زنگ سی‌شارپ – قسمت سی و یکم

نوشته‌ی مسعود درویشیان

لینک مستقیم این مطلب در وب‌تارگت

در قسمت قبلی زنگ سی‌شارپ، در حل تمرین شماره‌ی ۱۴، توانستید برای هر هنرمند آلبوم ذخیره کنید، هنرمند را حذف کنید و آلبوم‌های ذخیره شده را مشاهده کنید. همچنین به تفاوت == و متد Equals() پی بردید و اندکی با namespace آشنا شدید. در این قسمت در مورد Operator Overloading صحبت خواهیم کرد و سپس به ادامه‌ی حل تمرین شماره ۱۴ می‌پردازیم.

Operator Overloading

سی‌شارپ به شما اجازه می‌دهد operator (عملگر) هایی تعریف کنید که مرتبط به کلاس‌هایی است که خودتان می‌سازید. به این پروسه، operator overloading گفته می‌شود. با overload کردن یک operator، شما کاربرد آن operator را به کلاس خودتان اضافه می‌کنید. تاثیری که این operator روی کلاس شما می‌گذارد کاملاً تحت کنترل خودتان است و ممکن است برای هر کلاس متفاوت باشد. به‌عنوان مثال، کلاسی که یک لیست پیوندی تعریف می‌کند، ممکن است از عملگر + برای افزودن یک شیء به انتهای لیست، استفاده کند. کلاسی که stack را اجرا می‌کند، ممکن است از عملگر + برای افزودن یک شیء به بالای پشته، استفاده کند. کلاسی دیگر ممکن است از عملگر + به‌طور کاملاً متفاوت استفاده کند.

هنگامی‌که یک عملگر overload می‌شود، معنای واقعی خودش را از دست نمی‌دهد. بلکه فقط کاربرد آن به یک کلاس افزوده می‌شود. بنابراین (به‌عنوان مثال) overload کردن عملگر + برای افزودن یک شیء به انتهای لیست پیوندی، دلیل نمی‌شود که عملکرد آن operator برای جمع کردن دو عدد صحیح تغییر کند.

مزیت اصلی operator overloading این است که به شما اجازه می‌دهد به‌طور یکپارچه، یک کلاس جدید را در محیط برنامه‌نویسی خود، ادغام کنید. این ویژگی که به آن type extensibility می‌گویند، یکی از بخش‌های مهم یک زبان برنامه‌نویسی شی‌گرا مثل سی‌شارپ است. هنگامی‌که operator ها برای یک کلاس تعریف می‌شوند، می‌توانید آن operator را روی اشیای کلاس مربوطه، اعمال کنید. این نکته قابل ذکر است که operator overloading یکی از قدرمندترین ویژگی‌های سی‌شارپ است.

اصول Operator Overloading

Operator overloading شباهت زیادی با Method overloading دارد. برای overload کردن یک عملگر، از کلمه‌ی کلیدی operator برای تعریف یک operator method استفاده می‌کنیم که برای یک عمل خاص، مربوط به کلاس خودش، تعریف می‌شود.

دو حالت از operator method وجود دارد: unary operators (عملگرهای تکی) و binary operators (عملگرهای دوتایی). فرم کلی هرکدام را در زیر می‌بینید:

```
// General form for overloading a unary operator
public static ret-type operator op(param-type operand)
{
    // operations
}

// General form for overloading a binary operator
public static ret-type operator op(param-type1 operand1, param-type1 operand2)
{
    // operations
}
```

در این‌جا، عملگری که آن را overload می‌کنید، مثل + یا / ، جایگزین op می‌شود. ret-type مشخص کننده‌ی نوع مقداری است که return خواهد شد. اگرچه return type می‌تواند از هر نوعی باشد اما اغلب از نوع همان کلاسی است که operator در آن overload می‌شود. این ارتباط (یکسان بودن return type با جنس کلاس) باعث راحتی استفاده از عمگرهای overload شده می‌شود. برای unary operator ها، عملوند در قسمت operand قرار می‌گیرد. برای binary operator ها، عملوندها در قسمت operand1 و operand2 قرار خواهد گرفت. توجه داشته باشید که operator method ها باید هم public و هم static باشند.

در unary operator ها، نوع عملوند (operand) باید با نوع کلاسی که operator در آن تعریف می‌شود، یکسان باشد. بنابراین نمی‌توانید operator های سی‌شارپ را برای اشیایی که خودتان نساخته‌اید، تعریف کنید. برای مثال، نمی‌توانید مجدداً عملگر + را برای int و string تعریف کنید. نکته‌ی دیگر اینکه، operator parameters نباید از ref و out استفاده کنند.

