Observer Pattern

Design Patterns

Mark Swarner

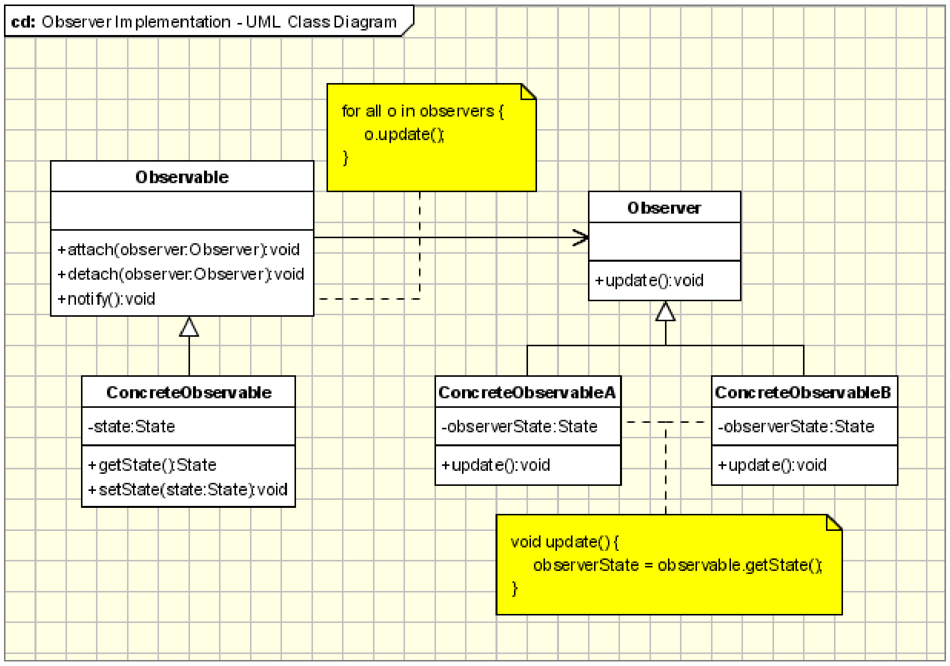
Fall 2016-17

Introduction:

For this assignment, we were required to write a C# application that correctly demonstrates the Observer Pattern by using the .NET Framework's event class.

The UML Diagram for the Observer Pattern:

While we didn't create a class structure utilizing the UML Diagram, .NET Framework's Event class takes care of the structuring for us. Below is a Table of Classes used:



Form1	Form 1 is the Observer
Form2	Form 2 is the Observable
Album	Album is an Type I created
Song	Song is another Type I created
Library	Library creates and stores the data as well as implements the Boolean functions
AlbumCheckEventArgs	The custom defined replacement for EventArgs in the context of Albums
SongCheckEventArgs	The custom defined replacement for EventArgs in the context of Songs

Narrative:

```
public class Song
    {
        public String Name
{ get; set; }
        public String Length
{ get; set; }
        public String TrackNum
{ get; set; }
    }
```

The "Song" class defines a custom Type with readable and writeable properties "Name", "Length, and "TrackNum".

```
public class Album
    {
        public List<Song> Songs
{ get; set; }
        public String Artist
{ get; set; }
        public String Genre
{ get; set; }
        public String Name
{ get; set; }
        public int Year { get;
set; }
        public String TrackTotal
{ get; set; }
        public String Length
{ get; set; }
    }
```

