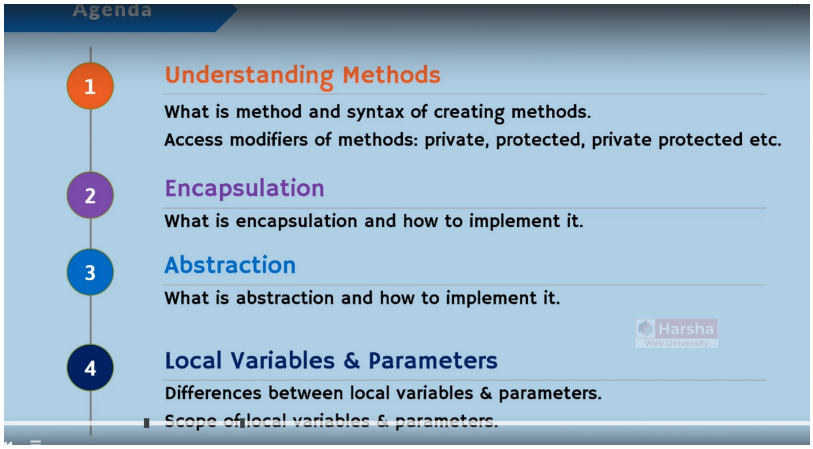
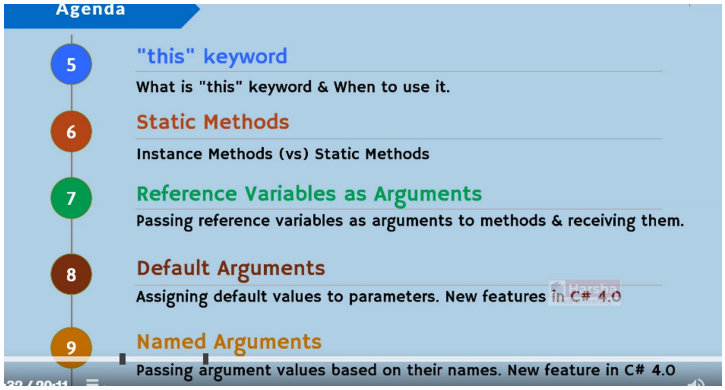
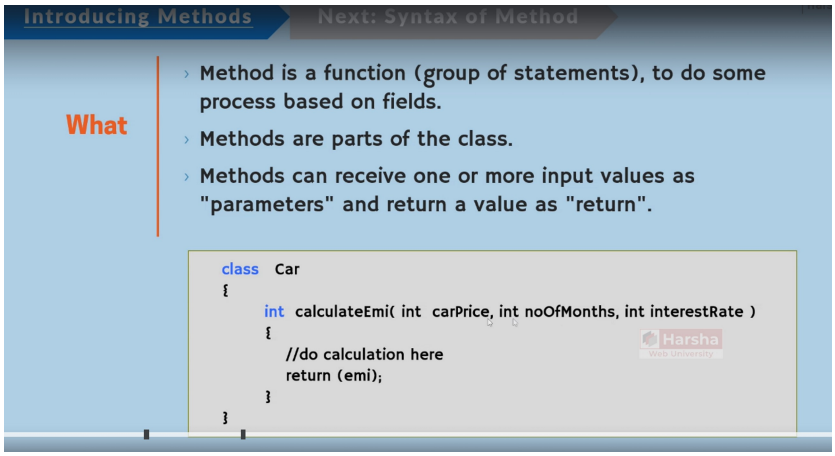
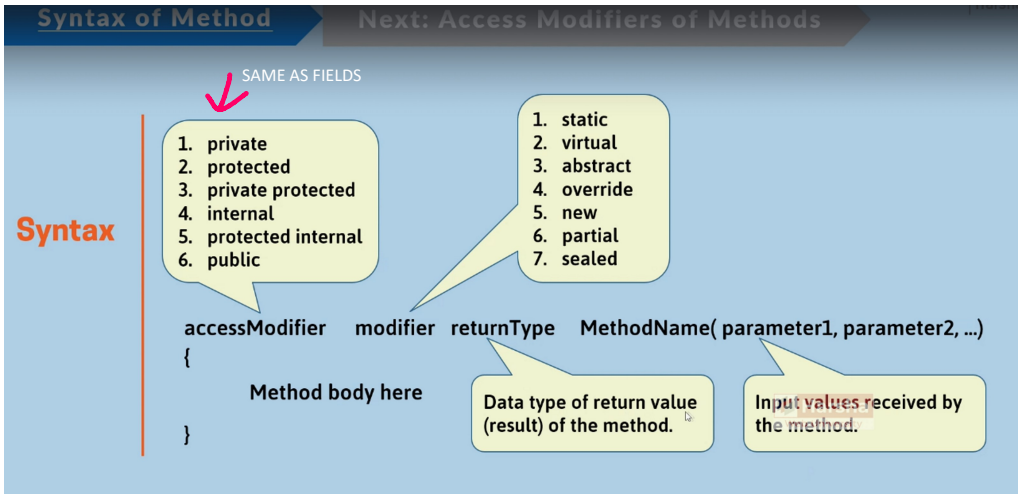
Section 6 : Methods

