

# Introduction to Number Systems

Number Systems

Eg:

Decimal	100
Binary	0b110 0100
Octal	0o144
Hexadecimal	0x64

 Harsha  
Web University

Number Systems

What In mathematics, a 'Number system' provides a set of digits for counting.

Base 10 <b>Decimal</b> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9	Base 8 <b>Octal</b> 0, 1, 2, 3, 4, 5, 6, 7
Base 2 <b>Binary</b> 0, 1	Base 16 <b>Hexadecimal</b> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

## Binary Number System



## Number Systems

Eg:

Decimal

Binary

Octal

Hexadecimal

100

0b110 0100

0o144

0x64



### decimal into binary

2	13	
2	6	1
2	3	0
(1)		1

ans: 1101

## Octal Number System



## Number Systems

Eg:

**Decimal**

100

**Binary**

0b110 0100

**Octal**

0o144.



**Hexadecimal**

0x64

The screenshot shows two conversion examples in Microsoft Paint:

**decimal into binary**

2	13	
2	6	1
2	3	0
(1)		1

**decimal into octal**

8	289	
8	36	1
(4)		4

**ans: 1101**

**ans: 441**

There is not possibility in octal literals in C#

## Hexadecimal Number System



## Number Systems

stem

What In mathematics, a 'Number system' provides a set of digits for counting.

Base 10

**Decimal**

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Base 8

**Octal**

0, 1, 2, 3, 4, 5, 6, 7

Base 2

**Binary**

0, 1

Base 16

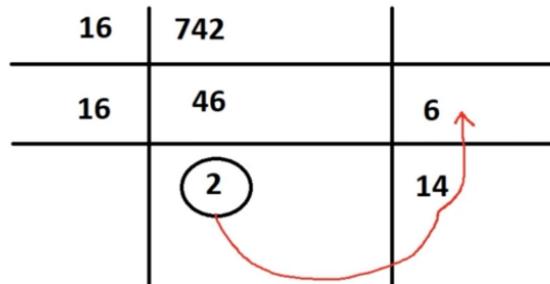
**Hexadecimal**

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

For representation of larger number, hashing and encrypted data, also for representation of color code, in these case you require to use hexadecimal number system in the real life.



### decimal to hexadecimal



ans: 2e6

## Introduction to Character Encoding



# Introduction to Encoding

In general encoding means writing the characters in the specific format based on certain rules in other words the process of converting the set of characters into another different format is called as encoding and converting original format is called as decoding.

but here we are trying to understand the character encoding in specific that means converting the characters into numerical format so here we have to assume a number for each character in the older computer character encoding format by default.

ASCII stands for American standard code for information interchange according to which by default it supports 128 characters according to ascii each character is represented as corresponding number so totally for 128 characters

we have 128 numbers that is 0 to 127 and each character occupies exactly 7 bits but in general it is considered as 1 byte.

ASCII Table											
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	�
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	.	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	�	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	�	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	:	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

These are the character that are represented by ASCII. but the problem with ASCII is that, it only supports english language. It does not support other natural language character such as arabic, bengali etc. In UniCode characters are introduced.

ing those characters back to the

er systems ascii was used as

ters that is 0 to 127 in fact

order to solve this problem,

**What**

- It is a concept that tells you represent characters into numbers (or any other specific format).

American Standard Code for Information Interchange

**ASCII**

- Each character occupies 7 bits (generally considered as 1 byte)
- 128 characters (0 to 127)
- Includes all keyboard characters with alphabets, digits, special characters etc.

Universal Code

**UTF-16 / Unicode**

- Each character occupies 2 or 4 bytes (generally considered as 2 bytes)
- About 144697 characters (approx)
- Includes all natural language characters along with ASCII.

**List of Uni-Code Characters**

[https://en.wikipedia.org/wiki/List\\_of\\_Unicode\\_characters](https://en.wikipedia.org/wiki/List_of_Unicode_characters)

**Bengali and Assamese** [edit]

Main article: [Bengali \(Unicode block\)](#)

Bengali <sup>[1][2]</sup>															
Official Unicode Consortium code chart <sup>[3]</sup> (PDF)															
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+098x	ঠ	ঁ	ং	ঃ		অ	আ	ই	ঈ	উ	ঊ	ঝ	ঙ		়
U+099x	ঞ			ও	ঔ	ক	খ	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঢ
U+09Ax	ঁ	ড	ঁ	ণ	ঁ	ত	থ	দ	ধ	ন		প	ফ	ব	ভ
U+09Bx	ৰ		ল				শ	ষ	স	হ			ঃ	হ	া
U+09Cx	ঁ	৭	৮	৯	৩		ে	ৈ			ো	ৌ	্	ঁ	
U+09Dx							ঁ						ড	ঁ	
U+09Ex	ঝ	ঁ	ঁ	ঁ	ঁ		০	১	২	৩	৪	৫	৬	৭	৮
U+09Fx	ঁ	ৰ	ু	ট	ু	ু	ু	ু	ু	ু	ু	ু	ু	ু	ু

**Notes**

1<sup>a</sup> As of Unicode version 15.1

2<sup>b</sup> Grey areas indicate non-assigned code points



ASCII Character Encoding

# ASCII Encoding

**How do you convert Character into ASCII Code and Vice-versa in the C# Program?**



## ASCII Table

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	'
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	:	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	-	127	7F	[DEL]

by default, ASCII code can be represented as byte data type. Because byte data type supports upto 256 character.

and ASCII code are available upto 127. so byte data type is more than enough for representation of ASCII code.



Decimal	Hex	Char	Decimal
0	20	[NULL]	32
1	21	[START OF HEADING]	33
2	22	[START OF TEXT]	34
3	23	[END OF TEXT]	35
4	24	[END OF TRANSMISSION]	36
5	25	[ENQUIRY]	37
6	26	[ACKNOWLEDGE]	38
7	27	[BELL]	39
8	28	[BACKSPACE]	40
9	29	[HORIZONTAL TAB]	41
A	2A	[LINE FEED]	42
B	2B	[VERTICAL TAB]	43
C	2C	[FORM FEED]	44
D	2D	[CARRIAGE RETURN]	45
E	2E	[SHIFT OUT]	46
F	2F	[SHIFT IN]	47
10	30	[DATA LINK ESCAPE]	48
11	31	[DEVICE CONTROL 1]	49
12	32	[DEVICE CONTROL 2]	50
13	33	[DEVICE CONTROL 3]	51
14	34	[DEVICE CONTROL 4]	52
15	35	[NEGATIVE ACKNOWLEDGE]	53
16	36	[SYNCHRONOUS IDLE]	54
17	37	[END OF TRANS. BLOCK]	55
18	38	[CANCEL]	56
19	39	[END OF MEDIUM]	57
1A	3A	[SUBSTITUTE]	58
1B	3B	[ESCAPE]	59
1C	3C	[FILE SEPARATOR]	60
1D	3D	[GROUP SEPARATOR]	61
1E	3E	[RECORD SEPARATOR]	62
1F	3F	[UNIT SEPARATOR]	63

These are the characters which will not print in the console. but you must be able to see all these following character

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
32	20	[SPACE]	64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	*	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	\$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	:	91	5B	{	123	7B	{
60	3C	<	92	5C	\	124	7C	
61	3D	=	93	5D	}	125	7D	}
62	3E	>	94	5E	"	126	7E	~
63	3F	?	95	5F	-	127	7F	[DEL]

## UniCode Character Encoding



# Unicode Encoding

C# by default supports unicode character.

## Introduction to System.IO namespace

### Introduction to System.IO

0 namespace

'System.IO' namespace Harsha

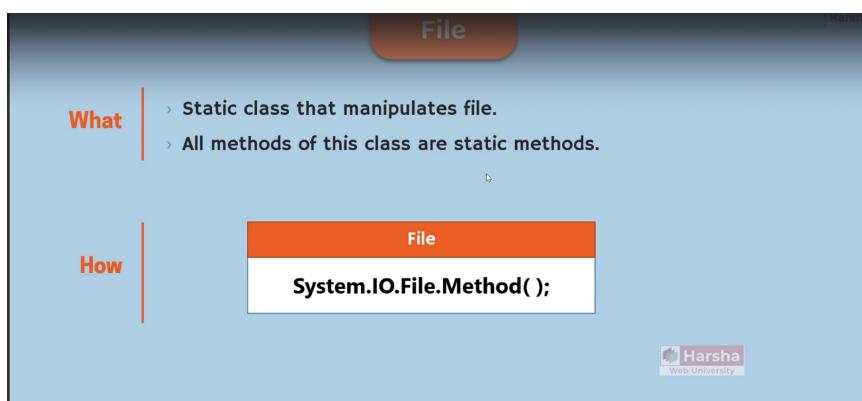
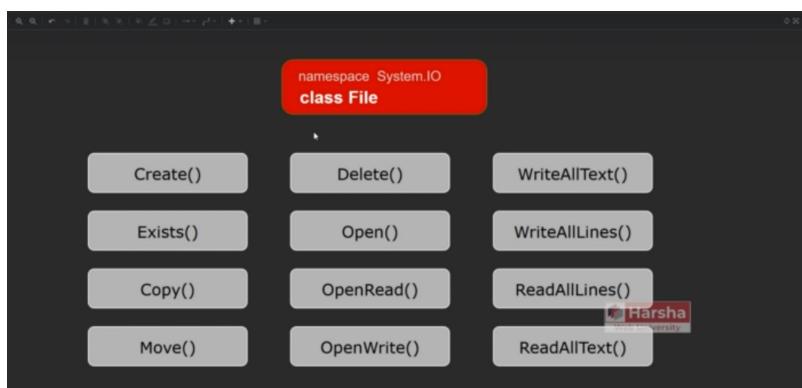
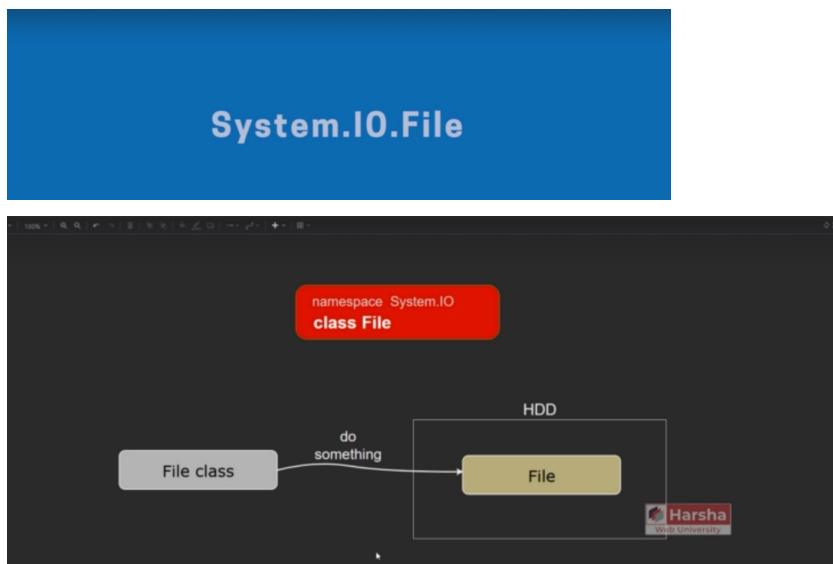
What This namespace contains classes to perform File I/O operations.

System.IO class <b>File</b>	System.IO class <b>DriveInfo</b>	System.IO class <b>FileStream</b>
System.IO class <b>Directory</b>	System.IO class <b>FileNotFoundException</b>	
System.IO class <b>FileInfo</b>	System.IO class <b>StreamWriter</b>	System.IO class <b>BinaryWriter</b>
System.IO class <b>DirectoryInfo</b>	System.IO class <b>StreamReader</b>	System.IO class <b>BinaryReader</b>

In dot net we have a predefined namespace called system.io  
it contains several predefined classes as you can see which are used to perform various manipulations on files in the hard disk



## 'File' class

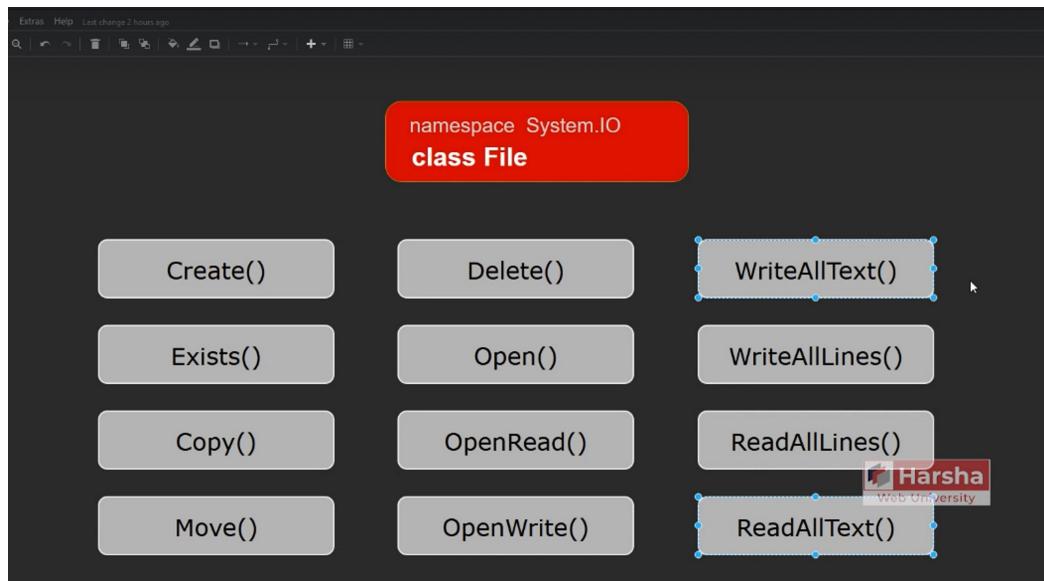




File	
Methods	# Method
1	<code>static FileStream Create( string path )</code> Create / overwrites a file at the specified path.
2	<code>static bool Exists( string path )</code> Determines whether the file exists in the disk or not.
3	<code>static void Copy( string sourceFile, string destFile)</code> Copies the source file to the destination location.
4	<code>static void Move( string sourceFile, string destFile)</code> Moves the source file to the destination location.
5	<code>static void Delete ( string path )</code> Deletes the specified file permanently.

