**Methods in C#**

A method is a parameterized, **reusable block of code**.

A method must be **defined** inside of a class, and then can be **invoked** by the object which owns it.

Methods can also be marked as **static** and exist on the class instead of on instance objects.

Every method has a distinct **signature** which consists of:

**Name**: The method’s name.

**Parameters**: The list of parameters, including the type and name of each parameter.

**Return Type**: The type of data the method returns when completed.

The code block inside of the method is called the **body** or **implementation** of the method.

In C#, methods that do not return anything are given a return type of **void**.

**Example Code – Class with Method Definitions:**

public class Person

{

private string \_firstName;

private string \_lastName;

public string GetFirstName()

{

return \_firstName;

}

public string GetLastName()

{

return \_lastName;

}

public string GetFullName()

{

return this.GetFirstName() + " " + this.GetLastName();

}

public void SetFirstName(string firstName)

{

\_firstName = firstName;

}

public void SetLastName(string lastName)

{

\_lastName = lastName;

}

public void SetFullName(string firstName, string lastName)

{

this.SetFirstName(firstName);

this.SetLastName(lastName);

}

}

**Encapsulation & Constructors**

One of the primary goals of object oriented programming is **encapsulation** or **information hiding**.

This is accomplished by making everything private that doesn’t absolutely need to be public.

The purpose of this process is to **simplify** the outward appearance and usage of a complex object.

The full set of public variables and methods of a class are called the **Application Programmer Interface (API).**

Not only the usage, but also the **creation** of objects should facilitate encapsulation.

We can control the creation of objects by writing **constructors**, the class method which instantiates objects.

A constructor is a method with **no return type** and having the **same name as the class**.

By defining a constructor, you will force the user to instantiate objects with the required parameters.

The actual parameter values passed to the constructor method can then be used in its body to set private data.

In this way, we get all the information we need to create the object without exposing its internal complexities.

In the example code below, by creating this constructor, the user **must** give a name to every Person they create.

**Example Code – Creating a simple constructor**

public class Person

{

private string \_name;

public Person(string name)

{

\_name = name;

}

}

You should note that if you do not define a constructor inside of your class, one will be provided for you.

This is called the **default constructor**, and will allow un-parameterized creation of instance objects.