Objektovo Orientované programovanie

7. Prednáška
LS 2024/2025
Juraj Petrík

Projekt checkpoint

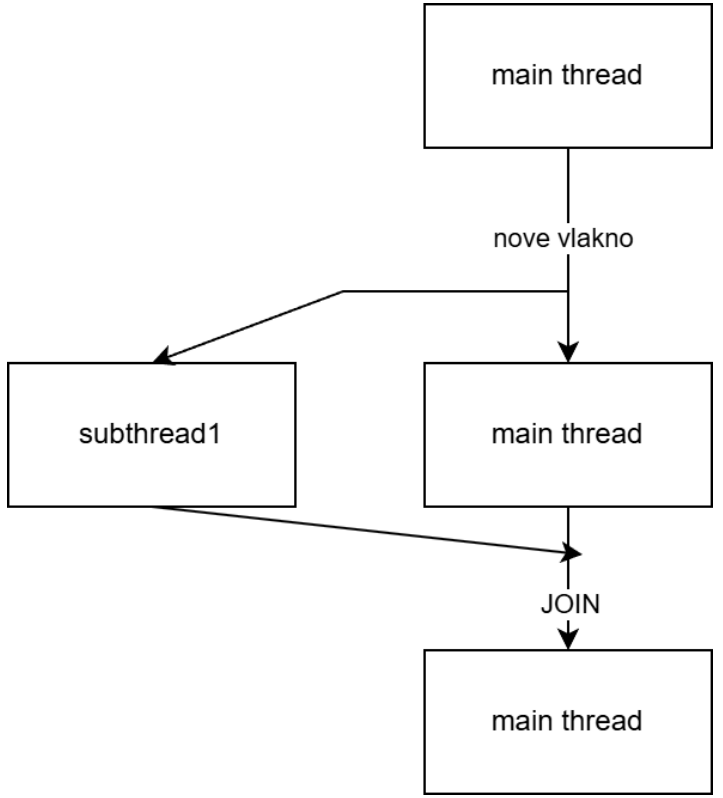
- 6.4.2025 – 20:00

Test

- 8.4.2025 – 8:00 – 11:00
- 1. - 7. prednáška a cvičenia
- 20 bodov, 8 minimum

Paralelné spracovávanie pomocou vlákien

- 2 a viac vecí robíme naraz (súbežne)
- Vlákna:
  - “Lightweight procesy”
  - **Spoločná pamäť**
- Pozor na deterministickosť



- Implementujem Runnable
- Swing napr. javax.swing.SwingUtilities.invokeLater

javax.swing.SwingUtilities.invokeLater

- Swing event handling beží na samostatnom vlákne (Event Dispatch Thread)
- Swing nie je thread-safe
- https://docs.oracle.com/javase/tutorial/uiswing/concurrency/index.html
- SwingWorker zabezpečí správnu interakciu s EDT

Streamy (prúdy)

- Vychádzajú z funkcionálneho programovania, deklaratívne
- Možné ľahké paralelné spracovanie
- Začíname zdrojom (source), potom pracujeme (intermediate operations) a na konci je terminal expression, stream je možné “použiť/prejsť” **iba raz**
- Sú lenivé (lazy)
- Nemodifikujeme pôvodné dáta
- Collections podporujú streamy (.stream(), .parallelStream())

