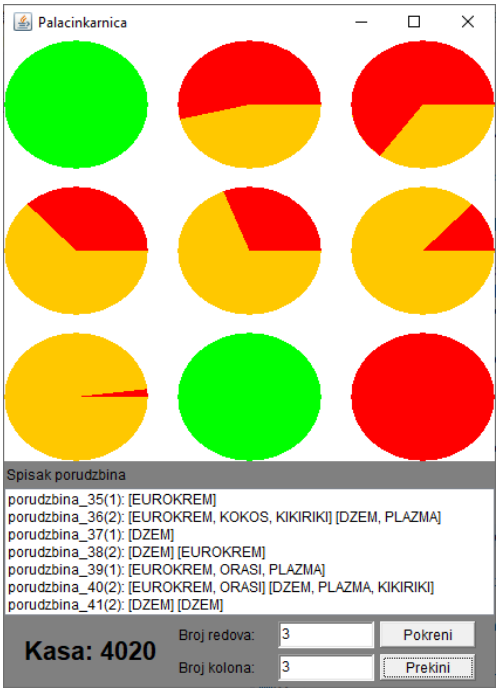
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Одсеци СИ, ИР, ОС, ОФ Београд 30.06.2019.

Завршни поправни колоквијум из Објектно оријентисаног програмирања II

1) *(укупно 100 поена)* Саставити на језику *Java* следећи пакет класа:

- *(10 поена)* ***Прилог*** садржи састојак (*PLAZMA, KIKIRIKI, KOKOS, ORASI*) и целобројну цену, који се задају приликом стварања и могу да се дохвате. Прилози су једнаки уколико су њихови састојци једнаки.
- *(20 поена)* ***Палачинка*** садржи премаз (*EUROKREM, DZEM*) који се задаје приликом стварања и произвољан број прилога, који се само накнадно могу додавати један по један. Грешка је уколико се покуша додавање прилога чији је састојак исти као састојак већ додатог прилога (`GSastojak`). Може да се одреди цена палачинке као збир цене премаза (цена еурокрема је 100 дин; цена џема је 70 дин) и цена свих додатих прилога. Текстуални опис палачинке садржи називе премаза и састојака садржаних прилога одвојене зарезима (нпр: **`EUROKREM, PLAZMA`**).
- *(15 поена)* ***Поруџбина*** садржи аутоматски генерисан јединствен целобројан идентификатор и произвољан број палачинки. Ствара се празна, након чега се палачинке додају једна по једна. Могуће је избацити све палачинке из поруџбине и дохватити број палачинки у поруџбини. Може да се дохвати цена поруџбине као збир цена свих палачинки које садржи. Текстуални опис је облика **`porudzbina_`***ид* **`(`***број_палачинки***`):`** након чега се палачинке исписују у облику **`[`***палачинка***`]`** **`[`***палачинка***`]`**...
- *(25 поена)* Активан ***сто*** је платно које се ствара са задатом палачинкарницом којој је сто придружен и он може бити у стању (*SLOBODAN, PORUCIO*) које може да се дохвати. Сто се исцртава као зелени круг док је у стању *SLOBODAN* и као наранџасти круг док је у стању *PORUCIO*. Приликом стварања, сто је у стању *SLOBODAN*. Сто циклично обавља радњу у зависности од стања у ком се он налази. У стању *SLOBODAN* сто чека док га палачинкарница не обавести да је пристигао гост, након чега он генерише нову поруџбину коју прослеђује палачинкарници, и прелази у стање *PORUCIO*. Поруџбина се генерише тако што се у њу додаје случајан број палачинки између 1 и 2, при чему се приликом додавања палачинки на случајан начин бирају премаз, број прилога од 0 до 2 и прилози које палачинка садржи. Додавање свих врста прилога треба да буде подједнако вероватно. Цене прилога се формирају случајно, у опсегу од 5 до 20 динара. У стању *PORUCIO* поставља се временски интервал инкремента спремања палачинке на случајну вредност између 50 и 100ms, а затим се циклично исцртава један проценат прогреса спремања, у виду црвеног кружног исечка, на свако истицање постављеног временског интервала. Када се прогрес исцрта до комплетног круга сто чека 2s, након чега обавештава палачинкарницу да је слободан и мења стање у *SLOBODAN*. Сто може да се покрене и трајно прекине свој рад.
- *(30 поена)* Активна ***палачинкарница*** је главни прозор апликације (са слике) који садржи столове у матричном распореду, списак поруџбина столова који су поручили палачинке и новчано стање касе. Могуће је покренути палачинкарницу кликом на одговарајуће дугме, при чему се креира одређен број столова (на основу задатих параметара) који се одмах покрећу. Могуће је трајно прекинути рад палачинкарнице кликом на одговарајуће дугме при чему се прекида рад свих њених столова. Палачинкарница циклично на сваких 1s обавештава неки слободан сто о пристизању госта. Уколико ниједан сто није слободан, палачинкарница чека док се не појави слободан сто. Палачинкарници може да се проследи задата поруџбина која се додаје у списак поруџбина. Палачинкарница може да се обавести да је задати сто слободан и тада се поруџбина коју је генерисао задати сто уклања из списка и повећава се новчано стање касе за цену поруџбине.



