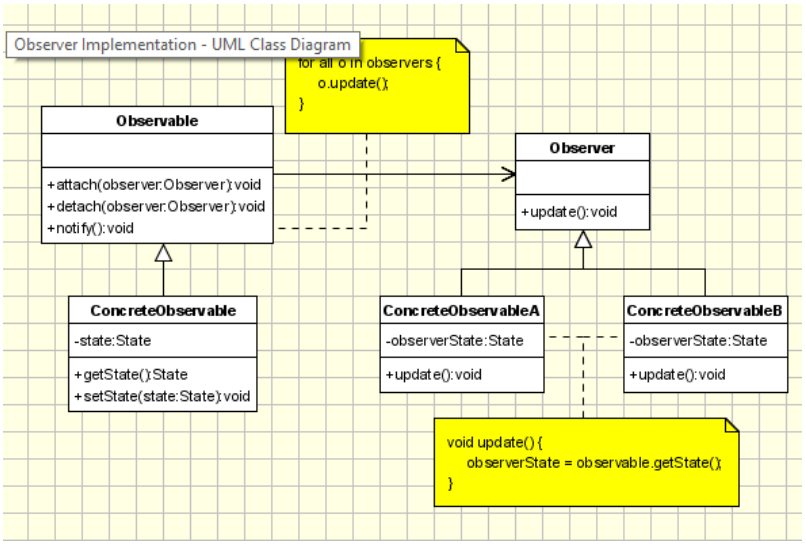
Alex Lang

The Observer Pattern

Design Patterns

9/22/16

For this assignment, we were told to do the observer pattern. The observer pattern serves to alert something when a change of state occurs or an action occurs. An observer class subscribes to an event, waits for the event to occur, and then alerts the observer to perform an action. The UML diagram for the observer pattern is as follows.



For my program, I created a donation alert system. I based this heavily on the streaming website Twitch.tv. Many people that stream on Twitch leave options for people to donate and support their streams. When a user donates, the streamer gets an alert of who donated and how much, and sometimes with a message the donator has included. I did a very simple version of this donation system, that used the observer pattern to alert the main form (the streamer) when a donation has occurred.

The donateEvent class is the heart of this entire program.

```
public class DonateEventArgs : EventArgs    //Donate event inherits from eventArgs
    {

        private string name;

        public string getName()     //Returns the name passed in from DonateEventArgs
        {
            return name;
        }

        private string donation;

        public string getDonation()    //Returns donation passed in
        {
            return donation;
        }

        private string comment;

        public string getComment()   //Returns comment passed in
        {
            return comment;
        }

        //DonateEventArgs constructor

        public DonateEventArgs(string newName, string newDonation, string newComment)
        {
            name = newName;
            donation = newDonation;
            comment = newComment;
        }
    }
```

This is the class that handles all of the donation information. The DonateEvent class inherits from EventArgs. The information from the forms is passed into this donateEventArgs object when it is created.

```
public partial class Form2 : Form
    {

        //DonateEvent Delegate,
        //Delegate meaning reference type variable that references method

        public delegate void DonateEventHandler(object sender, DonateEventArgs e);
        public event DonateEventHandler DonationMade;

        public Form2()
        {
            InitializeComponent();
        }


        //When the donate button is clicked, send in donate credentials.
        private void donateBtn_Click(object sender, EventArgs e)
        {
            DonationMade(this, new DonateEventArgs(nameBox.Text, donateBox.Text, messageBox.Text));
        }
    }
```

Form2 is where the actual donation page is. Here is where I created the event that does the actual alerting. First, I created a delegate DonateEventHandler. A delegate is used to reference variables that reference methods. Next I created an instance of the DonateEventHandler, which I named DonationMade since this event alerts the main form when a donation is made. Finally, I have the donate button. The donate button at the bottom of the page is what triggers the event and sends in all of the information into the DonateEventArgs. The this in donationMade refers to form2 since it is sending the event.

