C# Equality Operator Overloading

Here’s a good example for how to implement operator for equality (==, !=, .Equals()).

This is taken from the Version3 type in OGA.SharedKernel.Lib.

NOTE: These overrides include method signatures for older and newer NET Framework versions.

Equal / Not Equal (==, !=, Equals())

This first block is for equality and inequality overloading.

```
        #region Operator Overloads


        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
        /// <param name="v1"></param>
        /// <param name="v2"></param>
        /// <returns></returns>
        // Force inline as the true/false ternary takes it above ALWAYS_INLINE size even though the asm ends up
smaller
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
#if (NET452 || NET47 || NET48)
        public static bool operator ==(cVersion3 v1, cVersion3 v2)
#else
        public static bool operator ==(cVersion3? v1, cVersion3? v2)
#endif
        {
            // Test "right" first to allow branch elimination when inlined for null checks (== null)
            // so it can become a simple test
```

```
            if (v2 is null)
            {
                // return true/false not the test result https://github.com/dotnet/runtime/issues/4207
                return (v1 is null) ? true : false;
            }

            // Quick reference equality test prior to calling the virtual Equality
            return ReferenceEquals(v2, v1) ? true : v2.Equals(v1);
        }

#if (NET452 || NET47 || NET48)
        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
        /// <param name="v1"></param>
        /// <param name="v2"></param>
        /// <returns></returns>
        public static bool operator !=(cVersion3 v1, cVersion3 v2) => !(v1 == v2);
#else
        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
        /// <param name="v1"></param>
        /// <param name="v2"></param>
        /// <returns></returns>
        public static bool operator !=(cVersion3? v1, cVersion3? v2) => !(v1 == v2);
#endif

        #endregion
```

The above equality logic requires an override of Equals().
Here’s what that looks like:

NOTE: The virtual Equals() method that we override accepts a type of Object.
So, we include an override for that, and a type-specific public overload of the same method name that does the work for both.

```
#if (NET452 || NET47 || NET48)
        /// <summary>
        /// Implementation of the IEquatable interface.
```

```
        /// </summary>
        /// <param name="obj"></param>
        /// <returns></returns>
        public override bool Equals(object obj)
#else
        /// <summary>
        /// Implementation of the IEquatable interface.
        /// </summary>
        /// <param name="obj"></param>
        /// <returns></returns>
        public override bool Equals([NotNullWhen(true)] object? obj)
#endif
        {
            return Equals(obj as cVersion3);
        }

#if (NET452 || NET47 || NET48)
        /// <summary>
        /// Implementation of the IEquatable interface.
        /// </summary>
        /// <param name="obj"></param>
        /// <returns></returns>
        public bool Equals(cVersion3 obj)
#else
        /// <summary>
        /// Implementation of the IEquatable interface.
        /// </summary>
        /// <param name="obj"></param>
        /// <returns></returns>
        public bool Equals([NotNullWhen(true)] cVersion3? obj)
#endif
        {
            return object.ReferenceEquals(obj, this) ||
                (!(obj is null) &&
                _Major == obj._Major &&
                _Minor == obj._Minor &&
                _Patch == obj._Patch);
        }
```

When overriding the == operator, like above, the compiler will present warnings if you have not overridden the GetHashCode() method. Overriding it, is straightforward, but has a different implementation in modern NET (NETCore) than classic NET Framework.
This is because modern NET added the HashCode.Combine() method, that makes things much easier.

So, here's an example of how to override the GetHashCode method, that includes both modern and classic methods:

```
        /// <summary>
        /// Public override of GetHashCode to satisfy compiler warning for overriding Equality.
        /// </summary>
        /// <returns></returns>
        public override int GetHashCode()
        {
#if (NET452 || NET47 || NET48)
            int hash = 11;
            hash = hash * 18 + this._Major.GetHashCode();
            hash = hash * 18 + this._Minor.GetHashCode();
            hash = hash * 18 + this._Patch.GetHashCode();
            return hash;
#else
            return HashCode.Combine(this._Major, this._Minor, this._Patch);
#endif
        }
```

Comparison Overloading (>, <, >=, <=)