НАПОМЕНЕ:

а) Израда решења задатка траје **150** 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/8/.
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 palacinkarnica.*;

public class Main {

  public static void main(String args[]) {

    Prilog prilog1 = new Prilog(Prilog.Sastojak.PLAZMA, 30);
    Prilog prilog2 = new Prilog(Prilog.Sastojak.KIKIRIKI, 30);

    Palacinka palacinka1 = new Palacinka(Palacinka.Premaz.EUROKREM);
    Palacinka palacinka2 = new Palacinka(Palacinka.Premaz.EUROKREM);

    try {
      palacinka1.dodaj(prilog1);
      palacinka1.dodaj(prilog2);
      palacinka2.dodaj(prilog1);
    } catch (GSastojak e) {}

    Porudzbina porudzbina = new Porudzbina();

    porudzbina.dodaj(palacinka1);
    porudzbina.dodaj(palacinka2);

    System.out.println("Broj palacinki koje ste porucili je: " +
porudzbina.broj());
    System.out.println("Cena Vase porudzbine je: " + porudzbina.cena());

  }

}
```

```
// Primer izvršavanja
Broj palacinki koje ste porucili je: 2
Cena Vase porudzbine je: 290
```

```
public class Prilog {
    public enum Sastojak { PLAZMA, KIKIRIKI,
                           KOKOS, ORASI }
    private Sastojak s; private int c;
    public Prilog(Sastojak s, int c){
        this.s = s; this.c = c;
    }
    public Sastojak dohvSastojak() { return s;}
    public int dohvCenu() { return c; }
    public boolean equals(Object obj) {
      return obj instanceof Prilog &&
               s == ((Prilog)obj).s;}
}
```

```
public class Palacinka {
    public enum Premaz { EUROKREM, DZEM }
    private Premaz premaz;
    private List<Prilog> pril;
    public Palacinka(Premaz p) {
      this.premaz = p;
      pril = new ArrayList<>();
    }
    public void dodaj(Prilog p)throws
                                GSastojak{
      for(Prilog pr:pril) {
        if(pr.equals(p))throw new GSastojak ();
      }
      pril.add(p);
    }
    public int cena() {
      int c = 0;
      for(Prilog p:pril){c+=p.dohvCenu();}
      return c+=(premaz==Premaz.DZEM ? 70:100);
    }
    public String toString() {
      StringBuilder sb = new StringBuilder();
      sb.append(premaz);
      for(Prilog p : pril) {
        sb.append(", ");
        sb.append(p.dohvSastojak());
      }
      return sb.toString();}
}
```

```
public class GSastojak extends Exception {
    public String toString(){return "Greska"; }
}
```

```
public class Porudzbina {
  private static int posId = 0;
  private int id = ++posId;
  private List<Palacinka> pal;
  public Porudzbina(){pal =new ArrayList<>();}
  public void dodaj(Palacinka p) {pal.add(p);}
  public int broj(){ return pal.size(); }
  public int cena() {
    int c = 0;
    for(Palacinka p:pal) {c+=p.cena();}
    return c;
  }
  public void izbaciSve(){ pal.clear(); }
  public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("porudzbina_" + id + ": ");
    for(Palacinka p : pal) {
      sb.append("[").append(p).append("] ");
    }
    return sb.toString();
  }}
```

```
public class Sto extends Canvas implements Runnable
{
  private Palacinkarnica palacinkarnica;
  public enum Stanje { SLOBODAN, PORUCIO };
  private Stanje stanje; private Thread nit;
  private int progres; private Porudzbina por;
  private static Prilog.Sastojak[] prilozi =
  {Prilog.Sastojak.PLAZMA, Prilog.Sastojak.KOKOS,
   Prilog.Sastojak.ORASI, Prilog.Sastojak.KIKIRIKI};
  public Sto(Palacinkarnica p) {
    palacinkarnica = p; stanje = Stanje.SLOBODAN;
  }
  synchronized public Stanje dohvStanje() {
    return stanje;
  }
  public void paint(Graphics g) {
    if(stanje == Stanje.SLOBODAN)
      g.setColor(Color.GREEN);
    else if(stanje == Stanje.PORUCIO)
      g.setColor(Color.ORANGE);
    g.fillOval(0, 0, getWidth(), getHeight());
    if(stanje == Stanje.PORUCIO) {
      g.setColor(Color.RED);
      g.fillArc(0,0,getWidth(),getHeight(),
                                  0,360*progres/100);
    }
  }
  public Porudzbina dohvPor() { return por; }
  public void run() {
    try {
     while(!Thread.interrupted()) {
       switch(stanje) {
        case SLOBODAN:
          synchronized(this) {  wait(); }
          int broj = (int)(Math.random() * 2 + 1);
          por = new Porudzbina();
          for(int i = 0; i < broj; i++) {
            Palacinka.Premaz premaz=Math.random()<0.5
              ? Palacinka.Premaz.DZEM :
                Palacinka.Premaz.EUROKREM;
            Palacinka pal = new Palacinka(premaz);
            int brojPriloga = (int)(Math.random()*3);
            for(int j = 0; j < brojPriloga; j++) {
              while(true) {
                try {
                  int poz = (int)(Math.random() * 4);
                  pal.dodaj(new Prilog(prilozi[poz],
                      (int)(Math.random() * 15 + 5)));
                  break;
                } catch (GSastojak e) {}
              }
            }
            por.dodaj(pal);
          }
          palacinkarnica.prosledi(por);
          stanje = Stanje.PORUCIO; repaint(); break;
        case PORUCIO:
          int inkrement=(int)(Math.random()*50+50);
            while(true) {
              Thread.sleep(inkrement); repaint();
              if(progres++ == 100) break;
            }
            Thread.sleep(2000);
            palacinkarnica.obavesti(this);
            progres = 0;
            synchronized(this) {
              stanje = Stanje.SLOBODAN;
            }
            repaint();
          }
        }
      } catch(InterruptedException e) {}
    nit = null;
  }
  public void pokreni() {
    if(nit != null) nit.interrupt();
    nit = new Thread(this); nit.start();
    stanje = Stanje.SLOBODAN; repaint();
  }
  public void prekini() {
    if(nit != null) nit.interrupt();
  }
}
```