## 'File' class - Read and Write

### File Read & Write



For single string, `WriteAllText()` and `ReadAllText()` is best. But for a list of string, `WriteAllLines()` and `ReadAllLines()` is best.

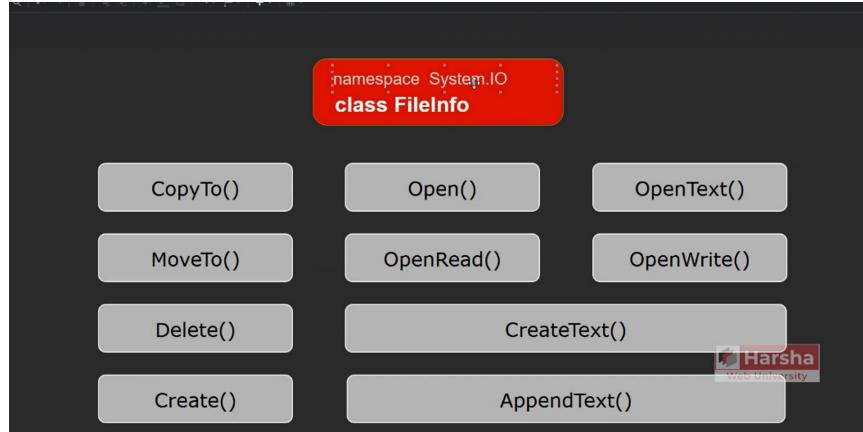


File	
Methods	# Method
6	<b>static void WriteAllLines( string path, IEnumerable&lt;string&gt; contents)</b> Creates / overwrites a file; writes specified lines of content to the file; and close the file.
7	<b>static string[] ReadAllLines( string path)</b> Reads file content as text and returns all lines.
8	<b>static string ReadAllText( string path)</b> Reads file content as text and returns the same.
9	<b>static void WriteAllText( string path, string contents)</b> Creates / overwrites a file; writes specified content to the file; and close the file.

(contd...)

## FileInfo Class

### System.IO.FileInfo



FileInfo	
Methods	# Method
1	<b>FileInfo CopyTo(string destFilePath, bool overwrite)</b> Copies the file into the new destination path. It returns an object of <code>FileInfo</code> class, that represents the newly created file.
2	<b>void MoveTo(string destFilePath)</b> Moves the file into the new destination path (should be in the same drive).
3	<b>void Delete( )</b> Deletes the file permanently.

(contd...)



## FileInfo

**What** ➤ Represents a file and provides methods to manipulate the file.

**How**

### FileInfo

```
 FileInfo referenceVariable = new FileInfo( "Your File Path Here");
```

**Constructors**

Constructor	Description
<code>FileInfo(string filePath)</code>	It initializes the file path which needs to be manipulated.

**Methods**

## File

#	Method	Description
1	<code>static FileStream Create( string path )</code>	Create / overwrites a file at the specified path.
2	<code>static bool Exists( string path )</code>	Determines whether the file exists in the disk or not.
3	<code>static void Copy( string sourceFile, string destFile)</code>	Copies the source file to the destination location.
4	<code>static void Move( string sourceFile, string destFile)</code>	Moves the source file to the destination location.
5	<code>static void Delete ( string path )</code>	Deletes the specified file permanently.

**Methods**

## FileInfo

#	Method	Description
1	<code>FileInfo CopyTo(string destFilePath, bool overwrite)</code>	Copies the file into the new destination path. It returns an object of FileInfo class, that represents the newly created file.
2	<code>void MoveTo(string destFilePath)</code>	Moves the file into the new destination path (should be in the same drive).
3	<code>void Delete( )</code>	Deletes the file permanently.



(contd...)



**Methods**

#	Method
1	<code>FileInfo CopyTo(string destFilePath, bool overwrite)</code> Copies the file into the new destination path. It returns an object of FileInfo class, that represents the newly created file.
2	<code>void MoveTo(string destFilePath)</code> Moves the file into the new destination path (should be in the same drive).
3	<code>void Delete()</code> Deletes the file permanently.



(contd...)

File class is good for **single operation**; here you do not have to create an instance time you have to supply the **FilePath**,

if you have to perform multiple operation, then use FileInfo class...

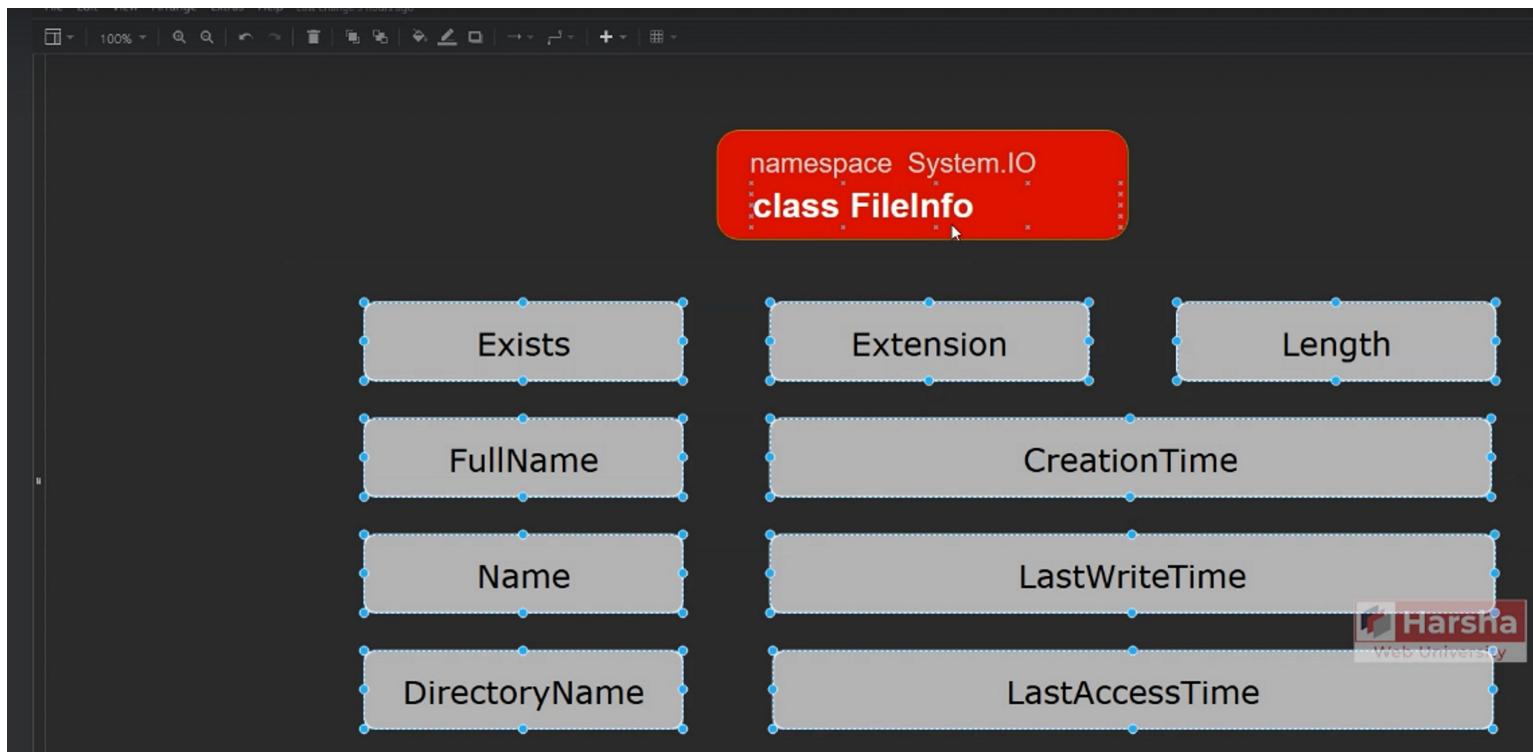
## Properties of 'FileInfo' class

The file info class can provide additional information or details about the file for example you have a hard disk and it has a file you would like to read it as full path file extension length of the file and also the date on time of file creation etc you will be able to read all the file details in the form of these properties of the file info class.

e but every

s details such

ies



**FileInfo**

**Properties**

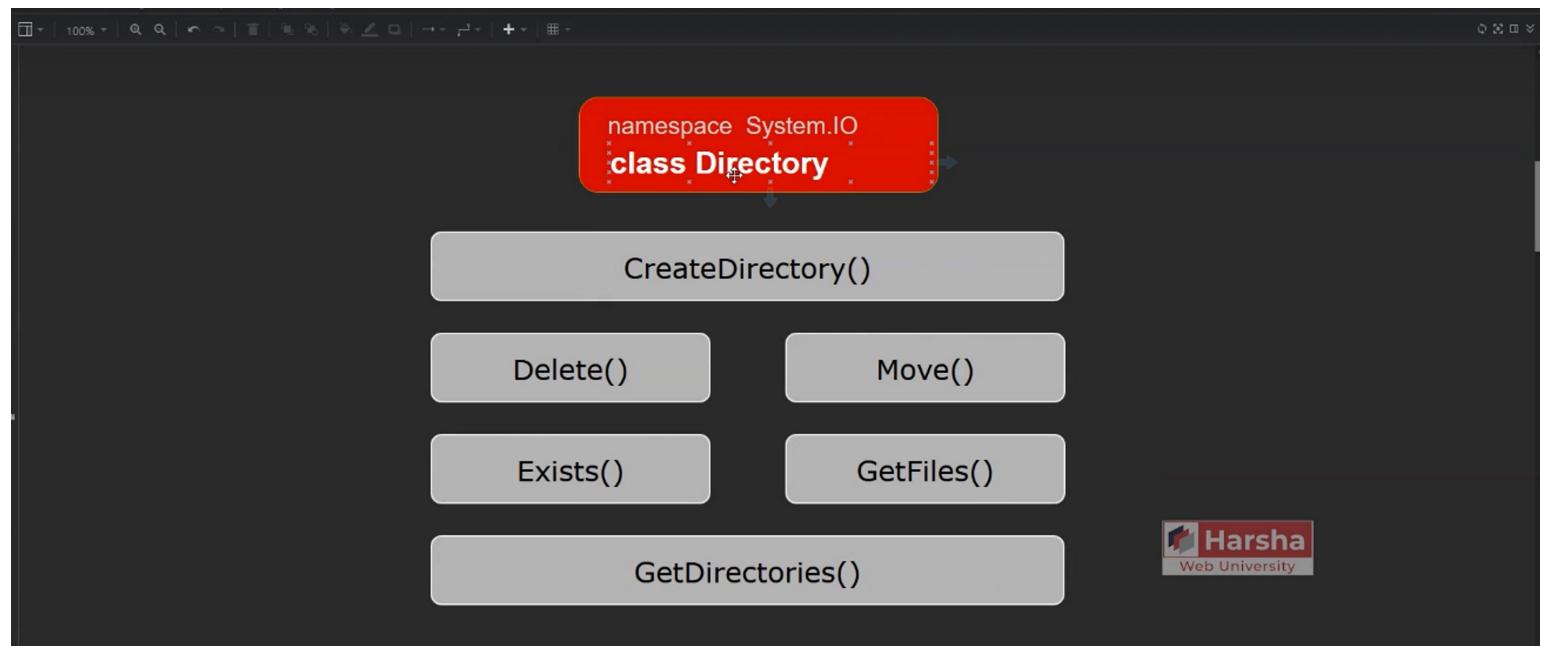
Property	Description
<code>bool Exists</code>	Determines whether the file exists in the disk or not.
<code>string FullName</code>	Represents full path (including file name and extension) of the file.
<code>string Name</code>	Represents only name of the file (without path).
<code>string DirectoryName</code>	Represents only path of the file (without file name).
<code>string Extension</code>	Represents only file extension (without file name).
<code>DateTime CreationTime</code>	Represents date and time of file creation.
<code>DateTime LastWriteTime</code>	Represents date and time of last modification of the file.
<code>DateTime LastAccessTime</code>	Represents date and time of last access of the file.
<code>long Length</code>	Represents file size (in the form of no. of bytes).



