

**De La Salle University • College of Computer Studies**

**Variables**

Name (last name first) : Fernandez, Ryan Austin

Poblete, Clarisse Felicia M.

San Pedro, Marc Dominic

Tan, Johansson E.

Section : G01

Date of Submission : March 3, 2016

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Type Binding** | **Scope** | **Address** | **Lifetime** |
| **C++** | Explicit  Static | Static scoping | * The memory of a program is divided into different areas called segments including the heap and the call stack. * The heap is where dynamically allocated variables are allocated from * The call stack is where function parameters, local variables, and other function-related information are stored | * Function scope variables die when the function dies. * Global variables last until the lifetime of the document. |
| **C#** | Implicit  Static | Static scoping | * Local variables (function scope and block scope) and function parameters are stored on the stack. * Objects referenced by reference type variables are stored in the heap, but the references themselves are on the stack. * Static variables are stored in the heap. | * Function scope variables die when the function dies. * Global variables last until the lifetime of the document. * Block scope variables (if, for, while, etc) dies when the block ends. * Static variables are loaded the first time they are accessed and remain in memory until the app dies.1 |
| **Python** | Implicit  Static | Static scoping | * Objects and data structures are stored in a private heap * Memory management is done by the interpreter and cannot be controlled by the user | * Function scope variables die when the function dies. * Global variables last until the lifetime of the document. |
| **Javascript** | Implicit  Dynamic | Static scoping | * Most variables have a static address. * Variables declared in functions have a stack address, unless they share a name with a global variable, then they have static address. * Undeclared variables that are used have static address. | * Function scope variables die when the function dies. * Global variables last until the lifetime of the document. |
| **Scala** | Explicit or Implicit  Static | Static Scoping | * Fields exist inside an object and can be accessed from outside if public. Since objects are dynamic, these fields come from the heap. * Method parameters exist inside the method they are declared in and have stack addressing. * Local variables exist inside the method they are declared and have stack addressing. | * Fields die when the object is collected by the garbage collector. * Method parameters and local variables die after the function is finished executing. * Block scope variables (if, for, while, etc) dies when the block ends. |

1: All of the above rules apply to classes as well