C# for Potatoes Cheat Sheet

by PotatoCodes via cheatography.com/205795/cs/43920/

String Manipulation

ToLower	Converts all characters to lowercase.	str.ToLower() "HELLO".ToLower() // "hello"
ToUpper	Converts all characters to uppercase.	str.ToUpper() "hello".ToUpper() // "HELLO"
Trim	Removes leading and trailing white-space characters.	str.Trim() " hello ".Trim() // "hello"
TrimStart	Removes leading white-space characters.	str.TrimStart() " hello ".TrimStart() // "hello "
TrimEnd	Removes trailing white-space characters.	str.TrimEnd() " hello ".TrimEnd() // " hello"
Substring	Retrieves a substring starting at a specified position.	str.Substring(0, 5) "hello world".Substring(0, 5) // "hello"
Replace	Replaces all occurrences of a specified string with another	str.Replace("world", "C#") "hello world".Replace("world", "C#") // "hello C#"
Split	Splits a string into an array based on a delimiter.	str.Split(' ') "hello world".Split(' ') // ["hello", "world"]
Join	Concatenates an array into a single string with a delimiter	string.Join(" ", words) string.Join(" ", new[] { "hello", "-world" }) // "hello world"
Contains	Checks if a string contains a specified substring.	str.Contains("world") "hello world".Contains("world") // true
IndexOf	Finds the first occurrence of a substring.	str.IndexOf("world") "hello world".IndexOf("world") // 6
LastIndexOf	Finds the last occurrence of a substring.	str.LastIndexOf("world") "hello world world".LastIndex-Of("world") // 12
StartsWith	Checks if a string starts with a specified substring.	str.StartsWith("hello") "hello world".StartsWith("hello") // true
EndsWith	Checks if a string ends with a specified substring.	str.EndsWith("world") "hello world".EndsWith("world") // true
IsNullOrEmpty	Checks if a string is null or empty.	string.IsNullOrEmpty(str) string.IsNullOrEmpty("") // true
IsNullOrWhit-eSpace	Checks if a string is null, empty, or whitespace.	string.IsNullOrWhiteSpace(str) string.IsNullOrWhiteS-pace(" ") // true
Format	Replaces format items in a string with the string representation of corresponding objects.	string.Format("Hello, {0}!", "world") // "Hello, world!"

String Manipulation (cont)

PadLeft	Pads a string on the left with a specified character.	str.PadLeft(10) "hello".PadLeft(10) // " hello"
PadRight	Pads a string on the right with a specified character.	str.PadRight(10) "hello".PadRight(10) // "hello "
ToCharArray	Converts the string to a character array.	str.ToCharArray() "hello".ToCharArray() // ['h', 'e', 'l', 'l', 'o']
Compare	Compares two strings and returns an integer indicating their relative position in the sort order.	string.Compare("apple", "banana") // -1
Equals	Checks if two strings are equal.	"hello".Equals("hello") // true

Errors

SystemException	Base class for all system-defined exceptions.	catch (SystemException ex)
ArgumentException	Thrown when one of the arguments provided to a method is not valid.	throw new ArgumentException("message", "paramName");
ArgumentNullException	Thrown when a null argument is passed to a method that does not accept it.	throw new ArgumentNullException("paramName");
ArgumentOutOfRangeException	Thrown when an argument is outside the range of valid values.	throw new ArgumentOutOfRangeException("paramName");
InvalidOperationException	Thrown when a method call is invalid for the object's current state.	throw new InvalidOperationException("message");
IndexOutOfRangeException	Thrown when an index is outside the bounds of an array or collection.	throw new IndexOutOfRangeException("message");
FileNotFoundException	Thrown when an attempt to access a file that does not exist on disk is made.	throw new FileNotFoundException("message", innerException);
IOException	Base class for exceptions that occur during I/O operations.	catch (IOException ex)
NullReferenceException	Thrown when there is an attempt to dereference a null object reference.	throw new NullReferenceException("message");
OverflowException	Thrown when an arithmetic, casting, or conversion operation in a checked context overflows.	throw new OverflowException("message");
FormatException	Thrown when the format of an argument is invalid, e.g., parsing numbers.	throw new FormatException("message");
UnauthorizedAccessException	Thrown when the operating system denies access to a file or directory.	throw new UnauthorizedAccessException("message");

