

Lab 06: Using Methods

We have seen some examples of method in the earlier lab exercises. In this exercise, we are to create an application that uses similar method with added features, i.e. parameters and return of value.

We often create a method to group together codes that are performing a specific task or sequence. This method can be enhanced with parameters and returns of value. The declaration varies depending on the variant of method.

Defining a simple method

A simple method that only does an execution, which is without any dynamic arguments and returns no value, can be declared like the sample code below. This is similar to methods we practised previously.

```
//Define a simple method
//This method displays a fixed message (number 7) on a console screen
//The keyword void indicates that this method does not return a value
1 reference
private static void mv_DisplayNumber()
{
    Console.WriteLine("7");
}
```

Defining a method with a return of value

A method can return a value to the code that calls for it. This value is passed on using the keyword *return*, usually at the end of a method. The data type of this return depends on the type a method is declared with.

```
//Define a method with return of value
//The keyword int indicates that this method may return a value of int type
//When defining a method with return value, it is more
1 reference
private static int mi_GivingReturnValue()
{
    return 7;
}
```

Defining a method with parameters

There are times when we need a method for repetitive use, with possible varying arguments. This allows more flexibility in using a method, especially if the method contains codes of a generic process.

```
//Define a method with parameters to accept dynamic inputs
//Parameters declared by specifying the variable type and variable name
//One or more parameters can be declared
1 reference
private static void mv_ShowReturnValue (int i_InputValue, String s_PrefixText)
{
    Console.WriteLine(s_PrefixText + ": " + i_InputValue);
}
```

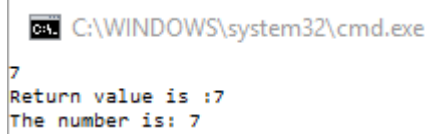
Sample codes to call methods

To use a method, simply call by specifying its name. Take note on the parameters, if any. Variable and value types must match.

```
static void Main(string[] args)
{
    //To use a declared method, simply call by specifying its name
    mv_DisplayNumber();

    //Call a method that returns a value, and display it on screen
    Console.WriteLine("Return value is : " + mi_GivingReturnValue());

    //Call a method with 2 arguments
    mv_ShowReturnValue(7, "The number is");
}
```



```
C:\WINDOWS\system32\cmd.exe
7
Return value is :7
The number is: 7
```

Full codes

```
class Program
{
    //Define a simple method
    //This method displays a fixed message (number 7) on a console screen
    //The keyword void indicates that this method does not return a value
    1 reference
    private static void mv_DisplayNumber()
    {
        Console.WriteLine("7");
    }

    //Define a method with return of value
    //The keyword int indicates that this method may return a value of int type
    //When defining a method with return value, it is more
    1 reference
    private static int mi_GivingReturnValue()
    {
        return 7;
    }

    //Define a method with parameters to accept dynamic inputs
    //Parameters declared by specifying the variable type and variable name
    //One or more parameters can be declared
    1 reference
    private static void mv_ShowReturnValue (int i_InputValue, String s_PrefixText)
    {
        Console.WriteLine(s_PrefixText + ": " + i_InputValue);
    }

    0 references
    static void Main(string[] args)
    {
        //To use a declared method, simply call by specifying its name
        mv_DisplayNumber();

        //Call a method that returns a value, and display it on screen
        Console.WriteLine("Return value is : " + mi_GivingReturnValue());

        //Call a method with 2 arguments
        mv_ShowReturnValue(7, "The number is");
    }
}
```