ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за СИ Београд 04.05.2016.

Други колоквијум из Објектно оријентисаног програмирања II

1) *(30 поена)* Одговорити са "да"/"не" на сваки нумерисани део следећих питања:

а) Да ли интерфејс може да садржи: (1) симболичке константе, (2) промењљива поља, (3) апстрактне методе, (4) подразумеване конкретне методе, (5) статичке методе, (6) конструкторе?

б) Да ли се објекат анонимне класа која (1) имплементира интерфејс `I`, (2) проширује апстрактну класу `A`, (3) проширује конкретну класу `K`, може створити уз позив конструктора наткласе са неким аргументима?

в) Да ли виртуелна машина (1) открива, (2) спречава или (3) разрешава узајамно блокирање нити (*deadlock*)?

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

- *(30 поена)* ***Игру*** игра задати број играча за партију игре, од оних који су се пријавили за игру. Памте се играчи који играју партију, стање игре, број тренутно пријављених играча, максимално време трајања партије и време трајања текуће партије (0 ако партија није у току). Грешка је ако је максимално време трајања партије негативно. Игра се ствара без играча. Могућа стања игре су `PARTIJA_NIJE_ZAPOCETA` и `PARTIJA_U_TOKU`. Игра се ствара са стањем `PARTIJA_NIJE_ZAPOCETA`. Играч може да се пријави за игру, само ако је одговарајућег типа. Уколико је, приликом пријављивања играча, партија већ у току или број пријављених играча још није довољан за партију, играч који је покушао да се пријави се привремено блокира. Уколико, приликом пријављивања играча, партија није у току и број пријављених играча је довољан за партију, партија може да почне. Непосредно пред почетак партије (у тренутку када се пријави довољан број играча) одређује се време трајања партије, као случајно време у опсегу од 0 до максималног времена трајања партије. Играч може да заврши партију. Покушај завршавања партије од стране играча који је није играо се игнорише. Партија је завршена када је заврше сви играчи који су је играли. Нова партија може да почне ако у игри постоји довољно пријављених играча. Ако је у тој ситуацији број пријављених играча већи него што је довољно за партију, небитно је који ће од њих добити прилику да играју партију. Могуће је дохватити број пријављених играча, стање игре, време трајања партије и назив игре. Могуће је саставити текстуални опис у облику *назив – стање<нова линија>играч<нова линија>играч... <нова линија>играч*.
- ***Образовна игра*** је конкретна игра задатог назива и свих параметара игре. Могу да је играју само ученици.
- (*30 поена*) Активан ***играч*** има име, игру коју игра, минимално и максимално време чекања између завршавања једне партије и пријављивања за следећу (грешка је ако играч не може да се пријави на задату игру и ако при стварању минимално и максимално време нису исправно задати). Играч се циклички пријављује за игру, игра партију игре и завршава партију. Текстуални опис је у облику *име*.
- ***Ученик*** је конкретан играч који игра образовну игру. Његова игра партије се симулира спавањем у времену трајања те партије.

