عنوان: **اثبات قانون مشاهده‌گر در برنامه نویسی**
نویسنده: میثم نوایی
تاریخ: ۱۳۹۳/۰۷/۲۷ ۱:۳۰
آدرس: www.dotnettips.info
گروه‌ها: C#, Threading, Random

امروز حین کدنویسی به یک مشکل نادر برخورد کردم. کلاسی پایه داشتم (مثلا Person) که یک سری کلاس دیگر از آن ارث بری میکردند (مثلا کلاس‌های Student و Teacher).در اینجا در کلاس پایه بصورت اتوماتیک یک ویژگی(Property) را روی کلاس‌های مشتق شده مقدار دهی میکردم؛ مثلا به این شکل:

```
public class Person
    {
        public Person()
        {
            personId= this.GetType().Name + (new Random()).Next(1, int.MaxValue);
        }
     }
```

سپس در یک متد مجموعه‌ای از Studentها و teacherها را ایجاد کرده و به لیستی از Personها اضافه میکنم:

```
var student1=new Student(){Name="Iraj",Age=21};
var student1=new Student(){Name="Nima",Age=20};
var student1=new Student(){Name="Sara",Age=25};
var student1=new Student(){Name="Mina",Age=22};
var student1=new Student(){Name="Narges",Age=26};
var teacher1=new Student(){Name="Navaei",Age=45};
var teacher2=new Student(){Name="Imani",Age=50};
```

اما در نهایت اتفاقی که رخ میداد این بود که PersonId همه Studentها یکسان می‌شد ولی قضیه به همین جا ختم نشد؛ وقتی خط به خط برنامه را Debug و مقادیر را Watch می‌کردم، مشاهده می‌کردم که PersonId به درستی ایجاد می‌شود.
در فیزیک نوین اصلی هست به نام عدم قطعیت هایزنبرگ که به زبان ساده میتوان گفت نحوه رخداد یک اتفاق، با توجه به وجود یا عدم وجود یک مشاهده‌گر خارجی نتیجه‌ی متفاوتی خواهد داشت.
کم کم داشتم به وجود قانون مشاهده‌گر در برنامه نویسی هم ایمان پیدا میکردم که این کد فقط در صورتیکه آنرا مرحله به مرحله بررسی کنم جواب خواهد داد!
جالب اینکه زمانیکه personId را نیز ایجاد میکردم، یک دستور برای دیدن خروجی نوشتم مثل این

```
public class Person
    {
        public Person()
        {
            personId= this.GetType().Name + (new Random()).Next(1, int.MaxValue);
            Debug.Print(personId)
        }
     }
```

در این حالت نیز دستورات درست عمل میکردند و personId متفاوتی ایجاد می‌شد! قبل از خواندن ادامه مطلب شما هم کمی فکر کنید که مشکل کجاست؟
این مشکل ربطی به قانون مشاهده‌گر و یا دیگر قوانین فیزیکی نداشت. بلکه بدلیل سرعت بالای ایجاد وهله ها(instance) از کلاسی‌های مطروحه (مثلا در زمانی کمتر از یک میلی ثانیه) زمانی در بازه یک کلاک CPU رخ می‌داد.

هر نوع ایجاد کندی (همچون نمایش مقادیر در خروجی) باعث می‌شود کلاک پردازنده نیز تغییر کند و عدد اتفاقی تولید شده فرق کند.
همچنین برای حل این مشکل میتوان از کلاس تولید کننده اعداد اتفاقی، شبیه زیر استفاده کرد:

```
using System;
using System.Threading;

public static class RandomProvider
```

```
{
    private static int seed = Environment.TickCount;

    private static ThreadLocal<Random> randomWrapper = new ThreadLocal<Random>(() =>
        new Random(Interlocked.Increment(ref seed))
    );

    public static Random GetThreadRandom()
    {
        return randomWrapper.Value;
    }
}
```

نظرات خوانندگان

نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۳/۰۷/۲۷ ۱۴:۴

به جای کلاس Random از جایگزین بهتر آن RNGCryptoServiceProvider استفاده کنید و دوباره برنامه رو تست کنید.

نویسنده: سعید
تاریخ: ۱۳۹۳/۰۸/۰۶ ۱۲:۴۸

شما اگر برای تولید عدد تصادفی از یک آبجکت کلاس Random استفاده می‌کردید به چنین مشکلی بر نمی‌خوردید.

نویسنده: میثم نوایی
تاریخ: ۱۳۹۳/۰۸/۰۶ ۱۳:۲

اگه مطلب را کامل مطالعه بفرمایید و شبیه سازی کنید به مشکل مطروحه برخواهید خورد.

نویسنده: سعید
تاریخ: ۱۳۹۳/۰۸/۰۶ ۱۴:۱۵

من نیاز مسئله شما را به صورت زیر نوشتم و برای هر شی Person یک مقدار متفاوت دارم.

```
class Program
    {
        static void Main(string[] args)
        {
            var lst = new List<Person>();
            Random rnd = new Random();
            for (int i = 0; i < 50; i++)
            {
                lst.Add(new Person(rnd));
            }


            foreach (var item in lst)
            {
                Console.WriteLine(item.PersonId);
            }

            Console.ReadKey();
        }
    }

    public class Person
    {
        public Person(Random rnd)
        {
            PersonId = this.GetType().Name + rnd.Next(1, int.MaxValue);
        }

        public string PersonId { get; set; }
    }
```

البته من فکر میکنم اگر شما نیاز به یک ای دی منحصر به فرد دارید راه بهتر استفاده از GUID باشد.

نویسنده: میثم نوایی
تاریخ: ۱۳۹۳/۰۸/۰۶ ۱۴:۴۱

ممنون بابت نظراتتان.
عرض کردم مطلب را کامل بخوانید.این مشکل در سیستم هایی با پردازش بالا و در زمان‌های هزارم ثانیه رخ میدهد.به این معنی که کلاس Random مقادیر متفاوتی ایجاد نمیکند.و عملا کارایی ندارد.