

Introduction

C# is a modern, general-purpose, object-oriented programming language developed by Microsoft. C# was developed by Anders Hejlsberg and his team during the development of .Net Framework. C# is a widely used professional language around the world.

Read more about C#: [http://msdn.microsoft.com/en-us/library/vstudio/z1zx9t92\(v=vs.120\).aspx](http://msdn.microsoft.com/en-us/library/vstudio/z1zx9t92(v=vs.120).aspx)

Where to download C#?

You can download C# at <http://www.visualstudio.com/> Click on the Downloads tab button.

Which version to download?

Download **Visual Studio Express 2013 for Windows Desktop** or **Visual C# 2010 Express**

If you are using other Operating System, go to http://www.mono-project.com/Main_Page

Learning Materials

Tutorials in English: <http://homeandlearn.co.uk>

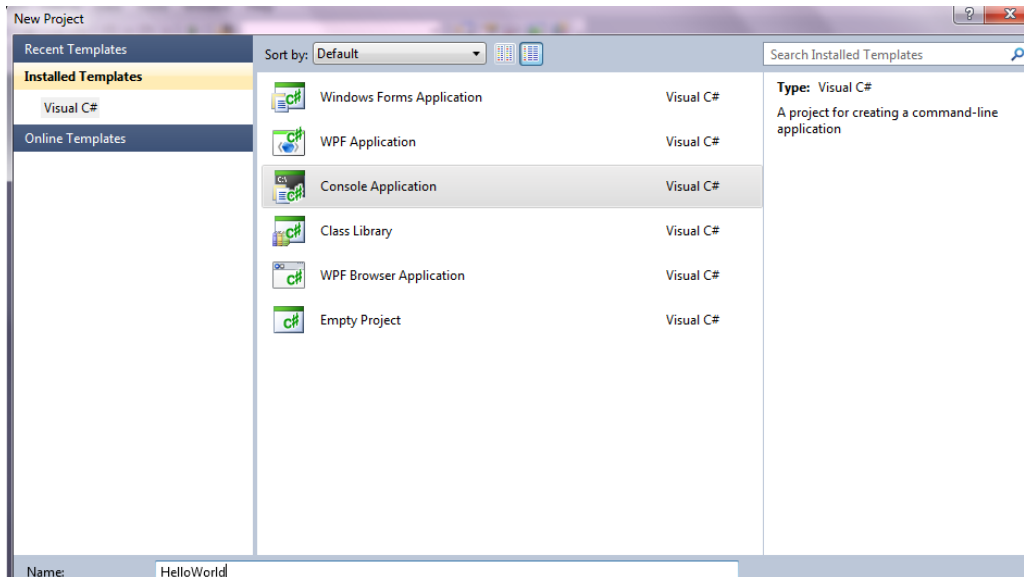
<http://www.completecsharp tutorial.com/index.php>

Tutorials in Dutch: <http://www.ivohbo.be/cursusweb>

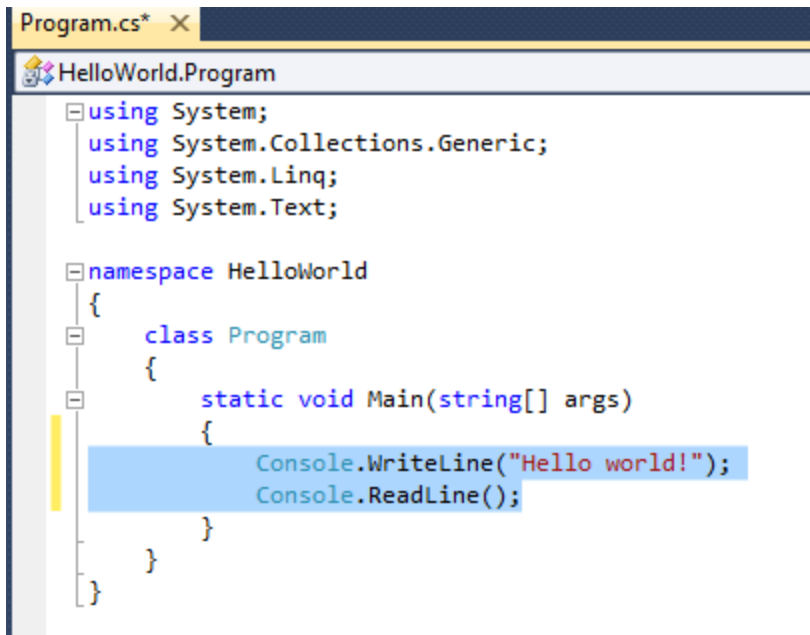
Video Tutorials in English: <http://thenewboston.org>


Getting Started

1) Launch Visual Studio Express. Select **New Project** and select **Console Application**. For the newer version, make sure C# is selected on your left. Name your Project **HelloWorld**.



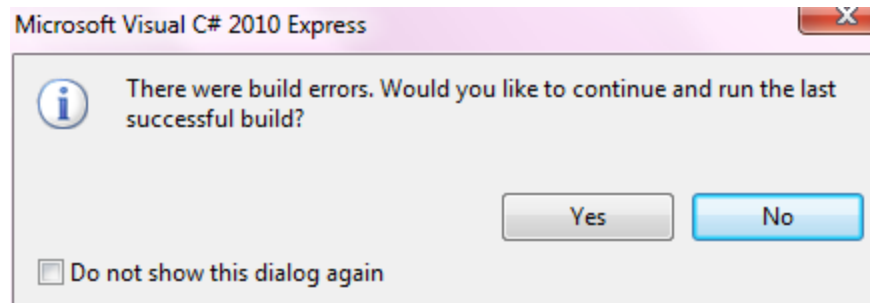
2) Under static void Main and between the { }, Type **Console.WriteLine("Hello world!");**
Console.ReadLine();



3) Now run the program by clicking the green arrow button  or press F5.

If your program is correct, you will see Hello world!

If your program is incorrect, you see a pop up below. Click **No** to check the error.



Some common errors are the Capital Letters. C# is case sensitive. Console.ReadLine is not the same as Console.Readline. All statements and expression must end with a semicolon (;).

4) Save by clicking on the multi disk icon  or press Ctrl+Shift+S.

5) Let us look at the program below and know them better line by line

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HelloWorld //A namespace is a collection of classes
{
    class Program
    {
        static void Main(string[] args)
        {
            /* my first C# program*/
            Console.WriteLine("Hello world!");
            Console.ReadLine();
        }
    }
}
```

- The words in Blue are keywords reserved for C#.
- The words in Green are Comments that will be ignored by the compiler. Comments are words between /*...*/ or after //
- The words in Light Blue are Class names.
- The words in Red between “...” are the words that will appear on the screen.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

- using is used for including C# class library. C# has huge collection of classes and objects. If you want to use those classes then you will have to include their library name in your program.