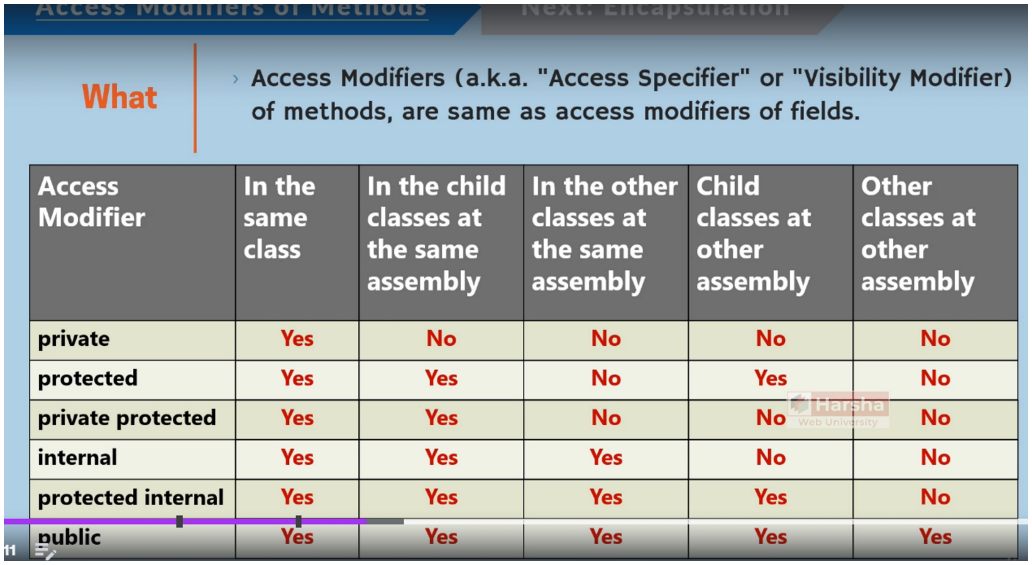
09 July 2024 15:53

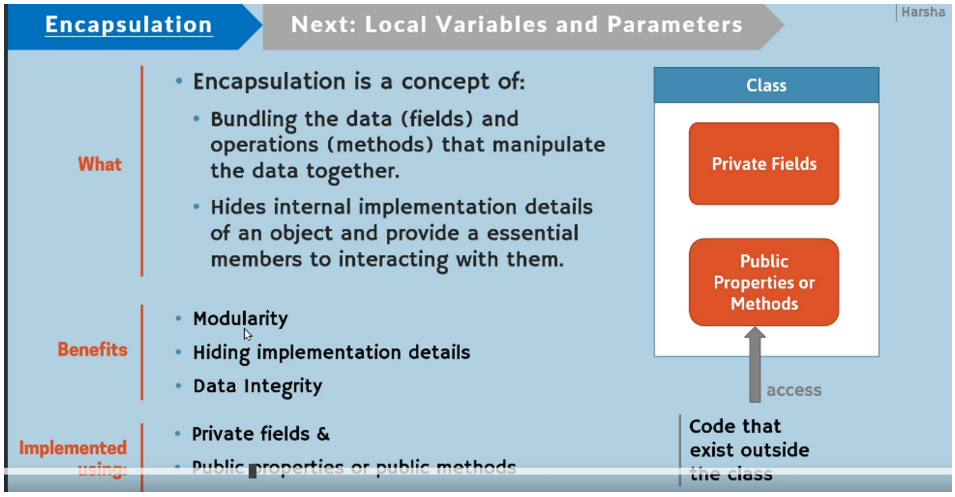


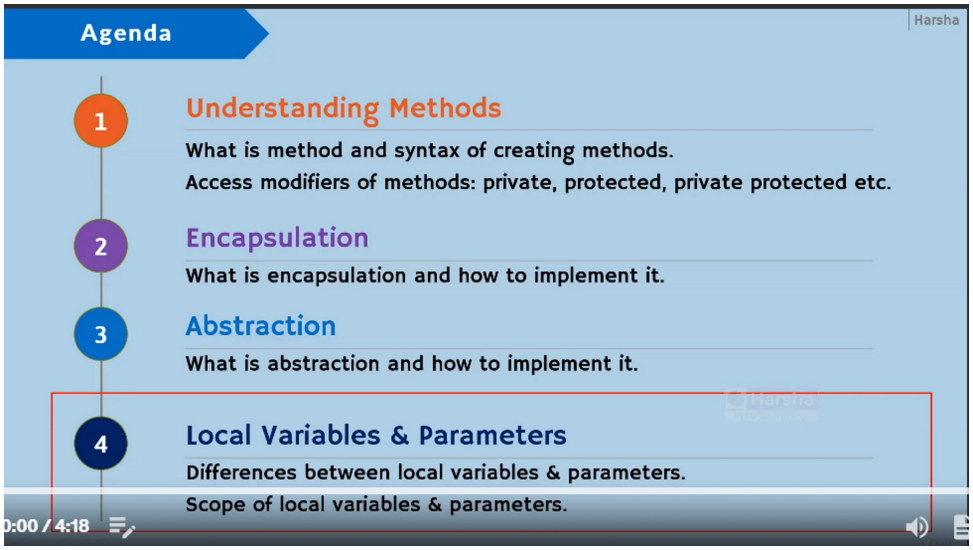


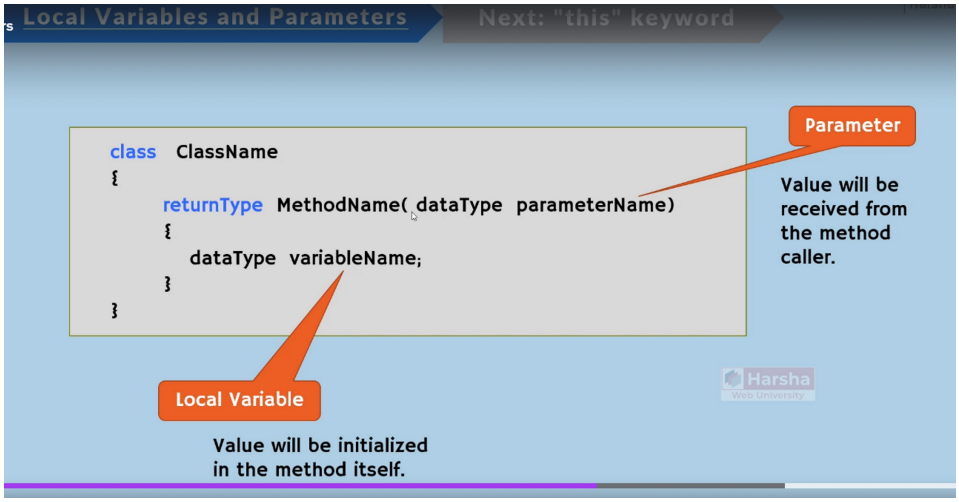


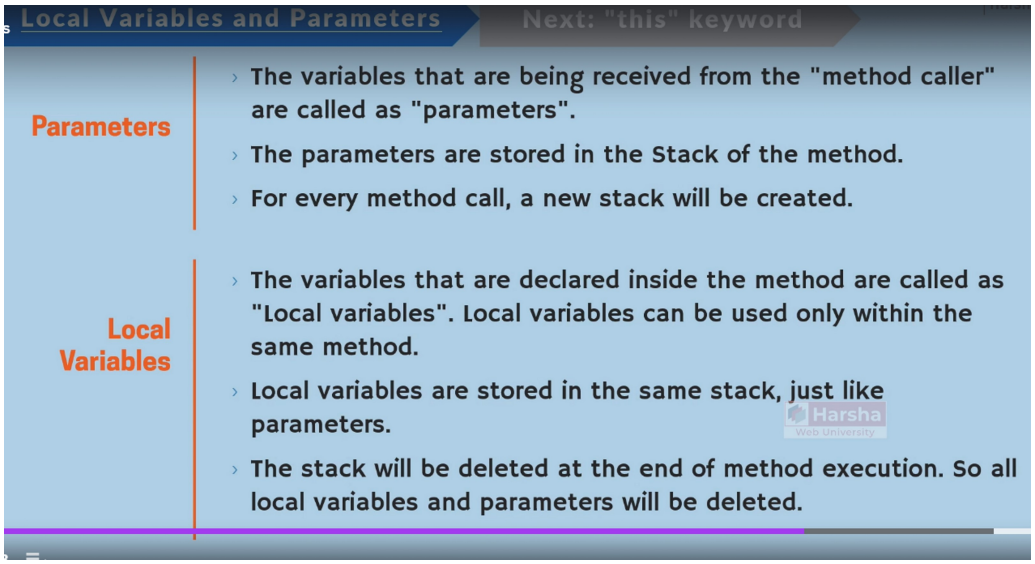


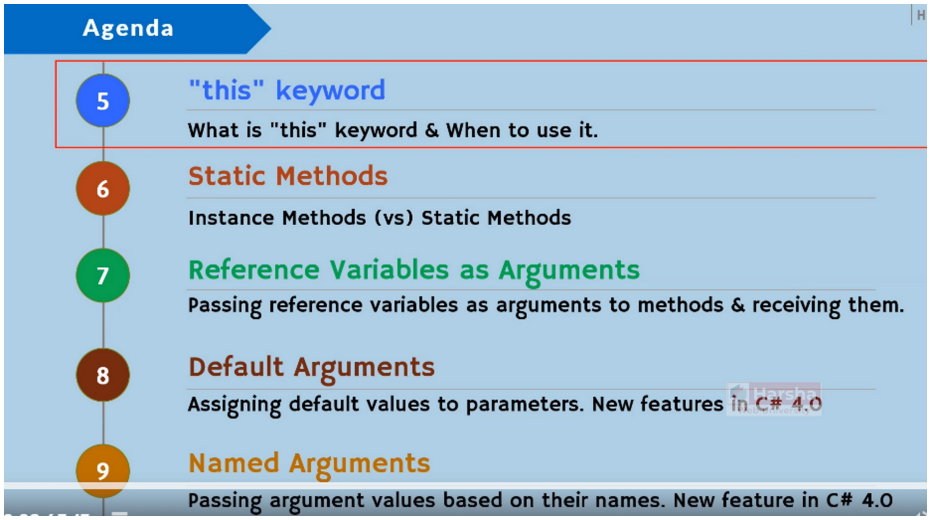

Access Modifiers of Methods
Next: Encapsulation
What
Access Modifiers (a.k.a. "Access Specifier" or "Visibility Modifier) of methods, are same as access modifiers of fields.
Access Modifier | In the same class | In the child classes at the same assembly | In the other classes at the same assembly | Child classes at other assembly | Other classes at other assembly
private | Yes | No | No | No | No
protected | Yes | Yes | No | Yes | No
private protected | Yes | Yes | No | No | No
internal | Yes | Yes | Yes | No | No
protected internal | Yes | Yes | Yes | Yes | No
public | Yes | Yes | Yes | Yes | Yes

