Object Oriented Programming

Lecture 2 Methods

Arsalan Rahman Mirza

Computer Science Department

Faculty of Science

2022-2023

Topics

- Methods
- Return Type
- Void Method

Methods

- Methods in C# are portions of a larger program that perform specific tasks. They can be used to keep code clean by separating it into separate pieces. They can also be used in more than one place allowing you to reuse previous code.
- In C# methods can be in the main program or can be in libraries, which are external files, containing classes and subroutines which can be imported into a program. This allows them to be distributed easily and used by multiple programs.

Methods

- Most computer programs that solve real-world problems are much larger than programs that you have written yet!
- Experience has shown that the best way to develop and maintain a large program is to construct it from small, simple pieces, or *modules.*
- Methods are the building blocks of C# and the place where all program activity occurs.
- Divide and conquer technique
  - Construct a large program from small, simple pieces (e.g., components)

C# method definition

- A method is a code block containing a series of statements. Methods must be declared within a class or a structure. It is a good programming practice that methods do only one specific task. Methods bring modularity to programs. Proper use of methods bring the following advantages:

1. Reducing duplication of code
2. Decomposing complex problems into simpler pieces
3. Improving clarity of the code
4. Reuse of code
5. Information hiding

C# method characteristics

Basic characteristics of methods are:

- Access level
- Return value type
- Method name
- Method parameters
- Parentheses
- Block of statements

Example

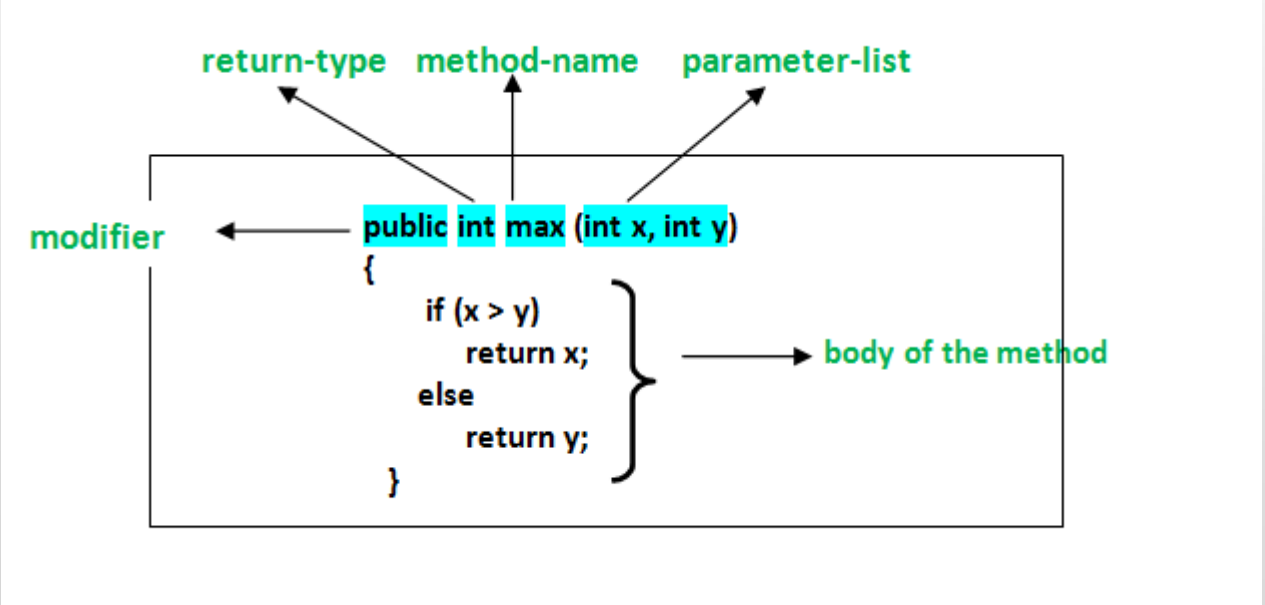
```
// Method Name --> GetCircleArea()
// Return Type ---> double
static double GetCircleArea(double radius)
{
   const float pi = 3.14F;
   double area = pi * radius * radius;   return area;
}
```

Method Declaration

- Method declaration means the way to construct method including its naming.

Syntax :

```
<Access_Modifier> <return_type> <method_name>([<param_list>])
```



Methods

- Methods in C# are defined like this:

```
[Modifiers (E.G public or static)][Type of output] [Name] ( [parameter 1],[parameter 2] ...)
{
}
```

Example

```
public static int Multiply(int a, int b)
{
    return a * b;
}
```

- This method has been passed two parameters, integer a and integer b, this is how you provide input for a subroutine (method).
- The return statement stops the subroutine and (depending on the output type) can output a value of the same type as the output as above.