```
namespace HelloWorld
```

- A namespace is a collection of classes. In the above example, HelloWorld namespace contains the class Program.

```
class Program
```

- The class Program contains the data and method definitions that your program uses. Classes generally would contain more than one method.

```
static void Main(string[] args)
```

- The Main method is the entry point for all C# programs. The Main method states what the class will do when executed.

```
Console.WriteLine("Hello world!");
```

- WriteLine is a method of the Console class defined in the System namespace. This statement prints the message "Hello world!" on the screen.

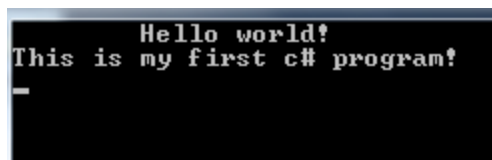
```
Console.ReadLine();
```

- Console.ReadLine(); waits for the user to enter an input and it prevents the screen from running and closing quickly.

Assignment 1

Using the HelloWorld program, change it and try to get an output shown below:

Hint: You may use `\t` or `\n`



```
Hello world!
This is my first c# program!
_
```

Chapter 2 - Variables & Data Types

Introduction

Variables can be used to store temporary data. The data may consist of numbers (integers) or texts (strings). It can also be true or false data (booleans) and many more...

Declaring a variable

Declaring a variable as an integer:

```
int a;
```

- **int** is a data type to declare integer(number).
- **a** is a name you give to the variable.

Note: you cannot use names that are reserved for C#.

See the complete list: <http://msdn.microsoft.com/en-us/library/x53a06bb.aspx>

```
a = 12;
```

- storing data **12** into variable **a**.

```
int a = 12;
```

- you can combine the above statement into one.

What is the output of the program below?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace variable
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 12;

            Console.WriteLine("The number stored in a is {0}.", a);

            Console.ReadLine();
        }
    }
}
```

Data Types

The table below shows some commonly used data types.

Data Types	Size	Values	Example
byte	8 bit	0 to 255	byte a = 8;
short	16 bit	-32,768 to 32,767	short b = 658;
int	32 bit	-2,147,483,648 to 2,147,483,647	int c = 78900;
long	64 bit	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	long d = 9000000;
float	32 bit	-1.5 x 10 ⁴⁵ to 3.4 x 10 ³⁸	float e = 89.567f;
double	64 bit	-5 x 10 ³²⁴ to 1.7 x 10 ³⁰⁸	double f = 100.111;
char	16 bit	A unicode character	char g = 'a';
bool	---	true or false	bool h = true;
string	---	A string of unicode characters	string i = "Hello!1";

See the complete list here: [http://msdn.microsoft.com/en-us/library/cs7y5x0x\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/cs7y5x0x(v=vs.90).aspx)

Assignment 2

Which data type is correct for the following values:

1) Welcome to C#!

2) 260

3) -56

4) 2001.99

5) k1ngd0m

Conversion

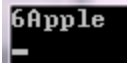
Study the program below:

```
int x = 6;
string y = "Apple";

string result = x + y;

Console.WriteLine(result);
Console.ReadLine();
```

The output of the program is

A screenshot of a console window with a black background. The text '6Apple' is displayed in white. Below the text, there is a white cursor line.

Question:

x is declared as an integer. How can it be displayed as a string?

Answer:

C# has converted x to a string by default. The actual code should be:

```
int x = 6;
string y = "Apple";

string result = x.ToString() + y;

Console.WriteLine(result);
Console.ReadLine();
```

ToString() converts x from int to string.

Instead of string result, I have changed the datatype to int result.

```
int x = 6;
string y = "Apple";

int result = x + y;

Console.WriteLine(result);
Console.ReadLine();
```

There will be an error because C# cannot convert "Apple" to an integer(number).

But what if I change “Apple” to “2”?

```
int x = 6;
string y = "2";

int result = x + y;

Console.WriteLine(result);
Console.ReadLine();
```


There will still be an error because C# do not convert a string to an integer by default. I must convert y to an integer manually in order for the program to work.

```
int x = 6;
string y = "2";

int result = x + int.Parse(y);

Console.WriteLine(result);
Console.ReadLine();
```

int.Parse() converts y to an integer.

Hence, the output is .

Assignment 3

Type out the code below

```
int x = 6;
string y = "2";

string result1 = x.ToString() + y;
int result2 = x + int.Parse(y);

Console.WriteLine(result1);
Console.WriteLine(result2);
Console.ReadLine();
```

What is the output? Explain why.

Chapter 3 - Operator

Introduction

Operators are used to form expression in C#. For example, to calculate the value of a variable.

Operators

The table below shows some commonly used operators.

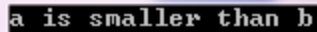
Operator	Description	Example
=	Assigning a value to a variable	a = 12; // a =12
+	Addition	b = 12 + 2; // b = 14
-	Subtraction	c = 12 - 2; // c = 10
*	Multiplication	d = 12 * 2; // d = 24
/	Division	e = 12 / 2; // e = 6
==	Equally	if (a == b) { if the value of a is equal to the value of b, do this code. }
>	Greater than	if (a > b) { if the value of a is greater than the value of b, do this code. }
>=	Greater than or equal to	if (a >= b) { if the value of a is greater than or equal to the value of b, do this code. }
<	Less than	if (a < b) { if the value of a is less than the value of b, do this code. }
<=	Less than or equal to	if (a <= b) { if the value of a is less than or equal to the value of b, do this code. }
&&	Conditional AND	if (a == b && c == d) { if the value of a is equal to the value of b AND the value if c is equal to the value of d, do this code. }
	Conditional OR	if (a == b c == d) { if the value of a is equal to the value of b OR the value if c is equal to the value of d, do this code. }

See the complete list here: <http://msdn.microsoft.com/en-us/library/6a71f45d.aspx>

Example

```
int a = 2, b = 3;
if (a < b)
{
    Console.WriteLine("a is smaller than b");
    Console.ReadLine();
}
else
{
    Console.WriteLine("a is not smaller than b");
    Console.ReadLine();
}
```

The output will be:

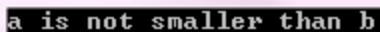


```
a is smaller than b
```

a is 2 and b is 3. The if statement checks if a is smaller than b. Since 2 is smaller than 3, it will print "a is smaller than b" and it will not continue to the else statement.

```
int a = 7, b = 3;
if (a < b)
{
    Console.WriteLine("a is smaller than b");
    Console.ReadLine();
}
else
{
    Console.WriteLine("a is not smaller than b");
    Console.ReadLine();
}
```

The output will be:



```
a is not smaller than b
```

a is 7 and b is 3. The if statement checks if a is smaller than b. Since 7 is greater than 3, it will go to the else statement and print "a is not smaller than b".

