Zpravodaj moderníProgramování 02/2012: Úskalí práce s náhodou

Předpoklad ke studiu tohoto čísla: 9. kapitola učebnice pro začátečníky (modrá), pro závěrečnou část také učebnice pro středně pokročilé (žlutá).

Jak se pracuje s náhodnými čísly, je popsáno v 9. kapitole učebnice pro začátečníky:

- **Jednou** se vytvoří generátor náhodných čísel jako instance třídy `Random`;
- **Opakovaně** se volá metoda `Next` této instance pro každé náhodné číslo, které potřebujeme.

Na začátku získáme opravdu náhodné číslo, poté opakovaně pracujeme s čísly pseudonáhodnými.

Přestože tato teorie je jasná, často se chybuje v její aplikaci a netýká se to pouze začátečníků. Všiml jsem si, že i třeba p. MacDonald, který napsal řadu pěkných, celosvětově prodávaných knih (o programování, nikoli o hamburgerech), s náhodou pracuje nesprávně. V tomto čísle Zpravodaje si tedy na příkladech ukážeme, jaké chyby mohou vznikat a jak se jim vyhnout.

Úskalí horní meze

Než se pustíme do hlavního problému, připomenu na rozehřátí jinou, třebaže jednodušší záludnost. Nic nového, pokud jste důkladně prostudovali 9. kapitolu, nicméně je to věc, ve které se často chybuje, takže není od věci si to připomenout.

Jedná se o to, že metoda `Next` vyžaduje ve druhém parametru horní mez rozmezí náhodných čísel **zvětšenou o jedničku**. Máme-li tedy v proměnné `náhoda` instanci `Random` a chceme-li získat náhodné číslo z rozmezí 10, 11, 12, …, 19, 20, zapíšeme to

```
int náhodnéČíslo = náhoda.Next(10, 20+1);
```

nebo samozřejmě nemusíme to „plus jedna“ explicitně vypisovat:

```
int náhodnéČíslo = náhoda.Next(10, 21);
```

Pokud ve druhém parametru uvedete pouze dvacítku, dvacítka nikdy nepadne. Můžete si to ověřit ručně, nebo i automaticky programem podobným tomu, jaký je na str. 46 žluté učebnice.

Proč to tak je? Dědictví minulosti, nedokonalost v Redmondu, jak chcete…

Hlavní úskalí – chybně opakované vytváření generátoru

Pokud již mozek dosáhl provozní teploty, můžeme se pustit do hlavního problému. Mnoho lidí, třeba nevědomky, nerespektuje pravidlo uvedené v úvodu a vytváří generátor vícekrát, typicky před každým náhodným číslem.

Uvažujme program simulující kostku ze str. 127 modré učebnice. Správné řešení ukládá generátor do členské proměnné okna:

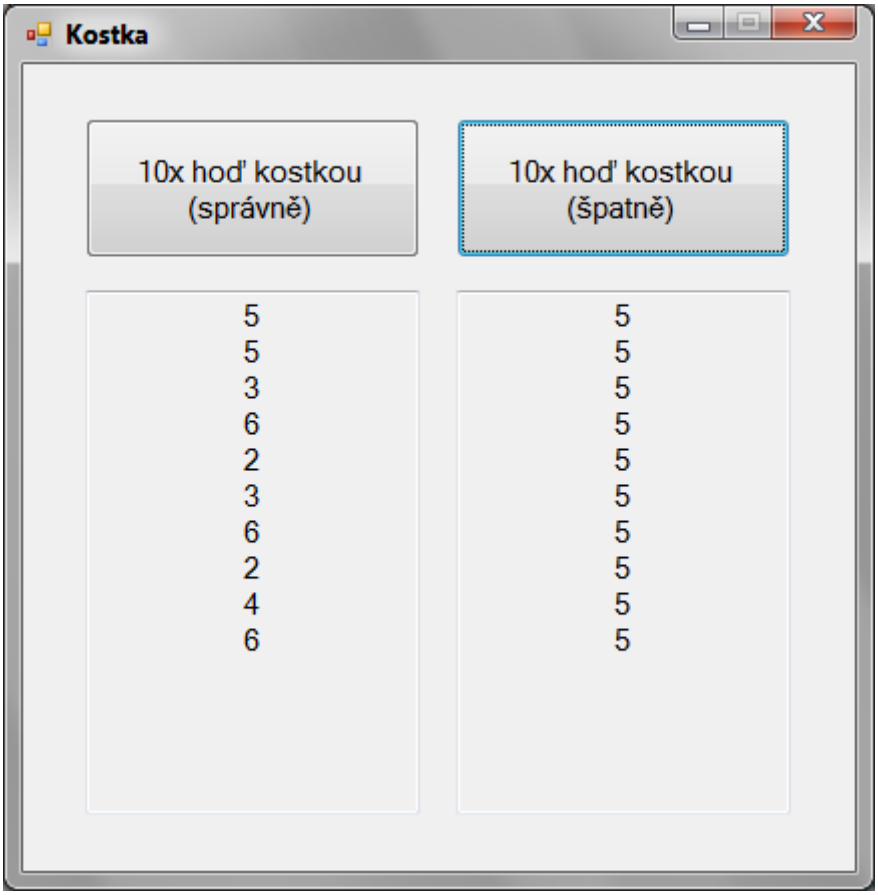
```
Random náhoda = new Random();

private void tlačítkoHoď_Click(object sender, EventArgs e)
{
   int číslo = náhoda.Next(1, 6+1);
   poleČíslo.Text = číslo.ToString();
}
```

