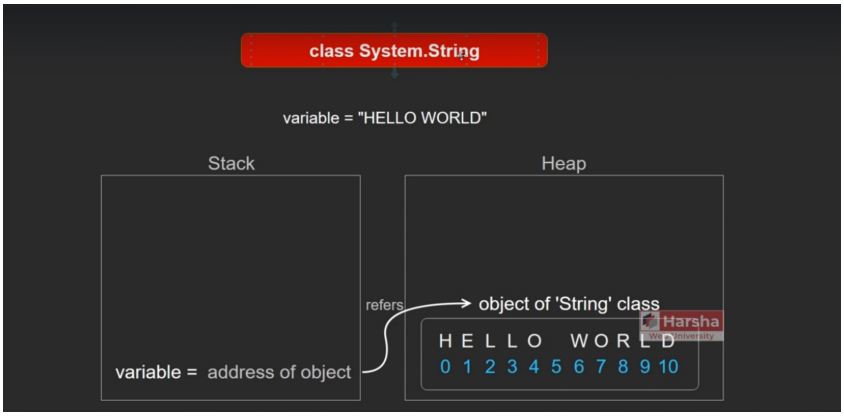
Section 26: String, DateTime, Math

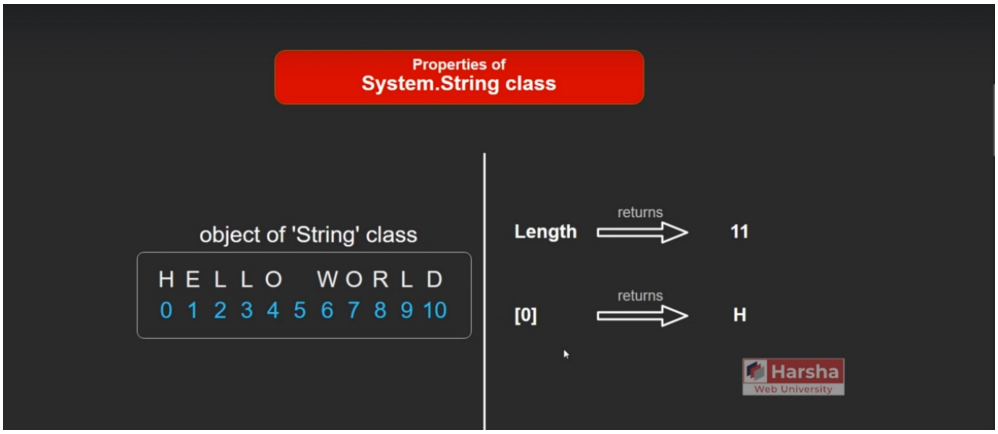
29 July 2024 11:32

In C sharp all the strings by default treated as objects of a predefined class called system.String



System.String == string

both are equal internally.



String

What

- The System.String is a class that represents an array of Unicode characters.
- String stores a set of characters as char[].
- Max no. of characters in the string: 2 billion.
- String is immutable. That means, it can't be modified.

String	
0	H
1	e
2	l
3	l
4	o
5	1
6	2
7	3

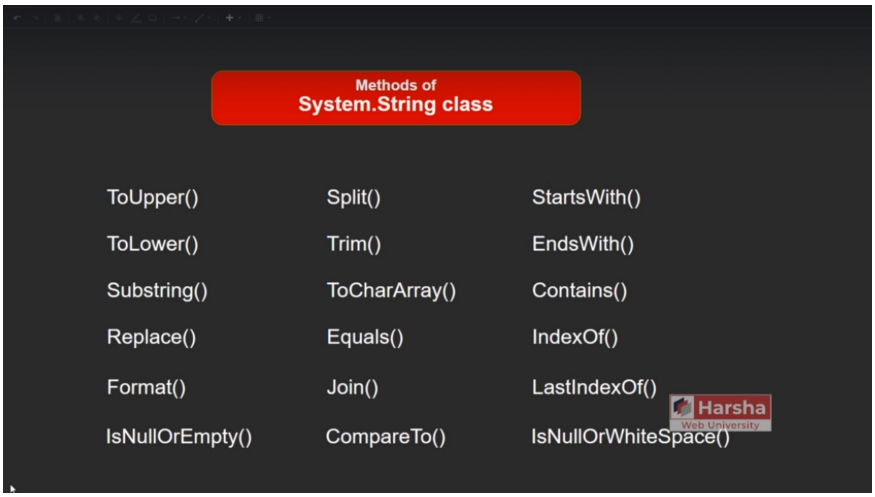
How

```
String
string referenceVariable = "Hello123";
```


String

String Properties

#	Property
1	int Length { get; }
	This property returns the no. of characters of the string.
2	char [int index] { get; }
	This indexer returns the single character at the specified character index.

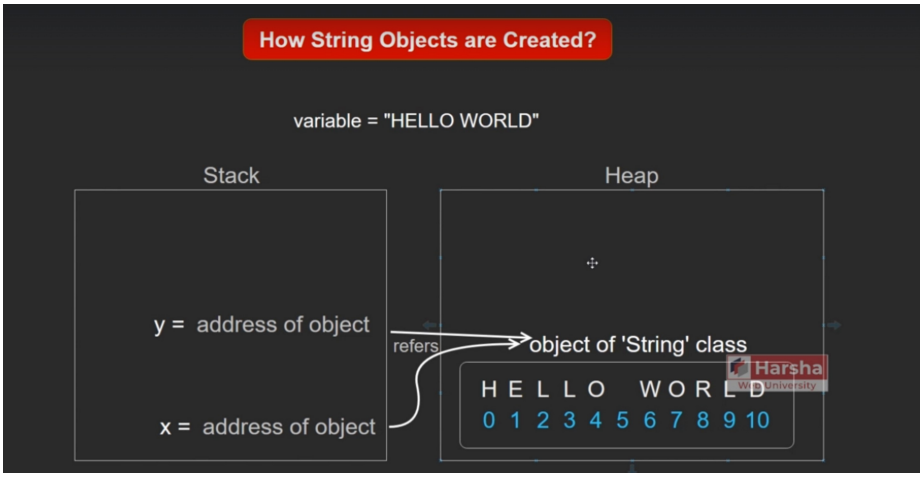


How String Objects are Created

```
using System;

namespace StringObjectsExample
{
    0 references
    class Program
    {
        0 references
        static void Main()
        {
            string x = "HELLO WORLD";
            string y = "HELLO WORLD";
        }
    }
}
```

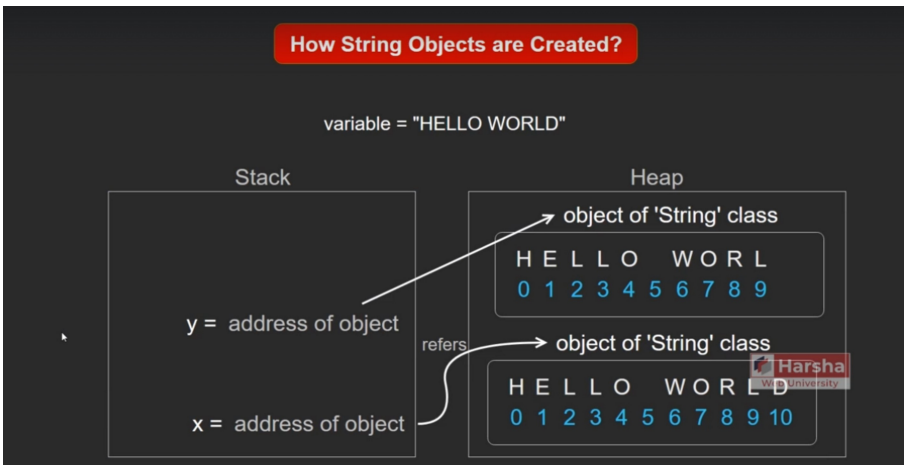
The CLR of dot net, that is Common Language Runtime will automatically recognize that, there is already an existing string object with the same value.

Instead of creating a new object, it will just create a reference variable and point to the same object to the memory.

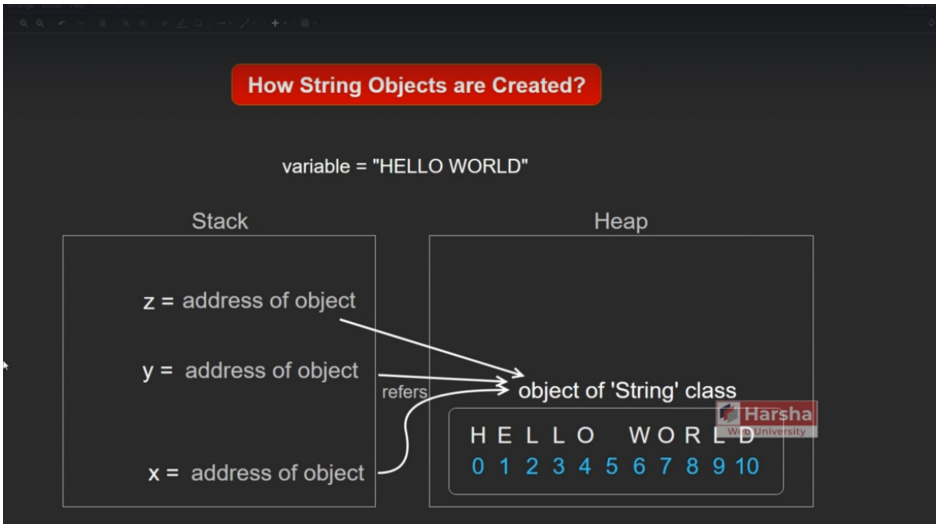


This is the internal strategy automatically done by CLR itself. This is basically meant for avoiding the memory wastage.

```
using System;

namespace StringObjectsExample
{
    0 references
    class Program
    {
        0 references
        static void Main()
        {
            string x = "HELLO WORLD";
            string y = "HELLO WORL";
        }
    }
}
```

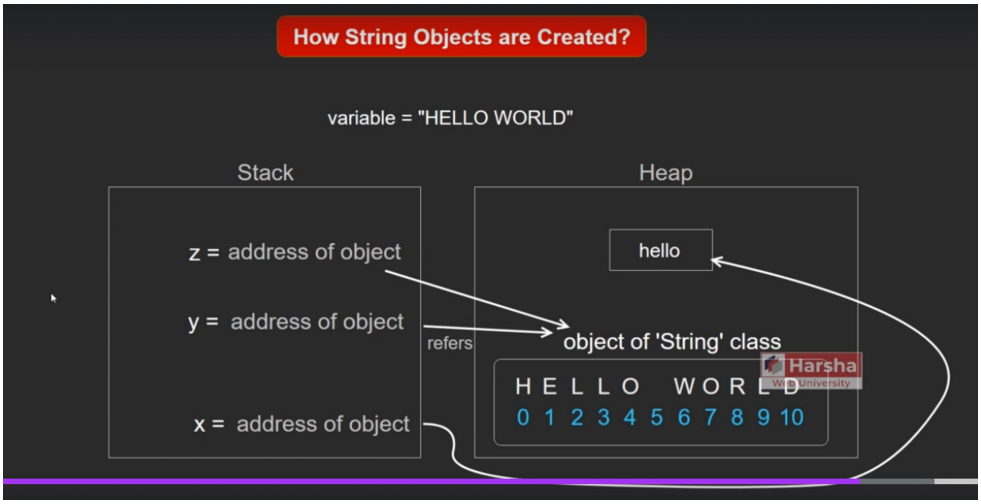

How String Objects are Created?
variable = "HELLO WORLD"
Stack
Heap
object of 'String' class
H E L L O W O R L
0 1 2 3 4 5 6 7 8 9
y = address of object
refers
object of 'String' class
x = address of object
H E L L O W O R L D
0 1 2 3 4 5 6 7 8 9 10
Harsha


namespace StringObjectsExample
{
0 references
class Program
{
0 references
static void Main()
{
string x = "HELLO WORLD";
string y = "HELLO WORLD";
string z = x;
}
}
}


How String Objects are Created?
variable = "HELLO WORLD"
Stack
Heap
z = address of object
y = address of object
refers
object of 'String' class
x = address of object
H E L L O W O R L D
0 1 2 3 4 5 6 7 8 9 10
Harsha

```
namespace StringObjectsExample
{
    0 references
    class Program
    {
        0 references
        static void Main()
        {
            string x = "HELLO WORLD";
            string y = "HELLO WORLD";
            string z = x;
            x = "hello";
        }
    }
}
```

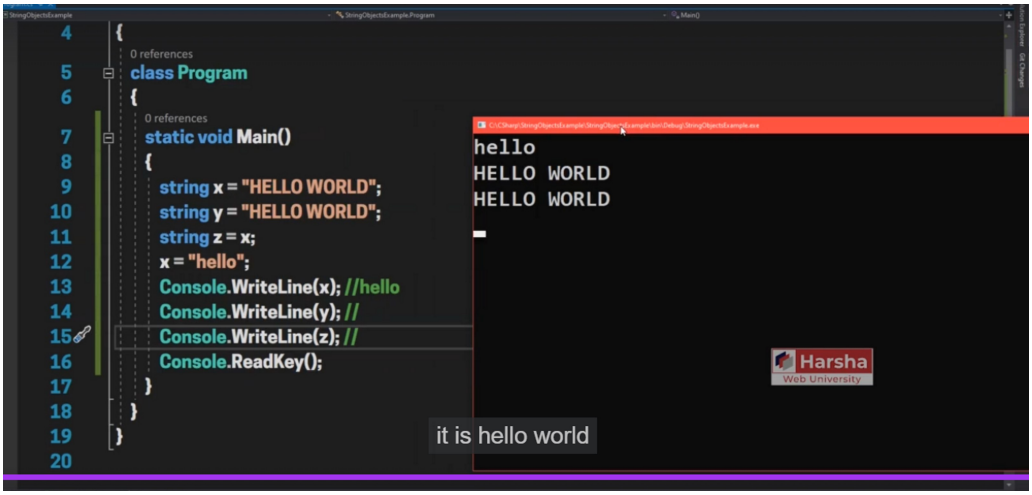
this right hand side part of the code



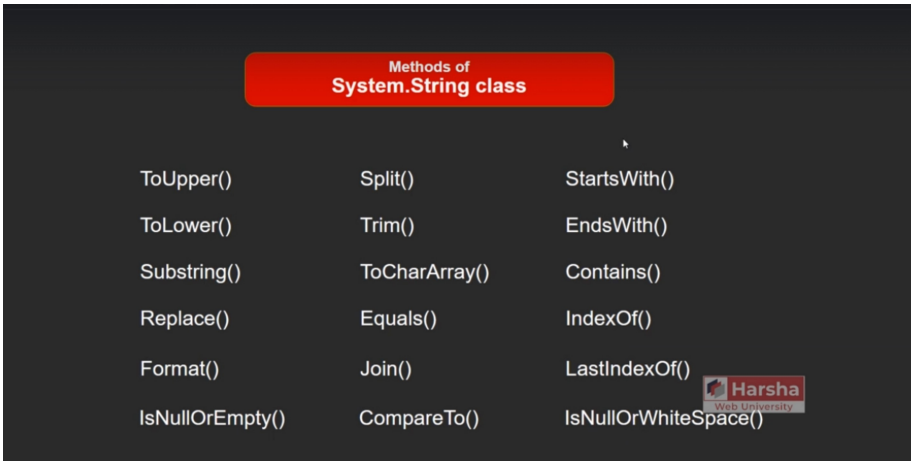

```
1  using System;
2
3  namespace StringObjectsExample
4  {
5      0 references
       class Program
6      {
           0 references
7          static void Main()
8          {
9              string x = "HELLO WORLD";
10             string y = "HELLO WORLD";
11             string z = x;
12             x = "hello";
13             Console.WriteLine(x); //hello
14             Console.ReadKey();
15         }
16     }
17 }
```

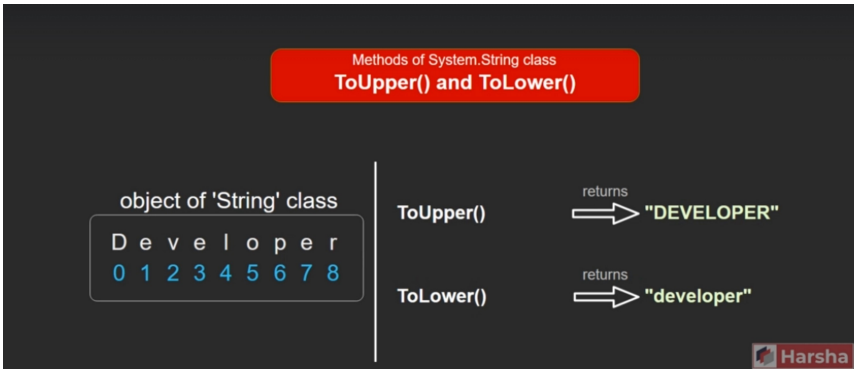
```
hello
```

Harsha
Web University


0 references
class Program
{
0 references
static void Main()
{
string x = "HELLO WORLD";
string y = "HELLO WORLD";
string z = x;
x = "hello";
Console.WriteLine(x); //hello
Console.WriteLine(y); //
Console.WriteLine(z); //
Console.ReadKey();
}
}
}
hello
HELLO WORLD
HELLO WORLD
Harsha
Web University
it is hello world


Converting Strings
Part 1


Methods of
System.String class
ToUpper()
ToLower()
Substring()
Replace()
Format()
IsNullOrEmpty()
Split()
Trim()
ToCharArray()
Equals()
Join()
CompareTo()
StartsWith()
EndsWith()
Contains()
IndexOf()
LastIndexOf()
IsNullOrWhiteSpace()
Harsha
Web University


Methods of System.String class
ToUpper() and ToLower()
object of 'String' class
D e v e l o p e r
0 1 2 3 4 5 6 7 8
ToUpper()
returns
"DEVELOPER"
ToLower()
returns
"developer"
Harsha

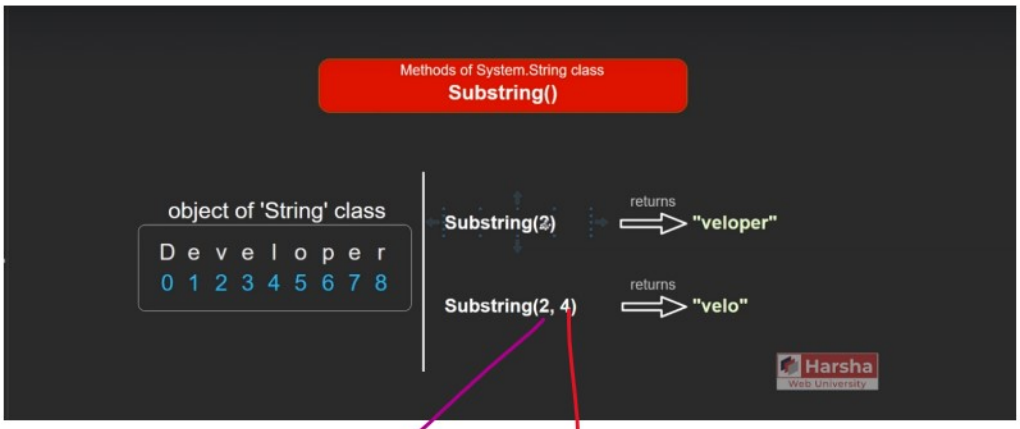
```
namespace ConvertingStringExample
{
    0 references
    class Program
    {
        0 references
        static void Main()
        {
            string str = "Developer";
            Console.WriteLine(str);  //Output: Developer
            string str_upr = str.ToUpper();
            Console.WriteLine(str_upr);
            Console.ReadKey();
        }
    }
}
```

Internally, creates a new object and send the reference of it.

```
4   namespace ConvertingStringExample
5   {
        0 references
6       class Program
7       {
            0 references
8           static void Main()
9           {
10              string str = "Developer";
11              Console.WriteLine(str);  //Output: Developer
12              string str_upr = str.ToUpper();
13              Console.WriteLine(str_upr); //Output: DEVELOPER
14              string str_lwr = str.ToLower();
15              Console.WriteLine(str_lwr); //Output: developer
16
17              Console.ReadKey();
18          }
19      }
20  }
```