```
public class Palacinkarnica extends Frame implements
                                              Runnable {
  private int red, kol;
  private java.util.List<Sto> stolovi;
  private Thread nit; private List porudzbine;
  private Label labela; private int cena;
  private TextField m = new TextField("2"),
                    n = new TextField("2");
  private Panel panelStolovi;
  private Button pokreni, prekini;
  public Palacinkarnica() {
    super("Palacinkarnica");
    setSize(380, 550);
    popuniProzor();
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        prekini(); dispose();
      }
    });
    setVisible(true);
  }
  private void pokupiPodatke() {
    cena = 0;
    porudzbine.removeAll();
    red = Integer.parseInt(m.getText());
    kol = Integer.parseInt(n.getText());
    stolovi = new ArrayList<>();
    panelStolovi.removeAll();
    remove(panelStolovi);
    panelStolovi = new Panel();
    panelStolovi.setLayout(new GridLayout(red, kol,
                                          25, 15));
    for(int i = 0; i < red * kol; i++) {
      stolovi.add(new Sto(this));
      panelStolovi.add(stolovi.get(i));
    }
    add(panelStolovi);
    labela.setText("Kasa: 0");
    revalidate(); repaint();
  }
  private void popuniProzor() {
    panelStolovi = new Panel();
    porudzbine = new List(7);
    labela = new Label("Kasa: 0");
    labela.setFont(new Font("Arial", Font.BOLD, 22));
    labela.setAlignment(Label.LEFT);
    pokupiPodatke();
    Panel jug = new Panel(new BorderLayout());
    jug.setBackground(Color.GRAY);
    jug.add(new Label("Spisak porudzbina"),
                                BorderLayout.NORTH);
    jug.add(porudzbine);
    Panel ctrl = new Panel();
    ctrl.add(labela);
    Panel ctrlZapad =
           new Panel(new GridLayout(2, 3, 5, 5));
    ctrlZapad.add(new Label("Broj redova: "));
    ctrlZapad.add(m);
    ctrlZapad.add(pokreni=new Button("Pokreni"));
    ctrlZapad.add(new Label("Broj kolona: "));
    ctrlZapad.add(n);
    ctrlZapad.add(prekini=new Button("Prekini"));
    ctrl.add(ctrlZapad);
    pokreni.addActionListener(e -> {pokreni();});
    prekini.addActionListener(e -> {prekini();});
    jug.add(ctrl, BorderLayout.SOUTH);
    add(jug, BorderLayout.SOUTH);
  }
  synchronized public void pokreni() {
    prekini(); pokupiPodatke();
    nit = new Thread(this); nit.start();
    for(Sto s : stolovi) { s.pokreni(); }
  }
  synchronized public void prekini() {
    for(Sto s : stolovi) { s.prekini(); }
    if(nit != null) nit.interrupt();
    nit = null;
  }
  public void run() {
    try {
      while(!nit.isInterrupted()) {
        Thread.sleep(1000);
        boolean nadjen = false;
        for(Sto s : stolovi) {
          synchronized(s) {
            if(s.dohvStanje()
                        ==Sto.Stanje.SLOBODAN){
              nadjen = true; s.notify(); break;
            }
          }
          if(nadjen) break;
        }
        synchronized(this) {if(!nadjen) wait(); }
      }
    } catch(InterruptedException e) {}
    nit = null;
  }
  synchronized public void prosledi(Porudzbina p)
  { porudzbine.add(p.toString()); }
  synchronized public void obavesti(Sto s) {
    porudzbine.remove(s.dohvPor().toString());
    cena += s.dohvPor().cena();
    labela.setText("Kasa: " + cena); notify();
  }
  public static void main(String[] argv) {
    new Palacinkarnica();
  }
}
```