Aia Al-Qasab

T5616SN

OBJECT-ORIENTED PROGRAMMING

ASSIGNMENT 2

VISUAL STUDIO CONSOLE APPLICATION

CLASS FUNDAMENTALS

September 2017

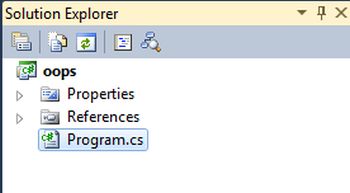


VISUAL STUDIO CONSOLE APPLICATION

CLASS FUNDAMENTALS

In object-oriented programming, you use abstraction and encapsulation to create well-designed classes. A class is a template or a blueprint for an object. This blueprint defines attributes for storing data and defines operations for manipulating that data. A class also defines a set of restrictions to allow or deny access to its attributes and operations. To create a well-designed class, you will use abstraction. In implementing abstraction, you will define a concept by using a minimal set of carefully considered functionality that provides the essential behavior of the class in an easy-to-use manner. Unfortunately, creating good software abstractions is not easy. Finding good abstractions usually requires a deep understanding of the problem being solved by the class and its context, great clarity of thought, and plenty of experience. The Visual Basic .NET form you have been working with is a good example of abstraction. The essential properties of a form, such as caption and background colour, have been abstracted in the Form class. Some essential operations that have been abstracted are open, close, and minimize.

To create a class by Visual studio :



And by pressing right click on Program.cs and chose add class, so we can make editing and change the name of the class.

C# console application project must require a single entry point Main () function that is already generated in the program class. If you add a new class and want to define one or more Main () entry points here then .NET will throw an error of multiple entry points. So it is advisable to delete or exclude the Program.cs file from the solution.

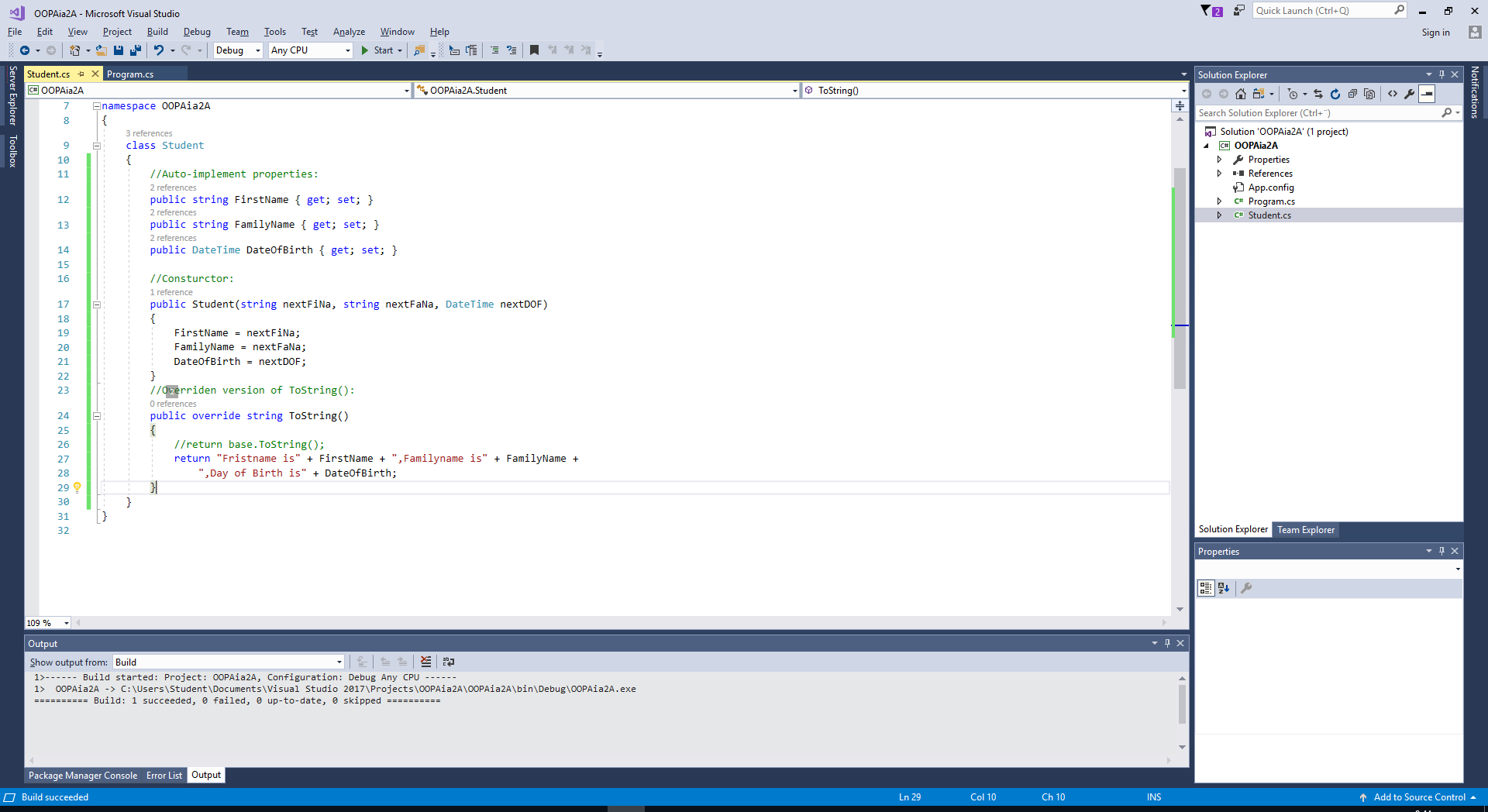
An object is a unit of software that contains a collection of related methods and data. An object is a specific instance of a class, and includes the characteristics of that class

