ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсеци СИ, РТИ, ОС, ОТ Београд 10.06.2016.

Испит из
Објектно оријентисаног програмирања II

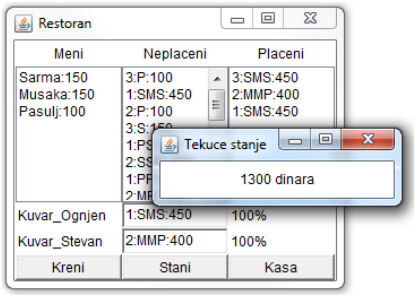
1) *(30 поена)* Одговорити концизно и прецизно на следећа питања:

а) (*Јава*) Да ли следеће класе обезбеђују да сви објекти тог типа садрже компоненте корисничког интерфејса: (1) `Component`, (2) `Container`, (3) `Graphics`, (4) `Frame`, (5) `Panel`?

б) (*C#*) Повезати операторе `is` и `as` са операторима језика Java који им најбоље одговарају.

в) (*C#*) За следеће корисничке типове навести да ли су указни (P) или вредносни (B), да ли им се дефинише само уговор и да ли им се могу стварати објекти: (1) конкретна класа, (2) апстрактна класа, (3) интерфејс, (4) структура, (5) набрајање, (6) делегат? (Пример: P, да, да).

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



- *(15 поена)* ***Јело*** има назив, цену, време припреме (ms) и једнословну ознаку, које могу да се дохвате. Једнословна ознака је прво слово назива. Текстуални облик је *назив*`:`*цена*.
- ***Мени*** има придружену графичку листу и низ јела. Садржана јела се приказују на графичкој листи при стварању менија и може се дохватити укупан број ставки менија. Могуће је дохватити јело са менија по индексу (грешка је ако тражено јело не постоји).
- ***Поруџбина*** има број стола са ког потиче и низ поручених јела, који могу да се дохвате. Може се одредити вредност поруџбине као збир цена поручених јела. Текстуални облик је *број_стола*`:`*ознака_јела ознака_јела... ознака_јела*`:`*вредност*.
- *(40 поена)* ***Пулт*** има зараду, неограничене збирке неплаћених и плаћених поруџбина и њима придружене графичке листе. Пулт се ствара без поруџбина, које се додају једна по једна на крај збирке неплаћених поруџбина. Са пулта може да се узме прва неплаћена поруџбина што подразумева ажурирање приказа одговарајуће графичке листе. Може да се наплати задата поруџбина, што подразумева увећавање зараде пулта за вредност поруџбине, додавање поруџбине на крај збирке плаћених поруџбина и ажурирање приказа одговарајуће графичке листе. Текстуални облик исписа на графичким листама је једна поруџбина по реду.
- ***Активан сто*** има јединствени целобројни идентификатор и придружен мени и пулт. Сто циклички генерише случајне поруџбине које садрже од 1 до 3 јела са менија, додаје их на пулт, а затим паузира између 4ms и 7ms. Рад стола може да се покрене, привремено заустави и трајно прекине.
- ***Активан кувар*** има име, придружен пулт и графичку плочу на којој се редом налазе текстуални опис кувара, поруџбина коју тренутно обрађује и проценат обраде, као на приказаној слици. Кувар циклички узима поруџбине са пулта (блокира се ако нема ниједне) и припрема једно по једно јело узето из поруџбине. Припрема јела се симулира успављивањем у трајању припреме јела. Проценат обрађеног дела поруџбине се повећава након завршетка припреме сваког јела из поруџбине. Након припреме последњег јела из поруџбине, врши се њена наплата. Текстуални опис кувара је **`Kuvar_`***име*.

(15 поена) ***Ресторан*** је прозор који према приказаној слици симулира рад једног менија, једног пулта, три стола и два кувара. Рад ресторана може да се привремено заустави и покрене, као и да се прикаже тренутна зарада у посебном дијалогу.

НАПОМЕНЕ: **а)** Испит траје **180** минута. **б)** Дозвољена је употреба ***Подсетника за AWT***

в) 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.

