ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек СИ Београд 03.05.2019.

Други колоквијум из Објектно оријентисаног програмирања II

1) *(укупно 70 поена)* Саставити на језику *Java* следећи пакет класа:

- *(10 поена)* ***Карта*** се ствара са задатим знаком (*PIK, TREF, KARO, HERC*) и бројем (10-14) (није потребно вршити проверу опсега). Може да се одреди целобројна вредност карте по следећем критеријуму: 10 и 11 су 1 поен, остало 0 поена. Може да се испита да ли је број карте једнак броју задате карте. Текстуални опис карте је облика *знакБрој*.
- *(15 поена)* ***Шпил*** се састоји од задатог броја карата. Ствара се празан, након чега се карте могу додавати једна по једна на крај шпила (операција је без ефекта уколико је шпил пун). Могуће је узети карту из шпила на два начина. Први начин је узимање последње додате карте. Други начин је узимање карте чији је број једнак броју задате карте, почевши претрагу од прве додате карте (грешка *GPrazan* је уколико је шпил празан). Уколико у шпилу не постоји карта са задатим бројем и шпил није празан, узима се последња додата карта. Могуће је дохватити (без уклањања) карту са произвољне позиције у шпилу (индекси почињу од 0, при чему је карта на позицији 0 прва додата карта). Може да се одреди тренутни број карата у шпилу. Текстуални опис шпила је облика `[`*карта*`, `*карта*`,...,`*карта*`]`.
- *(20 поена) Активан* ***играч*** се ствара са именом и задатим шпилом, који представља карте које се налазе на столу. Сваки играч ствара шпил од 2 карте који држи у руци, а који је на почетку празан. Играчу може да се дохвати име, да се дода карта, при чему је он додаје у шпил који држи у руци. Може да се увећа тренутни целобројни број поена играча за задати број поена, као и да се дохвати тренутни број поена. Играч се покреће приликом стварања и не започиње одмах играње потеза, док му то не сигнализира делилац (видети ниже). Потез играча састоји се од следећих акција: играч размишља 200ms, дохвата прву додату карту из шпила на столу, а затим проналази (уколико постоји) у шпилу који држи у рукама карту са истим бројем као на карти коју је дохватио са стола. Уколико је у шпилу у руци пронашао тражену карту, додаје њу шпилу на столу. У супротном, у шпил на столу додаје последње додату карту из шпила који држи у руци. Након тога на стандардном излазу исписује карту коју је изабрао из шпила који држи у руци, јавља делиоцу да је завршио потез и чека на сигнал од делиоца да започне свој следећи потез. Уколико је празан шпил у руци, играч трајно завршава свој рад. Играч може да се обавести да треба да одигра потез и да трајно заустави свој рад.
- *(15 поена) Активан* ***делилац*** се ствара са задата 2 шпила (шпил који представља карте на столу и шпил који представља карте које делилац треба да дели) и низом играча којима треба да дели карте. Приликом стварања делилац дели сваком играчу у круг по једну карту, све док сваки играч не попуни свој шпил карата који држи у руци, након чега се делилац покреће, али не започиње одмах са послом. Посао делиоца састоји се од следећих акција: делилац сигнализира првом играчу да игра потез, затим чека док му први играч не јави да је завршио потез, и тако редом свим играчима. Након што сваки играч одигра потез завршава се један круг и делилац одређује победника круга. Победник круга је играч који је последњи у шпил на столу додао карту са истим бројем као на карти коју је додао први играч у том кругу. Уколико не постоји такав играч, први играч је победник. Победик круга први започиње следећи круг. Делилац увећава тренутни број поена победника круга за суму вредности свих карата које су додате у шпил на столу у том кругу. Након тога делилац исписује на стандардном излазу име играча победеника круга, уклања све карте из шпила на столу и дели преостале карте из свог шпила у круг, по једну карту сваком играчу, почевши од победника круга. Након што сваки играч остане без карата и прогласи победника последњег круга делилац трајно завршава свој рад. Делилац може да започне свој рад и да трајно заустави свој рад.
- *(10 поена)* ***Игра*** се ствара са низом играча и једним делиоцем. Игра може да се започне и да се заврши. Може да се одреди да ли је игра завршена. Може да се одреди победник игре (играч са највише поена) само ако је игра завршена.

