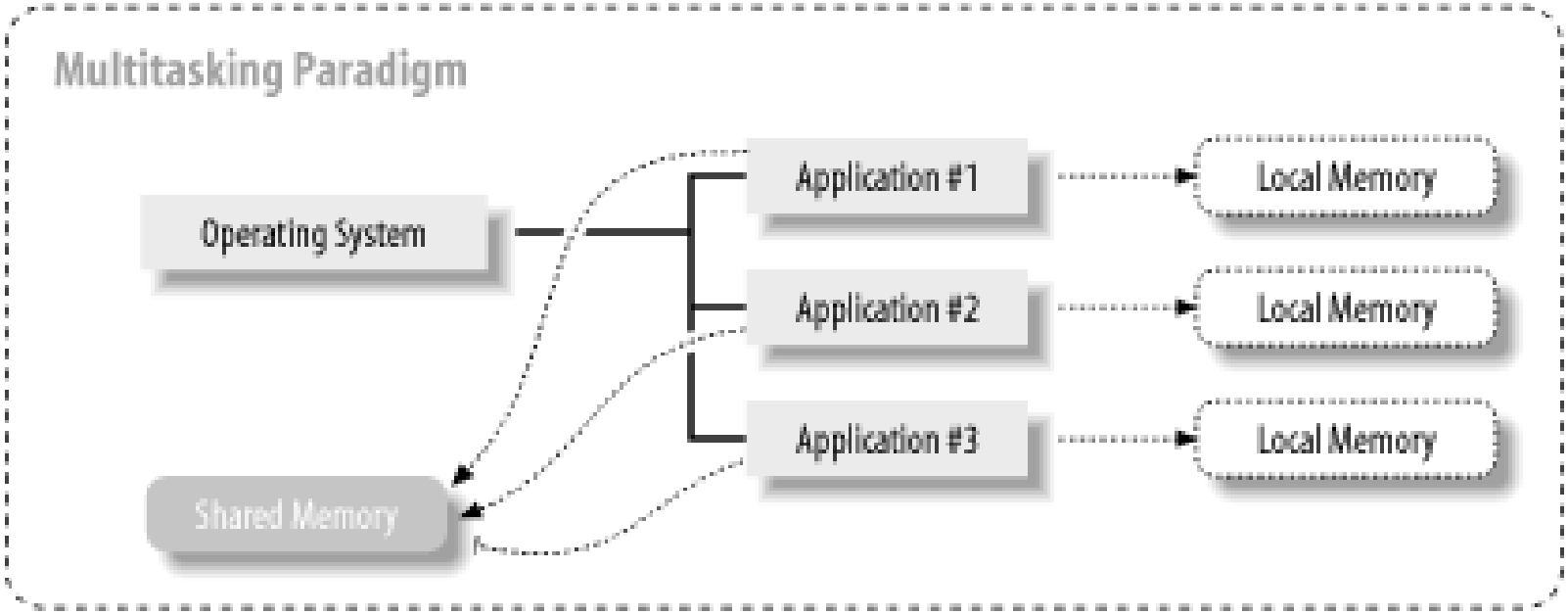