The “Album” class defines a custom Type with readable and writeable properties “Songs”, “Artist”, “Genre”, “Name”, “Year”, “TrackTotal”, and “Length”. “Songs” is a collection of variables with Type “Song”.

```
public class Library
    {
        public static List<Album> library = new List<Album>()
        {
            new Album
            {
                Songs = new List<Song>
                {
                    new Song { Name = "Papercut", Length = "3:04", TrackNum = "01" },
                    new Song { Name = "One Step Closer", Length = "2:35", TrackNum =
"02" },
                    new Song { Name = "With You", Length = "3:23", TrackNum = "03" },
                    new Song { Name = "Points Of Authority", Length = "3:20", TrackNum
= "04" },
                    new Song { Name = "Crawling", Length = "3:29", TrackNum = "05" },
                    new Song { Name = "Runaway", Length = "3:04", TrackNum = "06" },
                    new Song { Name = "By Myself", Length = "3:09", TrackNum = "07" },
                    new Song { Name = "In The End", Length = "3:36", TrackNum = "08" },
                    new Song { Name = "A Place For My Head", Length = "3:04", TrackNum
= "09" },
                    new Song { Name = "Forgotten", Length = "3:14", TrackNum = "10" },
                    new Song { Name = "Cure For The Itch", Length = "2:37", TrackNum =
"11" },
                    new Song { Name = "Pushing Me Away", Length = "3:11", TrackNum =
"12" }
                },
                Name = "Hybrid Theory", Artist = "Linkin Park", Genre = "Nu Metal",
Year = 2000, TrackTotal = "12", Length = "37:46"
            },
            new Album
            {
                Songs = new List<Song>
                {
                    new Song { Name = "Human Race", Length = "4:09", TrackNum = "01" },
                    new Song { Name = "Painkiller", Length = "2:58", TrackNum = "02" },
                    new Song { Name = "Fallen Angel", Length = "3:06", TrackNum =
"03" },
                    new Song { Name = "Landmine", Length = "3:25", TrackNum = "04" },
          new Song { Name = "Tell Me Why", Length = "3:30", TrackNum =
"05" },
          new Song { Name = "I Am Machine", Length = "3:21", TrackNum =
"06" },
          new Song { Name = "So What", Length = "2:57", TrackNum = "07" },
          new Song { Name = "Car Crash", Length = "2:50", TrackNum = "08" },
          new Song { Name = "Nothing's Fair in Love and War", Length =
"3:44", TrackNum = "09" },
          new Song { Name = "One Too Many", Length = "2:41", TrackNum =
"10" },
          new Song { Name = "The End Is Not the Answer", Length = "2:52",
TrackNum = "11" },
          new Song { Name = "The Real You", Length = "3:54", TrackNum =
"12" }
        },
        Name = "Human", Artist = "Three Days Grace", Genre = "Alternative
Metal", Year = 2015, TrackTotal = "12", Length = "39:27"
      },
      new Album
      {
        Songs = new List<Song>
        {
          new Song { Name = "Dark", Length = "2:10", TrackNum = "01" },
          new Song { Name = "Failure", Length = "3:34", TrackNum = "02" },
          new Song { Name = "Angels Fall", Length = "3:48", TrackNum =
"03" },
          new Song { Name = "Breaking the Silence", Length = "3:01", TrackNum
= "04" },
          new Song { Name = "Hollow", Length = "3:51", TrackNum = "05" },
          new Song { Name = "Close to Heaven", Length = "4:09", TrackNum =
"06" },
          new Song { Name = "Bury Me Alive", Length = "4:04", TrackNum =
"07" },
          new Song { Name = "Never Again", Length = "3:43", TrackNum =
"08" },
          new Song { Name = "The Great Divide", Length = "4:12", TrackNum =
"09" },
          new Song { Name = "Ashes of Eden", Length = "4:53", TrackNum =
"10" },
          new Song { Name = "Defeated", Length = "3:25", TrackNum = "11" },
          new Song { Name = "Dawn", Length = "1:52", TrackNum = "12" }
        },
        Name = "Dark Before Dawn", Artist = "Breaking Benjamin", Genre =
"Rock", Year = 2015, TrackTotal = "12", Length = "42:42"
      },
      new Album
      {
        Songs = new List<Song>
        {
          new Song { Name = "Shepherd Of Fire", Length = "5:23", TrackNum =
"01" },
          new Song { Name = "Hail To The King", Length = "5:05", TrackNum =
"02" },
          new Song { Name = "Doing Time", Length = "3:27", TrackNum = "03" },
          new Song { Name = "This Means War", Length = "6:09", TrackNum =
"04" },
          new Song { Name = "Requiem", Length = "4:23", TrackNum = "05" },
                    new Song { Name = "Crimson Day", Length = "4:57", TrackNum =
"06" },
                    new Song { Name = "Heretic", Length = "4:55", TrackNum = "07" },
                    new Song { Name = "Coming Home", Length = "6:26", TrackNum =
"08" },
                    new Song { Name = "Planets", Length = "5:56", TrackNum = "09" },
                    new Song { Name = "Acid Rain", Length = "6:38", TrackNum = "10" },
                    new Song { Name = "St. James", Length = "5:01", TrackNum = "11" }
                },
                Name = "Hail To The King", Artist = "Avenged Sevenfold", Genre =
"Metalcore", Year = 2013, TrackTotal = "11", Length = "58:20"
            }
        };

        public static bool SongDoesExist(String song)
        {
            for (int i = 0; i < library.Count(); i++)
            {
                for (int j = 0; j < library[i].Songs.Count(); j++)
                {
                    if (song.ToLower() == library[i].Songs[j].Name.ToLower())
                    {
                        return true;
                    }
                }
            }
            return false;
        }

        public static bool AlbumDoesExist(String album)
        {
            for (int i = 0; i < library.Count(); i++)
            {
                if (album.ToLower() == library[i].Name.ToLower())
                {
                    return true;
                }
            }
            return false;
        }
    }
```
The “Library” class defines a variable “library” which is a collection of variables of Type “Album”, as well as defines and implements the “SongDoesExist” and “AlbumDoesExist” functions which check whatever variable they are passed to determine if it exists as a song or an album title respectively.

```
public class AlbumCheckEventArgs : EventArgs
    {
        private String _AlbumName;

        public String a_AlbumName
        {
            get { return _AlbumName; }
            set { _AlbumName = value; }
        }

        public AlbumCheckEventArgs()
        {

        }

        public AlbumCheckEventArgs(String
Album)
        {
            a_AlbumName = Album;
        }
    }
```

