**Класс SortedList**

Класс SortedList предназначен для создания коллекции, в которой пары "ключ-

значение" хранятся в порядке, отсортированном по значению ключей. В классе

SortedList реализуются интерфейсы IDictionary, ICollection, IEnumerable и

ICloneable.

В классе SortedList определено несколько конструкторов, включая следующие.

public SortedList()

public SortedList(IDictionary d)

public SortedList(int initialCapacity)

public SortedList(IComparer comparer)

В первом конструкторе создается пустая коллекция, первоначальная емкость

которой равна нулю. Во втором конструкторе создается пустая коллекция типа

SortedList, которая инициализируется элементами из коллекции *d*. Ее первоначальная

емкость равна количеству указанных элементов. В третьем конструкторе создается

пустая коллекция типа SortedList, первоначальный размер которой определяет емкость,

задаваемая параметром *initialCapacity.* Эта емкость соответствует размеру

базового массива, используемого для хранения элементов коллекции. И в четвертой

форме конструктора с помощью параметра *comparer* указывается способ, используемый

для сравнения объектов по списку. В этой форме создается пустая коллекция,

первоначальная емкость которой равна нулю.

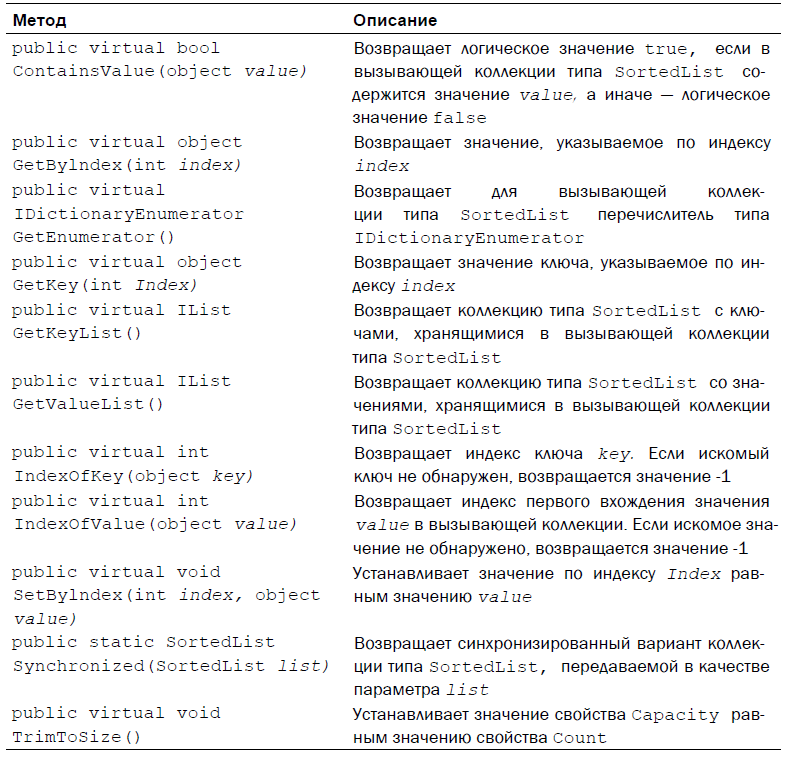
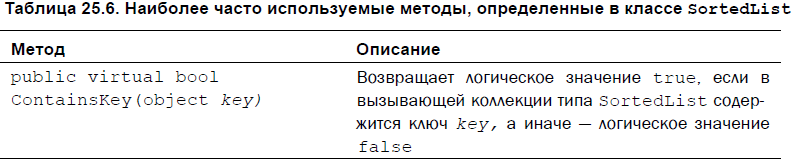
При добавлении новых элементов в список емкость коллекции типа SortedList

увеличивается автоматически по мере надобности.

В классе SortedList определяется ряд собственных методов, помимо тех, что

уже объявлены в интерфейсах, которые в нем реализуются. Некоторые из наиболее

часто используемых методов этого класса перечислены в табл. 25.6.



Кроме того, для получения списка всех значений из коллекции служит метод GetValueList().

Безусловно, в классе SortedList также поддерживается индексатор,

определяемый в интерфейсе IDictionary и позволяющий устанавливать и получать

значение по заданному ключу.

В классе SortedList доступны также открытые свойства, определенные в тех интерфейсах,

которые в нем реализуются. Как и в классе Hashtable, в данном классе

особая роль принадлежит двум свойствам, Keys и Values, поскольку с их помощью

можно получить доступную только для чтения коллекцию ключей или значений из коллекции типа SortedList. Эти свойства определяются в интерфейсе IDictionary

следующим образом.

public virtual ICollection Keys { get; }

public virtual ICollection Values { get; }

Порядок следования ключей и значений отражает порядок их расположения в коллекции

типа SortedList.

Аналогично коллекции типа Hashtable, пары "ключ-значение" сохраняются

в коллекции типа SortedList в форме структуры типа DictionaryEntry, но, как

правило, доступ к ключам и значениям осуществляется по отдельности с помощью

методов и свойств, определенных в классе SortedList.

В приведенном ниже примере программы демонстрируется применение класса

SortedList. Это переработанный и расширенный вариант предыдущего примера,

демонстрировавшего применение класса Hashtable, вместо которого теперь используется

класс SortedList. Глядя на результат выполнения этой программы, вы можете

сами убедиться, что теперь список полученных значений оказывается отсортированным

по заданному ключу.

(***glava25\_5***)

class SLDemo

{

static void Main()

{

//create hash table

SortedList ht = new SortedList();

//add elements

ht.Add("buildings", "living area");

ht.Add("car", "transport");

ht.Add("book", "collection of words");

ht.Add("apple", "fruit");

//add elements through indexator

ht["tractor"] = "agrarian machine";

//get keys collection

ICollection c = ht.Keys;

//use keys to get values

foreach (var str in c)

Console.WriteLine(str + ": " + ht[str]);

Console.WriteLine();

//show list with int index

for (int i = 0; i < ht.Count; i++)

Console.WriteLine(ht.GetByIndex(i));

Console.WriteLine();

//show int indexec of elements

Console.WriteLine("Int indexes of elements in list.");

foreach (var str in c)

Console.WriteLine(str + ": " + ht.IndexOfKey(str));

}

}