ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Одсеци ИР, ОС, ОФ Београд 10.06.2018.

Завршни колоквијум из Објектно оријентисаног програмирања II

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

- *(10 поена)* ***Град*** има задате назив и реалне координате x и y. Може да се одреди реално растојање до задатог града у километрима, као и да се дохвати назив града.
- *(10 поена)* ***Пут*** има почетни и крајњи град које повезује, реалну константну брзину (km/s) којом се по путу може возити и реалну цену путовања по путу. Сви подаци се задају приликом стварања и могу да се дохвате. Може да се одреди дужина пута (растојање) између градова које пут повезује.
- *(25 поена)* ***Путања*** садржи произвољан број надовезаних путева. Ствара се празна, са задатим називом који може да се дохвати. Пут се може додати у путању само уколико је крајњи град на последње додатом путу једнак почетном граду пута који се управо додаје или уколико на путањи не постоји ниједан пут. Повратна вредност додавања је индикатор успеха. Може да се одреди укупна дужина путање као сума дужине свих путева на путањи. Може да се одреди број градова на путањи, да се дохвати град са задатим индексом (индекси крећу од 0), да се дохвати пут за задати почетни град, као и да се одреди цена коју је неопходно платити за прелазак свих путева између задатог почетног и крајњег града (уколико пут између два града не постоји, резултат је највећа реална вредност). Може да се састави текстуални опис путање у облику **`Putanja:`** *назив*<*нови_ред*>**`Dužina:`** *укупна_дужина*<*нови_ред*>**`Stanice:`**<*нови_ред*>, а затим се у броју потребних редова исписују градови на путањи (један град по реду).
- *(5 поена)* ***Путник*** има задате почетни град из којег путује и одредишни град у који путује и који могу да се дохвате.
- *(25 поена) Активан* ***аутобус*** је простор за текст (текстуална површ). Ствара се са задатом путањом по којој се креће, задатим бројем путника које може да прими и задатим временом чекања у сваком граду у који долази на путањи. Аутобус може бити у једном од 3 стања – *NASTANICI*, *VOZI*, *KRAJ*. Ствара се у стању *NASTANICI*, при чему је позадина текстуалне површи обојена жутом бојом. Путник се може додати у аутобус уколико у аутобусу има места и уколико се аутобус налази у стању *NASTANICI* и то у почетном граду путника. Повратна вредност додавања је индикатор успеха. Може да се дохвати путања по којој се аутобус креће као и да се трајно заустави аутобус. Аутобус креће из почетног града на својој путањи тек када се напуни, а затим вози преко појединачних путева на својој путањи све док не стигне до крајњег града на путањи када се његов рад завршава и када прелази у стање *KRAJ*. Може се проверити да ли аутобус тренутно вози и да се дохвати тренутни број путника. Док аутобус вози по путу, позадина текстуалне површи обојена је зеленом бојом и налази се у стању *VOZI*, а док стоји у граду наранџастом бојом и налази се у стању *NASTANICI*. У стању *KRAJ* позадина текстуалне површи је црвене боје. Када аутобус стигне у град, најпре избаци све путнике којима је тај град одредишни град, а затим чека до поласка. Може да се састави текстуални опис аутобуса у облику *путања*<*нови_ред*>**`Broj putnika:`** *број_путника*/*капацитет*<*нови_ред*>, а затим у зависности од стања: **`Vozi:`** *назив_града_од* - *назив_града_до* или **`Stoji:`** *назив_града* или **`Stigao:`** *назив_града*.
- *(25 поена)* ***Превозник*** је главни прозор апликације (са слике) и може да садржи произвољан број аутобуса. Ствара се празан, након чега се аутобуси могу додавати један по један. Путник се може пријавити код превозника, избором почетног и одредишног града, при чему превозник распоређује путника у аутобус са најнижом ценом пута између почетног и крајњег града путника. Повратна вредност распоређивања је индикатор успеха. Успешним распоређивањем путника превозник остварује зараду једнаку цени пута увећаној за 10%. Може да се одреди укупан приход превозника. Може да се састави текстуални опис превозника у облику **`Prevoznik:`** *приход*.

(0 поена) Приложена је класа са главном функцијом која испитује основне функционалности пакета класа.

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. Дозвољено је коришћење библиотечких класа за збирке.

г) На располагању је приступ *Web* адреси: https://docs.oracle.com/javase/8/docs/api/. Није дозвољено имати поред себе друге материјале, нити уз себе имати електронске уређаје, без обзира да ли су укључени или искључени.

