

System.Collections Classes

Class	Description
-------	-------------

ArrayList	Represents an array of objects whose size is dynamically increased as required.
-----------	---

Hashtable	Represents a collection of key/value pairs that are organized based on the hash code of the key.
-----------	--

Queue	Represents a first in, first out (FIFO) collection of objects.
-------	--

Stack	Represents a last in, first out (LIFO) collection of objects.
-------	---

List<T> Class

Namespace: System.Collections.Generic**Assembly:** System.Collections.dll

Represents a strongly typed list of objects that can be accessed by index. Provides methods to search, sort, and manipulate lists.

```
// Create a list of parts.
```

```
List<Part> parts = new List<Part>();
```

```
// Add parts to the list.
```

```
parts.Add(new Part() { PartName = " crank arm", PartID = 1234 });
```

```
parts.Add(new Part() { PartName = " chain ring", PartID = 1234 });
```

```
parts.Add(new Part() { PartName = " regular seat", PartID = 1444 });
```

```
;
```

```
parts.Add(new Part() { PartName = " banana seat", PartID = 1444 });
```

```
parts.Add(new Part() { PartName = " cas set te", PartID = 1444 });
```

```
parts.Add(new Part() { PartName = " shift lever", PartID = 1444 });
```

List<T> Methods

List<T> Methods (cont)

List<T> .Sort	Sorts the elements or a portion of the elements in the List<T> using either the specified or default IComparer<T> implementation or a provided Comparison<T> delegate to compare list elements.	parts.Sort();
---------------	---	---------------

For further information and examples visit this link

Stack<T> Class

Namespace: System.Collections.Generic**Assembly:** System.Collections.dll

Specifies the type of elements in the stack.

```
// Create a stack of strings
```

```
Stack<string> numbers = new Stack<string>();
```

```
// Add items to the stack
```

```
numbers.Push("one");
```

```
numbers.Push("two");
```

Stack<T> Methods

Method	Usage	Example
Stack<T>.Push(T)	Inserts an object at the top of the Stack<T>.	numbers.Push("one");
Stack<T>.Pop()	Removes and returns the object at the top of the Stack<T>.	numbers.Pop();
Stack<T>.Peek()	Represents a first in, first out (FIFO) collection of objects.	numbers.Peek();
Stack<T>.Contains(T)	The object at the top of the Stack<T>.	stack2.Contains("four");
Stack<T>.Clear()	Determines whether an element is in the Stack<T>.	stack2.Clear();

For further information and examples visit this link

Method	Usage	Example
List<T>.Add(T)	Adds an object to the end of the List<T>.	<pre>parts.Add(new Part() { PartName = " crank arm", PartId = 1234 })</pre>
List<T>.Remove(T)	Removes the first occurrence of a specific object from the List<T>.	<pre>parts.Remove(new Part() { PartId = 1534, PartName = " cog s" })</pre>
List<T>.Clear	Removes all elements from the List<T>.	<pre>parts.Clear();</pre>
List<T>.Contains(T)	Determines whether an element is in the List<T>.	<pre>parts.Contains(new Part { PartId = 1734, PartName = " " });</pre>



By **masterofcode**

cheatography.com/masterofcode/

Published 2nd December, 2021.

Last updated 3rd December, 2021.

Page 1 of 5.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

HashSet<T> Class

Namespace: System.Collections.Generic

Assembly: System.Collections.dll

Represents a set of values.

```
HashSet<int> evenNumbers = new HashSet<int>();
HashSet<int> oddNumbers = new HashSet<int>();

for (int i = 0; i < 5; i++)
{
    // Populate numbers with just even numbers.
    evenNumbers.Add(i * 2);

    // Populate oddNumbers with just odd numbers.
    oddNumbers.Add((i * 2) + 1);
}
```

HashSet<T> Methods

Method	Usage	Example
HashSet<T>.Add(T)	Adds the specified element to a set.	evenNumbers.Add(i * 2);
HashSet<T>.Remove(T)	Removes all elements from a HashSet<T> object.	numbers.Remove(0);
HashSet<T>.Clear()	Represents a first in, first out (FIFO) collection of objects.	numbers.Clear();
HashSet<T>.Contains(T)	Determines whether a HashSet<T> object contains the specified element.	numbers.Contains(0)

For further information and examples visit [this link](#)

System.Collections.Generic Classes

Queue<T> Class

Namespace: System.Collections.Generic

Assembly: System.Collections.dll

Represents a first-in, first-out collection of objects.

```
Create a queue of strings
Queue<string> numbers = new Queue<string>();
Add items in the queue
numbers.Enqueue("one");
numbers.Enqueue("two");
numbers.Enqueue("three");
```

Queue<T> Methods

Method	Usage	Example
Queue<T>.Enqueue(T)	Adds an object to the end of the Queue<T>.	numbers.Enqueue("one");
Queue<T>.Dequeue()	Removes and returns the object at the beginning of the Queue<T>.	numbers.Dequeue();
Queue<T>.Peek()	The object at the beginning of the Queue<T>.	numbers.Peek();
Queue<T>.Contains(T)	Determines whether an element is in the Queue<T>.	numbers.Contains(0)

For further information and examples visit [this link](#)

Dictionary<TKey,TValue> Class

Namespace: System.Collections.Generic

Assembly: System.Collections.dll

Represents a collection of keys and values.

```
// Create a new dictionary of strings, with string keys
Dictionary<string, string> openWith =
    new Dictionary<string, string>();
openWith.Add("txt", "notepad.exe");
```

Dictionary<TKey,TValue> Methods

Class	Description
Dictionary<T-Key,TV-value>	Represents a collection of key/value pairs that are organized based on the key.
List<T>	Represents a list of objects that can be accessed by index. Provides methods to search, sort, and modify lists.
Queue<T>	Represents a first in, first out (FIFO) collection of objects.
SortedList<T-Key,TV-value>	Represents a collection of key/value pairs that are sorted by key based on the associated IComparer<T> implementation.
Stack<T>	Represents a last in, first out (LIFO) collection of objects.

Method	Usage	Example
Dictionary<T-Key,TV-value>.Add(TKey, TVValue)	Adds the specified key and value to the dictionary.	<pre>openWith.Add ("txt", "notepad.exe");</pre>
Dictionary<T-Key,TV-value>.Remove(TKey)	Removes the value with the specified key from the Dictionary<T-Key,TV-value>.	<pre>public bool Remove (TKey key); openWith.Remove ("doc");</pre>



By **masterofcode**

cheatography.com/masterofcode/

Published 2nd December, 2021.

Last updated 3rd December, 2021.

Page 2 of 5.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Dictionary<TKey,TValue> Methods (cont)

Dictionary<TKey,TValue>.Clear() Removes all keys and values from the Dictionary<TKey,TValue>.

Dictionary<TKey,TValue>.ContainsKey(TKey key) Determines whether the Dictionary<TKey,TValue> contains the specified key.

Dictionary<TKey,TValue>.ContainsValue(TValue value) Determines whether the Dictionary<TKey,TValue> contains a specific value.

For further information and examples visit this link



By masterofcode

Published 2nd December, 2021.

Last updated 3rd December, 2021.

Page 3 of 5.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>