Methods

```
//int is the output type
int Foo()
{
    //So you return an integer
    return 0;
}
```

- That is how you can receive output from a subroutine. But in some cases you do not need to return a value, for instance:

```
void Foo()
{
    //the output type is void, so you don't return a value
    return;
}
```

Methods

- In this case the return statement simply stops the subroutine and does not give any output, so if you assigned a variable to the output of a void, the variable's value would be null.

```
public static int Main(string[] args)
{
    // a = null
    int a = Foo();
}
void Foo()
{
    return;
}
```

Method Definitions

- The general form of a function is

  ret-type method-name(parameter list)

  {

  body of the function

  }

- The *ret-type specifies the type of data that the method returns.*
- A method may return any type of data except an array.
- The *parameter list is a comma-separated list of variable* names and their associated types that receive the values of the arguments when the function is called.
- **Methods must be declared within a class or a structure.**

Method Definitions

- A C# method consists of two parts
  - The method header, and
  - The method body
- The method header has the following syntax

  <return value> <name> (<parameter list>)
- The method body is simply a C# code enclosed between { }

Basic characteristics of methods are:

- Return value type
  - output: data type or void
- Method name
  - Any legal character can be used in the name of a method.
  - method names begin with an uppercase letter.
  - The method names are verbs or verbs followed by adjectives or nouns.
- Method parameters
  - inputs for the method
  - Empty parentheses indicate that the method requires no parameters.
- Parentheses
- Block of statements
  - The method block is surrounded with { } characters.
- Access level
  - who can call the method

```
using System;

public class Base
{
    public void ShowInfo()
    {
        Console.WriteLine("This is Base class");
    }
}

namespace SimpleMethod
{
    class Program
    {
        static void Main(string[] args)
        {
            Base ob = new Base();
            ob.ShowInfo();
        }
    }
}
```

Methods must be declared inside class

Inside class every method should be public or private (later)

This method return nothing (void)

Define an object for calling the method

Calling the method

Example 1 (Calling Method)

```
using System;
namespace ConsoleApplication1
{
        class method_test
        {
                static int Sum(int x, int y)
                {
                                int a = x;
                                int b = y;
                                int result = a + b;
                        return result;
                }
                static void Main(string[] args)
                {
                        int a = 12;
                        int b = 23;
                        int c = Sum(a, b);
                        // Display Result
                        Console.WriteLine("The Value of the sum is " + c);
                }
        }
}
```

Output :
The Value of the sum is 35

Example 2 (Calling method Without Parameters & Without Return Type)

```
using System;
namespace ConsoleApplication2
{
    class method_test
    {
        static void PrintSentence()
        {
            Console.WriteLine("No parameters and return type void");
        }
        // Main Method
        static void Main(string[] args)
        {
            // Method Invoking or Method calling
                PrintSentence();
        }
    }
}
```

Output :
No parameters and return type void

Example 3 (calling method Without Parameters & With Return Value Type)

```
using System;
namespace ConsoleApplication3
{
    class method_test
    {
        static int sum()
        {
            int a = 78, b = 70, add;
            add = a + b;
            return add;
        }
        static void Main(string[] args)
        {
            int getresult = sum();
            Console.WriteLine(getresult);
        }
    }
}
```

Output :

148

Example 4 (calling method With Parameters & Without Return Value Type)

```
using System;
namespace ConsoleApplication4
{
	class method_test
	{
		static void perimeter(int p)
		{
			Console.WriteLine("Perimeter of the Square is " + 4 * p);
		}
		static void Main(string[] args)
		{
				int p = 5;
			perimeter(p);
		}
	}
}
```

Output :
Perimeter of the Square is 20

Example 5 (calling method With Parameters & With Return Value Type)

```
using System;
namespace ConsoleApplication5
 {
        class method_test
        {
                static int factorial(int n)
                {
                        int f = 1;
                        for (int i = 1; i<= n; i++)
                        {
                                f = f * i;
                        }
                        return f;
                }
                static void Main(string[] args)
                {
                        int p = 4;
                        Console.WriteLine("Factorial is : " + factorial(p));
                }
        }
}
```

Output :
Factorial is : 24

Class activity 1

- Write a program by creating a method void with name print_text and 1 string as a parameter. Then in the method print the string. Call it in the main program.
- Try to use a new method to return a value (no void). Call it and use it in the main program
- Write a program to compute the tax from income. If income less than 100 no tax otherwise compute tax as tax = 0.5 * (income - 1000). Create 3 method (computeTaxt) (getIncome) (printTax) and finally test your program.