**Operator Overloading in C# (Overloading the - operator)**

What is the purpose of overloading an operator?

When dealing with lists, more often than not, we do not know whether the array values will be of int, string, double, etc.

In these cases, we use custom lists.

public class CustomClassList<T>

{

// Code will be entered here

}

When we attempt to use operators such as +,-,\*,%, etc, the data type(int, string, double, etc.) is not known to the program.

So when you try to subtract one list from the other (list1 and list2) like the following:

CustomClassList<int> actual = list2 - list1;

list1 + list2 gets the red error underline with the following message:

" Operator '-' cannot be applied to operands of type 'CustomClassList<int>' and 'CustomClassList<int>' "

When you get this error, you must overload the operator. Here is a basic 'skeleton' to fix this issue.

**Syntax:**

public static CustomClassList<T> operator -(**CustomClassList<T> list1, CustomClassList<T> list2**)

{

}

When using the operator *keyword*, you must use a public and static modifier like above.

**Parameters:**

The operator overload also takes 2 ***parameters***. The code inside this 'skeleton' then checks the first and second lists and updates a third list.

(**CustomClassList<T> list1, CustomClassList<T> list2**)

**Return Type:**

You must then return a value after writing your conditionals inside the skeleton on the operator overload. The return value must match the list data type. It cannot return a string, int, double, etc. The return must be of the same value as what is declared after the public static modifiers.

return comboList;

**My Example Below - Not All Code required for Overloading Operators**

My example takes in 2 *parameters*, or 2 lists.  
I then instantiate a new, 3rd list, that is used to add the index values that match my conditions. I am looping through the first list, then the 2nd list, then checking to see if the index values match. A Boolean is declared and referenced in an if statement to register as false, therefore, it is not added. The new list is returned in the end to display the leftover numbers or strings.

**Here is the final code:**

public static CustomClassList<T> operator -(CustomClassList<T> list1, CustomClassList<T> list2)

{

CustomClassList<T> comboList = new CustomClassList<T>();

for (int i = 0; i < list1.count; i++)

{

bool isEqual = false;

for (int j = 0; j < list2.count; j++)

{

if (list1[i].Equals(list2[j]))

{

isEqual = true;

}

else if((j == list2.count-1) && (isEqual == false)){

comboList.Add(list1[i]);

}

}

}

return comboList;

}