برای این‌که بیشتر با operator overloading آشنا شوید، مثالی خواهیم زد که در آن binary operator های + و – را overload می‌کنیم. برنامه زیر کلاسی به اسم TwoD دارد که مختصات دوبعدی یک شیء را نگه‌داری می‌کند. عمگر +

که آن را overload کرده‌ایم، مختصات یک شیء TwoD را با یک شیء دیگر TwoD جمع می‌کند. عملگر - نیز مختصات یک شیء را از دیگری کم می‌کند.

```
using System;
// A two-dimensional coordinate class.
class TwoD
{
    private int X, Y; // 2-D coordinates

    // Constructors
    public TwoD() { X = 0; Y = 0; }
    public TwoD(int x, int y)
    {
        this.X = x;
        this.Y = y;
    }

    // Overload binary +
    public static TwoD operator +(TwoD ob1, TwoD ob2)
    {
        TwoD result = new TwoD();

        result.Y = ob1.Y + ob2.Y;
        result.X = ob1.X + ob2.X;

        return result;
    }

    // Overload binary -
    public static TwoD operator -(TwoD ob1, TwoD ob2)
    {
        TwoD result = new TwoD();

        result.Y = ob1.Y - ob2.Y;
        result.X = ob1.X - ob2.X;

        return result;
    }

    public void Show()
    {
        Console.WriteLine("X: {0}    Y: {1}", this.X, this.Y);
    }
}
class TwoDDemo
{
    static void Main()
    {
        TwoD a = new TwoD(45, 30);
        TwoD b = new TwoD(15, 13);
        TwoD c;

        Console.Write("Here is a: ");
        a.Show();

        Console.Write("Here is b: ");
        b.Show();
```

```
        c = a + b;

        Console.WriteLine();
        Console.WriteLine("After add operator: ");
        c.Show();

        c = a - b;
        Console.WriteLine();
        Console.WriteLine("After subtract operator: ");
        c.Show();
    }
}
```

هنگامی‌که دو شیء از نوع TwoD توسط عملگر + باهم جمع می‌شوند، مقدار مختصات مربوطه‌شان، همان‌طور که در operator+() می‌بینید، باهم جمع می‌شود. توجه داشته باشید که این متد مقدار عملوندهایش را تغییر نمی‌دهد، بلکه یک شیء جدید از نوع TwoD که حاصل این جمع را در خودش دارد توسط متد، return می‌شود. برای این‌که بفهمید چرا عمل‌گر + عمل‌وندهای خودش را تغییر نمی‌دهد، عمل‌گر استاندارد ریاضی + که در آن (مثلاً) ۱۰ + ۱۲ می‌کنید را در نظر بگیرید. حاصل این جمع ۲۲ است اما ۱۰ و ۱۲ توسط آن عوض نمی‌شوند. البته هیچ قانونی وجود ندارد که جلوی عمل‌گر overload شده را بگیرد تا مقدار عمل‌وندهایش را تغییر ندهد، اما بهتر است وقتی‌که یک عمل‌گر overload می‌شود، معنای اصلی خودش را نیز حفظ کند.

توجه کنید که operator+() یک شیء از نوع TwoD را return می‌کند. اگرچه این متد می‌تواند نوع‌های دیگری که در سی‌شارپ مجاز هستند را نیز return کند اما return کردن یک شیء از نوع TwoD به عمل‌گر + اجازه می‌دهد تا بتواند در عبارت‌های ترکیبی مثل a + b + c مورد استفاده قرار گیرد. در این‌جا a + b نتیجه‌ای از جنس TwoD تولید می‌کند که می‌توانیم آن را در c ذخیره کنیم.

نکته‌ی مهم دیگری در این‌جا وجود دارد. وقتی‌که مختصات اشیاء، درون operator+() باهم جمع می‌شوند، حاصل جمع x و y، عدد صحیح است. درواقع، عمل‌گر + که برای اشیای TwoD، overload شده است تاثیری بر عمل‌گر + که روی اعداد صحیح اعمال می‌شود ندارد.