Assignment 4

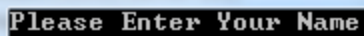
Write a program that display "The number is within range!" if the number is **between 1 to 5**, else display "The number is out of range!".

Chapter 4 - Getting input from the user

Introduction

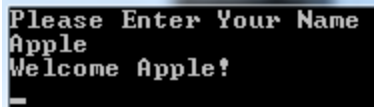
The program interacts with the user in many ways, below is a program that ask for the user's name.

Program ask for user's name



```
Please Enter Your Name
```

After the user entered Apple, the program displays Welcome Apple!



```
Please Enter Your Name
Apple
Welcome Apple!
_
```

Let see the code below:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace input
{
    class Program
    {
        static void Main(string[] args)
        {
            string name;

            Console.WriteLine("Please Enter Your Name");

            name = Console.ReadLine();

            Console.WriteLine("Welcome {0}!", name);

            Console.ReadLine();
        }
    }
}
```

Explanation

```
string name;
```

- Declare name as a string data type.

```
Console.WriteLine("Please Enter Your Name");
```

- Print message

```
name = Console.ReadLine();
```

- Console.ReadLine() accepts the user's input and store it in name.
- C# accepts string value by default.

```
Console.WriteLine("Welcome {0}!", name);
```

- Display output.

Assignment 5

Write a program that calculate the area of rectangle. The user can enter the width and the height of the rectangle. Display the output in **two decimal space**.

The output should look like this if the width is 23.45 and the height is 30.34.

```
Please Enter the width:
23.45
Please Enter the height:
30.34
The area of rectangle is 711.47cm
```

Hint: You may use double.Parse() and string.Format()

Assignment 6

Write a program that calculate the user's age. The user will enter his year of birth and the program will display his age. Note: The program should get the current year from the computer's date and time.

In the year 2014, the output should look like this:

```
Please Enter your Year of Birth:
1994
Your age is 20.
```

Hint: You may use int.Parse() and DateTime()

Console.ReadKey()

The above example we used Console.ReadLine(). Let us look at Console.ReadKey().

The main differences between the two methods:

	Console.ReadLine()	Console.ReadKey()
Default Data Type	String	Char (KeyChar)
Accepts Value	Accepts a line of words	Accepts ONE console key or Alt, Ctrl, Shift + a console key
After user keys in an input	Waits for the user to press Enter before executing	Executes immediately

There are three types of keys: Key, KeyChar and Modifiers.

The program below shows the output of Key, KeyChar and Modifiers.

```
ConsoleKeyInfo keyName = new ConsoleKeyInfo();  
keyName = Console.ReadKey();  
Console.WriteLine("Key is {0}, KeyChar is {1}, Modifier is {2}", keyName.Key, keyName.KeyChar, keyName.Modifiers);  
Console.ReadLine();
```

Alt g is pressed and the output is:

```
gKey is G, KeyChar is g, Modifier is Alt
```

Type out the program above and try different keys to see the different outputs.

Note that pressing a modifier key by itself will not cause the ReadKey method to return.

Explanation

```
ConsoleKeyInfo keyName = new ConsoleKeyInfo();
```

- Initializes a new instance of the ConsoleKeyInfo.

```
keyName = Console.ReadKey();
```

- Access the Readkey() method.

```
Console.WriteLine("Key is {0}, KeyChar is {1}, Modifier is {2}", keyName.Key, keyName.KeyChar, keyName.Modifiers);
```

- Print out Key, KeyChar and Modifiers.

Assignment 7

Using Console.ReadKey(), write a program that guess the user's lucky number 5. If the input is 5, it will display "Correct! My lucky number is 5!" if the input is a number other than 5, display "<input> is incorrect." if the input is something else display "<input> is incorrect. Please key in a number."

The output should look like this:

if the user keys in 5:

```
What is my lucky number?  
Correct! My lucky number is 5!
```

if the user keys in a number other than 5:

```
What is my lucky number?  
8 is incorrect.
```

if the user keys in something else:

```
What is my lucky number?  
k is incorrect. Please key in a number.
```

Reminder: Console.ReadKey() accepts **char**.

Chapter 5 - Selection Statements

Introduction

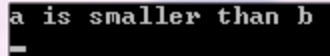
Selection statements decide which codes to execute according to the condition.

If Statement

If statement is not new to you. Here is a recap.

```
int a = 2, b = 3;
if (a < b)
{
    Console.WriteLine("a is smaller than b");
    Console.ReadLine();
}
else
{
    Console.WriteLine("a is not smaller than b");
    Console.ReadLine();
}
```

The output is:



```
a is smaller than b
_
```

Explanation

```
int a = 2, b = 3;
```

- declare a and b as an integer and assign the value 2 and 3.

```
if (a < b)
```

- the if statement checks if a is smaller than b
- if the condition is true, print “a is smaller than b”

```
else
```

- if the condition is not true, print “a is not smaller than b”

Conditional Operator

Conditional Operator is an Operator.

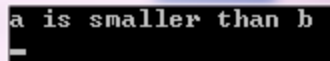
If there are only 2 outputs, we can shorten the above code using the Conditional Operator.

```
int a = 2, b = 3;

string result = (a < b) ? "a is smaller than b" : "a is not smaller than b";

Console.WriteLine(result);
Console.ReadLine();
```

The output is:



```
a is smaller than b
_
```

Explanation

```
string result = (a < b) ? "a is smaller than b" : "a is not smaller than b";
```

- the conditional operator checks if a is smaller than b. If it is true, assign “a is smaller than b” to string result.
- If it is not true, assign “a is not smaller than b” to string result.

Assignment 8

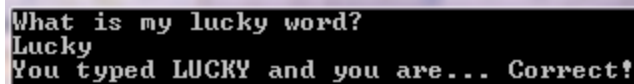
Using the Conditional Operator, write a program that guess the user’s lucky word.

If the input is lucky, it will display “You typed LUCKY and you are... Correct!”

If the input is something else, it will display “You typed APPLE and you are... Wrong!”

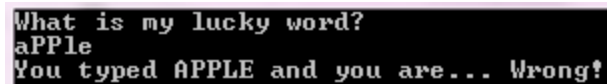
The output should look like this:

if the user keys in lucky:



```
What is my lucky word?
Lucky
You typed LUCKY and you are... Correct!
```

if the user keys in something else:



```
What is my lucky word?
aPPle
You typed APPLE and you are... Wrong!
```

Hint: You may use ToUpper()

Switch Statement

Switch statement is useful when there are multiple selections.