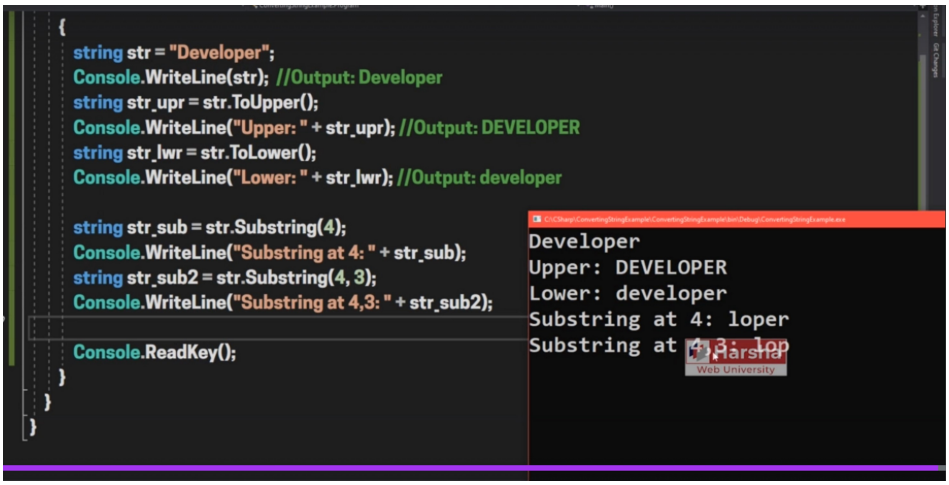
```
Developer
DEVELOPER
developer
```

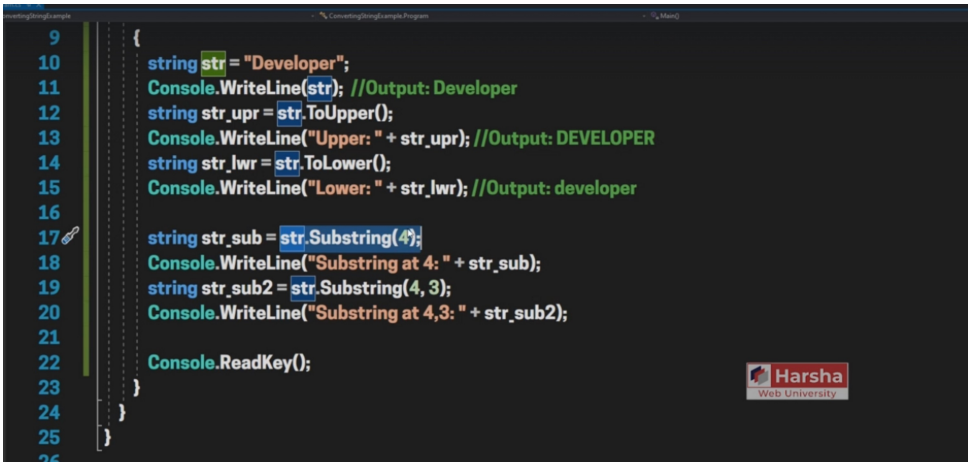
Harsha Web University

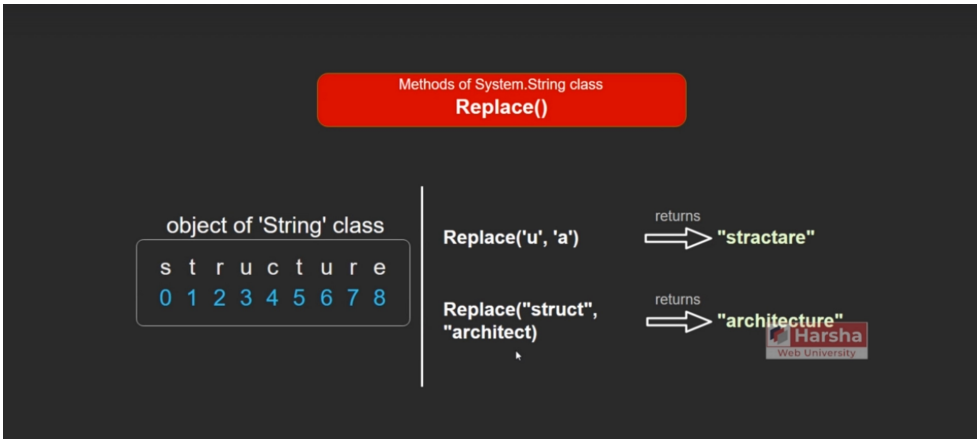


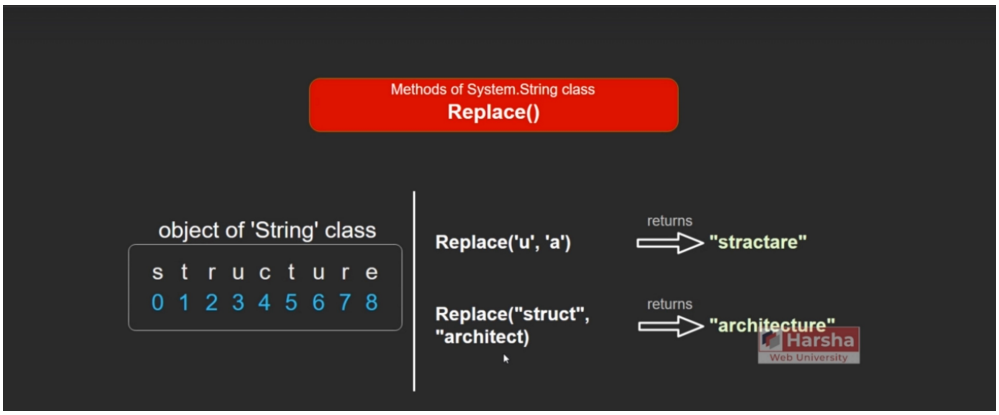
length

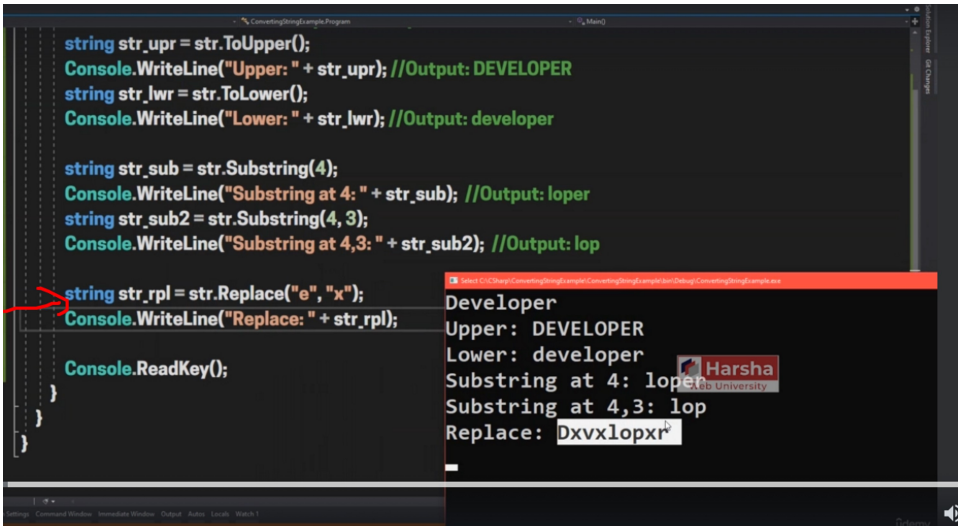
starting index

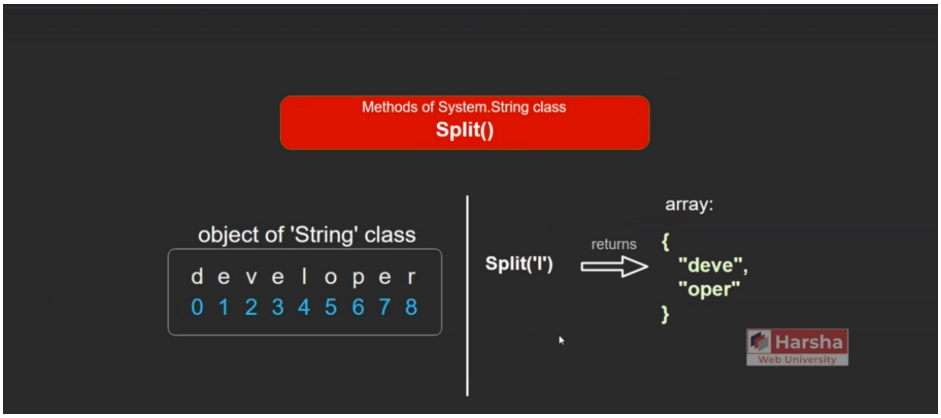


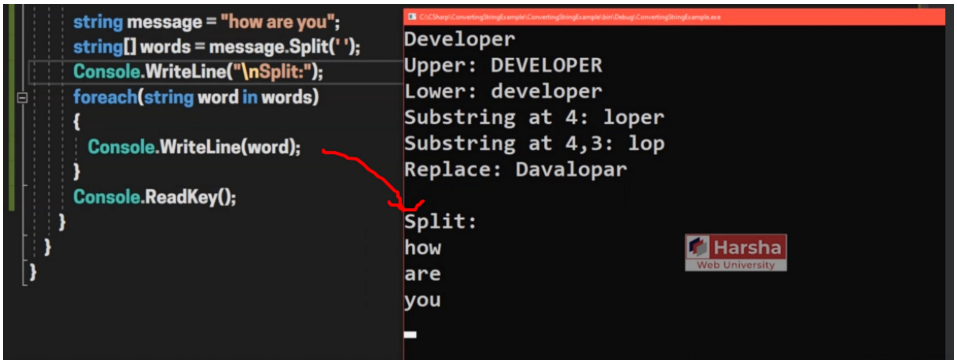


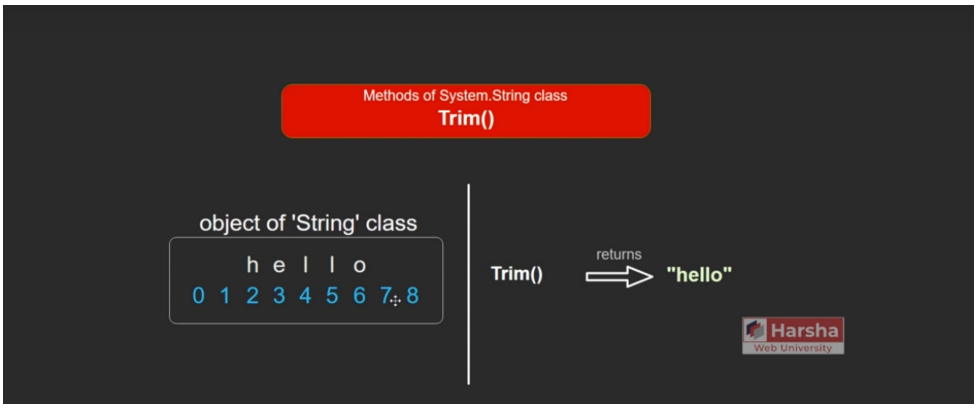

Methods of System.String class
Replace()
object of 'String' class
s t r u c t u r e
0 1 2 3 4 5 6 7 8
Replace('u', 'a')
returns
"stractare"
Replace("struct", "architect)
returns
"architecture"
Harsha
Web University


Methods of System.String class
Replace()
object of 'String' class
s t r u c t u r e
0 1 2 3 4 5 6 7 8
Replace('u', 'a')
returns
"stractare"
Replace("struct", "architect)
returns
"architecture"
Harsha
Web University

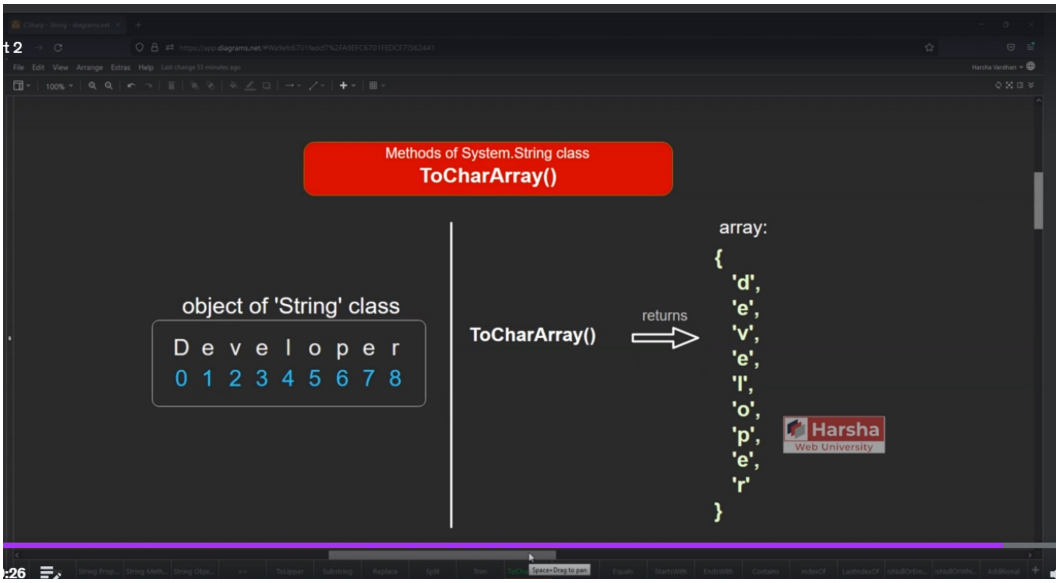

string str_upr = str.ToUpper();
Console.WriteLine("Upper: " + str_upr); //Output: DEVELOPER
string str_lwr = str.ToLower();
Console.WriteLine("Lower: " + str_lwr); //Output: developer
string str_sub = str.Substring(4);
Console.WriteLine("Substring at 4: " + str_sub); //Output: loper
string str_sub2 = str.Substring(4, 3);
Console.WriteLine("Substring at 4,3: " + str_sub2); //Output: lop
string str_rpl = str.Replace("e", "x");
Console.WriteLine("Replace: " + str_rpl);
Console.ReadKey();
}
}
}
Developer
Upper: DEVELOPER
Lower: developer
Substring at 4: loper
Substring at 4,3: lop
Replace: Dxvxlopxr
Harsha
Web University

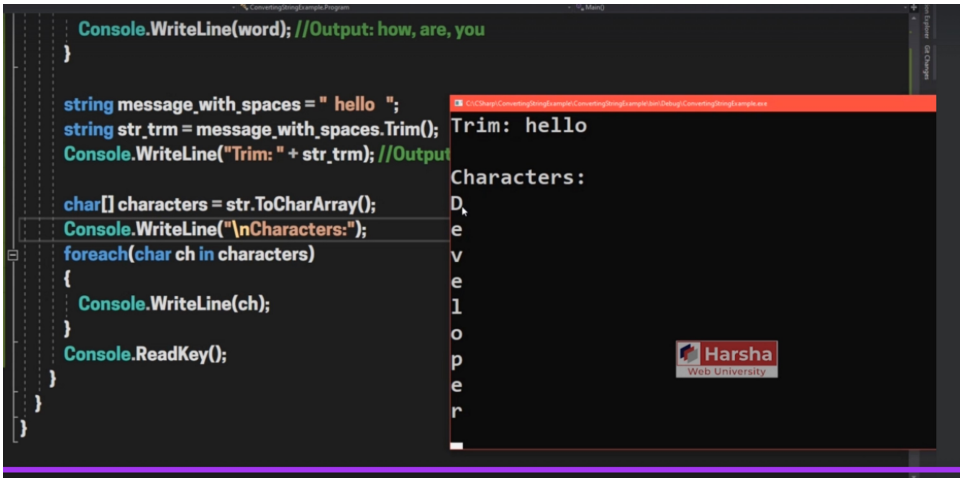


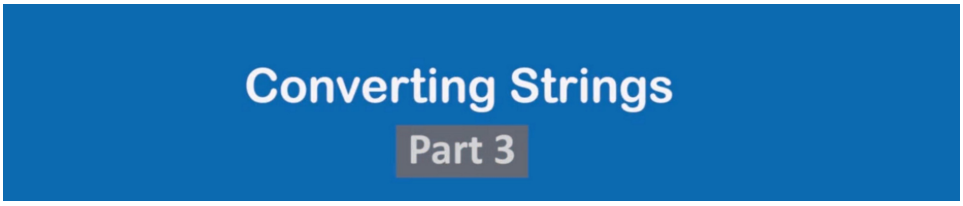


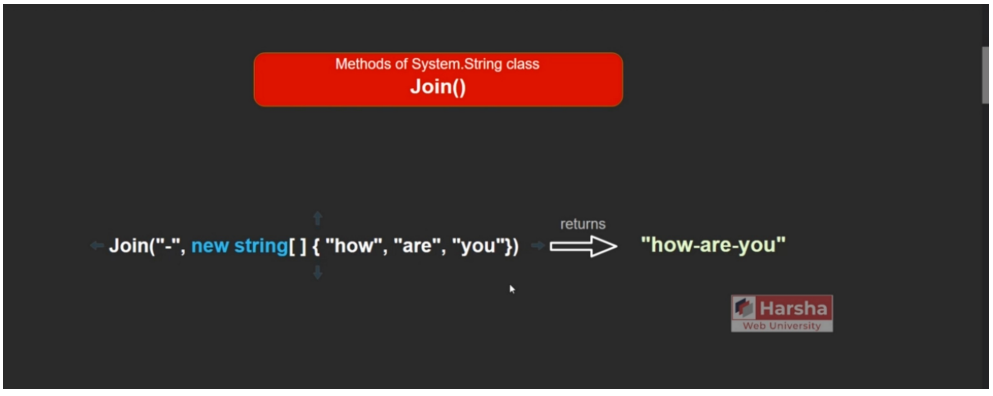


Trim() method removes unnecessary spaces from left and right side.
but not in middle.


Methods of System.String class
ToCharArray()
object of 'String' class
D e v e l o p e r
0 1 2 3 4 5 6 7 8
ToCharArray()
returns
array:
{
'd',
'e',
'v',
'e',
'l',
'o',
'p',
'e',
'r'
}
Harsha
Web University


Console.WriteLine(word); //Output: how, are, you
}
string message_with_spaces = " hello ";
string str_trm = message_with_spaces.Trim();
Console.WriteLine("Trim: " + str_trm); //Output
char[] characters = str.ToCharArray();
Console.WriteLine("\nCharacters:");
foreach(char ch in characters)
{
Console.WriteLine(ch);
}
Console.ReadKey();
}
}
}
Trim: hello
Characters:
D
e
v
e
l
o
p
e
r
Harsha
Web University


Converting Strings
Part 3


Methods of System.String class
Join()
Join("-", new string[ ] { "how", "are", "you"})
returns
"how-are-you"
Harsha
Web University

```
string[] my_words = new string[] { "how", "are", "you"}; //how-are-you
string str_jn = string.Join("--", my_words);
Console.WriteLine("Join: " + str_jn);

char[] characters2 = new char[] { 'h', 'e', 'l', 'l', 'o' };
string str2 = new string(characters2);
Console.WriteLine("New string: " + s
```

earlier

Harsha Web University

String

Converting Strings

#	Method
1	string ToUpper( )
	Returns the same string in upper case.
2	string ToLower( )
	Returns the same string lower case.
3	string Substring(int startIndex, int length)
	Returns a string with a set of characters starting from the "startIndex" up to the "length" no. of characters. The "length" parameter is optional.
4	string Replace(string oldString, string newString)
	Returns a string after replacing all occurrences of oldString with newString.
5	string[] Split(char separator)
	Returns a string[] after splitting the current string into an array of characters. At each occurrence of separator character, a new string is formed-up. At last, all the pieces of strings are converted as a string[].

String

Converting Strings

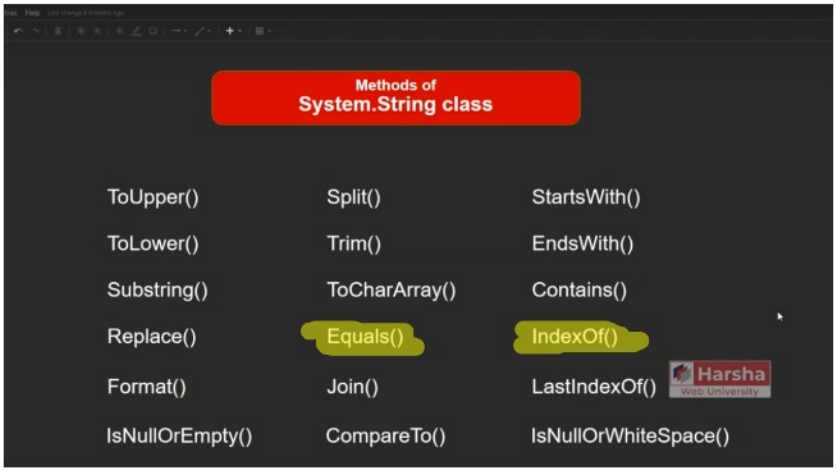
#	Method
6	string Trim()
	Returns the same string after removing extra spaces at beginning and ending of the current string. It returns the same string if no spaces found at L.H.S. and R.H.S. of the current string.
7	string ToCharArray()
	Returns the same string as an array of characters (char[]).
8	string Join(string separator, IEnumerable<T> values)
	Returns the a string with joined values with separator in between each value.

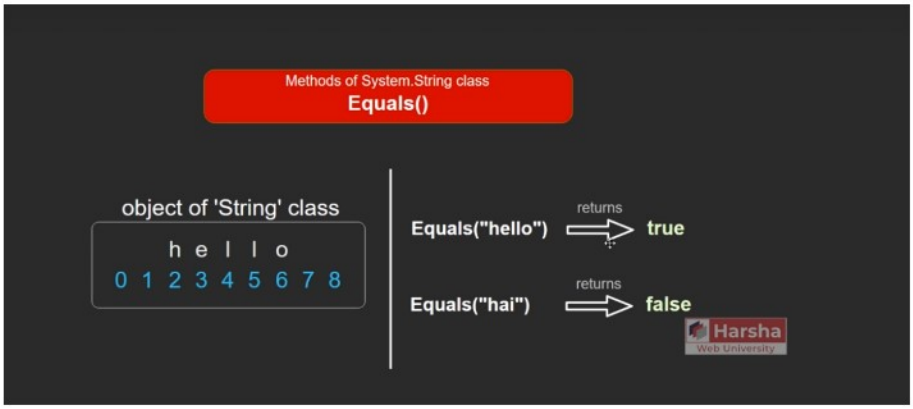
'String' class Constructor

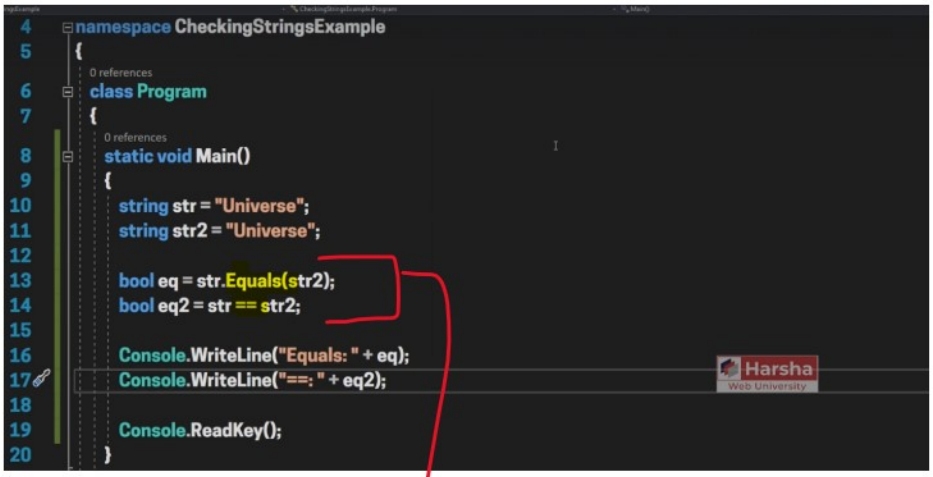
Constructor	Description
String(char[])	It initializes the string with specified char[].