(0 поена) Приложена је класа са главном функцијом која испитује основне функционалности пакета класа уз исписивање резултата на стандардном излазу (конзоли).

НАПОМЕНЕ:

а) Израда решења задатка траје **100** минута.
б) Рад се предаје искључиво на предвиђеном мрежном диску.
в) Називе типова ускладити са називима апстракција из текста задатка, али користити латинично писмо и велико почетно слово.
г) На располагању је приступ *Web* адреси: https://docs.oracle.com/en/java/javase/11/.
д) Није дозвољено имати поред себе електронске уређаје, без обзира да ли су укључени или искључени.
ђ) Резултати колоквијума биће објављени на *Web*-у на адреси: http://rti.etf.bg.ac.rs/rti/ir2oo2/index.html

```java
package main;

import igra.*;

public class Main {

   public static void main(String[] args) throws GPrazan, InterruptedException {

      Karta k1 = new Karta(Karta.Znak.HERC, 10), k2 = new Karta(Karta.Znak.PIK, 10),
           k3 = new Karta(Karta.Znak.TREF, 11), k4 = new Karta(Karta.Znak.PIK, 11),
           k5 = new Karta(Karta.Znak.KARO, 10), k6 = new Karta(Karta.Znak.TREF, 12);

      Spil zaDeljenje = new Spil(6);
      zaDeljenje.dodaj(k1);
      zaDeljenje.dodaj(k2);
      zaDeljenje.dodaj(k3);
      zaDeljenje.dodaj(k4);
      zaDeljenje.dodaj(k5);
      zaDeljenje.dodaj(k6);
      Spil naStolu = new Spil(2);

      //==============Kreirano u svrhe imenovanja metoda
      Igrac testIgrac = new Igrac("I3", naStolu);
      Delilac testDelilac = new Delilac(naStolu, new Spil(0), new Igrac[] {});
      testIgrac.zapocni();
      testIgrac.prekini();
      testDelilac.zapocni();
      testDelilac.prekini();
      Thread.sleep(50);
      //================================================

      Igrac i1 = new Igrac("I1", naStolu), i2 = new Igrac("I2", naStolu);
      Igrac[] igraci = new Igrac[] { i1, i2 };
      Delilac d1 = new Delilac(naStolu, zaDeljenje, igraci);
      Igra igra = new Igra(igraci, d1);

      igra.zapocni();
      Thread.sleep(2000);
      igra.prekini();
      Thread.sleep(50);
      System.out.println("Igrac pobednik: " + igra.pobednik().dohvIme() +
                         " | Poena: " + igra.pobednik().dohvPoene());

      /*=============Izlaz============= //imena mogu biti razlicita
         I1 dodaje PIK11
         I2 dodaje TREF11
         Pobednik runde: I2
         I2 dodaje PIK10
         I1 dodaje HERC10
         Pobednik runde: I1
         I1 dodaje TREF12
         I2 dodaje KARO10
         Pobednik runde: I1
         Igrac pobednik: I1 | Poena: 3
      */
   }
}
```

```
package igra;
public class Karta {
   public enum Znak {HERC, KARO, TREF, PIK};
   private Znak znak;
   private int br;
   public Karta(Znak z, int b) { znak = z; br = b; }
   public int vrednost() { return br==10 || br==11 ? 1 : 0; }
   public boolean istiBroj(Karta k) { return br == k.br; }
   public String toString() { return znak + "" + br;}
}
```

```
public class Spil {
   private Karta []karte;
   private int broj;
   public Spil(int kap) { karte = new Karta[kap]; }
   public int broj() { return broj; }
   public void dodaj(Karta k) {
      if(broj < karte.length) karte[broj++] = k;
   }
   public Karta dohvati(int ind) throws GPrazan {
      if(broj == 0) throw new GPrazan();
      return karte[ind];
   }
   public Karta uzmi() throws GPrazan {
      if(broj == 0) throw new GPrazan();
      Karta k = karte[broj-1]; karte[--broj] = null;
      return k;
   }
   public Karta uzmi(Karta k) throws GPrazan {
      for(int i=0; i<broj; i++)
         if(karte[i].istiBroj(k)) {
            Karta karta = karte[i];
            for(int j=i+1; j<broj; j++) karte[j-1]=karte[j];
            karte[--broj] = null;
            return karta;
         }
      return uzmi();
   }
   public String toString() {
      StringBuilder sb = new StringBuilder("[");
      for(int i=0; i<broj; i++) {
         sb.append(karte[i]);
         if(i < broj - 1) sb.append(",");
      }
      sb.append("]");
      return sb.toString();
   }
}
```

```
