زنگ سی‌شارپ – قسمت بیست و دوم

نوشته‌ی مسعود درویشیان

لینک مستقیم این مطلب در وب‌تارگت

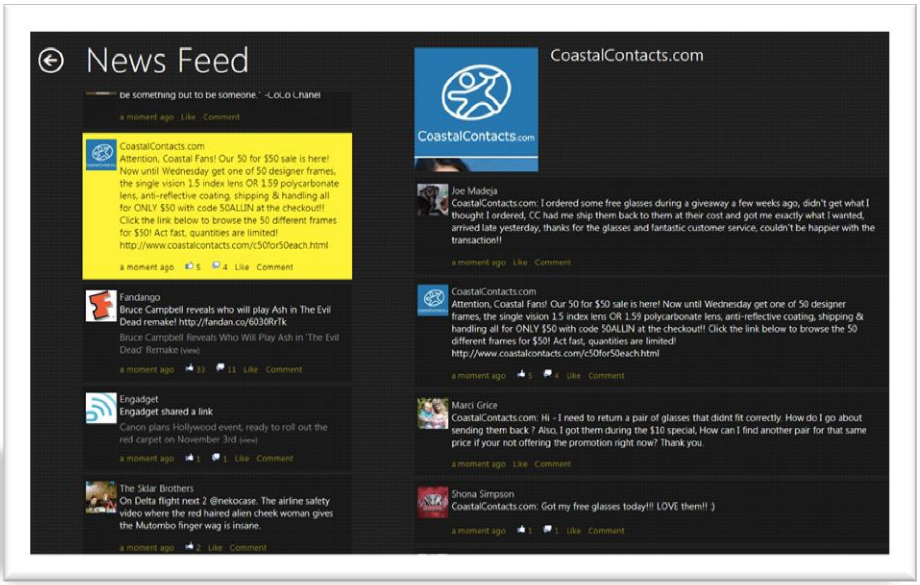
تا این قسمت از مقالات با دستورات و خصوصیات بسیاری از سی‌شارپ آشنا شده‌اید که به کمک آن‌ها می‌توانید کارهای زیادی انجام دهید. مطالبی که تا این‌جا ارائه شده‌اند همه‌گی از اهمیت بسیار بالایی برخوردار هستند و امید است که خوانندگان عزیز بدون مشکل تا این‌جا پیش آمده باشند. همانطور که پیش از نیز ذکر شد، لطفاً در صورت نامفهوم بودن هر مطلب و هر تمرین، حتماً مشکل خود را مطرح کنید تا در اسرع وقت به شما پاسخ داده شود. سوالات، نظرات و پیشنهادات شما موجب بالا رفتن کیفیت مطالب و همچنین درک بهتر مطلب، هم برای شما و هم دیگر خوانندگان می‌شود.

بسیاری از دوستان و خوانندگان از این‌که مطالب و آموزش‌ها در محیط کنسول ارائه می‌شود ابراز نگرانی می‌کنند و خواهان ارائه مطالب در محیطی مدرن‌تر و کاربردی‌تر هستند. البته یک خواننده‌ی آگاه حق اظهار چنین نظر و درخواستی را دارد. او نمی‌خواهد وقت خود را در این دنیای سرشار از اطلاعاتی که روز به روز در حال تغییر و گسترش است، هدر دهد. یک خواننده‌ی باهوش چیزی را می‌خواند که واقعاً برایش مفید باشد و وقت ارزشمند او را نگیرد. لازم به ذکر است که نباید هیچ نگرانی به دل خود راه دهید چراکه شما در حال یادگیری اصل زبان سی‌شارپ هستید و پس از گذاردن این سطح می‌توانید به‌راحتی در محیط‌های دل‌خواه خود فعالیت کنید و از این‌که می‌توانید در محیطی مثل موبایل یا ویندوز اپلیکیشن و... برنامه بنویسید لذت ببرید. شیرین‌ترین لحظه‌ی برنامه‌نویسی برای شما آن زمانی است که درحال آشنایی با



