C# Learning

1 C# vs .NET

- C# is a programming language while .NET is a framework for building applications on Windows. There are different languages that can be used to build .NET.
- .NET combines of CLR and class library.
- After the C# program is run,it will be converted into IL Code(Intermediate Language Code). Then it will be translated into the Native Code (Machine Code) to run it.

2 Architecture of .NET applications

- Application combines of class. Class is a container that contains data and methods (behaviors).
- As numbers is growing, we need the namespace to store the related class for our code.
- An assembly is a container for related namespace. It can be .exe file or .dll file. When building an application, the application will build up several assembly.

3 Primitive types and expressions

3.1 Variables and constants

- variables: is the name we give to a storage in the memory
- constant: is an immutable value
Here is the way to declare variables

```
int number;
int Number = 1;
const float Pi = 3.14f;
```

Identifiers cannot start with numbers, we need to use the characters and it cannot be a keyword. For declaring variables, we should give it a meaningful one. Some naming conventions:
- camelCase: firstName
- PascalCase: FirstName

- Hungarian Notation: strFirstName
Name variables convention: local variable: camel case, constant: pascal case
Some data types in C#:

	C# Type	.NET Type	Bytes	Range
Integral Numbers	**byte**	Byte	1	0 to 255
	short	Int16	2	-32,768 to 32,767
	int	Int32	4	-2.1B to 2.1B
	long	Int64	8	…
Real Numbers	**float**	Single	4	-3.4×10^{38} to 3.4×10^{38}
	double	Double	8	…
	decimal	Decimal	16	-7.9×10^{28} to 7.9×10^{28}
Character	**char**	Char	2	Unicode Characters
Boolean	**bool**	Boolean	1	True / False

Remember that by default, C# will compile the real numbers as double instead of float like other languages so we need to include f after the real numbers if we want to declare a real numbers in a float type and for the decimal it will be m.

3.2 Overflowing

Ex:

```
byte number = 255;
number = number + 1; //0
```

In C#, there is no auto overflowing checking, if we go beyond the boundaries it will go around. We can used the checked function to check the overflow of the variables

```
checked {
    byte number = 255;
    number = number + 1;
}
```

3.3 Scope

Scope is where a variable has meaning and accessibility. A block is defined by indicies or {}
Instead of using the datatypes, we can use *var* and C# can auto detect the datatype for you.
When we Ctrl + Click on the datatype, we can see the object browser and

3.4 Type conversion

We can convert from one type to another in C#. Here is an example:

```
byte i = 1; \\ 1 bit
int f  =  i; \\ 4 bit
```

However, we can not convert from one with more bit to another datatype with smaller memory: int to byte. If we want to do that we have to cast the datatype before it. Ex:

```
int i = 1;
byte f  = byte(i);
float z  = 1.0f;
int y =  int(z);
```

For some non-explicit conversion datatype like string to int, etc. We will need to use the *Convert* or *Parse* function that is made in C#.

3.5 Operators

Some different types of operators in C#: arithmetic operator, logical operator, comparison operator, assignment operator, bitwise operator.

3.6 Comments

A comment is a text that we put into our code to increase the readability and maintenance of our code. We can use the single-line comments: // or /**/ for multiple-lines comments.
We use the comment to explain whys, hows, constrains, not the whats.

4 Non-Primitive Types

4.1 Classes

Class combines of related variables (fields) and functions(methods). We can declare classes by using the access modifier + class + class_name.
Inside the class, we can declare the access modifier , method and patterns inside the class.
Creating objects is pretty simple, we only need to declare class_name + object_name = new class_name(); This code here is used to declare the memory for this object.
The static is a member (or even an entire class) belongs to the type itself, not to any specific instance of the class. It means that we do not have to declare new objects to use the method or patterns. We can only access that method through the class, not the method.
We should not build all clasees in one file as there may exist a lot of classes and the file will be too long to modify.

4.2 Structs

It is pretty similar to class but we do not use Structs often. We only need to use the Structs for lightweight object, but still we can do it using the class.

