1. Zadatak

Implementirati aplikaciju koja upravlja radom magacina. Tri viljuškara se koristi za utovar robe u šleper, dok ukupna težina robe ne bude 10 000 kg. Svaki viljuškar može da ponese 500 kg. Sinhronizaciju izvesti korišćenjem **synchronized bloka.**

1. Napraviti nit za svaki viljuskar: *ViljuskarNit* nasleđuje klasu *Thread* i ima polje String registracija
2. Za synchronized blok se ne mora koristiti posebna klasa već je dovoljno koristiti običnu statičku promenljivu
3. Viljuškaru treba 1 sekunda da prebaci robu iz magacina u šleper (*sleep*)
4. Nakon što viljuškar utovari robu, ispisati na ekran: “Viljuskar <registracija> je utovario robu. Trenutna tezina je sad <trenutnaTezina>.
5. Instancirati 3 niti za viljuškare sa registracijama “NS111”, “NS222” i “NS333” (ili po želji).

2. Zadatak

Implementirati aplikaciju za upravljanje transakcijama banke. Banka u toku dana rukuje velikim brojem transakcija koje menjaju stanja računa. Na kraju dana, kada se sve transakcije završe, ispisuje stanje svih računa. Postoji 4 računa sa po 3 transakcije. Sinhronizaciju izvesti korišćenjem **synchronized metode.**

1. Račun je predstavljen klasom Racun i ima polja String imeVlasnika i double stanjeRacuna. Pristup ovim promenljivima treba da bude sinhronizovan (gde ovo staviti?)
2. Transakcija je nit i poseduje dve promenljive: referencu na račun (Racun racun) I promenu računa (double promenaRacuna). Promena računa može biti i pozitivan i negativan broj
3. Transakciji treba 1.5 sekundi da izračuna novu vrednost (sleep). Nakon toga menja stanje računa
4. Instancirati 4 računa sa po 3 transakcije. Računima i transakcijama staviti proizvoljne vrednosti
5. Nakon što se sve transakcije završe, ispisati stanja svih računa: “Racun korisnika <imeVlasnika> je <stanjeRacuna>”

3. Zadatak

Implementirati aplikaciju koja simulira pekaru. Pekar pravi kolače, a deca ih jedu. Pekar konstantno svakih 200ms napravi jedan kolač (daemon nit). Deca svakih 500ms jedu kolače i naješće se kada ih pojedu 5. Četvoro dece jede kolače. Na početku je na tacni 5 kolača.

1. Pekar pravi kolače u svojoj niti (*PekarNit).* U konstruktoru pozvati *setDaemon(true).* Svakih

200ms povećava broj kolača za 1 i ispisuje na ekran “Pekar je napravio kolac! Sada ih ima <brojKolaca>.”

2. Dete u svojoj niti (*DeteNit)* i ima polje ime tipa String *i* svakih 500ms pojede kolač tako što smanji broj kolača za 1. Kada pojede kolač, ispisati na ekran “<imeDeteta> jede kolač! Ostalo ih je <brojKolaca>.”
3. Dete se najede kada pojede 5 kolača i izvršavanje niti prestaje
4. Napraviti statičku volatile promenljivu tipa int brojKolaca i inicijalizovati na 5
5. Instancirati 4 niti za decu, 1 za pekara i sve ih pokrenuti

4. Zadatak

Implementirati aplikaciju koja simulira dom zdravlja. Četiri lekara radi u četiri ordinacije i za pregledanje jednog pacijenta im je potrebno 1500ms. Dok pregleda pacijenta, lekar je zauzet (*setZauzet(true)*) i ne može primati pacijente. Nakon što pregleda pacijenta, lekar postaje dostupan (*setZauzet(false)*). Novi pacijenti stižu svakih 500ms i odlaze kod prvog slobodnog lekara. Dom zdravlja se zatvara nakon što lekari pregledaju 24 pacijenta.

1. Implementirati klasu *Lekar*, sa poljima **boolean zauzet** i **String ime** i klasu *LekarNit* koja nasleđuje klasu *Thread*, sa poljem **Lekar lekar.** Na početku pregledanja pacijenta u niti ispisati na ekran “Lekar <ime> prima pacijenta.” i sinhronizovano postaviti polje lekara **zauzet** na true. Na kraju pregledanja nit ispisuje “Lekar <ime> završava pregled pacijenta” i sinhronizovano postaviti polje lekara **zauzet** na false. Početak pregledanja se izvršava u metodi *pocniPregled(),* gde se kreira i pokreće nova nit (*LekarNit*). U konstruktoru proslediti lekara iz kojeg se poziva metoda (kako se dobija trenutni objekat?).
2. U *main metodi* instancirati i pokrenuti 4 niti (*Lekar)* i smestiti ih u vektor (*Vector*). Ispisati na ekran “Dom zdravlja počinje sa radom”.
3. U for petlji na svakih 500ms (*sleep*) u vektoru lekara naći prvog koji je slobodan, tj. **zauzet == false** I pozvati nad njim metodu *pocniPregled()* (napomena: vektor se koristi isto kao lista).
4. Na kraju izvršavanja programa ispisati na ekran “Dom zdravlja završava sa radom.”