**Класс Queue**

Еще одной распространенной структурой данных является *очередь,* действующая

по принципу: первым пришел — первым обслужен. Это означает, что первым из очереди

извлекается элемент, помещенный в нее первым.

В программировании очереди применяются для хранения

таких элементов, как процессы, выполняющиеся в данный момент в системе, списки

приостановленных транзакций в базе данных или пакеты данных, полученные по Интернету.

Кроме того, очереди нередко применяются в области имитационного моделирования.

Класс коллекции, поддерживающий очередь, носит название Queue. В нем реализуются

интерфейсы ICollection, IEnumerable и ICloneable. Этот класс создает

динамическую коллекцию, которая расширяется, если в ней необходимо хранить вводимые

элементы. Так, если в очереди требуется свободное место, ее размер увеличивается

на коэффициент роста, который по умолчанию равен 2,0.

В классе Queue определяются приведенные ниже конструкторы.

public Queue()

public Queue(int capacity)

public Queue(int capacity, float growFactor)

public Queue(ICollection col)

В первой форме конструктора создается пустая очередь с выбираемыми по умолчанию

емкостью и коэффициентом роста 2,0. Во второй форме создается пустая очередь,

первоначальный размер которой определяет емкость, задаваемая параметром

*сараcity,* а коэффициент роста по умолчанию выбирается для нее равным 2,0. В третьей

форме допускается указывать не только емкость (в качестве параметра *capacity),*

но и коэффициент роста создаваемой очереди (в качестве параметра *growFactor*

в пределах от 1,0 до 10,0). И в четвертой форме создается очередь, состоящая из элементов

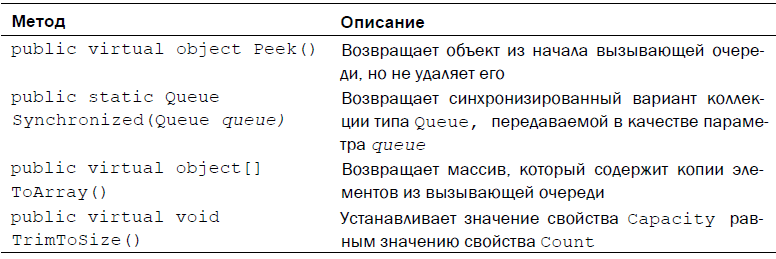
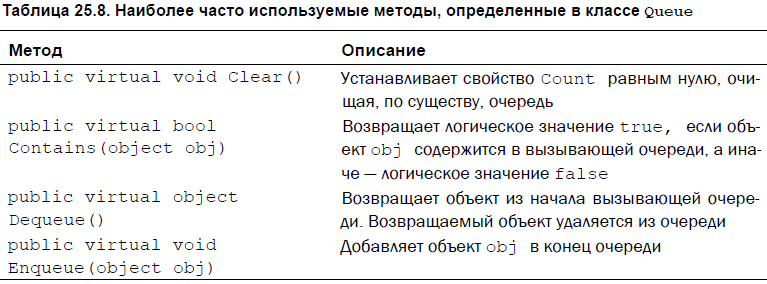
указываемой коллекции *col.* Ее первоначальная емкость равна количеству указанных

элементов, а коэффициент роста по умолчанию выбирается для нее равным 2,0.

В классе Queue 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. 25.8.



(***glava25\_7***)

class QueueDemo

{

static void ShowEnq(Queue q, int a)

{

q.Enqueue(a);

Console.WriteLine("Take into queue: " + a);

Console.Write("Content of queue: ");

foreach (var i in q)

Console.Write(i + " ");

Console.WriteLine();

}

static void ShowDeq(Queue q)

{

Console.Write("Dequeue -> ");

int a = (int)q.Dequeue();

Console.WriteLine(a);

Console.Write("Content of queue: ");

foreach (var i in q)

Console.Write(i + " ");

Console.WriteLine();

}

static void Main()

{

Queue q = new Queue();

foreach (var i in q)

Console.Write(i + " ");

Console.WriteLine();

ShowEnq(q, 22);

ShowEnq(q, 65);

ShowEnq(q, 91);

ShowDeq(q);

ShowDeq(q);

ShowDeq(q);

try

{

ShowDeq(q);

}

catch(InvalidOperationException)

{

Console.WriteLine("Queue is empty.");

}

}

}