Errors (cont)

NotSupportedException	Thrown when a method or operation is not supported.	throw new NotSupportedException("message");
DivideByZeroException	Thrown when there is an attempt to divide an integer by zero.	throw new DivideByZeroException("message");
TimeoutException	Thrown when an operation exceeds the allotted time.	throw new TimeoutException("message");

List

List<Type> listName = new List<T->();	Declares a new list.
listName.Count	Gets the number of elements contained in the List<T>.
listName.Add(T);	Adds an object to the end of the List<T>.
listName.Clear();	Removes all elements from the List<T>.
listName.Contains(T);	Determines whether an element is in the List<T>.
listName.Equals(Object);	Determines whether the specified object is equal to the current object.
listName.IndexOf(T);	Searches for the specified object and returns the zero-based index of the first occurrence within the entire List<T>.
listName.Remove(T);	Removes the first occurrence of a specific object from the List<T>.
listName.RemoveAt(Int32);	Removes the element at the specified index of the List<T>.

Arrays

Initializing array	int[] numbers = { 1, 2, 3, 4, 5 };
Accessing element	int firstNumber = numbers[0]; // 1
Modifying element	numbers[0] = 10; // { 10, 2, 3, 4, 5 }

For Loop	for (int i = 0; i < numbers.Length; i++) {
	  Console.WriteLine(numbers[i]);
	}

Foreach Loop	foreach (int number in numbers) {
	  Console.WriteLine(number);
	}

Index of Element	int index = Array.IndexOf(numbers, 3); // 2
Element based on condition	int foundNumber = Array.Find(numbers, n => n > 2); // 10
Sorting	Array.Sort(numbers); // { 2, 3, 4, 10, 50 }
Reverse	Array.Reverse(numbers); // { 50, 10, 4, 3, 2 }
Copy	Array.Copy(numbers, copy, numbers.Length); // copy = { 50, 10, 4, 3, 2 }

Arrays (cont)

Clear	Array.Clear(numbers, 0, numbers.Length); // { 0, 0, 0, 0, 0 }
Resize	Array.Resize(ref numbers, 7); // { 0, 0, 0, 0, 0, 0, 0 }
ToList	List<int> numbersList = numbers.ToList();

LINQ

Where	var evenNumbers = numbers.Where(n => n % 2 == 0);
Select	var squares = numbers.Select(n => n * n);
OrderBy	var sortedNumbers = numbers.OrderBy(n => n);
OrderByDescending	var sortedNumbersDesc = numbers.OrderByDescending(n => n);
ThenBy	var sortedPeople = people.OrderBy(p => p.LastName).ThenBy(p => p.FirstName);
ThenByDescending	var sortedPeopleDesc = people.OrderBy(p => p.LastName).ThenByDescending(p => p.FirstName);
GroupBy	var groupedByAge = people.GroupBy(p => p.Age);
Join	var joinQuery = customers.Join(orders, customer => customer.Id, order => order.CustomerId, (customer, order) => new { customer.Name, order.OrderId });
GroupJoin	var groupJoinQuery = customers.GroupJoin(orders, customer => customer.Id, order => order.CustomerId, (customer, orders) => new { customer.Name, Orders = orders });
SelectMany	var allOrders = customers.SelectMany(c => c.Orders);
Take	var firstThreeNumbers = numbers.Take(3);
Skip	var allButFirstThreeNumbers = numbers.Skip(3);
TakeWhile	var takeWhileQuery = numbers.TakeWhile(n => n < 5);
SkipWhile	var skipWhileQuery = numbers.SkipWhile(n => n < 5);
Distinct	var distinctNumbers = numbers.Distinct();
Union	var unionQuery = numbers1.Union(numbers2);
Intersect	var intersectQuery = numbers1.Intersect(numbers2);
Except	var exceptQuery = numbers1.Except(numbers2);
Concat	var concatQuery = numbers1.Concat(numbers2);
Any	bool hasEvenNumbers = numbers.Any(n => n % 2 == 0);
All	bool allPositive = numbers.All(n => n > 0);
Contains	bool containsNumber = numbers.Contains(5);
First	int firstNumber = numbers.First();
FirstOrDefault	int firstOrDefaultNumber = numbers.FirstOrDefault();
Last	int lastNumber = numbers.Last();
LastOrDefault	int lastOrDefaultNumber = numbers.LastOrDefault();
Single	int singleNumber = numbers.Single(n => n == 5);