## 'Directory' class - Part 1

# System.IO.Directory

## Part 1







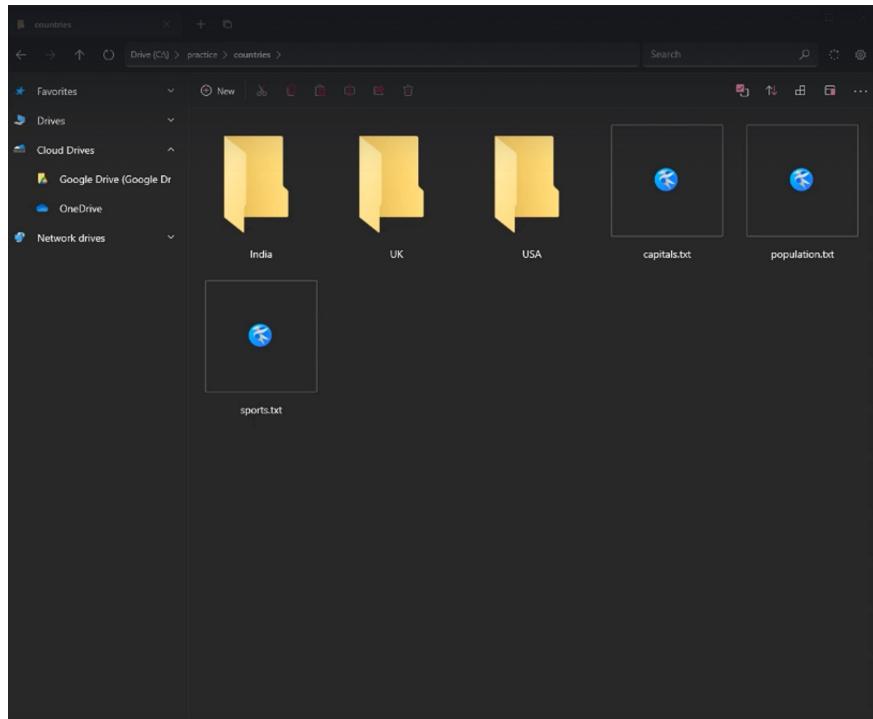
```
new FileStream()  
File.Create(capitalsFilePath).Close();  
File.Create(sportsFilePath).Close();  
File.Create(populationFilePath).Close();  
Console.ReadKey();  
}
```



## 'Directory' class - Part 2

# System.IO.Directory

## Part 2

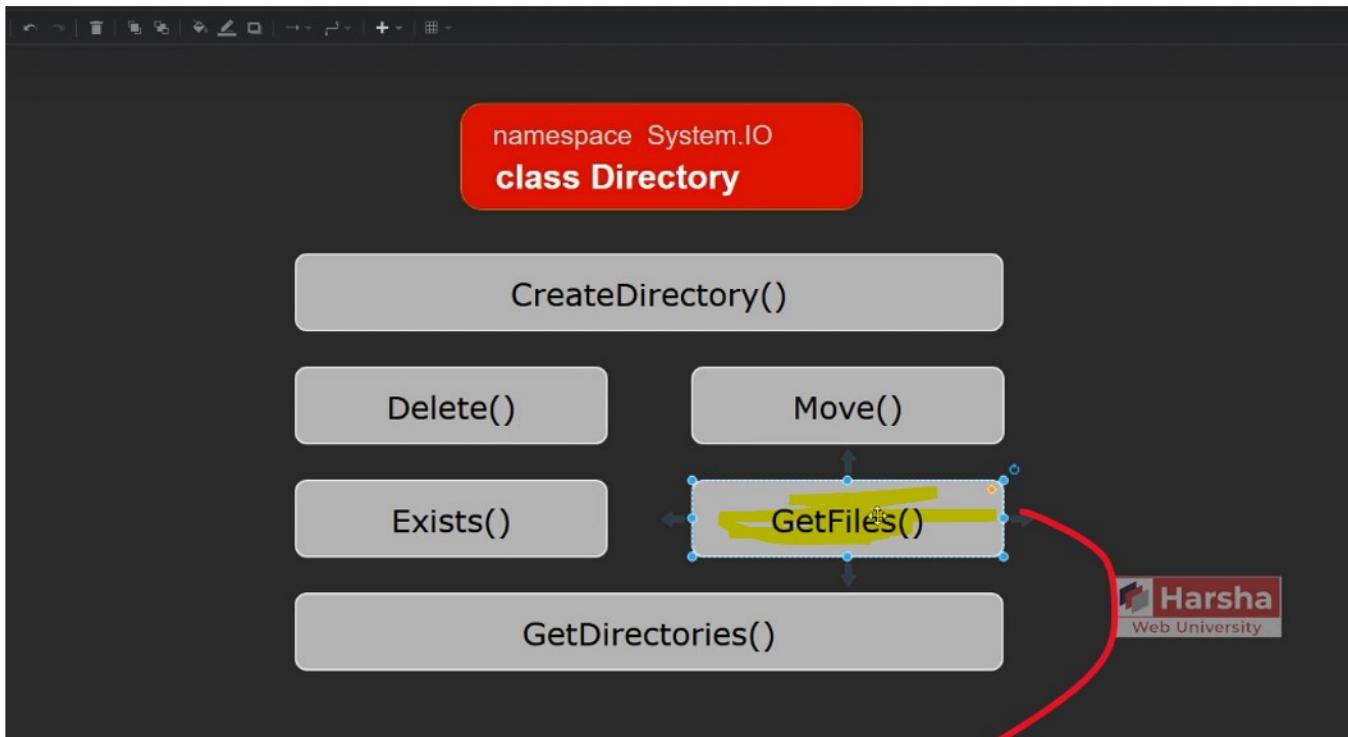


**Task:**

1. Create a folder "countries" in "c:\practice".
2. Create three folders "India", "UK", "USA" in "countries" folder.
3. Create three files "capitals.txt", "sports.txt" and "population.txt" in "countries" folder.
4. Move/rename "countries" as "world".
5. List the files of "world" folder.
6. List of folders of "world" folder.
7. Delete the "world" folder, including its files and sub folders.

Marsha  
Web University









In Order to get files from a specific folder, we have `GetFiles()` folder

But the problem with `Directory` class is that we have to supply the `FolderPath` every time  
But if we want to do multiple operation with the same file, use '`DirectoryInfo`' class

**Directory**

**What**

- > Static class that manipulates directory (folder).
- > All methods of this class are static methods.

**How**

**Directory**

**System.IO.Directory.Method( );**

Harsha Web University

This slide compares the `Directory` and `DirectoryInfo` classes. It uses a diagram with two columns: 'What' and 'How'. The 'What' column describes the `Directory` class as a static class that manipulates directories and contains static methods. The 'How' column shows the code `System.IO.Directory.Method( );` enclosed in a box. The `Directory` class is also highlighted in orange at the top of the slide.

der

ytime.  
s.

## Directory

### Methods

#	Method
1	<code>static DirectoryInfo CreateDirectory( string path )</code> Create a directory at the specified path.
2	<code>static void Delete ( string path, bool recursive )</code> <u>true</u> : Deletes the directory including sub directories and all files. <u>false</u> : Deletes the directory only if it is empty.
3	<code>static bool Exists( string path )</code> Determines whether the directory exists in the disk or not.
4	<code>static string[ ] GetDirectories( string path )</code> Returns string[ ] that contains paths of sub directories of specified directory.

## Directory

### Methods

#	Method
5	<code>static string[ ] GetDirectories( string path, string searchPattern )</code> Returns string[ ] that contains paths of sub directories that matches with specified search pattern.
6	<code>static string[ ] GetFiles( string path )</code> Returns string[ ] that contains paths of files of specified directory.
7	<code>static string[ ] GetFiles( string path, string searchPattern )</code> Returns string[ ] that contains paths of files that matches with specified search pattern.
8	<code>static void Move(string sourceDir, string destDir)</code> Moves the source directory to the specified destination location in the same drive.

ctory.

contd...)

```
38  
39 //GetFiles  
40 string[] files = Directory.GetFiles(worldFolderPath, "c*.txt");  
41 Console.WriteLine("\nFiles:");  
42 foreach (string file in files)  
43 {  
44     Console.WriteLine(file);
```

### ' DirectoryInfo' Class

# System.IO.DirectoryInfo



namespace System.IO

**class DirectoryInfo**

CreateSubDirectory()

Create()

MoveTo()

Delete()

GetFiles()

GetDirectories()



## Let's do the same task using FileInfo Class

Task:

1. Create a folder "countries" in "c:\practice".
2. Create three folders "India", "UK", "USA" in "countries" folder.
3. Create three files "capitals.txt", "sports.txt" and "population.txt" in "countries" folder.
4. Move/rename "countries" as "world".
5. List the files of "world" folder.
6. List of folders of "world" folder.
7. Delete the "world" folder, including its files and sub folders.



# DirectoryInfo

Harsh

## What

- > Class that represents a directory (folder) on the disk and performs manipulations on directories.

## How

### DirectoryInfo

```
 DirectoryInfo referenceVariable = new DirectoryInfo("Directory Path Here");
```

## Constructors

Constructor	Description
<code> DirectoryInfo(string path)</code>	It initializes the directory path which needs to be manipulated.



## Methods

#	Method
1	<code> void Create( )</code> Create the directory at the current path.
2	<code> DirectoryInfo CreateSubDirectory ( string path )</code> Creates a sub directory at the current path with specified name.
3	<code> void Delete( bool recursive )</code> <code>true</code> : Deletes the directory including sub directories. <code>false</code> : Deletes the directory only if it is empty.
4	<code> DirectoryInfo[ ] GetDirectories()</code> Returns <code> DirectoryInfo[ ]</code> that represents sub directories of current directory.



(contd...)



**Methods**

#	Method
5	<code> DirectoryInfo[ ] GetDirectories ( string searchPattern )</code> Returns DirectoryInfo[ ] that represents sub directories that matches with specified search pattern.
6	<code> FileInfo[ ] GetFiles( )</code> Returns FileInfo[ ] that represents files of current directory.
7	<code> FileInfo[ ] GetFiles(string searchPattern)</code> Returns FileInfo[ ] that represents files that matches with specified search pattern.
8	<code> void MoveTo(string destDirName)</code> Moves the current directory to the specified location in the same drive.

**' DirectoryInfo' class - Properties**

Properties of  
**System.IO.DirectoryInfo**



namespace System.IO  
class DirectoryInfo

Exists

Root

FullName

CreationTime

Name

LastWriteTime

Parent

LastAccessTime



When you create an object of this **directoryinfo** class it will automatically read some information regarding that particular folder for example there is a folder called sample in my system if I create an object of directory info class and specified the path of the same folder then the object will automatically read the details of that sample folder and those details are available in the form of properties.

## Properties

### DirectoryInfo

Properties	Property	Description
	bool <b>Exists</b>	Determines whether the directory exists in the disk or not.
	string <b>FullName</b>	Represents full path of the directory
	string <b>Name</b>	Represents only name of the directory (without path).
	DirectoryInfo <b>Parent</b>	Represents parent directory of the current directory.
	DirectoryInfo <b>Root</b>	Represents root (drive) of the directory.
	DateTime <b>CreationTime</b>	Represents date and time of directory creation.
	DateTime <b>LastWriteTime</b>	Represents date and time of last modification of the directory.
	DateTime <b>LastAccessTime</b>	Represents date and time of last access of the directory.

me additional details or  
mple in c drive and you  
older then it automatically  
f these properties.

## **'DriveInfo' class**

# **System.IO.DriveInfo**

In order to manipulate the file, we have FileInfo class,



```
namespace System.IO  
class FileInfo
```

CopyTo()

Open()

OpenText()

MoveTo()

OpenRead()

OpenWrite()

Delete()

CreateText()

Create()

AppendText()



In order to directory, we have DirectoryInfo class.

```
namespace System.IO  
class DirectoryInfo
```

CreateSubDirectory()

Create()

MoveTo()

Delete()

GetFiles()

GetDirectories()



and in order to manipulate the Drive I mean get details about a particular drive, w  
'DriveInfo' class.

we have

```
namespace System.IO  
class DriveInfo
```

Name

RootDirectory

DriveType

TotalSize

VolumeLabel

AvailableFreeSpace



## DriveInfo

### What

- › Class that represents a drive and performs manipulations on drives.
- › You can read both fixed / removable drives.

### How

#### DriveInfo

```
DriveInfo referenceVariable = new DriveInfo( "Your Drive Name Here");
```

### Constructors

Constructor	Description
DriveInfo(string path)	It initializes the drive name which needs to be manipulated.



