Join and Unions Lab

Lab 1: Inner Join

Right mouse-click on the **EntityClasses** folder.

Create a new class named **SongGenre** with the following code.

```
#nullable disable

using System.Text;

namespace LINQLab.EntityClasses;

public partial class SongGenre
{
  public int SongId { get; set; }
  public string SongName { get; set; }
  public string Artist { get; set; }
  public string Album { get; set; }
  public int? GenreId { get; set; }
  public string GenreName { get; set; }

  #region ToString Override
  public override string ToString() {
    StringBuilder sb = new(1024);

    sb.AppendLine($"Song: {SongName}  ID: {SongId}");
    sb.AppendLine($"  Artist: {Artist}");
    sb.AppendLine($"  Album: {Album}");
    sb.AppendLine($"  Genre: {GenreName}");

    return sb.ToString();
  }
  #endregion
}
```

Open the **Program.cs** file and replace the code with the following.

```
using LINQLab.EntityClasses;
using LINQLab.RepositoryClasses;

// Declare variables and fill data
List<Song> songs = SongRepository.GetAll();
List<MusicGenre> genres = MusicGenreRepository.GetAll();
List<SongGenre> songGenres = new();

// TODO: Write Your Query Here


// Display the Results
foreach (var song in songGenres) {
  System.Diagnostics.Debug.WriteLine(song);
}
```

Write a query to join the Song list with the Genre list and create a new SongGenre object with the appropriate data from each of the corresponding properties of the SongGenre object.

Try it Out

Run the application

Select **View | Output** from the Visual Studio menu

View the results in the Debug window

```
Output
Show output from: Debug
Song: Friday Is Forever  ID: 6181
  Artist: We The Kings
  Album: Sunshine State of Mind
  Genre: Alternative

Song: Say You Like Me  ID: 6182
  Artist: We The Kings
  Album: Sunshine State of Mind
  Genre: Alternative

Song: Secret Valentine  ID: 6183
  Artist: We the Kings
  Album: We the Kings
  Genre: Alternative

Song: Troublemaker  ID: 6184
  Artist: Weezer
  Album: Weezer (Red Album) [Deluxe Edition]
  Genre: Alternative

Song: This Is War  ID: 6285
  Artist: 30 Seconds to Mars
  Album: This Is War
  Genre: Alternative

The program '[20792] LINQLab.exe' has exited
```

Lab 2: One to Many

Right mouse-click on the **EntityClasses** folder.

Create a new class named **GenreSongs** with the following code.

```
#nullable disable

using System.Text;

namespace LINQLab.EntityClasses;

public partial class GenreSongs
{
  public int GenreId { get; set; }
  public string GenreName { get; set; }
  public List<Song> Songs { get; set; }

  #region ToString Override
  public override string ToString() {
    StringBuilder sb = new(1024);

    sb.AppendLine($"Genre: {GenreName}  Genre ID: {GenreId}");
    foreach (var song in Songs) {
      sb.AppendLine($"  {song.SongName} - {song.Artist}");
    }

    return sb.ToString();
  }
  #endregion
}
```

Open the **Program.cs** and replace the code with the following:

```
using LINQLab.EntityClasses;
using LINQLab.RepositoryClasses;

// Declare variables and fill data
List<Song> songs = SongRepository.GetAll();
List<MusicGenre> genres = MusicGenreRepository.GetAll();
List<GenreSongs> genreSongs = new();

// TODO: Write Your Query Here


// Display the Results
foreach (var song in genreSongs) {
  System.Diagnostics.Debug.WriteLine(song);
}
```

Write a query to get all genres ordering by the ***GenreId***. Join these to the songs collection where the ***GenreId*** equals the ***GenreId*** in the genre collection. Use the **into** keyword to get the list of songs for each genre. Create a new ***GenreSongs*** object and put the data from songs and genres into this new object.

Try it Out

Run the application and view the results:

```
Output
Show output from: Debug
   Troublemaker - Weezer
   Tsunami - Frame Of Mind
   Two Is Better Than One (feat. Taylor Swift
   Uncertain Times - Frame Of Mind
   Up All Night - Counting Crows
   Valentino - Bowling for Soup
   Waiting - Green Day
   Wake Me Up When September Ends - Green Day
   Walking Contradiction - Green Day
   Warning - Green Day
   Weight of the World - Evanescence
   Weightless - All Time Low
   Welcome to Paradise - Green Day
   Welcome to the Black Parade - My Chemical
   When I Come Around - Green Day
   Why Should You Come When I Call - Counting
   Wisk - Bowling for Soup
   World Turns - Secondhand Serenade
   You and I - Secondhand Serenade
   Your Own Eyes - Frame Of Mind
   Your Star - Evanescence

 Genre: Death Metal  Genre ID: 44

 The program '[15992] LINQLab.exe' has exited
```

Lab 3: Union using Comparer

Open the **Program.cs** file and replace the code with the following:

```
using LINQLab.EntityClasses;
using LINQLab.RepositoryClasses;

// Declare variables and fill data
List<Song> songs1 = SongRepository.GetAll()
                    .Where(row => row.GenreId == 1).ToList();
List<Song> songs2 = SongRepository.GetAll()
                    .Where(row => row.GenreId == 2).ToList();
SongComparer sc = new();
List<Song> songs = new();

// TODO: Write Your Query Here


// Display the Results
foreach (var song in songs) {
  System.Diagnostics.Debug.WriteLine(song);
}

// Display count
System.Diagnostics.Debug.WriteLine($"Total Count: {songs.Count}");
```

Write a query to union the songs in the two lists together.