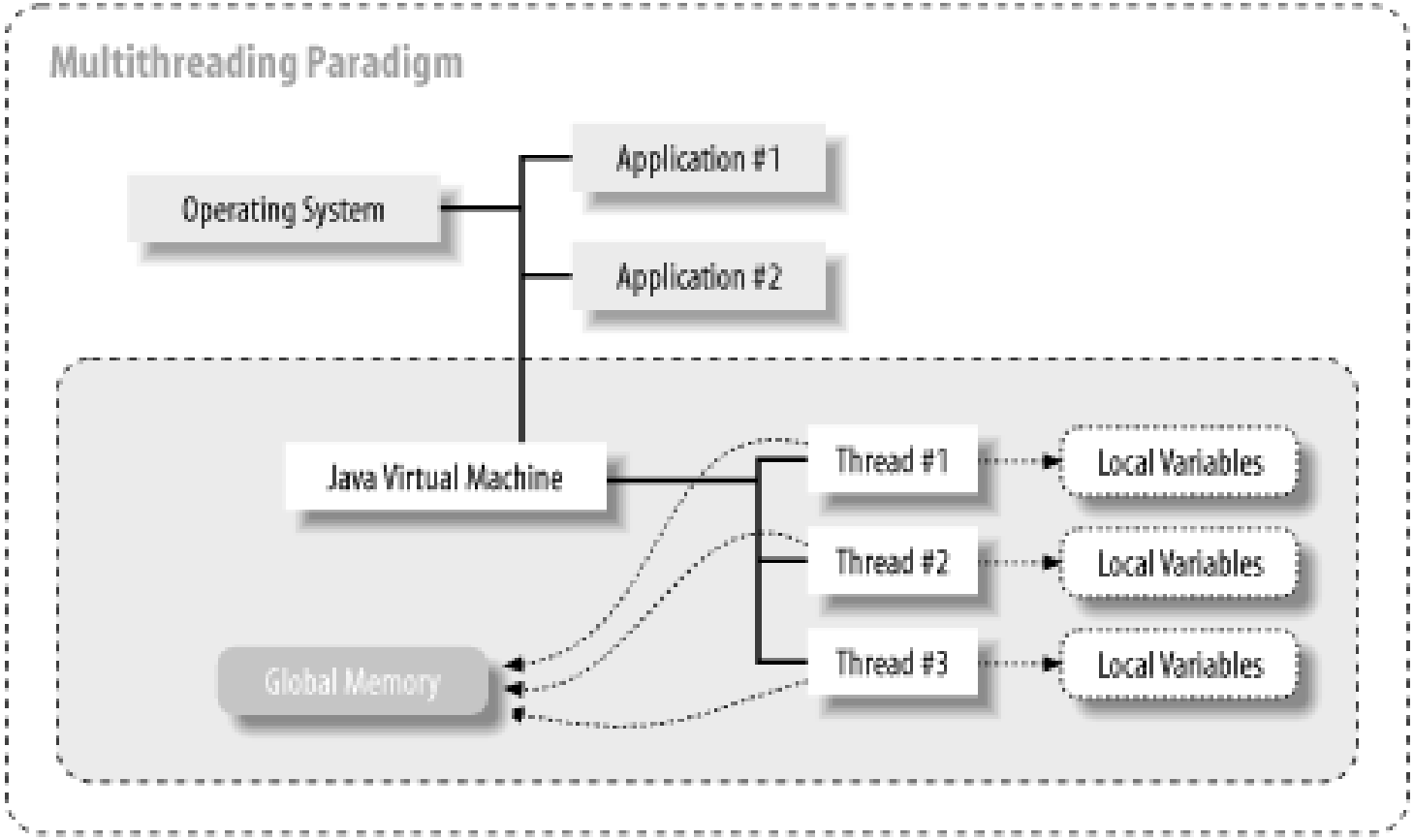
Java konkurentno programiranje

