زنگ سی‌شارپ – قسمت بیست و هشتم

نوشته‌ی مسعود درویشیان

لینک مستقیم این مطلب در وب‌تارگت

حل تمرین شماره ۱۴

بدون شک این تمرین برای دوستانی که مقالات زنگ سی‌شارپ را دنبال می‌کنند تا حدی مشکل بود بنابراین اگر موفق به حل آن نشدید اصلاً نگران نباشید. هدف از قرار دادن این تمرین این بود که با برنامه‌های بزرگ‌تر آشنا شوید و ارتباط بین اشیاء در برنامه‌ها را بهتر و بیشتر درک کنید.

برای نوشتن هر برنامه‌ای، ابتدا می‌بایست آن را برای خودتان (تا آنجا که می‌توانید) تجزیه و تحلیل کنید و مستقیماً Code Editor را باز نکرده و شروع به برنامه‌نویسی نکنید! مسلماً اگر تازه‌کار باشید و برای نوشتن برنامه، بدون تحلیل اولیه مستقیماً سراغ کدنویسی بروید، احتمالاً بیشتر دچار مشکل می‌شوید بنابراین سعی کنید همیشه تحلیل اولیه را انجام دهید، هرچند که تحلیل کاملی نباشد!

لازم به ذکر است که حل این تمرین بر اساس مباحثی است که تاکنون مطرح شده‌اند و این تمرین می‌تواند از راه‌های بهتر و کدنویسی بسیار تمیزتری حل شود. بنابراین ما این تمرین را بر اساس مباحثی که تا قسمت ۲۷ گفته شده‌اند انجام می‌دهیم.

در این برنامه قصد داشتیم یک جعبه‌ی موسیقی درست کنیم. این جعبه‌ی موسیقی شامل یک‌سری خواننده است بنابراین یک کلاس برای MusicBox تعریف می‌کنیم که شامل آرایه‌ای از خواننده‌ها است. نکته‌ی قابل توجه در این برنامه آهنگ‌ها هستند زیرا در نهایت کاری که انجام می‌دهیم، یک‌سری عملیات به‌روی آهنگ‌هاست پس نیاز است تا یک تعریف برای آهنگ داشته باشیم. به نظر شما یک آهنگ چه چیزهایی دارد؟ اگر آلبوم یا آهنگی را به‌صورت اورجینال تهیه کنید یه سری اطلاعات را در مورد آن اثر می‌توانید روی کاور آن آلبوم یا آن آهنگ ببینید. این اطلاعات می‌توانند نام آهنگ، نام صاحب اثر، آهنگ‌ساز، تنظیم‌کننده، ترانه‌سرا، سبک آهنگ و سال انتشار باشند ما نیز به همین اطلاعات بسنده می‌کنیم. بنابراین برای این منظور کلاسی به‌نام Tune تعریف می‌کنیم:

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

```
}
```

فعلاً از نوشتن متد و constructor ها صرف‌نظر می‌کنیم و در جای خود به آن‌ها می‌پردازیم.

در مرحله‌ی بعد می‌بایست یک تعریف برای خواننده (Artist) داشته باشیم. از این‌رو یک کلاس برای Artist تعریف می‌کنیم. فکر می‌کنید یک هنرمند (خواننده) برای اینکه در برنامه ما باشد به چه چیزهایی نیاز دارد؟ مسلماً هر خواننده اسم، فامیل، تعدادی آلبوم و تعدادی تک‌آهنگ دارد. بنابراین به فیلدهایی برای نام و نام خانوادگی و آرایه‌ای برای ذخیره آلبوم‌ها و آرایه‌ای برای ذخیره‌ی تک‌آهنگ‌ها نیاز داریم. همچنین باید بتوانیم برای این خواننده آلبوم و تک آهنگ اضافه کنیم پس متدهایی برای این منظور در نظر می‌گیریم:

```
class Artist
{
    private string ArtistName;
    private string ArtistFamily;
    private Album[] Albums;
    private Tune[] SingleTunes;

    public void AddAlbum()
    {
        // ...
    }
    public void RemoveAlbum()
    {
        // ...
    }
    public void AddSingleTune()
    {
        // ...
    }
    public void RemoveSingleTune()
    {
        // ...
    }
}
```

متدها و constructor اجرایی مربوط به این کلاس را بعداً تکمیل خواهیم کرد. در مرحله‌ی بعد باید بدانیم یک آلبوم چه ویژگی‌هایی می‌تواند داشته باشد. آلبوم نیز شامل اسم آلبوم، صاحب اثر، سبک آلبوم، سال انتشار آلبوم و تعدادی آهنگ است. همچنین به متدهایی برای اضافه و حذف کردن آهنگ‌های آلبوم نیاز داریم:

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

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

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

تا اینجا کلاس‌هایی که یک جعبه‌ی موسیقی می‌تواند داشته باشد را مشخص کردیم. این کلاس‌ها باید به طریقی به هم مرتبط شوند تا داده‌ها بتوانند بین اشیایی که از کلاس‌ها ساخته می‌شوند، انتقال یابند. برای همین منظور به‌جای اینکه همه‌ی کدها و ارتباط بین کلاس‌ها را در متد Main() بنویسیم، کلاسی به اسم UI تعریف می‌کنیم و ارتباط لازم را در آن کلاس برقرار می‌سازیم و در نهایت در متد Main() کلاس UI را به اجرا در می‌آوریم تا برنامه اجرا شود. برای این منظور کلاس UI را (به‌طور خلاصه) به شکل زیر تعریف می‌کنیم:

```
class UI
{
    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":
                // Exit
                break;
            default:
                Console.WriteLine("Invalid Choice!");
                break;
        }
    }
}
```

کلاس UI کلاسی شلوغ و پر از کد است چراکه همه‌ی کارها به‌طور عمده در همین‌جا انجام می‌شود. البته تا حد ممکن از متد و شی‌گرایی استفاده خواهد شد اما همان‌طور که ذکر کردیم حل این تمرین با توجه با مباحثی که تا اینجا مطرح شده انجام می‌شود و بهتر است بدانید این روشی که با آن، تمرین را انجام می‌دهیم تا حد زیادی غیر استاندارد است و در دنیای برنامه‌نویسی حرفه‌ای جایی ندارد. اما دلیلی برای نگرانی نیست چراکه شما باید پله پله مراحل برنامه‌نویسی حرفه‌ای را طی کنید و مطمئن باشید نمی‌توانید یک شبه ره صد ساله را بروید.

در متد Main() برنامه به‌طور کامل به‌هم مرتبط و قابل اجرا می‌شود:

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

برنامه ما به این ترتیب قابل اجرا خواهد بود. هنگامی‌که برنامه را اجرا می‌کنید می‌توانید از بین سه گزینه‌ی انتخابی یکی را انتخاب کرده تا کدهای مربوط به هر قسمت اجرا شود اما همان‌طور که می‌بینید، متد Proccess() که عملیات مربوط به هر قسمت را انجام می‌دهد، خالی و تنها شامل یک دستور switch است.

در مرحله‌ی بعد، برای این‌که بتوانیم از کلاس‌هایی که تعریف کرده بهتر استفاده کنیم، برای هر کدام constructor تعریف خواهیم کرد. با کلاس MusicBox شروع می‌کنیم. همان‌طور که پیش‌تر بیان شد، MusicBox شامل آرایه‌ای از خواننده‌ها است. این کلاس نیز می‌تواند متدهایی به‌منظور کم و زیاد کردن و نمایش خواننده‌ها داشته باشد:

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

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

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

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

همان‌طور که می‌بینید، این کلاس شامل یک فیلد به‌نام Artists (که آرایه‌ای از جنس Artist است)، یک constructor و سه متد است. البته در طی تکمیل برنامه فیلدها و متدهایی به برنامه افزوده می‌شود. در constructor این کلاس، آرایه‌ای از جنس Artist ساخته شده که اندازه‌ی این آرایه، در حین ساختن شیء از MusicBox مشخص می‌شود.

در قسمت بعد به کلاس Artist یک constructor اضافه می کنیم. هنگامی که یک شیء از Artist ساخته می‌شود چه اطاعاتی را باید مقداردهی شوند؟ به‌نظر می‌رسد نام و نام‌خانوادگی، تعداد آلبوم و تعداد تک‌آهنگ‌های منتشر شده از آن خواننده بهتر است که در constructor مشخص شوند:

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

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

اگر به constructor این کلاس توجه کنید، می‌بینید که دو پارامتر artistName و artistFamily وجود دارند که به فیلدهای ArtistName و ArtistFamily اختصاص می‌یابند. همچنین آرایه‌ای از Album به اندازه ۵ (تعدادی آلبومی که هر خواننده می‌تواند در این جعبه موسیقی داشته باشد) و آرایه‌ای از Tune به اندازه ۱۰ (تعداد تک‌آهنگی که هر خواننده می‌تواند داشته باشد) در این constructor به‌وجود آمده است. این مقادیر (اندازه آرایه‌ها) را نیز می‌توانستیم مستقیماً از کاربر دریافت کنیم تا اندازه آن‌ها ثابت نباشد اما در این‌جا آن‌ها را ثابت در نظر می‌گیریم.