The “AlbumCheckEventArgs” class defines a custom EventArgs.to be used in the context of albums.

```
public class SongCheckEventArgs : EventArgs
    {
        private String _SongName;

        public String s_SongName
        {
            get { return _SongName; }
            set { _SongName = value; }
        }

        public SongCheckEventArgs()
        {

        }

        public SongCheckEventArgs(String
Song)
        {
            s_SongName = Song;
        }
    }
```

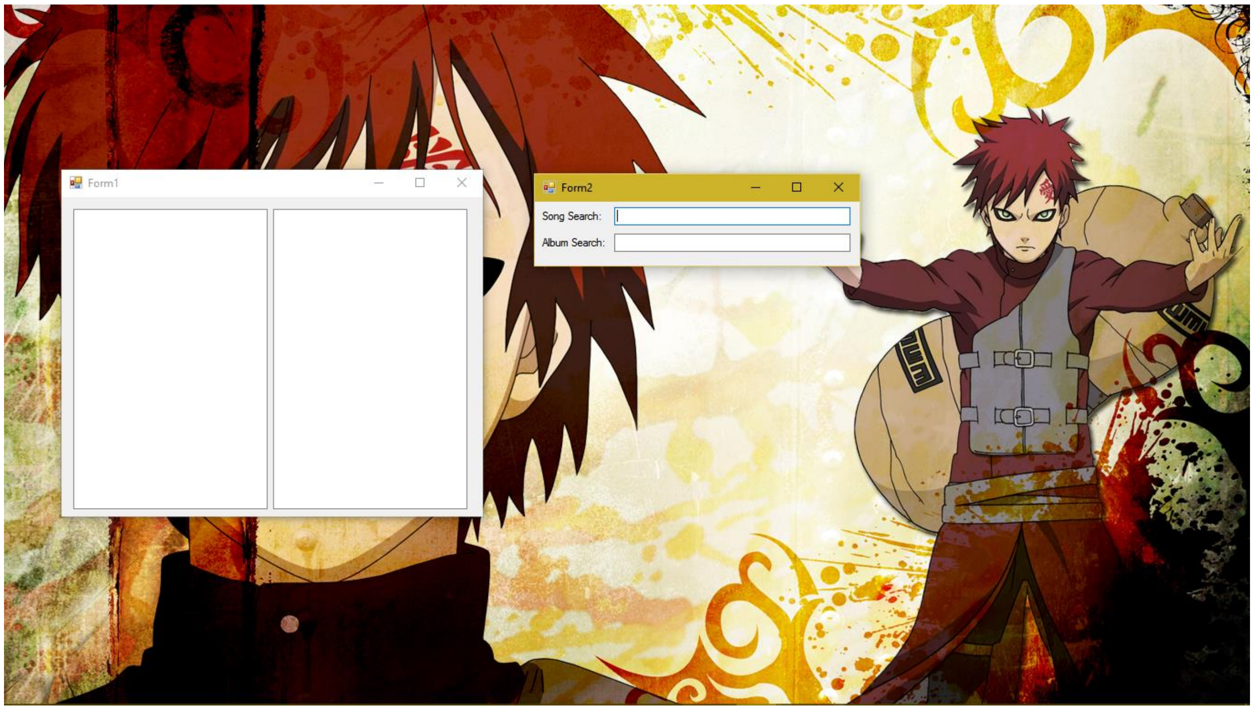
The “SongCheckEventArgs” class defines a custom EventARgs to be used in the context of songs.