Životni ciklus niti
i problemi sinhronizacije resursa

Multitasking

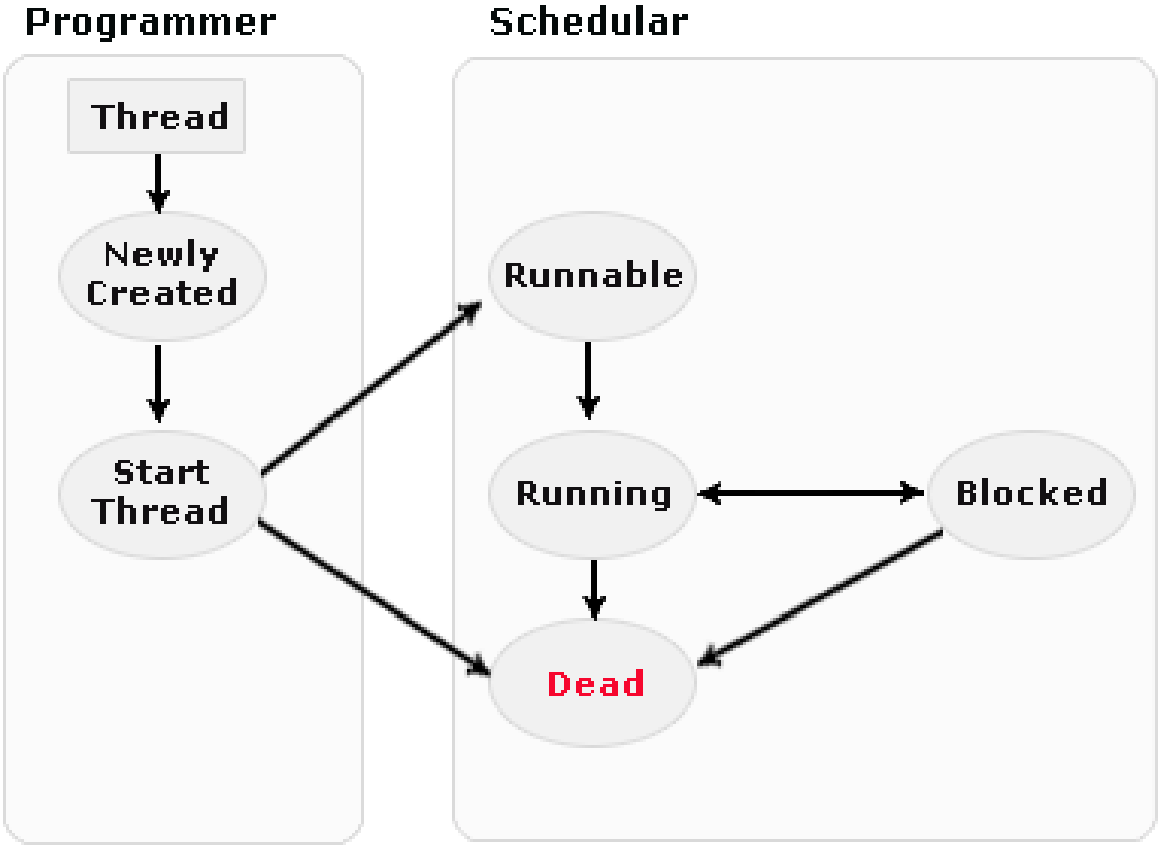


Multithreading

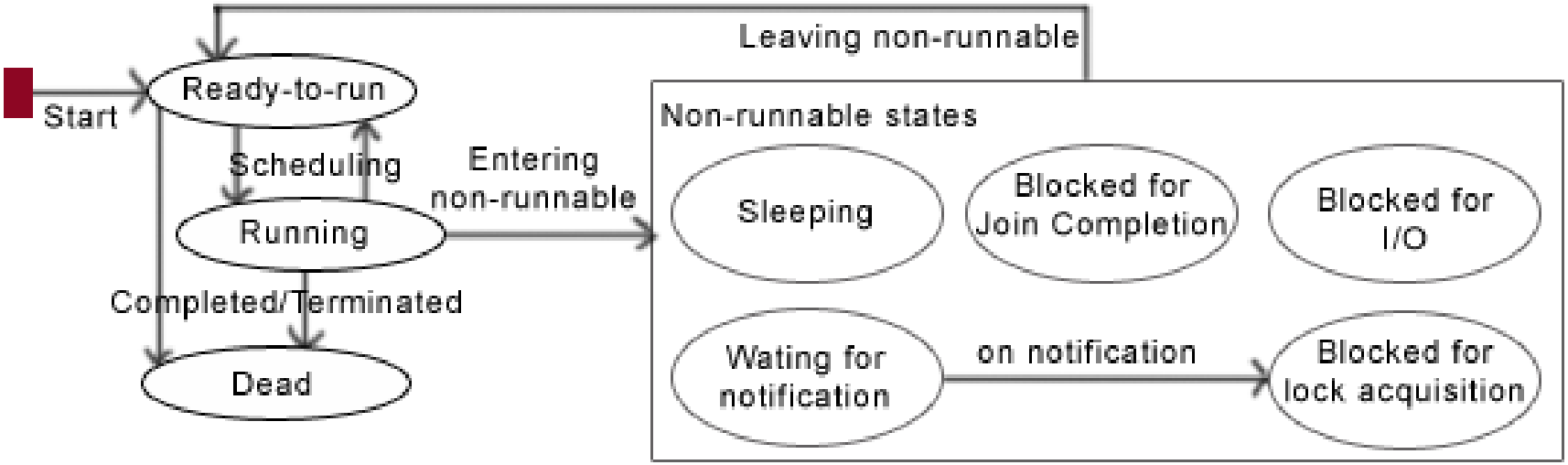


Životni ciklus niti

- http://www.roseindia.net/java/thread/life-cycle-of-threads.shtml



Životni ciklus niti - detaljnije



Java podrška za konkurentnost

- **Nit (thread) klasa predstavlja stanje jedne nezavisne aktivnosti i ima sledeće osobine:**
  - **Metode start, sleep ...**
  - **Veoma slabu garanciju kontrole i redosleda aktivnosti**
  - **Svaki Thread je član tzv. ThreadGroup-e koja se koristi za kontrolu pristupa i čuvanje podataka**
  - **Kod koji se izvršava u Thread-u je definisan metodom run()**
- **Synchronized metodi i blokovi koda kontrolišu atomičnost korišćenjem katanaca (locks)**
  - **Java postavlja automatske katance na tipove: byte, char, short, int, float i Object reference, ali ne i na double i long tip**
  - **volatile ključna reč kontroliše atomičnost jedne promenljive**
  - **Monitor metodi u klasi Object: wait(), notify(), notifyAll()**

Klasa Thread

- Konstruktor
  - Thread(Runnable r)
  - Pokretanjem niti počinje da se izvršava sadržaj metoda run()
- Ostali nasleđeni metodi
  - start() - aktivira run()
  - isAlive() - vraća true ako je nit aktivirana ali ne i zaustavljena
  - join() - čeka na završetak
  - interrupt() - izlazi iz wait, sleep ili join metoda
