МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт–Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

ЛАБОРАТОРНАЯ РАБОТА № 2

Выполнил студент:

Ершов Михаил Николаевич
группа: M3206

Санкт-Петербург, 2019 г.

Вторая задача ООП

Реализовать Каталог музыки.

Есть Артисты, они выпускают Альбомы, в которых содержатся Песни (или треки).

Также есть музыкальные жанры. У жанров есть поджанры (например, Тяжелый рок - это поджанр Рока)

Кроме альбомов, в каталоге содержатся Сборники - в сборниках содержатся треки разных артистов, возможно, разных жанров.

Спроектировать и реализовать классовую модель такого Каталога, и сделать в нём небольшой поисковый движок. Например, чтобы можно было найти все альбомы и сборники, содержащие треки жанра Рок, вышедшие в 2017 году (при этом должны найтись и альбомы жанра Тяжелый рок, как поджанра Рока). Движок должен уметь искать а) артистов, б) альбомы и сборники, в) песни по ряду критериев (критерии определите сами)

Для реализации вам нужно продумать, какими свойствами должен обладать тот или иной объект (например, у песни точно есть название). Кроме того, можно задать некоторые ограничения для упрощения работы. Я приведу примеры таких ограничений (вы можете им следовать или нет, по желанию)

* Вы можете задать определенный жанр (или жанры) Артисту и считать, что все его треки сделаны в этом жанре. Или же считать, что каждый трек может иметь свой жанр (или альбом, или ещё как-то)

* Считаем, что артисты не могут выпустить альбом вместе (один альбом - один артист)

* Можно считать, что трек не существует вне альбома. То есть если трек выпущен, он должен присутствовать хотя бы в одном альбоме, даже если там этот трек - единственный (такие альбомы называются синглами)

* Жанры можно представить в виде дерева (в этом случае у каждого поджанра может быть только один предок). Но можно реализовать и другие отношения (например, считать, что жанр Поп-рок имеет родительские жанры Рок и Поп). Так или иначе, нужно реализовать минимум два уровня жанров (то есть базовые родительские, и поджанры)

* Можем считать, что Сборник - это произвольный набор треков из вышедших ранее альбомов (разных исполнителей)

Кроме этих ограничений вы можете придумать свои, если вы посчитаете их необходимыми (но помните, что преподаватель может вас попросить их ослабить)

Общие пожелания по реализации:

* Пожалуйста, не выдумывайте искусственных идентификаторов объектов, работайте с объектами напрямую. Помните, что переменная хранит только ссылку на существующий где-то в памяти объект, то есть по сути она хранит его идентификатор.

* Поисковый движок можно сделать двумя способами - либо набором конкретных методов, либо также написать его в объектном стиле, сделав Конструктор запросов

* Если вы хотите что-то добавить в объектную модель, или добавить свои ограничения, но в чем-то не уверены, спросите преподавателя. Он с радостью выслушает любые ваши идеи. То же самое касается вопросов по реализации (если вы зашли в тупик)

1 Program.cs

```
using System;
using System.Collections.Generic;

namespace lab2v2 {
    class Program {
        static void Main(string[] args) {
            Artist artist = new Artist("tehnarenok");
            Genre first_genre = new Genre("tehnarenokGenre");
            Genre second_genre = new Genre("tehnarenokGenre2");
            Song first_song = new Song("wrhwqr", artist, first_genre);
            Song second_song = new Song("wrhgfrwyq", artist, second_genre);
            Storage storage = new Storage();
            storage.AddArtist(artist);

            storage.PrintAll();

            var songs = storage.FindSong(name: "q", artistsNames: new[] {"
                tehnarenok"}, genresNames: new[] {"tehnarenokGenre2"});

            Console.WriteLine("Print find songs:");
            foreach(var song in songs) {
                Console.WriteLine($"{song.Name}");
            }
        }
    }
}
```

2 Album.cs

```
using System;
using System.Collections.Generic;

class Album {
    public string Name;

    public HashSet<Artist> Artists = new HashSet<Artist>();
    public HashSet<Song> Songs = new HashSet<Song>();
    public HashSet<Genre> Genres = new HashSet<Genre>();

    public Album(string name) {
        Name = name;
    }

    public Album(string name, Song song) {
        Name = name;
        Songs.Add(song);
        foreach(var item in song.Artists) {
            Artists.Add(item);
        }
        Genres.Add(song.Genre);
    }

    public void AddSong(Song song) {
        Songs.Add(song);
        foreach(var item in song.Artists) {
            Artists.Add(item);
        }
        Genres.Add(song.Genre);
    }
}
```

3 Artist.cs

```
using System.Collections.Generic;

class Artist {
    public string Name;
    public string _id;

    public HashSet<Song> Songs = new HashSet<Song>();
    //public HashSet<Genre> Genres = new HashSet<Genre>();

    public Artist(string name) {
        Name = name;
        _id = Name + "isartist";
    }

    public void AddSong(Song song) {
        Songs.Add(song);
    }
}
```

