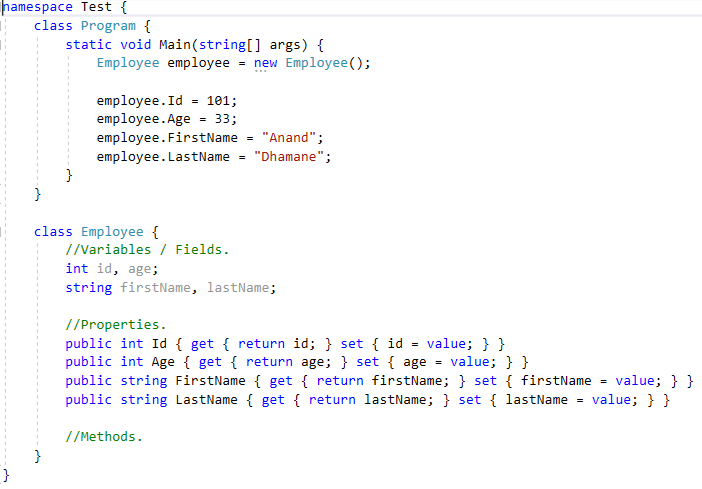
**CLASS**

Class is a template that defines the Data and Behaviours of the objects. The fields/variables and methods defined in the class act as a blueprint for objects that would be created from the class.

A Class allows variables and methods to be isolated to specific objects instead of being accessible by all parts of the program. This encapsulation of data protects each class from changes in other parts of the program. By using classes, developers can create structured programs with source code that can be easily modified.

Constructors are typically used to create objects from classes, while destructors are used to free up resources used by objects that are no longer needed.

e.g.

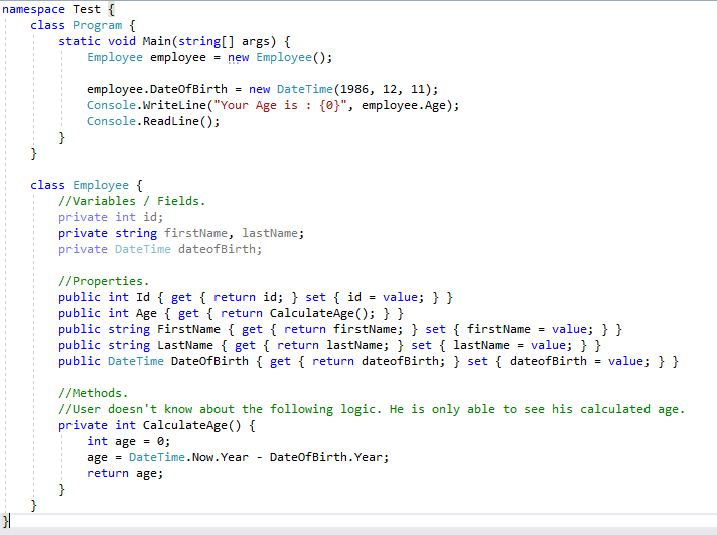


**ABSTRACTION**

Abstraction means displaying only essential information and hiding the background details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

Using Abstraction, we can change the internal implementation of a class independently without affecting the user. It also helps to increase security of an application or program as only important details are provided to the user. Its main goal is to handle complexity by hiding unnecessary details from the user.

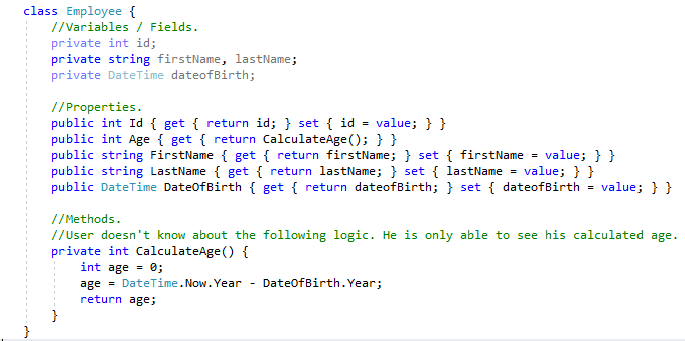
e.g.



**ENCAPSULATION**

Encapsulation is an Object-Oriented Programming concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse and preventing unauthorized parties’ direct access to them. Data encapsulation led to the important Object-Oriented Programming concept of data hiding.

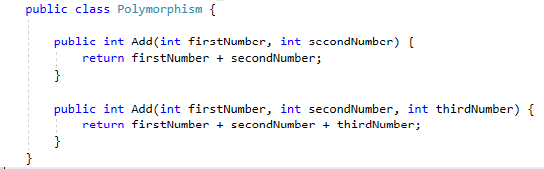
e.g.



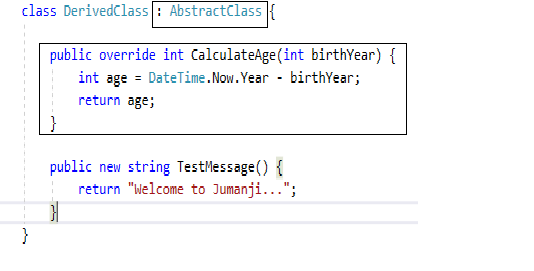
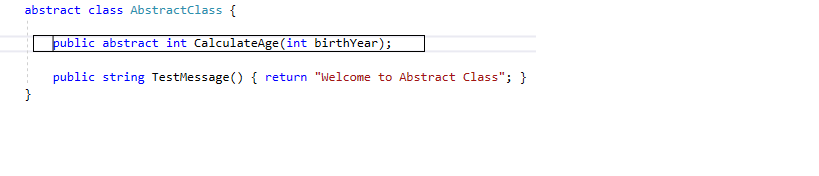
**POLYMORPHISM**

The term "Polymorphism" is the combination of **"poly"** + **"morphs"** which means many forms. There are two types of polymorphism in C#: Compile time polymorphism and Runtime polymorphism. Compile time polymorphism is achieved by Method Overloading and Operator Overloading in C#. It is also known as static binding or early binding. Runtime polymorphism is achieved by Method Overriding which is also known as dynamic binding or late binding.

e.g. Compile Time Polymorphism



e.g. Runtime Polymorphism



Polymorphism allows you to invoke Derived class methods through a Base class reference variable. Please refer to the below screenshot:

