1. **.Net Core vs .Net Framework**

Developers use the **.NET Framework** to create Windows desktop and server-based applications. This includes ASP.NET web applications. On the other hand, **.NET Core** is used to create server applications that run on Windows, Linux and Mac.

You should use .NET Core when: there are cross-platform needs, Using Microservices, Working with Docker containers, You have high-performance and scalable system needs…etc. Disadvantage of .net core is because it doesn't support the WINDOWS APP and WPF

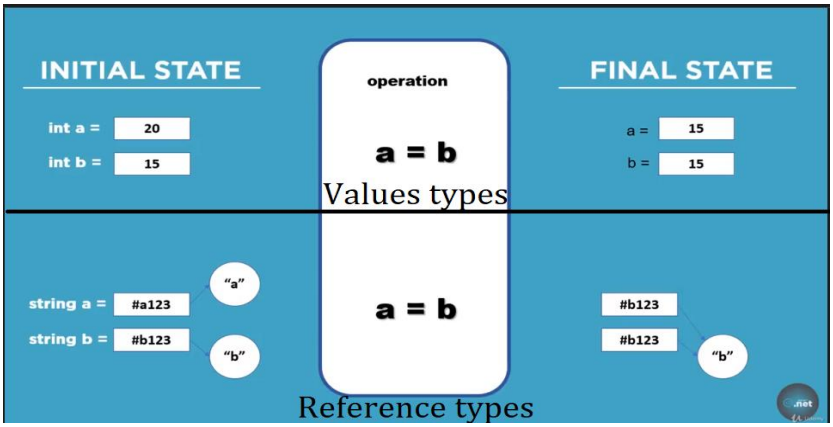
2. **Last versions of C# and .net Framework and ASP.net?**

For ASP.net last version is .NET 6.0

C# 6.0	.NET Framework 4.6	Visual Studio 2013/2015
C# 7.0	.NET Core	Visual Studio 2017

3. **Value Type Vs Reference types**

DataTypes ne C# ndahen ne dy grupe varesisht se si i ruajne te dhenat ne memorie: Nje data type quhet "Value type" nese e ruan ne memorie vleren e nje variable, kurse quhet "Reference type" nese e ruan adresen memorike te asaj variable. Zakonisht reference type i ruajne te dhenat ne HEAP kurse Value types ne STACK. Disa shembuj te Referece Types jane: Classes,Objects, Arrays,Strings,Interfaces..etc



4. **Access modifier PROTECTED-** nenkutpon qe mund t i qasemi prej klases baze si dhe klasave qe e kane trasheguar klasen baze.

   Access modifier Internal – mund tu qasemi vetem Brenda Assembly se njejt (Brenda projektit)

5. **STRUCTURES** – i perdorim pasi qe jane me te shpejta se klasat.
   - E kane vet te krijuar default Contructor-in.
   - Nuk kane Destructor
   - Jane shume te mira per Game Programming
   - Mund te kene: Metoda,Properties,Indexers,Events..etj
   - Mund te imlementojne nje ose me shume Interface
   - Nuk mund te trashegojne nga Structurat ose Klasat e tjera
   - Nuk mund te perdoren si Base class per structurat ose klasat e tjera
   - Anetaret e saj nuk mund te deklarohen si: Abstract,Virtual ose Protected

6. **Enums-** ne c# eshte nje ValueType DataType (pra ruan vlera e jo adresa memorike). Perdoret per te ruajtur nje liste te Integer Constanteve

   Nuk mund te perodret me String Type

```
enum  WeekDays
{
    Monday=1,
    Tuesday =2
}
```

7.Dallimi ne mes String dhe StringBuilder

Dallimi ne mes tyre eshte sepse String eshte Imutable(pra i pandryshueshme) qe nenkupton se cdohere kur deshirojme t'i japim nje vlere te re ose t'ia shtojme nje pjese atehere ne memorie shkatrrohet ai object dhe krijohet objket I ri. Kjo ndikon ne performance. Kurse StringBuilder eshte mutable dhe nuk krijon objekte te reja cdo here kur i nderrohet vlera.

```
string s = string.Empty;
  for (i = 0; i < 1000; i++) {
  s += i.ToString() + " ";
```

```
}
StringBuilder sb = new StringBuilder();
for (i = 0; i < 1000; i++) {
  sb.Append(i);
  sb.Append(' ');
}
```

8.STACK

Eshte nje Strukture e te dhenave qe i ruan te dhenat ne parimin LIFO. Disa nga metodat kryesore te saj jane:

PUSH() – vendos nje element ne Stack

Peak() – kthen elementin me te larte ne Stack (elementin qe ka hy se fundmi)

POP()- e kthen dhe fshin elementin me te larte ne Stack(elementin qe ka hy se fundmi)