## DriveInfo

### Properties

Property	Description
string <b>Name</b>	Represents name of the drive.
string <b>DriveType</b>	Represents type of drive either 'Fixed' or 'Removable'
string <b>VolumeLabel</b>	Represents label of the drive (set by user).
DirectoryInfo <b>RootDirectory</b>	Represents root directory of the drive.
long <b>TotalSize</b>	Represents total size (bytes) of the drive.
long <b>AvailableFreeSpace</b>	Represents total free space (bytes) of the drive.

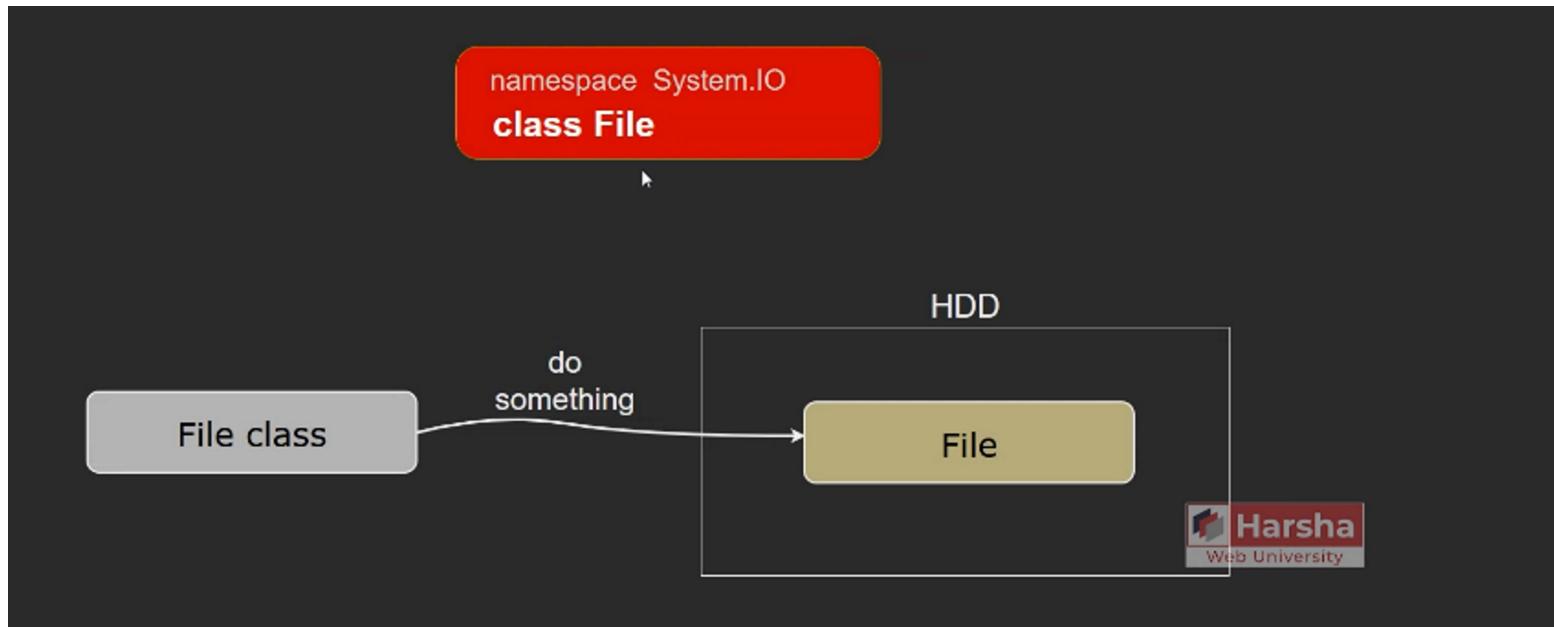
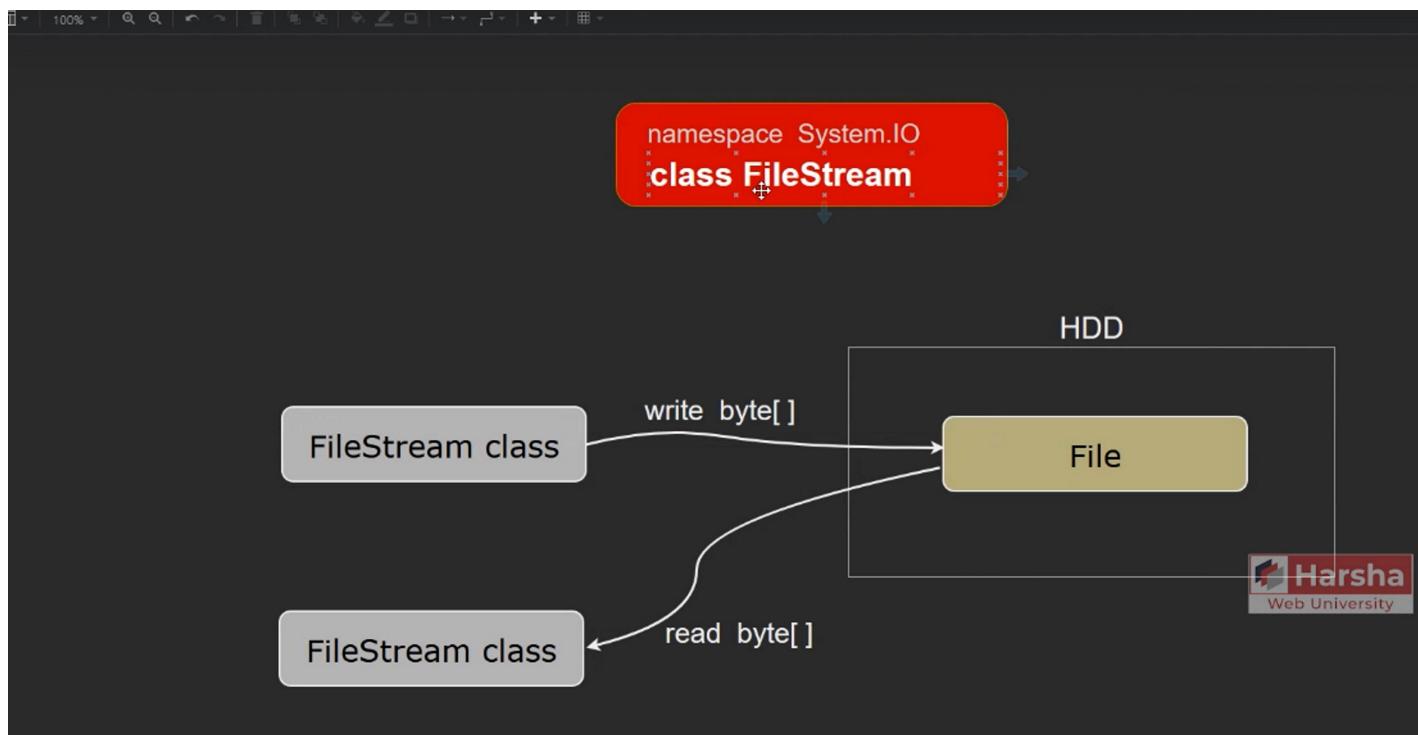
## FileStream Class - Part 1

# System.IO.FileStream

Part 1

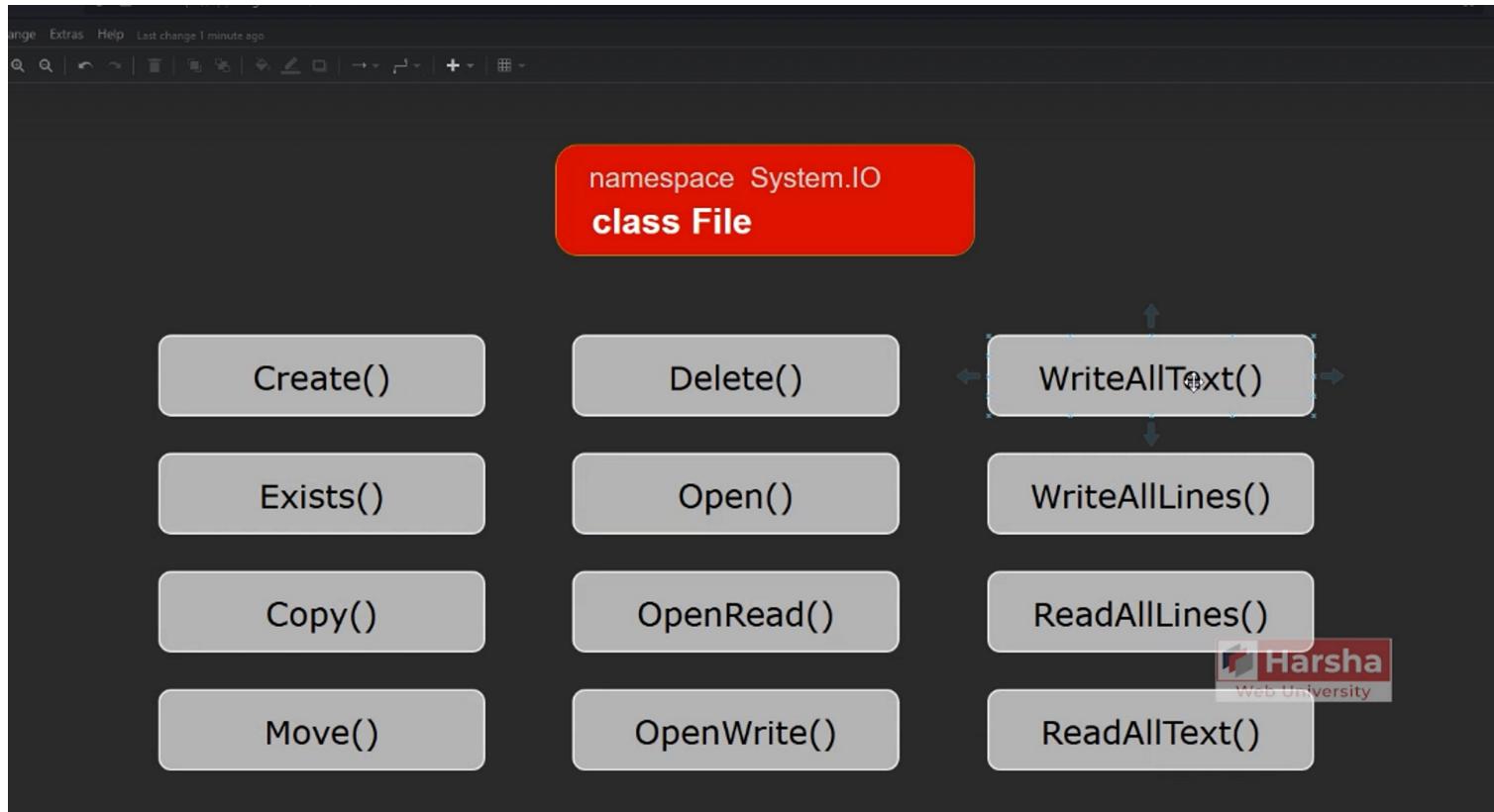
The file stream is such a class using which you can write content into the file or read from the file **but wait**

read existing content



Already we are able to perform file IO operations that is writing and reading operation by using the predefined class called `File`. the `File` class has methods like `readalltext`, `writealltext` etc in order to perform writing and reading operations so what is the `Stream` class.

rations  
xt  
e necessity of file



**The file class** is helpful when you want to perform a single operation with the file that means you have a string value with you at a single shot you would like to write some content into that particular file if you want to perform multiple operations to the file then you can quickly use file dot write alt text.

**FileStream:** you would like to write some text into the file and then after that execute some other code again you want to read some data from the file again execute some other code like this in the middle of the big program at different stages you would like to interact with the file.

file that's it you don't want to

to write some more information  
keep writing some information

**File**

**Methods**

#	Method
1	<b>static FileStream Create( string path )</b> Create / overwrites a file at the specified path.
2	<b>static bool Exists( string path )</b> Determines whether the file exists in the disk or not.
3	<b>static void Copy( string sourceFile, string destFile)</b> Copies the source file to the destination location.
4	<b>static void Move( string sourceFile, string destFile)</b> Moves the source file to the destination location.
5	<b>static void Delete ( string path )</b> Deletes the specified file permanently.

Harsha  
Web University

**File**

**Methods**

#	Method
I0	<b>static FileStream Open( string path, FileMode mode, FileAccess access )</b> Opens the file in specified file mode with specified file access permission and returns a new object of FileStream type.
II	<b>static FileStream OpenRead( string path)</b> Opens the file in 'Open' mode and returns a new object of FileStream type.
I2	<b>static FileStream OpenWrite( string path)</b> Opens the file in 'OpenOrCreate' mode and returns an object of FileStream type.

