Jazyk C# 2

2. seminář

Večeřa J., Janoštík R.

Univerzita Palackého v Olomouci

22.2.2023

Reakce na úkoly – Obecné

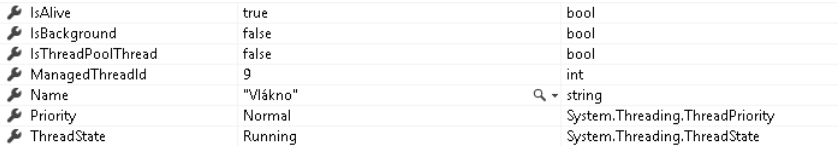
- Používejte Try-catch
- 1 class = 1 file
- Komentujte kód – ALE!
- Pomocné metody by měly být `private`
- Coding Convetions `https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions`
  - Je dobré držet jednotný styl – člověk se lépe orientuje

Vlákna

- Doteď platilo náš proces $\approx$ 1 vlákno $\Rightarrow$ 1 tok zpracování
- Možnost více vláken $\Rightarrow$ paralelní výpočty, oddělení GUI a „enginu“
- Vhodné i na jednoprocesorových počítačích (Preemptivní multitasking)
- Na víceprocesorových/jaderných počítačích lepší využití HW

Třída System.Threading.Thread

- `Thread.CurrentThread` – aktuální vlákno
  - `CurrentCulture` a `CurrentUICulture`
  - Základní vlastnosti



IsAlive	true	bool
IsBackground	false	bool
IsThreadPoolThread	false	bool
ManagedThreadId	9	int
Name	"Vlákno"	string
Priority	Normal	System.Threading.ThreadPriority
ThreadState	Running	System.Threading.ThreadState

- `ThreadPriority` má hodnoty `Highest, AboveNormal, Normal, BelowNormal, Lowest`

- `ThreadState` má hodnoty `Aborted,AbortRequested, Background, Running, Stopped, StopRequested, Suspended, SuspendRequested, Unstarted, WaitSleepJoin`

Vytvoření nového vlákna

- Konstruktor má parametr delegáta se signaturou

```
1 public delegate void ThreadStart();
2 Thread worker = new Thread(new ThreadStart(RunFactorial));
3 worker.Start();
```

- Spuštění vlákna `worker.Start()`

- Předání parametrů

```
1 public delegate void ParametrizedThreadStart(object o);
2 worker.Start(param);
```

- Nebo

```
1 Thread t = new Thread(new ThreadStart(delegate {
2    Console.WriteLine(Fac(7)); }));
3 t.Start();
4 Thread t2 = new Thread(() => Console.WriteLine(Fac(7)));
```

Manipulace s vlákny

- Pozastavení činnosti vlákna – `thread.Suspend();`
- Opětovné spuštění – `thread.Resume();`
- Zastavení činnosti – `thread.Abort();` – využití výjimek
- Ke změně může dojít se zpožděním
- Počkání na dokončení činnosti – `thread.Join();`

Synchronizace

- Při přístupu ke sdíleným proměnným více vlákny může dojít k nechtěnému stavu
- Problém atomicity operací: $x = x + 5$ vnitřně není jedna operace
- Příkaz `lock (o){ }` zapouzdří objekt o do kritické sekce (vzájemné vyloučení)
- Pozor na deadlock (zablokování) a zamykání stringů – viz. příklad

Async, Await a Tasky

- Je abstrakce nad systémem vláken, objekt
- Velmi zjednodušeně:
  - `async` a `await`

```
1  public async void DoWork()
2 { await Work(); }
```

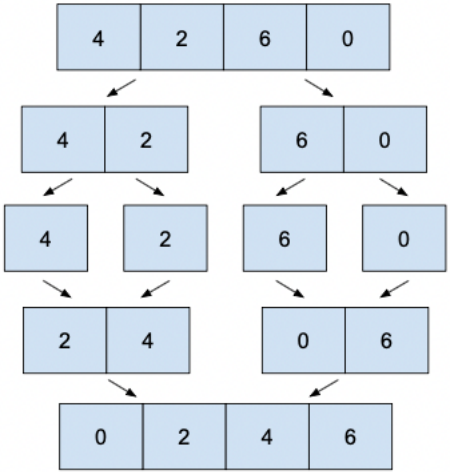
  - `Task<T>`

```
1 public Task Work()
2 {
3     return Task.Factory.StartNew(() =>
4     {
5         Thread.Sleep(4000);
6      });
7 }
```

  - Ukázka

Merge Sort

- Opakování



Úkol (1/3)

- Naprogramujte MergeSort tak, aby třídění polovin využívalo vlákna takto:
  - Bude si zaznamenávat hloubku zanoření
  - Od určité hloubky již nebude vytvářet nová vlákna, ale dopočítá svou část v jednom vlákně
- Naplnit pole více než 10 000 000 prvky ($\pm$ kolik se vám vleze do paměti)
- Porovnat čas běhu bez použití vláken a s různými hloubkami zanoření $(0, 1, 2, 3)$

Úkol (2/3)

- Náznak volání

```
1   int[] array = GetRandomArray(10000000);
2   // zapamatovat cas spusteni
3   ParalelMergeSort(array, 0); // Degradace na sekvencni
4   // vypsat dobu behu
5   array = GetRandomArray(10000000);
6   // zapamatovat cas spusteni
7   ParalelMergeSort(array, 1); // hloubka 1
8   // vypsat dobu behu
9   array = GetRandomArray(10000000);
10  // zapamatovat cas spusteni
11  ParalelMergeSort(array, 2); // hloubka 2
12  // vypsat dobu behu
```

Úkol (3/3)

- Měření času

```
1 DateTime startTime = DateTime.Now;
2 DoSomething();
3 DateTime endTime = DateTime.Now;
4 TimeSpan totalTimeTaken = endTime.Subtract(startTime);
```

- vs přesnější

```
1 Stopwatch sw = Stopwatch.StartNew();
2 DoSomething();
3 sw.Stop();
4 Console.WriteLine("Cas behu: {0}ms",
      sw.Elapsed.TotalMilliseconds);
```

- Bonus 0.5 bod: Zakomponovat sdílenou proměnou