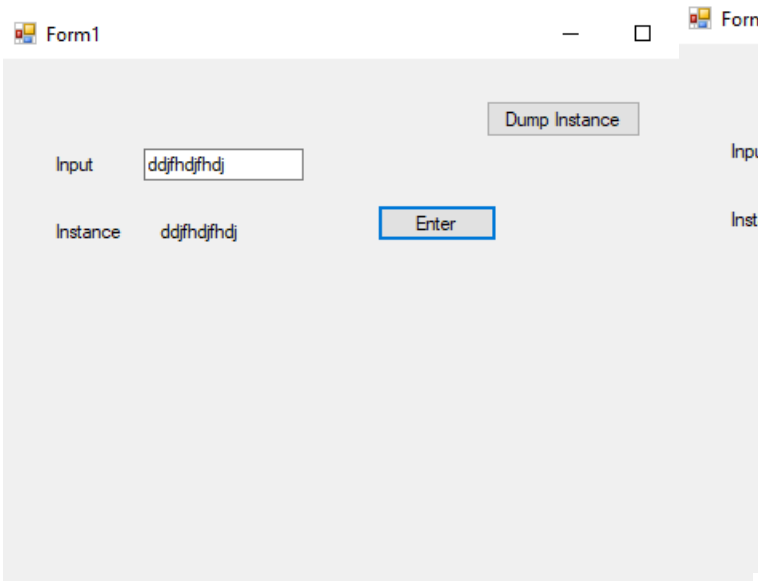
The Singleton Pattern

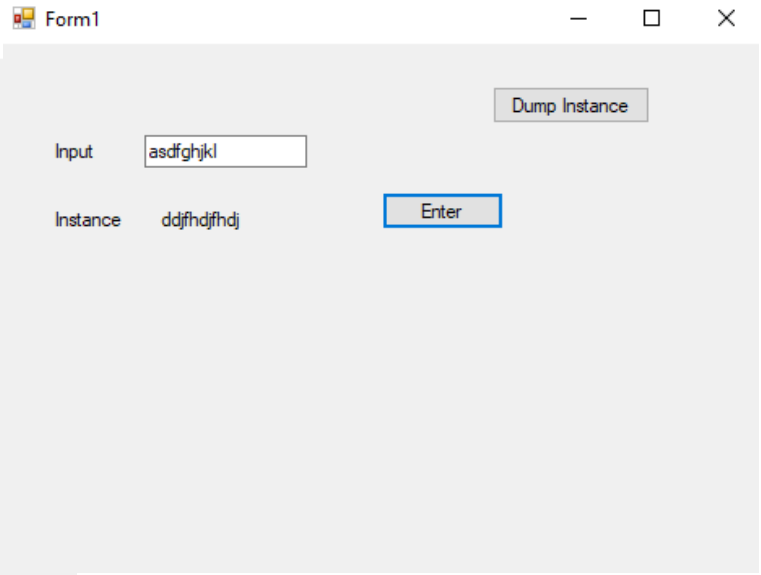
Introduction:

The purpose of this assignment was to create a sample application that made use of the singleton pattern as described: Ensure a class has only one instance and provide a global point of access to it. (definition obtained from http://www.dofactory.com/net/singleton-design-pattern)

Narrative:

For this assignment I created a simple application that makes an instance of a string and does not allow you to change the instance unless you first get rid of the other instance. In the first screenshot it shows that the instance has been assigned and in the second screenshot it shows that even after changing the text in the textbox and hitting enter, the instance does not change.






```
    class Singleton
    {
        private static Singleton
instance = new Singleton();

        private Singleton() { }

        public static Singleton
getInstance()
        {
            return instance;
        }
    }
```

```
        private void enterBtn_Click(object sender,
EventArgs e)
        {
            if (s == null)
            {
                instanceLbl.Text = txtBox.Text;
                s = Singleton.getInstance();
            }

        }

        private void changeBtn_Click(object sender,
EventArgs e)
        {
            s = null;
        }
```

Conclusion:

I didn't run into any problems while writing this app. Was a little rusty because I didn't do much programming over the summer and I don't have much experience with c#, so I did have to look a few things up. Like how to get the text from the text boxes, etc. This pattern wasn't difficult to understand and I can definitely see how it would be useful in a real world situation, because there are many times where you would only want a single instance of something running at one time.