```
Console.WriteLine("The Switch Test");
string userValue = Console.ReadLine();
switch (userValue)
{
    case "1":
        Console.WriteLine("Case 1");
        Console.ReadLine();
        break;
    case "2":
        Console.WriteLine("Case 2");
        Console.ReadLine();
        break;
    case "3":
        Console.WriteLine("Case 3");
        Console.ReadLine();
        break;
    default:
        Console.WriteLine("Default case");
        Console.ReadLine();
        break;
}
```

The output is:

1 is pressed

```
The Switch Test
1
Case 1
```

2 is pressed

```
The Switch Test
2
Case 2
```

3 is pressed

```
The Switch Test
3
Case 3
```

4 is pressed

```
The Switch Test
4
Default case
```

Explanation

`switch (userValue)`

- switch reads the userValue
- if the userValue is 1, it will go to `case "1":` and execute the code under case 1.
- if the userValue is 2, it will go to `case "2":` and execute the code under case 2.
- if the userValue is 3, it will go to `case "3":` and execute the code under case 3.
- if the userValue is not 1, 2 or 3, it will go to `default:` and execute the code under default.

`break;`

- break will end the switch statement.

Assignment 9

Using the Switch Statement, write a program that will ask the user for the drink size and display the cost accordingly.

The output should look like this:

The program asks for a drink size

```
Drink sizes: 1=small 2=medium 3=large  
Please enter your selection:
```

If the user enters **1 or small**:

```
Drink sizes: 1=small 2=medium 3=large  
Please enter your selection: 1  
Please insert 25 cents.  
Thank you. Please come again.
```

If the user enters **2 or medium**:

```
Drink sizes: 1=small 2=medium 3=large  
Please enter your selection: Medium  
Please insert 50 cents.  
Thank you. Please come again.
```

If the user enters **3 or large**:

```
Drink sizes: 1=small 2=medium 3=large  
Please enter your selection: 3  
Please insert 75 cents.  
Thank you. Please come again.
```

if the user enters something else:

```
Drink sizes: 1=small 2=medium 3=large  
Please enter your selection: 8  
Invalid selection. Please enter 1, small, 2, medium, 3 or large.
```

Chapter 6 - Iteration

Introduction

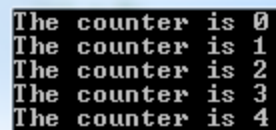
Iteration statements create loops that will execute a number of times until the criteria is reached.

For Loop

```
for (int i = 0; i < 5; i++)
{
    Console.WriteLine("The counter is {0}", i);
}

Console.ReadLine();
```

The output is:



```
The counter is 0
The counter is 1
The counter is 2
The counter is 3
The counter is 4
```

Explanation

```
for (int i = 0; i < 5; i++)
```

- declare i as an integer and assign the value 0.

```
for (int i = 0; i < 5; i++)
```

- the program checks if i is smaller than 5.
- if i is smaller than 5, it will execute the code

```
{ Console.WriteLine("The counter is {0}", i); }
```

```
for (int i = 0; i < 5; i++)
```

- after it executes the code, it will +1 to i
- i++ is the same as i = i+1;

```
for (int i = 0; i < 5; i++)
```

- the program checks again if i is smaller than 5.
- it executes the code until i is 5.

While Loop

```
int i = 0;
while (i < 5)
{
    Console.WriteLine("The counter is {0}", i);
    i++;
}

Console.ReadLine();
```

The output is:

```
The counter is 0
The counter is 1
The counter is 2
The counter is 3
The counter is 4
```

Explanation

`while (i < 5)`

- while i is smaller than 5, the program will execute the code

```
{
    Console.WriteLine("The counter is {0}", i);
    i++;
}
```

- the program executes the code until i is 5.

Do While Loop

```
int i = 0;
do
{
    Console.WriteLine("The counter is {0}", i);
    i++;
} while (i < 5);

Console.ReadLine();
```

The output is:

```
The counter is 0
The counter is 1
The counter is 2
The counter is 3
The counter is 4
```

Explanation

```
do
{
    Console.WriteLine("The counter is {0}", i);
    i++;
} while (i < 5);
```

- execute the code first

```
do
{
    Console.WriteLine("The counter is {0}", i);
    i++;
} while (i < 5);
```

- while i is smaller than 5, executes the code again.
- the program executes the code until i is 5.

Assignment 10

Write a program that will print 1 to 10 except 4.

The output should look like this:

```
The counter is 1
The counter is 2
The counter is 3
The counter is 5
The counter is 6
The counter is 7
The counter is 8
The counter is 9
The counter is 10
```

Hint: You may use continue.

Assignment 11

Using Console.ReadKey(), create a Menu List and a program that will display "Menu 1" if the user press 1, display "Menu 2" if the user press 2, exit if the user press 3 and display "Invalid Selection. Please enter 1, 2 or 3" if the user press something else. The program will wait for the user to enter an input until the user press 3 to exit.

The output should look like this:

Waiting for the user to enter a number.

```
*** MENU ***
1. Display Menu 1
2. Display Menu 2
3. Exit
```

if the user press 1, 2 and 4.

```
*** MENU ***  
1. Display Menu 1  
2. Display Menu 2  
3. Exit  
Menu 1  
Menu 2  
Invalid Selection. Please enter 1, 2 or 3
```

Chapter 7 - Arrays

Introduction

Arrays are useful when you need to declare multiple variable. It stores value in the index position starting with 0.

Declaration and Initialization of Arrays.

```
int[] num = new int[5];
```

```
num[0] = 8;  
num[1] = 15;  
num[2] = 16;  
num[3] = 25;  
num[4] = 42;
```

Explanation

```
int[] num = new int[5];
```

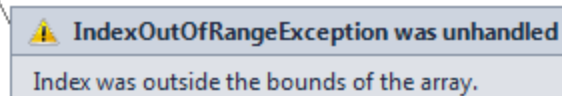
- declare num as an int array and set array's length to 5.

```
num[0] = 8;  
num[1] = 15;  
num[2] = 16;  
num[3] = 25;  
num[4] = 42;
```

- array's index position always start at **0**
- stores value 8 in index position 0
- stores value 15 in index position 1
- stores value 16 in index position 2
- stores value 25 in index position 3
- stores value 42 in index position 4
- notice that the array length is 5 but the index position stops at 4. That is because the array's index position starts at **0**.

```
int[] num = new int[5];
```

```
num[0] = 8;  
num[1] = 15;  
num[2] = 16;  
num[3] = 25;  
num[4] = 42;  
num[5] = 87;
```



- If I try to add "num[5] = 87;" It will show me an error message saying that index is out of range because I have setted the array's length to 5 but I am trying to add a 6th value.