(10 поена) Написати на језику *Java* програм (класу с главном функцијом, која није у горњем пакету) који направи образовну игру `"KVIZ"` за 2 играча, са максималним временом трајања партије 10 секунди, створи 5 играча за игру, и после 5 секунди испише игру на стандардном излазу. При писању програма 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`.

```java
// Igra.java
package igra;
public abstract class Igra {
   private static enum Stanje
            {PARTIJA_NIJE_ZAPOCETA, PARTIJA_U_TOKU}
   private Igrac[] igraci;
   private int pop = 0;
   private Stanje stanje =
                     Stanje.PARTIJA_NIJE_ZAPOCETA;
   private int brPrijavljenih = 0;
   private long maxT, t = 0;
   public Igra(int n, long max) throws GNeg
   {  if(max < 0) throw new GNeg();
      igraci = new Igrac[n]; maxT = max; }
   public synchronized void prijavi(Igrac i) throws
                        GTip, InterruptedException
   {  proveriTip(i);
      brPrijavljenih++;
      try { proveri(); }
      catch (InterruptedException e)
      { brPrijavljenih--; throw e; }
      dodaj(i); }
   public abstract void proveriTip(Igrac i) throws
                                                GTip;
   private void proveri() throws
                              InterruptedException
   {  if (stanje == Stanje.PARTIJA_NIJE_ZAPOCETA &&
                  brPrijavljenih >= igraci.length)
      { notifyAll(); }
      else
      {
         while (stanje == Stanje.PARTIJA_U_TOKU ||
                  brPrijavljenih < igraci.length)
            { wait(); } } }
   private void dodaj(Igrac i)
   {  igraci[pop] = i; pop++;
      if (pop == 1) t=(long)(Math.random()*(maxT));
      else if (pop == igraci.length)
      {  stanje = Stanje.PARTIJA_U_TOKU;
         brPrijavljenih -= pop; } }
   public synchronized void zavrsi(Igrac i)
   {  int j = 0;
      for (; j < igraci.length; j++)
      { if (igraci[j] == i) break; }
      if (j >= igraci.length) return;
      izbaci(j); }
   private void izbaci(int i)
   {  igraci[i] = null; pop--;
      if (pop == 0)
      {  stanje = Stanje.PARTIJA_NIJE_ZAPOCETA;
         t = 0; notifyAll(); } }
   public synchronized int brPrijavljenih()
   { return brPrijavljenih; }
   public synchronized Stanje stanje()
   { return stanje; }
   public synchronized long trajanje()
   { return t; }
   public abstract String naziv();
   public synchronized String toString()
   {  StringBuffer s = new StringBuffer();
      s.append(naziv()).append(" - ")
                     .append(stanje()).append('\n');
      if (igraci != null)
      {  for (int i=0; i<igraci.length; i++)
         {  if (i != 0) s.append('\n');
            if (igraci[i] != null)
               s.append(igraci[i]);
            else s.append("prazno"); } }
      return s.toString(); }
}
```

```java
// ObrazovnaIgra.java
package igra;
public class ObrazovnaIgra extends Igra
{  public String naziv;
   public ObrazovnaIgra(String n, int b, long max)
                                          throws GNeg
   { super(b, max); naziv = n; }
   public String naziv() { return naziv; }
   public void proveriTip(Igrac i) throws GTip
   { if (!(i instanceof Ucenik)) throw new GTip();}
}
```

```java
// Igrac.java
package igra;
public abstract class Igrac extends Thread
{  private String ime;
   protected Igra igra;
   private int minT, maxT;
   public Igrac(String i, Igra ii, int min, int max)
                              throws GOpseg, GTip
   {  proveriTip(ii);
      if(min > max) throw new GOpseg();
      ime = i; igra = ii;
      minT = min; maxT = max; start(); }
   public abstract void proveriTip(Igra i) throws
                                                GTip;
   public void run()
   {  try
      {  while(!interrupted())
         {  igra.prijavi(this);
            try { igraj(); }
            catch (InterruptedException e)
            {  igra.zavrsi(this); throw e; }
            igra.zavrsi(this);
            sleep((long)(minT + Math.random()*(maxT-
                                          minT))); }
      } catch (InterruptedException e) {
      } catch (GTip e) { System.out.println(e); } }
   public abstract void igraj() throws
                                 InterruptedException;
   public String toString(){ return ime; }
}
```

```java
// Ucenik.java
package igra;
public class Ucenik extends Igrac
{  public Ucenik(String i, Igra ii, int min,
                     int max) throws GOpseg, GTip
   { super(i, ii, min, max); }
   public void proveriTip(Igra i) throws GTip
   { if (!(i instanceof ObrazovnaIgra)) throw new
                                          GTip(); }
   public void igraj() throws InterruptedException
   { Thread.sleep(igra.trajanje()); }
}
```

```java
// GNeg.java
package igra;
public class GNeg extends Exception
{  public String toString()
   { return "Negativno maksimalno vreme!"; } }
```

```java
// GTip.java
package igra;
public class GTip extends Exception
{  public String toString()
   { return "Neispravno zadat tip!"; }}
```

```java
// GOpseg.java
package igra;
public class GOpseg extends Exception
{  public String toString()
   { return "Neispravno zadat opseg!"; }}
```

```java
// Main.java
import igra.*;
public class Main {
   public static void main(String[] args)
   {  try
      {  ObrazovnaIgra igra = new
                  ObrazovnaIgra("Kviz", 2, 10000);
         int brIgraca = 5;
         Igrac[] igraci = new Igrac[brIgraca];
         for (int i = 0; i < brIgraca; i++)
         {  igraci[i] = new
         Ucenik("ucenik_"+i,igra,100*i,200+100*i); }
         Thread.sleep(5000);
         System.out.println(igra);
         for (int i = 0; i < brIgraca; i++)
         {  igraci[i].interrupt();  }
      } catch (GNeg e) { System.out.println(e); }
      catch (GOpseg e) { System.out.println(e); }
      catch (GTip e) { System.out.println(e); }
      catch (InterruptedException e)
      { e.printStackTrace(); }
   }
}
```

```
Kviz - PARTIJA_U_TOKU
ucenik_0
ucenik_4
```