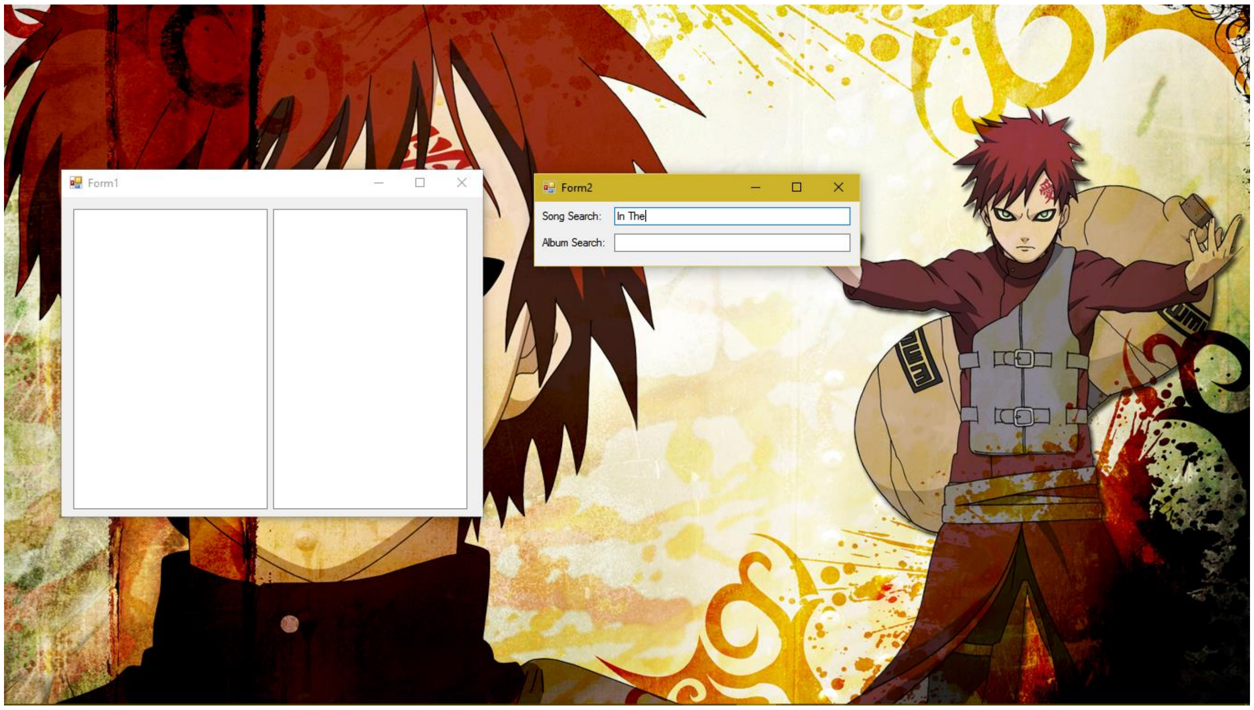
```
public partial class Form2 : Form
    {
        public delegate void SongCheckEventHandler(Object sender, SongCheckEventArgs
e);
        public delegate void AlbumCheckEventHandler(Object sender, AlbumCheckEventArgs
e);

        public SongCheckEventHandler SongExists;
        public SongCheckEventHandler SongDoesNotExist;
        public AlbumCheckEventHandler AlbumExists;
        public AlbumCheckEventHandler AlbumDoesNotExist;

        public Form2()
        {
            InitializeComponent();
        }

        private void songBox_TextChanged(object sender, EventArgs e)
        {
            if (Library.SongDoesExist(songBox.Text) && SongExists != null)
            {
                SongExists(this, new SongCheckEventArgs(songBox.Text));
            }
            else
            {
                SongDoesNotExist(this, new SongCheckEventArgs());
            }
        }

        private void albumBox_TextChanged(object sender, EventArgs e)
        {
            if(Library.AlbumDoesExist(albumBox.Text) && AlbumExists != null)
            {
                AlbumExists(this, new AlbumCheckEventArgs(albumBox.Text));
            }
            else
            {
                AlbumDoesNotExist(this, new AlbumCheckEventArgs());
            }
        }
    }
```