```
int[] num = new int[5] { 8, 15, 16, 25, 42 };
```

- You can shorten the above code to one line.

Displaying Arrays - using For Loop

```
int[] num = new int[5] { 8, 15, 16, 25, 42 };
```

```
for (int i = 0; i < num.Length; i++)  
{  
    System.Console.WriteLine(num[i]);  
}  
System.Console.ReadLine();
```

The output is:

```
8  
15  
16  
25  
42
```

Explanation

```
for (int i = 0; i < num.Length; i++)
```

- num.Length will get the length of the array which is 5.

Displaying Arrays - using Foreach Loop

```
int[] num = new int[5] { 8, 15, 16, 25, 42 };

foreach (int element in num)
{
    Console.WriteLine(element);
}
Console.ReadLine();
```

The output is:

```
8
15
16
25
42
```

Explanation

```
foreach (int element in num)
```

- foreach reads every int element (the value of the arrays) in num.

```
Console.WriteLine(element);
```

- displays all the element.

Assignment 12

Using arrays, write a program that will ask the user to enter 5 names and show the output.

The output should look like this:

The program asks for 5 names.

```
Enter 5 names.
name 1:
```

The program will display the names the user entered.

```
Enter 5 names.
name 1:Merry
name 2:John
name 3:Tim
name 4:Matt
name 5:Jeff

Merry
John
Tim
Matt
Jeff
```


Assignment 13

Write a program that will ask the user to enter 5 numbers and show the output. The program will check if the input is a number.

The output should look like this:

The program asks for 5 numbers.

```
Enter 5 numbers.  
number 1: _
```

The program will check if the input is a number. It will ask for an input until a number is entered.

```
Enter 5 numbers.  
number 1:67  
number 2:3  
number 3:23  
number 4:h  
Pleae enter a number.  
number 4:t  
Pleae enter a number.  
number 4:d  
Pleae enter a number.  
number 4:h  
Pleae enter a number.  
number 4:
```

The program will display the numbers the user entered.

```
Enter 5 numbers.  
number 1:67  
number 2:3  
number 3:23  
number 4:h  
Pleae enter a number.  
number 4:t  
Pleae enter a number.  
number 4:d  
Pleae enter a number.  
number 4:h  
Pleae enter a number.  
number 4:7#  
Pleae enter a number.  
number 4:8  
number 5:9  
  
67  
3  
23  
8  
9
```

Hint: You may use `int.TryParse(<string>, out <int>)`

Multidimensional Arrays

```
int[,] numbers = new int[3, 2];
numbers[0,0] = 1;
numbers[0,1] = 2;
numbers[1,0] = 3;
numbers[1,1] = 4;
numbers[2,0] = 5;
numbers[2,1] = 6;

Console.WriteLine(numbers[1,0]);
Console.ReadLine();
```

The output is:

3

Explanation

```
int[,] numbers = new int[3, 2];
```

- declare numbers as an int array and set array's length to 3 rows and 2 columns.
- below shows their index position.

index position [0,0]	index position [0,1]
index position [1,0]	index position [1,1]
index position [2,0]	index position [2,1]

```
numbers[0,0] = 1;
numbers[0,1] = 2;
numbers[1,0] = 3;
numbers[1,1] = 4;
numbers[2,0] = 5;
numbers[2,1] = 6;
```

- stores value 1 in index position [0,0]
- stores value 2 in index position [0,1]
- stores value 3 in index position [1,0]
- stores value 4 in index position [1,1]
- stores value 5 in index position [2,0]
- stores value 6 in index position [2,1]

```
Console.WriteLine(numbers[1,0]);
```

- displays the value stored in index position [1,0] which is 3.

```
int[,] numbers = new int[3, 2] { { 1, 2 }, { 3, 4 }, { 5, 6 } };
```

- You can shorten the above code to one line.

Assignment 14

```
int[,] numbers = new int[,] { { 3, 5, 7 }, { 9, 11, 13 }, { 15, 17, 19 } };  
Console.WriteLine(numbers[1, 2]);
```

Read the codes above, what is the output?

Note: DO NOT type the codes in the Visual Studio C# to get the answer.

Chapter 8 - Classes and Methods

Introduction

Classes are blueprints or recipes that is used for declaring an object. Methods perform a task that can be called multiple times.

A Class

```
class DisplayName
{
}
```

Explanation

```
class DisplayName
```

- declare DisplayName as a Class

A Method in a Class

```
class DisplayName
{
    public void printname()
    {
        Console.WriteLine("Merry");
    }
}
```

Explanation

```
public void printname()
```

- public - it is accessible in another class
- void - it do not return any value
- the method name is printname
- () - it do not take in any parameter

Calling a Method

```
namespace ClassAndMethod
{
    class DisplayName
    {
        public void printname()
        {
            Console.WriteLine("Merry");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            DisplayName name = new DisplayName();

            name.printname();
            Console.ReadLine();
        }
    }
}
```

output is:

A screenshot of a console window showing the output of the program, which is the word "Merry".

Explanation

- in namespace ClassAndMethod, there are 2 classes. DisplayName and Program.

```
DisplayName name = new DisplayName();
```

- Initializes name as a new instance of DisplayName();

```
name.printname();
```

- call the method printname();

Static Method

```
namespace ClassAndMethod
{
    class DisplayName
    {
        public static void printname()
        {
            Console.WriteLine("Merry");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            DisplayName.printname();
            Console.ReadLine();
        }
    }
}
```

The output is:

A screenshot of a console window showing the output of the program, which is the word "Merry".

Explanation

```
public static void printname()
```

- declaring a method as static

```
DisplayName.printname();
```

- If a method is declared as static, the method can be called directly.

Parameter

Parameter is a way to pass value to the method and the method returns the result. There are two types of Parameter. **Value type parameter** and **Reference type parameter**.

Value type parameter

```
namespace ClassAndMethod
{
    class Count
    {
        public int sum(int add1, int add2)
        {
            int result = add1 + add2;
            return result;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Adding: \nPlease enter the first number");
            int num1 = int.Parse(Console.ReadLine());
            Console.WriteLine("Please enter the second number");
            int num2 = int.Parse(Console.ReadLine());

            Count add = new Count();
            int total = add.sum(num1, num2);
            Console.WriteLine("The sum is {0}.", total);
            Console.ReadLine();
        }
    }
}
```

The output is:

The program asks for the first number.

```
Adding:
Please enter the first number
```

The program shows the sum of the two numbers.

```
Adding:
Please enter the first number
65
Please enter the second number
88
The sum is 153.
```

Explanation

```
public int sum(int add1, int add2)
```

- public - it is accessible in another class
- int - returns int value
- the method name is sum
- (int add1, int add2) - take in 2 int parameters

```
Count add = new Count();
```

- Initializes add as a new instance of Count();

```
int total = add.sum(num1, num2);
```

- gets the result returned from the method

```
int total = add.sum(num1, num2);
```

- call the method sum and pass num1 and num2 to it

Assignment 15

Edit the above method `public int sum(int add1, int add2)` to a **static method** and change the codes in `static void Main(string[] args)` accordingly.

Reference type parameter

Reference is a type of variable that holds the memory address of the value. The example below has a value type parameter that doesn't work.

