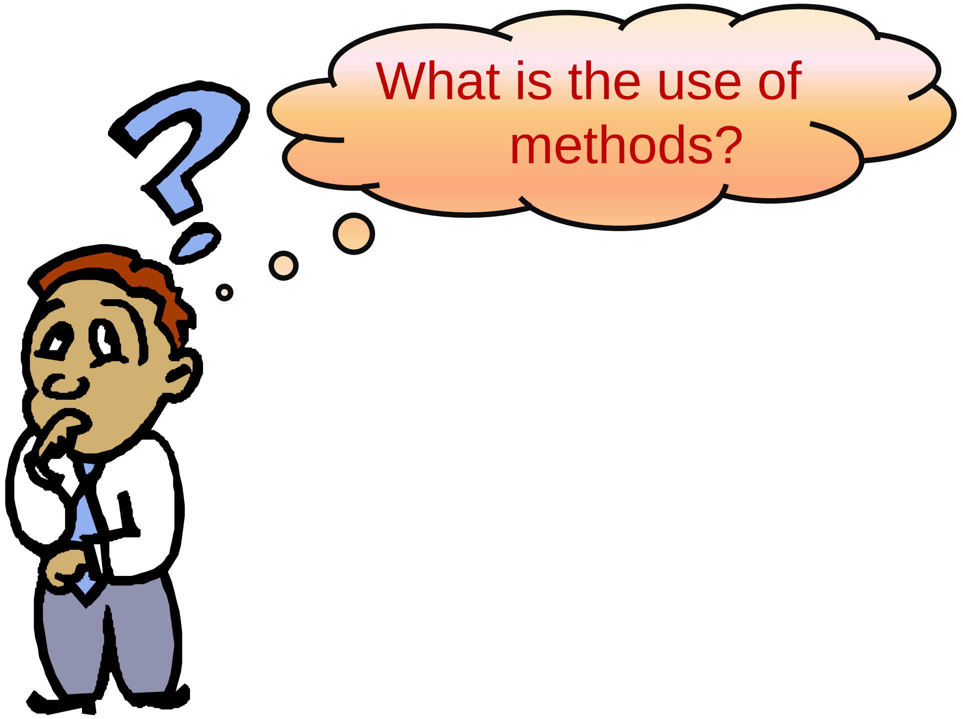
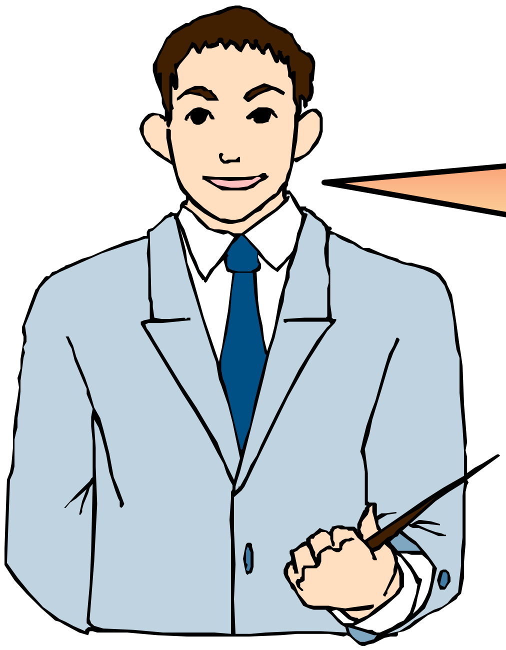
- In this session, you will learn to:
  - Use methods

What are methods?

- A method is a set of one or more program statements, which can be executed by calling the method name.





