Sorted Collection Types

Article • 09/15/2021 • 2 minutes to read

The System.Collections.SortedList class, the System.Collections.Generic.SortedList<TKey,TValue> generic class, and the System.Collections.Generic.SortedDictionary<TKey,TValue> generic class are similar to the Hashtable class and the Dictionary<TKey,TValue> generic class in that they implement the IDictionary interface, but they maintain their elements in sort order by key, and they do not have the O(1) insertion and retrieval characteristic of hash tables. The three classes have several features in common:

- All three classes implement the System.Collections.IDictionary interface. The two generic classes also implement the System.Collections.Generic.IDictionary<TKey,TValue> generic interface.
- Each element is a key/value pair for enumeration purposes.

  > **Note**
  >
  > The nongeneric **SortedList** class returns **DictionaryEntry** objects when enumerated, although the two generic types return **KeyValuePair<TKey,TValue>** objects.

- Elements are sorted according to a System.Collections.IComparer implementation (for nongeneric SortedList) or a System.Collections.Generic.IComparer<T> implementation (for the two generic classes).
- Each class provides properties that return collections containing only the keys or only the values.

The following table lists some of the differences between the two sorted list classes and the SortedDictionary<TKey,TValue> class.

SortedList nongeneric class and SortedList<TKey,TValue> generic class	**SortedDictionary<TKey,TValue> generic class**
The properties that return keys and values are indexed, allowing efficient indexed retrieval.	No indexed retrieval.

SortedList nongeneric class and SortedList<TKey,TValue> generic class	**SortedDictionary<TKey,TValue> generic class**
Retrieval is O(log n).	Retrieval is O(log n).
Insertion and removal are generally O(n); however, insertion is O(log n) for data that are already in sort order, so that each element is added to the end of the list. (This assumes that a resize is not required.)	Insertion and removal are O(log n).
Uses less memory than a SortedDictionary<TKey,TValue>.	Uses more memory than the SortedList nongeneric class and the SortedList<TKey,TValue> generic class.

For sorted lists or dictionaries that must be accessible concurrently from multiple threads, you can add sorting logic to a class that derives from ConcurrentDictionary<TKey,TValue>. When considering immutability, the following corresponding immutable types follow similar sorting semantics: ImmutableSortedSet<T> and ImmutableSortedDictionary<TKey,TValue>.

> **Note**
>
> For values that contain their own keys (for example, employee records that contain an employee ID number), you can create a keyed collection that has some characteristics of a list and some characteristics of a dictionary by deriving from the **KeyedCollection<TKey,TItem>** generic class.

Starting with .NET Framework 4, the SortedSet<T> class provides a self-balancing tree that maintains data in sorted order after insertions, deletions, and searches. This class and the HashSet<T> class implement the ISet<T> interface.

See also

- System.Collections.IDictionary
- System.Collections.Generic.IDictionary<TKey,TValue>
- ConcurrentDictionary<TKey,TValue>
- Commonly Used Collection Types