LINQ (cont)

SingleOrDefault	int singleOrDefaultNumber = numbers.SingleOrDefault(n => n == 5);
Count	int count = numbers.Count();
Sum	int sum = numbers.Sum();
Average	double average = numbers.Average();
Min	int min = numbers.Min();
Max	int max = numbers.Max();

DateTime

Now	Gets the current date and time.	DateTime now = DateTime.Now; // e.g., "2024-07-28 14:35:00"
UtcNow	Gets the current date and time in Coordinated Universal Time (UTC).	DateTime utcNow = DateTime.UtcNow; // e.g., "2024-07-28 18:35:00"
Today	Gets the current date with the time component set to 00:00:00.	DateTime today = DateTime.Today; // e.g., "2024-07-28 00:00:00"
Date	Gets the date component of the DateTime instance.	DateTime date = now.Date; // e.g., "2024-07-28 00:00:00"
Day	Gets the day of the month represented by the DateTime instance.	int day = now.Day; // e.g., 28
Month	Gets the month component of the DateTime instance.	int month = now.Month; // e.g., 7
Year	Gets the year component of the DateTime instance.	int year = now.Year; // e.g., 2024
Hour	Gets the hour component of the DateTime instance.	int hour = now.Hour; // e.g., 14
Minute	Gets the minute component of the DateTime instance.	int minute = now.Minute; // e.g., 35
Second	Gets the second component of the DateTime instance.	int second = now.Second; // e.g., 0
DayOfWeek	Gets the day of the week represented by the DateTime instance.	DayOfWeek dayOfWeek = now.DayOfWeek; // e.g., DayOfWeek.Sunday
DayOfYear	Gets the day of the year represented by the DateTime instance.	int dayOfYear = now.DayOfYear; // e.g., 210
AddDays	Adds the specified number of days to the DateTime instance.	DateTime futureDate = now.AddDays(5); // e.g., "2024-08-02 14:35:00"
AddMonths	Adds the specified number of months to the DateTime instance.	DateTime futureDate = now.AddMonths(1); // e.g., "2024-08-28 14:35:00"
AddYears	Adds the specified number of years to the DateTime instance	DateTime futureDate = now.AddYears(1); // e.g., "2025-07-28 14:35:00"

DateTime (cont)

AddHours	Adds the specified number of hours to the DateTime instance.	DateTime futureDate = now.AddHours(5); // e.g., "2024-07-28 19:35:00"
AddMinutes	Adds the specified number of minutes to the DateTime instance.	DateTime futureDate = now.AddMinutes(30); // e.g., "2024-07-28 15:05:00"
AddSeconds	Adds the specified number of seconds to the DateTime instance.	DateTime futureDate = now.AddSeconds(30); // e.g., "2024-07-28 14:35:30"
AddMilliseconds	Adds the specified number of milliseconds to the DateTime instance.	DateTime futureDate = now.AddMilliseconds(500); // e.g., "2024-07-28 14:35:00.500"
AddTicks	Adds the specified number of ticks to the DateTime instance.	DateTime futureDate = now.AddTicks(1000000); // e.g., "2024-07-28 14:35:00.0001000"
Subtract	Subtracts the specified date and time from this instance.	TimeSpan duration = now.Subtract(pastDate); // e.g., "2.00:00:00" (2 days)
ToString	Converts the value of the DateTime instance to its equivalent string representation.	string str = now.ToString("yyyy-MM-dd HH:mm:ss"); // "2024-07-28 14:35:00"
Parse	Converts the string representation of a date and time to its DateTime equivalent.	DateTime dt = DateTime.Parse("2024-07-28 14:35:00");
TryParse	Converts the string representation of a date and time to its DateTime equivalent and returns a value that indicates whether the conversion succeeded.	bool success = DateTime.TryParse("2024-07-28 14:35:00", out DateTime dt);

