnvironment(int x, int y){

this.x = x;

this.y = y;

}

public void myf1(){

int myf1\_var = 10;

int result = myf1\_var + 10;

myf2();

}

public void myf2(){

int myf2\_var = 20;

🡨 Immediate Scope

int result = myf2\_var + 20;

}

}

Referencing environment of block of code within function myf2():

myf2\_var: read/write access

result: read/write access

x: read/write access

y: read/write access