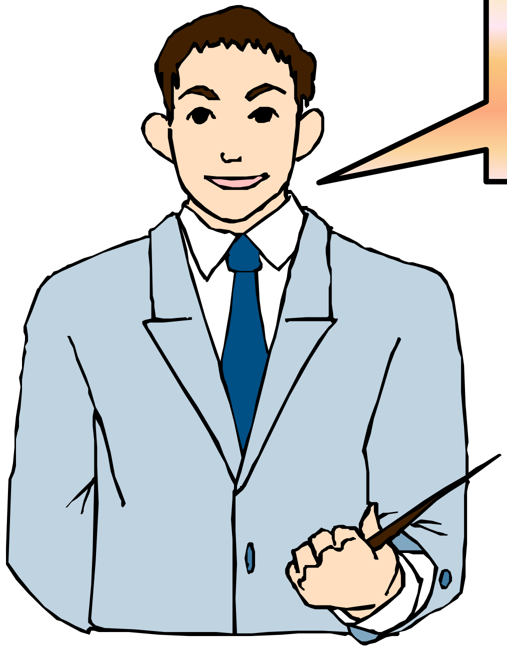
Let us discuss how methods can be useful in object-oriented programming.

- Methods:
  - Are useful to perform repetitive tasks, such as getting specific records and text.
  - Allow you to divide an application into logical units, which makes the application easy to read and easy to understand.
- To use methods, you need to:
  1. Define methods
  2. Call methods

- Defining a method means declaring the elements of its structure.
- The following syntax can be used to define a method:

```
<Access Specifier> <Return Type> <Method
Name>(Parameter List)
{
    Method body
}
```

- The building blocks of a method are:
  - Access specifier
  - Return type
  - Method name
  - Parameter list
  - Method body



Let us understand each of the building blocks of a method.

- Consider the following highlighted syntax for defining an access specifier:

```
<Access Specifier> <Return Type>
  <Method Name>(Parameter List)
  {
    Method body
  }
```

This determines the extent to which a variable or method can be accessed from another class.

- Consider the following highlighted syntax for defining a return type :

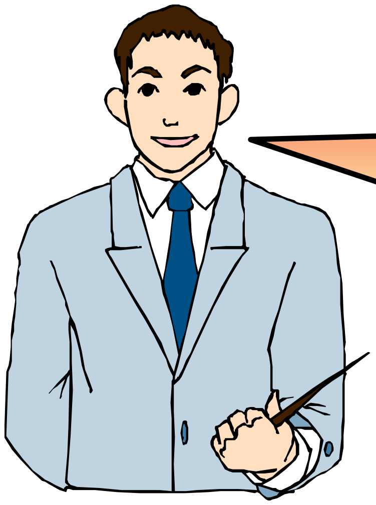
```
<Access Specifier> <Return Type>
  <Method Name>(Parameter List)
  {
   Method body
  }
```

A method can return a value of any type. If the method is not returning any value, use `void` as the return type.

- Consider the following highlighted syntax for defining a method name :

```
<Access specifier> <Return Type>
  <Method Name>(Parameter List)
  {
   Method body
  }
```

This is a unique identifier and is case-sensitive.



The method name cannot be the same as the variable name or any other item declared in the class.

- Consider the following highlighted syntax for defining a parameter list :

