24. Vlákna, Paralelní programování, Asynchronní metody, Concurrent design patterns

Vlákna

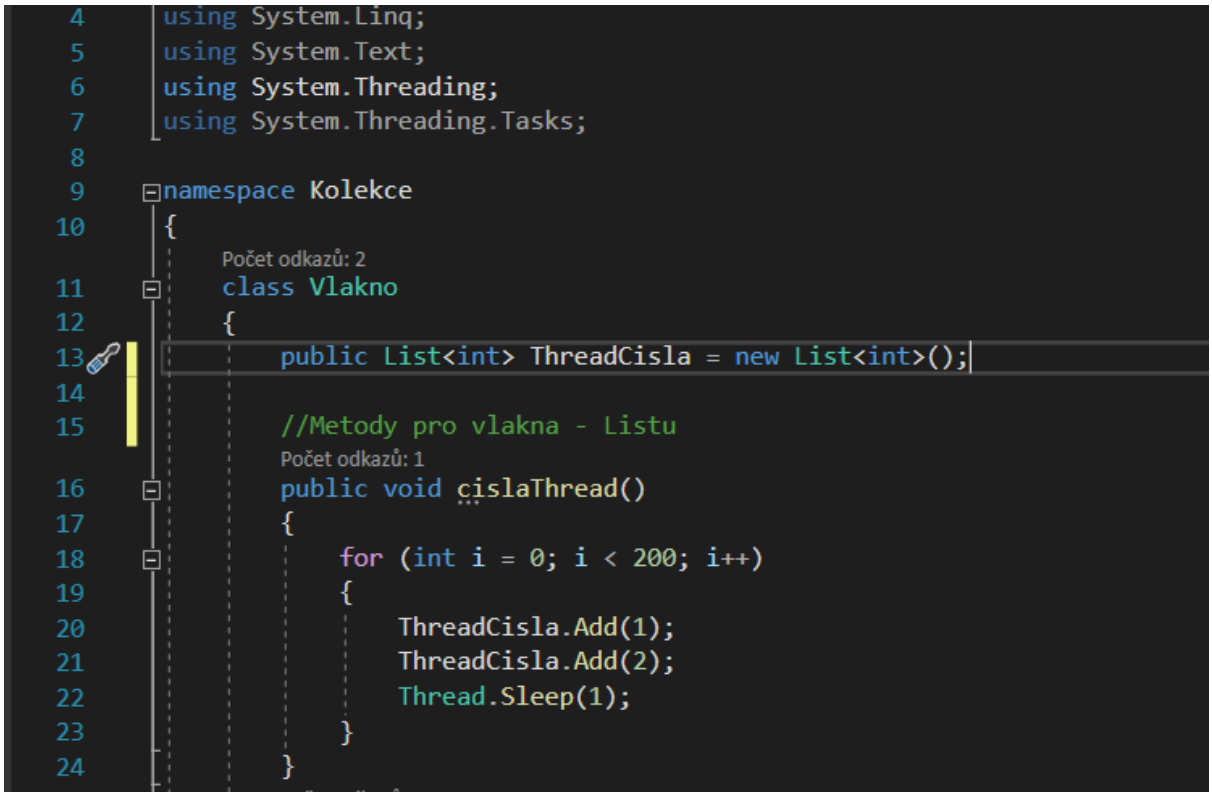
- Můžeme spouštět několik částí kódu najednou, každou část na samostatném vlákně
- Každá aplikace má minimálně jeden proces, ve kterém běží hlavní vlákno, případně další
- Vlákna běží na jádrech (pokud více jádrový procesor, tak se rozdělí)
- Zbytečně mnoho vláken je nevýhodou
    - Vyšší nároky na procesor
    - Nevhodné přepínání vláken
- Vlákna se mohou přerušovat, respekt dochází k přeskočení na jiné vlákno, z toho důvodu je nutné
- Uzamčení pro ostatní vlákna, dokud se nedokončí toto nikdo jiný k němu nemůže přistupovat

```
Static readonly object locker = new Object();
Static void GoU{
Lock(locker){
If…
}
}
```

- Vlákno můžeme uspat metodou Sleep()

```
//16 vlakna
Thread vlaknoStack = new Thread(v.StackAddThread);
Thread vlaknoStack2 = new Thread(v.StackAddThread2);
vlaknoStack.Start();
vlaknoStack2.Start();
vlaknoStack.Join();
vlaknoStack2.Join();
foreach (var val in v.stackThread)
{
    Console.Write(val + ", ");
}

Console.WriteLine();
Console.WriteLine("16.Očekávaný výsledek v 1000");
Console.Write("Realny " + v.stackThread.Count());
Console.WriteLine();
Console.WriteLine();
Console.WriteLine("Konec");
```



```
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace Kolekce
{
    Počet odkazů: 2
    class Vlakno
    {
        public List<int> ThreadCisla = new List<int>();

        //Metody pro vlakna - Listu
        Počet odkazů: 1
        public void cislaThread()
        {
            for (int i = 0; i < 200; i++)
            {
                ThreadCisla.Add(1);
                ThreadCisla.Add(2);
                Thread.Sleep(1);
            }
        }
```

