Lesson 20 - Classes & Objects

Classes

We have seen a lot of built-in or *primitive* type variables, which we use to store single values, and later we used Arrays, which were collections, and had some methods and properties.

In Object Oriented Programming, a `class` is a datatype that defines a set of variables and methods for a declared object. A `class` is a blueprint, and an `object` is an instance generated from that blueprint. The blueprint defines data and behavior for a type.

General syntax for a class defintion is as follows:

```
class Blueprint
{
        // variables, properties, methods.
}
```

Objects

The `type` is the class name, and when we create a variable of a class `type`, then we call that variable an `object`.

```
NOTE:
A class is NOT a variable. However, based on a class you can
create an object of that class and save it to a variable.
Objects are also called instances of a class.
```

Each object has its own characteristics, and these are called properties. Values of these properties describe the current state of the object.

Class Example

```
class Student
{
    public int Age;
    public string Name;
    public int[] Grades = new int[3];

    public void DisplayGrades()
    {
        Console.WriteLine("Student: " + Name);
        foreach (var grade in Grades)
            Console.WriteLine(grade);
    }
}
```

Object initialization is as follows:

```
Student student = new Student();
```

You can skip the type on right hand side if var isn't used. We can also access the variables and methods that have been defined as public.

```
Student student = new();
student.Name = "Talha";
student.Age = 21;
student.Grades = new int[] { 5, 6, 10 };
```

This can also be further simplified as,

```
Student student = new()
{
    Name = "Talha",
    Age = 21,
    Grades = new int[] { 5, 6, 10 }
};
```

We can also call its method using the `dot` operator just like how we used to do for arrays.

```
student.DisplayGrades();
```

```
OUTPUT:
Student: Talha
5
6
10
```