


C # | LinkedList Sınıfı

LinkedList < T > Class, **System.Collections.Generic** ad alanında bulunur. Bu generik tip elementlerin hızlı bir şekilde takılmasını ve çıkarılmasını sağlar. Klasik bir bağlantılı liste uygular. Her nesne ayrı ayrı tahsis edilir. LinkedList'te bazı işlemler tüm koleksiyonun kopyalanmasını gerektirmez.

LinkedList Sınıfının Özellikleri:

- **LinkedList** < T > genel amaçlı bir bağlantılı listedir. Enum destekler.
- Takma ve çıkarma işlemi **O (1)** işlemleridir.
- Düğümleri kaldırabilir ve bunları aynı listede veya başka bir listede yeniden yerleştirebilirsiniz; bu, öbek üzerinde ayrılan ek nesnelere neden olmaz.
- Liste aynı zamanda dahili bir sayıya sahip olduğundan, Count özelliğini almak bir O (1) işlemidir.
- **LinkedList** < T > nesnesindeki her düğüm **LinkedListNode** < T > türündedir.
- **LinkedList** sınıfı zincirleme, bölme, döngü ya da listeyi tutarsız bir durumda bırakabilecek diğer özellikleri desteklemiyor.
- **LinkedList** boşsa, **İlk** ve **Son** özellikleri null içerir.
- **LinkedList** iki kat bağlantılıdır, bu nedenle her bir düğüm bir sonraki düğüme ve geri bir önceki düğüme işaret eder.



```
// C# Linked List Koleksiyonu -Örnek
```

```
System;
```

```
System.Collections; using System.Collections;
```

```
System.Collections.Generic; using System.Collections.Generic;
```

```
namespace GFG {
```

```
class GFG {
```

```
    // Driver code
```

```
    public static void Main()
```

```
    {
```

```
        // Creating a LinkedList of Strings
```

```
        LinkedList<String> myList = new LinkedList<String>();
```

```
        // Adding nodes in LinkedList
```

```
        myList.AddLast( "Geeks" );
```

```
        myList.AddLast( "for" );
```

```
        myList.AddLast( "Data Structures" );
```

```
        myList.AddLast( "Noida" );
```

```
        // To check if LinkedList is empty or not
```

```
        if (myList.Count > 0)
```

```
            Console.WriteLine( "LinkedList is not empty" );
```

```
        else
```

```
            Console.WriteLine( "LinkedList is empty" );
```

```
    }
```

```
}
```

Methods

AddAfter(LinkedListNode<T>, LinkedListNode<T>)

AddAfter(LinkedListNode<T>, T)

AddBefore(LinkedListNode<T>, LinkedListNode<T>)

AddBefore(LinkedListNode<T>, T)

AddFirst(LinkedListNode<T>)

AddFirst(T)

AddLast(LinkedListNode<T>)

AddLast(T)

Clear()

Contains(T)

Equals(Object)

Find(T)

FindLast(T)

GetHashCode()

Remove(LinkedListNode<T>)

Remove(T)

RemoveFirst()

RemoveLast()

ToString()

Description

This method adds the specified new node after the specified existing node in the LinkedList<T>.

This method adds a new node containing the specified value after the specified existing node in the LinkedList<T>.

This method adds the specified new node before the specified existing node in the LinkedList<T>.

This method adds a new node containing the specified value before the specified existing node in the LinkedList<T>.

This method adds the specified new node at the start of the LinkedList<T>.

This method adds a new node containing the specified value at the start of the LinkedList<T>.

This method adds the specified new node at the end of the LinkedList<T>.

This method adds a new node containing the specified value at the end of the LinkedList<T>.

This method removes all the nodes from the LinkedList<T>.

This method determines whether a value is in the LinkedList<T>.

This method determines whether the specified object is equal to the current object.

This method finds the first node that contains the specified value.

This method finds the last node that contains the specified value.

This method serves as the default hash function.

This method removes the specified node from the LinkedList<T>.

This method removes the first occurrence of the specified value from the LinkedList<T>.

This method removes the node at the start of the LinkedList<T>.

This method removes the node at the end of the LinkedList<T>.

This method returns a string that represents the current object.