محیط کاربردی و جدیدی مثل برنامه‌نویسی برای ویندوز ۸ یا اندروید هستید و می‌بینید که دیگر درک بسیار زیادی از دستورات دارید و از این‌که با این زبان مشکلی ندارید (و تا حد قابل قبولی با این زبان آشنایی دارید) احساس خرسندی و رضایت می‌کنید. همان‌طور که می‌دانید با ارائه‌ی ویندوز جدید مایکروسافت، رابط کاربری

مترو محبوبیت زیادی پیدا کرده و با وجود نوظهور بودن آن، حجم بسیار زیادی از برنامه‌ها برای این رابط کاربری (که بسیار جذاب و دوست‌داشتنی است) روزانه در حال ارائه شدن هستند. همچنین ابزارهای خیلی کاربردی و بسیار زیادی برای کمک به توسعه برنامه‌های این رابط کاربری ارائه شده‌اند که به کمک آن‌ها می‌توانید کارهای خیلی جالبی انجام دهید و برنامه‌های پربارتری بنویسید. مسلماً تا چند سال آینده در ایران، ویندوز ۸ (که هم روی دستگاه‌های لمسی و هم پی‌سی ارائه می‌شود) محبوبیت زیادی پیدا می‌کند و می‌توانید در کنار ارائه برنامه‌های ویندور فرم، ساخت اپلیکیشن‌های مترو را نیز تجربه کنید.

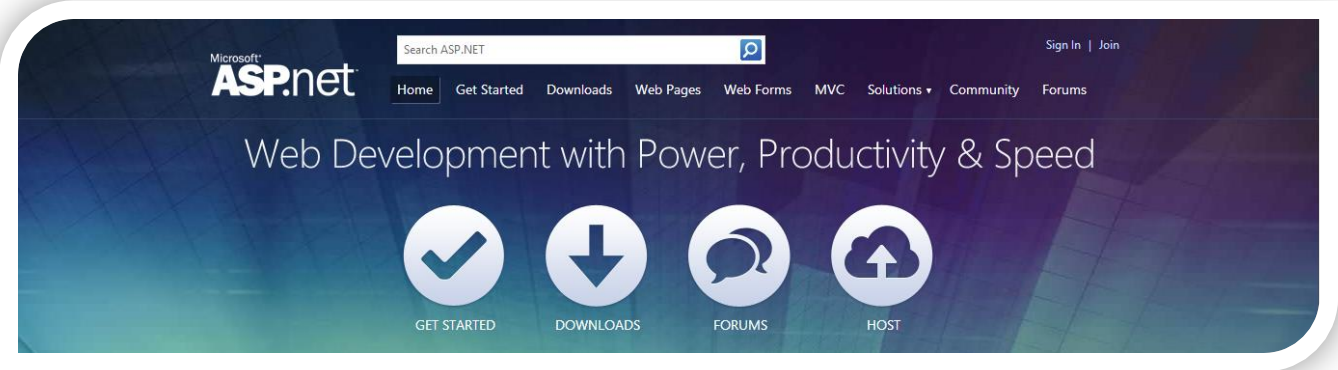




همین‌طور می‌توانید در محیط وب و موبایل، بازی‌های رایانه‌ای و بازی‌های موبایل، Xbox و Kinect مایکروسافت و یا در شاخه‌ی برنامه‌نویسی هوش مصنوعی با زبان سی‌شارپ و... فعالیت داشته باشید.