اکنون به operator-() توجه کنید. عمل‌گر – مثل عمل‌گر + کار می‌کند به‌استثنای این‌که ترتیب پارامترها در این‌جا اهمیت دارد و پارامترها جابه‌جایی‌پذیر نیستند (یعنی A-B با B-A متفاوت است). برای همه‌ی binary operator ها، پارامتر اول برای عمل‌وند سمت چپ و پارامتر دوم برای عمل‌وند سمت راست است. بنابراین هنگامی‌که ورژن overload شده‌ی

operator های جابه‌جایی‌ناپذیر را اجرا می‌کنید باید به‌یاد داشته باشید که کدام عمل‌وند سمت چپ و کدام یک سمت راست قرار دارد. (ادامه‌ی این بحث را در قسمت بعد دنبال کنید)

ادامه‌ی حل تمرین شماره ۱۴

اگر به یاد داشته باشید یکی از عملیات این برنامه اضافه کردن تک‌آهنگ برای هر هنرمند بود. برای افزودن تک‌آهنگ برای هر خواننده ما از متد زیر که در کلاس Artist قرار دارد، استفاده می‌کنیم:

```
public bool AddSingleTune(Tune tune)
{
    if (SingleTuneCounter < SingleTunes.Length)
    {
        SingleTunes[SingleTuneCounter] = tune;
        SingleTuneCounter++;
        return true;
    }
    return false;
}
```

در این متد یک متغیر به‌اسم SingleTuneCounter داریم که شمارنده و کنترل‌کننده‌ی تعداد تک‌آهنگ‌های ذخیره شده است. مقدار این متغیر را در constructor برابر با صفر قرار داده و پس از افزودن هر تک‌آهنگ، مقدار این متغیر را یک واحد افزایش می‌دهیم. این متد همچنین یک شیء از جنس Tune دریافت کرده و آن را در آرایه‌ای که برای ذخیره تک‌آهنگ‌ها در نظر گرفته شده است، ذخیره می‌کند.

از طرف دیگر در متد UI باید یک شیء Tune به‌وجود آوریم و در نهایت شیء ساخته شده را به متد AddSingleTune() پاس بدهیم تا در آرایه مربوطه ذخیره شود. نکته قابل توجه این‌جاست که باید شیء Tune را از طریق دومین constructor آن که برای تک‌آهنگ‌ها در نظر گرفته بودیم، بسازیم. به کلاس Tune و constructor های آن مجدداً توجه کنید:

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


    // Constructor for album tunes
    public Tune(
        string tuneName,
```

```
        string tuneGenre,
        string composer,
        string songwriter,
        string path
        )
    {
        TuneName = tuneName;
        TuneGenre = tuneGenre;
        Composer = composer;
        Songwriter = songwriter;
        Path = path;
    }

    // Constructor for single tunes
    public Tune(
        string tuneName,
        string tuneGenre,
        string composer,
        string songwriter,
        ushort tuneYear,
        Artist artist,
        string path
        )
        : this(tuneName, tuneGenre, composer, songwriter, path)
    {
        TuneOwner = artist.GetArtistNameAndFamily();
        TuneYear = tuneYear;
    }
}
```

به دومین constructor دقت کنید. در این constructor ما علاوه‌بر مواردی که در constructor قبلی وارد کردیم، یک شیء از جنس Artist و سال انتشار را نیز باید وارد کنیم. بنابراین برای ساخت تک‌آهنگ از این constructor استفاده کرده‌ایم.

همچنین برای دیدن تک‌اهنگ‌های ذخیره شده از متد زیر استفاده می‌کنیم:

```
public void ViewSingleTunes()
{
    for (byte b = 0; b < SingleTunes.Length; b++)
    {
        if (SingleTunes[b] == null) continue;
        Console.WriteLine(SingleTunes[b].GetTuneName());
    }
}
```

در متد بالا، تک‌آهنگ‌ها مستقیماً نمایش داده می‌شوند اما شاید نخواهیم که تک‌آهنگ‌ها به این صورت (در هر خط یک تک‌آهنگ) نمایش داده شود. شاید بخواهیم در قسمت‌های مختلف برنامه این تک‌آهنگ‌ها را به‌صورت‌های متفاوتی نمایش دهیم. بنابراین به‌جای نمایش مستقیم آن‌ها، یک متد می‌نویسیم و آرایه‌ای از تک‌آهنگ‌های موجود را return می‌کنیم تا هرجای برنامه که لازم بود این تک‌آهنگ‌ها را آن‌طور که خواستیم نمایش دهیم.

