**Part 58 - C# Tutorial - Why should you override Equals() method**

In this session, let's understand the difference between **"=="** operator and **Equals()** method. In C#, every type directly or indirectly inherits from System.Object. So, the **Equals()** virtual method, that has a default implementation is available in every type via inheritance. In this example, variables i and j are integers. So, == and Equals() method returns true, since i and j, both variables have a value of 10.

using System;  
namespace Pragim  
{  
    public class MainClass  
    {  
        private static void Main()  
        {  
            int i = 10;  
            int j = 10;  
            Console.WriteLine(i == j);  
            Console.WriteLine(i.Equals(j));  
        }  
    }  
}

Along the same lines, the sample program below, compares 2 enums and both, the == operator and Equals() method returns true, since bothe direction1 and direction2 enums has the same underlying integer value of 1.  
using System;  
namespace Pragim  
{  
    public class MainClass  
    {  
        private static void Main()  
        {  
            Direction direction1 = Direction.East;  
            Direction direction2 = Direction.East;  
  
  
            Console.WriteLine(direction1 == direction2);  
            Console.WriteLine(direction1.Equals(direction2));  
        }  
    }  
    public enum Direction  
    {  
        East = 1,  
        West = 2,  
        North = 3,  
        South = 4  
    }  
}  
  
  
However, if the type is a reference type, then by default **"=="** operator checks for **reference equality** and **.Equals()** method checks for **value equality**. Let's understand what we mean by reference and value equality.  
  
  
In the example below, C1 and C2 are 2 different object reference variables, but they point to the same object. Keep in mind, object reference variables are different from objects. Object reference variables, stay on the stack and are pointers to actual objects on the heap. Since, C1 and C2 both refer to the same object, the reference equality and value equality is true. Value equality means that two objects contain the same values. In this example, the actual object is only one, so obviously the values are also equal. If two objects have reference equality, then they also have value equality, but value equality does not guarantee reference equality.  
using System;  
namespace Pragim  
{  
    public class MainClass  
    {  
        private static void Main()  
        {  
            Customer C1 = new Customer();  
            C1.FirstName = "Simon";  
            C1.LastName = "Tan";  
  
  
            Customer C2 = C1;  
  
  
            Console.WriteLine(C1 == C2);  
            Console.WriteLine(C1.Equals(C2));  
        }  
    }  
    public class Customer  
    {  
        public string FirstName { get; set; }  
        public string LastName { get; set; }  
    }  
}  
  
  
For the example below, == operator returns False. This makes sense because C1 and C2 are referring to different objects. However, .Equals() method returns flase, inspite of the values across C1 and C2 being the same. Hence, it makes sense to override, the Equals() method to return true when the values across the objects are same.  
using System;  
namespace Pragim  
{  
    public class MainClass  
    {  
        private static void Main()  
        {  
            Customer C1 = new Customer();  
            C1.FirstName = "Simon";  
            C1.LastName = "Tan";  
  
  
            Customer C2 = new Customer();  
            C2.FirstName = "Simon";  
            C2.LastName = "Tan";  
  
  
            Console.WriteLine(C1 == C2);  
            Console.WriteLine(C1.Equals(C2));  
        }  
    }  
    public class Customer  
    {  
        public string FirstName { get; set; }  
        public string LastName { get; set; }  
    }  
}  
  
  
The example below overrides, Equals() method. When overriding Equals() method, make sure the passed in object is not null and can be casted to the type we are comparing. When overriding Equals(), you also need to override GetHashCode(), otherwise you get a compiler warning.  
using System;  
namespace Pragim  
{  
    public class MainClass  
    {  
        private static void Main()  
        {  
            Customer C1 = new Customer();  
            C1.FirstName = "Simon";  
            C1.LastName = "Tan";  
  
  
            Customer C2 = new Customer();  
            C2.FirstName = "Simon";  
            C2.LastName = "Tan";  
  
  
            Console.WriteLine(C1 == C2);  
            Console.WriteLine(C1.Equals(C2));  
        }  
    }  
    public class Customer  
    {  
        public string FirstName { get; set; }  
        public string LastName { get; set; }  
  
  
        public override bool Equals(object obj)  
        {  
            // If the passed in object is null  
            if (obj == null)  
            {  
                return false;  
            }  
            if (!(obj is Customer))  
            {  
                return false;  
            }  
            return (this.FirstName == ((Customer)obj).FirstName)  
                && (this.LastName == ((Customer)obj).LastName);  
        }  
  
  
        public override int GetHashCode()  
        {  
            return FirstName.GetHashCode() ^ LastName.GetHashCode();  
        }  
    }  
}