Paralelní programování

- Řeší problémy vznikající při spolupráci více procesů na jedné úloze
- Jakmile existuje více zájemců o jeden zdroj, dochází ke konfliktu
- Lze vyřešit pravidly nebo komunikací
- Slovník
    - Kritická sekce
        - Část kódu kde může dojít ke konfliktu mezi vlákny
        - Typicky kód ve kterém se pracuje se sdílenými datovými strukturami
    - Zámek
        - Mechanismus pro řízení přístupu vláken ke kritické sekci
    - Mutex
        - Typ zámku, který do kritické sekce vpustí jen jedno vlákno současně
        - Ostatní vlákna jsou před vstupem do kritické sekce zablokována

Asynchronní metody

- Modifikátor async
    - Určuje, že je metoda asynchronní
        - Ne že je automaticky zpracována asynchronně, ale obsahuje volání asynchronních metod pomocí operátorů await
        - Zpracovávaná v hlavním vlákně tak dlouho, jak je to jen možné
- Operátor await
    - Vrací kontrolu, dokud není úloha označená operátorem await dokončená
        - Nejde o čekání na zpracování (blokování hl vlákna sychronní operací), ale když taková úloha ještě neskončila, je zbytek metody označen jako pokračování této úlohy a to je vyvoláno po skončení této úlohy

        - Určuje, že od tohoto místa může metoda být zpracována v jiném než hlavním vlákně
        - Prakticky nahrazuje psaní pokračujících úloh
            - Umožňuje TPL – pokračování tedy vytváří kompilátor a ne programátor

Concurrent design patterns

- Typ vzoru, který se zabývá více vláknovými programy
- Active object
    - Tento návrhový vzor odděluje spouštění metod od provádění metod, přičemž spouštění metod může být ve svém vlastním vlákně. Cílem je přidat souběžnost použitím asynchronních volání metod a plánovače, který obsluhuje požadavky.
- Event-based asynchronous
    - Na událostech založený asynchronní návrhový vzor řešící problémy s Asynchronním vzorem, které nastávají ve vícevláknových programech.
- Balking
    - Tento vzor je softwarovým vzorem, který na objektu vykoná nějakou akci, pouze pokud je objekt v určitém stavu.
- Double checked Locking
    - Tento vzor je také znám jako „optimalizace zamykání s dvojnásobnou kontrolou“. Návrh je vytvořen tak, aby zredukoval zbytečné náklady na získávání zamčení tím, že nejdříve otestuje kritérium pro zamčení nezabezpečeným způsobem ('lock hint'). Pouze pokud uspěje, pak se opravdu zamkne.
    - Tento vzor může být nebezpečný, pokud je implementován v některých kombinacích programovacích jazyků a hardwaru. Proto je někdy považován také za proti-vzor.
- Guarded
    - Jde o vzor obstarávající operace, které požadují uzamčení a navíc mají nějakou podmínku, která musí být splněna předtím, než může být operace provedena.
- Monitor object
    - Monitor je přístup k synchronizaci dvou nebo více počítačových úloh, které používají sdílené zdroje, zpravidla hardwarové zařízení nebo sadu proměnných.
- Read write lock
    - Tento vzor, také známý jako RWL, je vzor, který umožňuje souběžný přístup k objektu pro čtení, ale vyžaduje exkluzivní přístup pro zápis.
- Scheduler
    - Jde o souběžný vzor, který se používá pro explicitní kontrolu, kdy mohou vlákna vyvolávat jednovláknový kód.
- Thread pool
    - V bazénku vláken je vytvořen nějaký počet vláken pro řešení nějakého množství úloh, které jsou organizovány ve frontě. Zpravidla je výrazně více úloh než vláken.
- Thread-specific storage
    - Thread-local storage (TLS) je programovací metoda, která používá statickou nebo globální paměť lokálně pro vlákno.
- Reactor
    - Jde o vzor používaný pro vyřizování požadavků na službu, které jsou z jednoho nebo více vstupů doručovány správci služeb. Správce služeb rozdělí příchozí požadavky a přidělí je synchronně přidruženým vyřizovačům požadavků.