Java konkurentno programiranje

Uvodni koncepti: konkurentnost, niti, procesi...

Konkurentnost

- Dve teme:
  1. Procesi
     - Samostalni vidovi izvršavanja: privatna memorija i drugi resursi
  2. **Niti**
     - Žive u okviru procesa
     - Najmanje jedna po procesu
     - Dele među sobom memoriju procesa

Procesi

- Single-core system:
  - U datom momentu se može izvršavati samo jedan proces
  - Privid izvršavanja više se postiže timesharing-om
  - Jedan program se može izvršavati kroz više procesa
- Multi-core system:
  - Više procesa istovremeno (pravi paralelizam)

Svojstva konkurentnosti

- Zašto da?
  - Dostupnost
  - Minimizacija vremena čekanja na odgovor, maksimizacija protoka
  - Modelovanje
  - Simuliranje autonomnih objekata, animacije
  - Paralelizam
  - Iskorišćavanje multiprocesorskog okruženja, više I/O istovremeno
  - Zaštita
  - Izolovanje aktivnosti u okviru niti
- Zašto ne?
  - Kompleksnost
  - Sigurnosni problemi, kompozicija aktivnosti
  - Preopterećenje
  - Visoko korišćenje resursa

Tipične primene

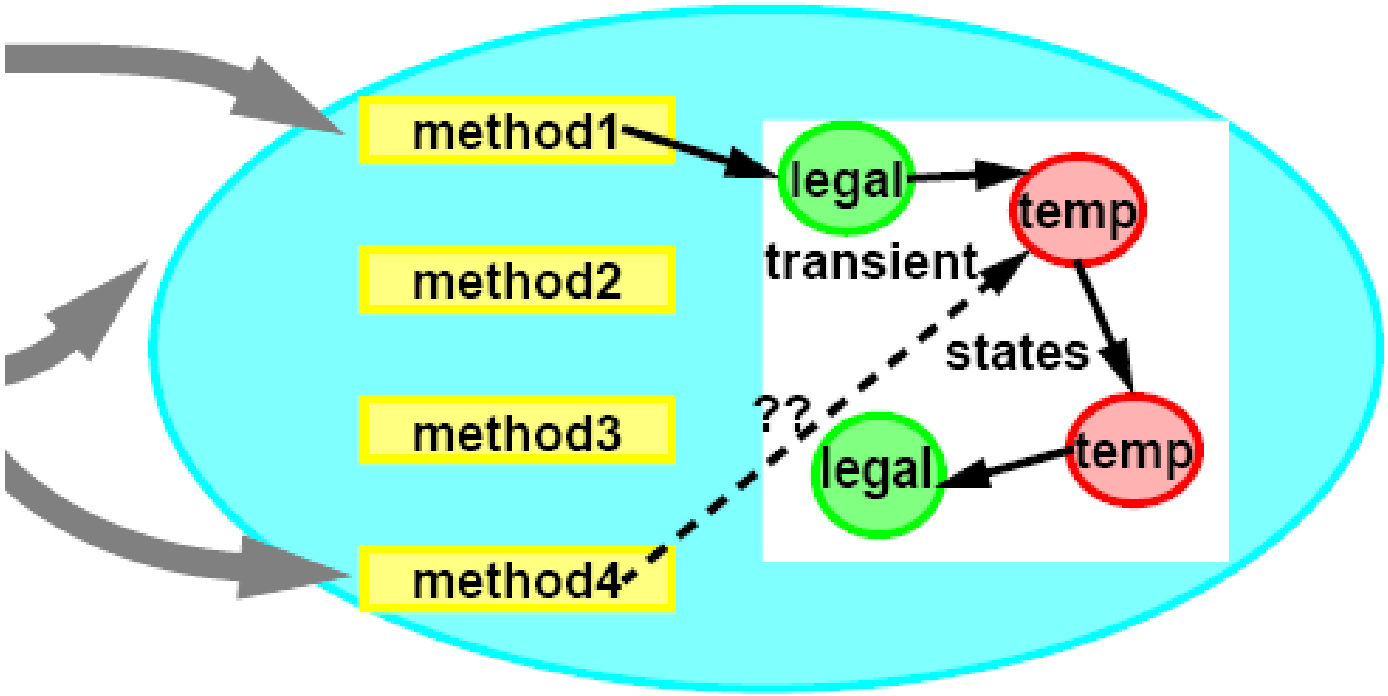
- I/O zavisni pristupi
  - Konkurentni pristu web stranicama, bazi podataka, soketima..
- GUI
  - Konkurentno hvatanje događaja, osvežavanje ekranskih kontrola..
- Izvršavanje stranog koda
  - Konkurentno izvršavanje apleta, JavaBeans...
- Demonski procesi (na Serverima obično)
  - Konkurentno opsluživanje više klijentskih zahteva
- Simulacije
  - Konkurentno simuliranje više realnih objekata

