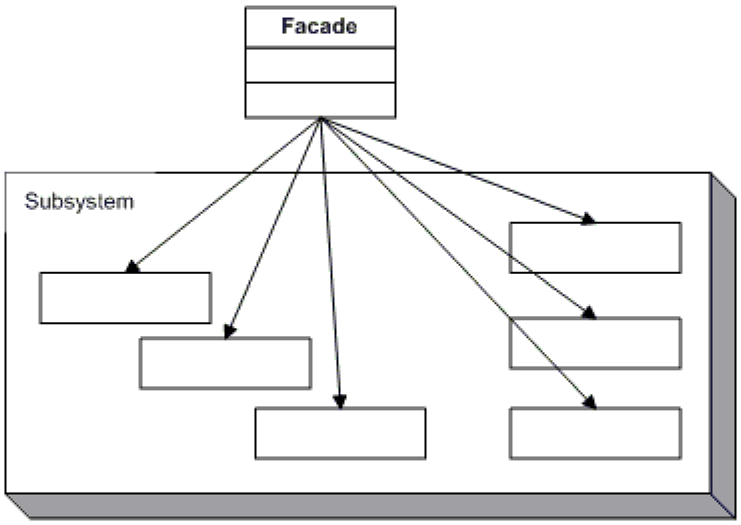
Introduction

The assignment required the program to provide a unified interface to a set of interfaces in a subsystem. It defined a higher level interface that made the other subsystems easier to use. The UML to the right helps to visualize the program's basic function. The form that I chose the program to take was that of a universal television remote that can control the TV, DVD player, sound system, and game system.

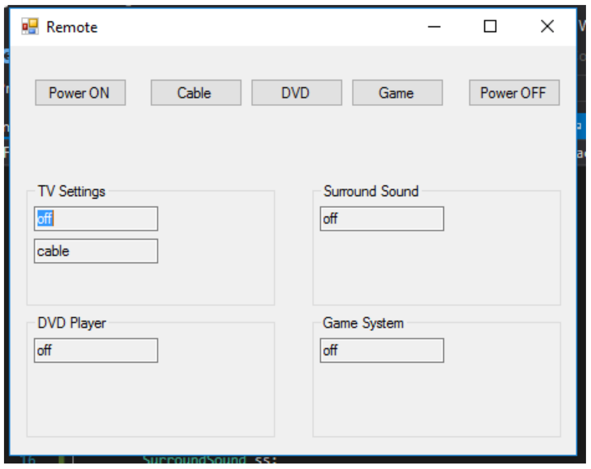


Façade Pattern

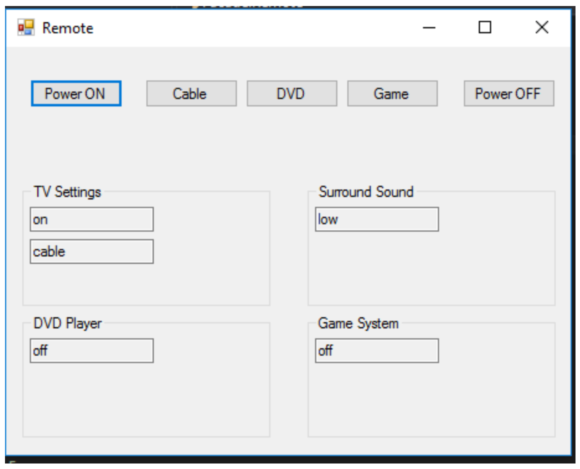
As you can see from the UML the Façade class is simply a controller for the other subsystems. I chose TV, DVD, Game, and SoundSystem for my subsystem names, and UniversalRemote as my façade class. Below you can see how I linked the subsystems to the façade class.

```
public class UniversalRemote
    {
        private TV m_tv;
        private SurroundSound m_ss;
        private DVD m_dvd;
        private Game m_gs;
```

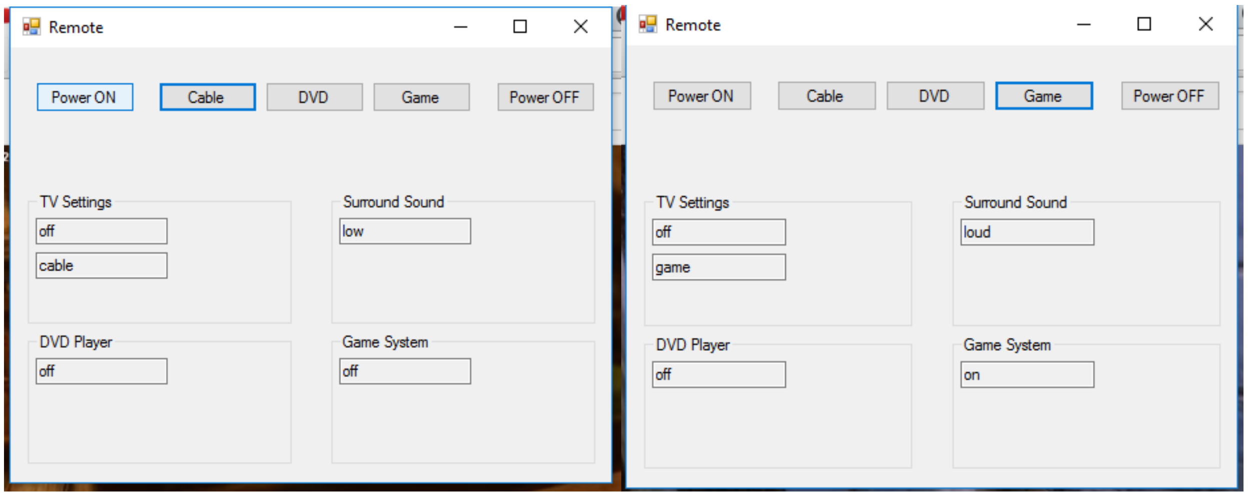
Each of the four subsystems have their own coding and methods that they execute, but the UniversalRemote class is what controls them and calls them to work. Without the main class, there would need to be four different forms opened and working. But as it is now, the whole app is in one contained window, that can be seen below and on the next page.