Checking Strings

Part 1

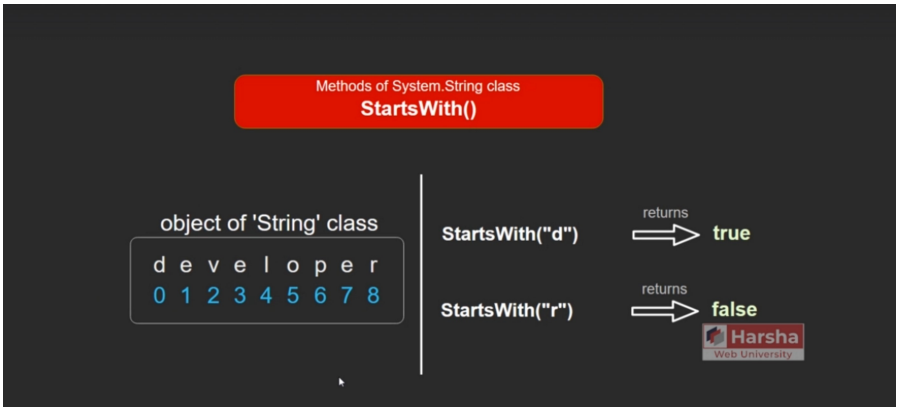


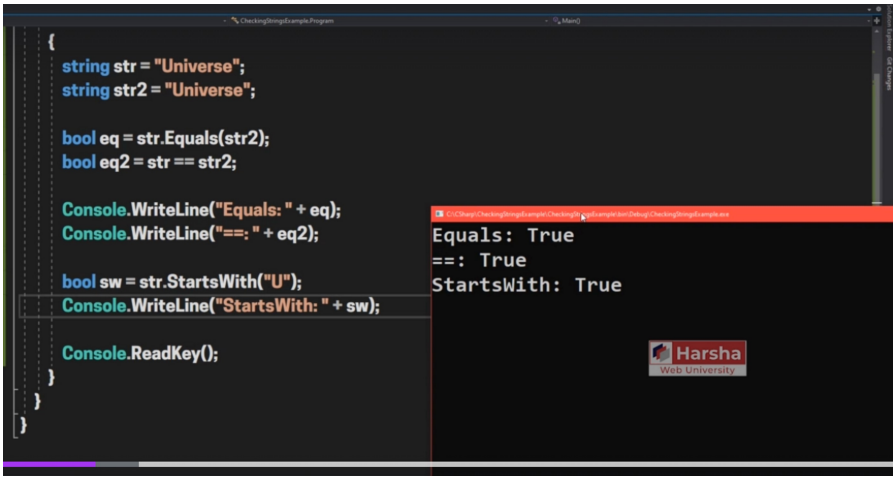


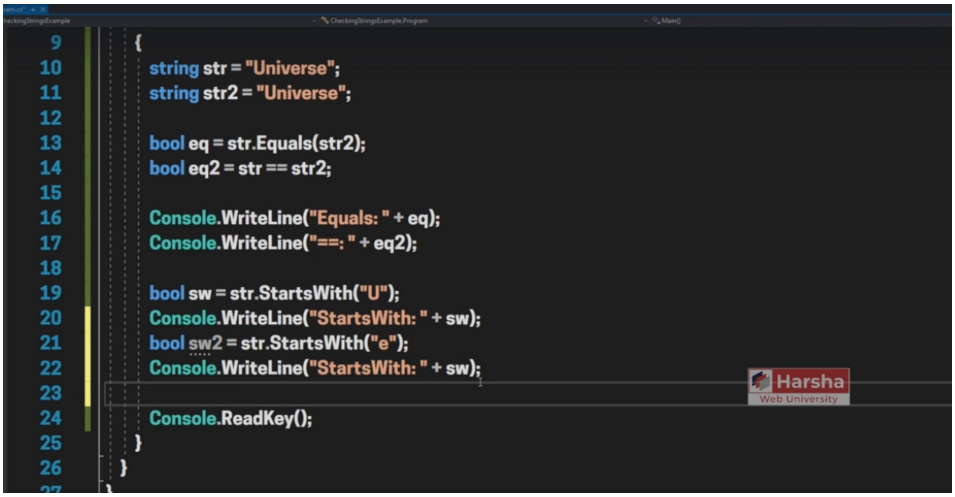


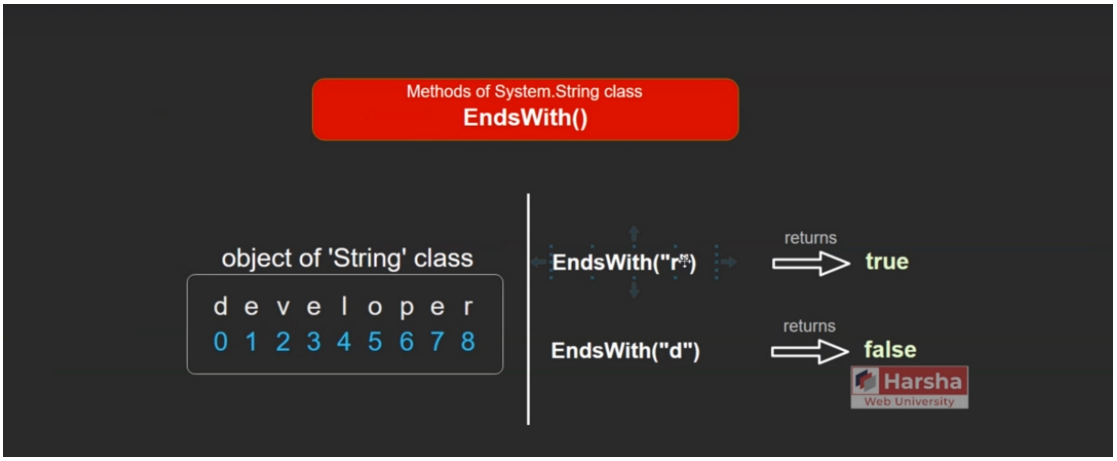
In C#, Equal() and ==

works in a same way for string.

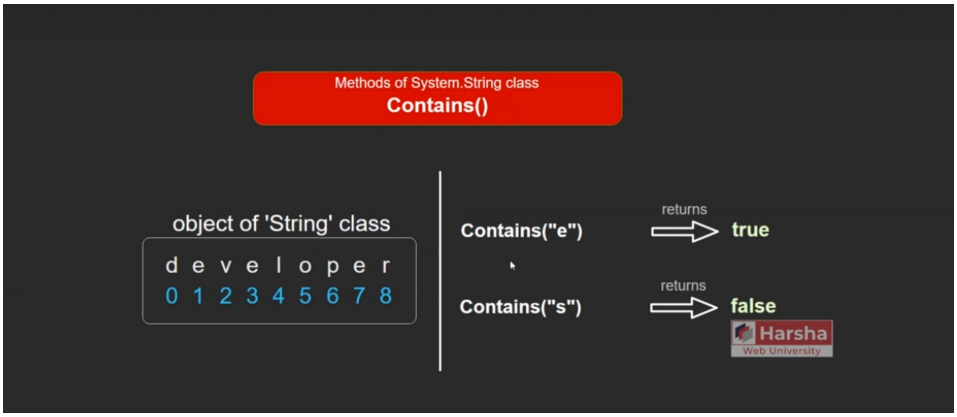

Methods of System.String class
StartsWith()
object of 'String' class
d e v e l o p e r
0 1 2 3 4 5 6 7 8
StartsWith("d")
returns
true
StartsWith("r")
returns
false
Harsha
Web University


{
string str = "Universe";
string str2 = "Universe";
bool eq = str.Equals(str2);
bool eq2 = str == str2;
Console.WriteLine("Equals: " + eq);
Console.WriteLine("==: " + eq2);
bool sw = str.StartsWith("U");
Console.WriteLine("StartsWith: " + sw);
Console.ReadKey();
}
}
}
Equals: True
==: True
StartsWith: True
Harsha
Web University

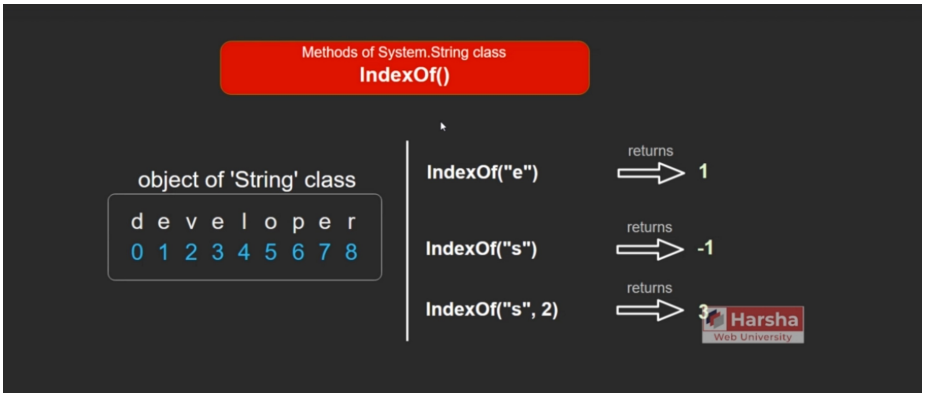

9 {
10 string str = "Universe";
11 string str2 = "Universe";
12
13 bool eq = str.Equals(str2);
14 bool eq2 = str == str2;
15
16 Console.WriteLine("Equals: " + eq);
17 Console.WriteLine("==: " + eq2);
18
19 bool sw = str.StartsWith("U");
20 Console.WriteLine("StartsWith: " + sw);
21 bool sw2 = str.StartsWith("e");
22 Console.WriteLine("StartsWith: " + sw);
23
24 Console.ReadKey();
25 }
26 }
Harsha
Web University

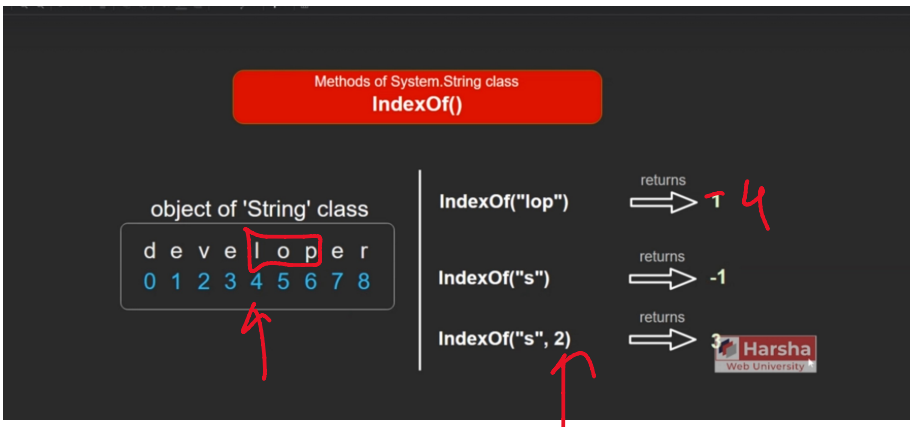



```
bool ew = str.EndsWith("e");
bool ew2 = str.EndsWith("U");
Console.WriteLine("Ends with: " + ew); //Output: true
Console.WriteLine("Ends with: " + ew); //Output: false

Console.ReadKey();
}
}
```

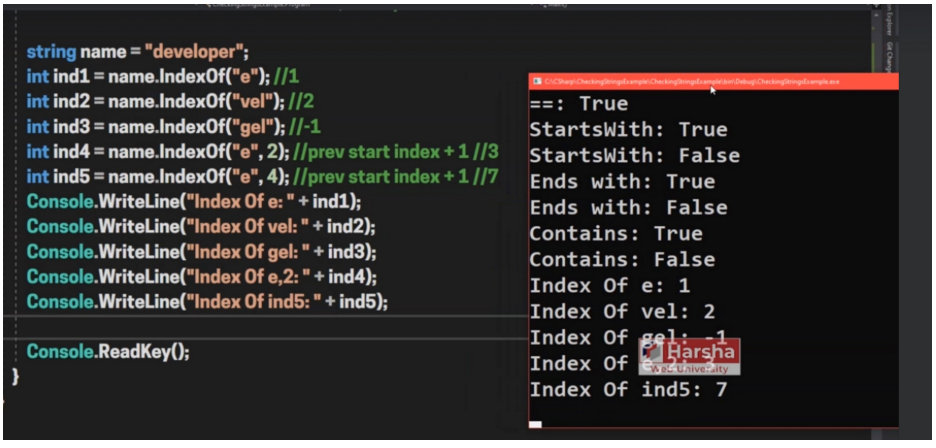




```
bool ct = str.Contains("e");
bool ct2 = str.Contains("t");
Console.WriteLine("Contains: " + ct); //Output: true
Console.WriteLine("Contains: " + ct2); //Output: false

Console.ReadKey();
```





startIndex

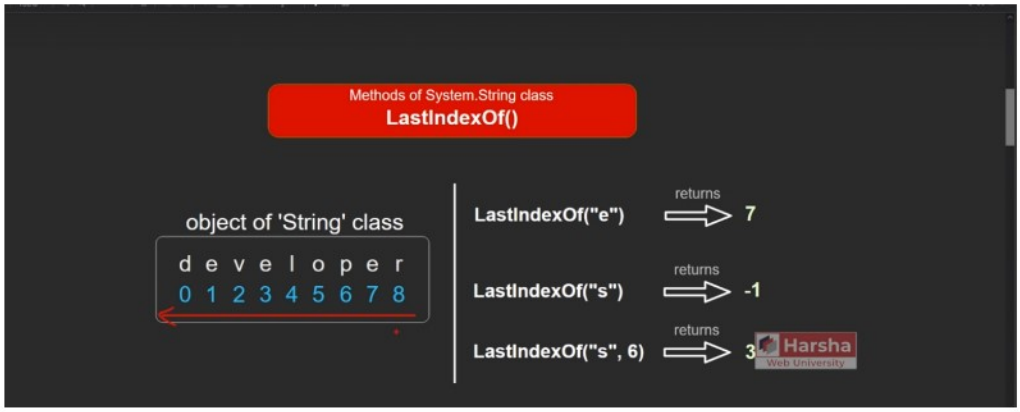


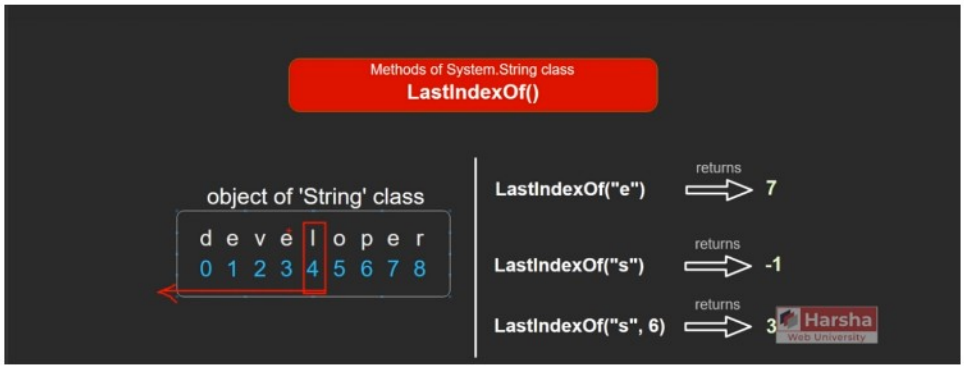
```
string name = "developer";
int ind1 = name.IndexOf("e"); //1
int ind2 = name.IndexOf("vel"); //2
int ind3 = name.IndexOf("gel"); //-1
int ind4 = name.IndexOf("e", 2); //prev start index + 1 //3
int ind5 = name.IndexOf("e", 4); //prev start index + 1 //7
Console.WriteLine("Index Of e: " + ind1);
Console.WriteLine("Index Of vel: " + ind2);
Console.WriteLine("Index Of gel: " + ind3);
Console.WriteLine("Index Of e,2: " + ind4);
Console.WriteLine("Index Of ind5: " + ind5);

Console.ReadKey();
}
```

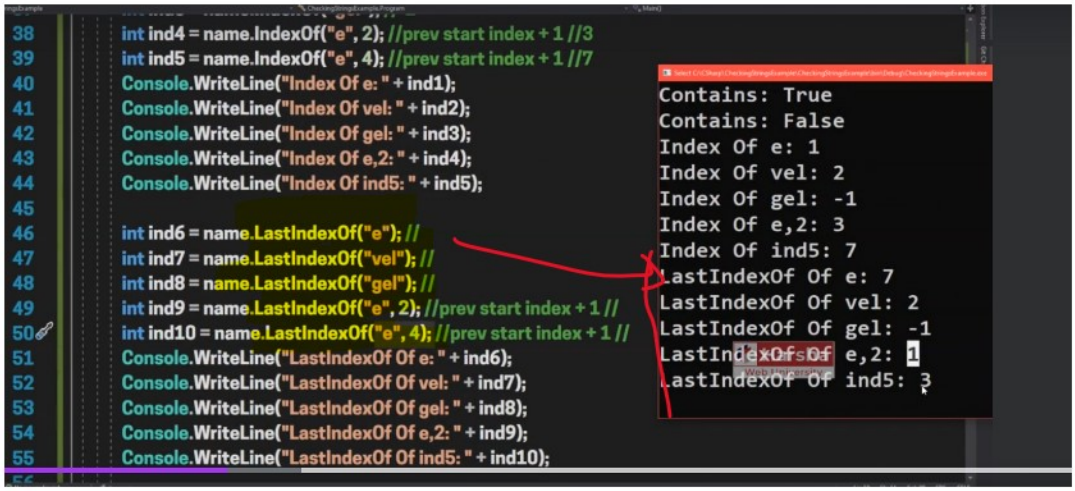
```
==: True
StartsWith: True
StartsWith: False
Ends with: True
Ends with: False
Contains: True
Contains: False
Index Of e: 1
Index Of vel: 2
Index Of gel: -1
Index Of
Index Of ind5: 7
```