```
namespace ClassAndMethod
{
    class Count
    {
        public void add(int num)
        {
            num++;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Please enter a number");
            int number = int.Parse(Console.ReadLine());

            Count add1 = new Count();
            add1.add(number);

            Console.WriteLine("number is {0}.", number);
            Console.ReadLine();
        }
    }
}
```


The output is:

```
Please enter a number
3
number is 3.
```

Question

Why is the value in number not 4?

Explanation

```
public void add(int num)
```

- note that the method did not return any value as it is **void**. Hence, number did not +1.

How to solve the problem?

We have to change it to a Reference type parameter.

```
namespace ClassAndMethod
{
    class Count
    {
        public void add(ref int num)
        {
            num++;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Please enter a number");
            int number = int.Parse(Console.ReadLine());

            Count add1 = new Count();
            add1.add(ref number);

            Console.WriteLine("number is {0}.", number);
            Console.ReadLine();
        }
    }
}
```

The output is:

```
Please enter a number
3
number is 4.
```

Explanation

```
public void add(ref int num)
```

- take in a reference parameter instead of a value parameter.

```
add1.add(ref number);
```

- passes a reference parameter to the method.
- ref number is **NOT** 3.
- ref number is the variable that holds the **memory address** in a computer of the value 3.

Assignment 16

Using methods, create a program that:

- asks for 2 numbers. (checks if it is a double)
- asks what to do with it.

press 1 for Addition & Subtraction

press 2 for Addition & Multiplication

press 3 for Addition & Division

-if the user did not press 1, 2 or 3, the program will display:

Invalid Selection. Please enter 1, 2 or 3

-the program will loop until 1, 2 or 3 is entered.

The output should look like this:

program asks for number 1

```
Please enter number 1
```

program checks if it is a number (g is entered)

```
Please enter number 1
g
Invalid. Please enter a number
```

user is able to enter decimals. (345,899 is entered) program asks for number 2

```
Please enter number 1
g
Invalid. Please enter a number
345,899
Please enter number 2
```

program checks if it is a number (> is entered)

```
Please enter number 1
g
Invalid. Please enter a number
345,899
Please enter number 2
>
Invalid. Please enter a number
```

user is able to enter decimals. (22,66 is entered) asks the user what to do with the numbers

```
Please enter number 1
g
Invalid. Please enter a number
345,899
Please enter number 2
>
Invalid. Please enter a number
22,66

What do you want to do with the numbers?
1. Addition & Subtraction
2. Addition & Multiplication
3. Addition & Division
```

program checks if it is 1, 2 or 3 (j is entered)

```
Please enter number 1
g
Invalid. Please enter a number
345,899
Please enter number 2
>
Invalid. Please enter a number
22,66

What do you want to do with the numbers?
1. Addition & Subtraction
2. Addition & Multiplication
3. Addition & Division
j
Invalid Selection. Please enter 1, 2 or 3
```

program keeps asking the user for an input until 1, 2 or 3 is entered (7 is entered)

```
Please enter number 1
g
Invalid. Please enter a number
345,899
Please enter number 2
>
Invalid. Please enter a number
22,66

What do you want to do with the numbers?
1. Addition & Subtraction
2. Addition & Multiplication
3. Addition & Division
j
Invalid Selection. Please enter 1, 2 or 3
7
Invalid Selection. Please enter 1, 2 or 3
```

program calculates Addition & Division and Exit (3 is entered)

```
Please enter number 1
g
Invalid. Please enter a number
345,899
Please enter number 2
>
Invalid. Please enter a number
22,66

What do you want to do with the numbers?
1. Addition & Subtraction
2. Addition & Multiplication
3. Addition & Division
j
Invalid Selection. Please enter 1, 2 or 3
7
Invalid Selection. Please enter 1, 2 or 3
3

Addition:      345,899 + 22,66 = 368,56
Division:     345,899 / 22,66 = 15,26
```

New program calculates Addition & Subtraction and Exit (1 is entered)

```
Please enter number 1
456677,2211
Please enter number 2
223,4

What do you want to do with the numbers?
1. Addition & Subtraction
2. Addition & Multiplication
3. Addition & Division
1

Addition:      456677,2211 + 223,4 = 456.900,62
Subtraction:   456677,2211 - 223,4 = 456.453,82
```

New program calculates Addition & Multiplication and Exit (2 is entered)

```
Please enter number 1
34,55
Please enter number 2
11,88888

What do you want to do with the numbers?
1. Addition & Subtraction
2. Addition & Multiplication
3. Addition & Division
2

Addition:      34,55 + 11,88888 = 46,44
Multiplication: 34,55 * 11,88888 = 410,76
```

C# Chapter 9 - Exception Handling

Introduction

Exceptions are unforeseen errors that happen in your program. Most of the time you can handle the error, for example, validating user input. However, there are unforeseen errors, for example, when the system runs out of memory or cannot find a file.

Try-Catch

```
double result;
string value = "hell0";

try
{
    result = Double.Parse(value);
    Console.WriteLine("Converted {0} to {1}", value, result);
}
catch (FormatException e)
{
    Console.WriteLine("Please enter a valid format. Error message: {0}", e.Message);
}
catch (OverflowException e)
{
    Console.WriteLine("{0} is outside the range of double. Error message: {1}", value, e.Message);
}
catch (Exception e)
{
    Console.WriteLine("There is an error: {0}", e.Message);
}
Console.ReadLine();
```

The output is:

```
Please enter a valid format. Error message: Input string was not in a correct format.
```

Try changing the value "hell0" and see the different outputs.

Explanation

```
try
{
    result = Double.Parse(value);
    Console.WriteLine("Converted {0} to {1}", value, result);
}
```

- try to convert value to double.

```
catch (FormatException e)
{
    Console.WriteLine("Please enter a valid format. Error message: {0}", e.Message);
}
```

- If there is an error, it will go to catch. This catch checks if the error is the wrong format.

```
catch (OverflowException e)
{
    Console.WriteLine("{0} is outside the range of double. Error message: {1}", value, e.Message);
}
```

- This catch checks if the error is out of range.

```
catch (Exception e)
{
    Console.WriteLine("There is an error: {0}", e.Message);
}
```

- This catch checks for a general error.

Assignment 17

Using try-catch, continue from the code below.

```
string[] names = new string[5] { "Merry", "John", "Tim", "Matt", "Jeff" };
```

The user enters the length of the array. The system will check if the input is valid.

If the input is valid, it displays the names.

If the input is invalid, it displays the names and/or the error message.

The output should look like this:

The program asks for the length of the array.

