INFORMATION PROCESSING TECHNIQUES

Extension Methods

WEEK 10

MURTAZA MUNAWAR FAZAL

For instance, you might like to know whether a certain string was a number or not. The usual approach would be to define a function and then call it each time, and once you got a whole lot of those kind of functions, you would put them together in a utility class, like this:

```
public class MyUtils

{
    public static bool IsNumeric(string s)
    {
        float output;
        return float.TryParse(s, out output);

    }
}
```

```
string test = "4";
if (MyUtils.IsNumeric(test))
    Console.WriteLine("Yes");
else
    Console.WriteLine("No");
```

```
string test = "4";
if (test.IsNumeric())
    Console.WriteLine("Yes");
else
    Console.WriteLine("No");
```

Extension methods enable you to add methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type.

An extension method is a special kind of static method, but they are called as if they were instance methods on the extended type.

However, with Extension Methods, you can actually extend the String class to support this directly. You do it by defining a static class, with a set of static methods that will be your library of extension methods. Here is an example:

```
public static class MyExtensionMethods
{
   public static bool IsNumeric(this string s)
   {

      float output;
      return float.TryParse(s, out output);
   }
}
```

The only thing that separates this from any other static method, is the "this" keyword in the parameter section of the method. It tells the compiler that this is an extension method for the string class, and that's actually all you need to create an extension method.

- Extension methods allow existing classes to be extended without relying on inheritance or having to change the class's source code.
- If the class is sealed than there in no concept of extending its functionality. For this a new concept is introduced, in other words extension methods.
- Extension methods are additional custom methods which were originally not included with the class.
- Extension methods can be added to custom, .NET Framework or third-party classes, structs or interfaces.
- The first parameter of the extension method must be of the type for which the extension method is applicable, preceded by the this keyword.
- Extension methods can be used anywhere in the application by including the namespace of the extension method.

Important points for the use of extension methods

- An extension method must be defined in a top-level static class.
- An extension method with the same name and signature as an instance method will not be called.
- Extension methods cannot be used to override existing methods.
- The concept of extension methods cannot be applied to fields, properties or events.
- Overuse of extension methods is not a good style of programming.

INFORMATION PROCESSING TECHNIQUES

Named And Optional Arguments

WEEK 10

MURTAZA MUNAWAR FAZAL

- public void ExampleMethod(int required, string optionalstr, int optionalint)
- public void ExampleMethod(int required, string optionalstr = "default string", int optionalint = 10)

- ExampleMethod(1,”Hello”,2);
- ExampleMethod(1,”Hello”);
- ExampleMethod(1);
- ExampleMethod(1,””,3);

Named Arguments

- public void ExampleMethod(int required, string optionalstr = "default string", int optionalint = 10)

- ExampleMethod(1,optionalstr: ”Hello”, optionalint: 2);
- ExampleMethod(1, optionalstr: ”Hello”);
- ExampleMethod(required: 1);
- ExampleMethod(1, optionalint: 3);

INFORMATION PROCESSING TECHNIQUES

Ref v/s Out

WEEK 10

MURTAZA MUNAWAR FAZAL

Ref v/s Out

Ref	Out
The parameter or argument must be initialized first before it is passed to ref.	It is not compulsory to initialize a parameter or argument before it is passed to an out.
It is not required to assign or initialize the value of a parameter (which is passed by ref) before returning to the calling method.	A called method is required to assign or initialize a value of a parameter (which is passed to an out) before returning to the calling method.
Passing a parameter value by Ref is useful when the called method is also needed to modify the pass parameter.	Declaring a parameter to an out method is useful when multiple values need to be returned from a function or method.
When we use REF, data can be passed bi-directionally.	When we use OUT data is passed only in a unidirectional way (from the called method to the caller method).

Both ref and out are treated differently at run time and they are treated the same at compile time.

Properties are not variables, therefore it cannot be passed as an out or ref parameter.