TV Settings: off, cable
Surround Sound: off
DVD Player: off
Game System: off



TV Settings: on, cable
Surround Sound: low
DVD Player: off
Game System: off



TV Settings: off, cable
Surround Sound: low
DVD Player: off
Game System: off

TV Settings: off, game
Surround Sound: loud
DVD Player: off
Game System: on

Each button has its own coding that references a subclass to change its setting. Below is the code for the Game button, which you can see above what it actually changes.

```
  private void GameBtn_Click(object sender, EventArgs e)
        {
            tv.m_InputState = TV.TVInputState.game;
            gs.m_GameState = Game.GameState.on;
            dvd.m_DVDState = DVD.DVDState.off;
            ss.m_SSState = SurroundSound.SoundState.loud;
        }
```

This chunk of code changes the input of the TV class, the power of the DVD and Game Class, and the intensity of the sound class, which was all instantiated elsewhere.

```
namespace Facade
{
    public delegate void TVStateChangedEventHandler(object sender, EventArgs e);
    public class TV
    {
       public enum TVPowerState {off, on}
       public enum TVInputState {cable, DVD, game}
       public event TVStateChangedEventHandler TVStateChanged;
       private TVPowerState _PowerState;
        private TVInputState _InputState;

       public TVPowerState m_State
       {
            get
            {
                return _PowerState;
            }

            set
            {
                _PowerState = value;
                if (TVStateChanged != null)
                    TVStateChanged(this, new EventArgs());
            }
       }
        public TVInputState m_InputState
        {
            get
            {
                return _InputState;
            }
            set
            {
                _InputState = value;
                if (TVStateChanged != null)
                    TVStateChanged(this, new EventArgs());
            }
        }
    }
}
```

Above is the TV class, where I establish the different settings that the TV power and input can have. It also gets and sets the current settings for the TV power and inputs by applying a value to the methods _State and _InputState. All the other subsystems have the same set up for ease of coding. After all the coding was typed, all that was left to do was set up the buttons on the form to call the functions and change the text boxes. Below I have the code I used to allow the buttons to interact with the different subsystems.

```
        public Remote()
        {
            InitializeComponent();
            tv = new TV();
            ss = new SurroundSound();
            dvd = new DVD();
            gs = new Game();
            tv.TVStateChanged += new TVStateChangedEventHandler(tv_TVStateChanged);
            ss.SurroundSoundStateChanged += new SurroundSoundStateChangedEventHandler(ss_SurroundSoundStateChanged);
            dvd.DVDStateChanged += new DVDStateChangedEventHandler(dvd_DVDStateChanged);
            gs.GameStateChanged += new GameStateChangedEventHandler(gs_GameStateChanged);
            UpdateControls();

        }

        void tv_TVStateChanged(object sender, EventArgs e)
        {
            UpdateControls();
        }

        void ss_SurroundSoundStateChanged(object sender, EventArgs e)
        {
            UpdateControls();
        }

        void dvd_DVDStateChanged(object sender, EventArgs e)
        {
            UpdateControls();
        }

        void gs_GameStateChanged(object sender, EventArgs e)
        {
            UpdateControls();
        }

        public void UpdateControls()
        {
            TVPowerState.Text = tv.m_State.ToString();
            TVInputState.Text = tv.m_InputState.ToString();
            SoundState.Text = ss.m_SSState.ToString();
            DVDPowerState.Text = dvd.m_DVDState.ToString();
            GamePowerState.Text = gs.m_GameState.ToString();

        }
```

After this was done I was able to set up the buttons to call on the different classes. An example of the button code is below. The game button is one of the buttons pressed in the pictures I cut of the program running. It Changes the input on the TV to game, turn the sound to loud, turns off the DVD player, and turns on the game player.

```
  private void GameBtn_Click(object sender, EventArgs e)
        {
            tv.m_InputState = TV.TVInputState.game;
            gs.m_GameState = Game.GameState.on;
            dvd.m_DVDState = DVD.DVDState.off;
            ss.m_SSState = SurroundSound.SoundState.loud;
        }
```

<u>Observations</u>

This program was harder than I expected it to be. There was a lot of work that went in to setting up the different subsystems and the main class. I made a lot of errors along the way, but it was a good learning experience. I came out of the assignment with a much better understanding of the C# language.