```
Please enter the length of the array
```

The user entered 5

```
Please enter the length of the array
5
Merry
John
Tim
Matt
Jeff
```

The user entered 9

```
Please enter the length of the array
9
Merry
John
Tim
Matt
Jeff
Out of range Error. Error Message: Index was outside the bounds of the array.
```

The user entered hello

```
Please enter the length of the array
hello
Format Error. Please enter a number. Error Message: Input string was not in a correct format.
```

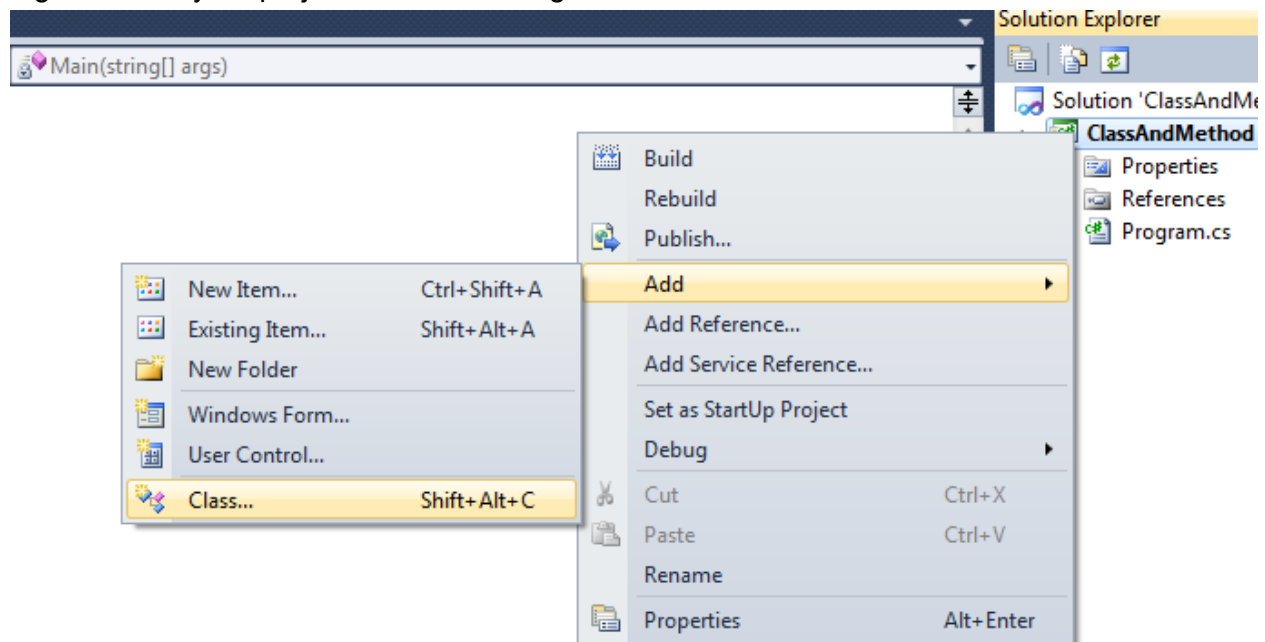
C# Chapter 10 - More about Classes

Adding a new class in a different file.

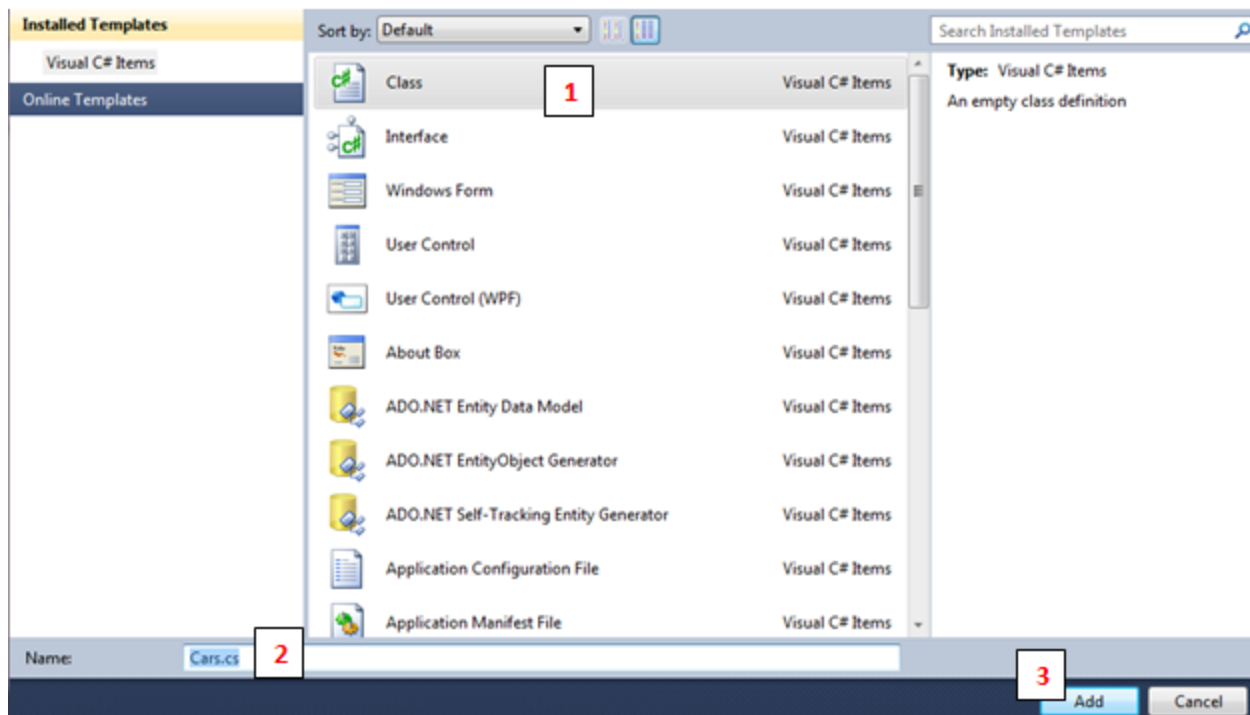
When your program gets bigger, you may have more than one class. Adding a new class in a different file allows you to organise your codes.

How to add a new class in a different file

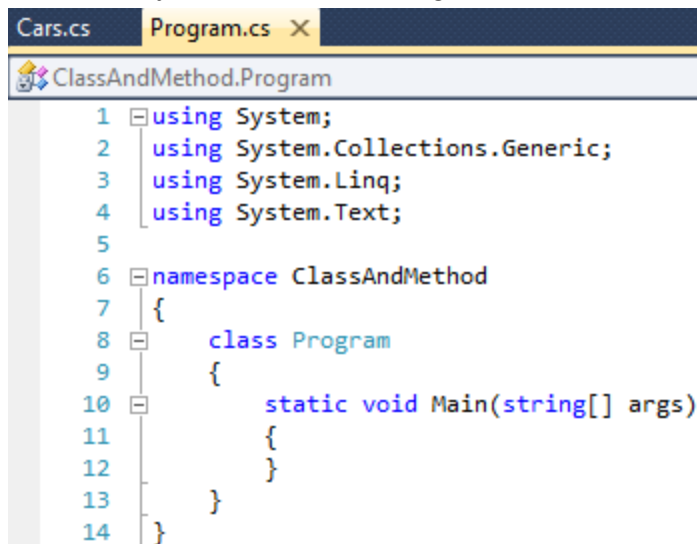
Right click on your project name on the right > Add > Class



- 1) Click Class
- 2) Rename it to Cars.cs
- 3) Click Add.



Notice that you have 2 tabs. Program.cs and Cars.cs



Both classes have the same namespace name.

Understanding Public and Private

Public	Public members are not restricted to anyone. Anyone who can see them can also access them.
Private	Private members can only be accessed within the class that contains them.

In Cars.cs