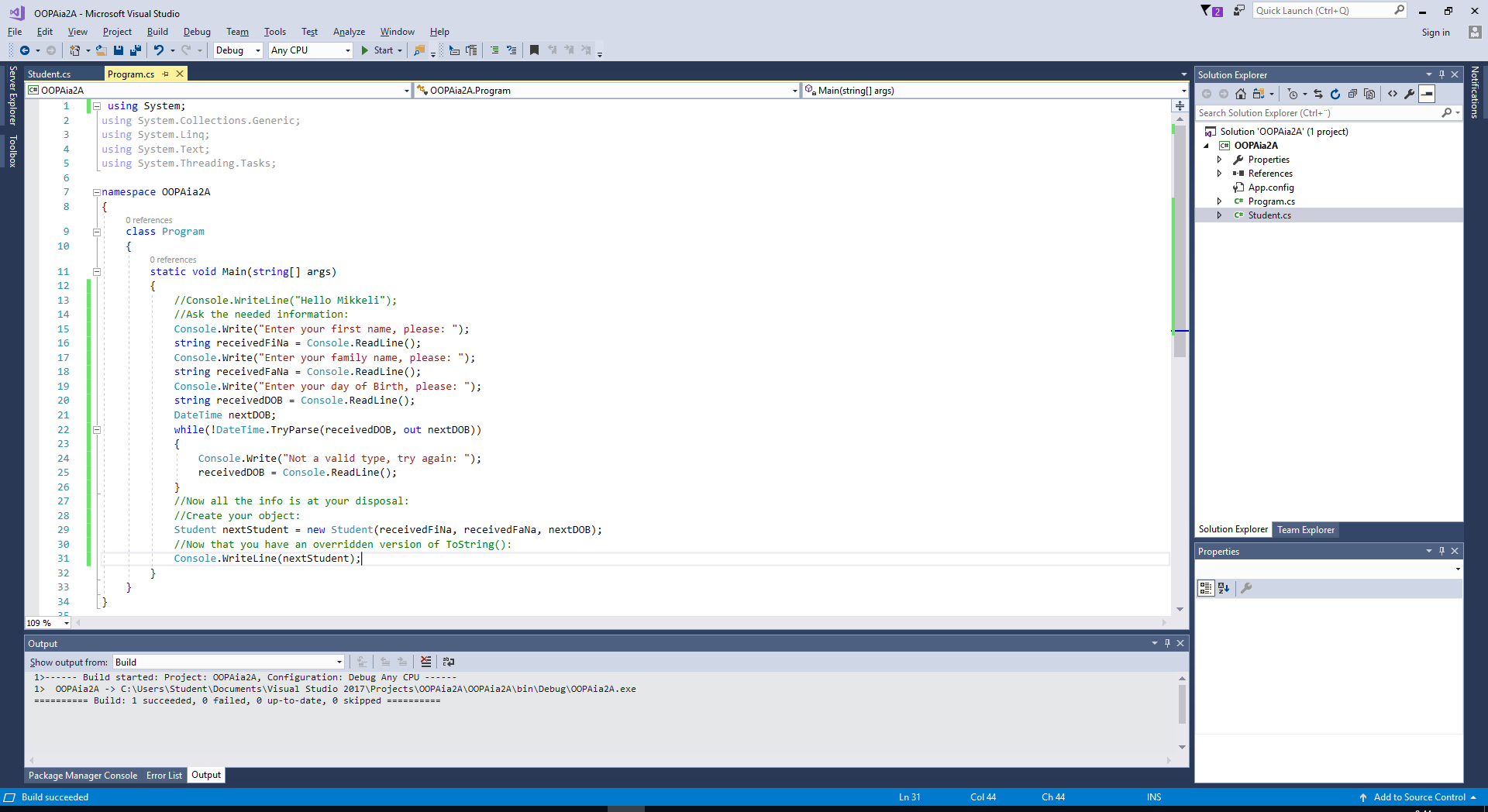
Objects provide behaviour. Behaviour is the characteristic that makes objects useful. Objects exist to provide behaviour. For example, most of the time you can ignore the inner workings of a car and think about its behaviour. Cars are useful because you can drive them. The inner workings exist, but they are mostly inaccessible. It is the behaviour of an object that is accessible. Objects of the same class share the same behaviour. A car is a car because you can drive it. Objects have state. State is the characteristic that refers to the inner workings of an object that enable it to provide its defining behaviour. A well-designed object keeps its state inaccessible. State is closely related to abstraction and encapsulation. You do not care how an object performs its actions, but you do care that it performs those actions. Two objects might coincidentally contain the same state, but they are still two different objects.

