**Класс SortedList<TKey, TValue>**

В коллекции класса SortedList<TKey, TValue> хранится отсортированный список

пар "ключ-значение". Это обобщенный эквивалент класса необобщенной коллекции

SortedList. В классе SortedList<TKey, TValue> реализуются интерфейсы IDictionary,

IDictionary<TKey, TValue>, ICollection, ICollection<KeyValuePair<TKey,

TValue>>, IEnumerable и IEnumerable<KeyValuePair<TKey, TValue>>. Размер

коллекции типа SortedList<TKey, TValue> изменяется динамически, автоматически

увеличиваясь по мере необходимости. Класс SortedList<TKey, TValue> подобен классу

SortedDictionary<TKey, TValue>, но у него другие рабочие характеристики. В частности,

класс SortedList<TKey, TValue> использует меньше памяти, тогда как класс

SortedDictionary<TKey, TValue> позволяет быстрее вставлять неупорядоченные элементы

в коллекцию.

В классе SortedList<TKey, TValue> предоставляется немало конструкторов.

Ниже перечислены наиболее часто используемые конструкторы этого класса.

public SortedList()

public SortedList(IDictionary<TKey, TValue> dictionary)

public SortedList(int capacity)

public SortedList(IComparer<TK> comparer)

В первой форме конструктора создается пустой список с выбираемой по умолчанию

первоначальной емкостью. Во второй форме конструктора создается отсортированный

список с указанным количеством элементов *dictionary.* В третьей форме

конструктора с помощью параметра *capacity* задается емкость коллекции, создаваемой

в виде отсортированного списка. Если размер списка заранее известен, то, указав

емкость создаваемой коллекции, можно исключить изменение размера списка во

время выполнения, что, как правило, требует дополнительных затрат вычислительных

ресурсов. И в четвертой форме конструктора допускается указывать с помощью параметра

*comparer* способ сравнения объектов, содержащихся в списке.

Емкость коллекции типа SortedList<TKey, TValue> увеличивается автоматически

по мере необходимости, когда в список добавляются новые элементы. Если текущая

емкость коллекции превышается, то она увеличивается.

В классе SortedList<TKey, TValue> определяется ряд собственных методов, помимо

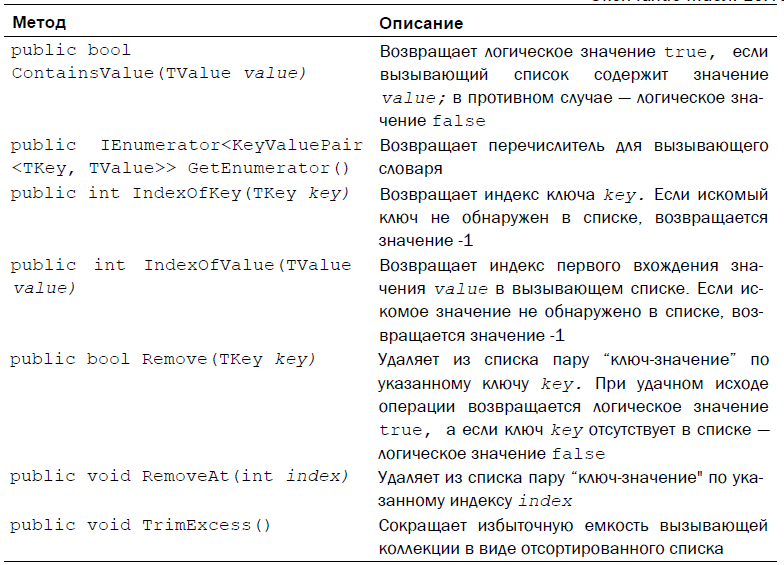
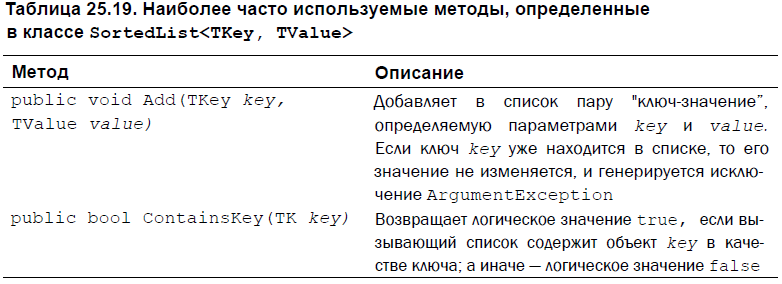
тех, что уже объявлены в интерфейсах, которые в нем реализуются. Некоторые

из наиболее часто используемых методов этого класса перечислены в табл. 25.19. Следует

иметь в виду, что перечислитель, возвращаемый методом GetEnumerator(), служит

для перечисления пар "ключ-значение", хранящихся в отсортированном списке в

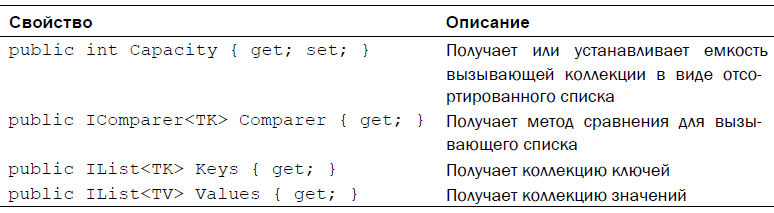
виде объектов типа KeyValuePair.



Кроме того, в классе SortedList<TK, TV> определяются собственные свойства,

помимо тех, что уже объявлены в интерфейсах, которые в нем реализуются. Эти свойства

приведены ниже.



И наконец, в классе SortedList<TKey, TValue> реализуется приведенный ниже

индексатор, определенный в интерфейсе IDictionary<TKey, TvalueX

public TValue this[TKey key] { get; set; }

Этот индексатор служит для получения и установки значения элемента коллекции,

а также для добавления в коллекцию нового элемента. Но в данном случае в качестве

индекса служит ключ элемента, а не сам индекс.

В приведенном ниже примере демонстрируется применение класса

SortedList<TKey, TValue>. Это еще один измененный вариант представленного ранее примера базы данных работников. В данном варианте база данных хранится в

коллекции типа SortedList.

(***glava25\_13***)

class SLDemo

{

static void Main()

{

//make dictionary for name lastname

//employes and sallary

SortedList<string, double> dict =

new SortedList<string, double>();

dict.Add("Batler, John", 73000);

dict.Add("Shwarz, Sarah", 59000);

dict.Add("Pike, Thomas", 45000);

dict.Add("Frank, Ed", 99000);

//get key collection lastname name

ICollection<string> c = dict.Keys;

//use keys to get sallary

foreach (var str in c)

Console.WriteLine("{0}, salarry: {1:C}", str, dict[str]);

}

}