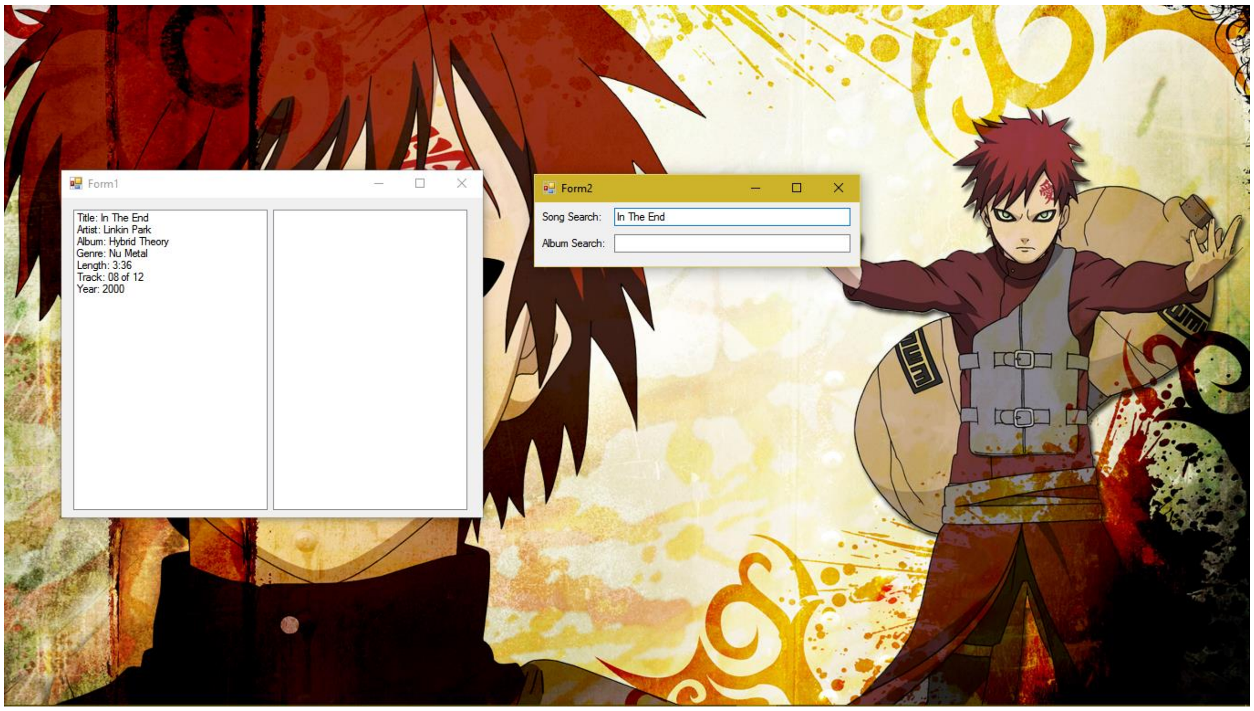
“Form2” contains the “songBox” and “albumBox”, which are used to trigger events. The “songBox” fires a SongCheckEventHandler if the text inside the box is equivalent to the Name of an existing song. Likewise, the “albumBox” fires an AlbumEventCheckHandler if the text inside the box is equivalent to the Name of an existing album. The class also defines what these two types of EventHandlers are, as well as creates one of each.

```
public Form1()
        {
            InitializeComponent();
            Form2 f2 = new Form2();
            f2.Location = new Point(
                            this.Location.X + this.Size.Width + 10,
                            this.Location.Y);
            f2.SongExists += new Form2.SongCheckEventHandler(SongExists);
            f2.SongDoesNotExist += new Form2.SongCheckEventHandler(SongDoesNotExist);
            f2.AlbumExists += new Form2.AlbumCheckEventHandler(AlbumExists);
            f2.AlbumDoesNotExist += new
Form2.AlbumCheckEventHandler(AlbumDoesNotExist);
            f2.Show();
        }

        void SongExists(Object sender, SongCheckEventArgs e)
        {
            for (int i = 0; i < Library.library.Count(); i++)
            {
                for (int j = 0; j < Library.library[i].Songs.Count(); j++)
                {
                    if (e.s_SongName.ToLower() ==
Library.library[i].Songs[j].Name.ToLower())
                    {
                        songResultBox.Items.Add("Title: " +
Library.library[i].Songs[j].Name);
                        songResultBox.Items.Add("Artist: " +
Library.library[i].Artist);
                        songResultBox.Items.Add("Album: " + Library.library[i].Name);
                        songResultBox.Items.Add("Genre: " + Library.library[i].Genre);
                        songResultBox.Items.Add("Length: " +
Library.library[i].Songs[j].Length);
                        songResultBox.Items.Add("Track: " +
Library.library[i].Songs[j].TrackNum + " of " + Library.library[i].TrackTotal);
                        songResultBox.Items.Add("Year: " + Library.library[i].Year);
                    }
                }
            }
        }

        void SongDoesNotExist(Object sender, SongCheckEventArgs e)
        {
            songResultBox.Items.Clear();
        }

        void AlbumExists(Object sender, AlbumCheckEventArgs e)
        {
            for(int i = 0; i < Library.library.Count(); i++)
            {
                if(e.a_AlbumName.ToLower() == Library.library[i].Name.ToLower())
                {
                    albumResultBox.Items.Add("Name: " + Library.library[i].Name);
                    albumResultBox.Items.Add("Artist: " + Library.library[i].Artist);
                    albumResultBox.Items.Add("Genre: " + Library.library[i].Genre);
                    albumResultBox.Items.Add("Number of Tracks: " +
Library.library[i].TrackTotal);
                    albumResultBox.Items.Add("Album Length: " +
Library.library[i].Length);
                    albumResultBox.Items.Add("Year: " + Library.library[i].Year);
                    albumResultBox.Items.Add("Songs:");
                    for (int j = 0; j < Library.library[i].Songs.Count(); j++)
                    {
                        albumResultBox.Items.Add("    " +
Library.library[i].Songs[j].TrackNum + " - " + Library.library[i].Songs[j].Name + " - "
+ Library.library[i].Songs[j].Length);
                    }
                }
            }
        }

        void AlbumDoesNotExist(Object sender, AlbumCheckEventArgs e)
        {
            albumResultBox.Items.Clear();
        }
    }
```