4.3 Arrays

An array is a data structure the we use to store a collection of variables of the same type. Ex:

```
int number1;
int number2;
int number3;

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

In array we need to allocate the fixed size for that array. To access the array, we need to know its index.
For array, when we first declare it, there will be a fixed value 0 for int, False for bool, etc.

4.4 Strings

A string is a sequence of character surrounded by "" (we can not use '' in C#). To declare a string we can do

```
string string_name  = "fill_in_this";
```

We can use the format to format a string or use the join to create a concat string. Ex:

```
var numbers  = new int[3] {1,2,3};
string list  = string.Join(",", numbers);
```

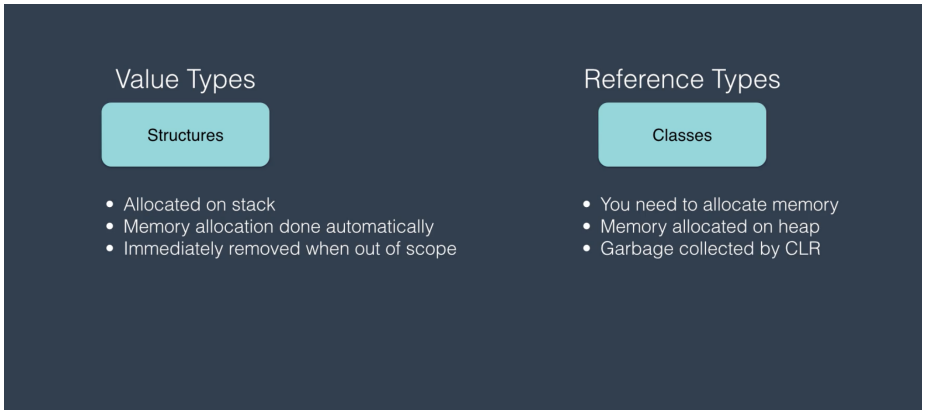
We can access the elements inside a string through its index. We need to remember that strings are **immutable**, we cannot change the value of the variables. There are some escape characters that we can use in string. With the verbatim string, we do not need to use the backslash for declaring some of the elements inside the string, instead we can use @.

4.5 Enums

Enum is a set of name/value pairs.Ex:

```
public enum ShippingMethod
{
    RegularAirMail = 1,
    RegisterAirMail = 2,
    Express = 3;
}
var method = ShippingMethod.Express;
```

4.6 Reference Types and Value Types



5 Control Flow

5.1 Conditional Statement

```
if(condition)
    somestatement
else if(condition)
    another
else
    yetanother
```

We can do the if else statement only to. We can do the nested if else statement. However, we should avoid this as it make the code harder to maintain and debug.
We can do also do the switch case statement for all of these things too.

```
switch(role)
{
    case Role.Admin:
        ...
        break;
    case Role.Moderator:
        ...
        break;
    default:
        ...
        break;
}
```

6 Iteration

6.1 For loops

```
for (var i = 0; i< 10; i++)
{
    ...
}
```

Foreach loop:

```
foreach(var number in numbers)
{
    ...
}
```

6.2 While loops

```
while(i<10)
{
    ...
    i++;
}
```

Do while loop:

```
do {
    ...
    i++;
} while(i<10);
```

break: jump out of the loop
continue: jump to the next iteration

7 Arrays (Deeper)

Quick review: Array is a fixed number of variables of a particular type. We have multi dimension and single dimension array
Multi dimension array include rectangular and jagged array. With the jagged array, the number of rows can be different. Ex for rectangular array:

```
var matrix = new int[3,5]
{
    {1,2,3,4,5},
    {6,7,8,9,10},
    {11,12,13,14,15}
};
// Access element
```

```
matrix[0][0];

//Jagged array is array of array
var array = new int[3][];
array[0] = new int[4];
array[1] = new int[5];
array[2] = new int[6];
```

For the array data type, we have some methods like Clear(), Copy(), IndexOf(), Reverse(), Sort().
Clear() means that we set the number to 0, not deleting it. The IndexOf(), Clear(), Copy(), Reverse() is a static method inside the Array class so it cannot be accessed the array its self. It need to be accessed through the Array class.

8 List

A list is an array with dynamic size. We use list when we donot know how many elements we are going to store. Ex:

```
var numbers  = new List<int>();
var numbers = new List<int> () {1,2,3,4};
```

We can do Add(), Remove(), Count, Contains(),IndexOf(),etc in List which are not usable in Array datatype.

9 Date

You can access datetime in C# through the class DateTime. You can access hour, minute in the class DateTime.
We can also use the TimeSpan, we can add or sparse timespan in C#

10 Procedural Programming

A programming paradigm based on procedure calls. We use function to display our code and make our code easier to maintain and debugging.

11 Working with files

In .NET we have System.IO to work with files and directories. File and FileInfo provide the methods for creating, copying, deleting, moving and opening files. While FileInfo provides instance methods and File provides static methods. There are different functions: Copy(), etc.
For Directories it is the same with Directory and DirectoryInfo but with just different in the functions of each class have.

We also have the Path class to work with directories and files. Path class is a static method class that all the methods inside this class can only be accessed through the Class itself, not the instance.

12 Debugging

You can set break point, run through the line of the code and find the bug in your code and continue debugging.