10 - Paralelní programování

Jamie king - url

Asynchronní a paralelní programování

- standardně se kód vykonává postupně (synchronně), tedy řádek po řádku
- proces běží v jednom vlákně
- asynchronní programování vytváří podprogramy, které běží současně
  - nazývají se úlohy
- kód pokračuje dál nehledě na to, zda úloha skončila, nebo ne
- tyto podprogramy vykonávají kód klasicky synchronně, ovšem může jich běžet více naráz
  - každý běží ve vlastním vlákně

Thread

- vlákno
- v C# jde o třídu
- vytvoření instance této třídy vytvoří nové vlákno v procesoru
- lze s ní pracovat (metody `Sleep()`, `Join()`, `Start()`, `Abort()`)

Task

- asyncronní úloha
- v C# jde o třídu
- vytvoření instance této třídy vytvoří novou úlohu
- třída je generická
- obsahuje informace o úloze
- úloh může běžet více naráz

Lock

- zámek
- jednotlová vlákna mohou přistupovat k datům, která jsou právě zpracovávaná
  - vede k nežádoucím výsledkům
- zámek jako parametr má jakýkoliv objekt
  - tímto objektem se daná část kódu zamkne
- zamknutá část se vždy musí dokončit, pokud se začne vykonávat
- jde o syntaktický cukr, výsledek se převádí na funkci `Monitor`
  - jednodušší zápis pro programátora

Použití paralelního programování

- hodí se pro výpočty, které pracují s velkým množstvím vzájemně nezávislých dat
- je nutné mít hardware, který toto umožňuje
  - pokud hardware nemá více jader, jediná možnost, kde se dá paralelizace použít byl multitasking OS
- na hardwaru s více jádry je výhodnější spouštět procesy naráz

Async / await

- klíčová slova v C# která umožňují vytváření asynchronních funkcí
- `async` určuje metodu jako asynchronní
- aby byla metoda asynchronní, musí zároveň v jejím těle obsahovat `await`
  - při použití `await` se vyčká na dokončení metody
- metoda by zároveň měla vracet instanci `Task`