

Working with Methods



Gill Cleeren

CTO Xpirit Belgium

@gillcleeren | www.xpirit.com/gill



Agenda



Understanding methods

Passing data with parameters

More options with methods and parameters



Understanding Methods



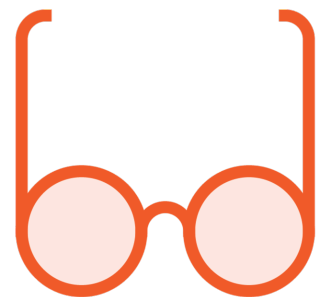
Methods in C#



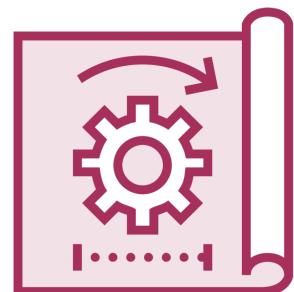
Code block



Receives parameters and (optionally) returns value



Readable code and code reuse



Declared within a class or struct



C# Method Syntax

```
<access modifier> <return type> Method_Name (Parameters)
{
    //method statements
}
```



```
public int AddTwoNumbers()  
{  
  
  
}
```

Looking at a First Method

```
public int AddTwoNumbers(int a, int b)
{

}
```

Adding Method Parameters

```
public int AddTwoNumbers(int a, int b)
{
    return a + b;
}
```

Returning a Value

Return type must be specified


```
public int AddTwoNumbers(int a, int b)
{
    if (a > b)
    {
        return a + b;
    }
    //no value returned if we get here → compile time error
}
```

Returning a Value

Value must be returned for all possible execution paths

```
public void DisplaySum(int a, int b)
{
    int sum = a + b;
    Console.WriteLine("The sum is " + sum);
}
```

A Method without Return Value

```
static void Main(string[] args)
{
    DisplaySum(3, 52);
}
```

Invoking a Method from main()

We can pass arguments: values for the parameter(s)

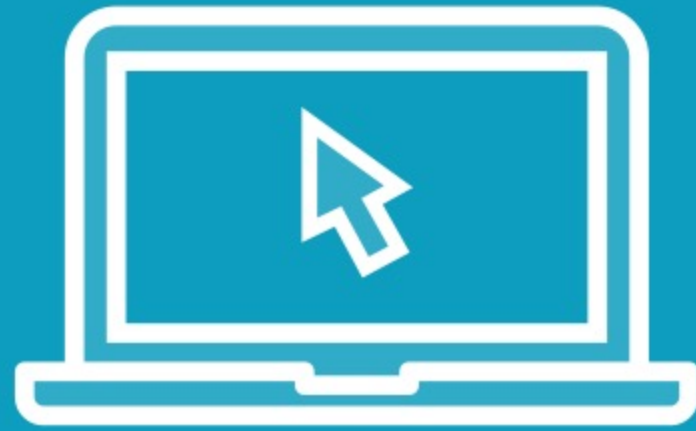
Note the use of static here

```
static void Main(string[] args)
{
    DisplaySum(3, 52);
    int result = AddTwoNumbers(55, 44);
}
```

Capturing a Return Value

Only possible if method isn't returning void

Demo



Creating a method

Adding parameters

Returning a value

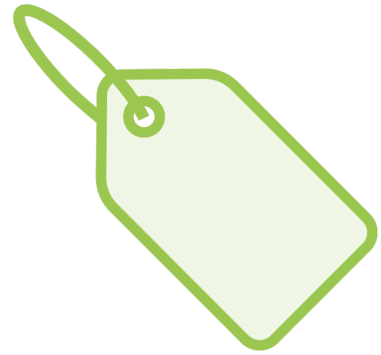
Invoking the method from main



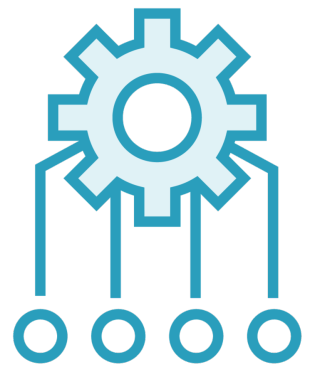
Passing Data with Parameters



Calling the Correct Method



Method name



Parameter types and arguments



Number of parameters



Matching the Parameters

Method overloading

Main

```
static void Main(string[] args)
{
    DisplaySum(3, 52);
}
```

Rest of Program.cs

```
public static void DisplaySum
    (int a, int b)
```

```
{ ... }
```

```
public static void DisplaySum
    (int a, int b, int c)
```

```
{ ... }
```


