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<P art> parts = new List<P art >();
// Add parts to the list.
parts.A dd(new Part() { PartName = " crank arm", PartId = 1234 });
parts.A dd(new Part() { PartName = " chain ring", PartId = 1334 });
parts.A dd(new Part() { PartName = " regular seat", PartId = 1434
;
parts.A dd(new Part() { PartName = " banana seat", PartId = 1444 });
parts.A dd(new Part() { PartName = " cas set te", PartId = 1534 });
parts.A dd(new Part() { PartName = " shift lever", PartId = 1634 });
```

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.S ort() ;`

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< str ing> numbers = new Stack< str ing >();
// Add items to the stack
number s.P ush ("on e");
number s.P ush ("tw o");
```

Stack<T> Methods

Method	Usage	Example
Stack<-T>.P-ush(T)	Inserts an object at the top of the Stack<T>.	`number s.P ush ("on e");`
Stack<-T>.Pop	Removes and returns the object at the top of the Stack<T>.	`number s.P op();`
Stack<-T>.P-ush(T)	Represents a first in, first out (FIFO) collection of objects.	`number s.P ush ("on e");`
Stack<-T>.Peek	The object at the top of the Stack<T>.	`number s.P eek();`
Stack<-T>.Con-tains(T)	Determines whether an element is in the Stack<T>.	`stack2.Co nta ins ("fo u r");`
Stack<-T>.Clear	Removes all objects from the Stack<T>.	`stack2.Cl ear();`

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->.	`parts.Add(new Part() { PartName = " crank arm", PartId = 1234 });`
List<T->.Remove(T)	Removes the first occurrence of a specific object from the List<T->.	`parts.Remove(new Part() { PartId = 1534, PartName = " cog s" });`
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 = " " }));`

Cheatography

C# Collection Cheat Sheet

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

HashSet<T> Class

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

Represents a set of values.

```
HashSe t<i nt> evenNu mbers = new HashSe t<i nt>();
HashSe t<i nt> oddNumbers = new HashSe t<i nt>();

for (int i = 0; i < 5; i++)
{
    // Populate numbers with just even numbers.
    evenNu mbe rs.A dd(i * 2);

    // Populate oddNumbers with just odd numbers.
    oddNum ber s.A dd((i * 2) + 1);
}
```

HashSet<T> Methods

Method	Usage	Example
HashSet<T>.Add(T)	Adds the specified element to a set.	`evenNu mbe rs.A dd(i * 2);`
HashSet<T>.Remove(T)	Removes all elements from a HashSet<T> object.	`number s.R emo ve(0);`
HashSet<T>.Clear	Represents a first in, first out (FIFO) collection of objects.	`number s.C lear();`
HashSet<T>.Contains(T)	Determines whether a HashSet<T> object contains the specified element.	`number s.C ont ains(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< str ing> numbers = new Queue< str ing >();
Add items in the queue
number s.E nqu eue ("on e");
number s.E nqu eue ("tw o");
number s.E nqu eue ("th ree ");
```

Queue<T> Methods

Method	Usage	Example
Queue<T>.Enqueue(T)	Adds an object to the end of the Queue<T>.	`number s.E nqu eue ("on e");`
Queue<T>.Dequeue	Removes and returns the object at the beginning of the Queue<T>.	`number s.D equ eue();`
Queue<T>.Peek	The object at the beginning of the Queue<T>.	`number s.P eek();`
Queue<T>.Contains(T)	Determines whether an element is in the Queue<T>.	`number s.C ont ains(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 ke
Dictio nar y<s tring, string> openWith =
   new Dictio nar y<s tring, string >();
openWi th.A dd ("tx t", " not epa d.e xe");
```

Dictionary<TKey,TValue> Methods

Class	Description
Dictionary<TKey,TValue>	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<TKey,TValue>	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<TKey,TValue>.Add(TKey, TValue)	Adds the specified key and value to the dictionary.	`openWi th.A dd ("tx t", " not epa d .e xe");`
Dictionary<TKey,TValue>.Remove	Removes the value with the specified key from the Dictionary<TKey,TValue>.	`public bool Remove (TKey key); openWi th.R em ove ("do c");`

Dictionary<TKey,TValue> Methods (cont)

Dictionary<TKey,TValue>.Clear	Removes all keys and values from the Dictionary<TKey,TValue>.	`public void Clear ();` `openWi th.Cl ear();`
Dictionary<TKey,TValue>.ContainsKey(TKey)	Determines whether the Dictionary<TKey,TValue> contains the specified key.	`public bool Contai nsKey (TKey key);` `openWi th.C on tai nsK ey( " ht");`
Dictionary<TKey,TValue>.ContainsValue(TValue)	Determines whether the Dictionary<TKey,TValue> contains a specific value.	`public bool Contai nsValue (TValue value);` `openWi th.C on tai nsV alu e("h ype rtr m. e xe");`

For further information and examples visit this link