```
abstract class Vehicle
{
    public string color { get; set; }
    public int year { get; set; }
}

class Car : Vehicle
{
    public Car() { } // empty constructor

    public Car( string c, int y) // constructor
    {
        this.color = c;
        this.year = y;
     }
}

class Program
{
    static void Main(string[] args)
    {
        Car bmw = new Car(”Red”, 2005);
    }
}
```

<

```
abstract class Vehicle
{
    protected string color { get; set; }
    protected int year { get; set; }

    public string Color
    {
        get { return this.color; }
        set { this.color = value; }
     }

    public string Year
    {
        get { return this.year; }
        set { this.year = value; }
     }
}

class Car : Vehicle
{
}

class Program
{
    static void Main(string[] args)
    {
        Car bmw = new Car();

        bmw.Color = “Red”;
        bmw.Year = 2005;
    }
}
```

<

```
abstract class Vehicle
{
    protected string color;
    protected int year;

    public string getColor()
    {
        return this.color;
    }

    public void setColor(string c)
    {
        this.color = c;
    }

    public int getYear()
    {
        return this.year;
    }

    public void setYear(string y)
    {
        this.year = y;
    }
}

class Car : Vehicle
{
}

class Program
{
    static void Main(string[] args)
    {
        Car bmw = new Car();

        bmw.setColor(”Red”);
        bmw.setYear(2005);
    }
}
```

Public = Can be called from anywhere.
Protected = Can be called only inside and from child classes.
Private = Can be called only inside a class = Encapsulation.

Static = Can be called without instantiation.
no **Static** = Needs to be instantiated before calling.

Void = Does something. Returns nothing.
int, string, float, double, bool, Car = Does something and returns a value of selected type.

Virtual = The method is declared in the abstract class for inheritance. Can be overridden in a child class.
Override = Overrides the parent method.