"Form1" creates an instance of "Form2" known as "f2" and subscribes to the events implemented in "Form2." The class then implements 4 functions: "SongExists", "SongDoesNotExist", "AlbumExists", and "AlbumDoesNotExist." These four function modify the "songResultBox" and "albumResultBox" as needed.



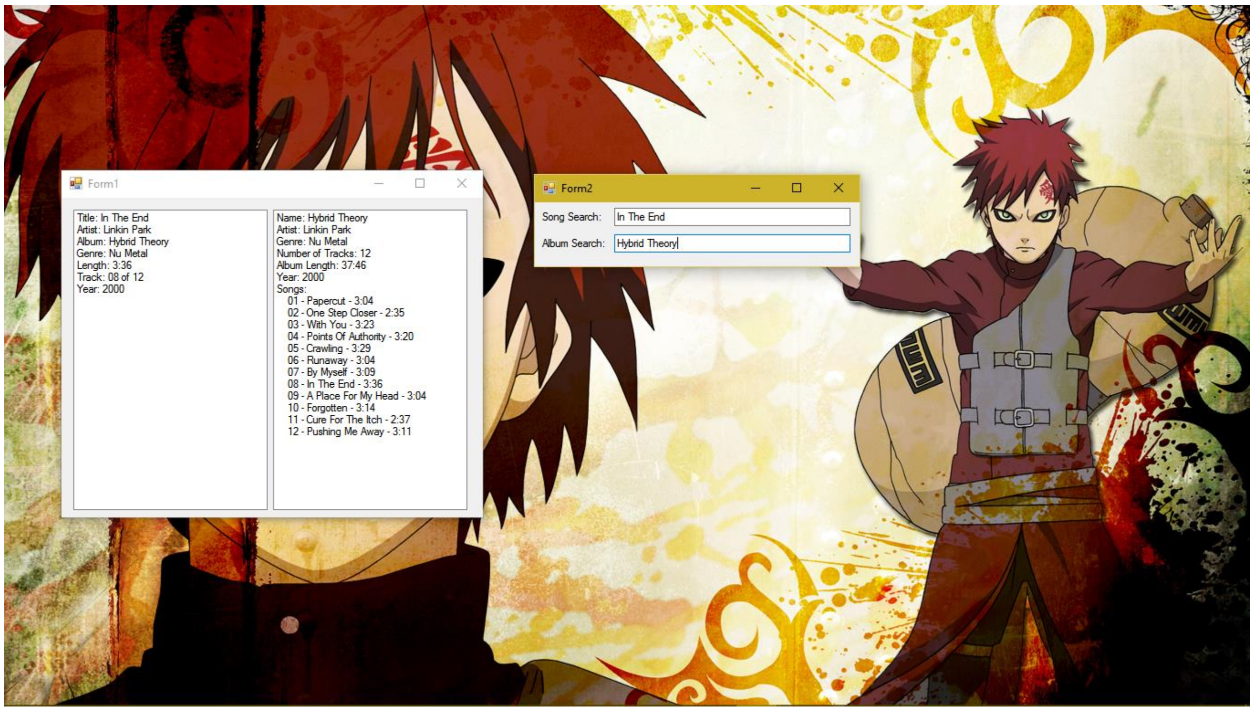




```
Title: In The End
Artist: Linkin Park
Album: Hybrid Theory
Genre: Nu Metal
Length: 3:36
Track: 08 of 12
Year: 2000
```