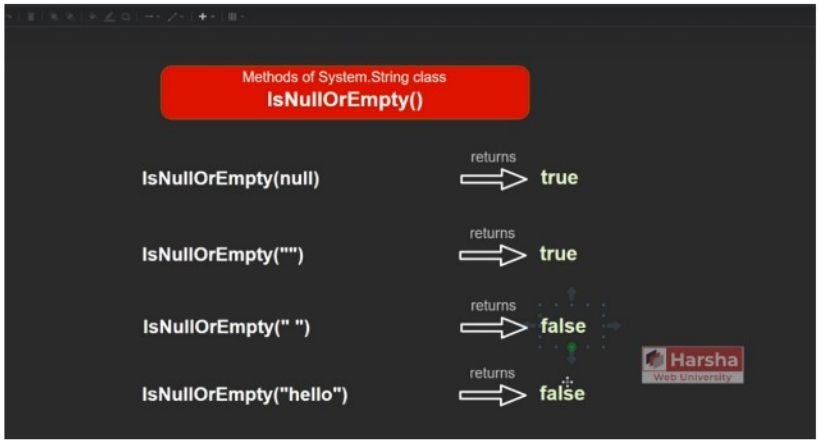
Checking Strings

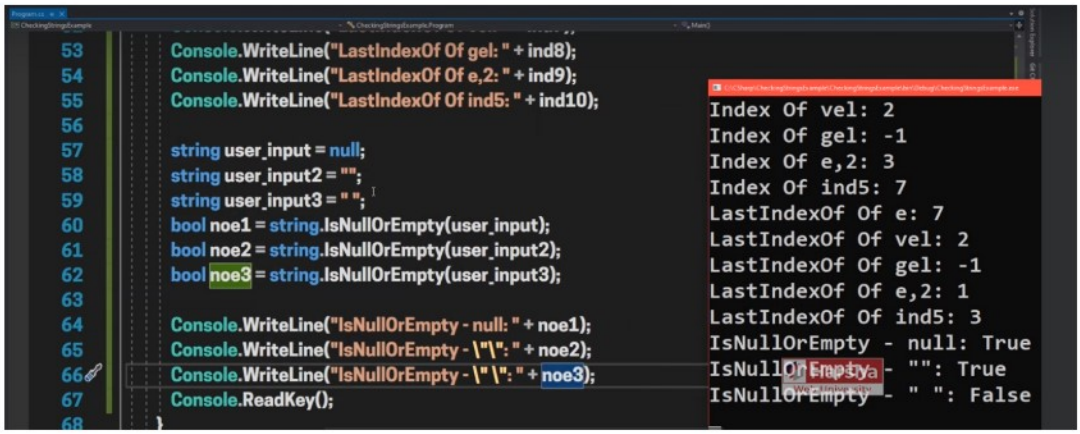
Part 2

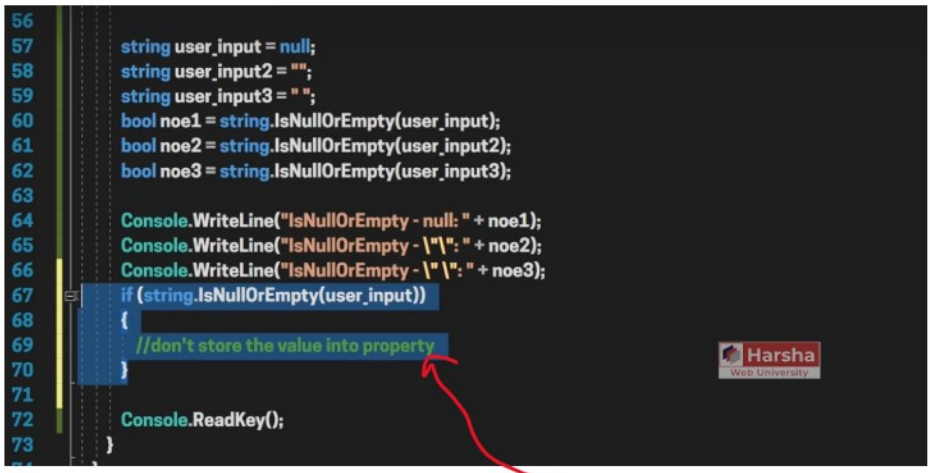




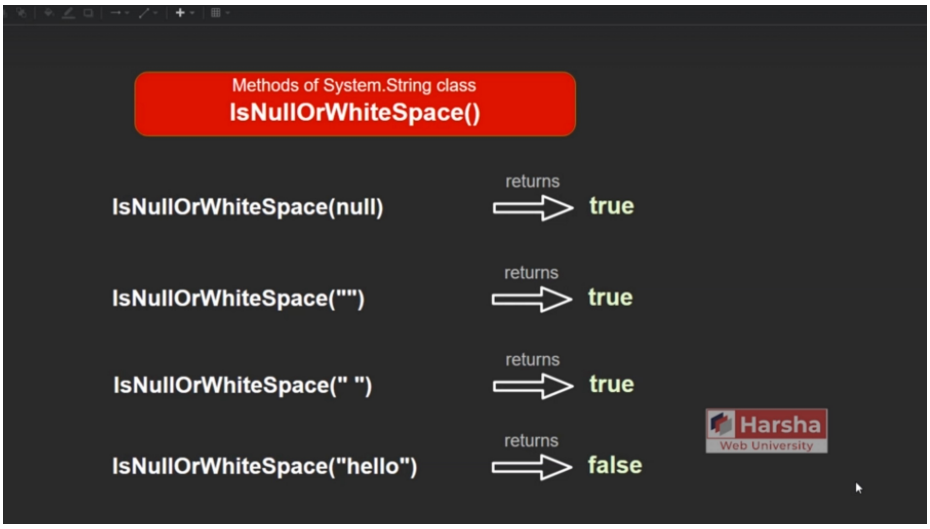


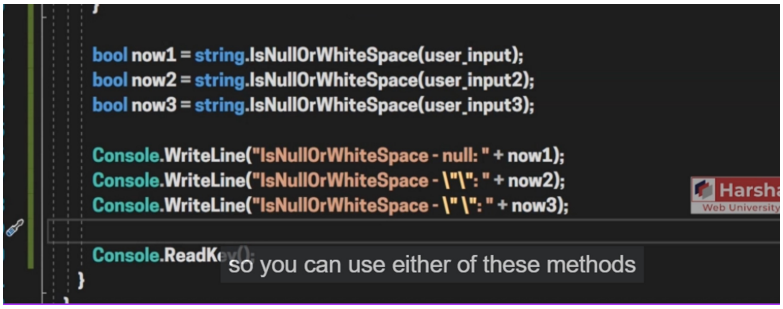


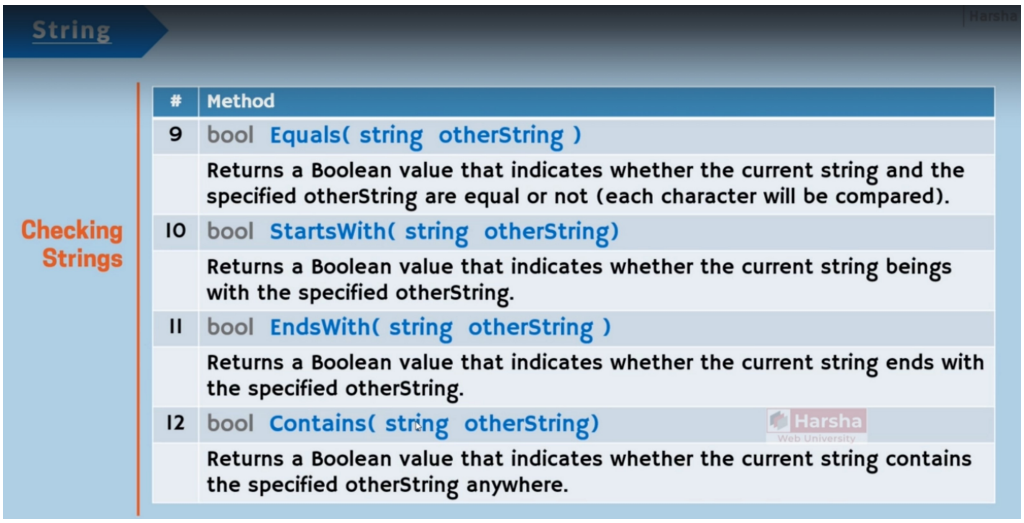


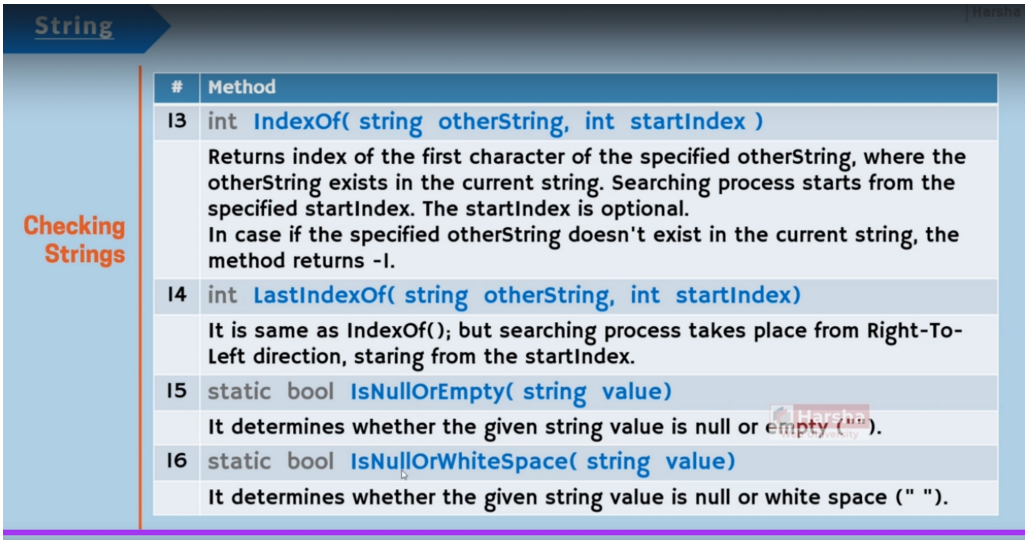


Practice: Create a Customer class with a customerName property along with this type of condition in the set accessor.

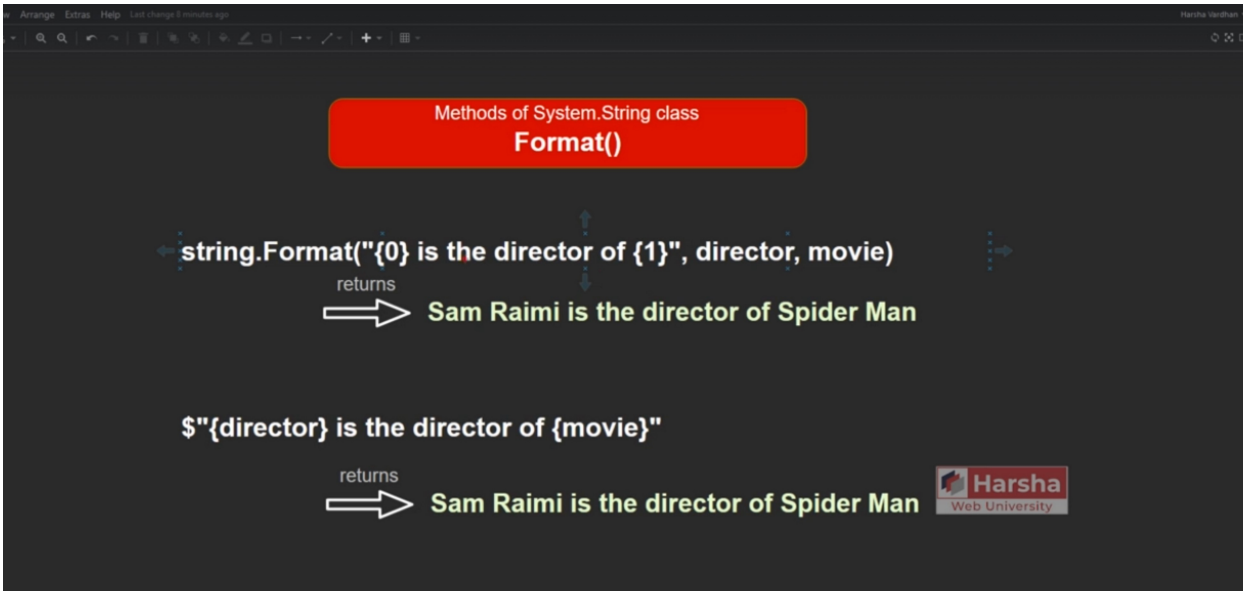

Methods of System.String class
IsNullOrWhiteSpace()
IsNullOrWhiteSpace(null) returns true
IsNullOrWhiteSpace("") returns true
IsNullOrWhiteSpace(" ") returns true
IsNullOrWhiteSpace("hello") returns false
Harsha
Web University

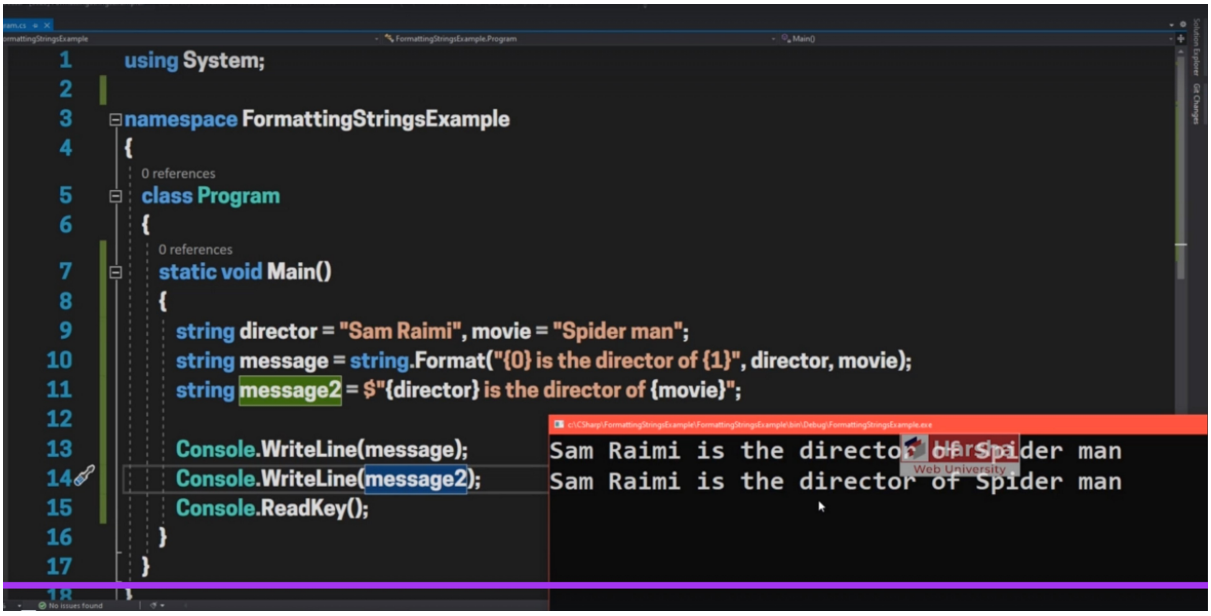

bool now1 = string.IsNullOrWhiteSpace(user_input);
bool now2 = string.IsNullOrWhiteSpace(user_input2);
bool now3 = string.IsNullOrWhiteSpace(user_input3);
Console.WriteLine("IsNullOrWhiteSpace - null: " + now1);
Console.WriteLine("IsNullOrWhiteSpace - \"\": " + now2);
Console.WriteLine("IsNullOrWhiteSpace - \" \": " + now3);
Console.ReadKey();
so you can use either of these methods

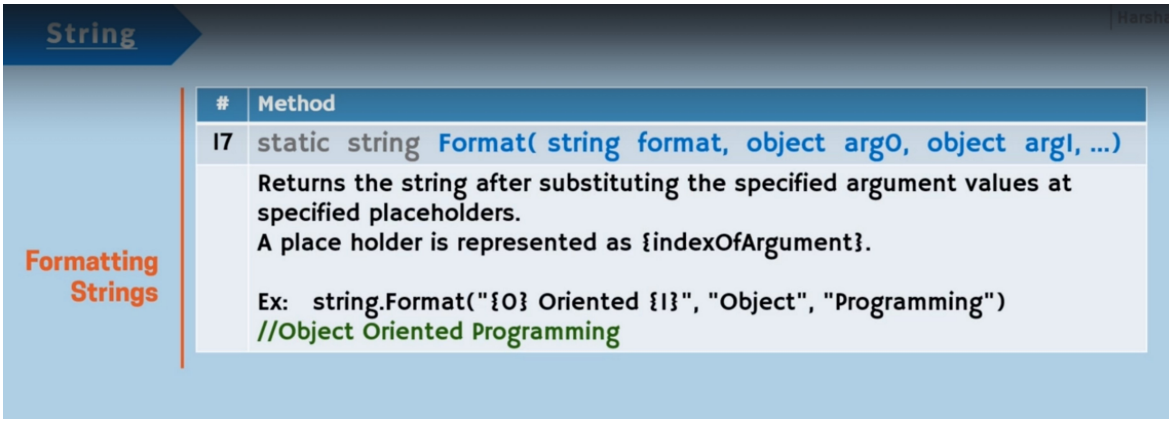

String
Checking Strings
Method
9 bool Equals(string otherString)
Returns a Boolean value that indicates whether the current string and the specified otherString are equal or not (each character will be compared).
10 bool StartsWith(string otherString)
Returns a Boolean value that indicates whether the current string beings with the specified otherString.
11 bool EndsWith(string otherString)
Returns a Boolean value that indicates whether the current string ends with the specified otherString.
12 bool Contains(string otherString)
Returns a Boolean value that indicates whether the current string contains the specified otherString anywhere.
Harsha
Web University



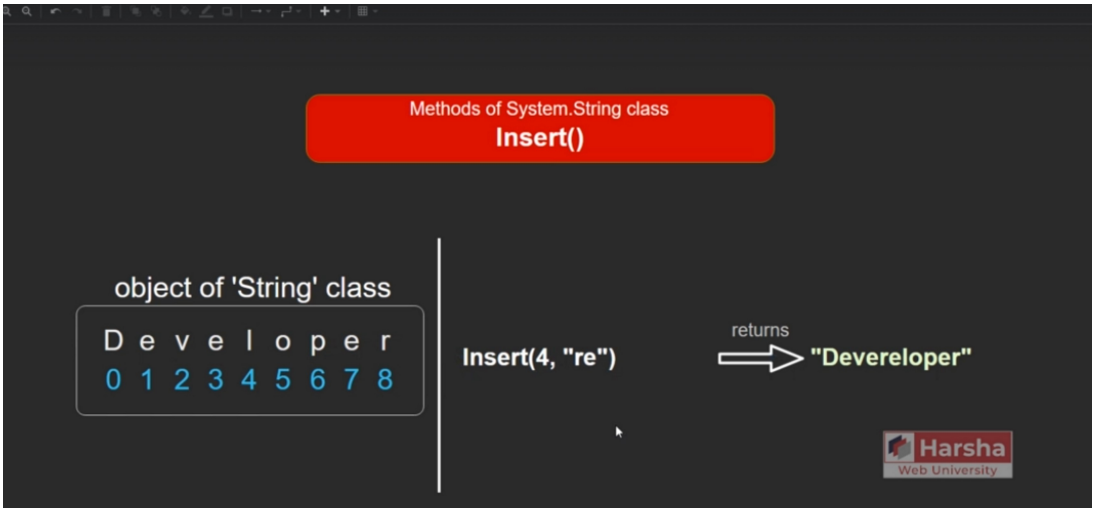
Formatting Strings

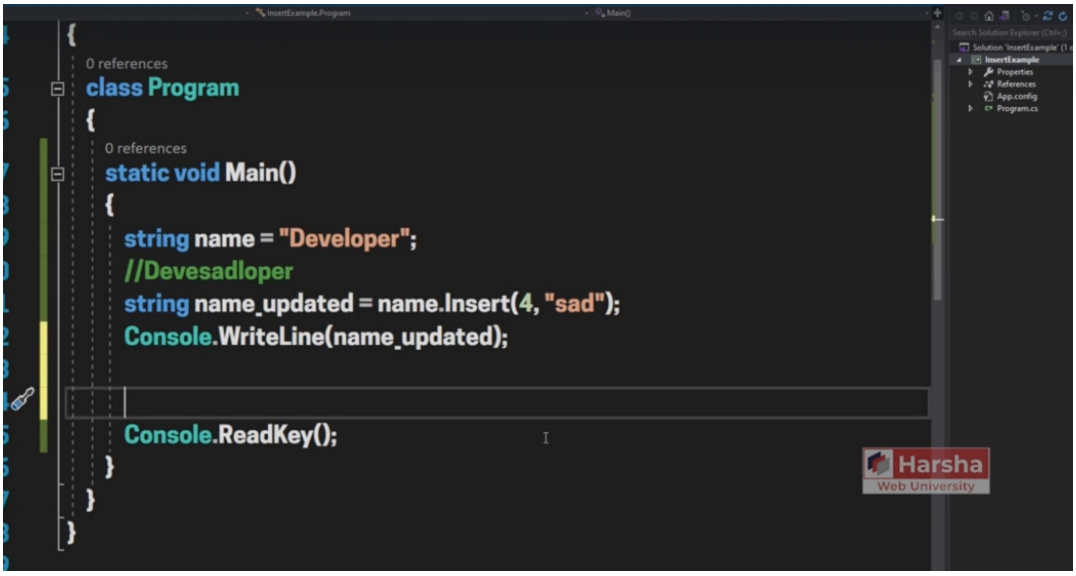


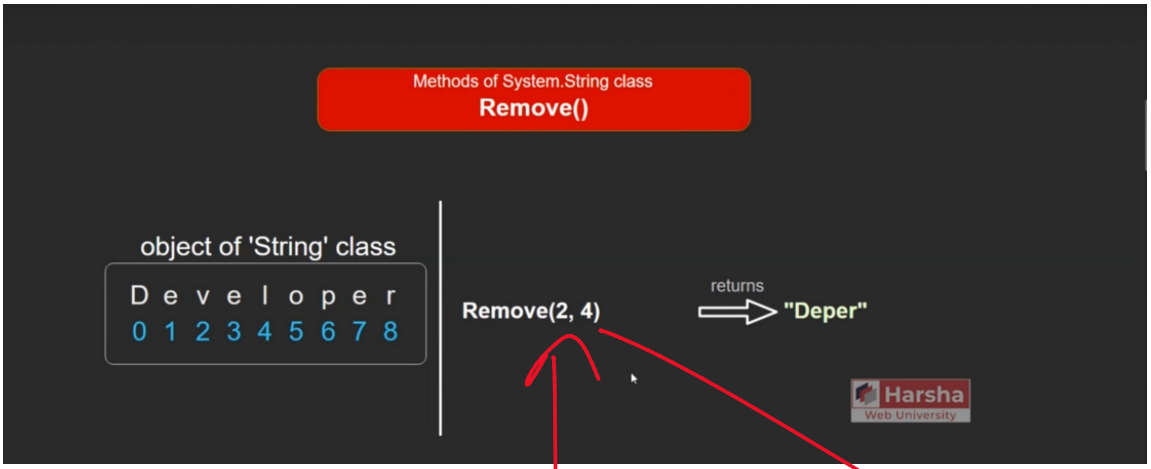

using System;
namespace FormattingStringsExample
{
0 references
class Program
{
0 references
static void Main()
{
string director = "Sam Raimi", movie = "Spider man";
string message = string.Format("{0} is the director of {1}", director, movie);
string message2 = $"{director} is the director of {movie}";
Console.WriteLine(message);
Console.WriteLine(message2);
Console.ReadKey();
}
}
}
Sam Raimi is the director of Spider man
Sam Raimi is the director of Spider man


String
Formatting Strings
Method
17 static string Format(string format, object arg0, object arg1, ...)
Returns the string after substituting the specified argument values at specified placeholders.
A place holder is represented as {indexOfArgument}.
Ex: string.Format("{0} Oriented {1}", "Object", "Programming")
//Object Oriented Programming


Modifying Strings







StartIndex

no. of character
(length)

StartIndex (length)

```
{
    0 references
    class Program
    {
        0 references
        static void Main()
        {
            string name = "Developer";
            //Devesadloper
            string name_updated = name.Insert(4, "sad");
            Console.WriteLine(name_updated);

            string name_updated2 = name.Remove(2, 4);
            Console.WriteLine(name_updated2);
            Console.ReadKey();
        }
    }
}
```

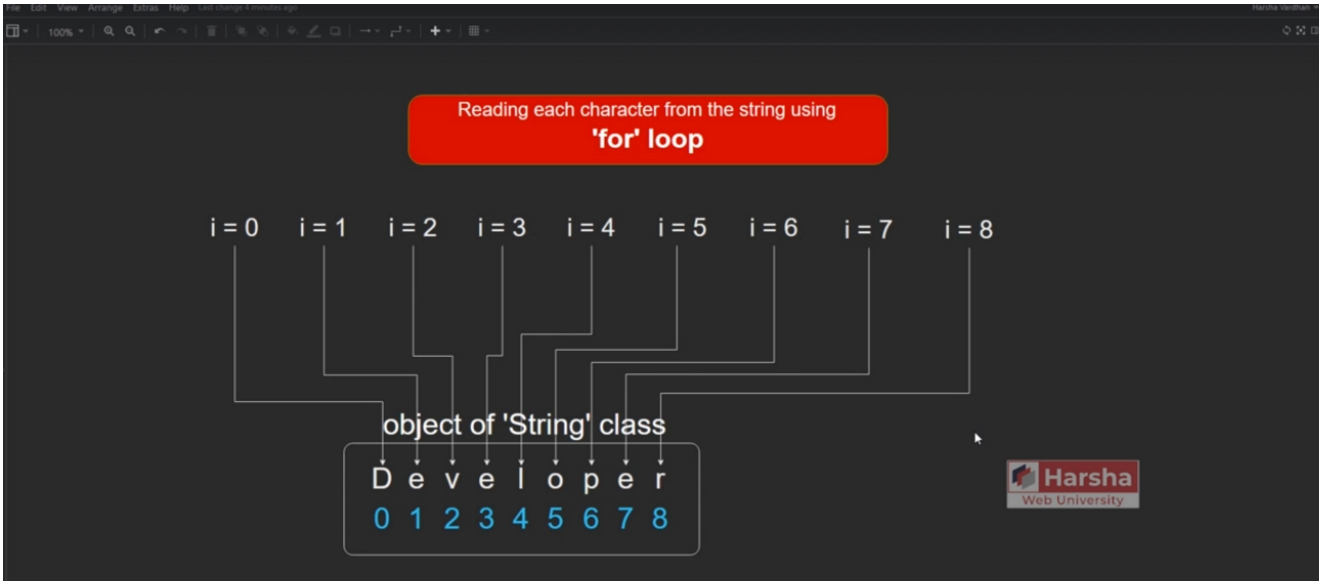
String

Modifying Strings

#	Method
18	string Insert(int startIndex, string value)
	Returns a new string object which inserts the new string value at the specified index in the existing string object.
19	string Remove(int startIndex, int count)
	It returns a new string object after removing specified count of characters at the specified start index, from the current string object.