Harsha  
Web University

**' FileMode' in FileStream**

#	Mode	Description
1	<b>CreateNew</b>	It specifies that the o/s should create a new file. If the file already exists an IOException will be thrown.
2	<b>Create</b>	It specifies that the o/s should create a new file. If the file already exists, it will be overwritten.
3	<b>Open</b>	It specifies that the o/s should open an existing file. If the file doesn't exist, a FileNotFoundException will be thrown.
4	<b>OpenOrCreate</b>	It specifies that the o/s should open an existing file. If the file doesn't exist, a new file will be created. It is useful for reading content of the file.
5	<b>Append</b>	It specifies that the o/s should open an existing file; and seek to end of the file, in order to write content at the end. It is useful with FileAccess.Write



## 'FileAccess' in FileStream

Constants in  
'FileAccess'  
enum

#	Mode	Description
1	Read	It is used to read content from an existing file.
2	Write	It is used to write content to a file.
3	ReadWrite	It can be used for both read / write operation on a file.

## FileStream class - Part 2

# System.IO.FileStream

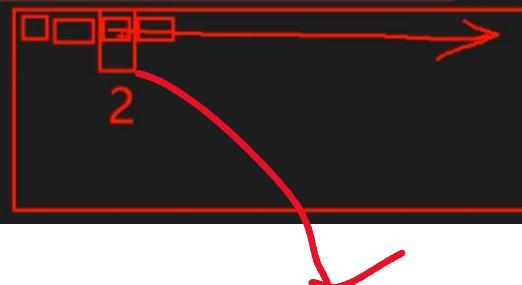
Part 2

| Harsh

ng file.

rations in

```
//create content  
string content = "Dog is one of the domestic animal";  
byte[] bytes = System.Text.Encoding.ASCII.GetBytes(content);  
  
//Write  
fileStream.Write(bytes, );  
}
```



# System.IO.FileStream

Part 3



## FileStream

### Constructors

#	Constructor
1	<b>FileStream(string filePath, FileMode mode, FileAccess access)</b>  It initializes opens the file at specified mode and specified access permission.  <b>FileMode:</b> CreateNew, Create, Open, OpenOrCreate, Append  <b>FileAccess:</b> Read, Write, ReadWrite

## ' FileMode' in FileStream

| Harsha

#	Mode	Description
1	CreateNew	It specifies that the o/s should create a new file. If the file already exists an IOException will be thrown.
2	Create	It specifies that the o/s should create a new file. If the file already exists, it will be overwritten.
3	Open	It specifies that the o/s should open an existing file. If the file doesn't exist, a FileNotFoundException will be thrown.
4	OpenOrCreate	It specifies that the o/s should open an existing file. If the file doesn't exist, a new file will be created. It is useful for reading content of the file.
5	Append	It specifies that the o/s should open an existing file; and seek to end of the file, in order to write content at the end. It is useful with FileAccess.Write

cess

cess

nd

## 'FileAccess' in FileStream

Constants in  
'FileAccess'  
enum

#	Mode	Description
1	<b>Read</b>	It is used to read content from an existing file.
2	<b>Write</b>	It is used to write content to a file.
3	<b>ReadWrite</b>	It can be used for both read / write operations in a file.

## FileStream

Methods

#	Method
1	<b>void Write( byte[] array, int offset, int count)</b>
	Writes the specified no. of bytes specified by count, based on the byte[] specified by 'array' into the file, after the no. of bytes specified by 'offset'.
2	<b>int Read( byte[] array, int offset, int count)</b>
	Read the specified no. of bytes specified by count, into the byte[] specified by 'array' from the file, after the no. of bytes specified by 'offset'.
3	<b>void Close( )</b>
	Closes the file.

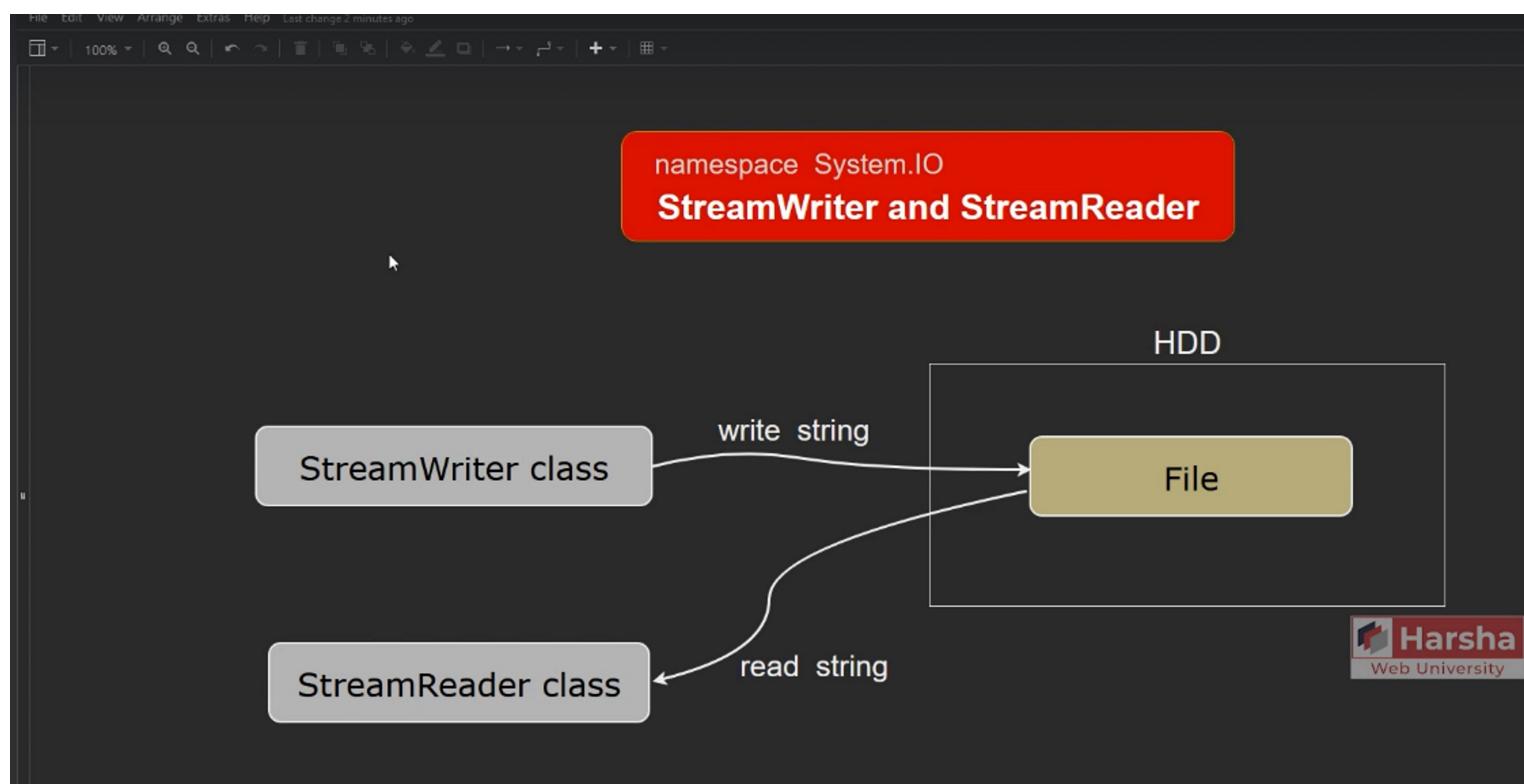
Only the problem with this file stream is that every time for writing as well as reading you have to convert the string into byte[] and vice versa every time. So this process is lengthy and confusing sometimes in order to perform the easier operation on the file, file reader and stream reader is better.

reading  
problem makes the code  
use the same stream writer

## StreamWriter and StreamReader - Part 1

# StreamWriter and StreamReader

Part 1



Masha Vardanyan



When you want to perform multiple operations on the same file, the **FileStream class** is better but of course when you want to perform **single operation like** only you would like to write a single content into the file or single reading operation in that case the **File class** always the best because it has the predefined methods which can be used directly like write all text read all text we have already seen those methods in the file class in the previous lectures but in case of multiple operations like for example at the beginning of the program you would like to write line number 1 into the file then after that after execution of some particular code you would like to write another line into the file again after some piece of code you would like to write the third line into to keep performing multiple operations like multiple times writing or multiple times reading if this is the case the **file stream is the best so far that we know but what's the problem with the** perform the writing and reading operation you require to manually convert your string value into byte array and byte arr

so this makes the code lengthy and cumbersome"

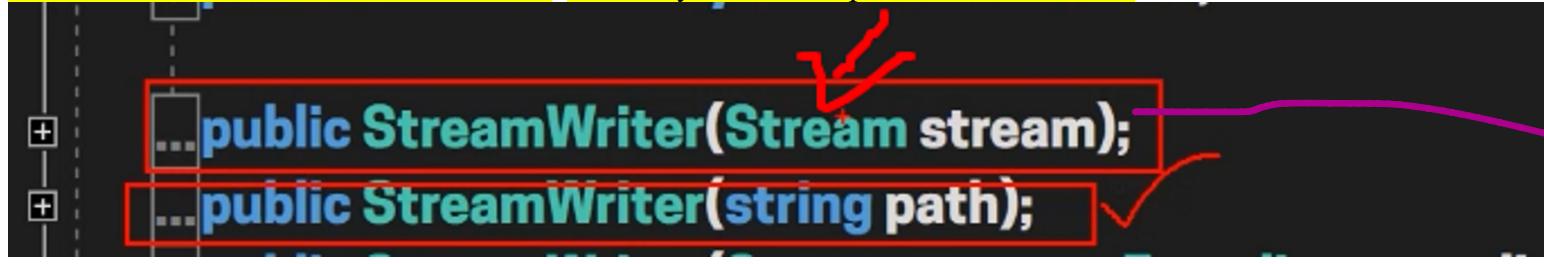
to **overcome this problem** if there is a class that allows you to **write the string values into the file directly** and also can **read** that would be better such classes are stream writer and stream reader the stream writer allows you to write a string directly not manually convert that string into byte array but of course it does the same automatically and internally without your effort and in the same way it allows you to read the existing content from the file and gives you the same as a string reads the content as a byte array and automatically converts the same into string and it gives you the string value directly so these allow you to operate on strings but internally they will handle the that is the beauty of these classes but the limitation for these classes is it works with only text information means text files they cannot work on other than text formats such as for example they cannot work on images audio files video files and also documents or pdf files they cannot work on that in case if you want to work on those files again you have to go back to the files frame but of course working with the audio files video files and pdf files etc require you some additional plugins so

it is out of context at this moment

so let's stick to the text files now

that is stream writer and stream reader

Stream is the base class for the FileStream. Generally, a stream represents a flow of text



For example i am giving you the first byte and the next byte and so on so as a flow I keep on giving the data to you and you are keep on receiving the same this flow of data is called as a stream.

Stream is a parent class which can represent any type of stream. Stream of information. The FileStream class is the child

o the file like that you would like

the file stream every time when you  
ay into string while reading

ad the strings from the file so  
tly into the file you need

g value directly but internally it

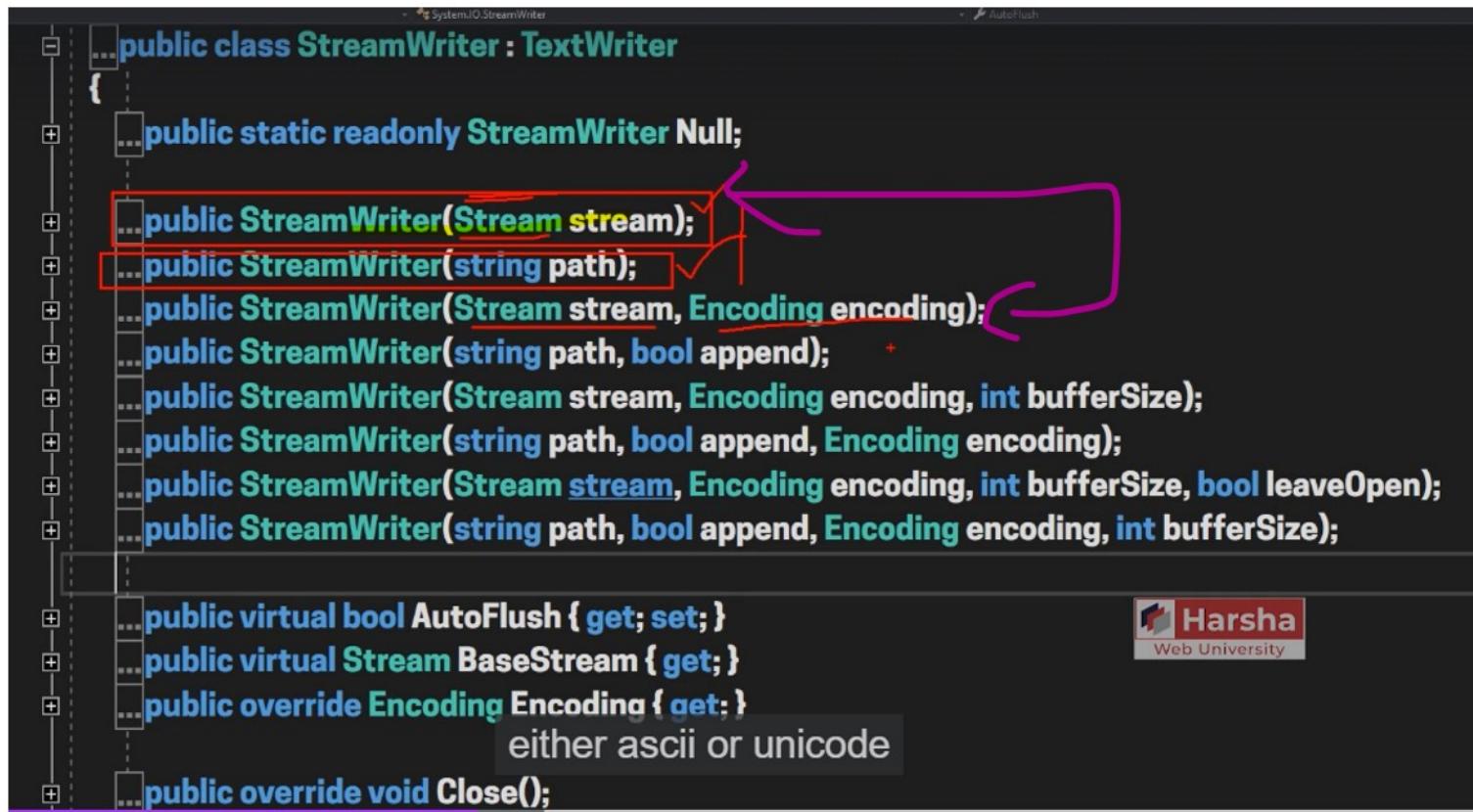
e conversions into byte array so



d of Stream Class.

