ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за РТИ и Одсек за СИ Београд 16.06.2014.

Испит из Објектно оријентисаног програмирања II

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

а) (*Java*) Да ли конструктор може бити синхронизован и зашто?

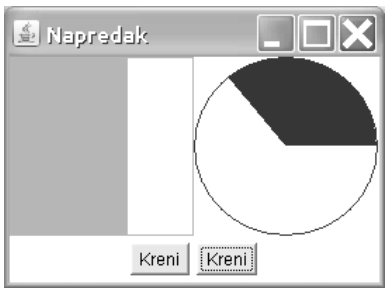
б) (*C#*) Навести врсте .NET склопова које могу да се извршавају самостално. Која опција преводиоца `csc` се користи за коју врсту и која је екстензија имена (тип) фајла који се добија као резултат превођења?

в) (*C#*) Шта исписује следећи програм?

```
class X{
  delegate int D(ref int i);
  private int i=10;
  int m1(ref int i){return i++;}
  int m2(ref int i){return i*=2;}
  int m3(ref int i){return i--;}
  static void Main(string[] args){ X x=new X();
    D d1=new D(x.m1), d2=new D(x.m2), d3=new D(x.m3);
    D d=d1+d2; d+=d1; d+=d3-d1;
    System.Console.WriteLine("d="+d(ref x.i)+" x.i="+x.i);
  }
}
```

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

- *(30 поена)* ***Обавестива*** ствар може да буде обавештена о процентуалном напредовању.
- Активан ***посао*** извршава неку радњу *N* пута. Понављање радње може привремено да се заустави и настави, као и да се трајно прекине. Може да се дохвати стање (заустављен/извршава се). Послу може да се пријави највише једна обавестива ствар (више од једне је грешка). После сваких *K* извршавања радње посао шаље пријављеној обавестивој ствари обавештење о процентуалном напредовању посла.
- ***Симулирани посао*** остварује радњу спавањем случајног трајања у задатом опсегу времена у ms.
- *(30 поена)* ***Индикатор*** напретка је обавестиво платно (*Canvas*) које служи за графички приказ напредовања одговарајућег посла задатом бојом.
- ***Правоугаони индикатор*** задатом бојом представља напредовање у облику правоугаоника који покрива целу компоненту. Напредовање се представља одговарајућим процентуалним попуњавањем правоугаоника у правцу дуже странице (слева-удесно или одозго-наниже).
- ***Кружни индикатор*** задатом бојом представља напредовање у облику исечка круга уписаног у целу компоненту. Напредовање се представља одговарајућим процентуалним попуњавањем исечка круга у обрнутом смеру од казаљке часовника.
- ***Напредак*** је обавестива ствар која садржи индикатор напретка. Може да се пријави код задатог посла, како би пратила његов напредак. Неуспех пријаве се игнорише. Обавештење о напретку прослеђује садржаном индикатору.



(10 поена) ***Програм*** на графичкој корисничкој површи према слици, приказује један правоугаони и један кружни индикатор напредовања одговарајућих симулираних послова. Одговарајућа дугмад за заустављање/наставак рада мењају натпис "Крени"/"Стани" у зависности од стања одговарајућег посла чијим радом управљају. Користити фиксне вредности свих параметара, не треба ништа читати са конзоле.

