-Operator Overload

Passes in two generic lists through the parameter of the method. Once the method is called a third list is instantiated to hold the elements we need from the two lists being passed in. The method starts with a for loop through the first list. Inside that for loop (looping through the first list) is another for loop to loop through the second list (the list we are subtracting from the first list). It starts at the 0 index of the first list and looks through all the elements of the second list and compares them to that 0 index to see if any of the elements contain the same values. Once it has compared all the objects in the second list to the 0 index of the first list it will move to the first index of the first list and now check all the vales of the second list against the first index of the first list to see if any values are the same. Inside the second for loop is an if, else if statement to check to make sure that if it doesn’t find values that match that it doesn’t add to our third list until it has looped through all values in the second list. This way if the 0 index of the first list doesn’t match the 0 index in the second list it will keep looping to see if the 0 index of the first list might match with an element further down the second list and not add that 0 index of the first list immediately.

C#

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

Parameters

Two generic lists.

Syntax

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

{

CustomList<T> list3 = new CustomList<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( isEqual == false && j == list2.count-1)

{

list3.Add(list1[i]);

}

}

}

return list3;

Example

public void CustomList\_Subtract\_OverloadSubtractOperator()

{

//arrange

CustomList<int> list1 = new CustomList<int>();

CustomList<int> list2 = new CustomList<int>();

string expected = "35";

//act

list1.Add(1);

list1.Add(3);

list1.Add(5);

list2.Add(2);

list2.Add(1);

list2.Add(6);

CustomList<int> actual = list1 - list2;

//assert

Assert.AreEqual(expected, actual.ToString());

}

Return

Third list that is made up of all the elements of first list except for any that were also found in the second list. In the example above by subtracting list2 from list1 we would expect to get a return of 35