4 Collection.cs

```
using System;
using System.Collections.Generic;

class Collection {
    public string Name;

    public HashSet<Artist> Artists = new HashSet<Artist>();
    public HashSet<Song> Songs = new HashSet<Song>();
    public HashSet<Genre> Genres = new HashSet<Genre>();

    public Collection(string name) {
        Name = name;
    }

    public Collection(string name, Song song) {
        Name = name;
        Songs.Add(song);
        foreach(var item in song.Artists) {
            Artists.Add(item);
        }
        Genres.Add(song.Genre);
    }

    public void AddSong(Song song) {
        Songs.Add(song);
        foreach(var item in song.Artists) {
            Artists.Add(item);
        }
        Genres.Add(song.Genre);
    }
}
```

5 Genre.cs

```
using System.Collections.Generic;

class Genre {
    public HashSet<Genre> parentGenres = new HashSet<Genre>();
    public HashSet<Genre> childGenres = new HashSet<Genre>();

    public string _id;

    public string Name;

    public Genre(string name) {
        Name = name;
        _id = Name + "isgenre";
    }

    public Genre(string name, List<Genre> parents) {
        Name = name;
        foreach(var item in parents) {
            parentGenres.Add(item);
        }
        //parentGenres = parents;
        _id = Name + "isgenre";
    }

    public Genre(string name, List<Genre> parents, List<Genre> children) {
        Name = name;
        foreach(var item in parents) {
            parentGenres.Add(item);
        }
        foreach(var item in children) {
            childGenres.Add(item);
        }
        //parentGenres = parents;
        //childGenres = children;
        _id = Name + "isgenre";
    }

    public void AddParentGenre(Genre genre) {
        parentGenres.Add(genre);
    }

    public void AddParentGenre(List<Genre> genres) {
        foreach(var item in genres) {
            parentGenres.Add(item);
        }
    }

    public void AddChildGenre(Genre genre) {
        childGenres.Add(genre);
    }

    public void AddChildGenre(List<Genre> genres) {
        foreach(var item in genres) {
            childGenres.Add(item);
```

```
        }
    }

    public List<Genre> GetChildGenres() {
        List<Genre> AllChildGenres = new List<Genre>() { this };
        foreach(var item in childGenres) {
            AllChildGenres.AddRange(item.GetChildGenres());
        }
        return(AllChildGenres);
    }
}
```

6 Song.cs

```
using System;
using System.Collections.Generic;

class Song {
    public string Name;
    public Genre Genre;
    public HashSet<Artist> Artists = new HashSet<Artist>();
    public string _id;

    public Song(string name, Artist artist, Genre genre) {
        Name = name;
        Genre = genre;
        Artists.Add(artist);
        _id = Name + "issong";
        artist.AddSong(this);
    }

    public Song(string name, List<Artist> artists, Genre genre) {
        Name = name;
        Genre = genre;
        foreach(var item in artists) {
            Artists.Add(item);
        }
        _id = Name + "issong";
        foreach (var item in artists) {
            item.AddSong(this);
        }
    }
}
```

