# 10 - Paralelní programování

# Asynchronní programování

* Asynchronní volání neblokují chod volajícího
* Cílem je responzivita aplikace (například aby UI nezamrzlo při stahování dat z databáze)

# Paralelní programování

* Rozdělení práce do několika vláken
* Cílem je výkon (rychlost) aplikace
* Např. Quicksort

# Thread

* Třída Thread reprezentuje vlákno (tok)
* Aplikace může běžet ve více paralelních vláknech
* Start() – spuštění vlákna
* Join() – čekání na dokončení vlákna
* Sleep(milisekundy) – pozastavení běhu vlákna
* Abort() – zabití vlákna

# Task

* Spouštíme asynchronní operaci a čekáme na její výsledek
* V C# jde o třídu
* Vytvoření instance této třídy vytvoří novou úlohu
* Třída je generická
* Obsahuje informace o úloze
* Úloh může běžet více narázObsah obrázku text

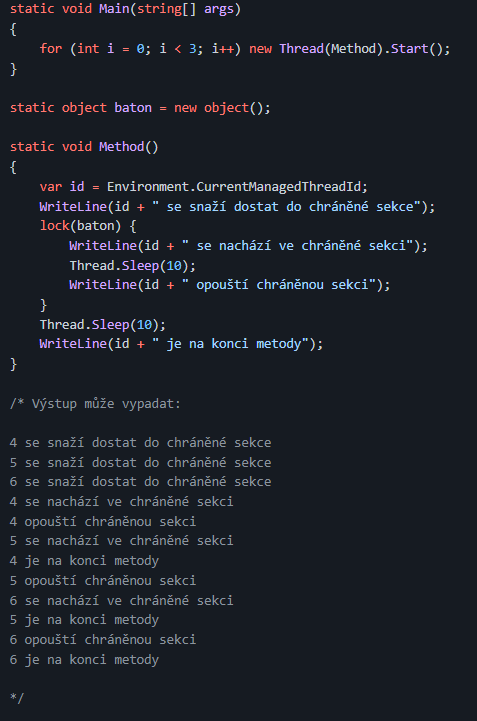
  Popis byl vytvořen automaticky

# Parallel.For

* Paralelní průchody cyklem
* Parallel.For(odInkluzivni, doExkluzivni, (iteracniPromenna) => { });
* Parallel.ForEach(kolekce, (iteracniPromenna) => { });

# Synchronizace vláken

* Synchronizace vláken – koordinovaný přístup ke sdíleným prostředkům
* Při zpracování dat vláknem by k těmto datům neměla mít ostatní vlákna přístup
  + Problém nastává, když vlákno s daty provádí neatomickou operaci a jiné vlákno začne s těmito daty také pracovat
  + (data ještě nejsou plně zpracovaná a připravená k další akci)

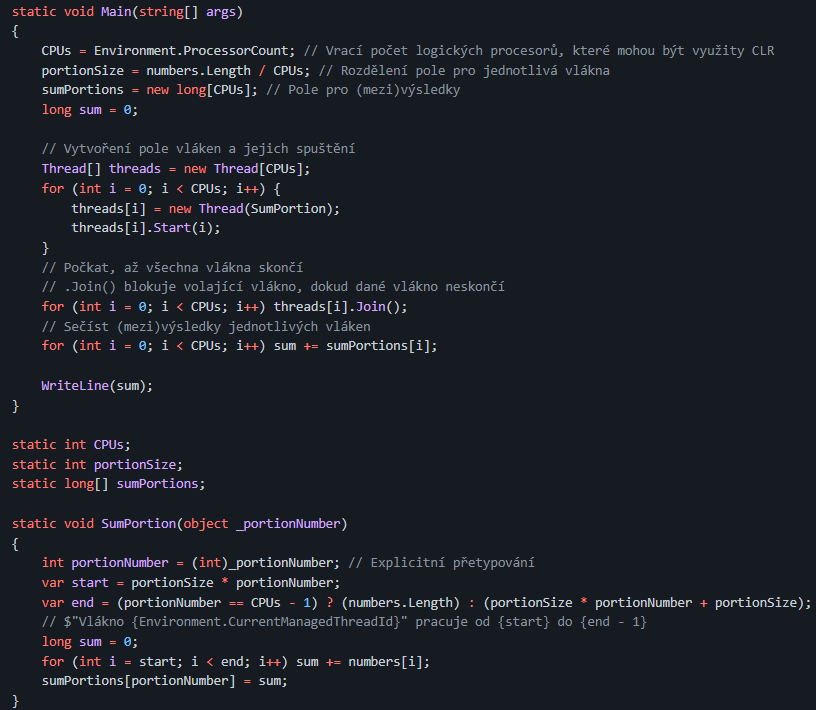


# Async, await

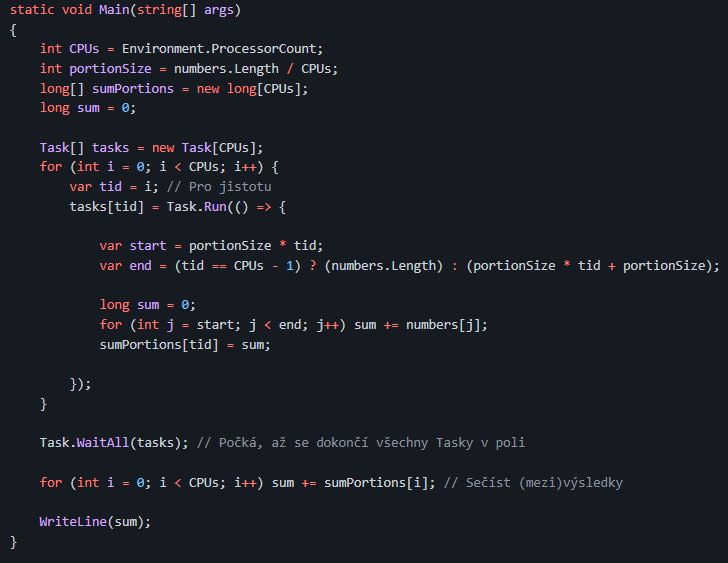
* Klíčová slova usnadňující vytváření asynchronního kódu
* async označí metodu jako asynchronní
* aby byla metoda asynchronní, musí zároveň v jejím těle obsahovat await (vyčkání na dokončení metody)
* Asynchronní metody vrací Task<T>

# Příklady

* Sečtení všech čísel v poli
* Pomocí Thread



* Pomocí Task



* Pomocí Parallel.For