Strings with 'for' loop

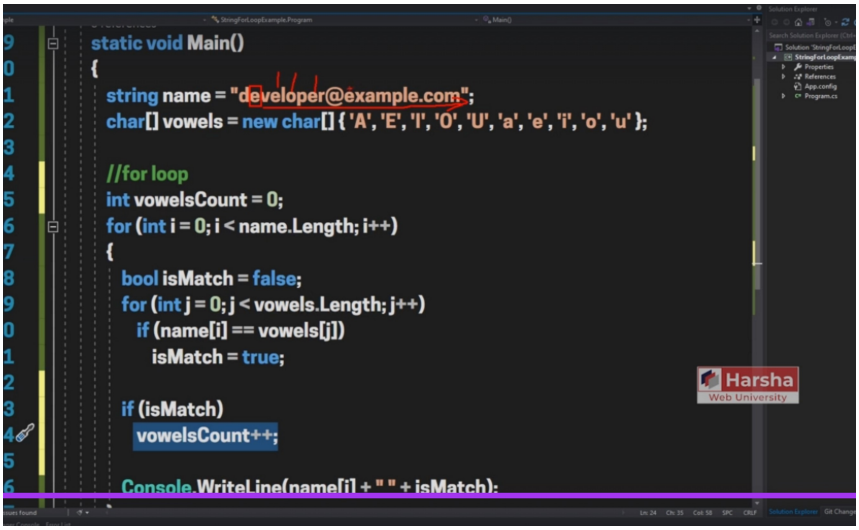
Part 1

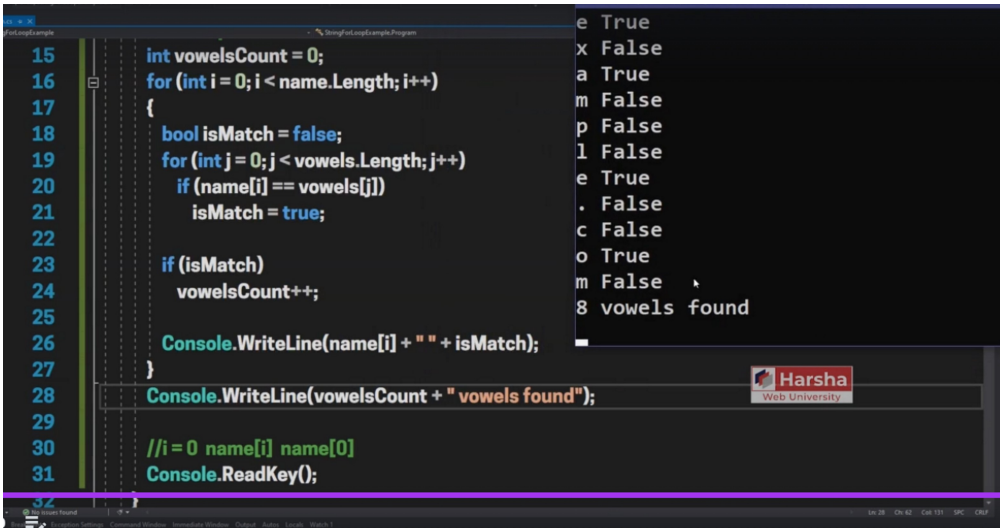


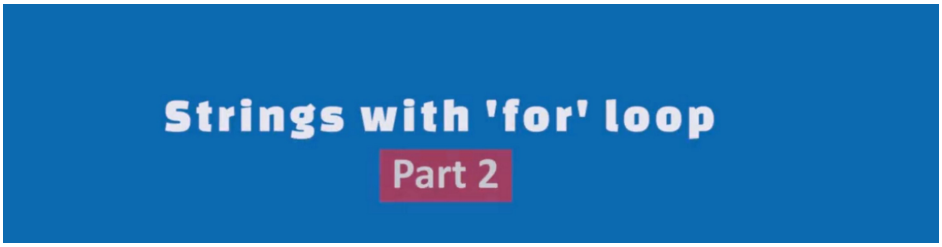
System.String class provides an Indexer

string str = "hello";

str[2];


static void Main()
{
string name = "developer@example.com";
char[] vowels = new char[] { 'A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u' };
//for loop
int vowelsCount = 0;
for (int i = 0; i < name.Length; i++)
{
bool isMatch = false;
for (int j = 0; j < vowels.Length; j++)
if (name[i] == vowels[j])
isMatch = true;
if (isMatch)
vowelsCount++;
Console.WriteLine(name[i] + " " + isMatch);
Harsha
Web University

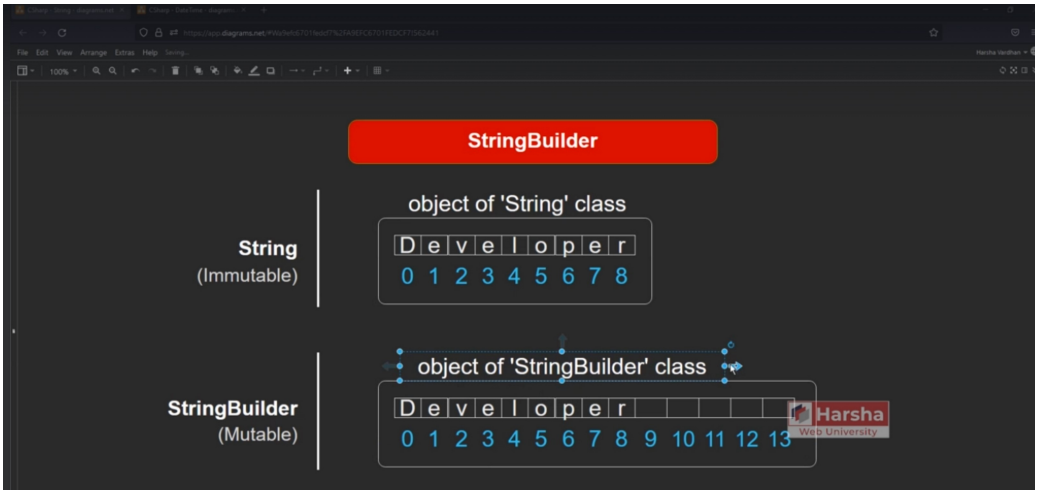

int vowelsCount = 0;
for (int i = 0; i < name.Length; i++)
{
bool isMatch = false;
for (int j = 0; j < vowels.Length; j++)
if (name[i] == vowels[j])
isMatch = true;
if (isMatch)
vowelsCount++;
Console.WriteLine(name[i] + " " + isMatch);
}
Console.WriteLine(vowelsCount + " vowels found");
//i = 0 name[i] name[0]
Console.ReadKey();
e True
x False
a True
m False
p False
l False
e True
. False
c False
o True
m False
8 vowels found
Harsha
Web University


Strings with 'for' loop
Part 2

```
9     static void Main()
10    {
11      string name = "developer@example.com";
12      char[] vowels = new char[] { 'A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u' };
30
31      //Using methods of arrays / collections
32      vowelsCount = 0;
33      for (int i = 0; i < name.Length; i++)
34      {
35        if (Array.IndexOf(vowels, name[i]) >= 0)
36          vowelsCount++;
37      }
38      Console.WriteLine(vowelsCount + " vowels found");
39      Console.ReadKey();
40    }
41  }
42 }
```


In this way, we can write the same code in a shortcut way if you know the proper usage of predefined methods.


```
static void Main()
{
  string name = "developer@example.com";
  char[] vowels = new char[] { 'A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u' };

  for (int i = 0; i < name.Length; i++)
  {
    if (Array.IndexOf(vowels, name[i]) >= 0)
      vowelsCount++;
  }
  Console.WriteLine(vowelsCount + " vowels found");

  //Using LINQ
  Console.WriteLine(name.Count(ch => Array.IndexOf(vowels, ch) >= 0));
  Console.ReadKey();
 }
}
```



string is immutable and string builder is mutable
so what does it mean exactly? the string object once created cannot be modified
means it is immutable means cannot be changed once it is created.

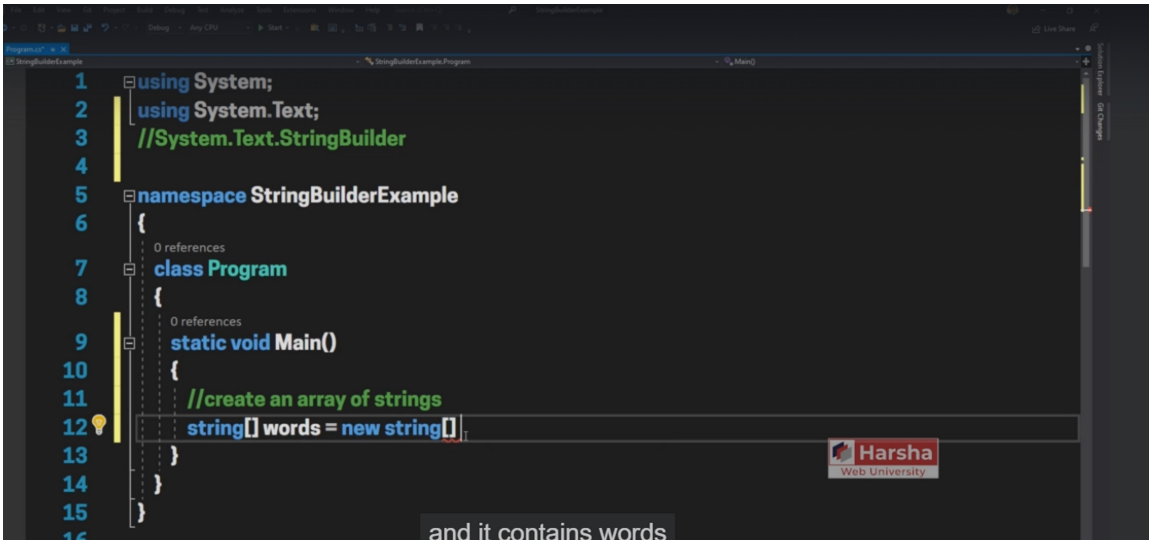
that is the design feature of string so what is the benefit of designing the string as immutable
because the string can be converted as hash. so the string can be used as key in the dictionary
okay that is the advantage of string being immutable.

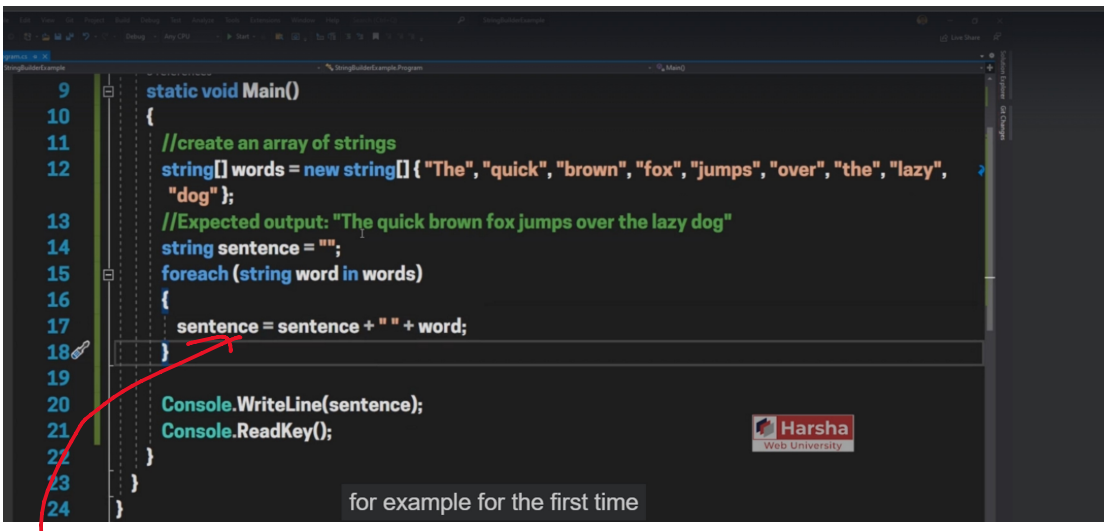
but in my case for example I would like to modify the string so what is the solution for that
so if you are expecting the string should be mutable means you should be able to make the changes in the string easily then you will use the string builder that means you will create a string builder object with some initial value and later you would like to modify certain characters or you want to append some extra substring means you would like to concatenate this existing string with another string or you want to remove the characters or you wanted to do continuous modifications on this string value that is absolutely possible in the string builder so in general by default you require to use the string order to store the string values but in case in particular cases that is you would like to keep performing continuous modifications on the string value like for example you would like to modify the string value in a loop and you don't know how many times that should be executed in this kind of cases you will use the string builder over the string.

so if you don't have a plan of modification or you wanted to do just one or two modifications in the string then the string is better to use so if you want to perform continuous modifications such as

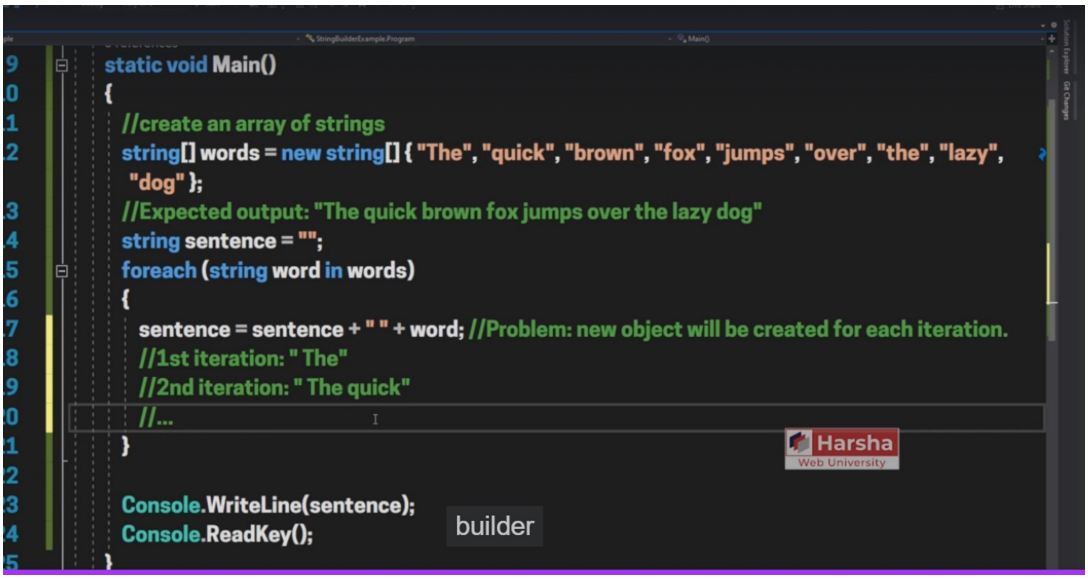
concatenating some extra string in a loop and that loop executes multiple times for this kind of cases string builder is better.

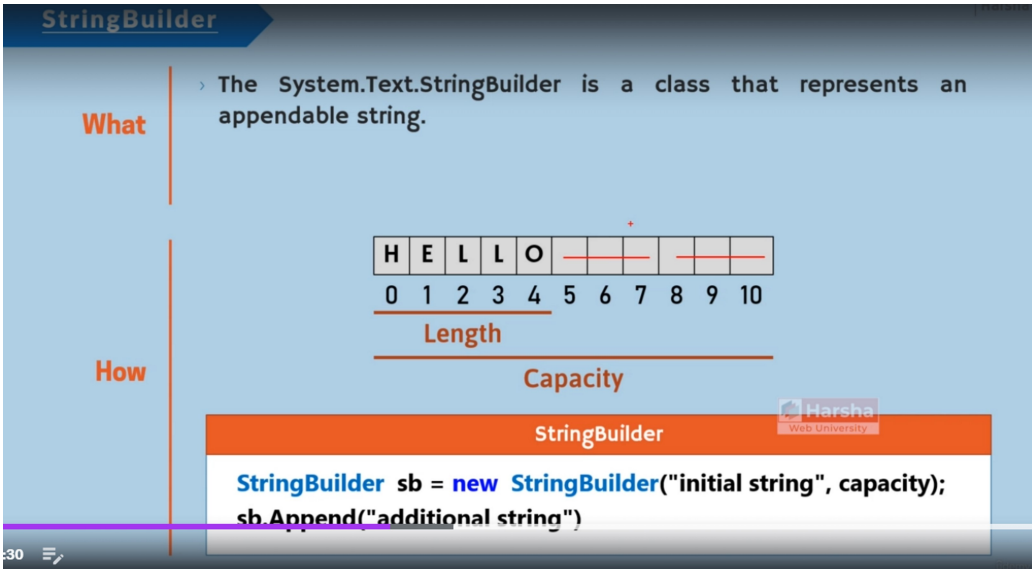
let's say for example there is a messenger application such as whatsapp.
every time when the user sends a message you would like to add that message in a
string means every time when the user sends a new message you would like to add that message to an existing string but in this case the point is you really don't know how many messages will be
sent by the user it can be one or two or hundreds of or thousands of messages so that is the reason in this case string builder is better than string because you would like to modify means
append the strings multiple times but for normal cases such as storing the username email etc the string is okay to use.



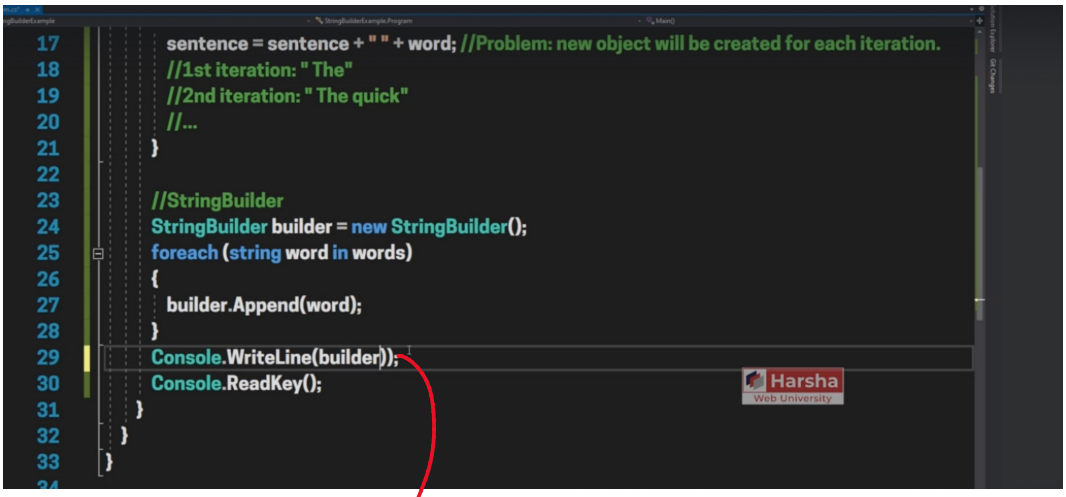


Every concatenation creates a new string object unnecessarily.

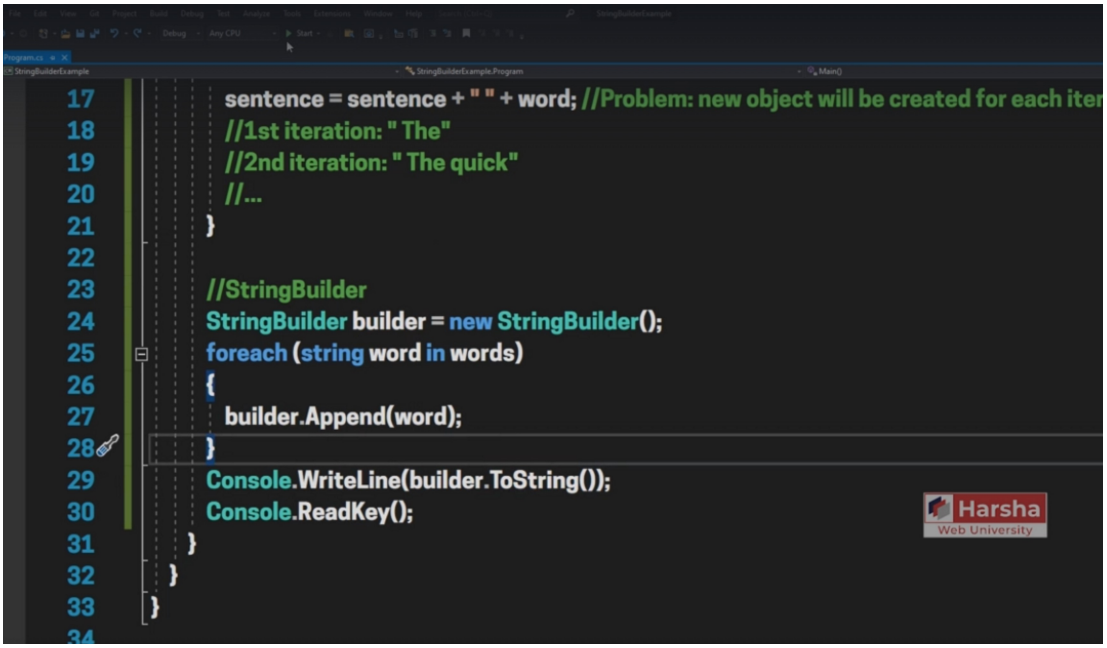


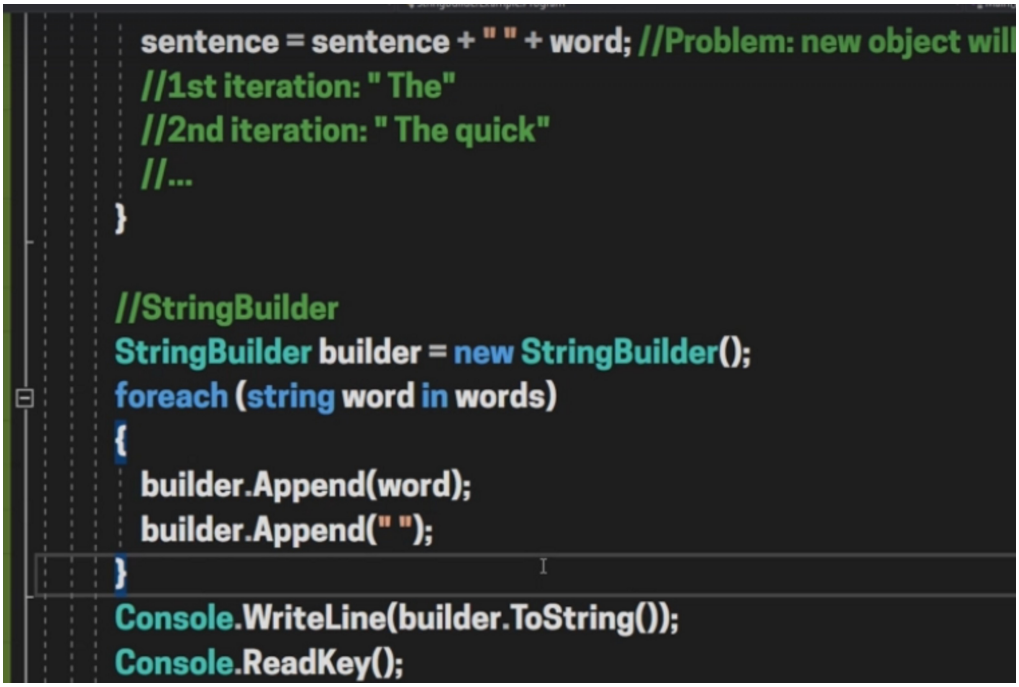


Space automatically increases internally without creating new object.