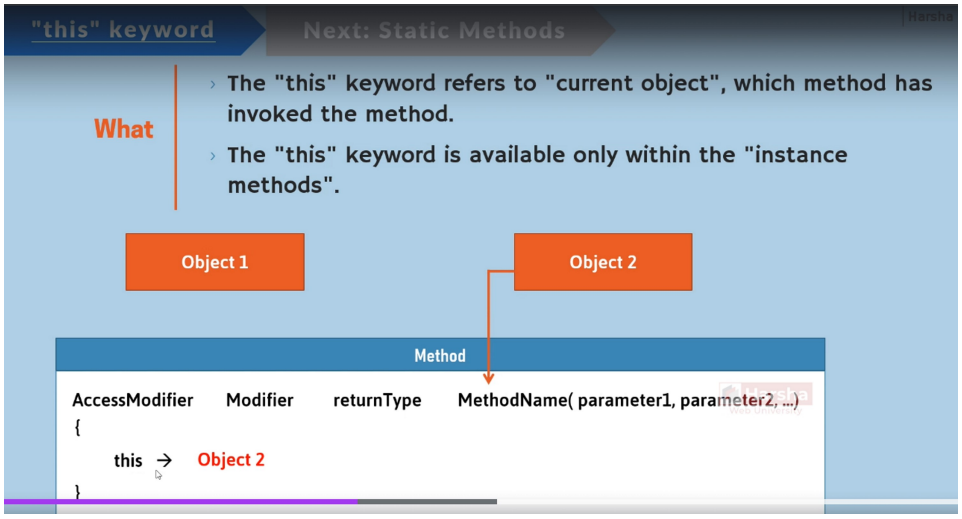

Encapsulation
Next: Local Variables and Parameters
Harsha
What
Encapsulation is a concept of:
Bundling the data (fields) and operations (methods) that manipulate the data together.
Hides internal implementation details of an object and provide a essential members to interacting with them.
Benefits
Modularity
Hiding implementation details
Data Integrity
Implemented using:
Private fields &
Public properties or public methods
Class
Private Fields
Public Properties or Methods
access
Code that exist outside the class

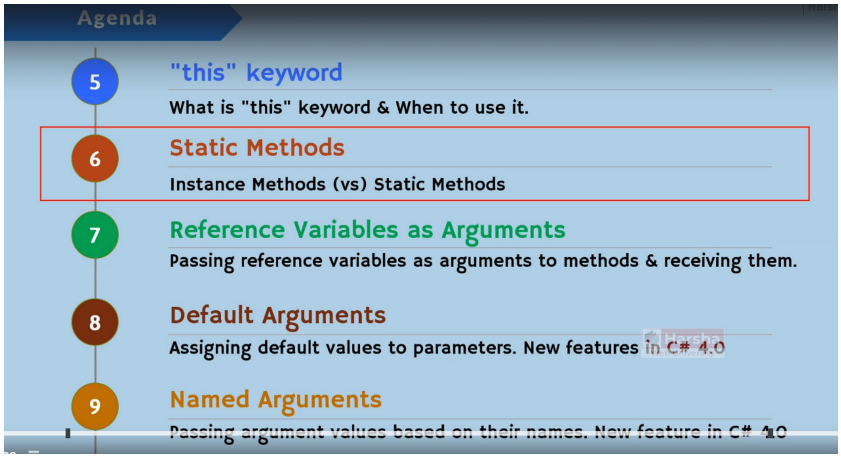

Agenda
Harsha
1
Understanding Methods
What is method and syntax of creating methods.
Access modifiers of methods: private, protected, private protected etc.
2
Encapsulation
What is encapsulation and how to implement it.
3
Abstraction
What is abstraction and how to implement it.
4
Local Variables & Parameters
Differences between local variables & parameters.
Scope of local variables & parameters.
0:00 / 4:18

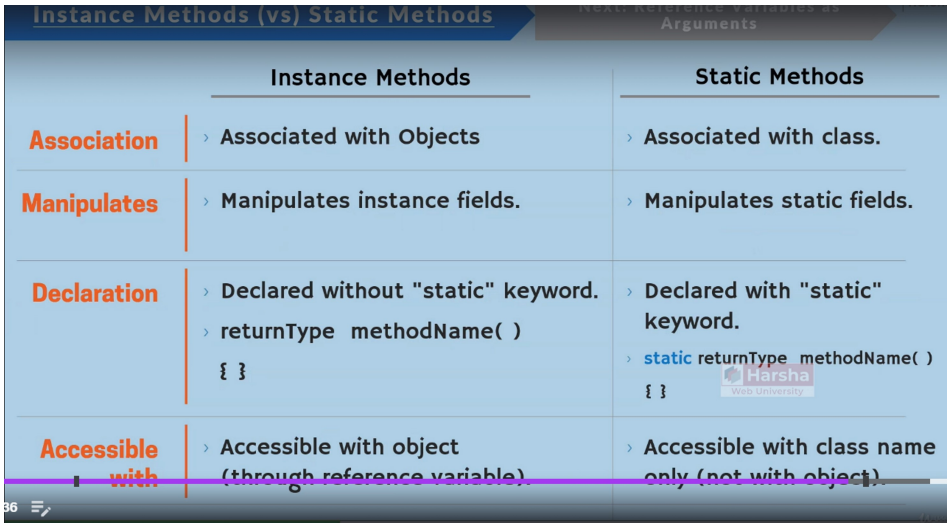

Local Variables and Parameters
Next: "this" keyword
class ClassName
{
returnType MethodName(dataType parameterName)
{
dataType variableName;
}
}
Parameter
Value will be received from the method caller.
Local Variable
Value will be initialized in the method itself.
Harsha
Web University

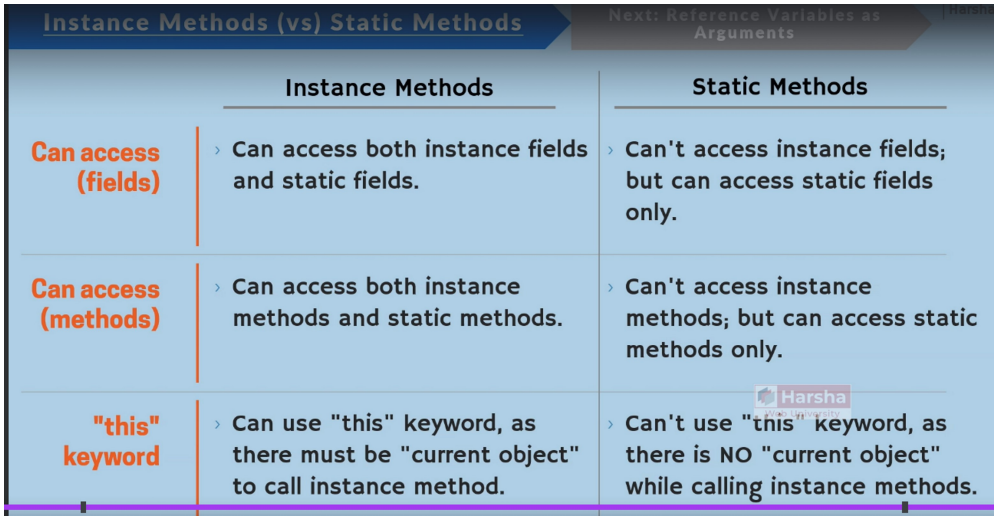

Local Variables and Parameters
Next: "this" keyword
Parameters
The variables that are being received from the "method caller" are called as "parameters".
The parameters are stored in the Stack of the method.
For every method call, a new stack will be created.
Local Variables
The variables that are declared inside the method are called as "Local variables". Local variables can be used only within the same method.
Local variables are stored in the same stack, just like parameters.
The stack will be deleted at the end of method execution. So all local variables and parameters will be deleted.
Harsha
Web University

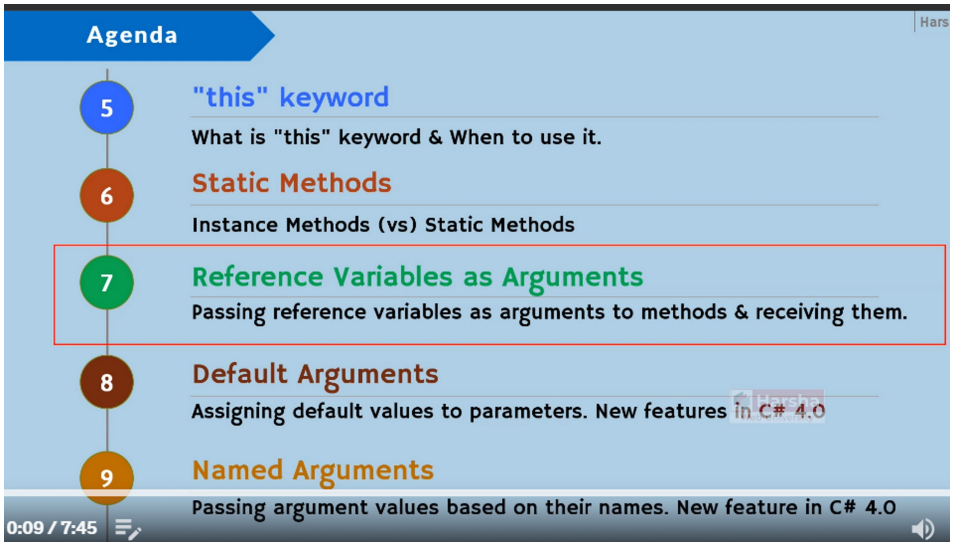

Agenda
5
"this" keyword
What is "this" keyword & When to use it.
6
Static Methods
Instance Methods (vs) Static Methods
7
Reference Variables as Arguments
Passing reference variables as arguments to methods & receiving them.
8
Default Arguments
Assigning default values to parameters. New features in C# 4.0
9
Named Arguments
Passing argument values based on their names. New feature in C# 4.0