  - isInterrupted() - vraća informaciju da li je nit prekinuta
  - getPriority() - vraća prioritet niti
  - setPriority(int) - podešava prioritet niti

Preostali statički metodi

- currentThread() - vraća referencu na nit koja je lokalna sa pozicije poziva
- sleep(ms) - susenduje nit na najmanje ms milisekundi (takođe se odnosi na aktuelnu nit)
- interrupted() - vraća i čisti status o prekinutosti

Dizajn konkurentnih objekata

- Obrasci za reprezentaciju i menadžment sigurnih stanja objekata:
  - Nepromenjivost (eng. Immutability)
    - Ne dopuštati promene
  - Zaključavanje (eng. Locking)
    - Garantovanje ekskluzivnog pristupa
  - Zavisnost stanja
    - Šta raditi kad ne može da se uradi ništa
  - Ostalo...

Nepromenjivost

- Nepromenjivi objekti nikad ne menjaju stanje
- Aktivnosti nad nepromenjivim objektima su uvek sigurne i efikasne

```
class ImmutableAdder {
    private final int offset_; // blank final
    ImmutableAdder(int x) { offset_ = x; }
    int add(int i) { return i + offset_; }
}
```

- Primeri nepromenjivih tipova u javi: String, Integer, Color

Primene nepromenjivosti

- **Nepromenjive reference ka promenjivim objektima**

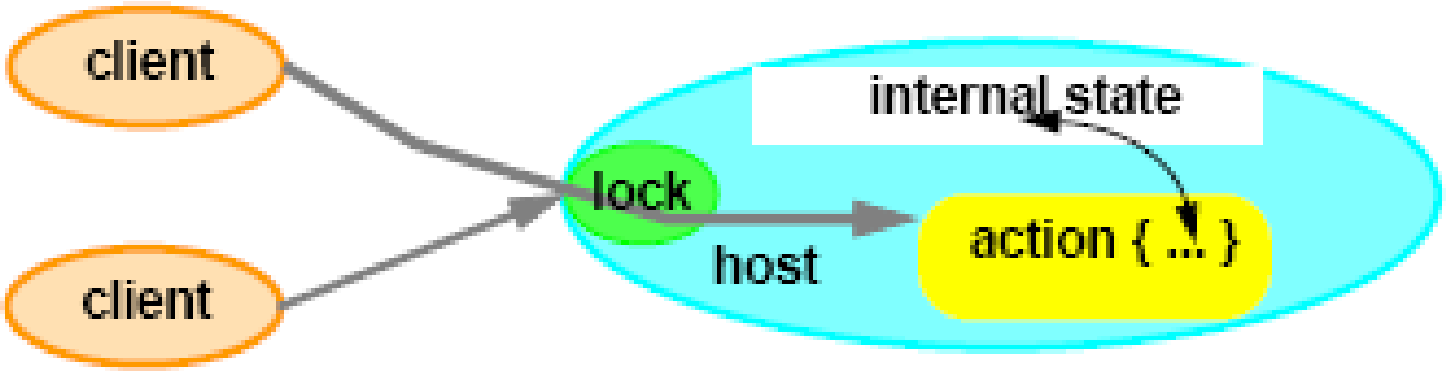
```
class Relay {
        private final Server delegate;
        Relay(Server s) { delegate = s; }
        void serve() { delegate.serve(); }
}
```