Stream is a parent class which can represent any type of stream. Stream of information. The FileStream class is the child of Stream.

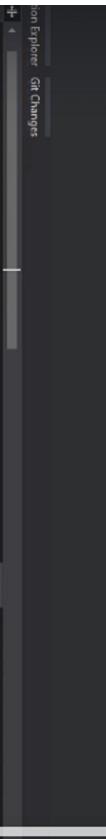
you can supply the child of FileStream class here



```
...public class StreamWriter : TextWriter
{
    public static readonly StreamWriter Null;
    ...public StreamWriter(Stream stream);
    ...public StreamWriter(string path);
    ...public StreamWriter(Stream stream, Encoding encoding);
    ...public StreamWriter(string path, bool append);
    ...public StreamWriter(Stream stream, Encoding encoding, int bufferSize);
    ...public StreamWriter(string path, bool append, Encoding encoding);
    ...public StreamWriter(Stream stream, Encoding encoding, int bufferSize, bool leaveOpen);
    ...public StreamWriter(string path, bool append, Encoding encoding, int bufferSize);

    public virtual bool AutoFlush { get; set; }
    public virtual Stream BaseStream { get; }
    public override Encoding Encoding { get; }
        either ascii or unicode
    public override void Close();
}
```

of Stream Class.



```
class Program
{
    static void Main()
    {
        string filePath = @"c:\practice\europ.txt";
        //StreamWriter streamWriter = new StreamWriter(filePath);
        FileStream fileStream = new FileStream(filePath, FileMode.Create, FileAccess.Write);
        StreamWriter streamWriter = new StreamWriter(fileStream);
        streamWriter.WriteLine("Hello")
    }
}
```

The diagram illustrates the relationship between the `StreamWriter` and `FileStream` classes. A yellow arrow points from the declaration of `StreamWriter` to a red box labeled "StreamWriter". A purple arrow points from the declaration of `FileStream` to the same red box. A green arrow points from the declaration of `FileStream` to a red box labeled "FileStream Class".

StreamWriter Class is a wrapper of FileStream Class.



**Methods**

#	Method
8	<b>StreamWriter CreateText( )</b>
	Opens the file in 'Create' mode & creates and returns an object of StreamWriter class that can write text to the file.
9	<b>StreamWriter AppendText( )</b>
	Opens the file in 'Append' mode & creates and returns an object of StreamWriter class that can write the appended text to the file.
10	<b>StreamReader OpenText( )</b>
	Opens the file in 'Open' mode & creates and returns an object of StreamReader class that can read text from the file.

```

class Program
{
    static void Main()
    {
        string filePath = @"c:\practice\europe.txt";
        FileInfo fileInfo = new FileInfo(filePath);
        //StreamWriter streamWriter = new StreamWriter(filePath);
        //FileStream fileStream = new FileStream(filePath, FileMode.Create, FileAccess.Write);
        //StreamWriter streamWriter = new StreamWriter(fileStream);
        using (StreamWriter streamWriter = fileInfo.CreateText())
        {
            streamWriter.WriteLine("Hello");
            //streamWriter.Close(); //optional
        }
    }
}

```

The code illustrates the creation of a StreamWriter object using two different approaches:

- Approach 1 (commented out): `StreamWriter streamWriter = new StreamWriter(filePath);`
- Approach 2 (used): `using (StreamWriter streamWriter = fileInfo.CreateText())`

A callout box highlights the `fileInfo.CreateText()` method call, showing its documentation:

**CreateText()**  
Creates a StreamWriter that writes a new text file.  
Returns: A new StreamWriter.  
Exceptions: IOException

A red circle labeled '3' points to the `new StreamWriter` part of the code, which is annotated with a callout box:

**new StreamWriter**  
Creates a StreamWriter that writes a new text file.  
Returns: A new StreamWriter.  
Exceptions: IOException

A large purple arrow points from the callout box back towards the original code.

this line is equivalent to 'StreamWriter streamWriter = new StreamWriter(fileStream);'

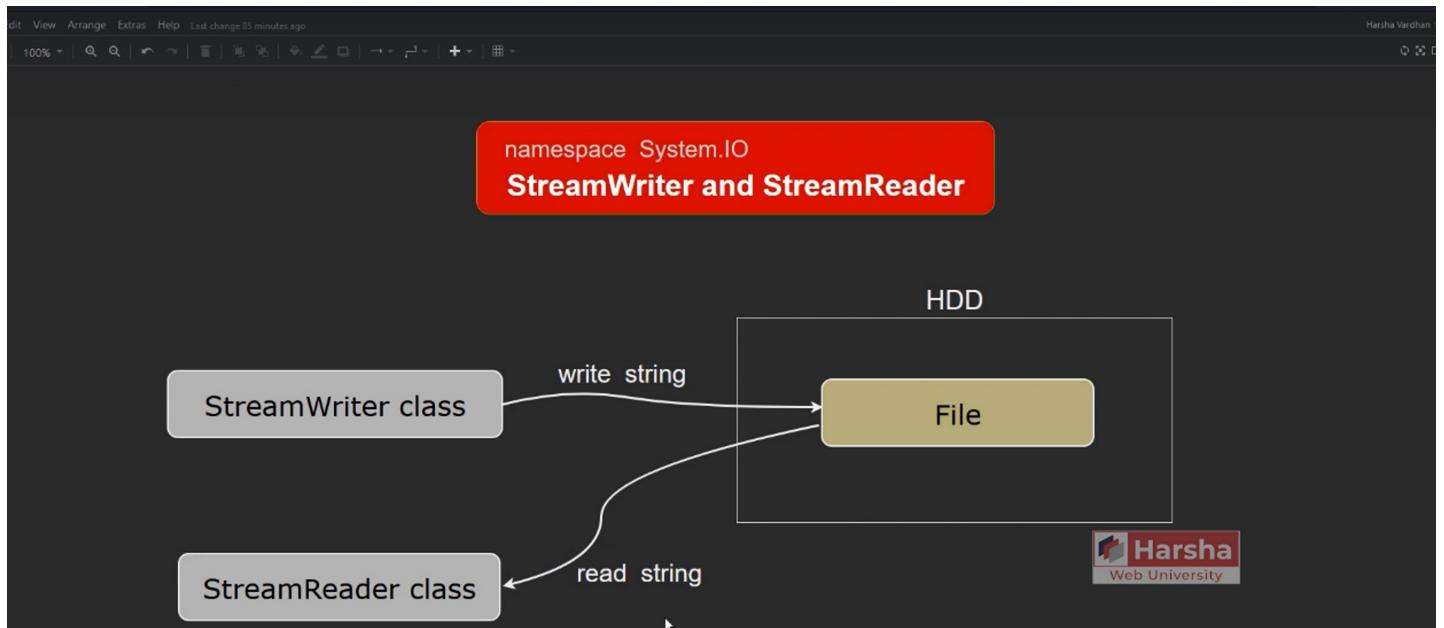


- These are the 3 ways of creating StreamWriter Object.

## **StreamWriter and StreamReader**

**Part 2**





```

using (StreamReader streamReader = new StreamReader(fileStream2))
{
    Console.WriteLine("\nFile read. File content is:");

    //To read full file
    //string content_from_file = streamReader.ReadToEnd();
    //Console.WriteLine(content_from_file);

    //To read part by-part (10 characters)
    char[] buffer = new char[10];
    streamReader.Read(buffer, 0, 10);
}

Console.ReadKey();
  
```

**IntelliSense tooltip:** int StreamReader.Read(char[] buffer, int index, int count)  
 Reads a specified maximum of characters from the current stream into a buffer, beginning at the specified index.  
 Index: The index of buffer at which to begin writing.

**Diagram:** A red box highlights the line `char[] buffer = new char[10];`. To its right, a red bracket groups the code `streamReader.Read(buffer, 0, 10);` and points to a diagram. The diagram shows a horizontal array of 10 cells. The first two cells contain 'a' and 'b'. An arrow labeled '10' points to the end of the array. Below the array, the characters 'a' through 'j' are shown in a sequence, with 'k' starting after the array's end.

Harsha Web University



**What**

- › Class that writes text data into the file.
- › Internally uses FileStream.

**How****StreamWriter**

```
StreamWriter referenceVariable = new StreamWriter( "File Path" );
```

**Constructors**

Constructor	Description
<code>StreamWriter(FileStream stream)</code>	It initializes the StreamWriter based on the specified FileStream.
<code>StreamWriter(string path)</code>	It initializes the StreamWriter for the specified file path.