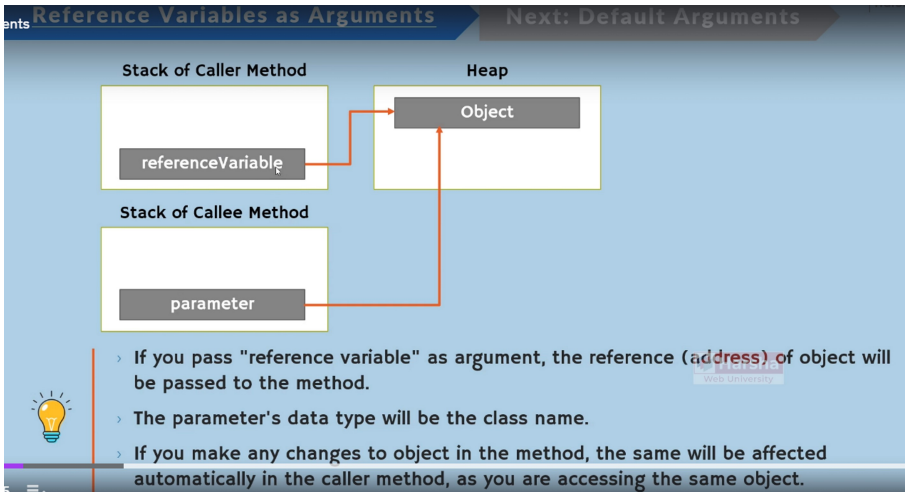

"this" keyword
Next: Static Methods
What
The "this" keyword refers to "current object", which method has invoked the method.
The "this" keyword is available only within the "instance methods".
Object 1
Object 2
Method
AccessModifier Modifier returnType MethodName(parameter1, parameter2, ...)
{
this → Object 2
}

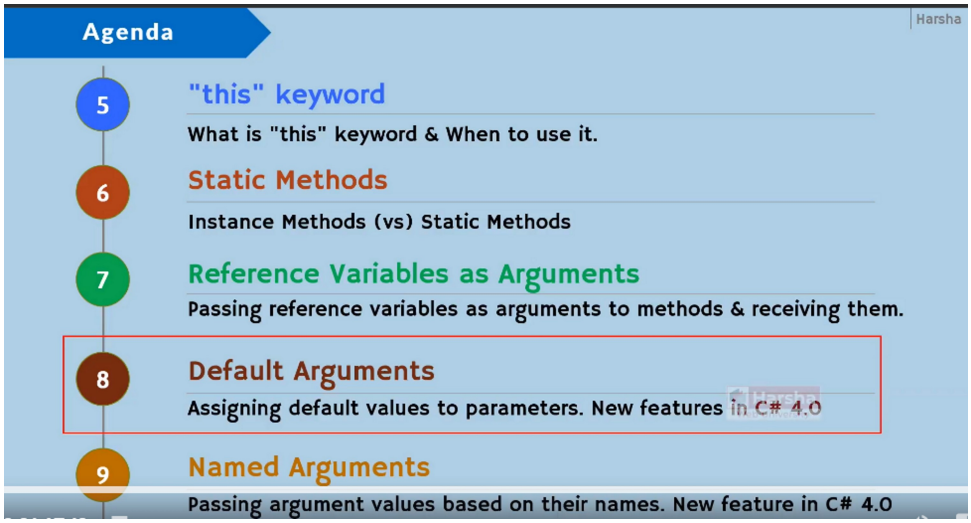

Agenda
5
"this" keyword
What is "this" keyword & When to use it.
6
Static Methods
Instance Methods (vs) Static Methods
7
Reference Variables as Arguments
Passing reference variables as arguments to methods & receiving them.
8
Default Arguments
Assigning default values to parameters. New features in C# 4.0
9
Named Arguments
Passing argument values based on their names. New feature in C# 4.0

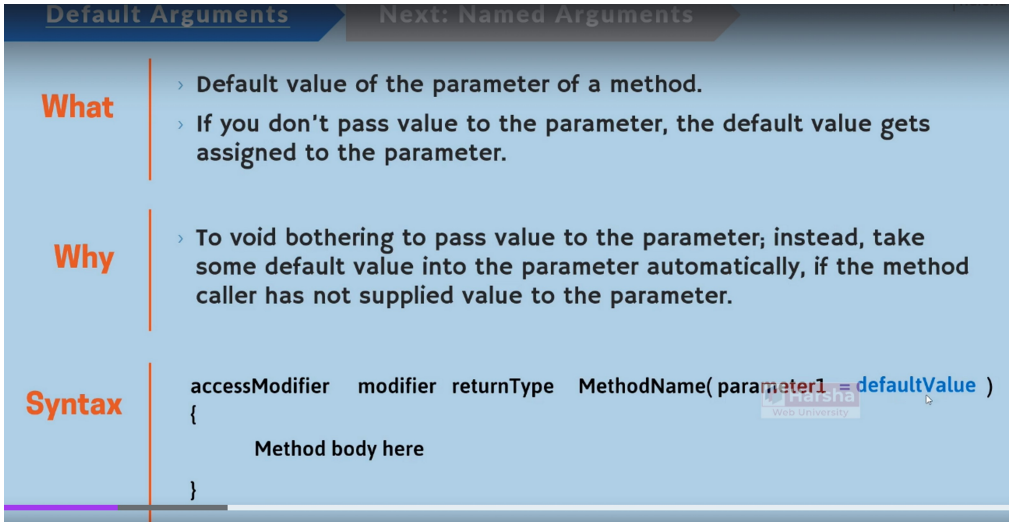

Instance Methods (vs) Static Methods
Next: Reference Variables as Arguments
Instance Methods
Static Methods
Association
Associated with Objects
Associated with class.
Manipulates
Manipulates instance fields.
Manipulates static fields.
Declaration
Declared without "static" keyword.
returnType methodName( )
{ }
Declared with "static" keyword.
static returnType methodName( )
{ }
Accessible with
Accessible with object (through reference variable).
Accessible with class name only (not with object).

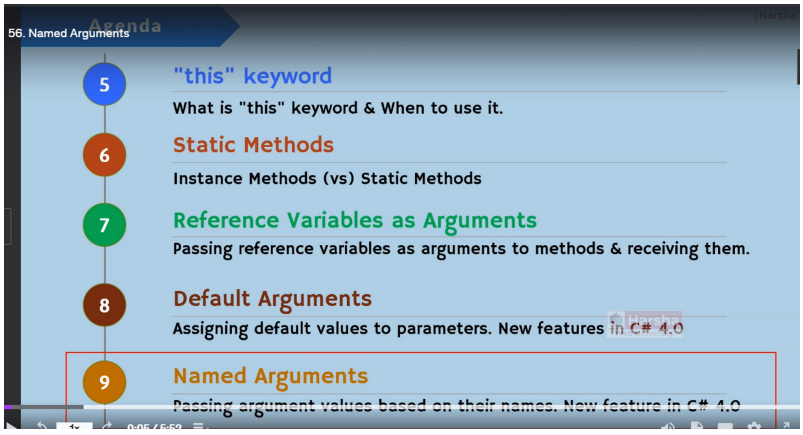

Instance Methods (vs) Static Methods
Next: Reference Variables as Arguments
Instance Methods
Static Methods
Can access (fields)
Can access both instance fields and static fields.
Can't access instance fields; but can access static fields only.
Can access (methods)
Can access both instance methods and static methods.
Can't access instance methods; but can access static methods only.
"this" keyword
Can use "this" keyword, as there must be "current object" to call instance method.
Can't use "this" keyword, as there is NO "current object" while calling instance methods.