به نظر شما constructor کلاس Album به چه چیزهایی نیاز دارد؟ هنگامی که یک شیء از کلاس Album در حال ساخته شدن است، چه فیلدهایی باید مقداردهی شوند؟ مسلم است که همه‌ی فیلدهای آن باید مقداردهی شوند. هنگامی که قصد دارید یک Album بسازید باید نام آلبوم، صاحب آن، سبک، سال انشار و آهنگ‌هایی که در آن آلبوم قرار دارند را مشخص کنید. مجدداً به کلاس Album توجه کنید:

```
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()
    {
        //...
    }
}
```

هنگامی‌که یک شیء از کلاس Album در حال ساخته شدن است، باید نام آلبوم، یک شیء Artist، سبک آلبوم، سال انتشار و یک آرایه از Tune را به آن بدهید تا شیء Album ساخته شود. نکته قابل توجه، شیء Artist و آرایه Tune است. هنگامی‌که شما یک شیء از Artist را ساخته و به constructor می‌دهید می‌توانید به تمام اعضای public شیء Artist دسترسی داشته باشید. همان‌طور که می‌بینید برای فیلد AlbumOwner ما به نام خواننده (صاحب آلبوم) نیاز داریم. برای این

منظور یک متد public به کلاس Artist اضافه می‌کنیم تا نام خواننده را return کند. (به‌دلیل این‌که فیلدهای Artist همه‌گی private هستند نمی‌توانیم مستقیماً به آن‌ها دسترسی داشته باشیم):

```
public string GetArtistName()
{
    return ArtistName;
}
```

این متد در کلاس Artist قرار دارد و رشته‌ی ArtistName را return می‌کند و به‌صورت زیر به رشته‌ی AlbumOwner اختصاص داده می‌شود:

```
AlbumOwner = artist.GetArtistName();
```

همچنین tunes که آرایه‌ای از کلاس Tune است به constructor داده شده و به فیلد Tunes اختصاص می‌یابد. دقت کنید که در این لحظه شما آرایه‌ای به‌اسم tunes از جنس Tune را در آرایه‌ای دیگر به‌اسم Tunes که از جنس Tune است، می‌ریزید. آرایه‌ها باید همجنس هم باشند (هردو از جنس Tune) در غیر این‌صورت برنامه شما کامپایل نخواهد شد. برای مثال نمی‌توانید آرایه‌ای از جنس Tune را به آرایه‌ای از جنس Artist اختصاص دهید.

نوبت به constructor کلاس Tune می‌رسد. Tune یا همان آهنگ مورد نظر برای هر خواننده، دو نوع دارد. آهنگ یا در یک آلبوم قرار دارد و یا به‌صورت تک‌آهنگ است. هنگامی‌که شما در حال ساخت شیء از Tune هستید، می‌توانید آهنگ را طوری به‌وجود آورید که مخصوص آلبوم یا به‌صورت تک‌آهنگ باشد. آهنگی که مخصوص آلبوم است، بعضی از اطلاعات آن مثل سال انتشار و صاحب آهنگ از قبل مشخص است اما یک تک‌آهنگ به سال انتشار و نام صاحب اثر نیز نیاز دارد. برای این منظور در این‌جا از constructor overloading استفاده کرده‌ایم:

```
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.GetNameAndFamily();
        TuneYear = tuneYear;
    }
}
```

به این ترتیب برای ساخت شیء از کلاس Tune به دو طریق می‌توانیم این کار را انجام دهیم. به‌منظور ساخت آهنگ برای آلبوم بایستی نام، سبک، شاعر، آهنگ‌ساز و تنظیم را مشخص کنید ولی به‌منظور ساخت تک‌آهنگ کافی است علاوه بر موارد قبلی، سال انتشار و Artist هم مشخص کنید. نکته‌ی قابل توجه در این‌جا کلمه‌ی کلیدی this در constructor تک‌آهنگ است. اگر در این‌جا از کلمه‌ی کلیدی this استفاده نمی‌کردیم، آن‌گاه constructor به شکل زیر می‌شود:

```
    /// 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
        )
    {
        TuneOwner = artist.GetNameAndFamily();
        TuneYear = tuneYear;

        // without this keyword
        TuneName = tuneName;
        TuneGenre = tuneGenre;
```

```
        Composer = composer;
        Songwriter = songwriter;
        Arrangement = arrengement;
    }
```

همان‌طور که می‌بینید بدون کلمه‌ی کلیدی this مجبور شدیم چند خط کد اضافه‌تر بنویسیم تا فلیدها را مقداردهی کنیم درحالی‌که constructor قبلی این مقداردهی‌ها را انجام می‌دهد و در این‌جا فقط اضافه‌نویسی کرده‌ایم. با کلمه‌ی کلیدی this فیلدهایی که توسط constructor قبلی می‌توانند مقداردهی شوند را مشخص می‌کنیم تا آن constructor مقادیر را به فیلدها اختصاص دهد و در constructor دومی، فقط TuneYear و TuneOwner را مقدادهی می‌کنیم.

ادامه حل تمرین را در قسمت بعد دنبال کنید.

کلیه حقوق مادی و معنوی برای وب‌سایت وب‌تارگت محفوظ است.

استفاده از این مطلب در سایر وب‌سایت‌ها و نشریات چاپی تنها با ذکر و درج لینک منبع مجاز است.