public class Igrac extends Thread {
   public static final int U_RUCI = 2;
   private String ime;
   private Spil uRuci = new Spil(U_RUCI), naStolu;
   private int poeni;
   private boolean radi = false;
   public Igrac(String i, Spil s) {
      ime = i; naStolu = s; start();
   }
   public String dohvIme() { return ime; }
   public void dodaj(Karta k) { uRuci.dodaj(k); }
   public void uvecajPoene(int p) { poeni += p; }
   public synchronized int dohvPoene() { return poeni; }
   public synchronized void zapocni() { radi=true; notify(); }
   public void prekini() { interrupt(); }
   public void run() {
      try {
         while(!isInterrupted()) {
            synchronized (this) {
               while(!radi)
                  wait();
            }
            Thread.sleep(200);
            Karta k = null, moja = null;
            try {
               k = naStolu.dohvati(0);
            } catch (GPrazan e) {}
            if(k != null) moja = uRuci.uzmi(k);
            else moja = uRuci.uzmi();
            naStolu.dodaj(moja);
            System.out.println(ime + " dodaje " + moja);
            synchronized (this) { radi = false; notify(); }
         }
      } catch(InterruptedException | GPrazan e) {
         synchronized (this) { radi = false; notify(); }
      }
   }
}
```

```
public class Delilac extends Thread {
   private Igrac []igraci;
   private Spil karte, naStolu;
   private boolean radi;
   public Delilac(Spil s, Spil d, Igrac []ig) throws GPrazan {
      igraci = ig; karte = d; naStolu = s;
      for(int i=0; i<Igrac.U_RUCI; i++)
         for(Igrac igrac : igraci)
            igrac.dodaj(karte.uzmi());
      start();
   }
   public synchronized void zapocni() {radi = true; notify();}
   public void prekini() { interrupt(); }
   public void run() {
      int igrac = 0;
      try {
         while(!isInterrupted()) {
            synchronized (this) {
               while(!radi)
                  wait();
            }
            Karta prva = null;
            for(int i=0; i<igraci.length; i++) {
               int ind = (igrac + i) % igraci.length;
               synchronized (igraci[ind]) {
                  igraci[ind].zapocni();
                  igraci[ind].wait();
               }
            }
            try {
               prva = naStolu.dohvati(0);
            } catch (GPrazan e) { return; }
            int ind, poeni = 0;
            Karta poslKar = null;
            boolean flag = true;
            for(int i = igraci.length-1; i >= 0; i--) {
               ind = (igrac + i) % igraci.length;
               try {
                  poslKar = naStolu.uzmi();
               } catch (GPrazan e) {}
               poeni += poslKar.vrednost();
               if(poslKar.istiBroj(prva) && flag) {
                  igrac = ind; flag = false;
               }
            }
            igraci[igrac].uvecajPoene(poeni);
            System.out.println("Pobednik runde: " +
                                 igraci[igrac].dohvIme());
            for(int i=0; i<igraci.length; i++)
               try {
                  igraci[(igrac + i) % igraci.length]
                                    .dodaj(karte.uzmi());
               } catch(GPrazan e) {}
         }
      } catch (InterruptedException ie) {}
   }
}
```

```
public class Igra {
   private Igrac []igraci;
   private Delilac delilac;
   public Igra(Igrac[] ig, Delilac d) {igraci=ig; delilac=d;}
   public void zapocni() { delilac.zapocni(); }
   public void prekini() {
      delilac.prekini();
      for(Igrac i : igraci) i.prekini();
   }
   public boolean zavrsena() {
      for(Igrac i : igraci)
         if(i.isAlive()) return false;
      if(delilac.isAlive()) return false;
      return true;
   }
   public Igrac pobednik() {
      if(!zavrsena()) return null;
      Igrac p = null;
      for(int i=0; i<igraci.length; i++)
         if(p == null || igraci[i].dohvPoene()>p.dohvPoene())
            p = igraci[i];
      return p;
   }
}
```

```
public class GPrazan extends Exception {}
```