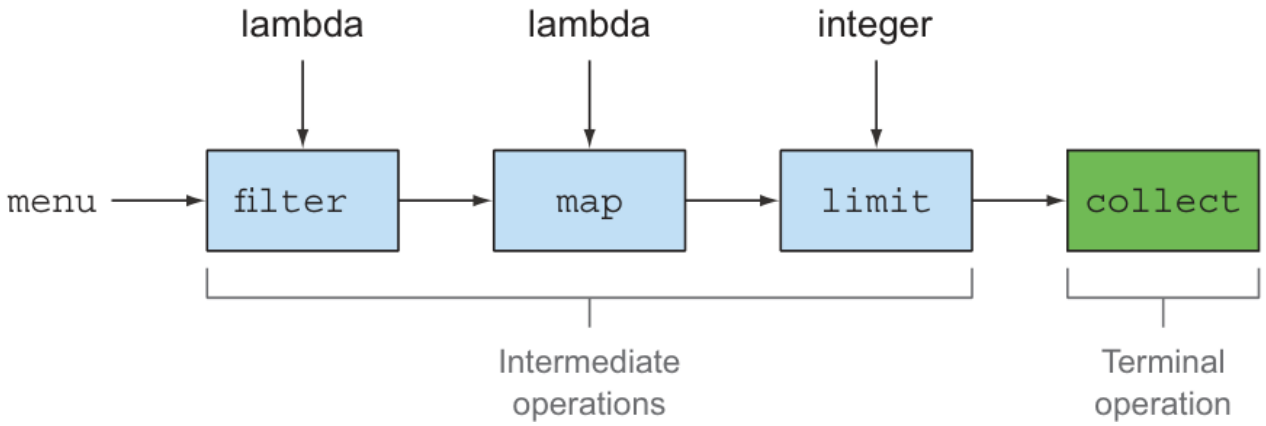
Streamy

- Sekvencia elementov - zdroj
- Data-processing operations

- Pipelining
- Internal iteration

- Computed on demand (DVD vs Netflix)

- Filter - filtrujeme
- Map - transformujeme
- Limit – zmenšujeme
- Reduce – „chceme výsledok“
- Collect – konverzia streamu na inú formu



- List<Integer> steveList2 = steveList.stream().filter((steve) -> steve.getName().equals("Steve")).sorted(comparing(Steve::getName)).map(Steve::getHunger).toList();

- Declarative
- Composable
- Parallelizable

- distinct()
- .forEach(System.out::println)
- .takeWhile()/.dropWhile()
- .skip()
- .limit()
- .flatMap()
- .allMatch()/noneMatch()
- .findAny() (Optional<T>)

Streams (reduce)

- .reduce(inicialna_hodnota, Operator)
- int product = numbers.stream().reduce(1, (a, b) -> a * b)

- Optional! (aj pri flatten napr.) - .isPresent(), .isEmpty()

Ukladanie objektov

- Uloženie stavu objektu
- Pomocou:
  - Serializácia:�� sr minecraft.player.SteveJHu��h Z godModeI hungerI maxHungerx
  - Plain Text: napr (csv like).: Juraj,1,1,1..

Serializácia

- Ukladá sa celý objektový graf, automaticky
- All or nothing
- Implements Serializable
- Pokiaľ nechcem/neviem serializovať - transient

Serializácia a **IO** streamy

- 1. Vytvoriť FileOutputStream – kam zapisujeme
- 2. Vytvoriť ObjectOutputStream – čo zapisujeme
- 3. Zapísať object .writeObject()
- 4. Zavrieť ObjectOutputStream – automaticky sa zatvorí aj FileOutputStream

Deserializácia

- **Nevykonáva** sa konštruktor
- 1. Vytvoriť FileInputStream – odkiaľ čítame
- 2. Vytvoriť ObjectInputStream – kam čítame
- 3. Prečítať objekt .readObject()
- 4. Zavrieť ObjectInputStream – automaticky sa zatvorí aj FileInputStream

Deserializácia

- 1. Objekt sa prečíta zo streamu
- 2. JVM zistí typ triedy (uložené)
- 3. JVM nájde a načíta príslušnú triedu
- 4. Vytvorí objekt na heape, bez pustenia konštruktora
- 5. Ak je niekde v grafe nedesiarizovateľná trieda, tak sa vykonajú konštruktor/y, pozor na reťazovú reakciu (super)
- 6. priradia sa atribúty triedy, ak boli transient, tak sú null, 0 a pod.

Čo keď sa trieda zmení? Problémy

- Zmazanie atribútu
- Zmena typov
- Zmena z non-transient na transient
- Posúvanie v hierarchii dedenia
- Odstránenie Serializable
- Zmena atribútu na statický atribút

Čo keď sa trieda zmení? Čo je (zvyčajne) okej

- Pridanie nového atribútu
- Pridanie novej triedy do hierarchie dedenia
- Odstránenie tried z hierarchie dedenia
- Zmena modifikátorov prístupu
- Transient na non-transient

Serializácia - serialVersionUID

- serialver Steve
- static final long serialVersionUID = -5849794470654667210L;

- Toto však v zásade nerieši nič za programátora, programátor si rieši všetko ďalšie sám

!Quiz time