НАПОМЕНЕ: **а)** Испит траје **180** минута.
б) Рад се предаје искључиво у вежбанци за испите. Није дозвољено имати поред себе друге листове папира, нити уз себе имати мобилни телефон, без обзира да ли је укључен или искључен.
в) Дозвољена је употреба ***Подсетника за AWT***.
г) 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.
д) Резултати испита биће објављени на *Web*-у на адреси: `home.etf.rs/~kraus/` (одреднице: *настава* | <име предмета> | *оцене* | *испити*).

```
// PrevisePratilaca.java
package napredovanje;

public class PrevisePratilaca
             extends Exception{}
```

```
//Obavestiv.java

package napredovanje;
public interface Obavestiv{
  void obavesti(int procenat);
}
```

```
// Posao.java
package napredovanje;

import java.awt.*;

public abstract class Posao extends Thread{
  private int N, K;
  private Obavestiv pratilac;
  private boolean radi;

  public Posao(int N, int K) {
     this.N=N; this.K=K; }

  public void prijavi(Obavestiv o)
           throws PrevisePratilaca{
    if (pratilac != null)
      throw new PrevisePratilaca();
    pratilac = o;
  }

  public void zaustavi(){radi=false; }

  public synchronized void  nastavi(){
     radi = true;
     notify();
  }

  public void prekini() { interrupt(); }

  public boolean radi() { return radi; }

  protected abstract void radnja()
    throws InterruptedException;

  public void run() {
    try {
      for (int i=1; i<=N; i++) {
        if (interrupted()) break;
        while(! radi )
          synchronized(this) { wait(); }
        radnja();
        if ((pratilac!=null)
             &&((i%K)==0))
          pratilac.obavesti(i*100/N);
      }
    }
    catch(InterruptedException e) { }
  }
}
```

```
// SimPosao.java
package napredovanje;

import java.awt.*;

public class SimPosao extends Posao {
  private int minT, maxT;
  public SimPosao(int N, int K,
                  int minT,int maxT)
  { super(N,K); this.minT=minT;
                this.maxT=maxT; }

  protected void radnja()
    throws InterruptedException{
    sleep(minT +
  (int)(Math.random()*(maxT-minT+1)));
  }
}
```

```
// Indikator.java
package napredovanje;

import java.awt.*;

public class Indikator extends Canvas
implements Obavestiv{
  private Color boja;
  protected int procenat;
  protected int sirina;
  protected int visina;
  public Indikator(Color boja){
    this.boja=boja;
  }
  public void obavesti(int procenat){
    this.procenat=procenat;
    repaint();
  }
  public void paint(Graphics g){
    sirina = this.getWidth();
    visina = this.getHeight();
    g.setColor(boja);
  }
}
```

```
// PravougaoniIndikator.java

package napredovanje;

import java.awt.*;

public class PravougaoniIndikator extends
Indikator{
  public PravougaoniIndikator(
     Color boja){ super(boja);  }

  public void paint(Graphics g){
    super.paint(g);
    g.drawRect(0,0,sirina-1,visina-1);
    if (sirina >= visina)
      g.fillRect(0,0,
       sirina*procenat/100-1,visina-1);
    else
      g.fillRect(0,0,
       sirina-1,visina*procenat/100-1 );
  }
}
```

```
// KruzniIndikator.java
package napredovanje;

import java.awt.*;

public class KruzniIndikator extends
Indikator{
  public KruzniIndikator(Color boja){
    super(boja);
  }
  public void paint(Graphics g){
    super.paint(g);
    g.drawOval(0,0,sirina-1,visina-1);
    g.fillArc(0,0, sirina-1,
        visina-1,0,360*procenat/100);
   }
}
```

```
// Napredak.java
package napredovanje;

import java.awt.*;

public class Napredak implements Obavestiv{
  private Indikator ind;

  public Napredak(Indikator ind){
     this.ind = ind;
  }

  public void prati(Posao p){
    try{  p.prijavi(this);
    } catch(PrevisePratilaca i){}
  }

  public synchronized void obavesti(

  int procenat){
     ind.obavesti(procenat);
  }
}
```

```
//Program.java

import napredovanje.*;
import java.awt.*;
import java.awt.event.*;

public class Program extends Frame {
  private Button dugPrav, dugKrug;
  private SimPosao sp1, sp2;
  private Napredak pravNap, krugNap;
  private Indikator pravInd, krugInd;

  public Program() {
    super("Napredak");
    setSize(200, 200);
    addWindowListener(new WindowAdapter() {
        public void
         windowClosing(WindowEvent e){
          sp1.prekini(); sp2.prekini();
          dispose();
      }});
    sp1 = new SimPosao(1000,50,10,15);
    sp1.start();
    sp2 = new SimPosao(1000,10,10,15);
    sp2.start();
    pravInd = new
    PravougaoniIndikator(Color.GREEN);
    krugInd = new
    KruzniIndikator(Color.RED);
    pravNap = new Napredak(pravInd);
    pravNap.prati(sp1);
    krugNap = new Napredak(krugInd);
    krugNap.prati(sp2);
    dodajKomponente();
    setVisible(true);
  }

  private void dodajKomponente() {
    Panel indikatori = new Panel();
    indikatori.setLayout(new
                GridLayout(1,2));
    indikatori.add(pravInd);
    indikatori.add(krugInd);
    add(indikatori, "Center");
    dugPrav = new Button("Kreni");
    defDugme(dugPrav, sp1);
    dugKrug = new Button("Kreni");
    defDugme(dugKrug, sp2);
    Panel dugmad = new Panel();
    dugmad.add(dugPrav);
    dugmad.add(dugKrug);
    add(dugmad, "South");
    validate();
  }

  private void defDugme(final Button dugme,
                        final Posao posao){
    dugme.addActionListener(new
             ActionListener() {
      public void actionPerformed(
          ActionEvent e){
        if (!posao.isAlive()){
          dugme.setEnabled(false);
          return;
        }

        if (posao.radi()) {
            posao.zaustavi();
            dugme.setLabel("Kreni");
          } else {
            posao.nastavi();
            dugme.setLabel("Stani");
          }
        }
      });
    }

    public static void
     main(String[] args){new Program();}
}
```