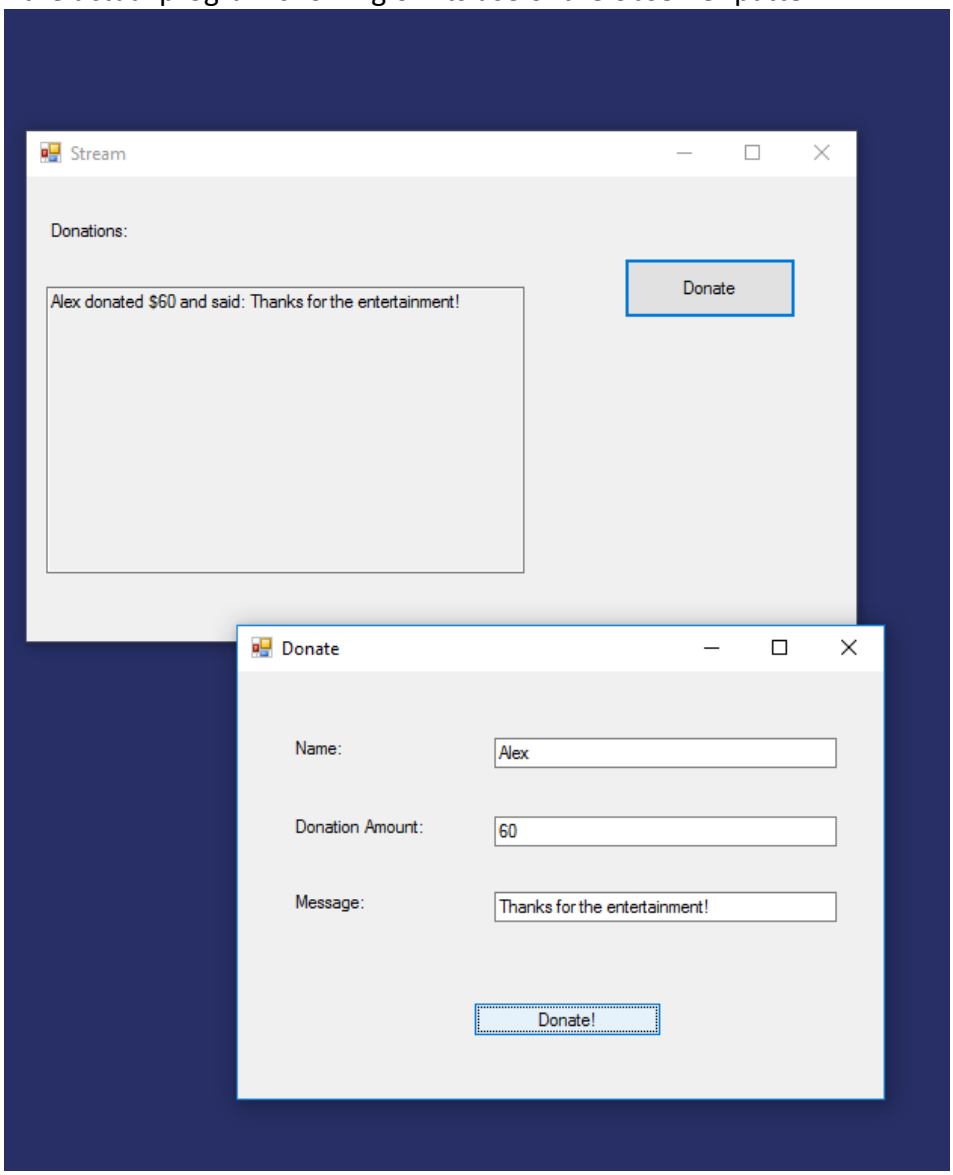
Finally, I coded the main form, which is the streamer's screen. This is where all of the alerts are sent to.

```
        public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        //Creates a new form for the donation page
        private void donateFormBtn_Click(object sender, EventArgs e)
        {
            Form2 f2 = new Form2();
            f2.DonationMade += new Form2.DonateEventHandler(donateSubmit); //Subscribes to the
donateHandler
            f2.Show();
        }
```

```
        //Takes information from event and outputs.
        void donateSubmit(object sender, DonateEventArgs e)
        {
            string newDonation = e.getName() + " donated $" + e.getDonation() + " and said: " +
e.getComment() + "\r\n";
            donationBox.Text += newDonation;
        }

    }
```

The donateFormBtn is the button that creates the second form where all of the donating occurs. The += here subscribes to the donate event that I created in form2. When the event occurs, it adds a new donation, which updates the text box for the streamer to see.

Here is a screenshot of the actual program showing off its use of the observer pattern.



Conclusion: The actual creation of the app was very easy, but reading up and understanding fully how events worked was pretty long and frustrating. Now that I have a decent understanding of what it does however, it is pretty basic. I really enjoyed making this app.