زنگ سی‌شارپ – قسمت بیست و نهم

نوشته‌ی مسعود درویشیان

لینک مستقیم این مطلب در وب‌تارگت

در قسمت قبل با ساختار کلی برنامه آشنا شدید، در این قسمت به ادامه‌ی مبحث جلسه‌ی قبل و تکمیل تمرین شماره‌ی ۱۴ می‌پردازیم. اگر به‌یاد داشته باشید در قسمت قبل یک‌سری کلاس تعریف کردیم که در این برنامه از آن‌ها استفاده خواهیم کرد. در در ادامه یک جمع‌بندی از کلاس‌هایی که قرار است از آن‌ها استفاده کنیم را مشاهده می‌کنید.

کلاس Tune: این کلاس شامل یک‌سری فیلد برای ذخیره اطلاعات مربوط به آهنگ و دو constructor است. این‌طور در نظر گرفته‌ایم که دو نوع آهنگ می‌تواند وجود داشته باشد: تک‌آهنگ و آهنگ آلبوم. بنابراین یک constructor برای ساخت تک‌آهنگ و یک constructor برای ساخت آهنگ آلبوم در نظر گرفته‌ایم.

```
class Tune
{
    private string TuneName;
    private string Composer;
    private string Songwriter;
    private string Arrangement;
    private string TuneOwner;
    private string TuneGenre;
    private ushort TuneYear;

    // Constructor for album tunes
    public Tune(
        string tuneName,
        string tuneGenre,
        string composer,
        string songwriter,
        string arrengement
        )
    {
        TuneName = tuneName;
        TuneGenre = tuneGenre;
        Composer = composer;
        Songwriter = songwriter;
        Arrangement = arrengement;
    }

    // Constructor for single tunes
    public Tune(
        string tuneName,
        string tuneGenre,
        string composer,
        string songwriter,
        string arrengement,
        ushort tuneYear,
        Artist artist
```

```
        )
        : this(tuneName, tuneGenre, composer, songwriter, arrengement)
    {
        TuneOwner = artist.GetArtistName();
        TuneYear = tuneYear;
    }
}
```

کلاس Album: این کلاس شامل یک‌سری فیلد برای نگهداری اطلاعات مربوط به آلبوم مثل نام، سبک، صاحب اثر و...، یک constructor و دو متد است. این کلاس همچنین شامل آرایه‌ای از کلاس Tune برای نگهداری آهنگ‌های آلبوم است.

```
class Album
{
    // Fields
    private string AlbumName;
    private string AlbumOwner;
    private string AlbumGenre;
    private ushort AlbumYear;
    private Tune[] Tunes;

    // Constructor
    public Album(string albumName, Artist artist,
        string albumGenre, ushort albumYear, Tune[] tunes)
    {
        AlbumName = albumName;
        AlbumOwner = artist.GetArtistName();
        AlbumGenre = albumGenre;
        AlbumYear = albumYear;
        Tunes = tunes;
    }

    // Methods
    public void AddTune()
    {
        // ...
    }
    public void RemoveTune()
    {
        //...
    }
}
```

کلاس Artist: این کلاس شامل فیلد، یک‌سری متد و constructor است. این کلاس همچنین شامل آرایه‌ای از Tune و آرایه‌ای از Album است. آرایه‌ی Tune به‌منظور ذخیره تک‌آهنگ‌ها و آرایه‌ی Album به‌منظور نگهداری آلبوم‌های خواننده‌ی مورد نظر است.

```
class Artist
{
    // Fields
    private string ArtistName;
    private string ArtistFamily;
```

```
    private Album[] Albums;
    private Tune[] SingleTunes;

    // Constructor
    public Artist(string artistName, string artistFamily)
    {
        ArtistName = artistName;
        ArtistFamily = artistFamily;

        Albums = new Album[5];
        SingleTunes = new Tune[10];
    }

    // Methods
    public string GetArtistNameAndFamily()
    {
        return ArtistName + " " + ArtistFamily;
    }

    public void AddAlbum()
    {
        // ...
    }

    public void RemoveAlbum()
    {
        // ...
    }
    public void AddSingleTune()
    {
        // ...
    }
    public void RemoveSingleTrack()
    {
        // ...
    }
}
```

کلاس MusicBox: این کلاس شامل آرایه‌ای از Artist، یک constructor و تعدادی متد است.

```
class MusicBox
{
    // Fields
    public Artist[] Artists;

    // Constructor
    public MusicBox(ushort size)
    {
        Artists = new Artist[size];
    }

    // Methods
    public void AddArtist()
    {
        // ...
    }
    public void RemoveArtist()
    {
        // ...
    }
```

```
    public void ShowArtists()
    {
        // ...
    }
}
```