Nesprávné řešení použije proměnnou lokální a chybně vytváří vždy nový generátor pro nové náhodné číslo:

```
private void tlačítkoHoď_Click(object sender, EventArgs e)
{
   Random náhoda = new Random();
   int číslo = náhoda.Next(1, 6+1);
   poleČíslo.Text = číslo.ToString();
}
```

Při běžném testování, kdy člověk mačká tlačítko jednou za čas, se problém viditelně neprojevuje. Pokud však za nás kostkou opakovaně hodí počítač, který je mnohem rychlejší, budeme se divit.

Ilustrujme si problém na programu s tímto uživatelským rozhraním:



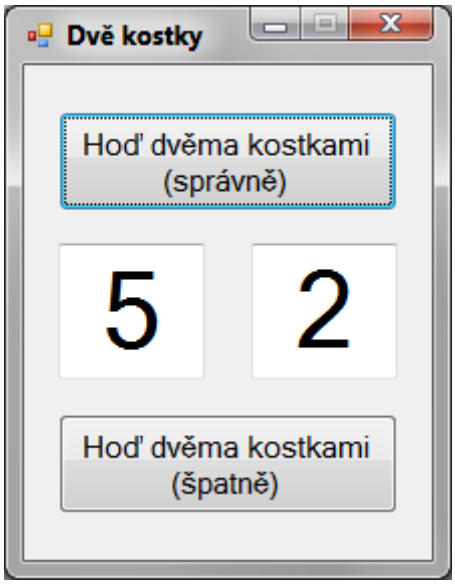
Kód:

```
Random správnáNáhoda = new Random();

private void tlačítkoSprávně_Click(object sender, EventArgs e)
{
   poleSprávně.Text = "";
   for (int počet = 0; počet < 10; počet++)
   {
       int náhodnéČíslo = správnáNáhoda.Next(1, 6+1);
       poleSprávně.Text += náhodnéČíslo.ToString() +
                           Environment.NewLine;
   }
}

private void tlačítkoŠpatně_Click(object sender, EventArgs e)
{
   poleŠpatně.Text = "";
   for (int počet = 0; počet < 10; počet++)
   {
       Random špatnáNáhoda = new Random();
       int náhodnéČíslo = špatnáNáhoda.Next(1, 6 + 1);
       poleŠpatně.Text += náhodnéČíslo.ToString() +
                          Environment.NewLine;
   }
}
```

Otestujte, uvidíte.

Jiný projev téhož problému

Členská proměnná není vždy spásou. Prostě musíte pamatovat na to, že `Random` instancujete vždy jen jedenkrát. Vyzkoušejme program, který po stisku tlačítka hodí dvěma kostkami současně:



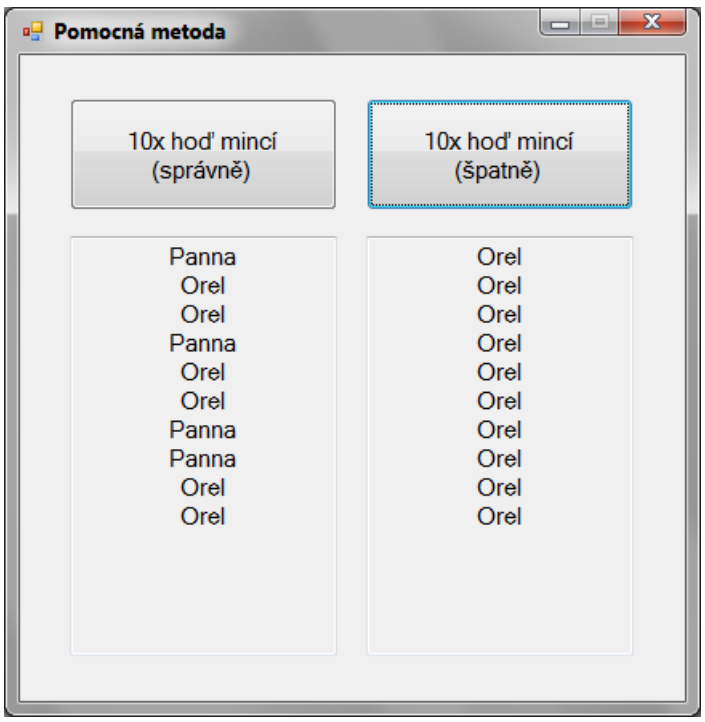
Člověk může mít tendenci vidět ty dvě kostky jako dvě instance `Random`, to ale právě vede k chybě:

```
// Správně 1 generátor
Random náhoda = new Random();

// Špatně 2 generátory
Random prvníKostka = new Random();
Random druháKostka = new Random();

private void tlačítkoHoďSprávně_Click(object sender, EventArgs e)
{
   polePrvní.Text = náhoda.Next(1, 6+1).ToString();
   poleDruhá.Text = náhoda.Next(1, 6+1).ToString();
}

private void tlačítkoHoďŠpatně_Click(object sender, EventArgs e)
{
   polePrvní.Text = prvníKostka.Next(1, 6+1).ToString();
   poleDruhá.Text = druháKostka.Next(1, 6+1).ToString();
}
```

Poslední projev - pomocná metoda nebo komponenta

Ještě více se problém může skrýt, pokud vytváříme nějakou opakovaně použitelnou komponentu pracující s náhodou. Pro jednoduchost uvažujme nikoli komponentu v samostatné DLL, nýbrž pomocnou metodu v tomtéž projektu, ve kterém ji použijeme.

Vytvoříme metodu `HoďMincí`, jejímž výsledkem bude hodnota výčtu Panna - Orel. Metodu pak 10× za sebou zavoláme z následujícího okna:



Do nového projektu vložte třídu `Pomůcky` a kód `Pomůcky.cs` upravte následovně (všimněte si, na začátku definujeme vlastní výčtový typ `Mince`):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Pomocná_metoda
{
    enum Mince { Panna, Orel }

    class Pomůcky
    {
        private static Random náhoda = new Random();

        public static Mince HoďMincíSprávně()
        {
            int náhodnéČíslo = náhoda.Next(1, 2+1);
            if (náhodnéČíslo == 1)
                return Mince.Panna;
            else
                return Mince.Orel;
        }

        public static Mince HoďMincíŠpatně()
        {
            Random špatnáNáhoda = new Random();
            int náhodnéČíslo = špatnáNáhoda.Next(1, 2+1);
            if (náhodnéČíslo == 1)
                return Mince.Panna;
            else
                return Mince.Orel;
        }
    }
}
```

Kód okna pak bude tento:

```
Random správnáNáhoda = new Random();

private void tlačítkoSprávně_Click(object sender, EventArgs e)
{
    poleSprávně.Text = "";
    for (int počet = 0; počet < 10; počet++)
        poleSprávně.Text += Pomůcky.HoďMincíSprávně().ToString() +
                            Environment.NewLine;
}

private void tlačítkoŠpatně_Click(object sender, EventArgs e)
{
    poleŠpatně.Text = "";
    for (int počet = 0; počet < 10; počet++)
        poleŠpatně.Text += Pomůcky.HoďMincíŠpatně().ToString() +
                           Environment.NewLine;
}
```

V čem je tento program z uvedených programů nejobtížnější? Člověk je již poučen, ví, že nemá `Random` instancovat vícekrát, ale když proměnnou `náhoda` ve třídě `Pomůcky` deklarujete, jak jste zvyklí, program se nesestaví. Statická metoda může totiž pracovat pouze se statickými proměnnými. Proto u proměnné `náhoda` musí být modifikátor `static`! Statická metoda ke svému volání nepotřebuje předchozí vytvoření instance třídy, může se tudíž spoléhat pouze na ty složky, které rovněž nevyžadují vytvoření instance třídy.

Radek Vystavěl, 12. února 2012

Pokud Vám Zpravodaje moderníProgramování připadají užitečné, doporučte jejich odběr svým známým. Mohou se přihlásit na webu www.moderniProgramovani.cz.