Internally calls 'builder.toString()' because WriteLine() mehtod can not print the object


sentence = sentence + " " + word; //Problem: new object will be created for each iter
//1st iteration: " The"
//2nd iteration: " The quick"
//...
}
//StringBuilder
StringBuilder builder = new StringBuilder();
foreach (string word in words)
{
builder.Append(word);
}
Console.WriteLine(builder.ToString());
Console.ReadKey();
Harsha
Web University

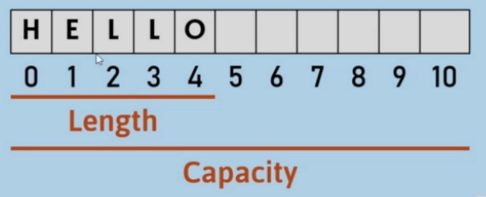

sentence = sentence + " " + word; //Problem: new object will
//1st iteration: " The"
//2nd iteration: " The quick"
//...
}
//StringBuilder
StringBuilder builder = new StringBuilder();
foreach (string word in words)
{
builder.Append(word);
builder.Append(" ");
}
Console.WriteLine(builder.ToString());
Console.ReadKey();

StringBuilder

What

- The System.Text.StringBuilder is a class that represents an appendable string.
- The string value stored in StringBuilder can be modified, without recreating StringBuilder object.

How



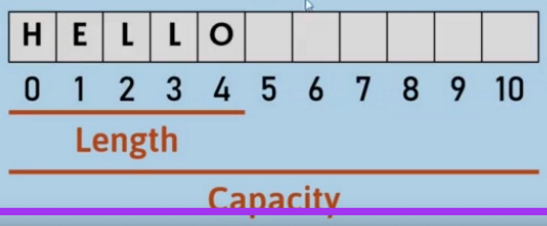
StringBuilder

```
StringBuilder sb = new StringBuilder("initial string", capacity);
sb.Append("additional string")
```


StringBuilder

'Capacity' of String Builder

- The 'Capacity' property of String Builder represents the number of characters that can be stored in the string builder, as per current memory allocation.
- When the programmer tries to store much more number of characters than the Capacity, String Builder automatically increases the 'Capacity' property to its double.
- The default value of 'Capacity' is 16; the programmer can assign its value via constructor or set accessor of the 'Capacity' property.



```
17          sentence = sentence + " " + word; //Problem: new object will be created for each iteration.
18          //1st iteration: " The"
19          //2nd iteration: " The quick"
20          //...
21        }
22
23        //StringBuilder
24        StringBuilder builder = new StringBuilder();
25        foreach (string word in words)
26        {
27          builder.Append(word);
28          builder.Append(" ");
29          Console.WriteLine(builder.ToString() + ", " + builder.Capacity);
30        }
31        Console.WriteLine(builder.ToString());
32        Console.ReadKey();
33      }
34    }
```

Harsha Web University

```
The , 16
The quick , 16
The quick brown , 16
The quick brown fox , 32
The quick brown fox jumps , 32
The quick brown fox jumps over , 32
The quick brown fox jumps over the , 64
The quick brown fox jumps over the lazy , 64
The quick brown fox jumps over the lazy dog , 64
The quick brown fox jumps over the lazy dog
```

Harsha Web University

Methods of StringBuilder

StringBuilder

'StringBuilder' class Constructor

Constructor	Description
StringBuilder( )	It initializes the StringBuilder type of object with the default capacity (16).
StringBuilder(int capacity)	It initializes the StringBuilder type of object with the specified capacity.
StringBuilder(string value)	It initializes the StringBuilder type of object with the specified value.
StringBuilder(string value, int capacity)	It initializes the StringBuilder type of object with the specified value and capacity.

StringBuilder

StringBuilder Properties

#	Property
1	int Length { get; set; }
	This property gets / sets the length (number of characters) in the string builder.
2	char [int index] { get; set; }
	This indexer gets / sets the single character at the specified character index.
3	int Capacity { get; set; }
	This property returns the number of characters that can be stored in the current string builder (currently memory allocated). Default is 16.
4	int MaxCapacity { get; }
	This property represents maximum number of characters up to which, the Capacity can be extended. Default is int.MaxValue.

```
20          //...
21      }
22      //Console.WriteLine(sentence);
23
24      //StringBuilder
25      StringBuilder builder = new StringBuilder("hello ", 20);
26      foreach (string word in words)
27      {
28        builder.Append(word);
29        builder.Append(" ");
30        Console.WriteLine(builder.ToString() + ", " + builder.Length + ", " + builder.Capacity);
31      }
32      builder[0] = 'v';
33      Console.WriteLine(builder.ToString());
34      Console.ReadKey();
35    }
36  }
```

Harsha Web University

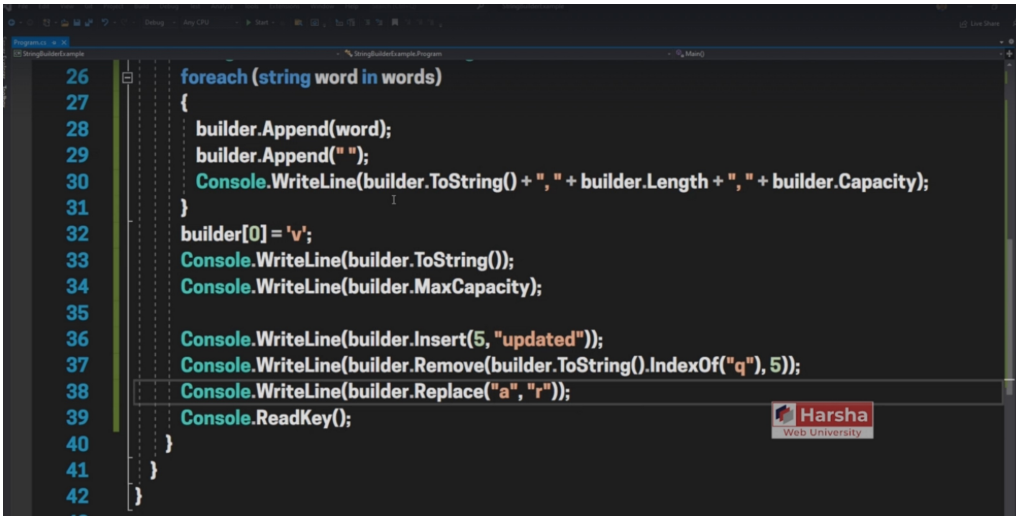
StringBuilder

Methods of String Builder

#	Method
1	string Append(string value)
	Adds the given string value at the end of current value of string builder.
2	string Insert(int startIndex, string value)
	Returns a new string object which inserts the new string value at the specified index in the existing string object.
3	string Remove(int startIndex, int count)
	It returns a new string object after removing specified count of characters at the specified start index, from the current string object.
4	string Replace(string oldString, string newString)
	Returns a string after replacing all occurrences of oldString with newString.
5	string ToString()
	Returns the current value of string builder as a string object.

```
26      foreach (string word in words)
27      {
28        builder.Append(word);
29        builder.Append(" ");
30        Console.WriteLine(builder.ToString() + ", " + builder.Length + ", " + builder.Capacity);
31      }
32      builder[0] = 'v';
33      Console.WriteLine(builder.ToString());
34      Console.WriteLine(builder.MaxCapacity);
35
36      Console.WriteLine(builder.Insert(5, "updated"));
37      Console.WriteLine(builder.Remove(builder.ToString().IndexOf("q"), 5));
38      Console.ReadKey();
39    }
40  }
41  }
```

Harsha Web University



```
26          foreach (string word in words)
27          {
28            builder.Append(word);
29            builder.Append(" ");
30            Console.WriteLine(builder.ToString() + ", " + builder.Length + ", " + builder.Capacity);
31          }
32          builder[0] = 'v';
33          Console.WriteLine(builder.ToString());
34          Console.WriteLine(builder.MaxCapacity);
35
36          Console.WriteLine(builder.Insert(5, "updated"));
37          Console.WriteLine(builder.Remove(builder.ToString().IndexOf("q"), 5));
38          Console.WriteLine(builder.Replace("a", "r"));
39          Console.ReadKey();
40        }
41      }
42    }
```


StringBuilder

Methods of String Builder

#	Method
1	string Append(string value)
	Adds the given string value at the end of current value of string builder.
2	string Insert(int startIndex, string value)
	Returns a new string object which inserts the new string value at the specified index in the existing string object.
3	string Remove(int startIndex, int count)
	It returns a new string object after removing specified count of characters at the specified start index, from the current string object.
4	string Replace(string oldString, string newString)
	Returns a string after replacing all occurrences of oldString with newString.
5	string ToString()
	Returns the current value of string builder as a string object.

StringBuilder

When to use 'String'

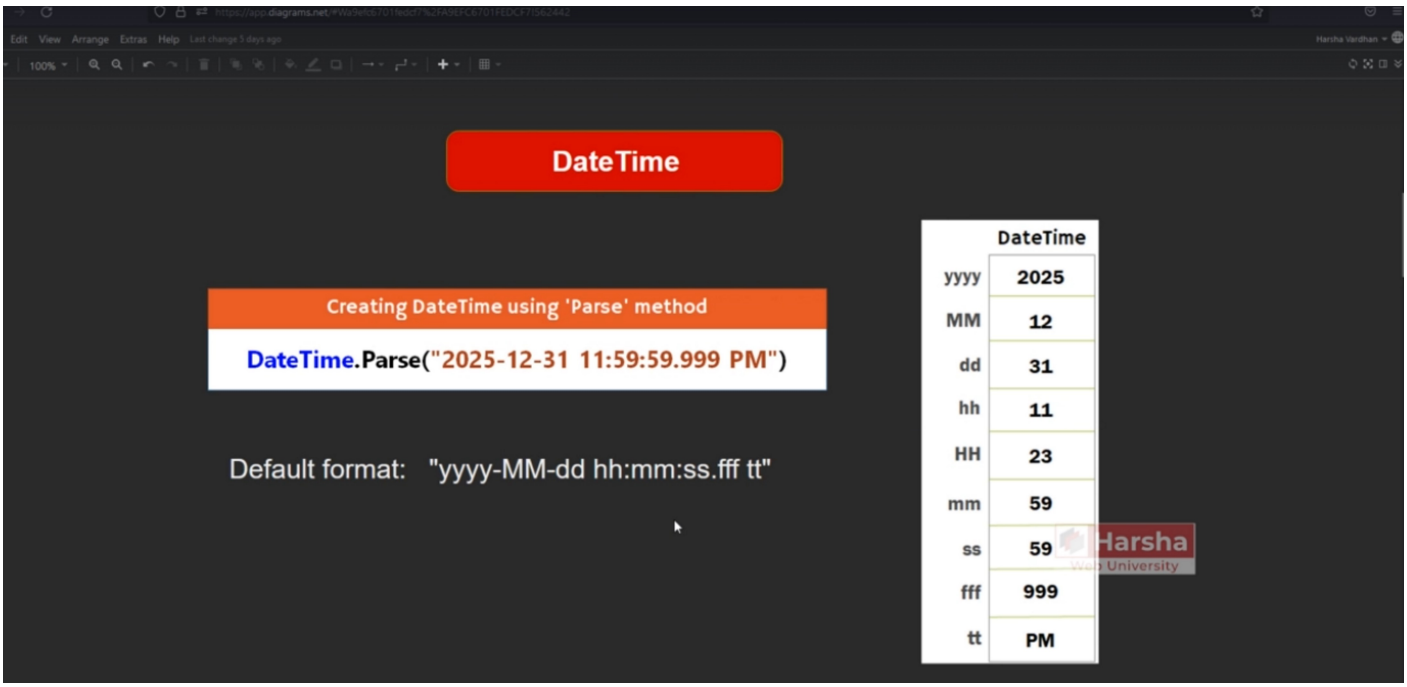
- When you would like to make less number of changes to the string.
- When you perform limited number of concatenation operations.
- When you want to perform extensive search operations using methods such as IndexOf, Contains, StartsWith etc.

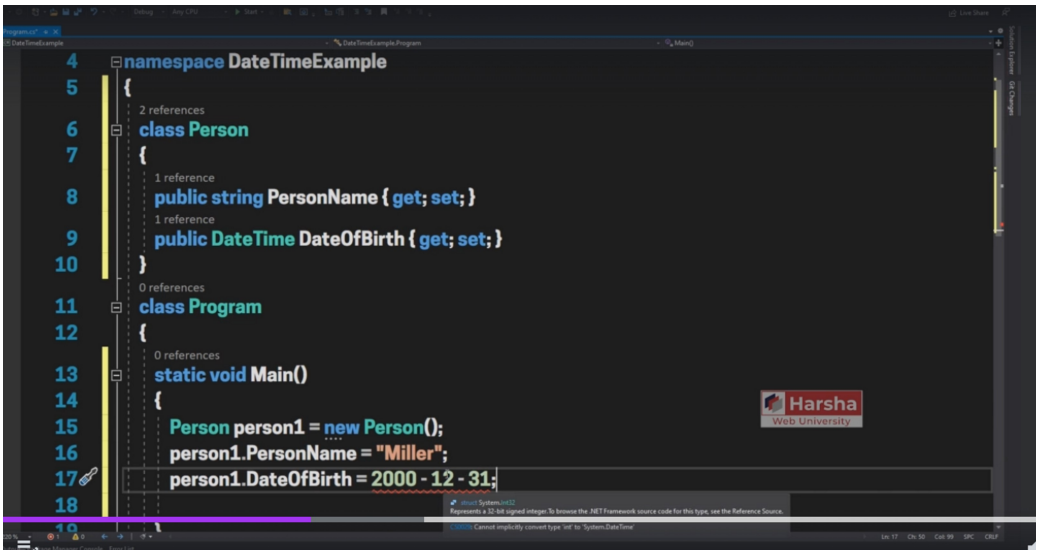
When to use 'String Builder'

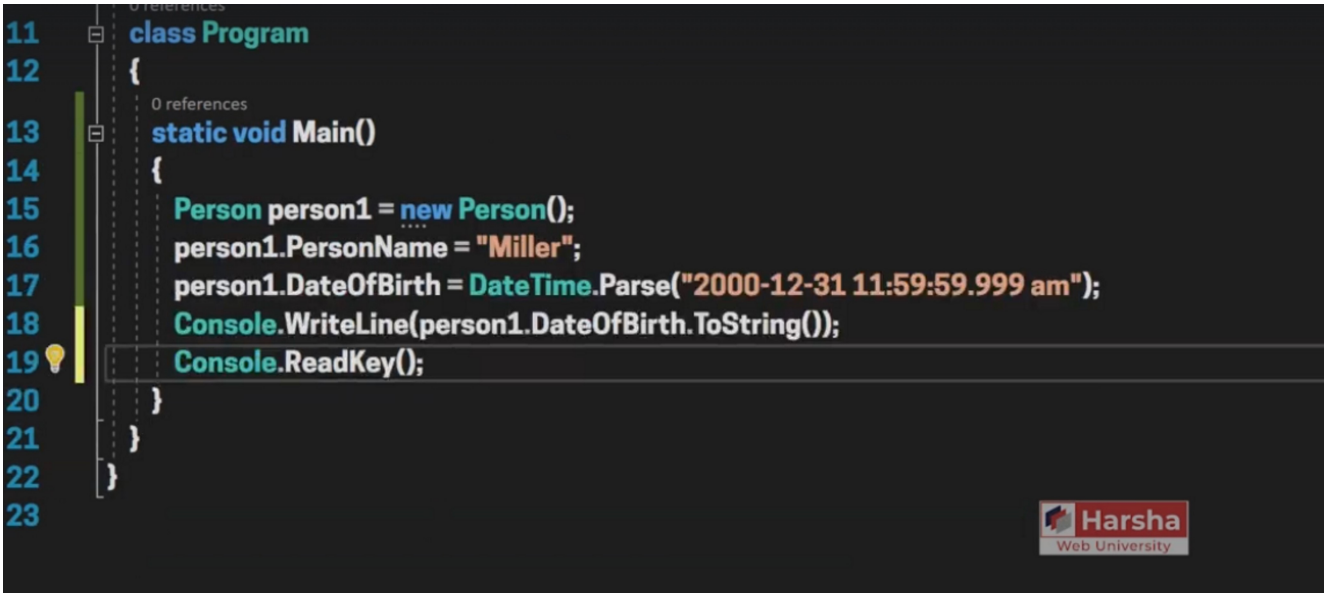
- When you would like to unknown number of / extensive number of changes to a string (Eg: making changes / concatenations to string through a loop).
- When you perform less number of search operations on strings.

DateTime

Part 1

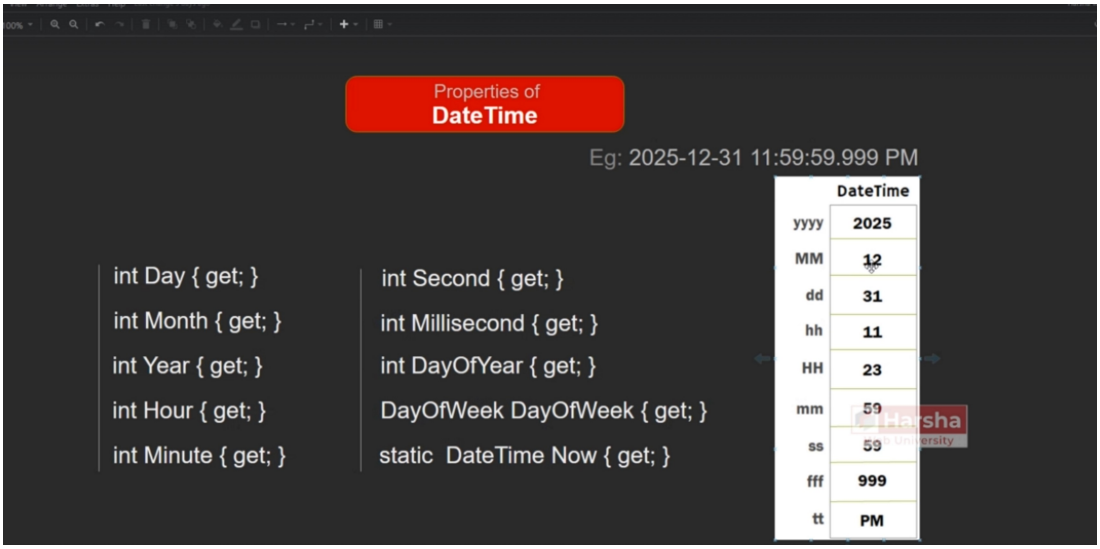


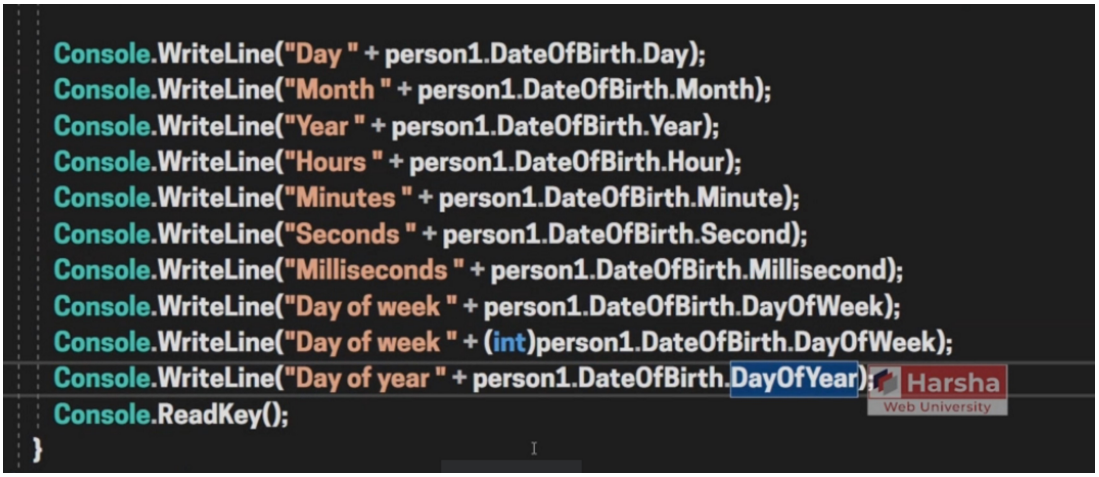

namespace DateTimeExample
{
2 references
class Person
{
1 reference
public string PersonName { get; set; }
1 reference
public DateTime DateOfBirth { get; set; }
}
0 references
class Program
{
0 references
static void Main()
{
Person person1 = new Person();
person1.PersonName = "Miller";
person1.DateOfBirth = 2000 - 12 - 31;
Harsha
Web University


class Program
{
0 references
static void Main()
{
Person person1 = new Person();
person1.PersonName = "Miller";
person1.DateOfBirth = DateTime.Parse("2000-12-31 11:59:59.999 am");
Console.WriteLine(person1.DateOfBirth.ToString());
Console.ReadKey();
}
}
}
Harsha
Web University

DateTime

Part 2





```
DateTime dt = DateTime.Now;
Console.WriteLine(dt.ToString());

Console.ReadKey();
```

```
static void Main()
{
    Person person1 = new Person();
    person1.PersonName = "Miller";
    person1.DateOfBirth = DateTime.Parse("2000-12-31 11:59:59.999 am");
    person1.DateOfBirth = Convert.ToDateTime("2000-12-31 11:59:59.999 am");
    Console.WriteLine(person1.DateOfBirth.ToString());

    Console.WriteLine("Day " + person1.DateOfBirth.Day);
    Console.WriteLine("Month " + person1.DateOfBirth.Month);
    Console.WriteLine("Year " + person1.DateOfBirth.Year);
    Console.WriteLine("Hours " + person1.DateOfBirth.Hour);
    Console.WriteLine("Minutes " + person1.DateOfBirth.Minute);
    Console.WriteLine("Seconds " + person1.DateOfBirth.Second);
    Console.WriteLine("Milliseconds " + person1.DateOfBirth.Millisecond);
    Console.WriteLine("Day of week " + person1.DateOfBirth.DayOfWeek);
    Console.WriteLine("Day of week " + (int)person1.DateOfBirth.DayOfWeek);
    Console.WriteLine("Day of year " + person1.DateOfBirth.DayOfYear);
```


DateTime

What

- The System.DateTime is a structure that represents date and time value.
- Default format: yyyy-MM-dd hh:mm:ss.fff tt

	DateTime
yyyy	2025
MM	12
dd	31
hh	11
HH	23
mm	59
ss	59
fff	999
tt	PM

How

Creating DateTime using 'Parse' method

```
DateTime.Parse("2025-12-31 11:59:59.999 PM")
```

Creating DateTime using 'ToDateTime' method

```
Convert.ToDateTime("2025-12-31 11:59:59.999 PM")
```


DateTime Formats

Part 1

Methods of
DateTime

string ToString()	string ToShortTimeString()
string ToString(string format)	string ToLongTimeString()
string ToShortDateString()	static int DaysInMonth(int year, int month)
string ToLongDateString()	static DateTime Parse(string value)

static DateTime ParseExact(string value, string format,
IFormatProvider provider, DateTimeStyle style)

