C# Cheat Sheet

INTRODUCTION

C# is a powerful Object Orientated language, for those coming from Java or C++ you should be able to pick up the syntax for C# quickly. A few points:

- ✓ The language is case-sensitive (So A and a are different)
- ✓ Lines terminate with semi-colons
- ✓ Code is put in code blocks { }
- ✓ Inline comments start with //
- ✓ Block comments start with /* */
- ✓ XML comments start with ///

VARIABLES

To declare a variable you specify the data type and variable name followed by a value.

SYNTAX	EXAMPLE

DataType variableName = value;
 	
NAMING RULES
· Variables must start with underscore or letter
· Variables cannot contain spaces
· variables can contain numbers
· Cannot contain symbols (accept underscore)
	`string Name = "thecodingguys";`
	`int Year = 2013;`
	
	I will use these two variables throughout.

ARRAYS

Arrays are similar to variables, but can hold more than one value.

SYNTAX	EXAMPLE

DataType[ ] ArrayName = { Comma Separated Values } // Array of any size
 	
DataType[] ArrayName = new DataType[3] {Command Separated Values } //Expects 3 values
	`string[] MyGamesOf2013 = {"GTAV", "Battlefield3"};`
	`string[] MyMoveisOf2013 = new string[3] {"The Amazing Spiderman", "The Expendables 2", "Rise of the planet of the apes"};`

Records

Record structures allow you to store multiple data types under one identifier name. You can create an array of them to store lots of data

Syntax	Example

Must be declared outside the of any method as a global
public struct StructName
{
public string field1;
public int field2;
public string field3;
public int field4;
}
 	
Run within a method:
StructName [] ArrayName = new StructName [20];
	public struct Results
	{
	public string hometeam;
	public int hometeamscore;
	public string awayteam;
	public int awayteamscore;
	}
	
	Run within a method:
	Results[] results = new Results[20];

STRINGS - CONCATENATION

Concatenation is done through the + operator.

EXAMPLE	NEW LINE *EXAMPLE*
`Console.WriteLine("Hello " + "World");`	`Console.WriteLine("Hello \n" + "World");`

STRING.FORMAT

Formats an object, you specify the formatting you wish to perform, the following formats an integer and displays the currency symbol.

EXAMPLE

```
Console.WriteLine(string.Format("{0:C}", 5));
```

Depending on your computers regional settings you will see £5.00 displayed (You'll see your countries currency symbol). The 0:C is the formatting we wish to do, in this case it means format the first parameter (0) and show a currency sign.

Random Number Generation

Generate a random number between user defined values

Syntax	Example
Random NAMEOFRANDOM = new Random(); number = NAMEOFRANDOM.Next(value1, value2);	Random r = new Random(); number = r.Next(0, 13);

IF STATEMENTS

if statement is used to execute code based on a condition the condition must evaluate to true for the code to execute.

SYNTAX	*EXAMPLE*

`if (true)`
`{`
 	
`}`
`else`
`{`
 	
`}`
	`if (Year > 2010)`
	`{`
	`Console.WriteLine("Hello World!");`
	`}`
	`else`
	`{`
	`Console.WriteLine("Year is: " + Year);`
	`}`

SWITCH STATEMENT

Similar to the If else statement, however it has these benefits.

- Much easier to read and maintain
- Much cleaner then using nested if else
- It only evaluates one variable

SYNTAX	*EXAMPLE*

`switch (switch_on)`
`{`
`default:`
`}`
	`switch (Year)`
	`{`
	`case 2013:`
	`Console.WriteLine("It's 2013!");`
	`break;`
	`case 2012:`
	`Console.WriteLine("It's 2012!");`
	`break;`
	`default:`
	`Console.WriteLine("It's " + Year + "!");`
	`break;`
	`}`

The break keyword is required as it prevents case falling.

WHILE LOOP

Continuously loops code until the condition becomes false.

SYNTAX	EXAMPLE

`while (true)`
`{`
`}`
	`while (Year >= 2013)`
	`{`
	`if (Year != 2100)`
	`{`
	`Console.WriteLine(Year++);`
	`}`
	`else`
	`{`
	`}`
	`}`
	`break;`

Make sure your condition evaluates to false at some point otherwise the loop is endless and it can result in errors.

FOR LOOP

Similar to the While Loop, but you specify when the loop will end.

SYNTAX	EXAMPLE

`for (int i = 0; i < length; i++)`
`{`
`}`
	`for (int i = 0; i <= 100; i++)`
	`{`
	`Console.WriteLine(i);`
	`}`
	This prints out 1 to 100. The expression can be easily broken down like this: I = 0;
	I Is less than or equal to 100? (True) Increment I by 1
	When I reaches 100 it will stop because I will no longer be less than100 and will equal 100 so the condition is false.

FOR EACH

The for each loop is used to loop around a collection. (Such as an array)

SYNTAX	EXAMPLE

`foreach (var item in collection)`
`{`
`}`
	`foreach (string movie in MyMoveisOf2013)`
	`{`
	`Console.WriteLine(movie);`
	`}`
	Outputs all the elements in the MyMoviesOf2013 array.

EXCEPTION Handling

To catch any exceptions which are likely to occur you use a try catch block.

SYNTAX	EXAMPLE

`try`
`{`
`}`
`catch (Exception)`
`{`
`}`
	`try`
	`{`
	`string result = "k";`
	`Console.WriteLine(Convert.ToInt32(result) +`
	`10);`
	`}`
	`catch (Exception ex)`
	`{`
	`Console.WriteLine(ex.Message);`
	`}`
	The above code results in a format exception, because you can't convert K to a number ☺

METHODS

SYNTAX

```
public void MethodName()
{
//Does not return a value
}

public static void MethodName()
{
//Does not return a value, the class does not need to be initialized
//for this method to be used.
}

public static DataType MethodName()
{
//Requires a value to be returned, class does not need to be initialized for this method to be used.
}
```

EXAMPLE

```
public static void WelcomeUser()
{
Console.WriteLine("Hello Guest!");
}
```

Passing Parameters

```
public static void WelcomeUser(string Name)
{
Console.WriteLine("Hello " + Name + "!");
}
```

Since both methods have the same name and different parameters (One takes no parameters and the other one does) this is said to be an *overloaded method.*

Returning Data

```
public static DateTime Tomorrow()
{
return DateTime.Now.AddDays(1);
}
```

All the examples above are static, this allows me to use the methods without initializing the class. You can read more about

CLASSES

SYNTAX

```
Class MyClassName
{

}
```

EXAMPLE

```
class MyCar
{
        public void Manufacturer(string Manf)
        {
        Console.WriteLine(Manf);
        }
}
```

To use the method in the class, the class must be initialized first.

```
MyCar NewCar = new MyCar(); NewCar.Manufacturer("Audi");
```

If the method was declared static I could simply do this:

```
MyCar.Manufacturer(“Audi”);
```

Static methods are useful, make sure you are using the right design for your classes and methods. A good example is the Math class, to perform simple calculations you do not want to be initializing the class all the time, that’s why most methods are static.