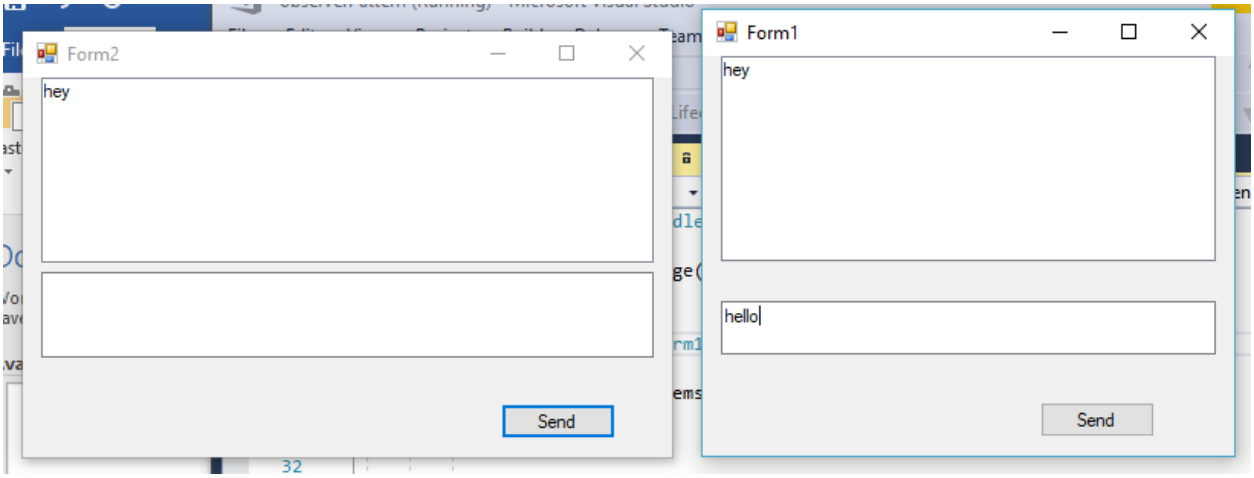
Observer Pattern

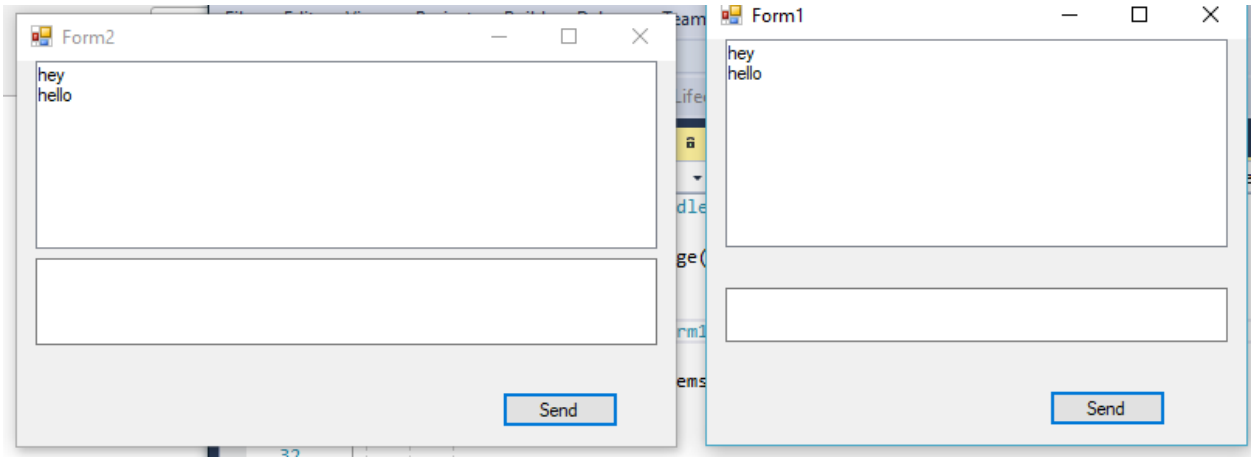
Introduction:

For this assignment I was asked to create a demo that implements the observer pattern, which is described as : Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

Narrative:

To do this, I created a demo that simulates text messenger toys for children, where you can only text the person that has the other text messenger. Kind of like walkie-talkies for texting. My app has two forms that can both send and receive messages between them. To do this, they both share a list that each message gets added to when a message is sent, this list is shown in a listbox. My main form (form 1) creates form 2 and attaches to the event listener of its send button, and form 2 attaches to the event on the other forms send button.





```
    public partial class Form2 : Form
    {
        private int count = 0;

        public Form2()
        {
            InitializeComponent();
        }

        public event EventHandler OnChildSendButtonClicked;

        public void sendMessage()
        {

            while (count < Form1.getList().Count)
            {

textWindow.Items.Add(Form1.getList().ElementAt(count));
                count++;
            }

        }

         private void sendBtn_Click(object sender, EventArg
        {
            Form1.sendList.Add(enterText.Text);
            if (OnChildSendButtonClicked != null)
                OnChildSendButtonClicked(null, null);
            sendMessage();
            enterText.Text = "";

        }
```

Conclusion: This pattern, conceptually wasn't hard to understand. It was more implementing the listeners between forms that was difficult, I spent a lot of time trying to figure out how to get the child forms event to also update the parents form. Overall I think that this pattern is useful, and it's probably used a lot without even knowing that its being used. I know that I have definitely used it in the past without knowing it.

```
public partial class Form1 : Form
  {
      public static List<String> sendList = new List<String>();
      private int count = 0;
      public Form1()
      {
          InitializeComponent();
          CreateAndShowForm();
      }
      void child_OnChildSendButtonClicked(object sender, EventArgs e)
      {
          while (count < sendList.Count)
          {
              textWindow1.Items.Add(sendList.ElementAt(count));
              count++;
          }
      }
      private event Action SendMessage
      private void CreateAndShowForm()
      {
          var form2 = new Form2();
          SendMessage += form2.sendMessage;
                  form2.OnChildSendButtonClicked += new
      EventHandler(child_OnChildSendButtonClicked);
          form2.Show();
      }
      public static List<String> getList()
      {
          return sendList;
      }

      private void sendButton_Click(object sender, EventArgs e)
      {
          sendList.Add(enterText1.Text);

          while (count < sendList.Count)
          {
              textWindow1.Items.Add(sendList.ElementAt(count));
              count++;
          }

          SendMessage();
          enterText1.Text = "";
      }
```