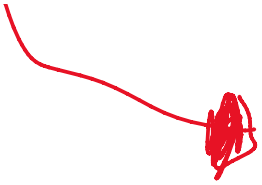
Harsha Web University

```
        DateTime dt = DateTime.Parse("2020-12-31 11:59:59.999");
        string str1 = dt.ToString(); //MM/dd/yyyy hh:mm:ss tt
        string str2 = dt.ToShortDateString(); //MM/dd/yyyy
        string str3 = dt.ToLongDateString(); //dd MMMM yyyy
        string str4 = dt.ToShortTimeString(); //hh:mm tt
        string str5 = dt.ToLongTimeString(); //hh:mm:ss tt

        Console.WriteLine(str1);
        Console.WriteLine(str2);
        Console.WriteLine(str3);
        Console.WriteLine(str4);
        Console.WriteLine(str5);
        Console.ReadKey();
    }
}
}
```

Harsha Web University

```
0 references
static void Main()
{
    DateTime dt = DateTime.Parse("2020-12-31 11:59:59.999");
    string str1 = dt.ToString(); //MM/dd/yyyy hh:mm:ss tt
    string str2 = dt.ToShortDateString(); //MM/dd/yyyy
    string str3 = dt.ToLongDateString(); //dd MMMM yyyy
    string str4 = dt.ToShortTimeString(); //hh:mm tt
    string str5 = dt.ToLongTimeString(); //hh:mm:ss tt
    string str6 = dt.ToString("dd-MM-yyyy HH:mm:ss");

    Console.WriteLine(str1);
    Console.WriteLine(str2);
    Console.WriteLine(str3);
    Console.WriteLine(str4);
    Console.WriteLine(str5);
    Console.ReadKey();
}
```

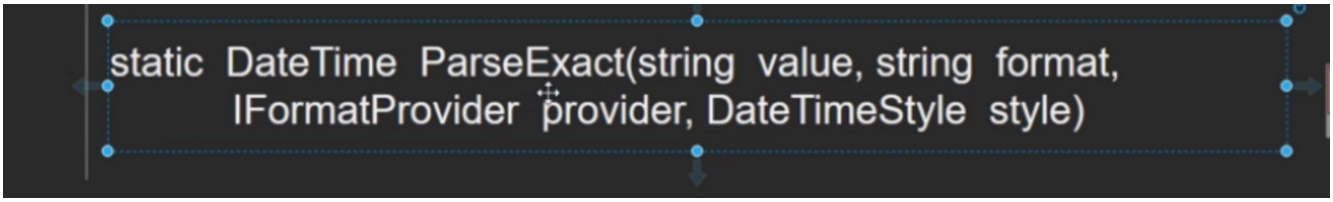
Harsha Web University

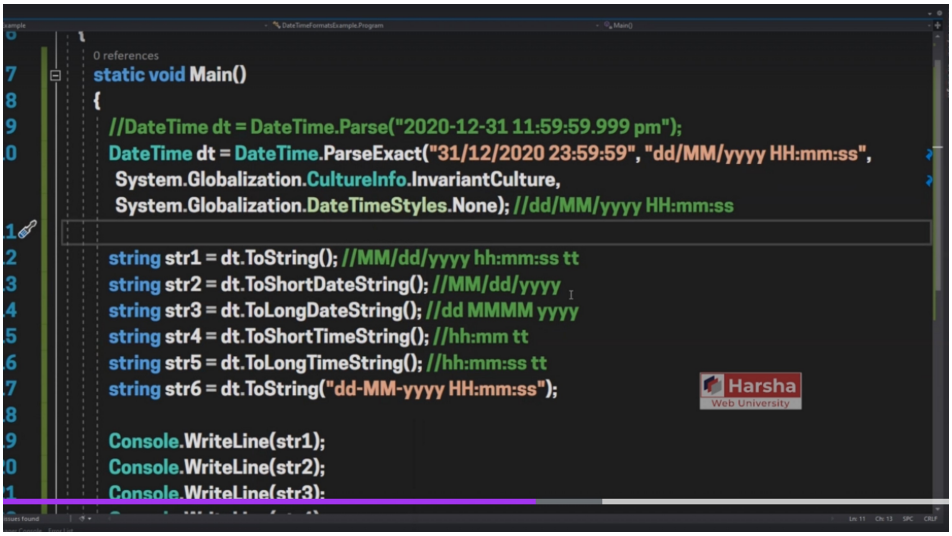


you can supply any other format.

DateTime.Parse() => this method works if the use enters (year-month-date) or (month-date-year) format.

but it doesn't work with other custom specific format. in that case, use





```
0 references
static void Main()
{
    //DateTime dt = DateTime.Parse("2020-12-31 11:59:59.999 pm");
    DateTime dt = DateTime.ParseExact("31/12/2020 23:59:59", "dd/MM/yyyy HH:mm:ss",
     System.Globalization.CultureInfo.InvariantCulture,
     System.Globalization.DateTimeStyles.None); //dd/MM/yyyy HH:mm:ss

    string str1 = dt.ToString(); //MM/dd/yyyy hh:mm:ss tt
    string str2 = dt.ToShortDateString(); //MM/dd/yyyy
    string str3 = dt.ToLongDateString(); //dd MMMM yyyy
    string str4 = dt.ToShortTimeString(); //hh:mm tt
    string str5 = dt.ToLongTimeString(); //hh:mm:ss tt
    string str6 = dt.ToString("dd-MM-yyyy HH:mm:ss");
```



```
    string str2 = dt.ToShortDateString(); //MM/dd/yyyy
    string str3 = dt.ToLongDateString(); //dd MMMM yyyy
    string str4 = dt.ToShortTimeString(); //hh:mm tt
    string str5 = dt.ToLongTimeString(); //hh:mm:ss tt
    string str6 = dt.ToString("dd-MM-yyyy HH:mm:ss");

    Console.WriteLine(str1);
    Console.WriteLine(str2);
    Console.WriteLine(str3);
    Console.WriteLine(str4);
    Console.WriteLine(str5);
    Console.WriteLine(str6);
    Console.WriteLine(DateTime.DaysInMonth(2022, 7));
    Console.ReadKey();
  }
 }
}
```


DateTime

DateTime Methods

#	Method
1	string ToString()
	It returns the date & time value in the default date format, based on current Windows settings.
2	string ToString(string format)
	It returns the date & time value in the specified format.
3	string ToShortDateString()
	It returns the date value in the default short date format, based on current Windows settings. Eg: MM/dd/yyyy | 12/31/2030
4	string ToLongDateString()
	It returns the date value in the default long date format, based on current Windows settings. Eg: dd MMMM yyyy | 31 December 2030

DateTime

DateTime Methods

#	Method
5	string ToShortTimeString()
	It returns the date value in the default short time format, based on current Windows settings. Eg: hh:mm tt | 11:59 pm
6	string ToLongTimeString()
	It returns the date value in the default long time format, based on current Windows settings. Eg: hh:mm:ss tt | 11:59:59 pm
7	static int DaysInMonth(int year, int month)
	It returns number of days in the specified month in the specified year.

DateTime

DateTime Properties

#	Property
1	int Day { get; }
	It returns the day (1 to 31) of current date & time value.
2	int Month { get; }
	It returns the month (1 to 12) of current date & time value.
2	int Year { get; }
	It returns the year (1 to 9999) of current date & time value.
3	int Hour { get; }
	It returns the hour (0 to 23) of current date & time value.
4	int Minute { get; }
	It returns the minute (0 to 59) of current date & time value.
5	int Second { get; }
	It returns the second (0 to 59) of current date & time value.

DateTime

DateTime Properties

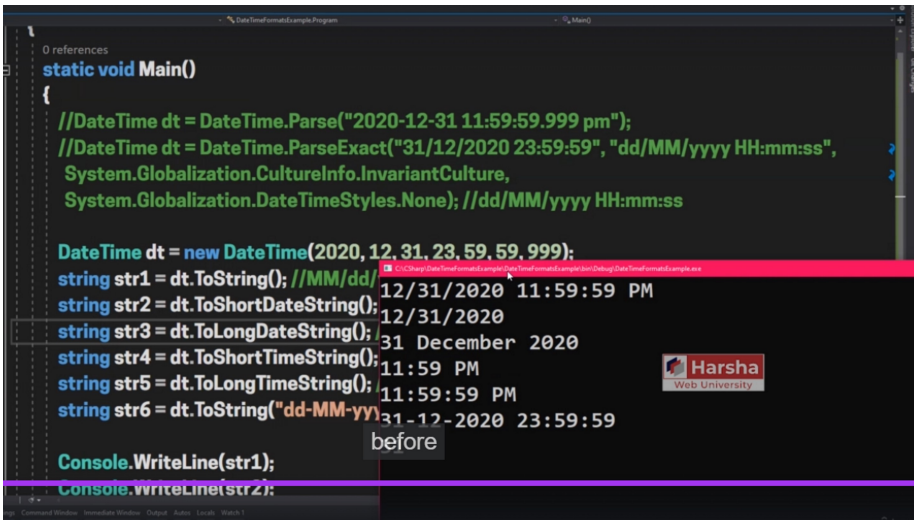
#	Property
6	int Millisecond { get; }
	It returns the millisecond (0 to 999) of current date & time value.
7	int DayOfYear { get; }
	It returns the day of year (1 to 366) based on current date & time value.
8	DayOfWeek DayOfWeek { get; }
	It returns the day of week - Sunday to Saturday (0 to 6) based on current date & time value.
9	static DateTime Now { get; }
	It returns an instance of DateTime structure that represents the current system date & time.

DateTime

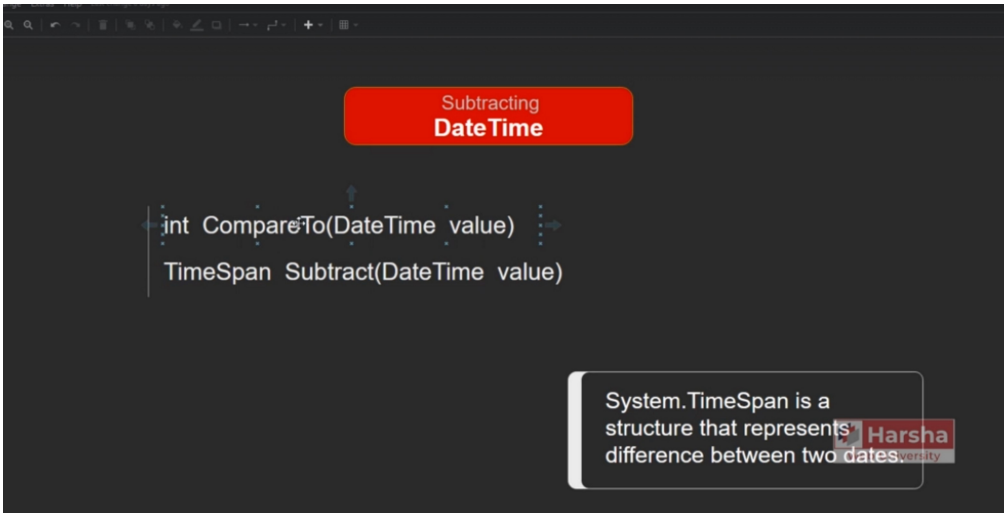
Harsha

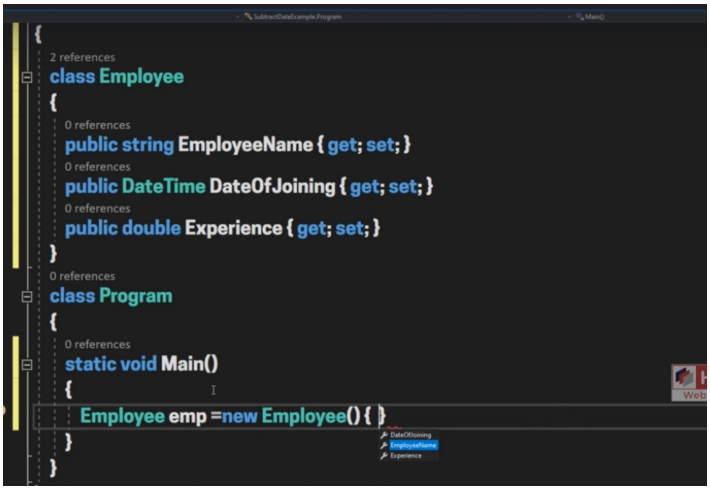
DateTime Constructors

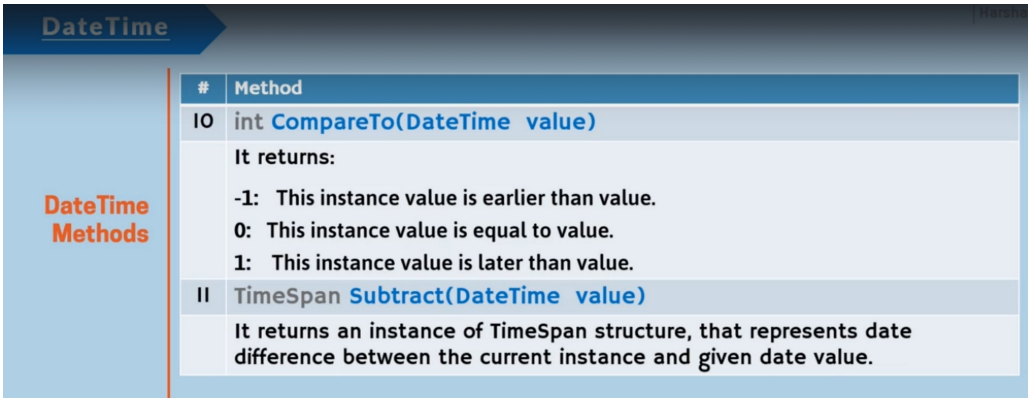
#	Property
I	DateTime(int year, int month, int day, int hour, int minute, int second, int millisecond)
	It creates a new instance (structure instance) with the specified date and time values. "hour" should specified 24-hour format (0 to 23).

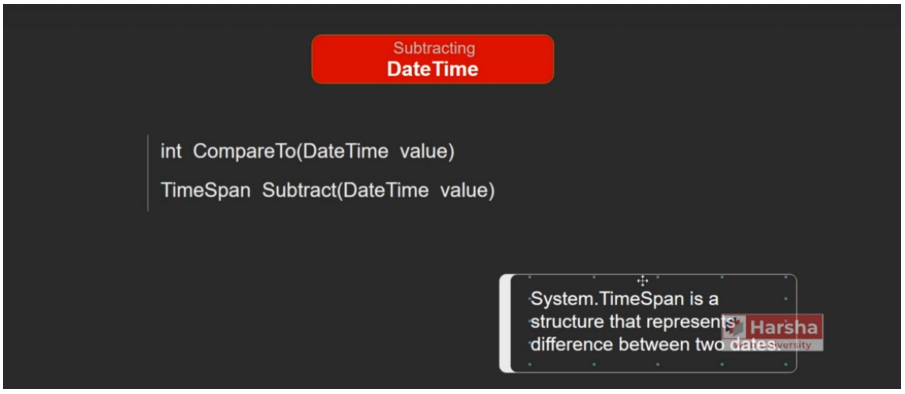


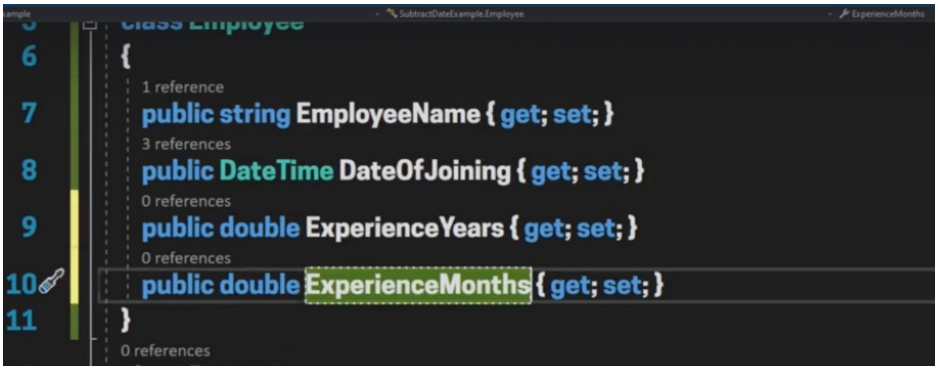
Date Subtraction

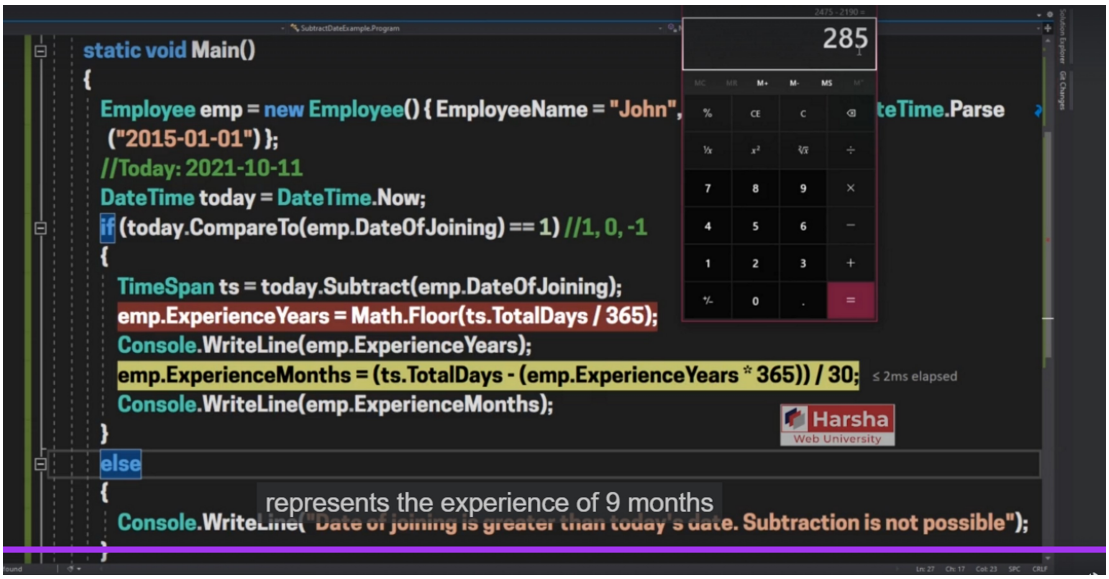

Subtracting
DateTime
int CompareTo(DateTime value)
TimeSpan Subtract(DateTime value)
System.TimeSpan is a structure that represents difference between two dates.
Harsha


{
2 references
class Employee
{
0 references
public string EmployeeName { get; set; }
0 references
public DateTime DateOfJoining { get; set; }
0 references
public double Experience { get; set; }
}
0 references
class Program
{
0 references
static void Main()
{
Employee emp =new Employee() { }
}
}


DateTime
DateTime Methods
Method
10 int CompareTo(DateTime value)
It returns:
-1: This instance value is earlier than value.
0: This instance value is equal to value.
1: This instance value is later than value.
11 TimeSpan Subtract(DateTime value)
It returns an instance of TimeSpan structure, that represents date difference between the current instance and given date value.


Subtracting
DateTime
int CompareTo(DateTime value)
TimeSpan Subtract(DateTime value)
System.TimeSpan is a structure that represents the difference between two dates
Harsha


{
1 reference
public string EmployeeName { get; set; }
3 references
public DateTime DateOfJoining { get; set; }
0 references
public double ExperienceYears { get; set; }
0 references
public double ExperienceMonths { get; set; }
}
0 references


static void Main()
{
Employee emp = new Employee() { EmployeeName = "John",
("2015-01-01") };
//Today: 2021-10-11
DateTime today = DateTime.Now;
if (today.CompareTo(emp.DateOfJoining) == 1) //1, 0, -1
{
TimeSpan ts = today.Subtract(emp.DateOfJoining);
emp.ExperienceYears = Math.Floor(ts.TotalDays / 365);
Console.WriteLine(emp.ExperienceYears);
emp.ExperienceMonths = (ts.TotalDays - (emp.ExperienceYears * 365)) / 30;
Console.WriteLine(emp.ExperienceMonths);
}
else
{
Console.WriteLine("Date of joining is greater than today's date. Subtraction is not possible");
}
285
≤ 2ms elapsed
Harsha
Web University
represents the experience of 9 months

```
static void Main()
{
  Employee emp = new Employee() { EmployeeName = "John", DateOfJoining = DateTime.Parse
   ("2015-01-01") };
  //Today: 2021-10-11
  DateTime today = DateTime.Now;
  if (today.CompareTo(emp.DateOfJoining) == 1) //1, 0, -1
  {
    TimeSpan ts = today.Subtract(emp.DateOfJoining);
    emp.ExperienceYears = Math.Floor(ts.TotalDays / 365);

    emp.ExperienceMonths = Math.Floor((ts.TotalDays - (emp.ExperienceYears * 365)) / 30);
    Console.WriteLine(emp.ExperienceYears + " years and " + emp.ExperienceMonths + "
     months");
  }
  else
  {
    Console.WriteLine("Date of joining is greater than today's date. Subtraction is not
```

Harsha Web University

Date Addition

Methods of
DateTime

DateTime AddDays(double value)

DateTime AddMonths(double value)

DateTime AddYears(double value)

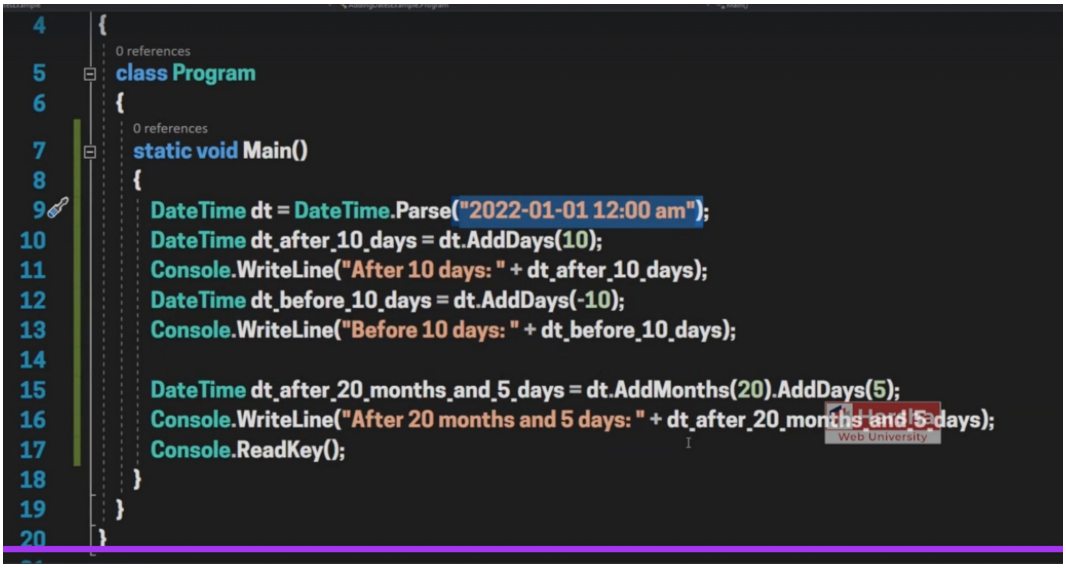
DateTime AddHours(double value)

DateTime AddMinutes(double value)

DateTime AddSeconds(double value)

DateTime AddMilliseconds(double value)

Harsha Web University



```
{
    0 references
    class Program
    {
        0 references
        static void Main()
        {
            DateTime dt = DateTime.Parse("2022-01-01 12:00 am");
            DateTime dt_after_10_days = dt.AddDays(10);
            Console.WriteLine("After 10 days: " + dt_after_10_days);
            DateTime dt_before_10_days = dt.AddDays(-10);
            Console.WriteLine("Before 10 days: " + dt_before_10_days);

            DateTime dt_after_20_months_and_5_days = dt.AddMonths(20).AddDays(5);
            Console.WriteLine("After 20 months and 5 days: " + dt_after_20_months_and_5_days);
            Console.ReadKey();
        }
    }
}
```

DateTime

DateTime Methods

#	Method
12	DateTime AddDays(double value)
	It creates and returns a new instance of DateTime structure, after adding specified days (+ve or -ve) to the current date & time value.
13	DateTime AddMonths(double value)
	It creates and returns a new instance of DateTime structure, after adding specified months (+ve or -ve) to the current date & time value.
14	DateTime AddYears(double value)
	It creates and returns a new instance of DateTime structure, after adding specified years (+ve or -ve) to the current date & time value.
15	DateTime AddHours(double value)
	It creates and returns a new instance of DateTime structure, after adding specified hours (+ve or -ve) to the current date & time value.

