C# | Queue Class

geeksforgeeks.org/c-sharp-queue-class

November 6, 2018

Queue represents a ***first-in, first out*** collection of object. It is used when you need a first-in, first-out access of items. When you add an item in the list, it is called **enqueue**, and when you remove an item, it is called **dequeue** . This class comes under **System.Collections** namespace and implements *ICollection, IEnumerable, and ICloneable* interfaces.

Characteristics of Queue Class:

- **Enqueue** adds an element to the end of the Queue.
- **Dequeue** removes the oldest element from the start of the Queue.
- **Peek** returns the oldest element that is at the start of the Queue but does not remove it from the Queue.
- The **capacity** of a Queue is the number of elements the Queue can hold.
- As elements are added to a Queue, the capacity is automatically increased as required by reallocating the internal array.
- Queue accepts **null** as a valid value for reference types and allows duplicate elements.

Constructors

Constructor	Description
Queue()	Initializes a new instance of the Queue class that is empty, has the default initial capacity, and uses the default growth factor.
Queue(ICollection)	Initializes a new instance of the Queue class that contains elements copied from the specified collection, has the same initial capacity as the number of elements copied, and uses the default growth factor.
Queue(Int32)	Initializes a new instance of the Queue class that is empty, has the specified initial capacity, and uses the default growth factor.
Queue(Int32, Single)	Initializes a new instance of the Queue class that is empty, has the specified initial capacity, and uses the specified growth factor.

Example:

filter_none
edit

play_arrow

brightness_4

```
using System;
using System.Collections;
class GFG {
public static void Main()
{
Queue myQueue = new Queue();
myQueue.Enqueue( "one" );
Console.Write( "Total number of elements in the Queue are : " );
Console.WriteLine(myQueue.Count);
myQueue.Enqueue( "two" );
Console.Write( "Total number of elements in the Queue are : " );
Console.WriteLine(myQueue.Count);
myQueue.Enqueue( "three" );
Console.Write( "Total number of elements in the Queue are : " );
Console.WriteLine(myQueue.Count);
myQueue.Enqueue( "four" );
Console.Write( "Total number of elements in the Queue are : " );
Console.WriteLine(myQueue.Count);
myQueue.Enqueue( "five" );
Console.Write( "Total number of elements in the Queue are : " );
Console.WriteLine(myQueue.Count);
myQueue.Enqueue( "six" );
Console.Write( "Total number of elements in the Queue are : " );
Console.WriteLine(myQueue.Count);
}
}
```

Output:

```
Total number of elements in the Queue are : 1
Total number of elements in the Queue are : 2
Total number of elements in the Queue are : 3
Total number of elements in the Queue are : 4
Total number of elements in the Queue are : 5
Total number of elements in the Queue are : 6
```

Properties

Property	Description
Count	Gets the number of elements contained in the Queue.

IsSynchronized	Gets a value indicating whether access to the Queue is synchronized (thread safe).
SyncRoot	Gets an object that can be used to synchronize access to the Queue.

Example:

filter_none
edit

play_arrow

brightness_4

```
using System;
using System.Collections;
class GFG {
public static void Main()
{
Queue myQueue = new Queue();
myQueue.Enqueue( "Chandigarh" );
myQueue.Enqueue( "Delhi" );
myQueue.Enqueue( "Noida" );
myQueue.Enqueue( "Himachal" );
myQueue.Enqueue( "Punjab" );
myQueue.Enqueue( "Jammu" );
Console.Write( "Total number of elements in the Queue are : " );
Console.WriteLine(myQueue.Count);
}
}
```

Output:

Total number of elements in the Queue are : 6

Methods

Method	Description
Clear()	Removes all objects from the Queue.
Clone()	Creates a shallow copy of the Queue.
Contains(Object)	Determines whether an element is in the Queue.
CopyTo(Array, Int32)	Copies the Queue elements to an existing one-dimensional Array, starting at the specified array index.

Dequeue()	Removes and returns the object at the beginning of the Queue.
Enqueue(Object)	Adds an object to the end of the Queue.
Equals(Object)	Determines whether the specified object is equal to the current object.
GetEnumerator()	Returns an enumerator that iterates through the Queue.
GetHashCode()	Serves as the default hash function.
GetType()	Gets the Type of the current instance.
MemberwiseClone()	Creates a shallow copy of the current Object.
Peek()	Returns the object at the beginning of the Queue without removing it.
Synchronized(Queue)	Returns a new Queue that wraps the original queue, and is thread safe.
ToArray()	Copies the Queue elements to a new array.
ToString()	Returns a string that represents the current object.
TrimToSize()	Sets the capacity to the actual number of elements in the Queue.

