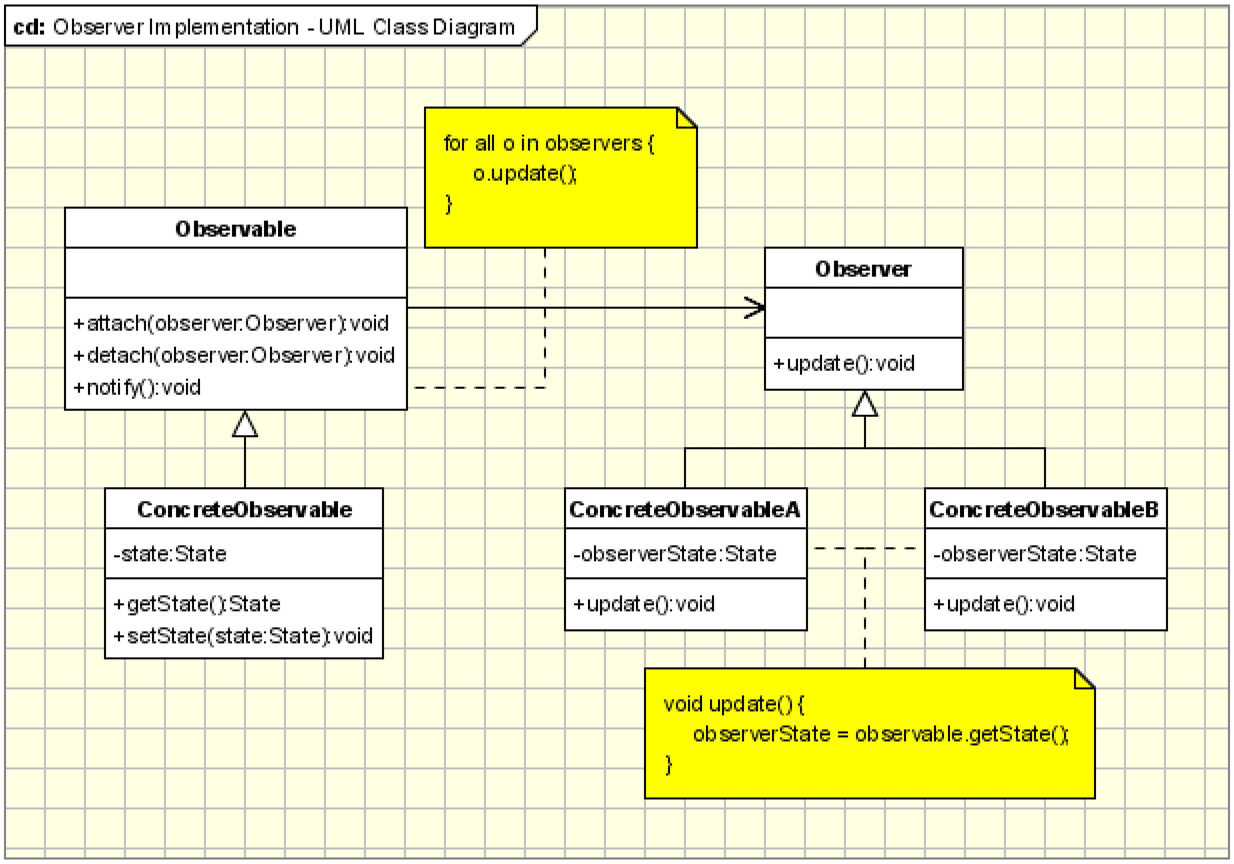
Jack Raney

Design Patterns

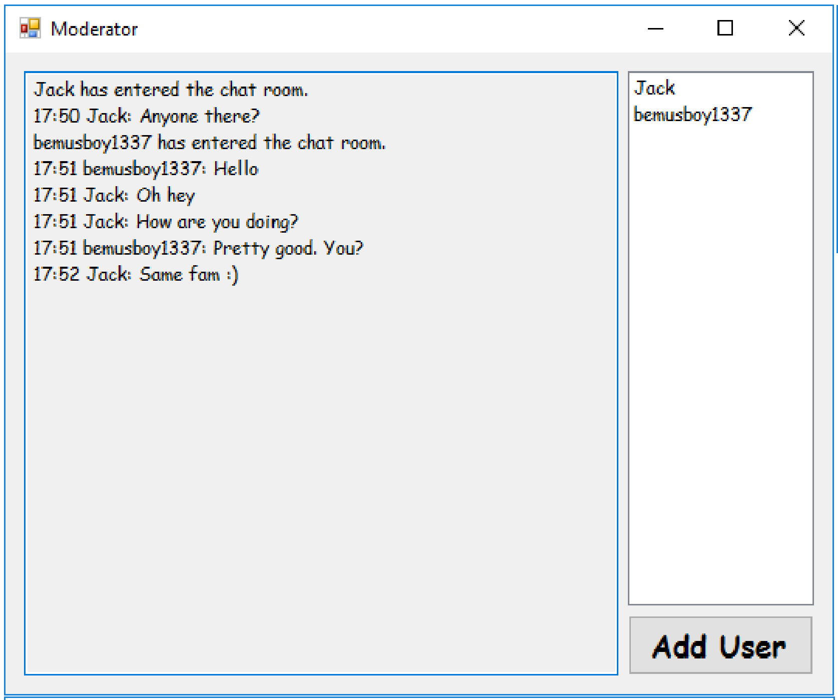
9/22/2016

The Observer Pattern

This paper will showcase the Observer Pattern, a pattern that's intent according to oodesign.com is to "[Define] a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically". What this means is that an "observer" class subscribes to an event handler/listener and waits for that event to occur. When it occurs, the observer performs an action. The UML diagram for the Observer Pattern is shown below:

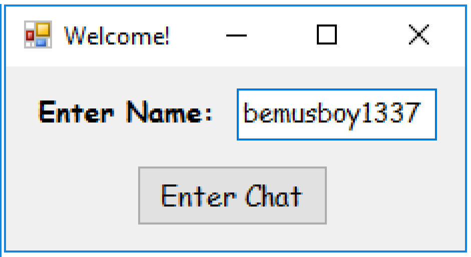


As an example showcasing the Observer Pattern using the .NET Framework's event class, I created the program ChatSim.cs:

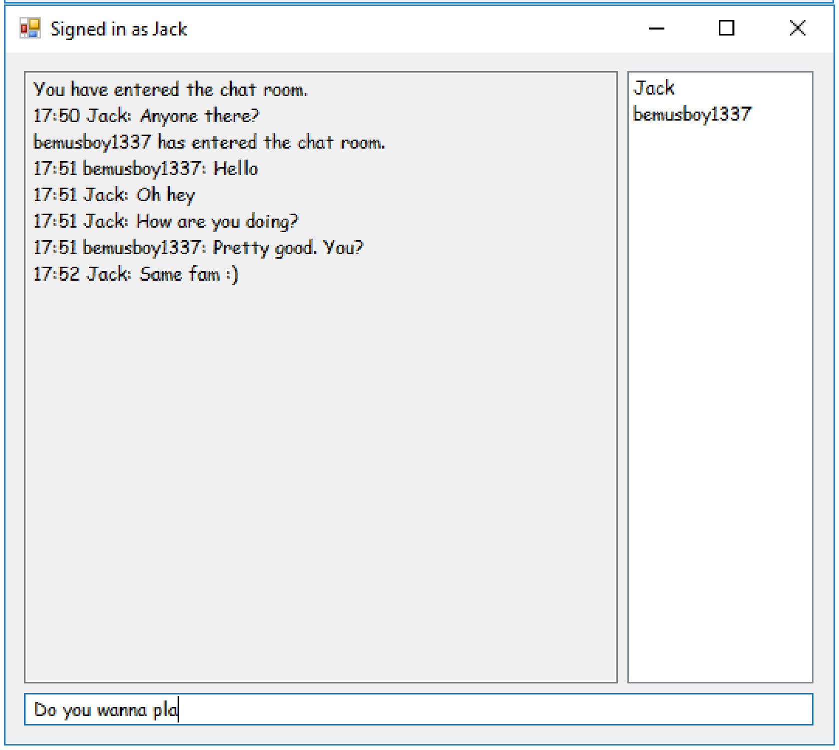


```
Jack has entered the chat room.
17:50 Jack: Anyone there?
bemusboy1337 has entered the chat room.
17:51 bemusboy1337: Hello
17:51 Jack: Oh hey
17:51 Jack: How are you doing?
17:51 bemusboy1337: Pretty good. You?
17:52 Jack: Same fam :)
```

```
Jack
bemusboy1337
```



Enter Name: bemusboy1337



```
You have entered the chat room.
17:50 Jack: Anyone there?
bemusboy1337 has entered the chat room.
17:51 bemusboy1337: Hello
17:51 Jack: Oh hey
17:51 Jack: How are you doing?
17:51 bemusboy1337: Pretty good. You?
17:52 Jack: Same fam :)
```

```
Jack
bemusboy1337
```

```
Do you wanna pla
```

```
public partial class Form1 : Form   //This is the observer
    {
        string time;

        public Form1()
        {
            InitializeComponent();
        }

        private void addButton_Click(object sender, EventArgs e)
        {
            Form2 newUser = new Form2();
            newUser.Location = new Point(this.Location.X, this.Location.Y);
            newUser.NameEntered += new Form2.NameEventHandler(userEntered);
            newUser.Show();
        }
```