CLEAR() – e pastron komplet Stackun

9.QUEUE

I ruan te dhenat ne parimin FIFO. Disa nga metodat kryesore te saj jane:

Enqueue() – vendos elementin ne Queue

Dequeue() – e kthen dhe e fshin elementin me te larte (ai qe ka hy me heret) ne Queue

Peak() – e kthen elementin me te larte (ai qe ka hy me heret) ne Queue

10. INDEXERS

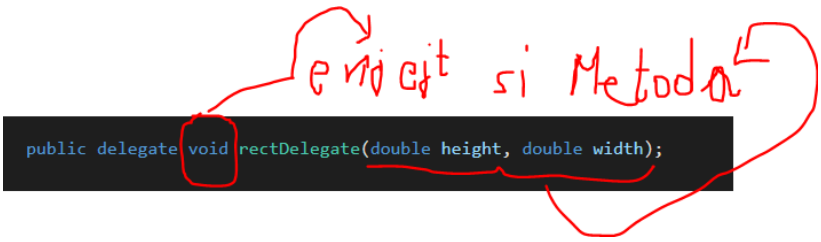
Ne c# indexers jane tip specific i propertive qe mundeson qe nje klase te trajohet sikurse nje Array.

```
1. class IndexerClass
2.         {
3.             private string[] names = new string[10];
4.
5.             public string this[int i]
6.             {
7.                 get
8.                 {
9.                     return names[i];
10.                 }
11.                 set
12.                 {
13.                     names[i] = value;
14.                 }
15.             }
```

```
16.         }
```

11. Delagates

Jane objekte te cilat qe u referohen metodave ose funkdioneve te ndryshme varesisht se kush I thirr per te kryer pune. Kur e deklarojme nje Delegate eshte me rendesi qe t'I caktojme TIPIN KTHYES dhe PARAMTERAT te njejta me metodat tona per te cilat dehsirojme ta thirrim.




```
public  class Rectangle
{
    public delegate void rectDelegate(double height, double width);

    // "area" method
    1 reference
    public void area(double height, double width)
    {
        Console.WriteLine("Area is: {0}", (width * height));
    }

    // "perimeter" method
    1 reference
    public void perimeter(double height, double width)
    {
        Console.WriteLine("Perimeter is: {0} ", 2 * (width + height));
    }
}
```

```
class Program
{
    0 references
    static void Main(string[] args)
    {

        Rectangle rect = new Rectangle();

        rectDelegate rectdele = rect.area;
        rectdele(12, 13);

        rectdele = rect.perimeter;
        rectdele(12, 13);

        Console.ReadKey();
    }
```

```
D:\Data2Know\Ligjeratat\Ligj2\K
Area is: 156
Perimeter is: 50
```

12. Generics

Ne c# generics eshte nje klase qe nenkupton nje forme gjenerale, jo e specifikuar. Ajo mundeson deklarimin te klasave apo metodave ne menyre te pergjithshme pa e definuar tipin e te dhenave, kjo nenkupton qe mund ta perdoresh ate metode ose klase per cfardo tipi te te dhenave. Duke perdorur Generics ju evitoni Double Write Code (nuk ka nevoje te shkruhet medota e njejt per 2 tipe te ndryshme te te dhenave.

```
class DataStore<T>
{
    public T Data { get; set; }
}

DataStore<string> strStore = new DataStore<string>();
strStore.Data = "Hello World!";

DataStore<int> intStore = new DataStore<int>();
intStore.Data = 100;
```

13.Partial Class

Ne c# partial classes mundesojne qe dy klasa me emer te njejt te trajtohen si nje dhe e vetme. Pra ju mund te deklaroni 2 partial Classes me emer te njejt, te shkruani metoda e ndryshme te secila, me pastaj kur krijoni objekte prej tyre ju mund te thirrni cilindo nga metodat. Partial Classes u mundesojne developerave te punojne ne te njejten kohe ne nje klase ne fajlla te ndare.

public partial class HelloClass

{

public void HelloWorld{ Console.WriteLine(“Hello 1”);

}

public partial class HelloClass

{

public void HelloWorld2 { Console.WriteLine(“Hello 2”);

}

14.STATIC

Eshte nje keyword qe nenkupton dicka qe nuk mund te instancohet. NUK mund te krijohet object prej nje klase statike. Ato rrijne te memorie gjate tere ciklit te programit. Per klasat statiek vlen:

1. Indexers and destructors cannot be static
2. Static classes are sealed class and therefore, cannot be inherited.
3. A static class cannot inherit from other classes.

15. REST Full API, Tokens, Sessions, Cookies, Microservices, Monolithic Arch, REDIS,DOCKER.

16. DOCKER, KUBERNETES (menahxon Dockerat)