

KONKURENTNÉ PROGRAMOVANIE

3. cvičenie: Kompozícia vláknovo bezpečných tried

Konzistentnosť 64-bitových premenných

- long a double
- volatile zabezpečí atomickosť čítania a zápisu
- synchronized zabezpečí atomickosť ľubovoľnej operácie
 - ▣ iba ak všetky operácie s premennou sú zamykané tým istým zámkom

Aktuálnosť premenných

- Ciel': hneď ako jedno vlákno nastaví novú hodnotu, všetky ostatné tovidia
- primitívne typy – volatile, AtomicXyz (AtomicInteger, AtomicLong, ...)
- referenčné typy – volatile, AtomicReference
- synchronized – na konci kritickej sekcie idú všetky hodnoty do RAM a na začiatku z RAM
- final – aktuálna hodnota do RAM hneď po nastavení

Bezpečné sprístupnenie stavu

- Ciel': nenechať stavový objekt, ktorý nie je vláknovo bezpečný, uniknúť
- vrátiť nemenné alebo efektívne nemenné objekty (také o ktorých vieme, že ich meniť nebudeme)
- sprístupniť thread-safe objekty
- sprístupnené objekty chrániť rovnakým zámkom

Uniknutie this v konšuktore

- predčasné sprístupňovanie objektu = nebezpečné sprístupňovanie
- odovzdáme referenciu na objekt, ktorý ešte nebol dotvorený

```
public class UnsafeModule {  
    ...  
    public UnsafeModule(List modules) {  
        ...  
        modules.add(this);  
        ...  
    }  
}
```

Uniknutie this v konšuktore

- Nedotvorený objekt
 - ▣ Inštančné premenné nemusia byť inicializované
 - ▣ Inštančné premenné ani nemusia byť vytvorené
- Podobne sa treba vyvarovať spúšťaniu vlákien v konšuktore
 - ▣ Vlákno môže napríklad pristupovať k svojmu tvorcovi alebo jeho stavovým premenným ešte pred dotvorením tvorca

Uniknutie this v konšuktore

□ Jedno z riešení - statický generátor

```
public class SafeModule {  
    ...  
    private SafeModule() {  
        ...  
    }  
  
    public static SafeModule newInstance(List modules) {  
        SafeModule safeModule = new SafeModule();  
        modules.add(safeModule);  
        ...  
        return safeModule;  
    }  
}
```

Stav objektu/triedy

- Spomeňme na zapúzdrenosť
 - ▣ stav objektu je uložený v jeho inštančných premenných
 - ▣ stav triedy je uložený v jej statických premenných
- Stav objektu môže mať definované obmedzenia
 - ▣ početKolies musí byť nezáporný a konečný
 - ▣ ak pohlavie==„žena“, rodnéČíslo musí mať 4. cifru 5 alebo 6
- Stav = stavové premenné + obmedzenia

Konzistentnosť stavu

- Bežná zmena stavu v metóde: Konzistentný stav sa dočasne zmení na nekonzistentný stav, ktorý sa pred skončením metódy zmení opäť na konzistentný
 - Napr. zvýším počet zobrazení stránky a zmením IP adresu „zdroja posledného kliknutia“
- Jednovláknový program
 - Ak je stav nekonzistentný, metóda ktorá predpokladá konzistentnosť zlyhá
- Viacvláknový program
 - Stavové premenné musia spĺňať obmedzenia **vždy**, keď s nimi môžu pracovať viaceré vlákna súčasne (aj uprostred metódy)

Vlastníctvo stavu

- Princíp: Ak vlastníš svoj stav, som vlákno bezpečný
- Vlastník stavu je zodpovedný za konzistentnosť stavu
 - ▣ Je dobré mať pre každú stavovú premennú určeného správcu
 - ▣ Prístup k stavu len cez bezpečné metódy správcu
 - ▣ Objekt/trieda vlastní svoj stav, iba pokiaľ nenechá uniknúť nejaký stavový objekt
 - ▣ **Stavové objekty nemusia byť thread-safe**
- Vlastnosť návrhu tried
 - ▣ Nemožné zadať v jazyku a nechať ho javu kontrolovať

Zabezpečenie vlastníctva

- Uväznenie v metóde
 - ▣ lokálna premenná metódy, ktorú nesprístupňujem inam
 - ▣ dočasný stav
- Uväznenie v inštancii bez možnosti uniknutia
 - ▣ private premenná triedy, ktorú nesprístupňujem
- Uväznenie vo vlákne
 - ▣ vlastníkom nie je objekt/trieda ale vlákno

Uväznenie premennej do vlákna

- Stav veselo sprístupňujem, nijako nestrážim, ale zabezpečím, že ho bude čítať iba jedno vlákno
- Ad-hoc riešenie môže byť zradné
 - ▣ nakódim to tak, že v druhom vlákne nebudem k premennej pristupovať
 - ▣ človek, čo po mne bude používať kód, to nemusí odhaliť a môže porušiť toto pravidlo (typicky začiatočník v Swingu)

Uväznenie premennej do vlákna

- 44,41% ľudí však chce istoty (voľby do NRSR 2012)!
 - ▣ Môžeme využiť triedu ThreadLocal, ktorá zabezpečí jednu hodnotu premennej pre každé vlákno
 - ▣ Najčastejšie použitie na vláknové singletony a vláknové globálne premenné

```
private static ThreadLocal<Connection> connectionHolder
    = new ThreadLocal<Connection>() {
        public Connection initialValue() {
            return DriverManager.getConnection(DB_URL);
        }
    };

public static Connection getConnection() {
    return connectionHolder.get();
}
```