کلاس UI: علت به‌وجود آوردن این کلاس این است که در این‌جا به عبارتی engine برنامه را (تا حدودی) جدا در نظر بگیریم. یک Music Player واقعی را تصور کنید، آیا شما وقتی‌که دکمه خاموش/روشن را فشار می‌دهید تا دستگاه خاموش/روشن شود از اتفاقاتی که پشت دکمه می‌افتد با خبر هستید؟ مسلماً نه! در این‌جا نیز قضیه به همین صورت است. همیشه سعی کنید تا آن‌جا که می‌توانید معماری و اصول شی‌گرایی را رعایت کنید. در مقالات زنگ سی‌شارپ تا حدودی با این اصول و قواعد آشنا خواهید شد.

```
class UI
{
    // Fields
    MusicBox MyMusicBox;

    // Constructor
    public UI()
    {
        // A music box with 10 artists.
        MyMusicBox = new MusicBox(10);
    }

    // Methods
    public string ShowMenu()
    {
        Console.Clear();
        Console.WriteLine();
        Console.WriteLine("╔═══════ Music Box ═══════╗");
        Console.WriteLine("║                         ║");
        Console.WriteLine("║ 1. Add Artist           ║");
        Console.WriteLine("║ 2. Show Artists         ║");
        Console.WriteLine("║ 3. Exit                 ║");
        Console.WriteLine("║                         ║");
        Console.WriteLine("╚═════════════════════════╝");
        Console.WriteLine();
        Console.WriteLine("▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒");
        Console.WriteLine();
        Console.Write("Pick out a number: ");
        return Console.ReadLine();
    }

    public void Proccess(string choice)
    {
        switch (choice)
        {
            case "1":
                // Add Artist
                break;
            case "2":
                // show Artists
                break;
```

```
            case "3":
                Environment.Exit(0);
                // Exit
                break;
            default:
                Console.WriteLine("Invalid Choice!");
                break;
        }
    }
}
```

کلاس Program: در این کلاس موتور برنامه به حرکت در می‌آید. ابتدا یک شیء از کلاس UI به‌وجود آورده‌ایم تا به اعضای آن دسترسی داشته باشیم و سپس در یک حلقه‌ی بی‌نهایت متد Proccess() را صدا زده‌ایم. علت وجود حلقه‌ی بی‌نهایت این است که برنامه تا زمانی که کاربر نخواهد، Exit نشود.

```
class Program
{
    static void Main()
    {
        UI engine = new UI();
        while (true)
        {
            engine.Proccess(engine.ShowMenu());
            Console.ReadLine();
        }
    }
}
```

در مرحله‌ی بعد قصد داریم یک خواننده را به جعبه‌ی موسیقی اضافه کنیم. با ساده‌ترین حالت ممکن مثال می‌زنیم. تمام کاری که قرار است انجام دهیم این است که یک شیء Artist درست کرده و آن را در جعبه‌ی موسیقی‌مان (MusicBox) ذخیره کنیم و تمام! به کلاس Artist نگاهی بیندازید، برای این‌که از این کلاس شیء بسازید به نام و نام‌خانوادگی هنرمند نیاز دارید (با توجه به constructor) و مسلماً قرار است که این نام و نام‌خانوادگی از کاربر گرفته شود پس در ابتدا در case "1" متد Proccess() این‌چنین می‌نویسیم:

```
// Add Artist
Console.Write("Enter artist's name: ");
string artistName = Console.ReadLine();
Console.Write("Enter artist's family: ");
string artistFamily = Console.ReadLine();
Artist anArtist = new Artist(artistName, artistFamily);
```

تا این‌جا ما فقط شیء anArtist را به‌وجود آورده‌ایم و هنوز آن را در جایی ذخیره نکردیم. در همین‌جا می‌بینید که عمل گرفتن دو ورودی از کاربر چهار خط کد شده است و این عمل در طول برنامه بسیار تکرار می‌شود پس یک متد برای

دریافت ورودی از کاربر در کلاس UI تعریف می‌کنیم تا هم خط کد کمتری داشته باشیم و همین‌طور عمل دریافت ورودی را برای خودمان راحت‌تر کنیم:

```
static string GetInput(string message)
{
    Console.Write(message);
    return Console.ReadLine();
}
```