Song Search: In The End

Album Search:

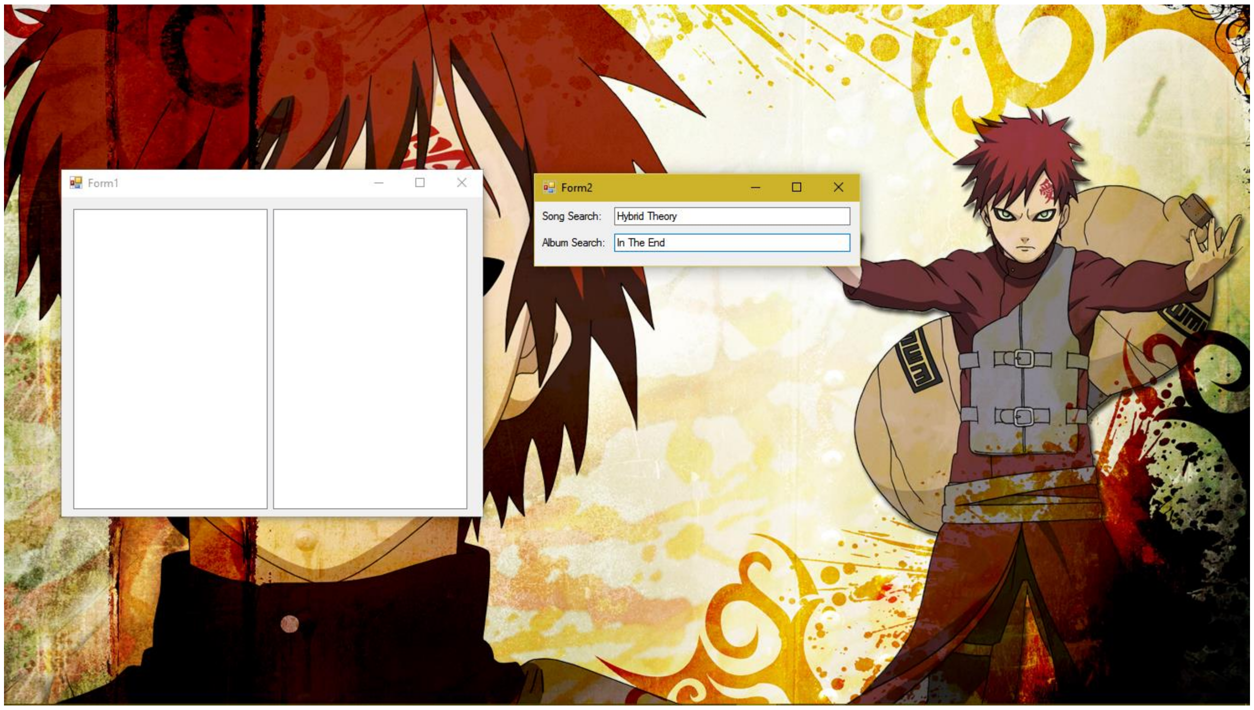


```
Title: In The End
Artist: Linkin Park
Album: Hybrid Theory
Genre: Nu Metal
Length: 3:36
Track: 08 of 12
Year: 2000
```

```
Name: Hybrid Theory
Artist: Linkin Park
Genre: Nu Metal
Number of Tracks: 12
Album Length: 37:46
Year: 2000
Songs:
   01 - Papercut - 3:04
   02 - One Step Closer - 2:35
   03 - With You - 3:23
   04 - Points Of Authority - 3:20
   05 - Crawling - 3:29
   06 - Runaway - 3:04
   07 - By Myself - 3:09
   08 - In The End - 3:36
   09 - A Place For My Head - 3:04
   10 - Forgotten - 3:14
   11 - Cure For The Itch - 2:37
   12 - Pushing Me Away - 3:11
```

