C# Classes | Object Oriented Programming

**What is a class in OOP?**

A class allows you to define a sort of Blueprint for a datatype.This new class does not actually define any data, but it does define what the class can do. For example, the luxury car Ferrari. Ferrari is an object of the luxury car type. The luxury car is a class that specify certain characteristic like speed, color, shape, interior etc. So any company that makes a car that meet those requirements is an object of the luxury car type. For example, every single car of BMW, Lamborghini, Cadillac are an object of the class called 'Luxury Car'. Here, 'Luxury Car' is a class and every single physical car is an object of the luxury car class.

**Key words and terms for classes**

**Class**                  Keyword for setting up a class. All of the properties and methods are defined within the { } curly braces of the class.

**Properties**          These are the parts of a class where we can store values, very similar to a variable *(Sometimes called Fields or Attributes)*

**Methods**                  These are the parts that a class can do - or exectue, very similar to Methods

**Getters & Setters**  Getters and Setters are similar to methods with their aim set to the values of properties. Using getters and setters we are able to                                       manipulate the value of a property without changing the value of that property. In a more stricter sense of OOP we can use getters                                   and setters in a way so that we can protect the properties if needed.

**Example Class**

Here i have created a class that we can use to make**Person**objects. Each **Person** object can hold information about the person's first name, last name, age and address.

To be able to access and manipulate these properties, each of them needs to have a getter and setter. As can be seen from the first 3 getters and setters that the format in which you lay out the code isn't super important as all companies have there own ways of setting these out but you need to be aware that they all do the same thing in the end.

The final getter and setter in this example shows that we can have different access modifiers for the getters and setters. An example of this would be a banking app where you may want people to see the account holders details but not the account number. In the example above the user could set the persons address but wouldn't be able to retrieve that information.

**How do we use the new objects?**

To use this new object we need to make a "copy" of it, this is call instantiation. That means we create an instance of the object that we can use.

In this example i create a new Person object the same way as i would an array (because an array is an object) by using the key word **new.**In this example the new person object is called p1 and this new object p1 has all of the properties of the person class, in that it contains properties for the first name, last name, age and address. To be able to use these we need to use the p1 object followed by a dot and the name of the getter and setter.

in the example above i have set the p1.Fname property to "Jacob". This needs to be a string so that it corresponds to the data type in the objects class. This is continued with the last name requiring a string, the age requiring an int and the address requiring another string.

**Constructors**

A constructor is another part of an object that allows us to put the information directly into the new object at the time of creation. (Think of when we created arrays with the data already loaded into it). These constructors are pretty much the only time your allowed to use the exact same name with 2 different things.

As you can see in the above example when we have a constructor in our object class we need to provide the the data for the objects properties at the time of the objects creation.

**Object Methods**

In our object class we have the ability to create object specific methods. These methods often provide extra functionality to the parts of the object. They are set up just the same way that we would make a method anywhere else.

In the example above i have created a method that when called will display the persons full name.

**Displaying the objects properties**

To display any of our objects properties we need to reference which instance we want the information for.

In the above example i display the p1 objects fname property and the p2 objects age. I then change the value of the p1 fname property from Jacob to Jason and then call the Get Full name method to display the new full name of p1.

**Exercises**

**Task 1.**

Create a class called car that contains properties for the cars make, model, colour and rego along with the necessary getters and setters (No constructor). Once your class has been created, make 5 car objects and populate them with some details. Display all of the car objects properties to the user.

**Task 2.**

Create a person class that has public properties for First and last names and a private property for their age. Make a constructor which is used to populate the properties at the time of the objects creation. Create a Person object and display their full name and age to the user.

*What issues did you encounter?*

**Task 3.**

Create a class for an animal object with properties for the animal type, the animals name and an id number. The id number needs to be a 4 digit random number. This class needs to have a constructor and a method inside of the class that returns all of the animals details as a string back to the main method to be displayed.