Introduction to Classes

This lesson discusses classes, their definition, and declaration of classes in detail using an example.

WE'LL COVER THE FOLLOWING

Definition

As in other *object-oriented programming* languages, the functionality of a **C#** program is implemented in *one* or *more* **classes**.

The *methods* and *properties* of a **class** contain the code that defines how the *class* behaves.

Several types of **C#** *classes* can be defined, including **instance classes** (*standard classes* that can be instantiated), **static classes**, and **structures**.

Declaring a Class

Skeleton of *declaring* **class** is:

```
//whatever is written inside [] is optional while declaring a class
//whatever is written inside <> is required while declaring a class

[private/public/protected/internal] class <Desired Class Name> [:[Inherited class][,][[Interf
{
 //Your code
}
```

Class Declaration Syntax

- Classes are defined using the **keyword** `class`
- The *access modifiers* like `private` and `public` are optional and don't necessarily need to be specified
- The keyword `class` is followed by an **identifier** to *name* the class.
- *Instances* of the `class` can then be created with the `new` keyword followed by the **name** of the `class`.

Don't worry if you can't understand the whole syntax, you'll get familiar with it soon.

Example

As an example consider the following *class*:

```
using System;

class Dog
{
  //all members of class defined without a public access modifier are by default private
  //So age will be a private member of the class
  //hence it cannot be accessed outside of the class directly using the dot operator

  int age=2;


  public void PrintAge()
  {
    Console.WriteLine("Doggo's age is: {0}",age);//Will print number 2 in output
  }

}

public class SampleExample
{
    public static void Main()
    {
        Dog lucy = new Dog(); //creating instance the class Dog using new keyword
        //lucy is an object of class Dog
        //we use the dot operator to call the method defined in class Dog
        lucy.PrintAge();
    }
}
```

Explanation

- The `class` `Dog` has *two* members
  - the `int` *variable* `age`
  - `public` `void` type *method* `PrintAge()` which displays the `age`.

In a `class` unless the `public` *access modifier* is used before *declaring* a **member** it'll by default be considered a `private` member. Hence, in the *example* above `age` will by *default* be considered a `private` member of the class.

- In `Main`, the *instance* of `class` `Dog` named `lucy` would be its *object*.
- The `new` keyword is used to create the *object*.
- You can have *multiple objects* of a *class* just like you can have multiple dogs. So just like `lucy`, a **dog** named `ruffy` would be another *object* of the *class*.

Class *members* are like "inner variables" of each *object* made of type `Dog`. We used the dot operator to access these *members* of a class *object*.

Static Classes

The `static` *keyword* means **2** things:

1. This **value** does not change from *object* to *object* but rather changes on a **class** as a whole.
2. `static` *properties* and methods don't require an *instance*.

The `static` *keyword* when referring to a *class* has **three** effects:

- You **cannot** create an *instance* of a `static` class (this even removes the *default constructor*)
- All *properties* and *methods* in the *class* must be `static` as well.
- A `static` class is a sealed *class*, meaning it cannot be *inherited*.

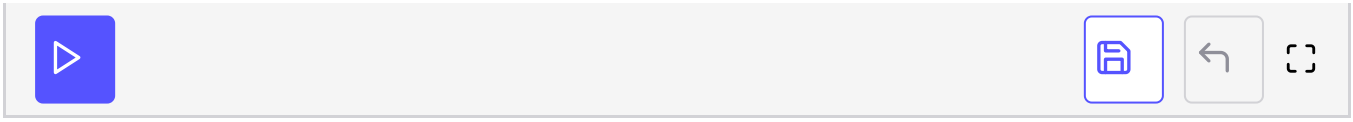
```
using System;

public static class Foo
{
 //Notice there is no constructor as this cannot be an instance
 public static int Counter;
 public static int GetCount()
 {
```

```
    return Counter;
  }
}


public class Program
{
  static void Main()
  {
    Foo.Counter++; //incrementing value of Counter
    Console.WriteLine("Value of Counter is: {0}",Foo.GetCount()); //this will print 1

    //this line would break the code as the Foo class does not have a constructor
    //var foo1 = new Foo();
  }
}
```



In the next lesson, we will discuss *access modifiers* that are used while making a *class*.