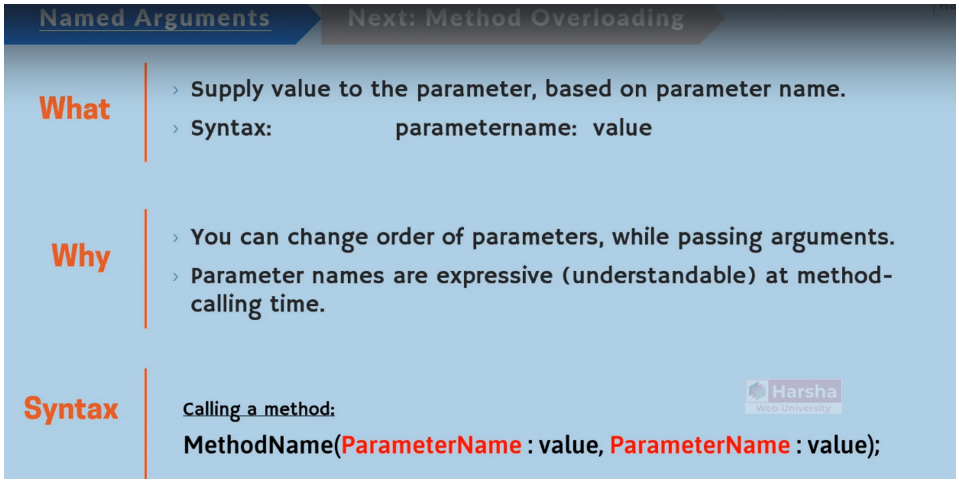

Agenda
5
"this" keyword
What is "this" keyword & When to use it.
6
Static Methods
Instance Methods (vs) Static Methods
7
Reference Variables as Arguments
Passing reference variables as arguments to methods & receiving them.
8
Default Arguments
Assigning default values to parameters. New features in C# 4.0
9
Named Arguments
Passing argument values based on their names. New feature in C# 4.0
0:09 / 7:45


Reference Variables as Arguments
Next: Default Arguments
Stack of Caller Method
Heap
Object
referenceVariable
Stack of Callee Method
parameter
If you pass "reference variable" as argument, the reference (address) of object will be passed to the method.
The parameter's data type will be the class name.
If you make any changes to object in the method, the same will be affected automatically in the caller method, as you are accessing the same object.

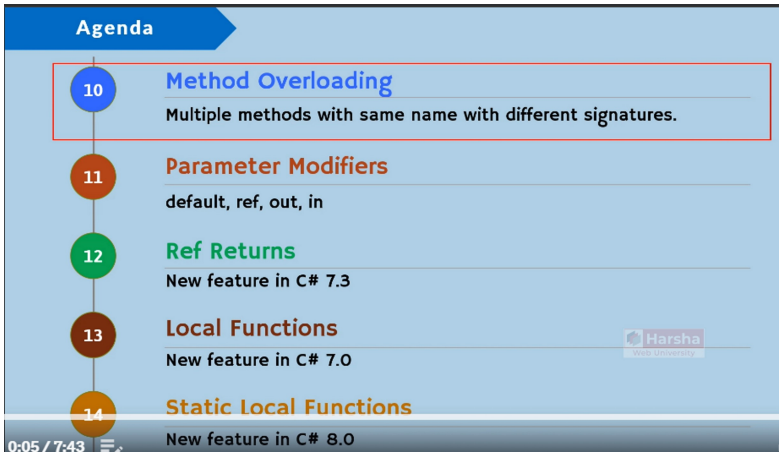

Agenda
Harsha
5
"this" keyword
What is "this" keyword & When to use it.
6
Static Methods
Instance Methods (vs) Static Methods
7
Reference Variables as Arguments
Passing reference variables as arguments to methods & receiving them.
8
Default Arguments
Assigning default values to parameters. New features in C# 4.0
9
Named Arguments
Passing argument values based on their names. New feature in C# 4.0

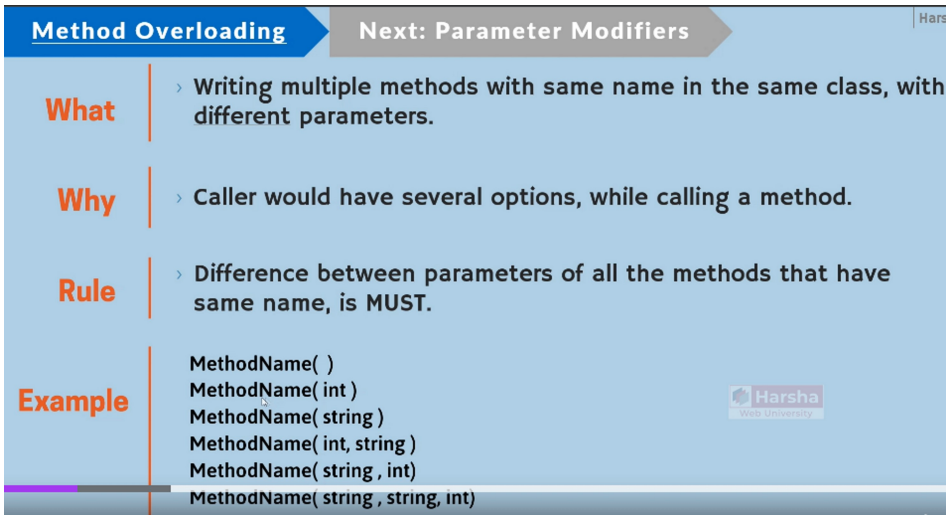

Default Arguments
Next: Named Arguments
What
Default value of the parameter of a method.
If you don't pass value to the parameter, the default value gets assigned to the parameter.
Why
To void bothering to pass value to the parameter; instead, take some default value into the parameter automatically, if the method caller has not supplied value to the parameter.
Syntax
accessModifier modifier returnType MethodName(parameter1 = defaultValue)
{
Method body here
}

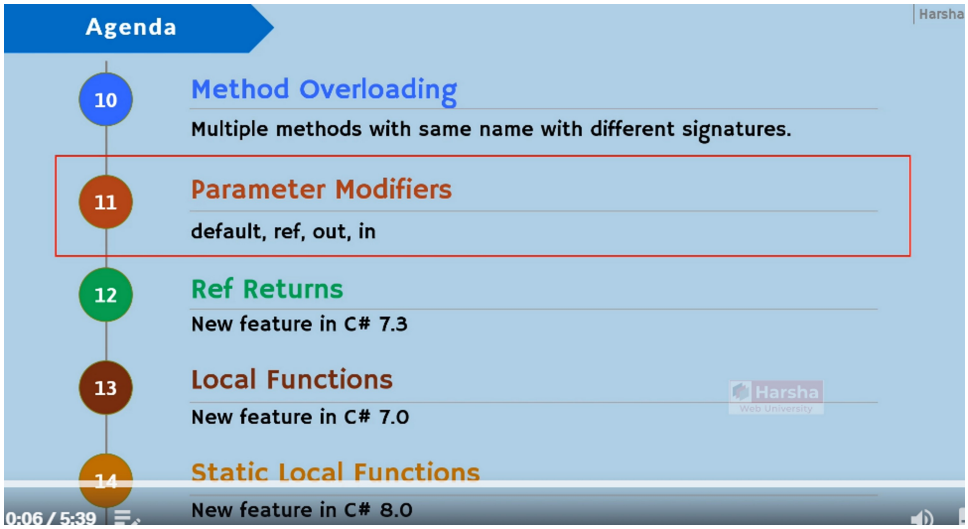

56. Named Arguments
Agenda
Harsha
5
"this" keyword
What is "this" keyword & When to use it.
6
Static Methods
Instance Methods (vs) Static Methods
7
Reference Variables as Arguments
Passing reference variables as arguments to methods & receiving them.
8
Default Arguments
Assigning default values to parameters. New features in C# 4.0
9
Named Arguments
Passing argument values based on their names. New feature in C# 4.0

