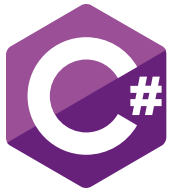
Methods, constructors and the static keyword

Agenda

- Methods
- Method overloading
- Named/Default argument parameters
- Pass parameters by value
- Pass parameters by reference
- Constructors
- static keyword
- this keyword

Methods



Methods, constructors and the static keyword

Methods

A method is a class member

A method is a block of code which contains one or more statements

- Has a name
- Has a set of parameters
- Returns a result of a known type
- Is defined in a class, struct or interface
- Has an access modifier

Arguments

The values/references passed to the method during the method call

Signature

The parameter combination defined by the method

Void method

```
public void VoidMethodWithoutParameters()
{
    // code
}
```

Void method

```
public void VoidMethodWithParameter(string stringParameter)
{
    // code
}
```

Method

```
public string MethodWithoutParameters()
{
    string result = string.Empty;
    // code, result = ...
    return result;
}
```

Method

```
public string MethodWithParameter(string stringParameter)
{
    string result = string.Empty;
    // use stringParameter
    // code, result = ...
    return result;
}
```

Method call

```
public void CallingMethod()
{
    VoidMethodWithoutParameters();
    VoidMethodWithParameters("word");
    string result1 = MethodWithoutParameters();
    string result2 = MethodWithParameters("word");
}
```

2 types

Instance methods

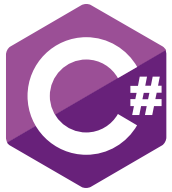
Static methods

Demo

Create methods

Instance method/static method

Method overloading



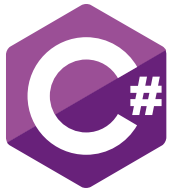
Method overloading

You can provide other methods with the same name with a different signature

Demo

Create method overloads

Optional parameters and named arguments



Named arguments

Possible to provide named arguments

Allows to specify an argument by matching the argument with its parameter name rather than with its position in the ordered parameters list

Optional parameters

Possible to define optional parameter

Each optional parameter has a default value

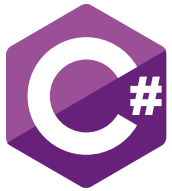
If no argument is passed the default value becomes the argument

Demo

Create methods with optional parameters

Use named arguments

Passing a value type parameter



Passing parameters

Parameters can be passed to a method :

- By value (default behavior) : the method receives the value of the argument
- By reference (ref and out keywords): the method receives the reference of the argument

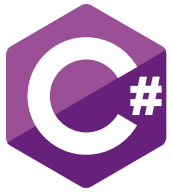
Passing value type parameter

By value : any change to the parameter that takes place inside the method has no effect on the argument variable

By reference : any change to the parameter that takes place inside the method changes the argument variable

Demo

Passing value type parameters



Passing a reference type parameter

Passing reference parameters

Any change to the parameter that takes place inside the method changes the argument variable

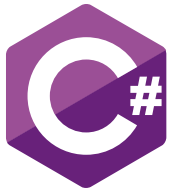
By value : reallocating a new object parameter inside the method has no effect on the argument variable

By reference : reallocating a new object parameter inside the method is also reallocated to the argument variable

Demo

Passing reference parameters

Constructor



Constructors

Called when a type is instantiated

It is where you initialize the type state

A type can have multiple constructors

A constructor can have zero, one or more parameters

Parameter-less constructors

- Constructor without parameters
- If no constructor is provided, a parameter-less constructor is created by the compiler
- Members are set to their default value
- Structures can’t contain an explicit parameter-less constructor

Parameter-less constructor

```
namespace MyNamespace
{
    public class MyClass
    {
        public MyClass()
        {
        }
    }
}
```

Constructors

The runtime calls the constructor to initialize the state of the class or structure

References of the type must be initialized to avoid `NullReferenceException`

Constructor with parameter

```
namespace MyNamespace
{
    public class MyClass
    {
        private string _field;
        public MyClass(string fieldValue)
        {
            _field = fieldValue;
        }
    }
}
```

readonly fields

Fields that can’t be changed

Can be assigned in the declaration or in the constructor

readonly

```
private readonly string _code = “ABC";
```

readonly fields

Immutable if value types

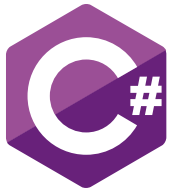
Prevents the field from being replaced if reference type

Demo

Create constructors

Create readonly fields

Static



Methods, constructors and the static keyword

Static classes

- A class that cannot be instantiated
- Members are reached through the class name
- Don’t have any constructor
- Cannot be inherited

Empty static class

```
namespace MyNamespace
{
    public static class MyStaticClass
    {
        // empty
    }
}
```

Static class

```
namespace MyNamespace
{
    public static class MyStaticClass
    {
        public static string Name { get; set; }
        public static void MyStaticMethod() {}
    }
}
```

Static members

Only one copy of a static member exists, independently of how many instances of the class are created

Static constructor

To initialize static members

Static class with static members

```
namespace MyNamespace
{
    public static class MyStaticClass
    {
        public static string Name { get; set; }
        public static void MyStaticMethod() {}
    }
}
```

Demo

Create static classes

Create static members

Summary

A method is a type member that has a signature (parameters) and a return type

You can change the visibility with access modifiers

2 types of methods : instance and static methods

You can overload a method (same name , different signature)

The values that are passed in during a method call are the arguments

A method can have named arguments and optional parameters

Summary

Parameters can be passed in by value or by reference

If you pass in a value or a reference type by reference, a change in the method changes the argument

If you pass in a reference type by reference, a reallocation in the method reallocates the argument

The constructor is a block of code that is called when a type is instantiated. To initialize the members. Can be static

A readonly field is a field that can be assigned in the constructor of the type or during its declaration

A type or a member can be static. Are independent from an instance

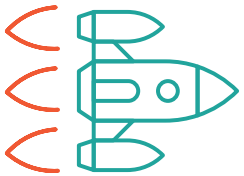
Challenge

A rocket is propelled by ejecting matter (fuel) at the rear of the rocket

An impulsion is when a determined quantity of fuel is ejected

Depending on the rocket an impulsion allows the rocket to cover a certain distance

2 Rockets will compete to determine the rocket that covers more distance with the same quantity of fuel



Power = P Fuel Per Impulsion = f Distance Per Impulsion = 100 x P = d Fuel Level = F Covered distance = 0 Number of Impulsion = 0	Power = P Fuel Per Impulsion = f Distance Per Impulsion = 100 x P = d Fuel Level = F – f Covered distance = d Number of Impulsion = 1	Power = P Fuel Per Impulsion = f Distance Per Impulsion = 100 x P = d Fuel Level = F – 3f Covered distance = 3d Number of Impulsion = 3	Power = P Fuel Per Impulsion = f Distance Per Impulsion = 100 x P = d Fuel Level = F – 2f Covered distance = 2d Number of Impulsion = 2	Power = P Fuel Per Impulsion = f Distance Per Impulsion = 100 x P = d Fuel Level = F – 4f Covered distance = 4d Number of Impulsion = 4

Requirements

Follow the requirements in the comments of the project