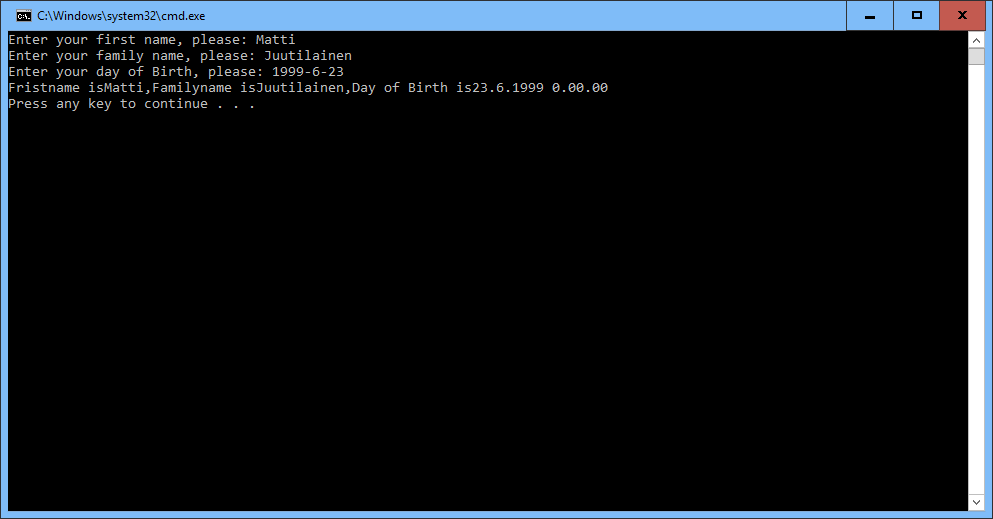
**Partial classes**  
  
Typically, a class will reside entirely in a single file. However, in situations where multiple developers need access to the same class, then having the class in multiple files can be beneficial. The partial keywords allow a class to span multiple source files. When compiled, the elements of the partial types are combined into a single assembly.  
  
There are some rules for defining a partial class as in the following;

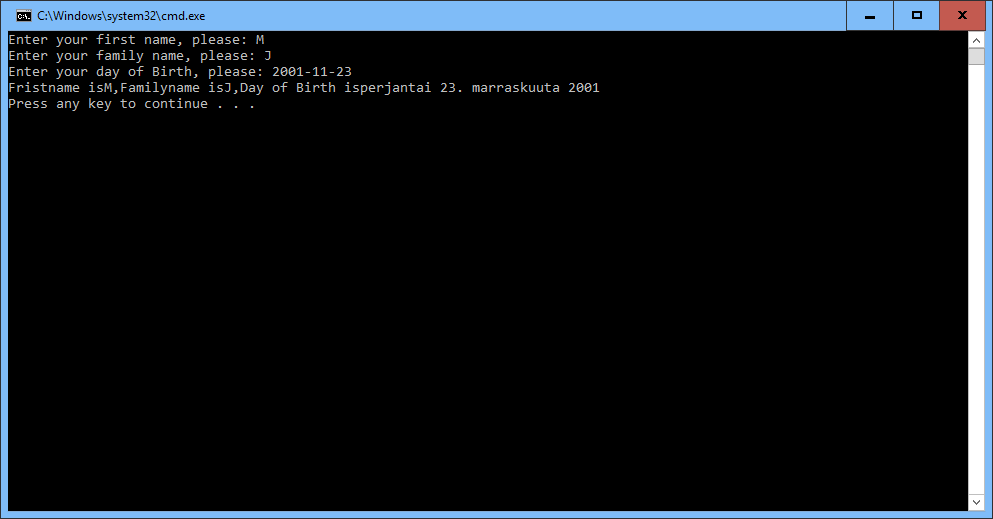
* A partial type must have the same accessibility.
* Each partial type is preceded with the "partial" keyword.
* If the partial type is sealed or abstract then the entire class will be sealed and abstract.

**Static classes**  
  
A static class is declared using the "static" keyword. If the class is declared as static then the compiler never creates an instance of the class. All the member fields, properties and functions must be declared as static and they are accessed by the class name directly not by a class instance object.









CultureInfo nextCulture = new CultureInfo("en-US");

//return base.ToString();

return "Fristname is" + FirstName + ",Familyname is" + FamilyName +

",Day of Birth is" + String.Format("{0:D}", DateOfBirth);