کافی است که برای یادگیری کاری که می‌خواهید انجام دهید پشت‌کار لازم را داشته باشید و به این راحتی‌ها دست برندارید تا بتوانید به‌زودی در محیط‌های دل‌خواه خود با سی‌شارپ برنامه‌نویسی کنید. خوشبختانه آن‌قدر حجم اطلاعات ارائه شده در مورد سی‌شارپ و فعالیت‌های مرتبط به آن زیاد هستند که اگر پا به دنیای بزرگ آن بگذارید برای انجام هر کاری، به سرعت مطلب و راه حل مورد نیاز را پیدا می‌کنید. تصور کنید که در سایت stackoverflow پرسش خود را مطرح می‌کنید و در کمتر از ۱۵ دقیقه حدود ۱۰ پاسخ و راهنمایی متنوع دریافت می‌کنید که همه‌ی پاسخ‌ها به نحوی

برای شما مفید هستند. این می‌تواند برای کار و تجربه‌ی شما بسیار سودمند باشد. در واقع شما در این مسیر هیچ‌گاه تنها نیستید، بلکه یاران بسیار زیادی هستند که به شما کمک می‌کنند تا در این راه موفق و موفق‌تر شوید.

در این قسمت قصد نداریم مطلب جدیدی ارائه دهیم بلکه به تمرین مباحث مطرح شده می‌پردازیم. اکنون دیگر با متد و کلاس آشنایی دارید و می‌توانید برنامه‌هایی که می‌نویسید را تا حدودی شی‌گرا کنید. متد را مثل یک دستگاه مکانیکی فرض کنید که یک کار خاص را برای شما انجام می‌دهد. گاهی این دستگاه ورودی می‌گیرد، گاهی نمی‌گیرد. گاهی چیزی را به شما برمی‌گرداند و گاه چیزی برنمی‌گرداند. به‌عنوان مثال، متدی می‌نویسیم که دو عدد را در هم ضرب می‌کند و حاصل را بر می‌گرداند (به همان‌جایی که متد صدا زده شده است):

```
using System;
class Example
{
    static void Main()
    {
        int result;
        result = Multipliction(5, 5);
        Console.WriteLine(result);
    }

    static int Multiplication(int a, int b)
    {
        int res;
        res = a * b;
        return res;

        // Or
        // return a * b;
    }
}
```

در این برنامه در کلاس Example دو متد قرار دارد. متد Main() که نقطه‌ی شروع برنامه است و متد Multiplication() که هر دوی این متدها به‌صورت static تعریف شده‌اند. اگر به‌یاد داشته باشید ذکر شد که یک متد static فقط می‌تواند (به‌صورت مستقیم) به اعضای static دسترسی داشته باشد و همچنین دانستید که یک عضو static وابسته به هیچ شیءای نیست. متد Main() که حتماً باید static باشد، بنابراین برای این‌که بتوانیم درون متد Main() مستقیماً به متد Multiplication() دسترسی داشته باشیم باید این متد را نیز به‌صورت static تعریف کنیم. درون متد Main()، متد Multiplication() را صدا زده‌ایم و دو مقدار را به آن داده‌ایم. این دو مقدار در هم ضرب می‌شوند و نتیجه‌ی آن return شده و در متغیر result ذخیره می‌شود.

در مثال بعد قصد داریم یک Amplifier را شبیه‌سازی کنیم. این Amplifier یک سری اطلاعات مثل نام، مدل، وات و... دارد.

```
class Amplifier
{
    private string Name;
    private string Model;
    private int Watt;
    private int Multichannel;
    private int EffectivePower;
    private bool ThreeDSurroundSound;
    private int CurrentVolume;
    private int MaxVolume;

    public Amplifier(string name, string model, int watt, int multichannel,
        bool threeDSurroundSound, int maxVolume)
    {
        Name = name;
        Model = model;
        Multichannel = multichannel;
        Watt = watt;
        ThreeDSurroundSound = threeDSurroundSound;
        EffectivePower = watt * multichannel;
        CurrentVolume = 0;
        MaxVolume = maxVolume;
    }
}
```