Example 1:

filter_none
edit

play_arrow

brightness_4

```
using System;
using System.Collections;
class GFG {
public static void Main()
{
Queue myQueue = new Queue();
myQueue.Enqueue(5);
myQueue.Enqueue(10);
myQueue.Enqueue(15);
myQueue.Enqueue(20);
myQueue.Enqueue(25);
Console.WriteLine(myQueue.Contains(7));
}
}
```

Output:

False

Example 2:

filter_none
edit

play_arrow

brightness_4

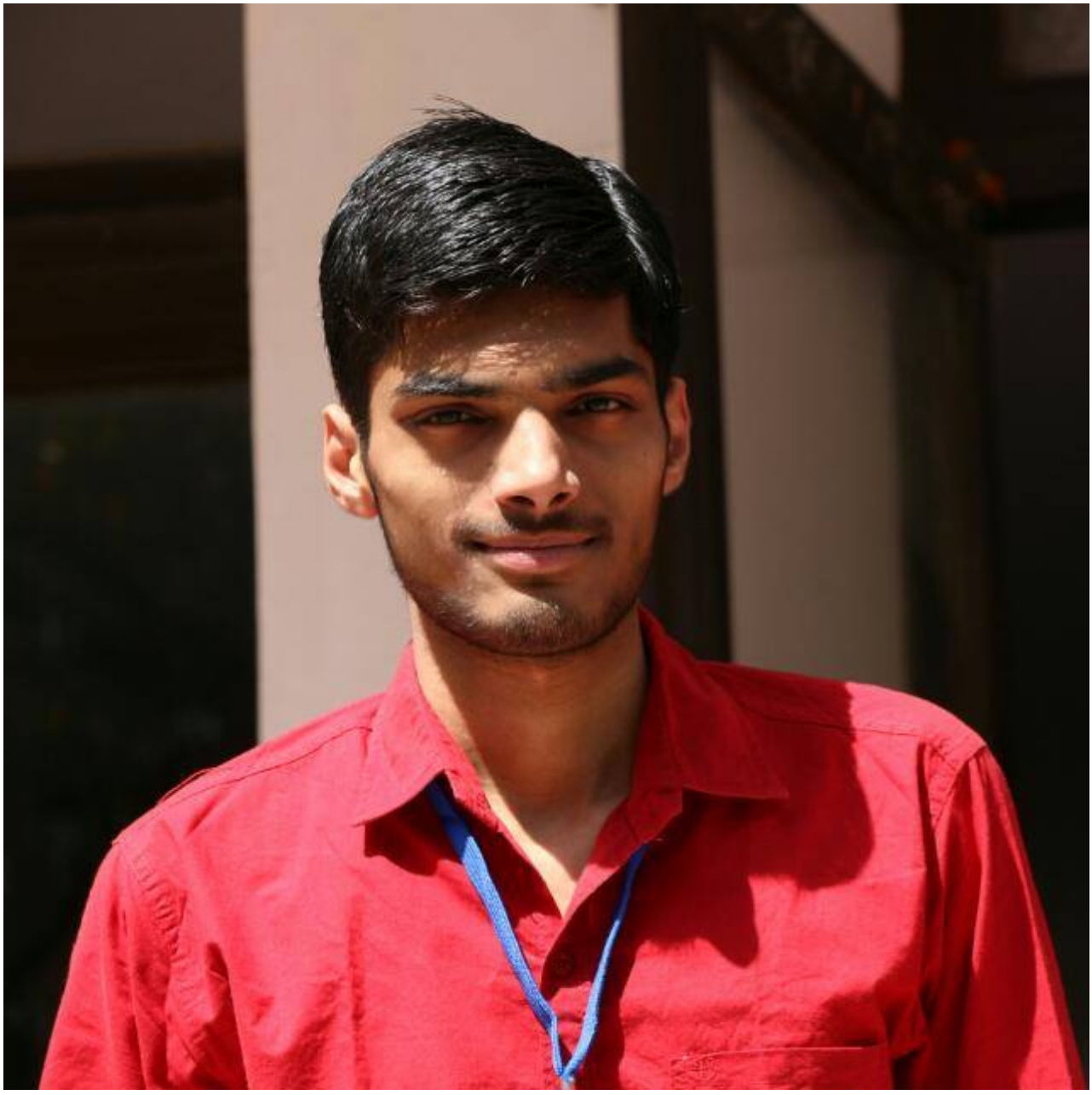
```
using System;
using System.Collections;
class GFG {
public static void Main()
{
Queue myQueue = new Queue();
myQueue.Enqueue( "Geeks" );
myQueue.Enqueue( "Geeks Classes" );
myQueue.Enqueue( "Noida" );
myQueue.Enqueue( "Data Structures" );
myQueue.Enqueue( "GeeksforGeeks" );
Object[] arr = myQueue.ToArray();
foreach (Object obj in arr)
{
Console.WriteLine(obj);
}
}
}
```

Output:

Geeks
Geeks Classes
Noida
Data Structures
GeeksforGeeks

Reference:

https://docs.microsoft.com/en-us/dotnet/api/system.collections.queue?view=netframework-4.7.2



Sahil_Bansall
In love with a semicolon because sometimes i miss it so badly)

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.