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

```
package restoran;
import java.awt.*;
import java.awt.event.*;
// Jelo.java
public class Jelo {
  private String ime;
  private long vreme; private int cena;
  public Jelo(String i, int c, int v){
    ime = i; cena = c; vreme = v; }
  public String ime() { return ime; }
  public int cena() { return cena; }
  public long vreme() { return vreme; }
  public String toString() {
    return ime + ":" + cena; }
  public char ozn(){return ime.charAt(0);}
}
```

```
// GNema.java
public class GNema extends Exception {
  public GNema(){ super("***Nema jela.");}
}
```

```
// Meni.java
public class Meni {
  private List prikaz;
  private Jelo[] jela;
  public Meni(List p, Jelo[] j){
    prikaz = p; jela = j;
    for(Jelo jelo:j)
      if(jelo!=null)
        prikaz.add(jelo.toString()); }
  public Jelo naruci(int ind) throws GNema{
    if(jela[ind]==null)throw new GNema();
    return jela[ind]; }
  public int brJela(){return jela.length; }
}
```

```
// Porudzbina.java
public class Porudzbina {
private int sto; private Jelo[] jela;
  public Porudzbina(Jelo[] j, int b){
    jela = j; sto = b; }
  public int dohvRacun(){
    int racun = 0;
    for(Jelo j: jela) racun += j.cena();
    return racun;
  }
  public String toString(){
    String sb = "" + sto + ":";
    for(int i=0; i < jela.length; i++)
      sb += jela[i].ozn();
      sb += ":" + dohvRacun();
    return sb.toString();
  }
  public Jelo[] dohvJela(){ return jela; }
}
```

```
// Pult.java
public class Pult {
  private int kasa;
  private List lstNeplacene, lstPlacene;
  private Elem prvaN, prvaP, poslN, poslP;
  private class Elem {
    Porudzbina p;
    Elem sled;
    Elem(Porudzbina por, boolean n){
      p = por;
      if(n) {
        if(poslN!=null) poslN.sled = this;
        else prvaN = this;
        poslN = this;
      }
      else {
        if(poslP!=null) poslP.sled = this;
        else prvaP = this;
        poslP = this;
      }
    }
  }
  public Pult(List lstN, List lstP){
    lstNeplacene=lstN; lstPlacene=lstP; }
  public synchronized void
                      dodaj(Porudzbina p){
    new Elem(p, true); notifyAll();
    lstNeplacene.add(p.toString());
  }
  public synchronized Porudzbina
                          dohvPorudzbinu()
                throws InterruptedException{
    while(prvaN == null) wait();
    Porudzbina p = prvaN.p;
    prvaN = prvaN.sled;
    if(prvaN == null) poslN = null;
    lstNeplacene.remove(0); return p;
  }
  public synchronized void
                      naplati(Porudzbina p){
    new Elem(p, false);
    lstPlacene.add(p.toString());
    kasa += p.dohvRacun();
  }
  public synchronized int kasa(){
    return kasa; } }
```

```
// Sto.java
public class Sto extends Thread{
  private Pult pult;
  private Meni meni;
  private boolean radi = false;
  private static int ID = 0;
  private int brStola = ++ID;
  public Sto(Pult pult, Meni meni){
    this.pult = pult;
    this.meni = meni; start();
  }
  private Porudzbina generisi(){
    int brJela = (int)(1+Math.random()*3);
    int ukJela = meni.brJela();
    Jelo[] jela = new Jelo[brJela];
    for(int i=0; i<jela.length;){
      int ind=(int)(Math.random()* ukJela);
      try{ jela[i] = meni.naruci(ind);i++;}
      catch(GNema g) {}
    }
    return new Porudzbina(jela, brStola);
  }
  public void run(){
    try{
      while(!interrupted()){
        synchronized (this){
          if(!radi) wait(); }
        pult.dodaj(generisi());
        sleep((long)(4000 + Math.random() *
3000));
      }
    } catch(InterruptedException e){}
  }
  public synchronized void kreni(){
    radi = true; notify(); }
  public void stani(){ radi = false; }
  public void prekini(){ interrupt();}
}
```

```
// Kuvar.java
public class Kuvar extends Thread{
  private String ime;
  private Pult pult;
  private TextField tfPor;
  private Label lblProg;
  private Panel ploKuvar;
  private boolean radi = false;