در ابتدا برای این کلاس یک constructor تعریف کرده‌ایم تا فیلدهای آن حتماً در لحظه‌ی ساخت شیء، مقداردهی شوند. در مرحله‌ی بعد قصد داریم یک شیء از این کلاس بسازیم و چند متد به کلاس نیز اضافه کنیم:

```
using System;
class AmpDemo
{
    static void Main()
    {
        Amplifier Kenwood = new Amplifier("KENWOOD", "RA-5000", 120, 5, true, 50);

        Console.WriteLine("Amplifier simulation ");
        Console.WriteLine("  Name and Model: " + Kenwood.GetNameAndModel());
        Console.WriteLine("  Effictive Power: " + Kenwood.GetEffectivePower());
        Console.WriteLine("  Multichannel: " + Kenwood.GetMultichannel());
        Console.WriteLine("  3D Surround Sound: " + Kenwood.GetThreeDSurroundSound());
    }
}
class Amplifier
{
    private string Name;
    private string Model;
    private int Watt;
    private int Multichannel;
    private int EffectivePower;
    private bool ThreeDSurroundSound;
    private int CurrentVolume;
    private int MaxVolume;
```

```
    public Amplifier(string name, string model, int watt, int multichannel,
        bool threeDSurroundSound, int maxVolume)
    {
        Name = name;
        Model = model;
        Multichannel = multichannel;
        Watt = watt;
        ThreeDSurroundSound = threeDSurroundSound;
        EffectivePower = watt * multichannel;
        CurrentVolume = 0;
        MaxVolume = maxVolume;
    }

    public int GetEffectivePower()
    {
        return EffectivePower;
    }

    public string GetNameAndModel()
    {
        return Name + " " + Model;
    }

    public int GetMultichannel()
    {
        return Multichannel;
    }

    public string GetThreeDSurroundSound()
    {
        if (ThreeDSurroundSound)
            return "Yes";
        else
            return "No";
    }
}
```

همان‌طور که می‌بینید، ابتدا یک شیء از این کلاس ساخته و مقادیر مربوطه را به آن داده‌ایم. سپس متدهای ساده‌ای تعریف کرده‌ایم تا بتوانیم مقدار فیلدهای private را بخوانیم. همین‌طور که می‌دانید فیلدهای private خارج از کلاس خودشان در دسترس نیستند و ما از طریق یک متد public توانستیم مقدار آن‌ها را به خارج از کلاس نشان دهیم. با این کار مقدار بعضی از فیلدهای private را از طریق یک متد public به‌صورت read-only در آوردیم.