Konkurentno programiranje

- Konkurentnost je **konceptualno** svojstvo softvera
- Konkurentni programi mogu imati sledeća svojstva:
  - **Operabilnost nad više CPU**: klasteri, arhitekture spec. namene
  - **Paralenost**: mapiranje softvera na više CPU, radi poboljšanja performansi
  - **Deljeni pristup resursima**: objekti, memorija, fajl deskriptori, soketi..
  - **Distribuiranost:** konkurentni programi koji NE dele resurse

Sigurni objekti

- Izvršavaju akcije **samo** kada su u konzistentnom stanju



Primeri nekonzistentnih stanja

- Istovremeno iscrtavanje i pomeranje figure na ekranu
- Povlačenje sredstava sa bankovnog računa u toku transfera novca
- Čitanje sa memorijske lokacije dok je u toku pisanje po istoj

Efikasne aktivnosti

- Svaka aktivnost bi trebalo da vrši napredak ka završenju i to što je pre moguće
  - Svaki pozvani metod bi trebalo u nekom momentu da se izvrši, a ideja je napraviti da se to desi što je pre moguće (efiksanost)

- Aktivnost može da se ne završi (nije efikasna) ukoliko:
  - Objekat ne može da primi poruku
  - Metod blokira čekanje na događaj, poruku ili uslov
  - Nedovoljni ili nepravedno raspodeljeni resursi
  - Različite greške i padovi sistema...

Problem dizajna konkurentnih programa

- Dva ekstremna pristupa:
  1. **Sigurnost na prvom mestu**
     - Pobrinuti se za sigurnost prvo, pa onda pokušati sa optimizacijom efikasnosti
     - Karakteristično za top-down OO dizajn
     - Može rezultirati sporim izvršavanjem i mrtvim stanjima (deadlocks)
  2. **Efikasnost na prvom mestu**
     - Dizajnirati održiv sistem, a onda pokušati sa poboljšanjem sigurnosti kroz koncepte zaključavanja i čuvanja resursa
     - Karakteristično za višenitno (multithreaded) sistemsko programiranje
     - Može rezultirati „bagovitim“ kodom

Garantovana sigurnost

„Nikad se ne izvrši nešto loše“

- Manifest na niskom nivou
  - Bitovi se uvek pravilno interpretiraju
  - Nema memorijskih konflikta tipa read/write ili write/write
- Manifest na visokom nivou
  - Objekti su dostupni samo kada su u konzistentnom stanju
  - Invarijantnost stanja

Garantovana efikasnost

„Uvek se bar nešto izvrši“ – uslov napretka

- Dostupnost
  - Izbegavanje bespotrebnog blokiranja
- Napredak
  - Izbegavanje zadržavanja resursa među više aktivnosti
  - Izbegavanje deadlock-a
  - Izbegavanje nepravednog organizovanja rasporeda
- Zaštita
  - Izbegavanje sukoba sa drugim programima
  - Sprečavanje „denial of service“ napada
  - Sprečavanje zaustavljanja rada pod uticajem stranih „agenata“

Particle applet

- t_particle\ParticleApplet.java

Zadatak

- Napraviti celularni automat zasnovan na tzv. Igri Života (eng. Game of Life)
- Applet dimenzije 500x500 je podeljen na matricu od 100x100 kvadrata svaki dimenzije od 5x5 piksela
- Kvadrat predstavlja pojedinačnu ćeliju, pri čemu ona može biti crna ili žuta (crne su mrtve, žute su žive)
- Inicijalno stanje je slučajno, podjednake šanse da je neka ćelija živa ili mrtva

Zadatak nastavak

- Potom svaka ćelija prelazi iz stanja u stanje na osnovu sledećih pravila:

1. Svaka živa ćelija sa manje od dva živa suseda umire od samoće
2. Svaka živa ćelija sa dva ili tri živa suseda preživjava
3. Svaka živa ćelija sa više od tri živa suseda umire zbog prenaseljenosti
4. Svaka mrtva ćelija sa tačno tri živa suseda oživljava

- Ćelija primenuje pravilo svaki put kada dobije mogućnost izvršavanja (kao move metoda u particles)