ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за ИР Београд 23.04.2016.

Колоквијум из
Објектно оријентисаног програмирања II

1) *(30 поена)* Одговорити концизно (по једна или две реченице) и прецизно на следећа питања:

а) Како треба пројектовати класу која омогућава безусловно клонирање својих објеката?

б) Да ли референца на језику Јава може да буде типа: (1) примитивног податка, (2) класе, (3) интерфејса, (4) типа набрајања? Одговорити навођењем броја и "да"/"не".

в) Да ли је дозвољен приступ локалним променљивама и параметрима окружујућег метода из локалне унутрашње класе?

2) *(укупно 70 поена)* Написати на језику *Java* следећи пакет типова (грешке пријављивати изузецима опремљеним текстовима порука):

- *(10 поена)* ***Датум*** има годину, месец и дан који могу да се дохвате. Може да се одреди да ли је један датум новији од другог. Текстуални опис је *дан***/***месец***/***година.*
- *(10 поена)* ***Рачун*** има јединствени аутоматски генерисани број, тренутно стање и дозвољени минус, који могу да се дохвате. Могуће је променити стање рачуна за задати износ. Грешка је ако се прекорачи дозвољени минус.
- *(10 поена)* ***Захтев за трансфер*** има јединствени аутоматски генерисан број, рачуне примаоца и уплатиоца, целобројан износ и датум подношења захтева. Није потребно проверавати исправност унетих података. Износ и датум подношења могу да се дохвате и промене. Може да се дохвати ознака врсте захтева (ниска), да се захтев изврши и да му се направи копија. Приликом прављења копије захтева не копирају се рачуни. Може да се увећа стање рачуна примаоца за износ захтева и да се умањи стање уплатиоца за износ захтева. Извршавање захтева реализује се увећањем стања примаоца и умањењем стања уплатиоца. Резултат извршавања захтева представља индикатор успеха трансфера. Текстуални опис је облика *врста* **(***ид***)** *датум*.
- *(10 поена)* ***Уплатница*** је захтев за трансфер. Приликом извршавања уплатнице, уплатиоцу се наплаћује провизија у износу од 1% уплате. Ознака врсте је **`Uplata`**. Текстуални опис уплатнице је *врста* **(***ид***)** *датум* **[***износ***:***провизија***]**.
- *(10 поена)* ***Кредитни захтев*** је захтев за трансфер. Ознака врсте је **`Kredit`**. Приликом извршавања кредитног захтева, примаоцу се наплаћује провизија у износу од 5% кредита. Текстуални опис исплате је *врста* **(***ид***)** *датум* **[***износ***]**.
- *(10 поена)* ***Банка*** има рачун и збирку захтева за трансфер, уређену по неопадајућем поретку датума подношења, која је подразумевано празна. Збирка је неограниченог капацитета и могу јој се додавати захтеви за трансфер појединачно, један по један. Захтеви могу да се узимају појединачно са почетка збирке (уклањају се из збирке; грешка је ако је збирка празна). Банка може да узме први захтев из збирке и да га изврши. Текстуални опис банке је у формату *стање_рачуна* ***(****захтев***,***<нова линија>***захтев***,***<нова линија>*** ... ,***<нова линија>***захтев****)**.

(10 поена) Написати на језику *Java* **програм** који направи рачун клијента и банке, затим направи банку и дода једну уплатницу, један кредитни захтев, и једну копију уплатнице са новијим датумом. Банка се након тога исписује на стандардни излаз. Затим банка изврши све захтеве за трансфер и опет се исписује на стандардни излаз. Користити константне параметре (не треба ништа учитавати).

НАПОМЕНЕ: **а)** Колоквијум траје **120** минута.

б) Рад се предаје искључиво у вежбанци за испите (-5 поена за неадекватну вежбанку). Није дозвољено имати поред себе друге листове папира, нити уз себе имати мобилни телефон, без обзира да ли је укључен или искључен.

в) Водити рачуна о уредности. Нечитки делови текста ће бити третирани као непостојећи. Решења задатака навести по горњем редоследу (-1 поен за лош редослед). Препоручује се рад обичном графитном оловком.