```
public Tune[] GetSingleTunes()
{
    return SingleTunes;
}
```

در متد بالا تک‌آهنگ‌های ذخیره شده return می‌شود و سپس در متد UI هرطور که خواستیم آن‌ها را نمایش می‌دهیم.

تا اینجا مراحل افزودن (هنرمند، آلبوم، تک‌آهنگ) و دیدن چیزهایی که اضافه کردید را به اتمام رساندید. اکنون وقت آن رسیده است که بتوانید موارد ذخیره شده را ویرایش کنید. در مرحله‌ی ویرایش، بایستی بتوانیم هنرمند، آلبوم و تک‌آهنگ‌های او را ویرایش (Edit) کنیم.

اگر پروژه قسمت قبل را دانلود کرده باشید، برای این منظور یک گزینه به اسم Edit برای هر هنرمند در نظر گرفته بودیم. هنگامی‌که Edit انتخاب می‌شود، باید سه گزینه اصلی برای Edit نمایش داده شود که در مورد دو گزینه در اینجا بحث خواهیم کرد. گزینه‌های Edit name and family و Edit Albums و Edit single tunes گزینه‌هایی هستند که برای این منظور در نظر گرفتیم.

```
1. Edit name and family     2. Edit Albums
3. Edit single tunes        4. Back
```

با گزینه‌ی Edit name and family شروع می‌کنیم. در اینجا شما باید نام و نام‌خانوادگی هنرمند را ویرایش کنید. برای این منظور تنها کافی است که نام و نام‌خانوادگی جدید را از کاربر دریافت، سپس آن را جایگزین نام و نام‌خانوادگی قبلی کنید. برای ویرایش نام و نام‌خانوادگی از متد زیر استفاده می‌کنیم:

```
public void EditNameAndFamily(string name, string family)
{
    ArtistName = name;
    ArtistFamily = family;
}
```

این متد که در کلاس Artist قرار دارد، نام و نام‌خانوادگی جدید را دریافت و آن را جایگزین قبلی می‌کند.

در مرحله‌ی بعد قصد داریم که تک‌اهنگ‌ها را ویرایش کنیم. برای ویرایش تک‌آهنگ گزینه‌های زیر را مد نظر داریم:

```
1. Edit Name    2. Edit Genre    3. Edit Composer    4. Edit Songwriter
5. Edit Year    6. Edit Path     7. Delete           8. Back
```

تنها کاری که در این‌جا باید انجام دهید این است که توسط یک switch همه‌ی این گزینه‌ها را وارد و یکی‌یکی ویرایش کنید:

```
switch (choice)
{
    case "1":
        // Edit name
        break;
    case "2":
        // Edit genre
        break;
    case "3":
        // Edit composer
        break;
    case "4":
        // Edit songwriter
        break;
    case "5":
        // Edit year
        break;
    case "6":
        // Edit path
        break;
    case "7":
        // Delete
        break;
    case "8":
        // Back
        break;
}
```

برای به‌انجام رساندن این ویرایش‌ها، در کلاس Tune متدهایی زیر را در نظر گرفته‌ایم:

```
// Methods (Get and Set)
public string GetTuneName() { return TuneName; }
public void SetTuneName(string name) { TuneName = name; }

public string GetComposer() { return Composer; }
public void SetComposer(string composer) { Composer = composer; }

public string GetSongwriter() { return Songwriter; }
public void SetSongwriter(string songwriter) { Songwriter = songwriter; }

public string GetTuneGenre() { return TuneGenre; }
public void SetTuneGenre(string genre) { TuneGenre = genre; }

public ushort GetTuneYear() { return TuneYear; }
public void SetTuneYear(ushort year) { TuneYear = year; }

public string GetTunePath() { return Path; }
public void SetPath(string path) { Path = path; }
```

بنابراین برای ویراش نام یک تک‌آهنگ کافی است که نام جدید را از کاربر دریافت و آن را توسط متد SetTuneName() جایگزین نام قبلی کنیم. بقیه‌ی فیلدها نیز به همین ترتیب ویرایش می‌شوند. برای delete کردن یک تک‌آهنگ کافی است که آن خانه از آرایه که تک‌آهنگ مورد نظر در آن قرار دارد را برابر با null قرار دهید. برای این منظور از متد زیر استفاده می‌کنیم:

```
public void RemoveSingleTune(int index)
{
    SingleTunes[index] = null;
}
```

در قسمت بعد، ویراش album و play کردن آهنگ‌ها را توضیح خواهیم داد. پروژه این تمرین را می‌توانید از این‌جا دانلود کنید. قابل ذکر است که پروژه اکنون تا ویرایش تک‌آهنگ‌ها برای دانلود قرار داده شده است و در قسمت بعد می‌توانید نسخه نهایی و کامل آن را دانلود کنید.