7 Storage.cs

```
using System;
using System.Linq;
using System.Collections.Generic;

class Storage {
    private HashSet<Song> AllSongs = new HashSet<Song>();
    private HashSet<Artist> AllArtists = new HashSet<Artist>();
    private HashSet<Album> AllAlbums = new HashSet<Album>();
    private HashSet<Collection> AllCollections = new HashSet<Collection>();
    private HashSet<Genre> AllGenres = new HashSet<Genre>();

    public void AddSong(Song song) {
        AllSongs.Add(song);
        foreach(var item in song.Artists) {
            AllArtists.Add(item);
        }
        AllGenres.Add(song.Genre);
    }

    public void AddArtist(Artist artist) {
        AllArtists.Add(artist);
        foreach(var song in artist.Songs) {
            this.AddSong(song);
        }
    }

    public void AddAlbum(Album album) {
        AllAlbums.Add(album);
        foreach(var song in album.Songs) {
            this.AddSong(song);
        }
    }

    public void AddCollection(Collection collection) {
        AllCollections.Add(collection);
        foreach(var song in collection.Songs) {
            this.AddSong(song);
        }
    }

    private List<Song> FindSongs(HashSet<Song> songs, string name = "", List<
       Genre> genres = null, List<Artist> artists = null) {
        List<Genre> allGenres = new List<Genre>();

        if(genres != null) {
            foreach(var item in genres) {
                allGenres.AddRange(item.GetChildGenres());
            }

            genres = allGenres;
        }

        var shouldLookupGenres = genres != null && genres.Count > 0;
        var shouldLookupArtists = artists != null && artists.Count > 0;
```

```
    return songs
        .Where(song => song.Name.IndexOf(name) != -1)
        .Where(song => shouldLookupGenres ? genres.Any(genre => genre ==
            song.Genre) : true)
        .Where(song => shouldLookupArtists ? song.Artists.Any(artist =>
            artists.Contains(artist)) : true)
        .ToList();
}

private List<Artist> ArtistName(IEnumerable<string> artistsNames) {
    List<Artist> artists = new List<Artist>();

    if(artistsNames != null) {
        foreach(var artist in AllArtists) {
            foreach(var artistName in artistsNames) {
                if(artist.Name.ToLower().IndexOf(artistName.ToLower()) !=
                    -1) {
                    artists.Add(artist);
                }
            }
        }
    } else {
        artists = null;
    }

    return artists;
}

private List<Genre> GenreName(IEnumerable<string> genresNames) {
    List<Genre> genres = new List<Genre>();

    if(genresNames != null) {
        foreach(var item in AllGenres) {
            foreach(var genreName in genresNames) {
                if(item.Name.ToLower().IndexOf(genreName.ToLower()) != -1) {
                    genres.Add(item);
                }
            }
        }
    } else {
        genres = null;
    }

    return genres;
}

private HashSet<Song> AlbumName(IEnumerable<string> albumsNames) {
    List<Album> albums = new List<Album>();

    if(albumsNames != null){
        foreach(var item in AllAlbums) {
            foreach(var albumName in albumsNames) {
                if(item.Name.ToLower().IndexOf(albumName.ToLower()) != -1) {
                    albums.Add(item);
                }
```

```
            }
        }
    }

    HashSet<Song> songs = new HashSet<Song>();
    foreach(var album in albums) {
        foreach(var song in album.Songs) {
            songs.Add(song);
        }
    }

    return songs;
}

public List<Song> FindSong(string name = "", IEnumerable<string> genresNames
    = null, IEnumerable<string> artistsNames = null) {
    return(this.FindSongs(AllSongs, name, GenreName(genresNames), ArtistName
        (artistsNames)));
}

public List<Song> FindSongInAlbum(string albumName, string name = "", List<
    string> genresNames = null, List<string> artistsNames = null) {

    return(this.FindSongs(AlbumName(new List<string>{albumName}), name,
        GenreName(genresNames), ArtistName(artistsNames)));
}

public List<Song> FindSongInAlbum(List<string> albumsNames, string name =
    "", List<string> genresNames = null, List<string> artistsNames = null) {
    // HashSet<Song> songs = new HashSet<Song>();
    // foreach(var album in albums) {
    //     foreach(var song in album.Songs) {
    //         songs.Add(song);
    //     }
    // }
    return(this.FindSongs(AlbumName(albumsNames), name, GenreName(
        genresNames), ArtistName(artistsNames)));
}

// public List<Song> FindSongInCollection(Collection collection, string name
    = "", List<Genre> genres = null, List<Artist> artists = null) {
//     return(this.FindSongs(collection.Songs, name, GenreName(genresNames),
    ArtistName(artistsNames)));
// }

// public List<Song> FindSongInCollection(List<Collection> collections,
    string name = "", List<Genre> genres = null, List<Artist> artists = null)
    {
//     HashSet<Song> songs = new HashSet<Song>();
//     foreach(var collection in collections) {
//         foreach(var song in collection.Songs) {
//             songs.Add(song);
//         }
//     }
//     return(this.FindSongs(songs, name, GenreName(genresNames), ArtistName
    (artistsNames)));
```

```
    // }

    // public List<Artist> FindArtists(string name = "", List<Song> songs = null
       ) {
    //     List<Artist> artists = new List<Artist>();

    //     foreach(var artist in AllArtists) {
    //         var flag = true;
    //         if(artist.Name.IndexOf(name) == -1) {
    //             flag = false;
    //         }
    //         if(songs != null) {
    //             var flag_1 = false;
    //             foreach(var mysong in songs) {
    //                 foreach(var song in artist.Songs) {
    //                     if(song == mysong) {
    //                         flag_1 = true;
    //                     }
    //                 }
    //             }
    //             flag = flag && flag_1;

    //         }
    //         if(flag)
    //     }
    // }

    public void PrintAll() {
        Console.WriteLine("Songs:");
        foreach(var item in AllSongs) {
            Console.WriteLine($"\t{item.Name}");
        }
        Console.WriteLine("Artists:");
        foreach(var item in AllArtists) {
            Console.WriteLine($"\t{item.Name}");
        }
        Console.WriteLine("Genres:");
        foreach(var item in AllGenres) {
            Console.WriteLine($"\t{item.Name}");
        }
        Console.WriteLine("Albums:");
        foreach(var item in AllAlbums) {
            Console.WriteLine($"\t{item.Name}");
        }
        Console.WriteLine("Collection:");
        foreach(var item in AllCollections) {
            Console.WriteLine($"\t{item.Name}");
        }
        Console.WriteLine("End print");
    }
}
```