- **Parcijalna nepromenjivost**

```
class FixedList { // cells with fixed successors
        private final FixedList next; // immutable
        FixedList(FixedList nxt) { next = nxt; }
        FixedList successor() { return next; }
        private Object elem = null; // mutable

        synchronized Object get() { return elem; }
        synchronized void set(Object x) { elem = x; }
}
```

Zaključavanje (eng. Locking)



Zaključavanje

- Sprečava konflikte sa memorijskim lokacijama, tzv. Problem sihronizacije resursa
- Može se koristiti za garantovanje atomičnosti (princip sve ili ništa) metoda
- Može da izazove deadlock (mrtvo zaključavanje)

Primer

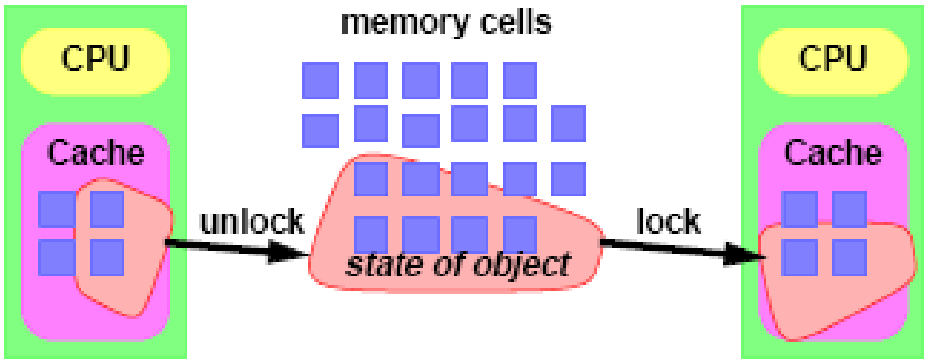
```
class Location {
    private double x_, y_;
    Location(double x, double y) { x_ = x; y_ = y; }
    synchronized double x() { return x_; }
    double y() {
        synchronized (this) {
        return y_;
        }
    }
    synchronized void moveBy(double dx, double dy) {
        x_ += dx;
        y_ += dy;
    }
}
```

Java katanci

- Svaki Java Object poseduje jedan katanac (tzv. Implicitni katanac)
  - Njime se manipuliše preko synchronized ključne reči
  - Class objekti sadrže katanac za zaštitu statičkih promenljivih i metoda
  - Skalari kao što su int, short nisu objekti, tako da njih ne možemo zaključavati
- Java katanci su više puta iskoristivi (reentrant)

Katanci i keširanje

- Zaključavanje generiše poruke između niti i memorije
- Uzimanje katanca inicira čitanje iz memorije u keš memoriju niti
- Otpuštanje katanca inicira pisanje keširanog sadržaja nazad u memoriju



Primer sa bankom

- U priloženom materijalu....

Zadatak

- Postoji veliko skladište slatkiša koje ima 100 hiljada slatkiša (npr. čokoladnih bananica) inicijalno.
- Pored velikog postoje 4 prodavnice slatkiša u koje može da se smesti najviše 1000 slatkiša. Inicijalno su prazne.
- Postoje 100 ljudi i svaki od njih na svakih pola sekunde oseti potrebu za čokoladnom bananicom i odlazi do neke od nasumično odabranih prodavnica.

Zadatak (nastavak)

- Potom u prodavnici, ako je dostupno, uzme nasumično od 1 do 7 slatkiša. Brzo nakon toga ih pojedu, ali uvek zapamte koliko su dosad uzeli komada.
- Prodavnice se snabdevaju periodično iz skladišta svakih 5 sekundi. One uvek uzmu toliko slatkiša da se dopune do maksimalnog kapaciteta.

Zadatak (nastavak)

- Modelovati sistem i pobrinuti se da u svakom momentu ukupna količina slatkiša u skladištu, prodavnicama i broj pojedenih komada bude 100 hiljada.
- Ispisavati promene stanja svake sekunde, koristiti odvojenu nit za ovo.
- Varirati uslove sistema: smanjenje i povećavanje broja ljudi, vremena njihovog dolaženja u prodavnicu i slično.