And, this block is for greater and less than overloading:

```
        #region Operator Overloads

#if (NET452 || NET47 || NET48)
        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
```

```
        /// <param name="v1"></param>
        /// <param name="v2"></param>
        /// <returns></returns>
        public static bool operator <(cVersion3 v1, cVersion3 v2)
#else
        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
        /// <param name="v1"></param>
        /// <param name="v2"></param>
        /// <returns></returns>
        public static bool operator <(cVersion3? v1, cVersion3? v2)
#endif
        {
           if (v1 is null)
           {
              return !(v2 is null);
           }

           return v1.CompareTo(v2) < 0;
        }

#if (NET452 || NET47 || NET48)
        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
        /// <param name="v1"></param>
        /// <param name="v2"></param>
        /// <returns></returns>
        public static bool operator <=(cVersion3 v1, cVersion3 v2)
#else
        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
        /// <param name="v1"></param>
        /// <param name="v2"></param>
        /// <returns></returns>
        public static bool operator <=(cVersion3? v1, cVersion3? v2)
#endif
        {
```

```
            if (v1 is null)
            {
                return true;
            }

            return v1.CompareTo(v2) <= 0;
        }

#if (NET452 || NET47 || NET48)
        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
        /// <param name="v1"></param>
        /// <param name="v2"></param>
        /// <returns></returns>
        public static bool operator >(cVersion3 v1, cVersion3 v2) => v2 < v1;
#else
        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
        /// <param name="v1"></param>
        /// <param name="v2"></param>
        /// <returns></returns>
        public static bool operator >(cVersion3? v1, cVersion3? v2) => v2 < v1;
#endif

#if (NET452 || NET47 || NET48)
        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
        /// <param name="v1"></param>
        /// <param name="v2"></param>
        /// <returns></returns>
        public static bool operator >=(cVersion3 v1, cVersion3 v2) => v2 <= v1;
#else
        /// <summary>
        /// Implements the IEquatable interface, same as the native Version class.
        /// </summary>
        /// <param name="v1"></param>
        /// <param name="v2"></param>
```

```
        /// <returns></returns>
        public static bool operator >=(cVersion3? v1, cVersion3? v2) => v2 <= v1;
#endif


        #endregion
```

NOTE: The above comparison overloads requires a type-specific CompareTo() implementation override.

So, here’s one that you can work from:

NOTE: The virtual CompareTo() method that we override accepts a type of Object.
So, we include an override for that, and a type-specific public overload of the same method name that does the work for both.

```
#if (NET452 || NET47 || NET48)
        /// <summary>
        /// Implementation of the IComparable interface.
        /// </summary>
        /// <param name="version"></param>
        /// <returns></returns>
        /// <exception cref="ArgumentException"></exception>
        public int CompareTo(object version)
#else
        /// <summary>
        /// Implementation of the IComparable interface.
        /// </summary>
        /// <param name="version"></param>
        /// <returns></returns>
        /// <exception cref="ArgumentException"></exception>
        public int CompareTo(object? version)
#endif
        {
            if (version == null)
            {
                return 1;
            }

            if (version is cVersion3 v)
            {
```

```
            return CompareTo(v);
        }

        throw new ArgumentException("Invalid Version Instance.");
    }

#if (NET452 || NET47 || NET48)
    /// <summary>
    /// Implementation of the IComparable interface.
    /// </summary>
    /// <param name="value"></param>
    /// <returns></returns>
    public int CompareTo(cVersion3 value)
#else
    /// <summary>
    /// Implementation of the IComparable interface.
    /// </summary>
    /// <param name="value"></param>
    /// <returns></returns>
    public int CompareTo(cVersion3? value)
#endif
    {
        if (value == null)
        {
            return 1;
        }

        return
            object.ReferenceEquals(value, this) ? 0 :
            value is null ? 1 :
            _Major != value._Major ? (_Major > value._Major ? 1 : -1) :
            _Minor != value._Minor ? (_Minor > value._Minor ? 1 : -1) :
            _Patch != value._Patch ? (_Patch > value._Patch ? 1 : -1) :
            0;
    }
```

Revision #3
Created 14 January 2025 16:42:02 by glwhite
Updated 19 March 2025 15:40:34 by glwhite