### Step 1:

Create a new C# script in your Unity project called ‘TrafficLight’.

Import the FiniteStateMachine namespace defined in our script called FiniteStateMachine

If this is not working:

* You might not have the FiniteStateMachine script in your project
* You might have named the namespace something other than ‘FiniteStateMachine’
* The script your namespace is contained in might be named something different, and needs to also be called ‘FiniteStateMachine’

A screen shot of a computer

Description automatically generated

Create a public ‘enum’ called TrafficLightID, which will be used for tracking the current colour of the traffic light.



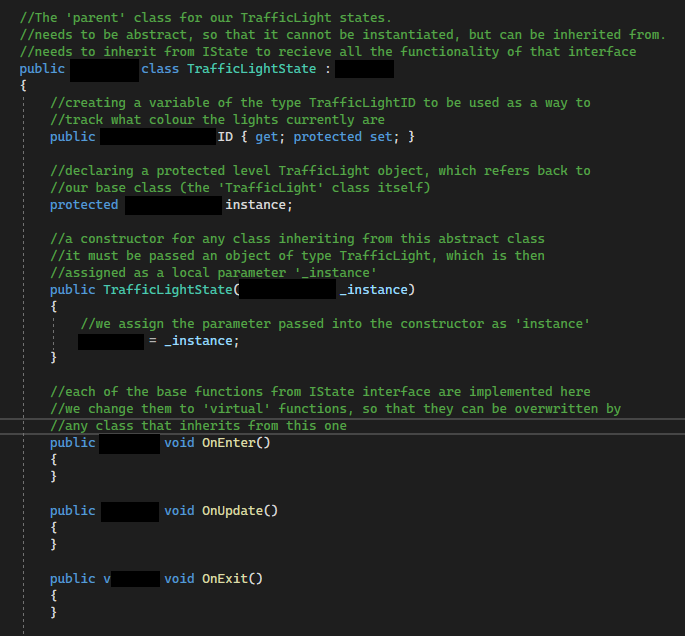
### Step 2:

Define the base (or ‘parent’ class) for our different states.

We will need to make this class abstract so that no object of this type can be created, but we can still inherit from it in other classes.

Create:

* A new public abstract class called *TrafficLightState*. Note that this class will be contained *inside* the *TrafficLight* class. *TrafficLightState* and all the other states we create will be nested classes (inside another class, *TrafficLight)*
* An public enum variable of ‘TrafficLightID’, with a public get and *protected* set.
* A *protected* variable of the TrafficLight class called ‘instance’
* A constructor for *TrafficLightState* which takes in a variable *\_instance* of type TrafficLight as a parameter
  + The constructor then assigned *\_instance* to the other variable *instance*
* Implement the *OnEnter(), OnUpdate()* and *OnExit()* methods from the *IState* interface
  + Add the *virtual* keyword to each, making it so that they can be overridden when used by other classes that inherit from this one



### Step 3:

Create the first of 3 states, the ‘GreenLight’ state

First, jump back to the top of the *TrafficLight* class and create a public MeshRenderer variable with a public get and private set. This will provide a way for our different states to check and set the colour of the material for the object this attaches to, without allowing other classes to do so.

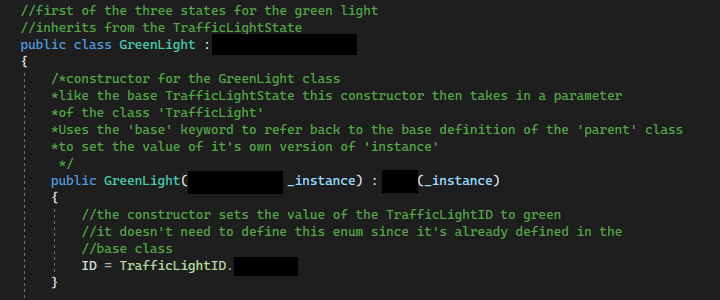
A black background with white text

Description automatically generated

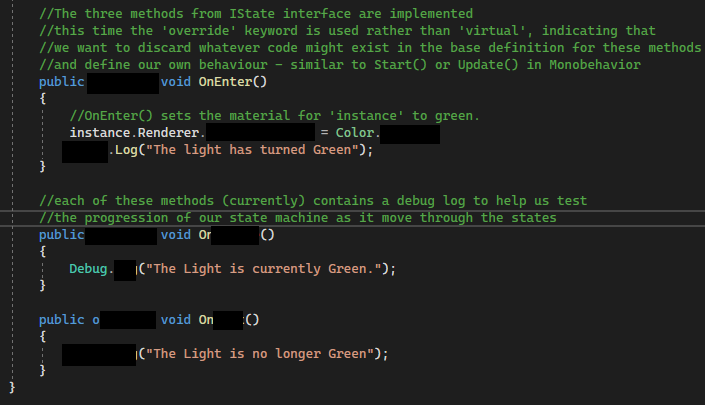
Next, create another nested class called ‘*GreenLight’* which inherits from *TrafficLightState*

Create:

* A constructor for the Greenlight class
  + The constructor sets the value for the *TrafficLightID* enum to green



* The three methods from the *IState* interface
  + The *OnEnter()* method, using the *override* keyword to replace the default functionality with our own code, sets the color of the Renderer variable in the *TrafficLight* class to green
  + *OnEnter()* will also contain a debug message stating the light has turned green.
  + *OnUpdate()* also uses *override* and currently will just show a debug message stating the light is still green.
  + *OnExit()* then shows a debug message stating the light is no longer green.



### Step 4

Set up a way to change the states & automatically assign key variables.

Back at the top of *TrafficLight*, create a public StateMachine variable also called StateMachine with a public get and private set

Inherit the *Awake()* function from *MonoBehaviour*.

* Use *GetComponent()* to automatically assign the value for Renderer
* Initialize the StateMachine variable

A screen shot of a computer program

Description automatically generated

In the *Start()* function we need to set what our initial state will be.

For now we will set the initial state as *GreenLight,* since we have not declared the other states yet.

