Thoughts

Task 1

I started with creating a *Singleton* class. I named it so since it does not have any meaningful purpose. From experience I have concluded that using a singleton that inherits from *MonoBehaviour* has some more benefits compared to a raw C# singleton. Therefore, I have forgotten how to implement one that does not and had to consult the internet. Since the singleton pattern is relatively standard, I placed it within a *singleton* region so that it can be easily hidden away. I proceeded with creating a public method which I named *CallDebugLog* since all it does is call a hard-coded *Debug.Log*. Finally, I created an empty object and another script that uses a serialized variable *KeyCode*, so that it can be changed from the inspector, to call the singleton and use the *CallDebugLog* method upon pressing the specified key. I placed the if statement within method which is called in the *Update* since it is standard practice to call control related code from there.

Task 2

During this task there were two distinct choices that I made. The first one has to do with the initial class which I named *Button*, the name deriving from a part of the instructions where it is mentioned that new objects should derive from the original button script. The choice itself has to do with using a list *List<Color>* in combination with an integer *currentColorIndex* instead of using two separate color variables. The reason for this is that one of the inheriting classes requires five colors instead of two. A list allows for this change. The second choice was to create getter/setters for two of the initial variables and leaving the other two as protected. The reasoning was that the Boolean and color list where to be displayed on the instructor. Furthermore, no significant control was needed around them. For the other two, *MeshRenderer* and *currentColorIndex*, which can result in exceptions and errors, I used getter/setters with conditions. These variables also do not need to be displayed in the inspector.

*When it comes to testing, I used a simple sphere object with a rigid body component which I manually moved around during runtime. I decided to leave the sphere in case you wish to run and test the program. I hope it is not a problem.

Use of Internet

I used the Internet for two subjects. The first one was the singleton pattern. I used the google search engine and search for “C# singleton pattern”. I chose to go with the site “stack overflow” which has been recommended by many colleagues and I have used a lot of times before. From there I followed a link that lead to a book which showed five different ways of implement the pattern and has a description for each of them. I chose one which is not overcomplicated but accounts for threads since threads are often used in game like software. The seconds was a search about Inheritance and Polymorphism. There was no specific thing I was searching for, but rather a general refreshment of how it works and common practices. For this I consulted Microsoft Docs.