همان‌طور که می‌دانید یکی از وظایف آمپلی‌فایر، بلند کردن و کم کردن صدا است. در این‌جا صدای آمپلی‌فایر ابتدا روی صفر تنظیم شده و حداکثر صدای آن ۵۰ در نظر گرفته شده است. در مرحله‌ی بعد دو متد به این کلاس اضافه می‌کنیم که وظیفه‌ی بلند کردن و کم کردن صدا را بر عهده دارند. این دو متد همچنین بررسی می‌کنند که صدا از عدد ۵۰ بیشتر و از عدد صفر کمتر نشود:

```
using System;
class AmpDemo
{
    static void Main()
    {
        Amplifier Kenwood = new Amplifier("KENWOOD", "RA-5000", 25, 5, true, 50);

        Console.WriteLine("Amplifier simulation");
        Console.WriteLine("  Name and Model: " + Kenwood.GetNameAndModel());
        Console.WriteLine("  Effictive Power: " + Kenwood.GetEffectivePower());
        Console.WriteLine("  Multichannel: " + Kenwood.GetMultichannel());
        Console.WriteLine("  3D Surround Sound: " + Kenwood.GetThreeDSurroundSound());

        Console.WriteLine();
        // Increase Volume
        Kenwood.IncreaseVolume(10);
        Kenwood.IncreaseVolume(20);
        Kenwood.IncreaseVolume(10);
        Kenwood.IncreaseVolume(5);
        Kenwood.IncreaseVolume(20);

        // Decrease Volume
        Kenwood.DecreaseVolume(10);
        Kenwood.DecreaseVolume(30);

        // Increase again...
        Kenwood.IncreaseVolume(20);
        Kenwood.IncreaseVolume(10);
    }
}
class Amplifier
{
    private string Name;
    private string Model;
    private int Watt;
    private int Multichannel;
    private int EffectivePower;
    private bool ThreeDSurroundSound;
    private int CurrentVolume;
    private int MaxVolume;

    public Amplifier(string name, string model, int watt, int multichannel,
        bool threeDSurroundSound, int maxVolume)
    {
        Name = name;
        Model = model;
        Multichannel = multichannel;
        Watt = watt;
        ThreeDSurroundSound = threeDSurroundSound;
        EffectivePower = watt * multichannel;
        CurrentVolume = 0;
        MaxVolume = maxVolume;
    }

    public void IncreaseVolume(int increaseValue)
    {
        if (CurrentVolume + increaseValue <= MaxVolume)
        {
            CurrentVolume += increaseValue;
            Console.WriteLine("Increaseing Volume\n  Current Volume: " + CurrentVolume);
            Console.WriteLine();
        }
```

```
        else
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine("Danger!\nloud volume can hurt your ears!");
            Console.WriteLine("Max Volume is {0}. You can't violate this", MaxVolume);
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine();
        }
    }

    public void DecreaseVolume(int decreaseValue)
    {
        if (CurrentVolume - decreaseValue >= 0)
        {
            CurrentVolume -= decreaseValue;
            Console.WriteLine("Decreaseing Volume\n  Current Volume: " + CurrentVolume);
            Console.WriteLine();
        }
        else
        {
            CurrentVolume = 0;
            Console.WriteLine("Decreaseing Volume\n  Current Volume: " + CurrentVolume);
            Console.WriteLine();
        }
    }

    public int GetEffectivePower()
    {
        return EffectivePower;
    }

    public string GetNameAndModel()
    {
        return Name + " " + Model;
    }

    public int GetMultichannel()
    {
        return Multichannel;
    }

    public string GetThreeDSurroundSound()
    {
        if (ThreeDSurroundSound)
            return "Yes";
        else
            return "No";
    }
}
```

همان‌طور که می‌بینید، با صدا زدن متد افزایش صدا، به مراتب صدا را زیاد کرده‌ایم و هنگامی که قصد تجاوز از حد را داشته‌ایم با پیغام خطر (قرمز رنگ) مواجه شده‌ایم. سپس صدا را توسط متد کاهش صدا، کاهش و مجدداً توسط متد افزایش صدا، افزایش داده‌ایم.

تمرین

تمرین شماره ۱۳: در تمرین شماره ۱۲ دفترچه تلفن ساده‌ای ساختیم که شی‌گرا نبود. در تمرین شماره ۱۳ باید دفترچه تلفنی بسازید که در آن فقط از متد static استفاده شده باشد. برای انجام این تمرین نیازی به ساختن کلاس و شیء نیست (در تمرین‌های بعد کلاس و شیء می‌سازید). برای حل این تمرین می‌توانید از الگوی زیر پیروی کنید:

```
using System;
class SimplePhoneBook
{
    static void Main()
    {
        while (true)
        {
            ShowMenu();

            // Use methods to build a phonebook

            Console.ReadLine();
        }
    }

    // You can use void methods or some ret-type else
    static void ShowMenu()
    {
        Console.Clear();
        Console.WriteLine("---- Simple Phonebook ----");
        Console.WriteLine();
        Console.WriteLine("1. Add");
        Console.WriteLine("2. Search");
        Console.WriteLine("3. Show all");
        Console.WriteLine("4. Exit");
        Console.WriteLine();
        Console.Write("Choose a number: ");
    }

    static void Add()
    {
        // Add statements
    }

    static void Search()
    {
        // Search statements
    }

    static void ShowAll()
    {
        // Show statements
    }
}
```

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.