Use the **SongComparer** class to ensure no duplicates are allowed.

Order by the song name.

Try it Out

Run the application and view the results:

```
Output
Show output from: Debug
  Album: A Twist In My Story
  Year: 2008

Song: Your Decision  ID: 165
  Artist: Alice In Chains
  Album: Black Gives Way to Blue (Bonus Trac
  Year: 2009

Song: Your Latest Trick (Live)  ID: 4016
  Artist: Dire Straits
  Album: Sultans of Swing, The Very Best of
  Year: 1998

Song: Your Number Or Your Name  ID: 3685
  Artist: The Knack
  Album: My Sharona
  Year: 1992

Song: YYZ  ID: 1486
  Artist: Rush
  Album: Moving Pictures
  Year: 1981

Total Count: 1840
The program '[23660] LINQLab.exe' has exited
```

Lab 4: UnionBy

Open the **Program.cs** file and remove the declaration for the SongComparer.

Write a query to union the two lists together using the UnionBy() method and the appropriate lambda expression.

Try it Out

Run the application and view the results:

```
Output
Show output from: Debug
  Album: A Twist In My Story
  Year: 2008

Song: Your Decision  ID: 165
  Artist: Alice In Chains
  Album: Black Gives Way to Blue (Bonus Trac
  Year: 2009

Song: Your Latest Trick (Live)  ID: 4016
  Artist: Dire Straits
  Album: Sultans of Swing, The Very Best of
  Year: 1998

Song: Your Number Or Your Name  ID: 3685
  Artist: The Knack
  Album: My Sharona
  Year: 1992

Song: YYZ  ID: 1486
  Artist: Rush
  Album: Moving Pictures
  Year: 1981

Total Count: 1840
The program '[7192] LINQLab.exe' has exited
```

Join and Unions Lab Answers

Lab 1: Inner Join (Query Syntax)

```
songGenres = (from row in songs
              join genreRow in genres
              on row.GenreId equals genreRow.GenreId
              select new SongGenre {
                SongId = row.SongId,
                SongName = row.SongName,
                Artist = row.Artist,
                Album = row.Album,
                GenreId = row.GenreId,
                GenreName = genreRow.Genre
              }).OrderBy(row => row.GenreId).ToList();
```

Lab 1: Inner Join (Method Syntax)

```
songGenres = songs.Join(genres,
              row => row.GenreId,
              genreRow => genreRow.GenreId,
              (row, genreRow) => new SongGenre {
                SongId = row.SongId,
                SongName = row.SongName,
                Artist = row.Artist,
                Album = row.Album,
                GenreId = row.GenreId,
                GenreName = genreRow.Genre
              }).OrderBy(row => row.GenreId).ToList();
```

Lab 2: One to Many (Query Syntax)

```
genreSongs = (from row in genres
              orderby row.GenreId
              join songRow in songs
              on row.GenreId equals songRow.GenreId
              into songsByGenre
              select new GenreSongs {
                GenreId = row.GenreId,
                GenreName = row.Genre,
                Songs = songsByGenre.OrderBy(song =>
                          song.SongName).ToList()
              }).ToList();
```

Lab 2: One to Many (Method Syntax)

```
genreSongs = genres
   .OrderBy(row => row.GenreId)
   .GroupJoin(songs,
              row => row.GenreId,
              songRow => songRow.GenreId,
              (row, songsByGenre) => new GenreSongs {
                GenreId = row.GenreId,
                GenreName = row.Genre,
                Songs = songsByGenre.OrderBy(song =>
                          song.SongName).ToList()
   }).ToList();
```

Lab 3: Union using Comparer (Query Syntax)

```
songs = (from row in songs1
         select row)
         .Union(songs2, sc)
         .OrderBy(row => row.SongName).ToList();
```

Lab 3: Union using Comparer (Method Syntax)

```
songs = songs1
        .Union(songs2, sc)
        .OrderBy(row => row.SongName).ToList();
```

Lab 4: UnionBy() Method Query Syntax)

```
songs = (from row in songs1
         select row)
         .UnionBy(songs2, row => row.SongId)
         .OrderBy(row => row.SongName).ToList();
```

Lab 4: UnionBy() Method (Method Syntax)

```
songs = songs1.UnionBy(songs2, row => row.SongId)
        .OrderBy(row => row.SongName).ToList();
```