Objektno-orijentisani dizajn softvera

Vježbe 9

Analizirati problem sinhronizacije niti kod višenitnog programiranja. Za demonstricju koristiti jednostavni primjer koji simulira transakcije na računima u banci. Za potrebe primjera implementirati klasu Račun sa promjenljivom stanje. Takođe implementirati klasu Banka, koja kao podatke članove ima 2 računa i metodu prenesi koja novac sa jednog računa prebacuje na drugi račun. Takođe implementirati klasu Transakcija koja implementira interfejs Runnable. Ova klasa kao podatak član ima objekat klase Banka, a u njoj je potrebno izvršiti proizvoljan broj transakcija sa jednog računa banke na drugi. Kreirane metode i funkcionalnost testirati u klasi Testiraj.

```
public class Racun {
   private int stanje;

   public Racun(int stanje) {
      this.stanje = stanje;
   }

   public void podigni(int iznos) {
      this.stanje -= iznos;
   }

   public void uplati(int iznos) {
      this.stanje += iznos;
   }

   public int getStanje() {
      return this.stanje;
   }
}

public class Banka {
   private Racun racun1, racun2;

   public Banka(Racun r1, Racun r2) {
      racun1 = r1;
      racun2 = r2;
   }

   public synchronized void prenesi(int iznos, int smjer) {
      if(smjer == 1) {
         racun1.podigni(iznos);
         racun2.uplati(iznos);
      }else {
         racun2.podigni(iznos);
         racun1.uplati(iznos);
      }
      String imeNiti = Thread.currentThread().getName();
      System.out.println(imeNiti + "  Racun1 => " + racun1.getStanje() +
                         "  Racun2 => " + racun2.getStanje() + "  Ukupno => " + u
                           kupnoStanje());
      notifyAll();
   }

   public synchronized int ukupnoStanje() {
      int ukupno = racun1.getStanje() + racun2.getStanje();
      return ukupno;
   }
}
```

```java
import java.util.Random;

public class Transakcija implements Runnable {

   private Banka banka;

   public Transakcija(Banka b) {
      this.banka = b;
   }

   public void run() {
      for(int i = 0; i < 20; i++) {
         Random r = new Random();
         int slucajniRacun = r.nextInt(2);
         int slucajnaVrijednost = r.nextInt(50);
         banka.prenesi(slucajnaVrijednost, slucajniRacun);
         try{
            Thread.sleep(10);
         }
         catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
   }
}

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Testiraj {
   public static void main(String[] args) {

      Racun r1 = new Racun(1000);
      Racun r2 = new Racun(2000);

      Banka b = new Banka(r1, r2);
      ExecutorService exec = Executors.newCachedThreadPool();

      for(int i = 0; i < 10; i++) {
         Thread t = new Thread(new Transakcija(b));
         exec.execute(t);
      }
      exec.shutdown();
   }
}
```