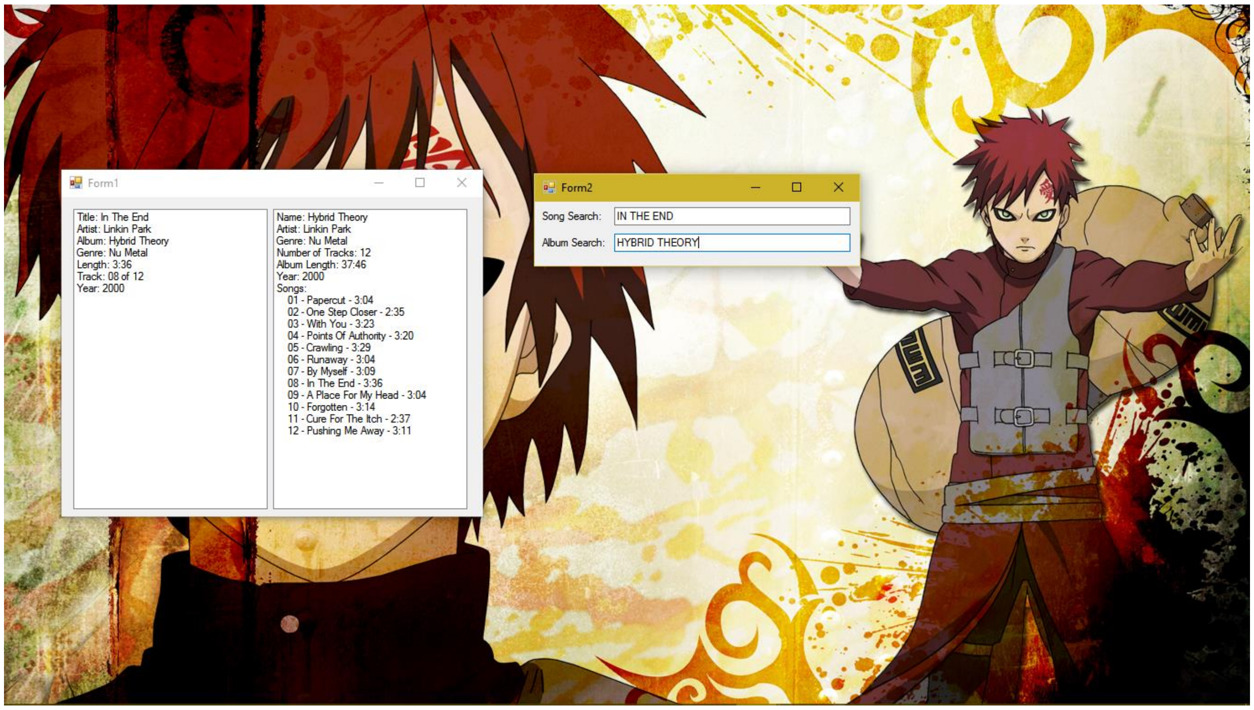
Song Search: In The End

Album Search: Hybrid Theory



Song Search: Hybrid Theory

Album Search: In The End



```
Title: In The End
Artist: Linkin Park
Album: Hybrid Theory
Genre: Nu Metal
Length: 3:36
Track: 08 of 12
Year: 2000
```

```
Name: Hybrid Theory
Artist: Linkin Park
Genre: Nu Metal
Number of Tracks: 12
Album Length: 37:46
Year: 2000
Songs:
   01 - Papercut - 3:04
   02 - One Step Closer - 2:35
   03 - With You - 3:23
   04 - Points Of Authority - 3:20
   05 - Crawling - 3:29
   06 - Runaway - 3:04
   07 - By Myself - 3:09
   08 - In The End - 3:36
   09 - A Place For My Head - 3:04
   10 - Forgotten - 3:14
   11 - Cure For The Itch - 2:37
   12 - Pushing Me Away - 3:11
```

Song Search: IN THE END

Album Search: HYBRID THEORY

Observations:

Overall this was a pretty interesting project. It was harder than I expected only because I had trouble figuring out what to do for the observer pattern. Once I figured out what kind of application I wanted everything fell into place nicely.