  private double delta, progres;
  public Kuvar(Pult p,String i,Panel plo){
    pult = p; ime = i; ploKuvar = plo;
    popuniPlocu(); start(); }
  public void run(){
    try{
      while(!interrupted()) {
        synchronized (this){
          if(!radi)wait();}
        Porudzbina p =
                    pult.dohvPorudzbinu();
        novaPor(p);
        for(Jelo j: p.dohvJela()) {
          sleep(j.vreme());uvecajProgres();
        }
        pult.naplati(p);
      } } catch(InterruptedException e){}
  }
  private void novaPor(Porudzbina p){
    progres = 0;
    int brJela = p.dohvJela().length;
    delta = 100.0/brJela;
    tfPor.setText(p.toString());
    lblProg.setText((int)progres + "%");
  }
  private void uvecajProgres(){
    progres += delta;
    lblProg.setText((int)progres + "%");
  }
  public String toString(){
    return "Kuvar_" + ime; }
  private void popuniPlocu(){
    tfPor = new TextField("");
    lblProg = new Label("0%");
    ploKuvar.add(new Label(toString()));
    ploKuvar.add(tfPor);
    ploKuvar.add(lblProg); }
  public synchronized void kreni(){
    radi = true; notify(); }
  public void stani(){ radi = false; }
  public void prekini(){ interrupt();}
}
```

```
// Restoran.java
public class Restoran extends Frame{
  private Pult pult;
  private Sto[] stolovi;
  private Kuvar[] kuvar;
  private Kasa kasa = new Kasa();
  public class Kasa extends Frame{
    Label lbl;
    Kasa(){
      super("Tekuce stanje u kasi");
      setBounds(250, 300, 150, 50);
      add(lbl=new Label("", Label.CENTER));
      this.addWindowListener
      (new WindowAdapter() {
public void windowClosing (WindowEvent e){
       setVisible(false); }));
    }
    void prikaziStanje(int stanje){
      lbl.setText(stanje+" dinara"); }
  }
  public Restoran(){
    super("Restoran");
    setBounds(200,200,300,250);
    popuniProzor();
    addWindowListener(new WindowAdapter() {
  public void windowClosing(WindowEvent e){
    for(int i=0; i<stolovi.length; i++)
      stolovi[i].prekini();
    for(int i=0;i<kuvar.length;i++)
      kuvar[i].prekini();
    kasa.dispose(); dispose();
  }});
  }
  private void popuniProzor(){
    // meni i pult
    Panel pl = new Panel();
    pl.setLayout(new GridLayout(1, 3));
    add(pl, "North");
    pl.add(new Label("Meni",Label.CENTER));
    pl.add(
      new Label("Neplaceni",Label.CENTER));
    pl.add(
      new Label("Placeni",Label.CENTER));
    pl= new Panel(new GridLayout(1, 2));
    add(pl, "Center");
    List lst = new List();
    Meni meni = new Meni(lst, new Jelo[]{
      new Jelo("Sarma", 150, 3000),
      new Jelo("Musaka", 150, 3000 ),
      new Jelo("Pasulj", 100, 3000)});
    pl.add(lst); lst = new List();
    List lst2 = new List();
    pult = new Pult(lst, lst2);
    pl.add(lst); pl.add(lst2);
    // kuvari
    pl = new Panel(new GridLayout(3, 3));
    add(pl, "South");
    kuvar = new Kuvar[2];
    String[] im = {"Ognjen", "Stevan"};
    for(int i = 0; i<kuvar.length; i++)
      kuvar[i]=new Kuvar(pult, im[i], pl);
    Button btn = new Button("Kreni");
    pl.add(btn);
    btn.addActionListener((e)->{
      kreniRestoran();}
    );
    btn = new Button("Stani"); pl.add(btn);
    btn.addActionListener((e)->{
      stopRestoran();}
    );
    btn = new Button("Kasa"); pl.add(btn);
    btn.addActionListener((e)->{
      kasa.prikaziStanje(pult.kasa());
      kasa.setVisible(true);
      }
    );
    // stolovi
    stolovi = new Sto[3];
    for(int i=0;i<stolovi.length;i++)
      stolovi[i]=new Sto(pult,meni);
  }
  private void stopRestoran(){
    for(int i = 0; i<kuvar.length; i++)
      kuvar[i].stani();
    for(int i = 0; i<stolovi.length; i++)
        stolovi[i].stani();
  }
  private void kreniRestoran(){
    for(int i = 0; i<stolovi.length; i++)
      stolovi[i].kreni();
    for(int i = 0; i<kuvar.length; i++)
        kuvar[i].kreni();
  }
  public static void main(String[] args){
    new Restoran().setVisible(true); }
}
```