When the program is opened, Form1 is shown first. When the "Add User" Button is clicked, a Form2 is created, and Form1 subscribes to Form2's NameEventHandler. (the time string will be talked about later).

```
public partial class Form2 : Form   //This is an observable
    {
        public delegate void NameEventHandler(object sender, NameEventArgs e);
        public event NameEventHandler NameEntered;

        public Form2()
        {
            InitializeComponent();
        }

        private void enterButton_Click(object sender, EventArgs e)
        {
            Form3 user = new Form3(nameTextBox.Text);
            user.Location = new Point(this.Location.X, this.Location.Y);
            this.Hide();
            user.Show();
            if (NameEntered != null)
                NameEntered(this, new NameEventArgs(user));
        }
    }

    public class NameEventArgs : EventArgs
    {
        private Form3 userChat;

        public Form3 getForm()
        {
            return userChat;
        }

        public NameEventArgs(Form3 form)
        {
            userChat = form;
        }
```

Form 2 asks for a username to be entered, and when the button is clicked, a Form3 (user) is created. Alongside this Form3, an event triggers and the Form2 closes.

The event, NameEventHandler, passes the NameEventArgs class, a subclass of EventArgs, into Form1, which carries with it the instantiation of the new Form3.

This event causes Form1 to call its userEntered method, which adds the new Form3 to the userListBox, subscribes Form1 to another event, adds the other into the new user's userListBox, adds a message announcing their arrival to the chat room to everyone present, adds the new user into the Moderator's (Form1's) userListBox, and adds that user to the users' userListBoxes.

```
void userEntered(object sender, NameEventArgs e)
        {
            Form3 user = e.getForm();
            user.TextEntered += new Form3.EnterEventHandler(updateChat);

            foreach (Form3 userName in userListBox.Items)
            {
                user.addUser(userName);
            }

            chatTextBox.Text += user.getUser() +
                " has entered the chat room." + Environment.NewLine;
            foreach (Form3 userName in userListBox.Items)
            {
                userName.updateChat(user.getUser() +
                    " has entered the chat room." + Environment.NewLine);
            }
            userListBox.Items.Add(user);
            foreach (Form3 userName in userListBox.Items)
            {
                userName.addUser(user);
            }
        }
```

On initialization, Form3 enters in the name form Form2 as its username and creates a starting message telling the user that he/she entered the chat room. The functons getUser and getText simply return the username and text, respectively, and a ToString override makes Form3 objects appear as only the username in the userListBoxes. updateChat passes text into the user's chatTextBox, and addUser simply adds a new user into the old users' userListBoxes. Finally, whenever the text within the enterTextBox changes (i.e. when the user types into the text bar) it checks if the user pressed the enter key. If they did, the EnterEventHandler triggers.

```
public partial class Form3 : Form   //This is an observable
    {
        private string userName;
        private string enteredText;
        const   string ENTER = "\n";

        public delegate void EnterEventHandler(object sender, EnterEventArgs e);
        public event EnterEventHandler TextEntered;

        public Form3(string name)
        {
            InitializeComponent();
            userName = name;
            this.Text = "Signed in as " + userName;
            chatTextBox.Text = "You have entered the chat room." + Environment.NewLine;
        }

        public string getUser()
        {
            return userName;
        }

        public string getText()
        {
            return enteredText;
        }

        public override string ToString()
        {
            return userName;
        }

        public void updateChat(string newText)
        {
            chatTextBox.Text += newText;
        }

        public void addUser(Form3 user)
        {
            userListBox.Items.Add(user);
        }

        private void enterTextBox_TextChanged(object sender, EventArgs e)
        {
            if (enterTextBox.Text.Contains(ENTER))
            {
                enteredText = enterTextBox.Text;
                enterTextBox.Text = "";
                if (TextEntered != null)
                    TextEntered(this, new EnterEventArgs(userName, enteredText));
            }
        }
```

This event passes the EnterEventArgs class, another EventArgs child class, into Form1, which brings the user's username and entered message with it.

```
public class EnterEventArgs : EventArgs
    {
        private string name;

        public string getName()
        {
            return name;
        }

        private string text;

        public string getText()
        {
            return text;
        }

        public EnterEventArgs(string user, string message)
        {
            name = user;
            text = message;
        }
    }
```

This event causes Form1 to call its last method, updateChat. This method sets the previously mentioned time string to the current time (in hours and minutes), and then passes that into a new string along with the user's username and message. This new string is entered into Form1 and every user's chatTextBoxes via their own updateChat functions.

```
void updateChat(object sender, EnterEventArgs e)
        {
            time = DateTime.Now.TimeOfDay.ToString().Substring(0, 5);
            string chatLine = time + " " + e.getName() + ": " + e.getText();
            chatTextBox.Text += chatLine;

            foreach (Form3 user in userListBox.Items)
            {
                user.updateChat(chatLine);
            }
        }
```

With all of this together, a functioning chat system is created.

Reflection

This program is one of the most complicated I have created so far in any language, but I am very proud of it and may revisit it later to add more features unrelated to the Observer Pattern. I had to learn several things for this program, such as how the event class works in the .NET Framework and how to use the foreach iterator. I would say that I am very proud of this program.