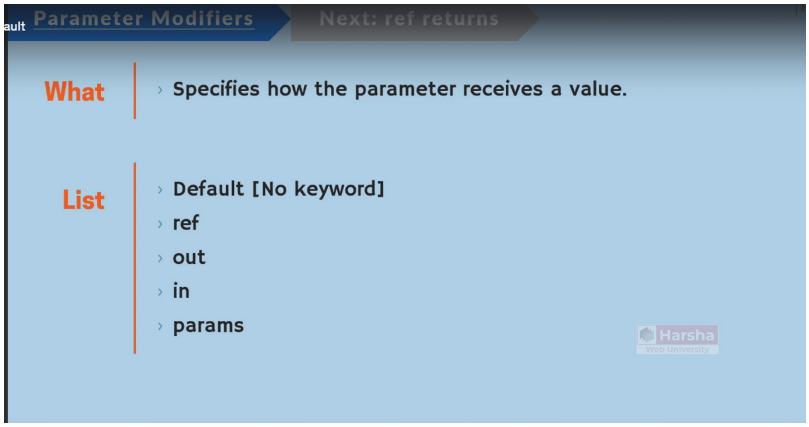


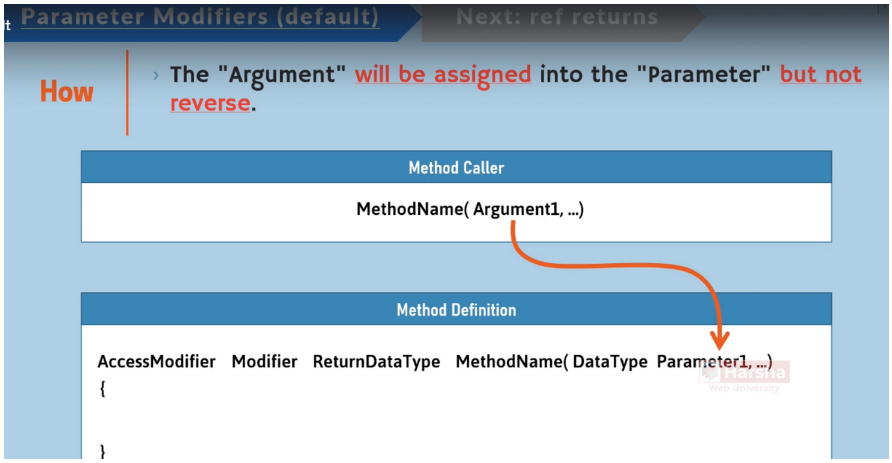
Useful if method has so many no. of parameters. That means more than 3

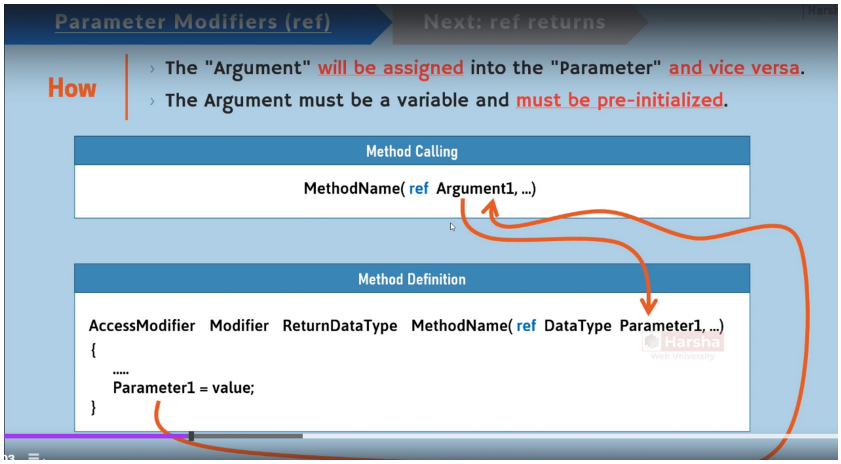


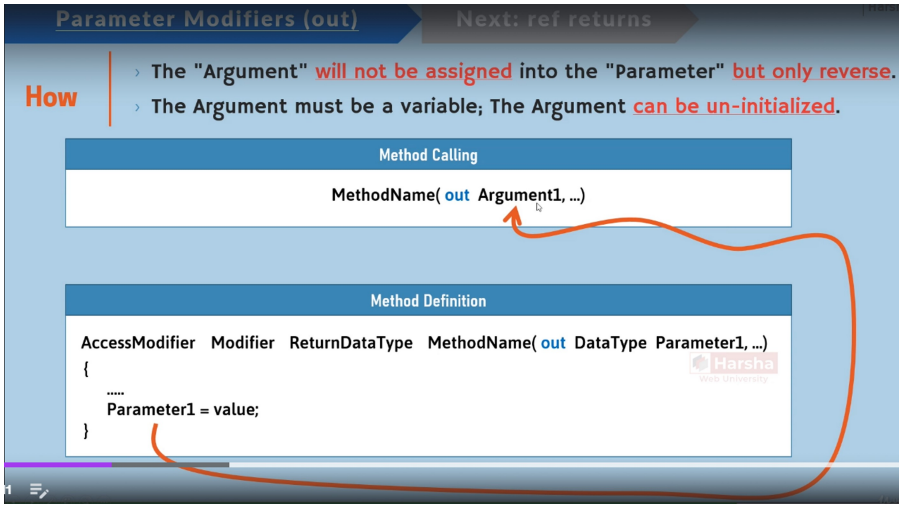



Agenda
10
Method Overloading
Multiple methods with same name with different signatures.
11
Parameter Modifiers
default, ref, out, in
12
Ref Returns
New feature in C# 7.3
13
Local Functions
New feature in C# 7.0
14
Static Local Functions
New feature in C# 8.0
Harsha
0:06 / 5:39

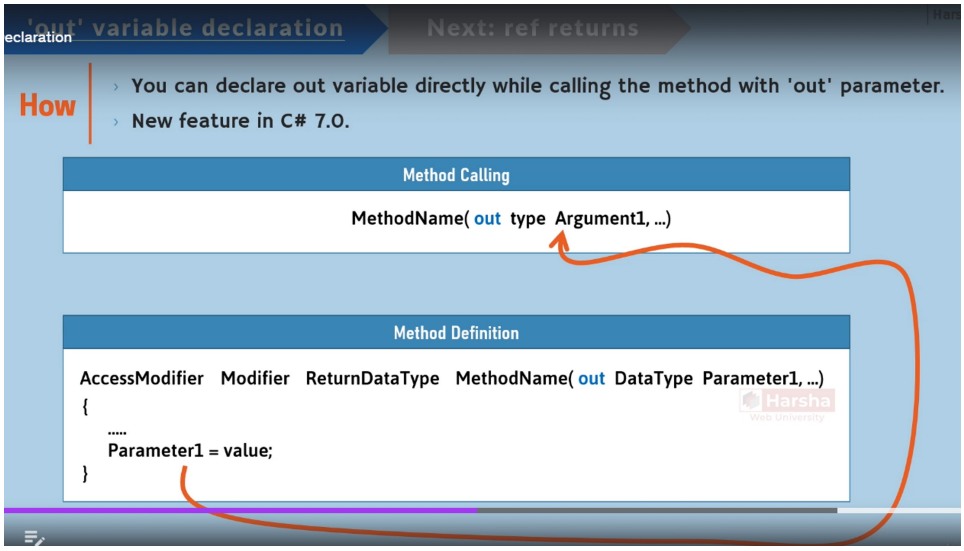

Parameter Modifiers
Next: ref returns
What
Specifies how the parameter receives a value.
List
Default [No keyword]
ref
out
in
params
Harsha

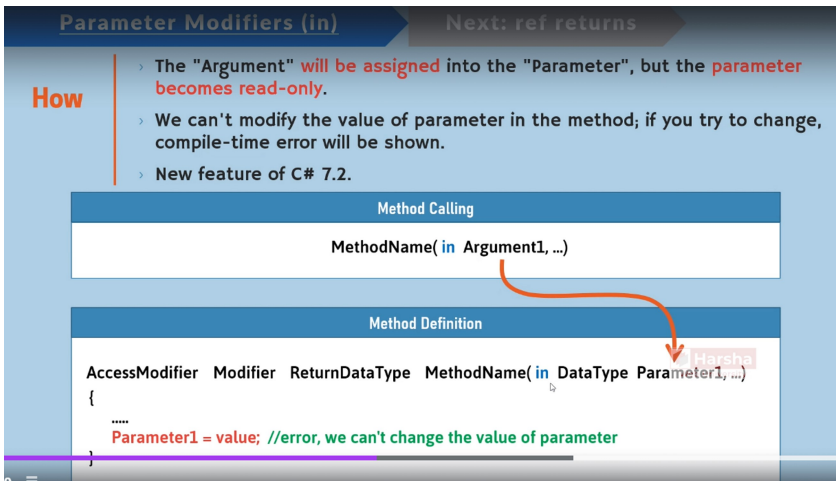

Parameter Modifiers (default)
Next: ref returns
How
The "Argument" will be assigned into the "Parameter" but not reverse.
Method Caller
MethodName(Argument1, ...)
Method Definition
AccessModifier Modifier ReturnDataType MethodName(DataType Parameter1, ...)
{
}