Demo



Using method overloading



Passing Parameters

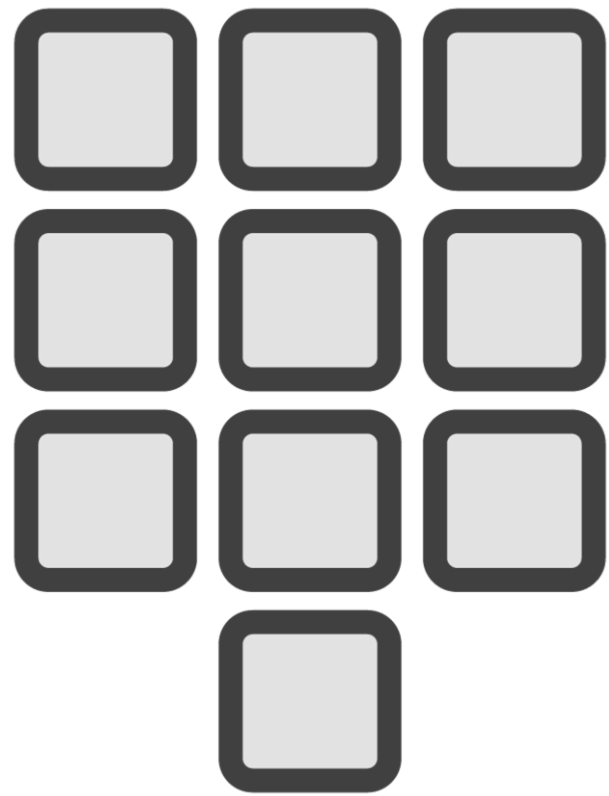
By value

Default if nothing else is specified

By reference

Require use of ref keyword on parameters





Passing parameters by value

- Default way of passing parameters
- A copy is created for the method
- Value in caller stays the same



Passing Parameters by Value

```
int a = 33;  
int b = 44;
```

```
AddTwoNumbers(a, b);  
    33  44
```

```
public int AddTwoNumbers(int a, int b)  
{  
    b += 10;  
    int sum = a + b;  
    return sum;  
}
```



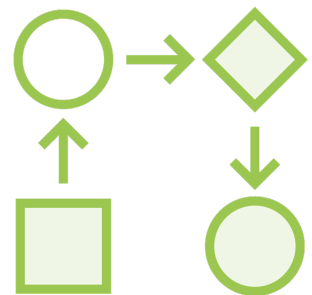
Passing Parameters by Reference



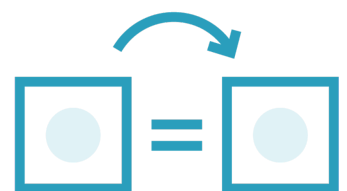
A reference to the value is passed



No copy is made



Changes made in method affect original values



ref keyword is used

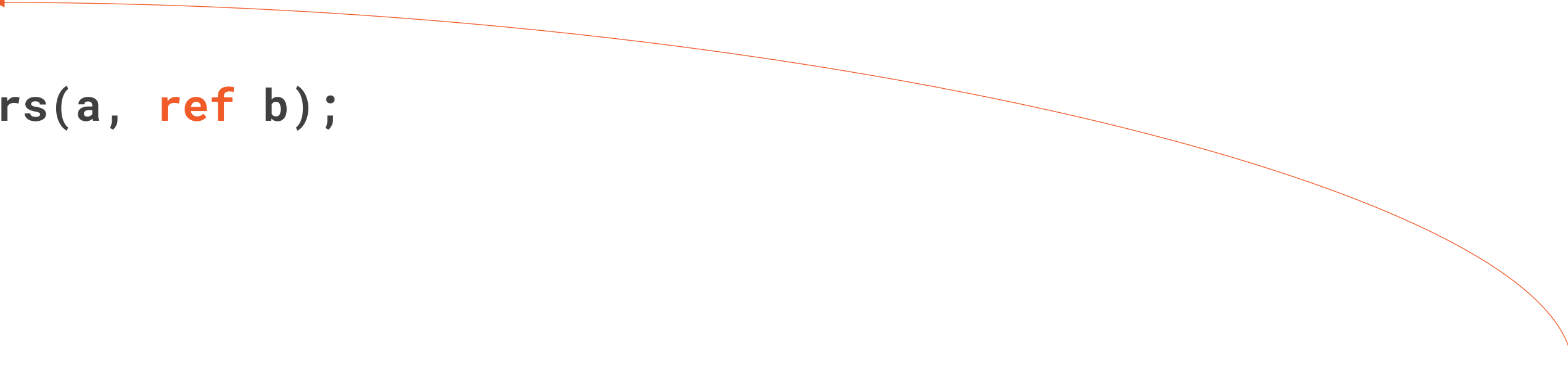


Passing Parameters by Value

```
int a = 33;  
int b = 44;
```

```
AddTwoNumbers(a, ref b);
```

```
public int AddTwoNumbers(int a, ref int b)  
{  
    b += 10;  
    int sum = a + b;  
    return sum;  
}
```