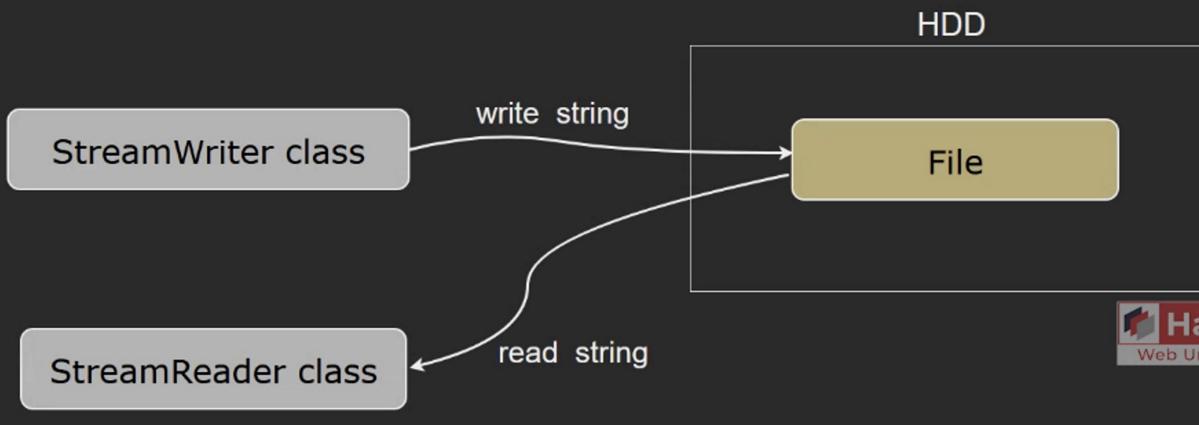
## BinaryWriter and BinaryReader

**Part 1**



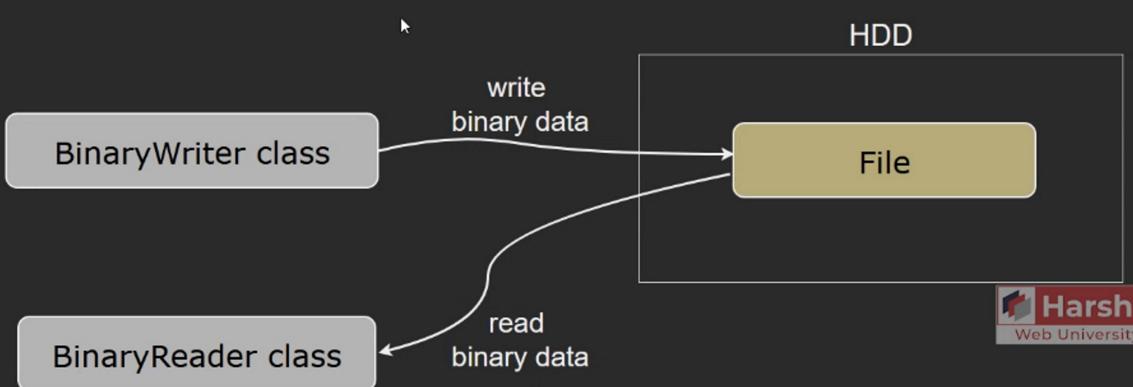
```
namespace System.IO
```

## StreamWriter and StreamReader

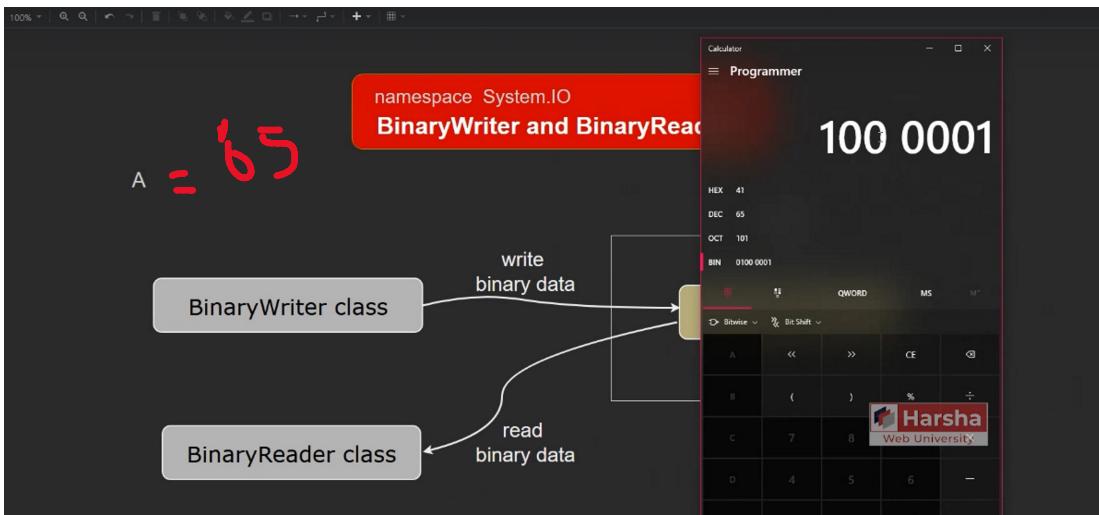
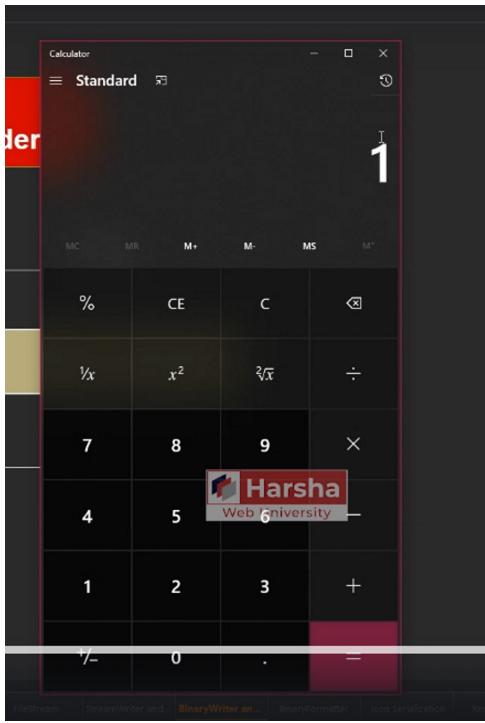


```
namespace System.IO
```

## BinaryWriter and BinaryReader







You can also write information into the file by using binarywriter . so what is the difference between streamwriter and binarywriter ? The stream writer can write the information only as a string even though you might be writing numbers and other data types. They are first converted into string and finally that string internally converted as a byte array and that byte array will be stored in the file but whereas the binary writer works in a little bit different way for example you have a number that is for example 1

at first this number will be converted as a binary format that is zeros and ones format already earlier we have understood about number systems and we know how do you convert the number from decimal number system into binary number system in the calculator in windows.

Open the Windows Calculator and select programmer and once you selected programmer if you give a number here for example 1 the same will be printed in all the other number systems .So the decimal number one is equal to the binary number zero zero zero one for example we have to write digit one by using the binarywriter at first this will be

d  
s

it

al

or

e

converted into binary format and that binary numbers are written into the file that's the point about binarywriter

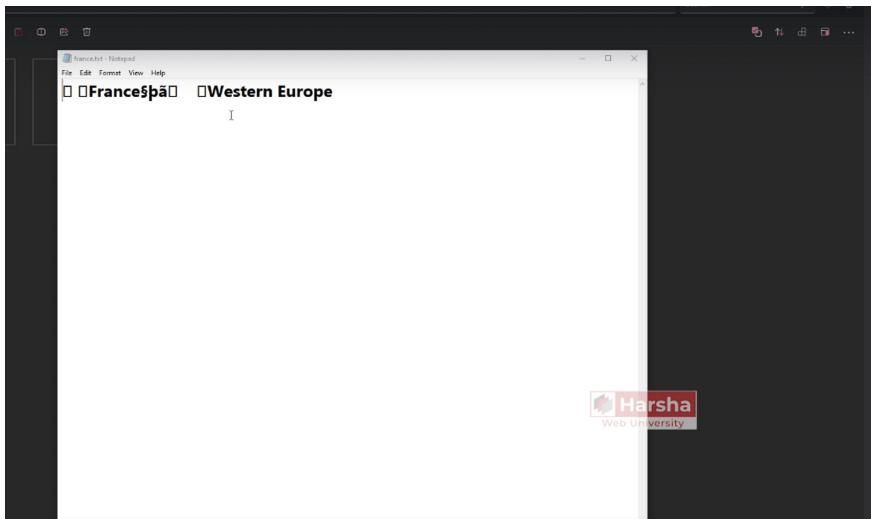
But what about the characters and strings? For example as a part of this string you have a character called a alphabet what is the equivalent ascii code of the character 'A'

For example if it is uppercase a it will be equivalent to 65 right so that 65 will be converted into binary format and it is 0 1 0 0 0 0 000 1 so this binary number will be stored in the file in that way the binder rater works and in the same way the binary reader also works means the existing information which is in binary format will be read and eventually you can convert the same into the dot net data type such as carr string int bool etc

The screenshot shows a code editor window with a dark theme. The code is as follows:

```
namespace BinaryWriterReaderExample
{
    class Program
    {
        static void Main()
        {
            short countryId = 1;
            string countryName = "France";
            long population = 65273511;
            string region = "Western Europe";
            string filePath = @"c:\practice\france.txt";
            FileStream fileStream = new FileStream(filePath, FileMode.Create, FileAccess.Write);
            BinaryWriter binaryWriter = new BinaryWriter(fileStream);
        }
    }
}
```

A red oval highlights the line `FileStream fileStream = new FileStream(filePath, FileMode.Create, FileAccess.Write);`. A red arrow points from the label "fileStream" to the variable `fileStream` in the code. The status bar at the bottom right shows "Ln 14 Ch 57 Col 1".



a

s

All the information are stored in binary format. But when you try to open the file in notepad, notepad try to convert the binary file to string format. But it can not convert the number to string.

# BinaryWriter and BinaryReader

## Part 2

Let us see how do you read existing binary file by using this Binary Reader if you write information through Binary Writer then only you are able to use the Binary Reader in order to read the same.

It's general opinion that, binary file written by binary writer takes less amount of memory.

Reader - Part 2

### BinaryWriter

**What**

- > Class that writes binary data into the file.
- > Internally uses FileStream.
- > For example, 65 is written as 100 0001.

**How**

```
BinaryWriter
```

```
BinaryWriter referenceVariable = new BinaryWriter( fileStream );
```

**Constructors**

Constructor	Description
<code>BinaryWriter(FileStream stream)</code>	It initializes the BinaryWriter based on the specified FileStream.

 Harsha  
Web University



## BinaryWriter

**Methods**

#	Method
1	<code>void Write( ... )</code> Writes any type of value (only primitive types / string) into the file.
2	<code>void Close( )</code> Close the file.

```

class Program
{
    static void Main()
    {
        short countryId = 1;
        string countryName = "France";
        long population = 65273511;
        string region = "Western Europe";
        string filePath = @"c:\practice\france.txt";
        FileStream fileStream = new FileStream(filePath, FileMode.Create, FileAccess.Write);
        using (BinaryWriter binaryWriter = new BinaryWriter(fileStream))
        {
            binaryWriter.Write(countryId); //0001
            binaryWriter.Write(countryName); //100 0100...
            binaryWriter.Write(population);
            binaryWriter.Write(region);
            //binaryWriter.Close(); // fields or properties
        }
    }
}

```

“If we had used an object, we wouldn’t have needed to write multiple read and write methods.”

This is called as **Serialization**.



**Methods**

#	Method
1	<code>byte ReadByte()</code>
	Reads and returns a byte value from the file.
2	<code>ReadSByte()</code> , <code>ReadInt16()</code> , <code>ReadUInt16()</code> , <code>ReadInt32()</code> , <code>ReadUInt32()</code> , <code>ReadInt64()</code> , <code>ReadUInt64()</code> , <code>ReadSingle()</code> , <code>ReadDouble()</code> , <code>ReadDecimal()</code> , <code>ReadChar()</code> , <code>ReadString()</code> , <code>ReadBoolean()</code>
	Reads and returns the specific type of value from the file.
3	<code>string Close()</code>
	Closes the file.

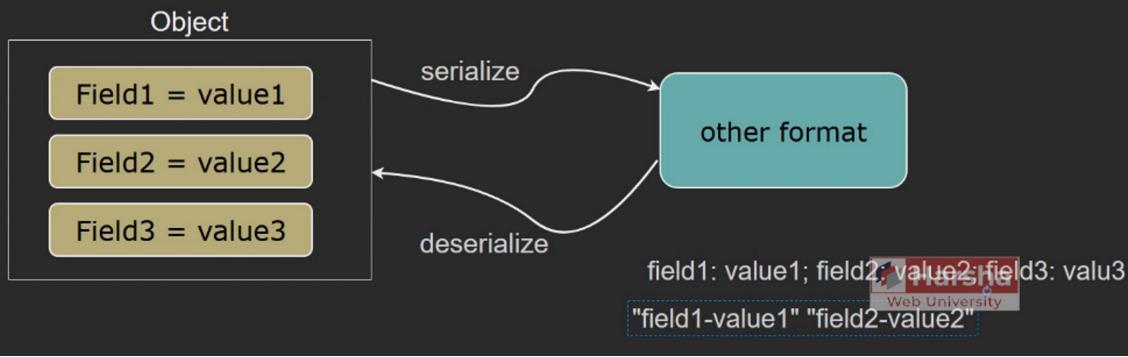
## Binary Serialization

### Part 1

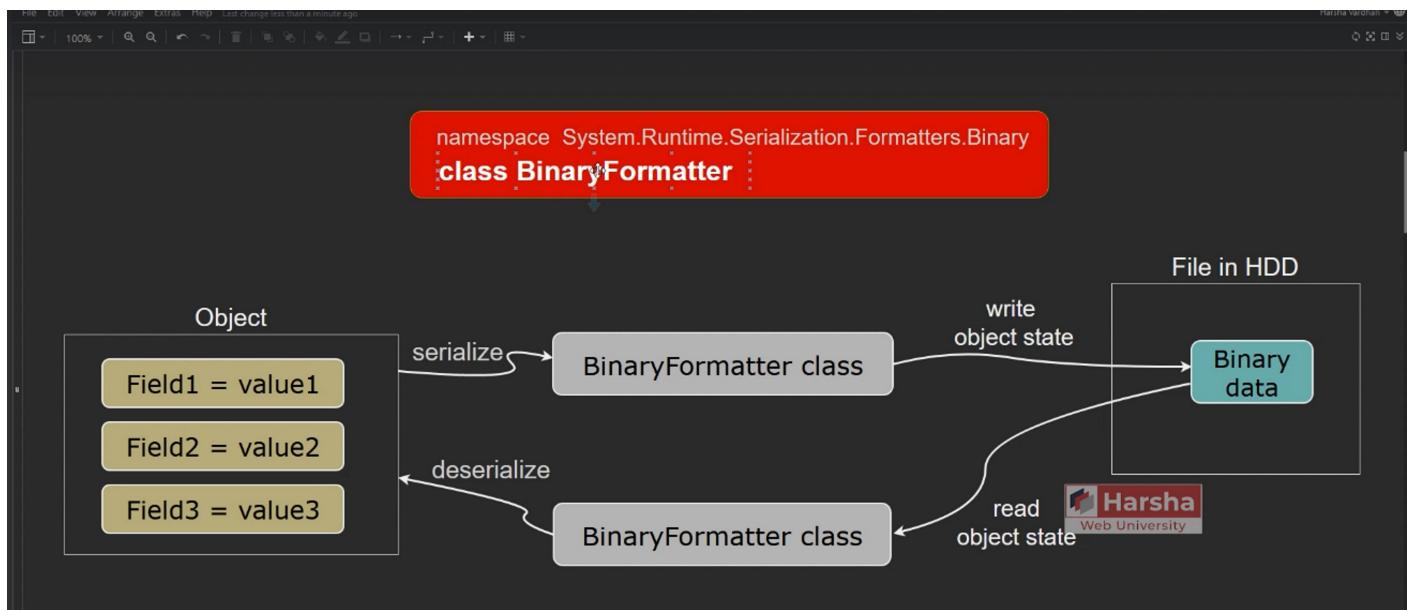
In c-sharp you have a concept of Serialization that means converting an object from one state to another state means from one format to another format. suppose for example you have converted your object from heap into a different maybe for example binary format or xml format or something else so that process of conversion is called a serialization.

erent format  
on

## Serialization



In c sharp mainly and importantly we have three predefined serializations that is binary serialization json serialization and xml serialization. In this lecture let us focus on binary serialization



So in this case through serialization concept you are serializing a specific state of the object into binary formats so that it can be saved in a file and later maybe from same application or from a different application you can rewrite means get back the serialized data into the object in this way two different applications can be interacted each other

for example you have created an application where you have an object with some fields with values for example there is a customer object with customer id name email etc so you have serialized that object and you have saved that serialized data in a file and you have given that file to me

and i can deserialize myself right so in my application i can read the content of your file

n be  
ame

so that i can load this exactly same data into another object in my application  
in that way your application can send the data to my application  
not directly but through a file.

so for saving the information permanently in a file otherwise communicating the data from one application to another application we can use this serialization concept in the real-world applications  
of course most of the cases in the real life we use databases instead so you can consider that.