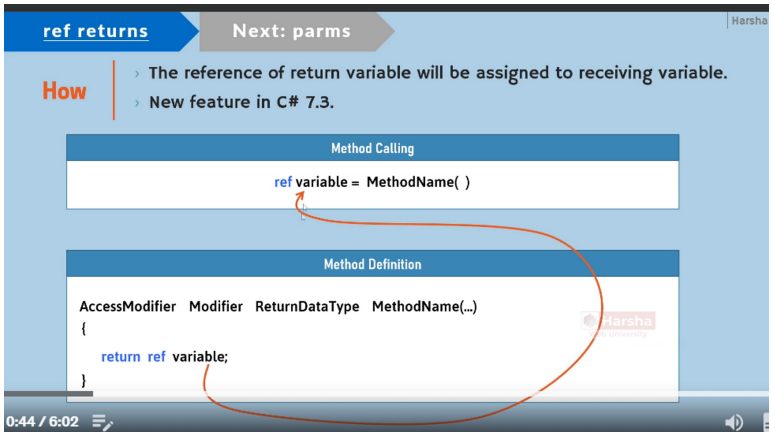

Parameter Modifiers (ref)
Next: ref returns
How
The "Argument" will be assigned into the "Parameter" and vice versa.
The Argument must be a variable and must be pre-initialized.
Method Calling
MethodName(ref Argument1, ...)
Method Definition
AccessModifier Modifier ReturnDataType MethodName(ref DataType Parameter1, ...)
{
.....
Parameter1 = value;
}
Harsha

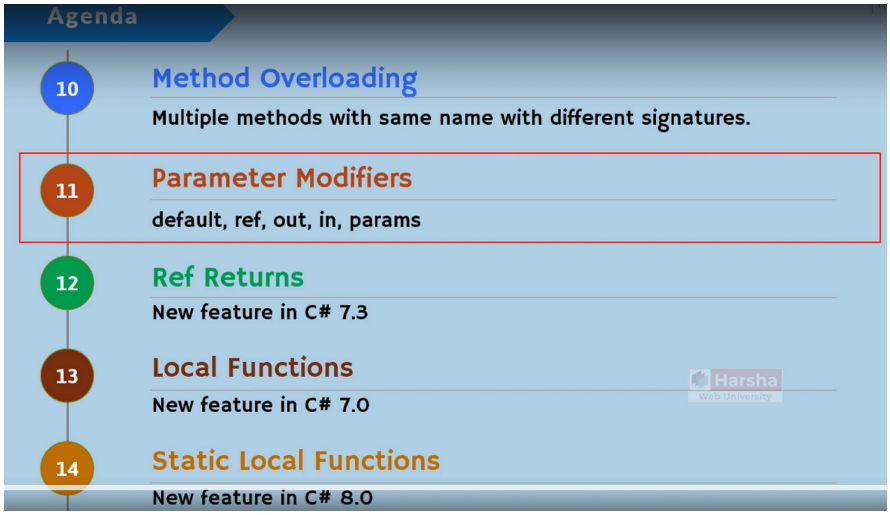


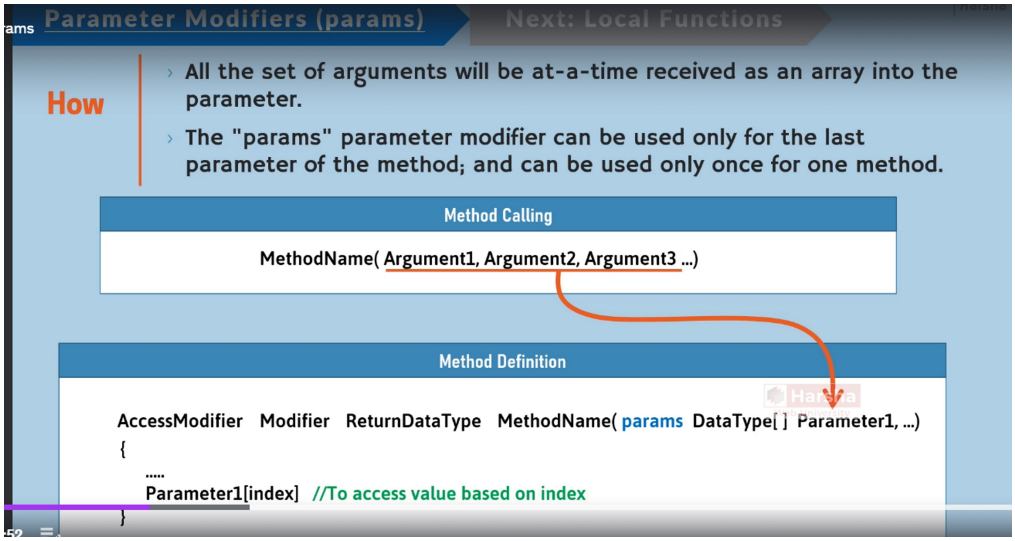
If you want to return multiple values from a function, use out parameters.