Demo



Passing parameters by value

Using ref to pass parameters by reference



```
public static int AddTwoNumbers(int a, out int b, out int c)
{
    b = 10;

    int sum = a + b;

    c = sum / 10;

    return sum;
}
```

Using the out Keyword

Out values don't need to be initialized

Multiple values can be returned

Demo



Using the out keyword



More Options with Methods and Parameters



Doing More with Methods

Using params

Optional parameters

Named arguments

Expression-bodied syntax



```
public int AddNumbers(params int[] values)
{
    for (int i = 0; i < values.Length; i++)
    {
        //Add values to calculate total
    }
}
```

Using the params Keyword

Represents array to capture multiple parameters

Are available through array in the method body

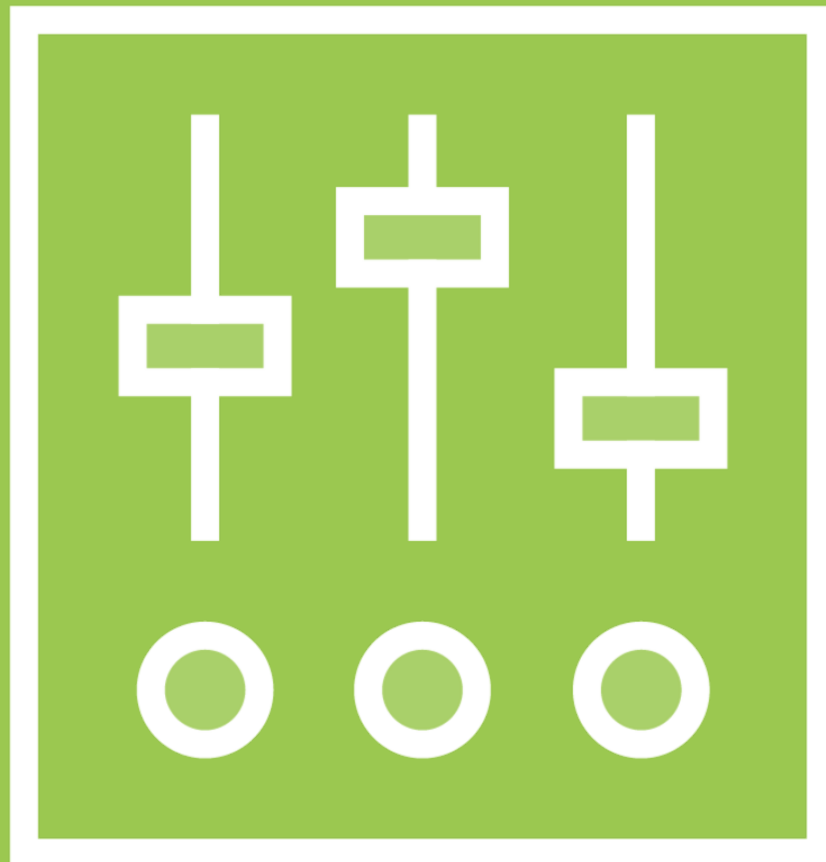
Demo



Adding a params parameter

Invoking a method with more parameters





Optional Parameters

Specify default value for one or more parameters

Caller can omit the optional ones



Working with Optional Parameters

Method with optional parameters

```
public int AddNumbers  
    (int a,  int b, int c = 100)  
{  
    int sum = a + b + c;  
    return sum;  
}
```

Calling the method

```
AddNumbers(10, 20); //no third parameter  
AddNumbers(10, 20, 30);
```



Named arguments

Not required to follow order of parameters

One or more parameters can have a name defined when invoking the method



Working with Named Arguments

Method with parameters

```
public static int AddNumbers  
    (int a, int b)  
{  
    ...  
}
```

Using named arguments

```
AddNumbers(b: 10, a: 20);
```

Demo



Using optional parameters

Working with named arguments



Demo



Using expression-bodied syntax



Summary



Methods are used to bring in reuse of code

Different options exist to pass parameters

- By value**
- By reference**



 **Up next:**
Understanding reference
types