г) Резултати колоквијума биће објављени на *Web*-у на адреси: `http://rti.etf.bg.ac.rs/rti/ir2oo2/index.html`

```
// Datum.java
package banka;
public class Datum {
   private int g, m, d;
   public Datum(int d, int m, int g)
   { this.g = g; this.m = m; this.d = d;}
   public boolean noviji(Datum datum){
      if (g > datum.g || (g == datum.g && m >
         datum.m) || (g == datum.g && m == datum.m &&
         d > datum.d)) return true;
      return false; }
   public String toString()
   { return d+"/"+m+"/"+g; }
   public int d(){return d;}
   public int m(){return m;}
   public int g(){return g; }
}
```

```
// Racun.java
package banka;
public class Racun {
   private static int ID = 0;
   private int id = ++ID;
   private int stanje, minus;
   public Racun(int minus, int stanje)
   { this.minus = minus; this.stanje = stanje; }
   public int id(){ return id; }
   public int stanje(){ return stanje; }
   public int minus(){ return minus; }
   public void promeniStanje(int iznos) throws GMinus{
      if (stanje + iznos < minus) throw new GMinus();
      stanje += iznos; }
}
```

```
// ZahtevZaTransfer.java
package banka;
public abstract class ZahtevZaTransfer
implements Cloneable{
   private static int ID = 0;
   private int id = ++ID;
   protected int iznos;
   protected Racun primalac, uplatilac;
   protected Datum datumPodn;
   public ZahtevZaTransfer(Racun p,
      Racun u, int i, Datum d) {
      primalac = p; uplatilac = u;
      iznos = i; datumPodn = d; }
   public float iznos(){ return iznos; }
   public void iznos(int iznos){ this.iznos = iznos; }
   public Datum predaja(){ return datumPodn; }
   public void predaja(Datum d){ datumPodn = d; }
   public abstract String vrsta();
   public String toString(){
      return vrsta() + "(" + id + ")" + datumPodn; }
   public ZahtevZaTransfer clone(){
      ZahtevZaTransfer kopija = null;
      try {
         kopija = (ZahtevZaTransfer) super.clone();
         kopija.id = ++ID;
         kopija.datumPodn = new Datum(datumPodn.g(),
                        datumPodn.m(), datumPodn.d());
      } catch (CloneNotSupportedException e)
      { e.printStackTrace(); }
      return kopija; }
   public boolean izvrsi(){
      try{
         umanjiUplatilac();
         uvecajPrimalac();
      } catch(GMinus e){
         System.out.println(e);
         return false;
      }
      return true; }
   protected abstract void uvecajPrimalac()throws
            GMinus { primalac.promeniStanje(iznos); }
   protected abstract void umanjiUplatilac()throws
            GMinus { uplatilac.promeniStanje(-iznos);}
}
```

```
// Uplatnica.java
package banka;
public class Uplatnica extends ZahtevZaTransfer{
   private static final int PROVIZIJA = 1;
   public Uplatnica(Racun primalac, Racun
                     uplatilac, int iznos, Datum datum)
   { super(primalac, uplatilac, iznos, datum); }
   public String toString(){
      return super.toString() + "[" + iznos() + ":"
                  + iznos()*PROVIZIJA/100.0 + "]"; }
   public String vrsta(){ return "Uplata"; }
   protected void umanjiUplatilac()throws GMinus{
     uplatilac.promeniStanje(
               (int)(-iznos - iznos*PROVIZIJA/100.0));
   }
}
```

```
// KreditniZahtev.java
package banka;
public class KreditniZahtev extends ZahtevZaTransfer{
   private static final int PROVIZIJA = 5;
   public KreditniZahtev(Racun primalac, Racun
                  uplatilac, int iznos, Datum datum)
   { super(primalac, uplatilac, iznos, datum); }
   public String toString()
   { return super.toString() +"["+this.iznos+ "]"; }
   public String vrsta(){ return "Kredit"; }
   protected void uvecajPrimalac() throws GMinus{
      primalac.promeniStanje(
               (int)(iznos - iznos*PROVIZIJA/100.0)); }
}
```

```
// Banka.java
package banka;
public class Banka {
   private class Elem {
      ZahtevZaTransfer zahtev;
      Elem sled;
      Elem(ZahtevZaTransfer zahtev){
         this.zahtev = zahtev;
         Elem tek = glava, pret = null;
         while(tek!=null && !tek.zahtev.predaja().
                           noviji(zahtev.predaja()))
         { pret = tek; tek = tek.sled; }
         if (pret != null) pret.sled = this;
         else glava = this;
         this.sled = tek;
      }
   }
   private Elem glava;
   private Racun racun;
   public Banka(Racun racun)
   { this.racun = racun; }
   public Banka dodaj(ZahtevZaTransfer zahtev)
   { new Elem(zahtev); return this; }
   public ZahtevZaTransfer uzmi() throws GPrazna{
      if (glava == null) throw new GPrazna();
      ZahtevZaTransfer zahtev = glava.zahtev;
      glava = glava.sled;
      return zahtev;
   }
   public boolean izvrsi() throws GPrazna{
      ZahtevZaTransfer zahtev = this.uzmi();
      return zahtev.izvrsi();
   }
   public String toString(){
      StringBuilder s = new StringBuilder(racun.stanje()+"(");
      for(Elem tek = glava; tek != null; tek = tek.sled){
         s.append(tek.zahtev);
         if (tek.sled != null) {s.append(",\n");}
      }
      return s + ")";
   }
}
```

```
// GMinus.java
package banka;
public class GMinus extends Exception{
   public String toString(){ return "Nedozvoljeni
                                          minus!"; }}
```

```
// GPrazna.java
package banka;
public class GPrazna extends Exception{
   public String toString(){ return "Zbirka je
                                       prazna!"; }}
```

```
// BankaT.java
package banka;
public class BankaT {
   public static void main(String[] args){
      Racun klijentR = new Racun(-100, 3000);
      Racun bankaR = new Racun(0, 600000);
      Banka banka = new Banka(bankaR);
      ZahtevZaTransfer upl1 = new Uplatnica(bankaR,
                  klijentR, 200, new Datum(5,4,2016));
      ZahtevZaTransfer kredit = new KreditniZahtev(
                  klijentR,bankaR, 1000, new Datum(5,4,2016));
      ZahtevZaTransfer upl2 = upl1.clone();
      upl2.predaja(new Datum(5,5,2016));
      banka.dodaj(kredit).dodaj(upl1).dodaj(upl2);
      System.out.println(banka);
      try { while(banka.izvrsi()){} }
      catch (GPrazna e) { System.out.println(banka);}
   }
}
```

```
//primer izvrsavanja
600000(Kredit(2)5/4/2016[1000],
Uplata(1)5/4/2016[200.0:2.0],
Uplata(3)5/5/2016[200.0:2.0])
599454()
```