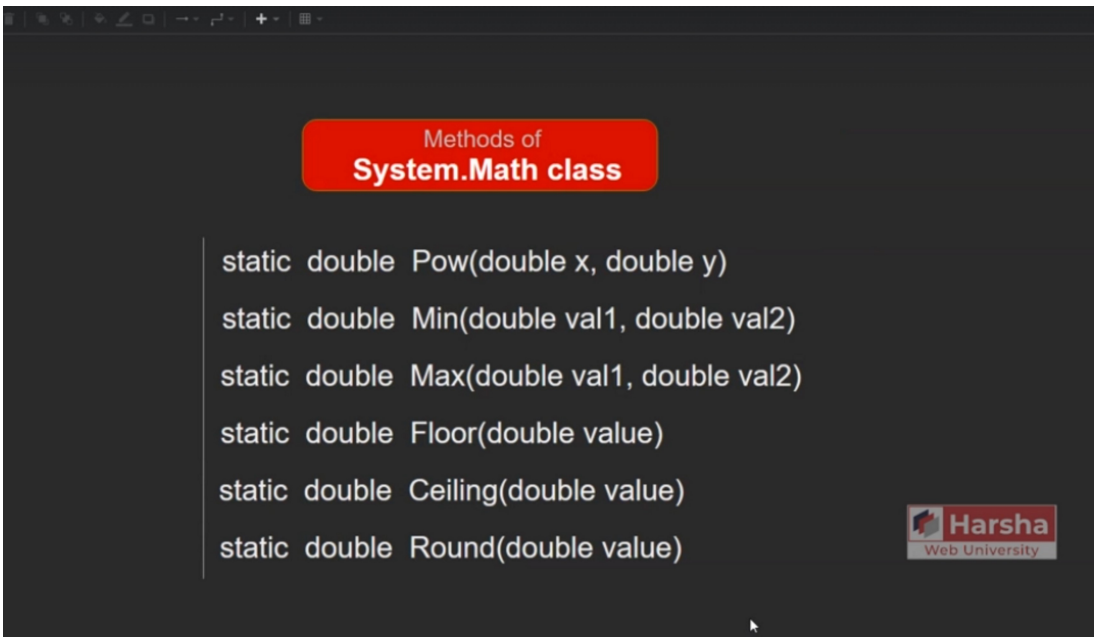
DateTime

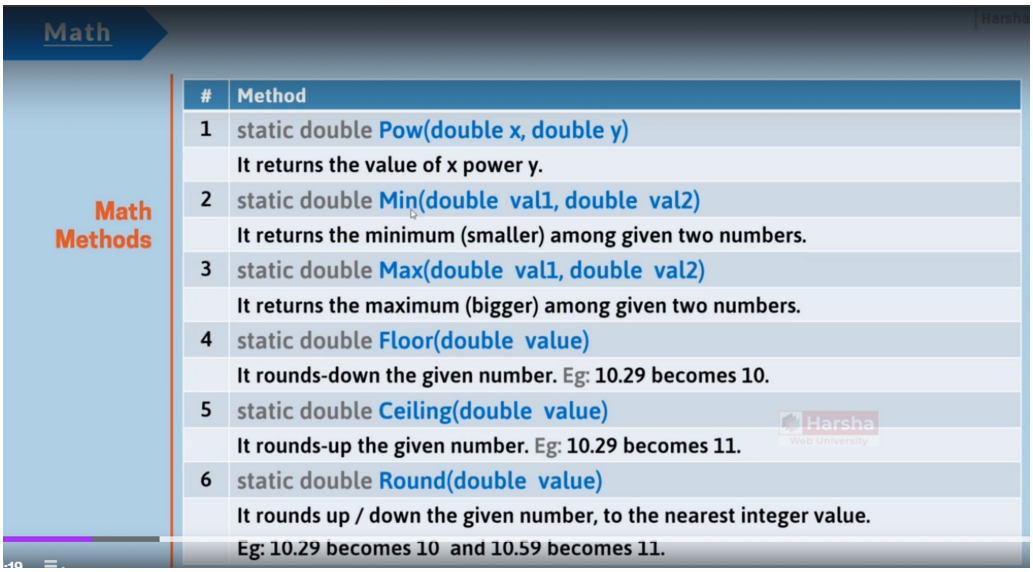
DateTime Methods

#	Method
16	DateTime AddMinutes(double value)
	It creates and returns a new instance of DateTime structure, after adding specified minutes (+ve or -ve) to the current date & time value.
17	DateTime AddSeconds(double value)
	It creates and returns a new instance of DateTime structure, after adding specified seconds (+ve or -ve) to the current date & time value.
18	DateTime AddMilliseconds(double value)
	It creates and returns a new instance of DateTime structure, after adding specified milliseconds to the current date & time value.

'Math' class

Part 1





```
MathExample.Program                                    Main()
double a = Math.Pow(10, 4); //10 power 4 (10 * 10 * 10 * 10)
double b = Math.Min(5.623, 10.82); //smaller among given numbers
double c = Math.Max(5.623, 10.82); //bigger among given numbers
double d = Math.Floor(20.83984); //returns the integer part
double e = Math.Ceiling(20.23984); //returns the next integer
double f = Math.Round(20.23984); //round down
double g = Math.Round(20.253984); //round up
Console.WriteLine(a); //Output: 10000
Console.WriteLine(b); //Output: 5.623
Console.WriteLine(c); //Output: 10.82
Console.WriteLine(d); //Output: 20
Console.WriteLine(e); //Output: 21
Console.WriteLine(f); //Output: 20
Console.WriteLine(g); //Output: 21
```

20.553984

It considers the first digit in the fraction part.

```
double g = Math.Round(20.53984); //round up
double h = Math.Round(20.53984, 2); //round up

Console.WriteLine(a); //Output: 10000
Console.WriteLine(b); //Output: 5.623
Console.WriteLine(c); //Output: 10.82
Console.WriteLine(d); //Output: 20
Console.WriteLine(e); //Output: 21
Console.WriteLine(f); //Output: 20
Console.WriteLine(g); //Output: 21
Console.WriteLine(h); //Output: 20.54
```

'Math' class

Part 2

Math

Math Methods

#	Method
7	static double Round(double value, int decimals) It rounds up / down the given number, to the specified fractional digits. Eg: Math.Round(value, 2) 10.272 becomes 10.27 and 10.275 becomes 10.38.
8	static int Sign(double value) It returns: -1: if the value is a negative value 0: if the value is equal to 0 1: if the value is a positive value
9	static double Abs(double value) It returns the positive value of given number.
10	static int DivRem(int val1, int val2, out int result) It returns the remainder value of val1/val2; and provides the quotient value as 'result' out parameter.


```
double h = Math.Round(20.53984, 2); //round up
double i = Math.Sign(-10); //-1 if negative value
double j = Math.Abs(-10); //returns positive number based on given negative number

Console.WriteLine(a); //Output: 10000
Console.WriteLine(b); //Output: 5.623
Console.WriteLine(c); //Output: 10.82
Console.WriteLine(d); //Output: 20
Console.WriteLine(e); //Output: 21
Console.WriteLine(f); //Output: 20
Console.WriteLine(g); //Output: 21
Console.WriteLine(h); //Output: 20.54
Console.WriteLine(i); //Output: -1
Console.WriteLine(j); //Output: 10

Console.ReadKey();
```

```
10000
5.623
10.82
20
21
20
21
20.54
-1
10
```

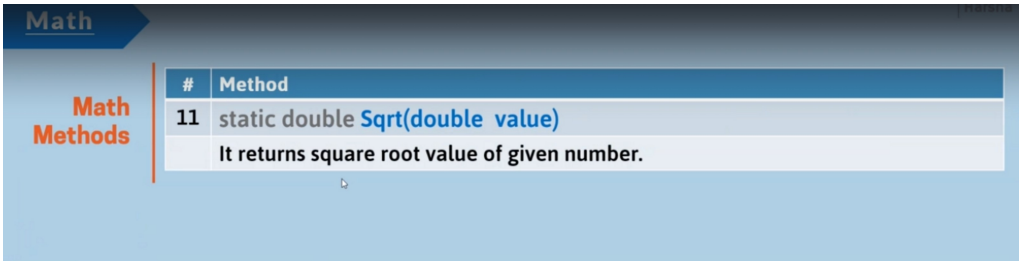
```
10  static int DivRem(int  val1, int  val2, out  int  result)
    It returns the quotient value of val1/val2; and provides the remainder value as
    'result' out parameter.
```

```
//10 / 3 = Quotient: 3; remainder: 1
int rem;
double quo = Math.DivRem(10, 3, out rem);
Console.WriteLine(a); //Output: 10000
Console.WriteLine(b); //Output: 5.623
Console.WriteLine(c); //Output: 10.82
Console.WriteLine(d); //Output: 20
Console.WriteLine(e); //Output: 21
Console.WriteLine(f); //Output: 20
Console.WriteLine(g); //Output: 21
Console.WriteLine(h); //Output: 20.54
Console.WriteLine(i); //Output: -1
Console.WriteLine(j); //Output: 10
Console.WriteLine("Quotient: " + quo + ", Remainder: " + rem);

Console.ReadKey();
```



```
32          Console.WriteLine("Quotient: " + quo + ", Remainder: " + rem);
33          Console.WriteLine(Math.Floor(10 / 3));
34          Console.WriteLine(10 % 3);
35          Console.ReadKey();
36      }
```

```
double j = Math.Abs(-10); //returns positive number based on given negative number
//10 / 3 = Quotient: 3; remainder: 1
int rem;
double quo = Math.DivRem(10, 3, out rem);
Console.WriteLine(a); //Ou
Console.WriteLine(b); //Ou
Console.WriteLine(c); //Ou
Console.WriteLine(d); //Output: 20
Console.WriteLine(e); //Output: 21
Console.WriteLine(f); //Output: 20
Console.WriteLine(g); //Output: 21
Console.WriteLine(h); //Output: 20.54
Console.WriteLine(i); //Output: -1
Console.WriteLine(j); //Output: 10
Console.WriteLine("Quotient: " + quo + ", Remainder: " + rem);
Console.WriteLine(Math.Floor(Convert.ToDouble(10) / 3));
Console.WriteLine(10 % 3);
Console.ReadKey();
```




```
double k = Math.Sqrt(25);

Console.WriteLine(a); //Output: 10000
Console.WriteLine(b); //Output: 5.623
Console.WriteLine(c); //Output: 10.82
Console.WriteLine(d); //Output: 20
Console.WriteLine(e); //Output: 21
Console.WriteLine(f); //Output: 20
Console.WriteLine(g); //Output: 21
Console.WriteLine(h); //Output: 20.54
Console.WriteLine(i); //Output: -1
Console.WriteLine(j); //Output: 10
Console.WriteLine("Quotient: " + quo + ", Remainder: " + rem); //3 an
Console.WriteLine(Math.Floor(Convert.ToDouble(10) / 3)); //3
Console.WriteLine(10 % 3); //1
Console.WriteLine(k); //25
Console.ReadKey();
```

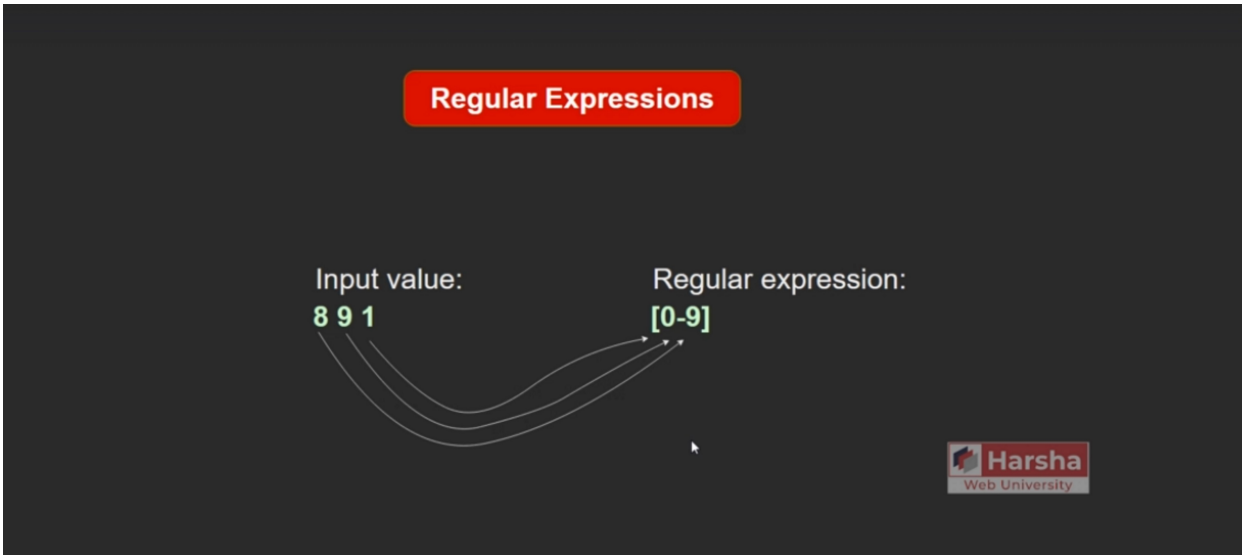
Harsha Web University

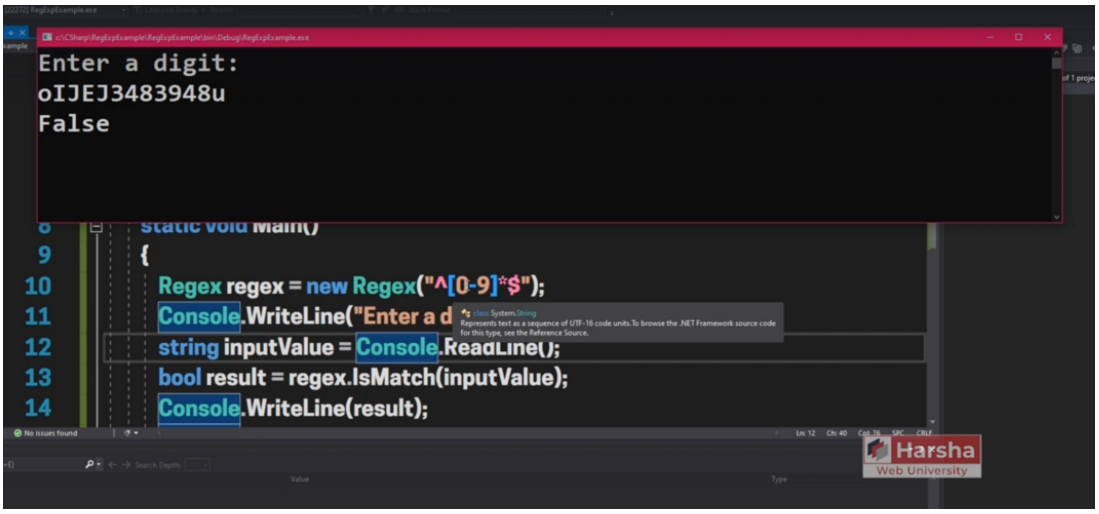
this variable

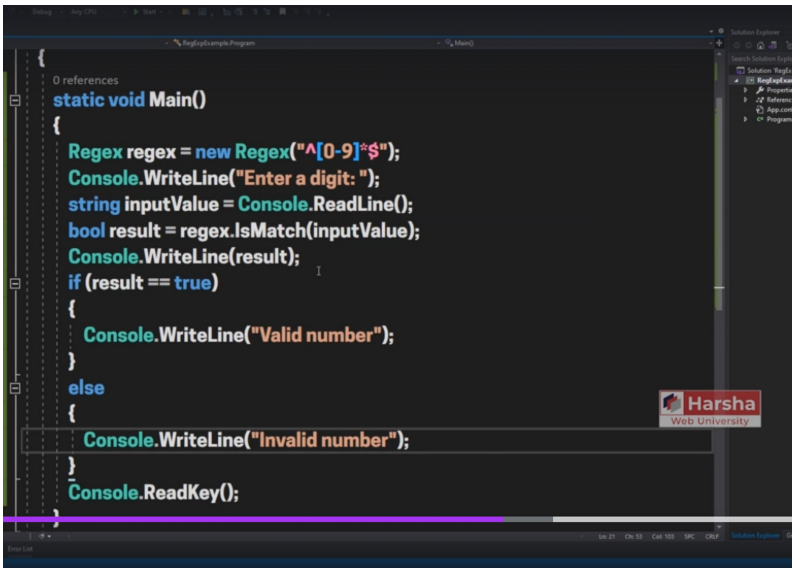
expects double but we supplied int. because of 'implicit casting'.

Regular Expressions

Regular expressions or predefined formats based on which you can validate your string values for example you're expecting that the user must enter only digits but the user might enter any kind of values at runtime by using that console.Readline() even in the windows forms applications or in web applications also in the text boxes the user might enter any kind of values so there is no guarantee that the user enters only digits but you are expecting only digits suppose in that case there should be a mechanism that can validate the user inputs whether it meets the specified set of characters that you want or not.


Regular Expressions
Input value:
8 9 1
Regular expression:
[0-9]
Harsha
Web University


Enter a digit:
oIJEJ3483948u
False
static void Main()
{
Regex regex = new Regex("^[0-9]*$");
Console.WriteLine("Enter a d
string inputValue = Console.ReadLine();
bool result = regex.IsMatch(inputValue);
Console.WriteLine(result);
Harsha
Web University


0 references
static void Main()
{
Regex regex = new Regex("^[0-9]*$");
Console.WriteLine("Enter a digit: ");
string inputValue = Console.ReadLine();
bool result = regex.IsMatch(inputValue);
Console.WriteLine(result);
if (result == true)
{
Console.WriteLine("Valid number");
}
else
{
Console.WriteLine("Invalid number");
}
Console.ReadKey();
}
Harsha
Web University

```
{
    0 references
    static void Main()
    {
        Regex regex = new Regex("^[A-Za-z]*$");
        Console.WriteLine("Enter a digit: ");
        string inputValue = Console.ReadLine();
        bool result = regex.IsMatch(inputValue);
        Console.WriteLine(result);
        if (result == true)
        {
            Console.WriteLine("Valid number");
        }
        else
        {
            Console.WriteLine("Invalid number");
        }
        Console.ReadKey();
    }
```

Harsha Web University

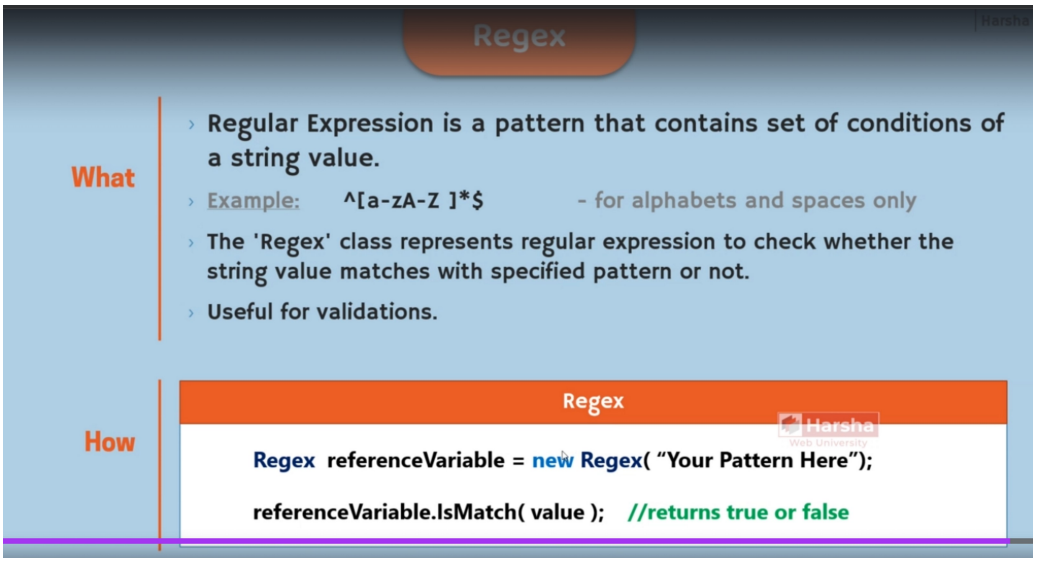
```
{
    Regex regex = new Regex("^[A-Za-z ]*$");
```

space

Regular Expressions

Symbol	Meaning
[0-9]	1, 2, 3, 4, 5, 6, 7, 8, 9, 0
[a-z]	a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
[A-Z]	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, , W, X, Y, Z
[1,2,3]	1, 2, 3
\d	1, 2, 3, 4, 5, 6, 7, 8, 9, 0
\w	word
\s	White space

Harsha Web University


Regex
What
Regular Expression is a pattern that contains set of conditions of a string value.
Example: ^[a-zA-Z]*$ - for alphabets and spaces only
The 'Regex' class represents regular expression to check whether the string value matches with specified pattern or not.
Useful for validations.
How
Regex
Regex referenceVariable = new Regex("Your Pattern Here");
referenceVariable.IsMatch(value); //returns true or false