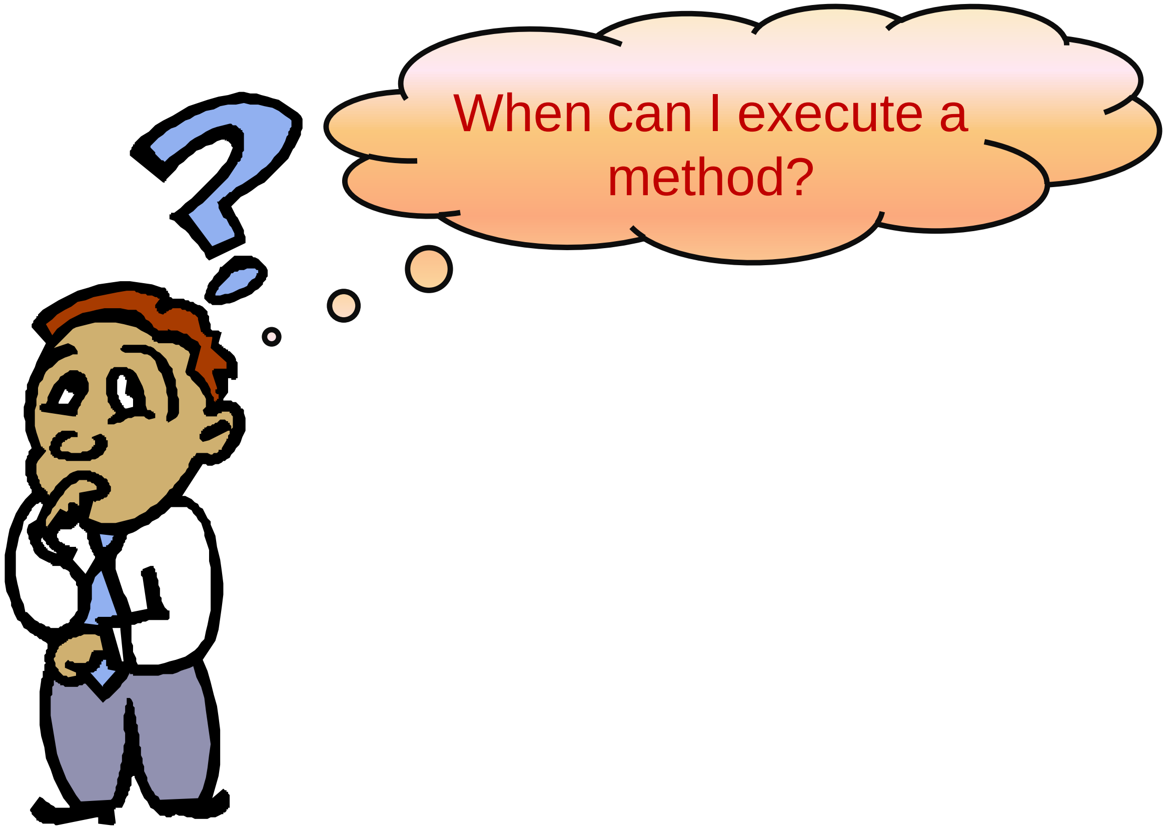
```
<Access Specifier> <Return Type>
   <Method Name>(Parameter List)
   {
    Method body
   }
```

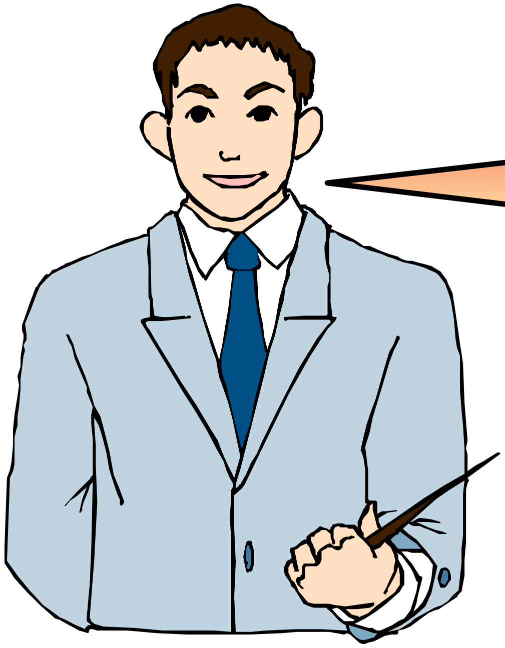
This is used to pass and receive the data from a method. It is enclosed between parentheses. The parentheses are included even if there are no parameters.

- Consider the following highlighted syntax for defining a method body :

```
<Access Specifier> <Return Type>
  <Method Name>(Parameter List)
  {
   Method body
  }
```

This contains the set of instructions that perform a specific task.


When can I execute a method?



After defining, you can execute the method by calling it.
Let us see how.