```
class Cars
{
    private int id;
    public string make;
    double price;

    public void PublicPrint()
    {
        Console.WriteLine("I am public");
    }
}
```

Explanation

`private int id;`

- `int id` is private. It can only be accessed within the class `Cars`.

`public string make;`

- `string make` is public. It can be accessed outside class `Cars`.

`double price;`

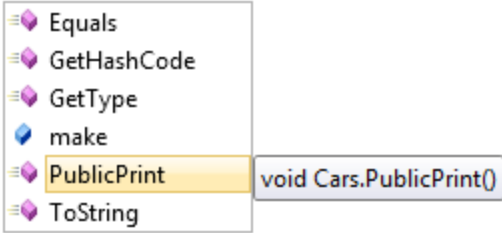
- in C# a field is **default private**. Hence, `double price` is private.

```
public void PublicPrint()
```

- this is a public method. It can be accessed outside class Cars.

In Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Cars car1 = new Cars();
        car1.
    }
}
```



Explanation

- in the class Program, we can only see the public variable **make** and method **PublicPrint**.
- we cannot see the private variables id and price.

Properties Get and Set

In order to access the private variables, we use the properties get, set.

In Cars.cs

```
public int Id
{
    get
    {
        return this.id;
    }
    set
    {
        this.id = value;
    }
}

public double Price
{
    get
    {
        return this.price;
    }
    set
    {
        this.price = value;
    }
}
```

Explanation

```
public int Id
```

- public property int Id

```
get  
{  
    return this.id;  
}
```

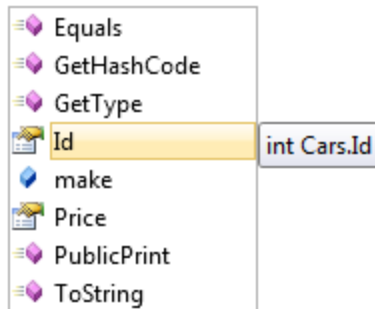
- returns the value in id.
- this.id - this refers to the current class Cars.

```
set  
{  
    this.id = value;  
}
```

- set the value of id to the value assigned to it.

In Program.cs

```
class Program  
{  
    static void Main(string[] args)  
    {  
        Cars car1 = new Cars();  
        car1.
```



Explanation

- now we can see the property of Id and Price.

Constructors

A constructor has the same name as the class and gets executed when its object is created. Usually, we put the initialization code in the constructor.

In Cars.cs

```
public Cars()
{
}
```

Explanation

- The above is a constructor that does nothing.
- If there is no constructor in a class, a default “does nothing” constructor will be created by c#

Overloading Constructors

We can have more than one constructors, constructors with different sets of parameters.

In Cars.cs

```
public Cars()
{
}

public Cars(int id1)
{
    this.id = id1;
    this.make = "unknow";
    this.price = 0.0;
}

public Cars(int id1, string make1, double price1)
{
    this.id = id1;
    this.make = make1;
    this.price = price1;
}
```

Explanation

- There are 3 Constructors. Each contains different parameters.
- `public Cars()`
- `public Cars(int id1)`
- `public Cars(int id1, string make1, double price1)`

In Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Cars car1 = new Cars();

        Cars car2 = new Cars(2);

        Cars car3 = new Cars(3, "Toyota", 5000.90);
    }
}
```

Explanation

- Depending on the parameters passed to the constructor, C# will decide which constructor to be called.
- car1 calls `public Cars()`
- car2 calls `public Cars(int id1)`
- car3 calls `public Cars(int id1, string make1, double price1)`

In Program.cs

```
Console.WriteLine("In car1 Id:{0} Make:{1} Price:{2}", car1.Id, car1.make, car1.Price);
Console.WriteLine("In car2 Id:{0} Make:{1} Price:{2}", car2.Id, car2.make, car2.Price);
Console.WriteLine("In car3 Id:{0} Make:{1} Price:{2}", car3.Id, car3.make, car3.Price);
Console.ReadLine();
```

Explanation

- Let us see the value in car1, car2 and car3.

The output is:

```
In car1 Id:0 Make: Price:0
In car2 Id:2 Make:unknown Price:0
In car3 Id:3 Make:Toyota Price:5000.9
```

Changing the value

In Program.cs

```
car3.Price = 6000.99;
Console.WriteLine("In car3 Id:{0} Make:{1} Price:{2}", car3.Id, car3.make, car3.Price);
Console.ReadLine();
```

Explanation

- change the value of price in car3

The output is:

```
In car1 Id:0 Make: Price:0
In car2 Id:2 Make:unknow Price:0
In car3 Id:3 Make:Toyota Price:5000.9
In car3 Id:3 Make:Toyota Price:6000.99
```

Assignment 18

1) Create a new class call Customers.

2) In class Customers, create 3 fields:

private int id

private string name

private int telephone

3) Create a **public static** method call **Initialize()** in class Customers.

It will initialize the following and add it to an Array:

	id	name	telephone
customer1	1	Merry	612345678
customer2	2	John	612345677
customer3	3	Tim	612345666
customer4	4	Matt	612345555
customer5	5	Jeff	612344444

Hint: You may use ArrayList(),

Add **using System.Collections;** in the using statements to use ArrayList().

4) Display all the objects on the screen.

The output should look like this:

```
ID:1 Name:Merry Telephone:612345678
ID:2 Name:John Telephone:612345677
ID:3 Name:Tim Telephone:612345666
ID:4 Name:Matt Telephone:612345555
ID:5 Name:Jeff Telephone:612344444
```

Congratulations!

You have completed C# Basic! You are now ready for SoundSharp Syntax.

Go to the school's server:

Applicatie Ontwikkelaar > C# > Opdrachten Syntax > Opdracht 1 - 13.