Object Lifetime and Storage Management

Key events between names and the objects they refer to:

- Creation and destruction of objects
- Creation and destruction of bindings
- Deactivation and reactivation of bindings that may be temporarily unusable
- References to variables, subroutines, types, and so on all of which use bindings

Binding lifetime: The period of time between the creation and the destruction of a name-to-object binding.

Obect lifetime: The period of time between the creation and destruction of an object. An *object* may retain its value and the potential to be accessed even when a given *name* can no longer be used to access it.

When a *variable* is passed to a subroutine by reference, the *binding* between the parameter name and the *variable* that was passed has a *lifetime* shorter than that of the variable itself.

A binding to an object that is no longer live is called a dangling reference.

Object lifetimes generally correspond to one of the three storage allocation mechanisms, used to manage the object's space.

Storage allocation mechanisms:

Static: Static objects are given an absolute address that is retained throughout the program's execution.

Stack: Stack objects are allocated and deallocated in last-in, first-out (LIFO) order, usually in conjunction with *subroutines* and *returns*.

Heap: Heap objects may be *allocated* and *deallocated* at arbitrary times. They require a more general (expensive) storage management algorithm.