Spustenie pri iba prvom
volaní get() z vlákna

Rozdelenie vlastníctva stavu

- Bežné rozdelenie vlastníctva
 - ▣ Správca konzistencie štruktúry
 - ▣ Správca konzistencie prvkov štruktúry
- Typické pre konkurentné kolekcie Javy
 - ▣ Kolekcia sa stará o to, aby každé vlákno pracovalo s konzistentnou štruktúrou (prechod prvkov, pridávanie prvkov, mazanie prvkov, ...)
 - ▣ O konzistenciu prvkov sa musí postarať používateľ kolekcie (ak vopchám do kolekcie prvok, nik mi ho nezmení)

Schéma práce so stavovými objektmi

- Zachovanie vlastníctva stavu (uväznenie stavu)
- Zdieľanie read-only objektov bez ďalšej synchronizácie, bezpečne prístupných
- Zdieľanie thread-safe objektov – vlákna prístupujú konkurentne bez ďalšej synchronizácie cez prístupné metódy
- Chránené objekty – prístup k nim je VŽDY cez rovnaký zámok
 - ▣ objekty v iných thread-safe štruktúrach
 - ▣ bezpečne prístupné objekty so známym zámkom

Zadanie

- Stiahnite si z GitHubu poslednú verziu:
 - ▣ <https://gitlab.science.upjs.sk/peter.gursky/kopr2019>
- Nasledovný balíček:
 - ▣ cviko3.zadanie
- Zabezpečte vláknovú bezpečnosť triedy TrafficTracker
 1. **Cez ochranu inštančnej premennej (kritické sekcie) a jej bezpečné prístupnenie**
 2. Cez nemenné body a thread-safe kolekciu
 3. Cez thread-safe body a thread-safe kolekciu

Riešenie 1

- Chrániť privátnu mapu rovnakým zámkom pri všetkých operáciách – atomickosť zmien a čítania
- Zabezpečiť si vlastníctvo privátnej mapy
 - ▣ kópia pri konštruovaní objektu
 - ▣ kópia pri sprístupňovaní stavu
- Zabezpečiť si vlastníctvo bodov
 - ▣ Nikdy nevrátiť inštancie uložené v privátnej mape

Zadanie

- Stiahnite si z GitHubu poslednú verziu:
 - ▣ <https://gitlab.science.upjs.sk/peter.gursky/kopr2019>
- Nasledovný balíček:
 - ▣ cviko3.zadanie
- Zabezpečte vláknovú bezpečnosť triedy TrafficTracker
 1. Cez ochranu inštančnej premennej (kritické sekcie) a jej bezpečné sprístupnenie
 2. **Cez nemenné body a thread-safe kolekciu**
 3. Cez thread-safe body a thread-safe kolekciu

Riešenie 2

- Bez kritických sekcií
 - ▣ Ale potom celý stav musí byť thread-safe
 - ▣ Celý stav = mapa + body
- Použiť vlákno bezpečnú mapu
 - ▣ ConcurrentMap
 - ▣ iterovanie cez ňu sa priebežne prispôsobuje zmenám
- Zabezpečiť nemenné body
 - ▣ Zmena súradníc znamená nahradenie starého bodu za nový
- Ak chcem zabezpečiť nemožnosť pridávania a odoberania bodov – sprístupňujem nemodifikovateľnú mapu
 - ▣ `Collections.unmodifiableMap()`

Zadanie

- Stiahnite si z GitHubu poslednú verziu:
 - ▣ <https://gitlab.science.upjs.sk/peter.gursky/kopr2019>
- Nasledovný balíček:
 - ▣ cviko3.zadanie
- Zabezpečte vláknovú bezpečnosť triedy TrafficTracker
 1. Cez ochranu inštančnej premennej (kritické sekcie) a jej bezpečné sprístupnenie
 2. Cez nemenné body a thread-safe kolekciu
 3. **Cez thread-safe body a thread-safe kolekciu**

Riešenie 3

- Bez kritických sekcií
 - ▣ Celý stav musí byť thread-safe = mapa + body
- Použiť vlákno bezpečnú mapu
 - ▣ ConcurrentMap
- Zabezpečiť vlákno bezpečné body
 - ▣ Celý stav bodu sa musí vrátiť naraz
 - Bud' ako nemenný bod alebo v inej štruktúre napr. dvojprvkové pole
 - ▣ Celý stav bodu sa musí nastaviť naraz
 - Kritická sekcia
- Ak nemožnosť pridávania a odoberania bodov
 - ▣ Collections.unmodifiableMap()

Dobré zásady pri chránení stavu

- Zapúzdrenie a rozumná kompozícia
- Skrývanie dát (stavové premenné private)
- Čím menej metód pristupuje k zdieľanej premennej, tým lepšie
- Ak viac vlákien pracuje s hodnotou stavu objektu/triedy, potrebujeme zabezpečiť atomickosť každej zmeny stavu

Stav objektu/triedy

- Ak chceme zachovať konzistentnosť stavu pri viacvláknovom programe máme 4 možnosti
 - ▣ nezdieľať stav medzi vláknami (čítanie ani zmena)
 - ▣ sprístupňovať konečný/nemenný stav
 - ▣ celý stav realizovať prostredníctvom thread-safe stavovej premennej
 - ▣ premenlivý stav strážiť **tým istým zámkom vždy**, keď sa k nemu pristupuje (čítanie aj zmena)