

C# - SORTEDLIST CLASS

http://www.tutorialspoint.com/csharp/csharp_sortedlist.htm

Copyright © tutorialspoint.com

The SortedList class represents a collection of key-and-value pairs that are sorted by the keys and are accessible by key and by index.

A sorted list is a combination of an array and a hash table. It contains a list of items that can be accessed using a key or an index. If you access items using an index, it is an ArrayList, and if you access items using a key, it is a Hashtable. The collection of items is always sorted by the key value.

Methods and Properties of the SortedList Class

The following table lists some of the commonly used **properties** of the **SortedList** class:

Property	Description
Capacity	Gets or sets the capacity of the SortedList.
Count	Gets the number of elements contained in the SortedList.
IsFixedSize	Gets a value indicating whether the SortedList has a fixed size.
IsReadOnly	Gets a value indicating whether the SortedList is read-only.
Item	Gets and sets the value associated with a specific key in the SortedList.
Keys	Gets the keys in the SortedList.
Values	Gets the values in the SortedList.

The following table lists some of the commonly used **methods** of the **SortedList** class:

Sr.No.	Methods
1	public virtual void Add(object key, object value); Adds an element with the specified key and value into the SortedList.
2	public virtual void Clear; Removes all elements from the SortedList.
3	public virtual bool ContainsKey(object key); Determines whether the SortedList contains a specific key.
4	public virtual bool ContainsValue(object value); Determines whether the SortedList contains a specific value.
5	public virtual object GetByIndex(int index); Gets the value at the specified index of the SortedList.
6	public virtual object GetKey(int index);

Gets the key at the specified index of the SortedList.

7 **public virtual IList GetKeyList;**

Gets the keys in the SortedList.

8 **public virtual IList GetValueList;**

Gets the values in the SortedList.

9 **public virtual int IndexOfKey***objectkey*;

Returns the zero-based index of the specified key in the SortedList.

10 **public virtual int IndexOfValue***objectvalue*;

Returns the zero-based index of the first occurrence of the specified value in the SortedList.

11 **public virtual void Remove***objectkey*;

Removes the element with the specified key from the SortedList.

12 **public virtual void RemoveAt***intindex*;

Removes the element at the specified index of SortedList.

13 **public virtual void TrimToSize;**

Sets the capacity to the actual number of elements in the SortedList.

Example

The following example demonstrates the concept:

```
using System;
using System.Collections;

namespace CollectionsApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            SortedList sl = new SortedList();

            sl.Add("001", "Zara Ali");
            sl.Add("002", "Abida Rehman");
            sl.Add("003", "Joe Holzner");
            sl.Add("004", "Mausam Benazir Nur");
            sl.Add("005", "M. Amlan");
            sl.Add("006", "M. Arif");
            sl.Add("007", "Ritesh Saikia");

            if (sl.ContainsValue("Nuha Ali"))
            {
                Console.WriteLine("This student name is already in the list");
            }
            else
            {
            }
```

```
        sl.Add("008", "Nuha Ali");
    }

    // get a collection of the keys.
    ICollection key = sl.Keys;

    foreach (string k in key)
    {
        Console.WriteLine(k + ": " + sl[k]);
    }
}
}
```

When the above code is compiled and executed, it produces the following result:

```
001: Zara Ali
002: Abida Rehman
003: Joe Holzner
004: Mausam Banazir Nur
005: M. Amlan
006: M. Arif
007: Ritesh Saikia
008: Nuha Ali
```

Loading [MathJax]/jax/output/HTML-CSS/jax.js