Managed Classes

- **Syntax:**

```
class N { … };               // native C++ class
ref class R { … };           // CLR reference type
value class V { … };         // CLR value type
interface class I { … };     // CLR interface type
enum class E { … };          // CLR enumeration type
```

  - N is a standard C++ class. None of the rules have changed.
  - R is a managed class of reference type. It lives on the managed heap and is referenced by a handle:
    - R^ rh = gcnew R;
    - delete rh; [optional: calls destructor which calls Dispose() to release unmanaged resources]
    - Reference types may also be declared as local variables. They still live on the managed heap, but their destructors are called when the thread of execution leaves the local scope.
  - V is a managed class of value type. It lives in its scope of declaration.
    - Value types must be bit-wise copyable. They have no constructors, destructors, or virtual functions.
    - Value types may be boxed to become objects on the managed heap.
  - I is a managed interface. You do not declare its methods virtual. You qualify an implementing class's methods with override (or new if you want to hide the interface's method).
  - E is a managed enumeration.

- N can hold "values", handles, and references to managed types.
- N can hold values, handles, and references to value types.
- N can call methods of managed types.
- R can call global functions and members of unmanaged classes without marshaling.
- R can hold a pointer to an unmanaged object, but is responsible for creating it on the C++ heap and eventually destroying it.