**Music Library**
GUI Programming
10 Points

The goal of this assignment is to give you practice with separating interface code (the View) from data storage code (the Model). You are to modify the MiniPlayer example from:

https://github.com/fmccown/MiniPlayerWpf

You are to move the logic used to read, store, add, update, and delete song data into a class called `MusicLib` in order to separate the UI code (View) from the data storage code (Model). If we decide to change how the music data is stored in the future, the UI code will not be affected. If we want to port our UI to Windows Forms or a Universal App, changes will be limited to just the UI code.

`MusicLib` should have a single class-level variable called `musicDataSet`. The class should implement the following:

```
// The constructor reads the music.xsd and music.xml files into the DataSet
public MusicLib()

// Adds a song to the music library and returns the song's ID. The song
// parameter's ID is also set to the auto-generated ID.
public int AddSong(Song s)

// Return a Song for the given song ID or null if no song was not found.
public Song GetSong(int songId)

// Update the given song with the given song ID.  Returns true if the song
// was updated, false if it could not because the song ID was not found.
public bool UpdateSong(int songId, Song song)

// Delete a song given the song's ID. Return true if the song was
// successfully deleted, false if the song ID was not found.
public bool DeleteSong(int songId)

// Save the song database to the music.xml file
public void Save()
```

In the `MusicLib` class, you should create a public, read-only property that returns all the song IDs. The SongIds property can then be used in the MainWindow's constructor when creating an ObservableCollection for populating the song ID combo box.

```
public EnumerableRowCollection<string> SongIds
{
    get
    {
        var ids = from row in musicDataSet.Tables["song"].AsEnumerable()
                  orderby row["id"]
                  select row["id"].ToString();
        return ids;
    }
}
```

Move the `PrintAllTables` function to the `MusicLib` as well.

In the MainWindow class, create a private instance of the `MusicLib` and instantiate it in the form's constructor. Where appropriate, replace the preexisting code in MainWindow with calls to the `MusicLib`'s

methods. When you are finished, the application should work identically to how it worked before, and there should not be any LINQ queries, `DataTable`, `DataRow`, or `DataSet` code in MainWindow.

You should adequately test your new library. Testing it with the GUI is not sufficient since the GUI code will likely always call `AddSong`, `UpdateSong`, and `DeleteSong` with valid song IDs. You should try calling these functions with bad IDs to verify they perform as required.

Submit your **MusicLib.cs** and **MiniPlayer.exe** to Easel before the next class period. I will replace my MusicLib.cs with yours to test it.

To help you get started, here's what the Add song button's code currently does:

```
private void addButton_Click(object sender, RoutedEventArgs e)
{

    // Add the selected file to the song table
    DataTable table = musicDataSet.Tables["song"];
    DataRow row = table.NewRow();

    row["title"] = titleTextBox.Text;
    row["artist"] = artistTextBox.Text;
    row["album"] = albumTextBox.Text;
    row["filename"] = filenameTextBox.Text;
    row["length"] = lengthTextBox.Text;
    row["genre"] = genreTextBox.Text;
    table.Rows.Add(row);

    // Now that the id has been set, add it to the combo box
    songIdComboBox.IsEnabled = true;
    string id = row["id"].ToString();
    (songIdComboBox.ItemsSource as ObservableCollection<string>).Add(id);
    songIdComboBox.SelectedIndex = songIdComboBox.Items.Count - 1;

    // There is at least one song that can be deleted
    deleteButton.IsEnabled = true;
}
```

You will remove the ADO.NET code and modify the function to call the `MusicLib`'s `AddSong` method like so:

```
private void addButton_Click(object sender, RoutedEventArgs e)
{
    // Add the selected file to the music library
    Song s = new Song
    {
        Title = titleTextBox.Text,
        Artist = artistTextBox.Text,
        Album = albumTextBox.Text,
        Genre = genreTextBox.Text,
        Length = lengthTextBox.Text,
        Filename = filenameTextBox.Text
    };

    string id = musicLib.AddSong(s).ToString();

    // Add the song ID to the combo box
    songIdComboBox.IsEnabled = true;
```

```
            (songIdComboBox.ItemsSource as ObservableCollection<string>).Add(id);
            songIdComboBox.SelectedIndex = songIdComboBox.Items.Count - 1;

            // There is at least one song that can be deleted
            deleteButton.IsEnabled = true;
        }
```

The AddSong function will contain the ADO.NET code and look like this:

```
        public int AddSong(Song s)
        {
            DataTable table = musicDataSet.Tables["song"];
            DataRow row = table.NewRow();

            row["title"] = s.Title;
            row["artist"] = s.Artist;
            row["album"] = s.Album;
            row["filename"] = s.Filename;
            row["length"] = s.Length;
            row["genre"] = s.Genre;
            table.Rows.Add(row);

            // Update this song's ID
            s.Id = Convert.ToInt32(row["id"]);

            return s.Id;
        }
```