Variables

int	int myNum = 5;
double	double myDoubleNum = 5.99D;
char	char myLetter = 'D';
bool	bool myBool = true;
string	string myText = "Hello";
float	float value = 6.3F;

Naming Conventions

Class	MyClass
Method	MyMethod
Local Variable	myLocalVariable
Private Variable	_myPrivateVariable
Constant	MyConstant

Assignemnt

=	Simple assignment.
+=	Addition assignment.
-=	Subtraction assignment.
*=	Multiplication assignment.
/=	Division assignment.
%=	Remainder assignment.
&&	AND assignment.
||	OR assignment.

Conditions (cont)

```
for (int i = 0; i < 5; i++) {
  Console.WriteLine(i);
}
```

```
do {...} while (true);
```

Commenting

```
// Single-Line Comment
/* Multiple-Line Comment */
```

Comparision

<
<
<=
>=
==
!=

Type Conversions

Exception Handling

```
try {
  // code that might throw an exception
}
catch (Exception ex) {
  // handle exception }
finally {
  // cleanup code
}
```

Conditions

```
if (condition) {
// if the condition is True
}
else {
// if the condition is False
}
```

```
switch(expression) {
  case x:
   // code block
    break;
  case y:
   // code block
   break;
  default:
  // code block
   break;
}
```

```
while (condition) {
  // code block to be executed
}
```

```
foreach (type variableName in arrayName) {
  // code block to be executed
}
```

ToBoolean
ToByte
ToChar
ToDateTime
ToDecimal
ToDouble
ToInt64
ToInt32
ToInt16
ToSbyte
ToSingle
ToString
ToType
ToUInt16
ToUInt32

Modifiers

abstract	abstract class Shape { ... }
async	private async void Task() { ... }
const	const int X = 0;
event	public event SampleEventHandler SampleEvent;
delegate	public delegate void SampleEventHandler(object sender, SampleEventArgs e;
new	public Random random = new Random();
override	public override void ToString() { ... }

Modifiers (cont)

readonly	private readonly int value = 6;
static	static int = 7;

Other Operators

sizeof()	Returns the size of a data type
typeof()	Returns the type of a class
&	Returns the address of a variable
*	Pointer to a variable
? :	Conditional expression
is	Determines whether an object is of a specific type
as	Cast without raising an exception if the cast fails

Inheritance

```
public class Animal {
  public virtual void MakeSound() {
   Console.WriteLine("Some sound");
   }
}
public class Dog : Animal {
  public override void MakeSound() {
    Console.WriteLine("Bark");
   }
}
```

Webstorm

Double Shift	Search Everywhere Quickly find any file, action, class, symbol, tool window, or setting in WebStorm, in your project, and in the current Git repository.
Ctrl Shift A	Find Action Find a command and execute it, open a tool window, or search for a setting.
Double Ctrl	Run Anything Launch run/debug configurations, run npm and yarn scripts, and reopen recent projects.
AltEnter	Show Context Actions Quick-fixes for highlighted errors and warnings, intention actions for improving and optimizing your code.

Webstorm (cont)

F2 or Shift F2	Navigate between code issues Jump to the next or previous highlighted error.
Ctrl E	View recent files Select a recently opened file from the list.
Ctrl W or Ctrl Shift W	Extend or shrink selection Increase or decrease the scope of selection according to specific code constructs.
Ctrl / or Ctrl Shift /	Add/remove line or block comment Comment out a line or block of code.
Alt F7	Find Usages Show all places where a code element is used across your project.