This might be a or technique where you cannot use the databases

```
0 references
static void Main()
{
    //create object
    Country country = new Country() { CountryID = 1, CountryName = "Russia", Population = 145934000, Region = "Eastern Europe" };

    //create FileStream
    string filePath = @"c:\practice\russia.txt";
    FileStream fileStream = new FileStream(filePath, FileMode.Create, FileAccess.Write);

    //create BinaryFormatter
    BinaryFormatter binaryFormatter = new BinaryFormatter();
    binaryFormatter.Serialize(fileStream, country);
}
```



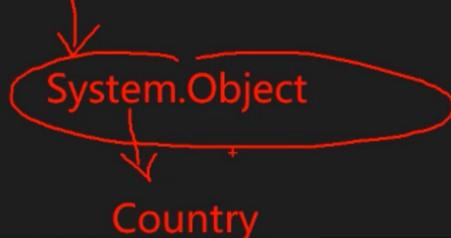
# Binary Serialization

Part 2



```
Console.WriteLine("Serialized");

//Code for deserialization
FileStream fileStream2 = new FileStream(filePath, FileMode.Open, FileAccess.Read);
(Country)binaryFormatter.Deserialize(fileStream2);
Console.ReadKey();
}
```



## BinaryFormatter

### What

- Class that serializes (converts) an object-state into binary format into and stores in binary file.
- Serialization is a process of converting an object from one format to another format.
- It can also read existing object-state from the binary file.
- Full Path: System.Runtime.Serialization.Formatters.Binary.BinaryFormatter

### How

#### BinaryFormatter



```
BinaryFormatter referenceVariable = new BinaryFormatter();
```



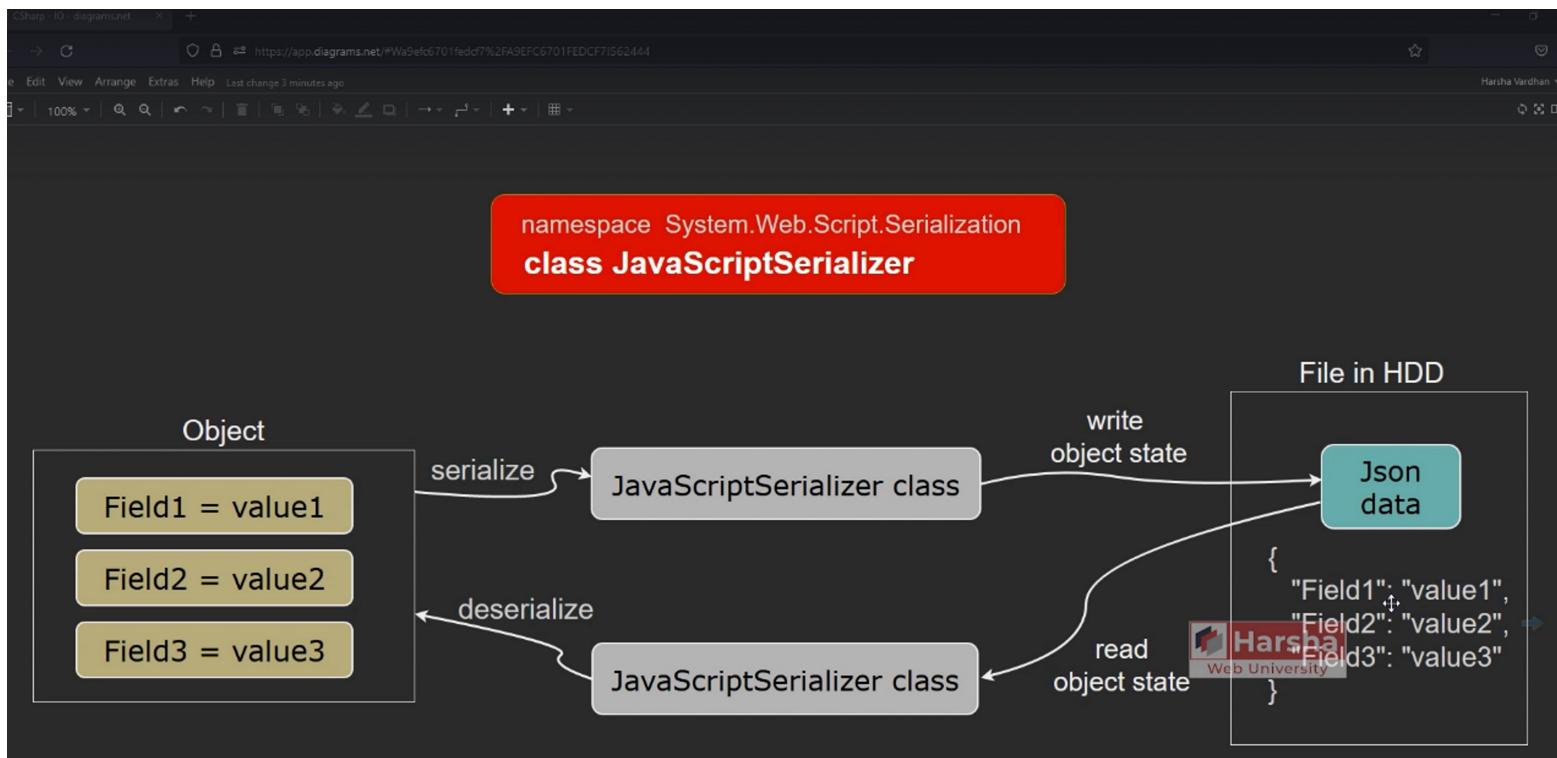
## BinaryFormatter

### Methods

#	Method
1	<code>void Serialize( FileStream FileStream, object data)</code> Converts the object-state into the binary file, using the specified <code>FileStream</code> (with Write access).
2	<code>object Deserialize( FileStream FileStream)</code> Reads the existing object-state from the binary file, using the specified <code>FileStream</code> (with Read access).

## Json Serialization





We have understood the purpose of the serialization and we have performed the binary serialization in the last lecture but the limitation of this **binary serialization** is that you can de-serialize means get back the same information into any dot.net application only. you cannot read the same information from other language applications

that means for example you cannot deserialize the same information from an application that was developed in java or python so you can serialize and deserialize only from dotnet application only. that is the problem with the binary serialization.

But in case if you want to write the content in a universally acceptable and understandable format which can be read from any type of application from a java application python application of course from the c-sharp application or from any other language application you will be able to read the information from application, if this is the requirement for you then you can think about json serialization which can be implemented by java's serializer.

```

3 [using System.Web.Script.Serialization;
4
5 namespace JsonSerializerExample
6 {
7     [Serializable]
8     public class Customer
9     {
10        public int CustomerId { get; set; }
    
```

any other  
script

```
3  using System.Web.Script.Serialization;
4
5  namespace JsonSerializerExample
6  {
7      [Serializable]
8      public class Customer
9      {
10         public int CustomerId { get; set; }
11         public string CustomerName { get; set; }
12         public int Age { get; set; }
13     }
14 }
```

It indicates to C# that, this class objects can be converted into any other format such as binary format or JSON format or XML format or any other different format.



## JavaScriptSerializer

Harsha

### What

- › Class that serializes (converts) an object-state into JSON format.
- › It can also convert JSON data into object of any class.
- › Full Path: [System.Web.Script.Serialization.JavaScriptSerializer](#)

## JavaScriptSerializer

**//serialization:**

```
javaScriptSerializer.Serialize( YourObject );
```



**//deserialization:**

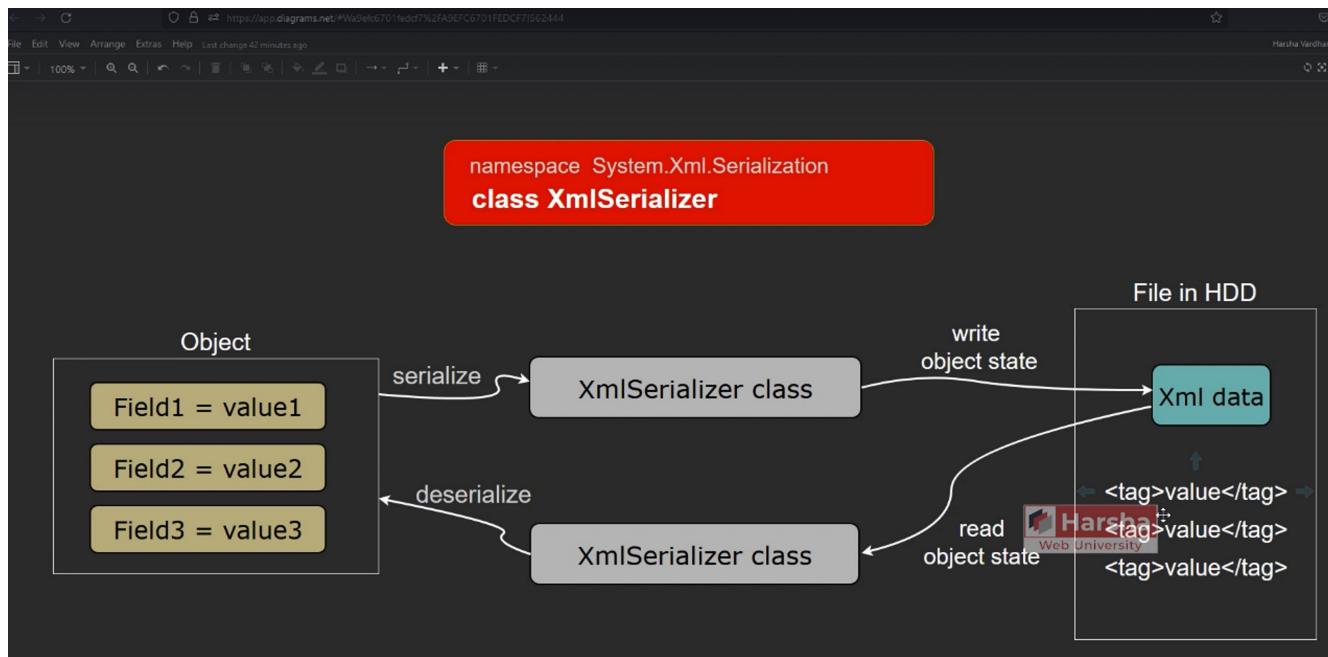
```
javaScriptSerializer.Deserialize( string jsonData, typeof(ClassName) );
```

For example you can serialize in c sharp application here and Deserialize the same in a java application or javascript application or something else and in the same way it can be reversed for example some other java developer has serialized the information in java application and you can perform deserialization here so across applications you can transfer the data that is the advantage or purpose of this aim

Not only data sharing but also you can save the data permanently in the form of a file with the help of this deserialization concept.

## Xml Serialization





## XmlSerializer

### What

- › Class that serializes (converts) an object-state into XML format and stores in XML file.
- › It can also read existing object-state from the XML file.
- › Full Path: **System.Xml.Serialization.XmlSerializer**

### How

#### XmlSerializer

```
XmlSerializer referenceVariable = new XmlSerializer( typeof(ClassName) );
```



**Methods**

#	Method
1	<b>void Serialize( FileStream FileStream, object data)</b> Converts the object-state into the xml file, using the specified FileStream (with Write access).
2	<b>object Deserialize( FileStream fileStream)</b> Reads the existing object-state from the xml file, using the specified FileStream (with Read access).