این متد پیغام را می‌گیرد و نمایش می‌دهد و ورودی کاربر (که یک رشته است) را return می‌کند. بعد از استفاده از این متد ساخت شیء بسیار راحت‌تر خواهد شد:

```
Artist anArtist = new Artist(
            GetInput("Enter artist's name: "),
            GetInput("Enter artist's family: ")
            );
```

در این‌جا برای ساخت شیء از Artist دو argument به آن داده‌ایم اما تفاوت در این‌جاست که این argument ها دو متد هستند که ابتدا یک پیغام را چاپ می‌کنند و ورودی گرفته شده از کاربر را تحویل constructor کلاس Artist می‌دهند تا شیء anArtist ساخته شود. اکنون باید این شیء ساخته شده را ذخیره کنیم اما در کجا؟ ما در کلاس MusicBox آرایه‌ای از جنس Artist در نظر گرفتیم تا هر خواننده را (تمام و کمال) در یک خانه از این آرایه ذخیره کنیم. آرایه از جنس Artist است و شما در خانه‌های آن می‌توانید اشیایی از جنس Artist ذخیره کنید. اگر به‌یاد داشته باشید متدی به‌نام AddArtist () در کلاس MusicBox تعریف کردیم اما بدنه‌ی آن خالی بود. ما از این متد برای ذخیره کردن خواننده استفاده می‌کنیم. متد AddArtist() باید به‌عنوان ورودی یک Artist را دریافت و آن را در آرایه‌ی Artists ذخیره کند:

```
public bool AddArtist(Artist artist)
{
    Artists[0] = artist;
    return true;
}
```

متد بالا، پارامتر artist را در خانه‌ی اول آرایه‌ی Artists ذخیره می‌کند اما ما نیاز داریم دفعه‌ی بعد، خواننده در خانه‌ی بعدی آرایه ذخیره شود همچنین اگر آرایه پر شده است، متد false را return کند پس یک متغیر به اسم Counter تعریف می‌کنیم تا index خانه‌های آرایه را نگه دارد. به متد AddArtist() و constructor در کلاس MusicBox مجدداً توجه کنید:

```
class MusicBox
{
    // Fields
    public Artist[] Artists;
    private byte Counter;
```

```
    // Constructor
    public MusicBox(ushort size)
    {
        Artists = new Artist[size];
        Counter = 0;
    }

    // Methods
    public bool AddArtist(Artist artist)
    {
        if (Counter < Artists.Length)
        {
            Artists[Counter] = artist;
            Counter++;
            return true;
        }
        else return false;
    }

    public void RemoveArtist()
    {
        // ...
    }

    public void ShowArtists()
    {
        // ...
    }
}
```

همین‌طور متد Process() در کلاس UI به‌صورت زیر است:

```
public void Proccess(string choice)
{
    switch (choice)
    {
        case "1":
            // Add Artist
            Artist anArtist = new Artist(
                GetInput("Enter artist's name: "),
                GetInput("Enter artist's family: ")
                );
            MyMusicBox.AddArtist(anArtist);
            break;
        case "2":
            // show Artists
        break;
        case "3":
            // Exit
            Environment.Exit(0);
        break;
        default:
            Console.WriteLine("Invalid Choice!");
        break;
    }
}
```

به "1" case توجه کنید که چگونه یک شیء از کلاس Artist ساخته‌ایم و سپس از طریق متد AddArtist() این شیء ساخته شده را ذخیره کرده‌ایم. متد AddArtist() یک شیء از جنس Artist دریافت کرده و آن را در خانه‌های آرایه ذخیره می‌کند.

در مرحله‌ی بعد قصد داریم خواننده‌های ذخیره شده را نمایش دهیم. برای این منظور در متد ShowArtists() کلاس MusicBox می‌نویسیم:

```
public void ShowArtists()
{
    for (int i = 0; i < Artists.Length; i++)
    {
        if (Artists[i] == null) continue;
            Console.WriteLine(Artists[i].GetArtistNameAndFamily());
    }
}
```

توسط این متد، نام و نام‌خانوادگی هر خواننده (که در خانه‌های آرایه‌ی Artists ذخیره شده‌اند) را نمایش می‌دهیم و هر خانه‌ی آرایه که خالی باشد نادیده در نظر گرفته می‌شود. در نهایت در "2" case متد Process() می‌نویسیم:

```
case "2":
    // show Artists
    Console.WriteLine();
    MyMusicBox.ShowArtists();
    break;
```

ادامه‌ی حل تمرین را در قسمت دنبال کنید.