AlmaMate

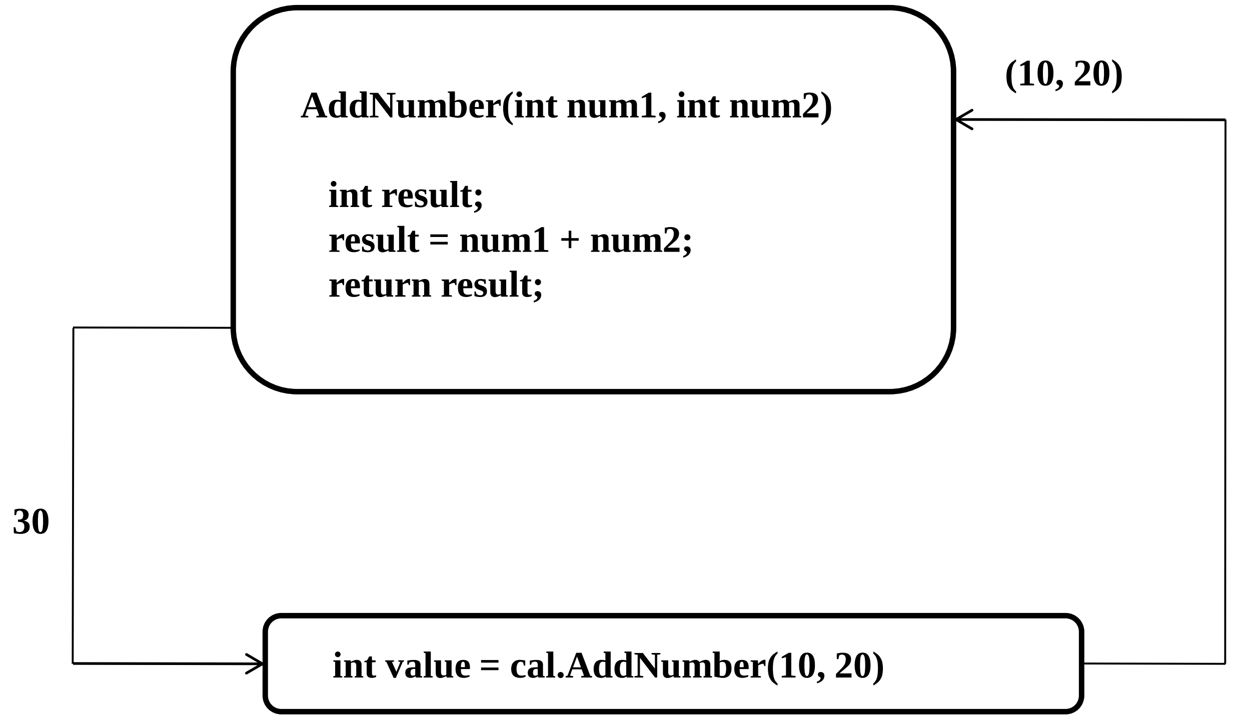
- You can call a method by using the name of the method.
- The method name is followed by parentheses even if the method call has no parameters, as shown in the following statement:

```
MethodName();
```

- The following code snippet shows how to call a method:

```
using System;
class Calculator
{
  public int AddNumber(int num1, int num2)
    {
      int result;
      result = num1 + num2;
      return result;
    }
static void Main(string[] args)
{
  Calculator cal = new
  Calculator();
  // The following statement is
  //calling the AddNumber method
  //and passing 10 and 20 as the
  //parameter list.
  int value=cal.AddNumber(10, 20);
  Console.WriteLine("The result is
                   {0}", value);
  Console.ReadLine();
}
}
```

- The following figure shows the process of calling a method as depicted in the preceding example.



```
AddNumber(int num1, int num2)

  int result;
  result = num1 + num2;
  return result;
```

```
int value = cal.AddNumber(10, 20)
```

- In this session, you learned that:
  - A method is a set of one or more program statements that can be executed by referring to the method name.
  - Defining a method means declaring the elements of its structure.
  - The access modifiers that can be used with methods are `public`, `private`, `protected`, `internal`, and `protected internal`.

- Solve the following exercises from the Object Oriented Programming Using C# - I book in the Machine Room:
  - Chapter 5: Exercise 1
  - Chapter 1: Exercise 1
  - Chapter 1: Exercise 2
  - Chapter 2: Exercise 1
  - Chapter 2: Exercise 2
  - Chapter 2: Exercise 4
  - Chapter 2: Exercise 5
  - Chapter 3: Exercise 6
  - Chapter 4: Exercise 1
  - Chapter 4: Exercise 2