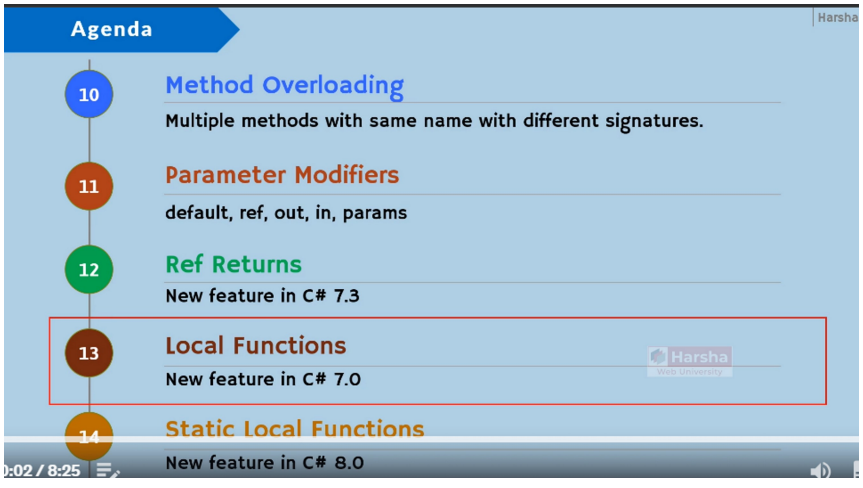


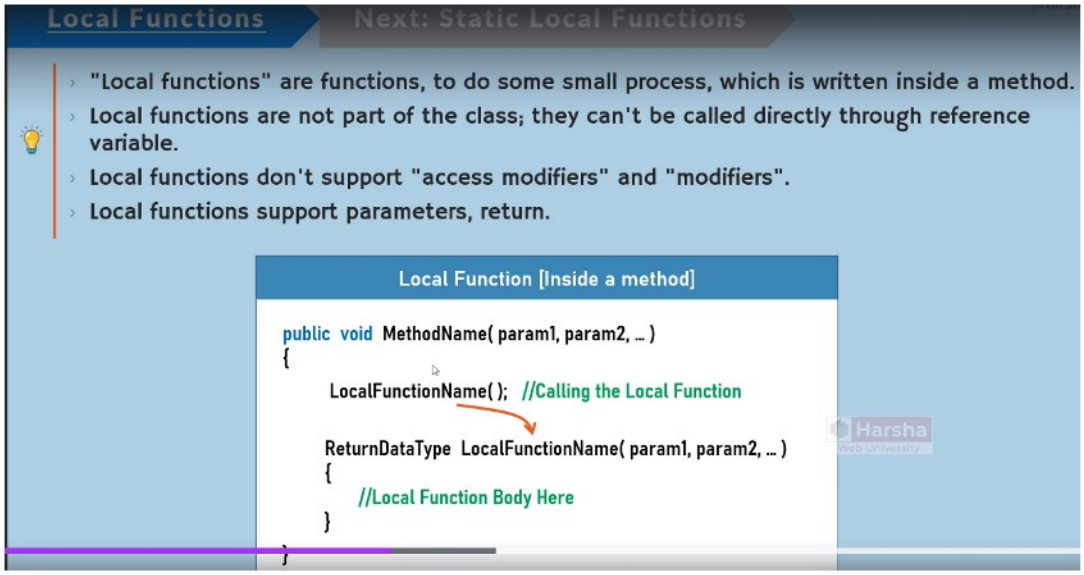


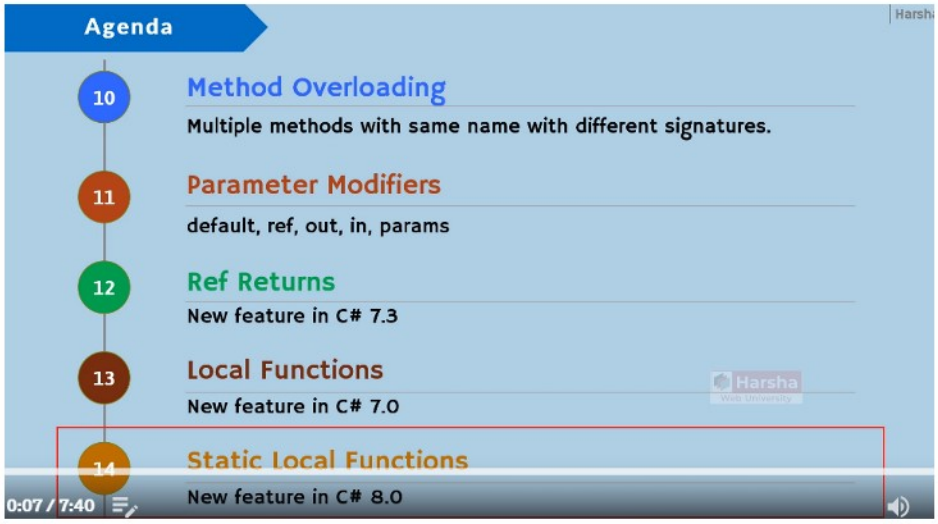

ref returns
Next: parms
How
The reference of return variable will be assigned to receiving variable.
New feature in C# 7.3.
Method Calling
ref variable = MethodName( )
Method Definition
AccessModifier Modifier ReturnDataType MethodName(...)
{
return ref variable;
}
0:44 / 6:02

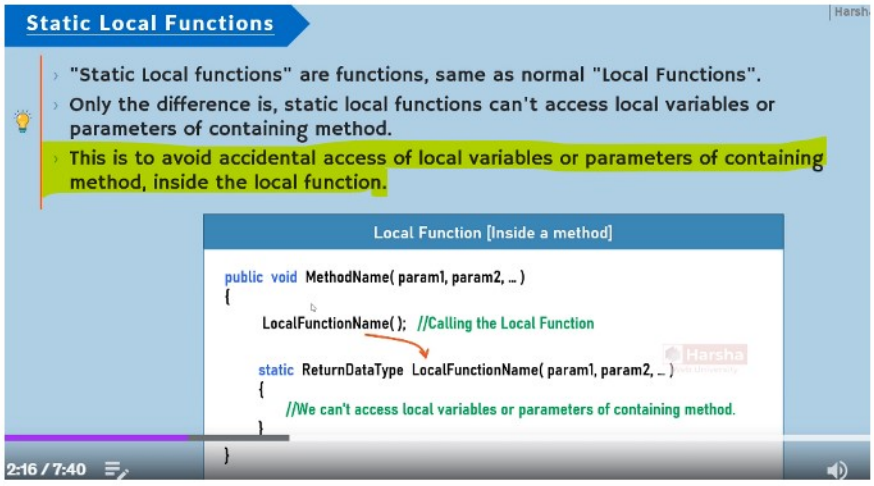

Agenda
10
Method Overloading
Multiple methods with same name with different signatures.
11
Parameter Modifiers
default, ref, out, in, params
12
Ref Returns
New feature in C# 7.3
13
Local Functions
New feature in C# 7.0
14
Static Local Functions
New feature in C# 8.0


Parameter Modifiers (params)
Next: Local Functions
How
All the set of arguments will be at-a-time received as an array into the parameter.
The "params" parameter modifier can be used only for the last parameter of the method; and can be used only once for one method.
Method Calling
MethodName(Argument1, Argument2, Argument3 ...)
Method Definition
AccessModifier Modifier ReturnDataType MethodName(params DataType[ ] Parameter1, ...)
{
.....
Parameter1[index] //To access value based on index
}


Agenda
10
Method Overloading
Multiple methods with same name with different signatures.
11
Parameter Modifiers
default, ref, out, in, params
12
Ref Returns
New feature in C# 7.3
13
Local Functions
New feature in C# 7.0
14
Static Local Functions
New feature in C# 8.0
0:02 / 8:25







It’s different than normal static method.

Recursion

- A method calls itself.
- Useful in mathematic computations, such as finding factorial of a number.

Recursive Method

```
public void MethodName( )
{
    if (condition)
    {
        MethodName( ); //Calling the same method
    }
}
```


Recursion

- A method calls itself.
- Useful in mathematic computations, such as finding factorial of a number.

Recursive Method

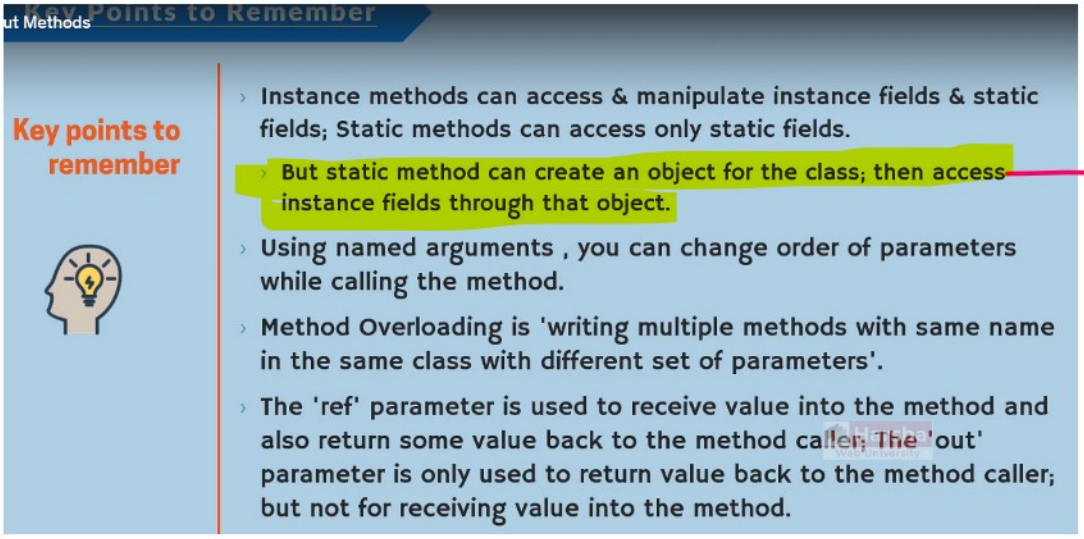
```
public void MethodName( )
{
    if (condition)
    {
        MethodName( ); //Calling the same method
    }
}
```

Harsha Web University

Key Points to Remember

Key points to remember

- Method is a part of class, that contains collection of statements to do some process.
- Access modifiers of Methods: private, protected, private protected, internal, protected internal, public.
- Modifiers of Methods: static, virtual, abstract, override, new, partial, sealed
- For each method call, a new stack will be created; all local variables and parameters of the method will be stored in that stack; will be deleted automatically at the end of method execution.
- In instance methods, the 'this' keyword refers to the current object, based on which the method is called'.




```
public class MyClass
{
   private int nonStaticField = 42;
   private static int staticField = 100;

   private void NonStaticMethod()
   {
      Console.WriteLine("Non-static method called.");
   }

   public static void StaticMethod()
   {
      MyClass instance = new MyClass(); // Creating an instance of the class
      Console.WriteLine(instance.nonStaticField); // Accessing the non-static field
      instance.NonStaticMethod(); // Calling the non-static method

      // Accessing static fields directly
      Console.WriteLine(staticField);
   }
}

class Program
{
   static void Main()
   {
      MyClass.StaticMethod(); // Calling the static method
   }
}
```