[0650] System.Object: The Object of Our Affection

Everything is a subclass of System.Object. That means that anything System.Object has, everything has. Seems like we should take a close look at System.Object - don't you agree? Fortunately, as you might expect, System.Object is a pretty simple class. Do you want to see it?

https://referencesource.microsoft.com/#mscorlib/system/object.cs (https://referencesource.microsoft.com/#mscorlib/system/object.cs)

That is literally it. (By the way, I give it a 68 for readability - needs improvement. Too many comments! lol!)

Here's a simplified version of that code:

```
namespace System
{
   public class Object
   {
      public Object() { }

      public virtual String ToString() { return GetType().ToString(); }

      public virtual bool Equals(Object obj) { return RuntimeHelpers.Equals(this, obj); }

      public static bool Equals(Object objA, Object objB) {
         if (objA == objB) { return true; }
         if (objA == null || objB == null) { return false; }
         return objA.Equals(objB);
      }

      public static bool ReferenceEquals(Object objA, Object objB) { return objA == objB; }

      public virtual int GetHashCode() { return RuntimeHelpers.GetHashCode(this); }

      public extern Type GetType();

      protected extern Object MemberwiseClone();
   }
}
```

So we have an empty constructor, three virtual (i.e., overrideable) methods, two static methods, and two "external" methods (i.e, low-level C code).

The overrideable methods are:

- ToString() - returns a String
  - returns a String which represents the object. The default is the fully qualified name of the class.

- Equals(Object) - returns a bool
  - returns true if Object is equal to this. The default approach is to compare the bits of value types and then ask any reference types if they think they are equal.
- GetHashCode() - returns an int
  - returns an int hash for the object. The default is the "sync block index" and is "less than ideal." Objects (especially value classes should override this.

The static methods are:

- Object.Equals(Object, Object) - returns a bool
- Object.ReferenceEquals(Object, Object) - returns a bool

ToString()

By far the most useful of these methods is ToString() which can (and should) be used to create a human-readable version of your class' objects. By overriding ToString() in every class you create, you will be able to get the internal state of any of your objects at almost any point in your program. This makes debugging, logging and console-level output much easier.