List<T>.RemoveAt(Int32)Method

Namespace: System.Collections.Generic
Assemblies: System.Collections.dll, mscorlib.dll, netstandard.dll

In this article

Removes the element at the specified index of the List<T>.

C# Copy

```
public void RemoveAt (int index);
```

Parameters

index Int32
The zero-based index of the element to remove.

Implements

RemoveAt(Int32) RemoveAt(Int32)

Exceptions

ArgumentOutOfRangeException
`index` is less than 0.

-or-

`index` is equal to or greater than Count.

Examples

The following example demonstrates how to add, remove, and insert a simple business object in a List<T>.

C# Copy

```
using System;
using System.Collections.Generic;
// Simple business object. A PartId is used to identify the type of part
// but the part name can change.
```

```
    public class Part : IEquatable<Part>
        {
            public string PartName { get; set; }

            public int PartId { get; set; }

            public override string ToString()
            {
                return "ID: " + PartId + "   Name: " + PartName;
            }
            public override bool Equals(object obj)
            {
                if (obj == null) return false;
                Part objAsPart = obj as Part;
                if (objAsPart == null) return false;
                else return Equals(objAsPart);
            }
            public override int GetHashCode()
            {
                return PartId;
            }
            public bool Equals(Part other)
            {
                if (other == null) return false;
                return (this.PartId.Equals(other.PartId));
            }
        // Should also override == and != operators.

        }
    public class Example
    {
        public static void Main()
        {
            // Create a list of parts.
            List<Part> parts = new List<Part>();

            // Add parts to the list.
            parts.Add(new Part() {PartName="crank arm", PartId=1234});
            parts.Add(new Part() { PartName = "chain ring", PartId = 1334 });
            parts.Add(new Part() { PartName = "regular seat", PartId = 1434 });
            parts.Add(new Part() { PartName = "banana seat", PartId = 1444 });
            parts.Add(new Part() { PartName = "cassette", PartId = 1534 });
            parts.Add(new Part() { PartName = "shift lever", PartId = 1634 });

            // Write out the parts in the list. This will call the overridden ToString method
            // in the Part class.
            Console.WriteLine();
            foreach (Part aPart in parts)
            {
                Console.WriteLine(aPart);
            }


            // Check the list for part #1734. This calls the IEquitable.Equals method
            // of the Part class, which checks the PartId for equality.
```

```
Console.WriteLine("\nContains(\"1734\"): {0}",
parts.Contains(new Part {PartId=1734, PartName="" }));

// Insert a new item at position 2.
Console.WriteLine("\nInsert(2, \"1834\")");
parts.Insert(2, new Part() { PartName = "brake lever", PartId = 1834 });


//Console.WriteLine();
foreach (Part aPart in parts)
{
    Console.WriteLine(aPart);
}

Console.WriteLine("\nParts[3]: {0}", parts[3]);

Console.WriteLine("\nRemove(\"1534\")");

// This will remove part 1534 even though the PartName is different,
// because the Equals method only checks PartId for equality.
parts.Remove(new Part(){PartId=1534, PartName="cogs"});

Console.WriteLine();
foreach (Part aPart in parts)
{
    Console.WriteLine(aPart);
}
Console.WriteLine("\nRemoveAt(3)");
// This will remove the part at index 3.
parts.RemoveAt(3);

Console.WriteLine();
foreach (Part aPart in parts)
{
    Console.WriteLine(aPart);
}

    /*

     ID: 1234   Name: crank arm
     ID: 1334   Name: chain ring
     ID: 1434   Name: regular seat
     ID: 1444   Name: banana seat
     ID: 1534   Name: cassette
     ID: 1634   Name: shift lever

     Contains("1734"): False

     Insert(2, "1834")
     ID: 1234   Name: crank arm
     ID: 1334   Name: chain ring
     ID: 1834   Name: brake lever
     ID: 1434   Name: regular seat
     ID: 1444   Name: banana seat
     ID: 1534   Name: cassette
```

```
            ID: 1634   Name: shift lever

            Parts[3]: ID: 1434   Name: regular seat

            Remove("1534")

            ID: 1234   Name: crank arm
            ID: 1334   Name: chain ring
            ID: 1834   Name: brake lever
            ID: 1434   Name: regular seat
            ID: 1444   Name: banana seat
            ID: 1634   Name: shift lever

            RemoveAt(3)

            ID: 1234   Name: crank arm
            ID: 1334   Name: chain ring
            ID: 1834   Name: brake lever
            ID: 1444   Name: banana seat
            ID: 1634   Name: shift lever


        */

    }
}
```

Remarks

When you call RemoveAt to remove an item, the remaining items in the list are renumbered to replace the removed item. For example, if you remove the item at index 3, the item at index 4 is moved to the 3 position. In addition, the number of items in the list (as represented by the Count property) is reduced by 1.

This method is an O(*n*) operation, where *n* is (Count - `index`).