Cheatography

C# Collection Cheat Sheet

by masterofcode via cheatography.com/140841/cs/30006/

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 = 1334 });
parts.Add(new Part() { PartName = "regular seat",
PartId = 1434 });
parts.Add(new Part() { PartName = "banana seat",
PartId = 1444 });
parts.Add(new Part() { PartName = "cassette",
PartId = 1534 });
parts.Add(new Part() { PartName = "shift lever",
PartId = 1634 });
```

List<T> Methods

Method	Usage	Example
List<T>.Add(T)	Adds an object to the end of the List<T>.	`parts.Add(new Part() { PartName = "crank arm", PartId = 1234 });`

List<T> Methods (cont)

List<T>.Remove(T)	Removes the first occurrence of a specific object from the List<T>.	`parts.Remove(new Part() { PartId = 1534, PartName = "cogs" });`
List<T>.Clear	Removes all elements from the List<T>.	`parts.Clear();`
List<T>.Contains(T)	Determines whether an element is in the List<T>.	`parts.Contains(new Part { PartId = 1734, PartName = "" }));`
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.P-op();`
Stack<-T>.Push(T)	Represents a first in, first out (FIFO) collection of objects.	`numbers.Push-("one");`
Stack<-T>.Peek	The object at the top of the Stack<T>.	`numbers.P-eek();`
Stack<-T>.Contai-ns(T)	Determines whether an element is in the Stack<T>.	`stack2.Conta-ins("four");`
Stack<-T>.Clear	Removes all objects from the Stack<T>.	`stack2.Cl-ear();`

For further information and examples visit this link

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
HashSe-t<T>.A-dd(T)	Adds the specified element to a set.	`evenNumbe-rs.Add(i * 2);`
HashSe-t<T>.R-emove(T)	Removes all elements from a HashSet<T> object.	`numbers.R-emove(0);`
HashSe-t<T>.Clear	Represents a first in, first out (FIFO) collection of objects.	`numbers.C-lear();`
HashSe-t<T>.Cont-ains(T)	Determines whether a HashSet<T> object contains the specified element.	`numbers.C-ontains(0)`

For further information and examples visit this link

System.Collections.Generic Classes

Class	Description
Dictionar-y<T-Key,TV-alue>	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.
SortedLis-t<T-Key,TV-alue>	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.

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

Method	Usage	Example
Dictionary<TKey,TValue>.Add(TKey, TValue)	Adds the specified key and value to the dictionary.	`openWith.Add("txt", "notepad.exe");`
Dictionary<TKey,TValue>.Remove	Removes the value with the specified key from the Dictionary<TKey,TValue>.	`public bool Remove (TKey key); openWith.Remove("doc");`
Dictionary<TKey,TValue>.Clear	Removes all keys and values from the Dictionary<TKey,TValue>.	`public void Clear (); openWith.Clear();`
Dictionary<TKey,TValue>.ContainsKey(TKey)	Determines whether the Dictionary<TKey,TValue> contains the specified key.	`public bool ContainsKey (TKey key); openWith.ContainsKey("ht");`
Dictionary<TKey,TValue>.ContainsValue(TValue)	Determines whether the Dictionary<TKey,TValue> contains a specific value.	`public bool ContainsValue (TValue value); openWith.ContainsValue("hypertrm.exe");`

For further information and examples visit this link