Methods

In this lesson, you will get to know the role of methods in classes.

WE'LL COVER THE FOLLOWING

- The Purpose of Methods
- Definition and Implementation
- Method Parameters, Return Type
- Return Statement
- Method Overloading
- Advantages of Method Overloading

The Purpose of Methods

In general, methods act as an interface between a program and the fields of a class in that program.

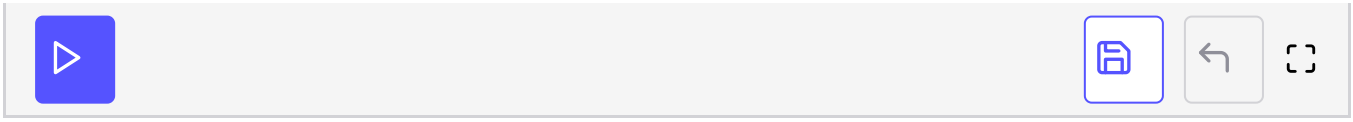
These methods can either alter the data stored in the fields or use their existing values to perform a certain computation. All the useful methods should be `public`, although, some methods which do not need to be accessed from the outside could be kept private.

Definition and Implementation

> A method is a group of statements that performs some operations and may or may not **return** a result.

Here is an example of a `Buy()` method implemented inside the `VendingMachine` class. The method is being invoked in the `Main()` method, inside the `Demo` class:

```
class VendingMachine {


  // Public method implementation to print print the bought product
  public void Buy(string product) {
    Console.WriteLine("You bought: " + product);
  }

}

class Demo {

  public static void Main(string[] args) {
    var vendingMachine = new VendingMachine();
    vendingMachine.Buy("Chocolate"); // Calling public method
  }

}
```



Method Parameters, Return Type

Method parameters make it possible to pass values to a method and **return type** makes it possible to get a result returned by the called method. The parameters are declared inside the parentheses after the method name while the `return` type is declared before method name.

Return Statement

For methods that define a return type, the **return statement** must be immediately followed by the return value.

```
// public method which returns the manufacturer
string manufacturer = "Vendy Inc.";

public string GetManufacturer() {

  return manufacturer; // return statement
}
```

The above method returns the name of the manufacturer.

The return type, `string`, which comes before the method name **GetManufacturer**, indicates that this method returns a `string`.

Method Overloading

> Overloading refers to making a method perform different operations based on the nature and type of its arguments.

Methods can be overloaded in C#.

We could redefine a method several times with the same name but with a different *number of arguments* and/or *types*. When the method is called, the appropriate definition will be selected by the compiler at the compile time.

Let's see this in action by overloading the `product` method in the Calculator class:

```
class Calculator {

  public double Product(double x, double y) {
    return x * y;
  }

  // Overloading the function to handle three arguments
  public double Product(double x, double y, double z) {
    return x * y * z;
  }

  // Overloading the function to handle int
  public int Product(int x, int y){
    return x * y;
  }

}

class Demo {

  public static void  Main(string[] args) {
    Calculator calculator = new Calculator();

    double x = 10;
    double y = 20;
    double z = 5;

    int a = 12;
    int b = 4;

    Console.WriteLine(calculator.Product(x, y));
    Console.WriteLine(calculator.Product(x, y, z));
    Console.WriteLine(calculator.Product(a, b));
  }

}
```

In the code above, we see the same method behaving differently when encountering different number and types of arguments.

> **Note**: Methods that have no arguments and differ only in the *return types* cannot be overloaded since the compiler won't be able to differentiate between their calls.

Advantages of Method Overloading

One might wonder why we wouldn't simply create new methods to perform different jobs rather than overloading the same method.

- An obvious benefit is that the code becomes simple and clean.
- We don't have to keep track of different method names.

Polymorphism is a very important concept in object-oriented programming. It will come up later on in the course, but method overloading plays a vital role in its implementation.

In the next lesson, you will get to know about static and non-static methods.