д) Резултати колоквијума биће објављени на *Web*-у на адреси: http://rti.etf.bg.ac.rs/rti/ir2oo2/index.html

```java
package main;
import putovanje.*;

public class Main {

    public static void main(String[] args) {
        Prevoznik p = new Prevoznik();
        Putanja putanja = new Putanja("Beograd-Nis");
        Grad bgd = new Grad("Beograd", 0, 0);
        Grad gradovi[] = new Grad[] {
                bgd,
                new Grad("M. Pozarevac", 50., 100.),
                new Grad("V. Plana", 150., 200.),
                new Grad("Cuprija", 150., 300.),
                new Grad("Aleksinac", 200., 400.),
                new Grad("Nis", 250., 500.)
                };
        double brzine[] = {50., 60., 60., 50., 70.},
                cene[] = {100., 150., 150., 100., 200.};

        for(int i=0; i<gradovi.length-1; i++) {
            Put put = new Put(gradovi[i], gradovi[i+1], brzine[i], cene[i]);
            putanja.dodaj(new Put(put.dohvGradOd(), put.dohvGradDo(), put.dohvBrzinu(), put.dohvCenu()));
        }


        Autobus a = new Autobus(putanja, 4, 1000);
        p.dodaj(a);

        putanja = new Putanja("Beograd-Novi Sad");
        gradovi = new Grad[] {
                bgd,
                new Grad("Dobanovci", -50., 0.),
                new Grad("Simanovci", -100., -50.),
                new Grad("Ruma", -200., -100.),
                new Grad("Novi Sad", -300., -250.)
                };
        brzine = new double[] {40., 70., 80., 70.};
        cene = new double[] {100., 200., 200., 150.};

        for(int i=0; i<gradovi.length-1; i++) {
            Put put = new Put(gradovi[i], gradovi[i+1], brzine[i], cene[i]);
            putanja.dodaj(put);
        }

        a = new Autobus(putanja, 5, 1500);
        p.dodaj(a);

        Putnik putnik = new Putnik(gradovi[0], gradovi[2]);
        Grad gradOd = putnik.dohvGradOd(), gradDo = putnik.dohvGradDo();
        p.prijavi(putnik);
    }
}
```

```
package putovanje;
import java.awt.geom.Point2D;

public class Grad {
    private String naziv;
    private Point2D.Double lokacija;

    public Grad(String n, double x, double y) {
        naziv = n;
        lokacija = new Point2D.Double(x, y);
    }
    public double rastojanje(Grad g) {
        return lokacija.distance(g.lokacija);
    }
    public String dohvNaziv() {
        return naziv;
    }
}
```

```
public class Put {
    private Grad gOd, gDo;
    private double brzina, cena;

    public Put(Grad poc, Grad kraj, double b,
                                    double c) {
        gOd = poc; gDo = kraj;
        brzina = b; cena = c;
    }
    public double duzina() {
        return gOd.rastojanje(gDo);
    }
    public Grad dohvGradOd() { return gOd; }
    public Grad dohvGradDo() { return gDo; }
    public double dohvBrzinu() {return brzina;}
    public double dohvCenu() { return cena; }
}
```

```
import java.util.*;

public class Putanja {
    private List<Put> putevi;
    private String naziv;

    public Putanja(String ime){
        naziv = ime; putevi = new ArrayList<>();
    }
    public Grad dohvGrad(int ind) {
        if(ind < 0 || ind > putevi.size())
            return null;
        return ind == putevi.size() ?
               putevi.get(ind-1).dohvGradDo() :
               putevi.get(ind).dohvGradOd();
    }
    public Put izGrada(Grad g) {
        for(Put p:putevi)
            if(p.dohvGradOd() == g) return p;
        return null;
    }
    public double cena(Grad god, Grad gdo) {
        double c = 0; int naPut = 0;
        for(Put p:putevi) {
            if(p.dohvGradOd() == god) naPut++;
            if(naPut>0) c += p.dohvCenu();
            if(p.dohvGradDo() == gdo) { naPut++;
                                        break; }
        }
        return naPut==2 ? c : Double.MAX_VALUE;
    }
    public boolean dodaj(Put p) {
        if(putevi.size()>0 &&
        putevi.get(putevi.size()-1).dohvGradDo()
        != p.dohvGradOd()) return false;
        putevi.add(p);
        return true;
    }
    public double duzina() {
        double d = 0;
        for(Put p:putevi) d+=p.duzina();
        return d;
    }
    public int brGradova() {
        return
        putevi.size()==0?0:putevi.size()+1;
    }
    public String dohvNaziv() { return naziv; }
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("Linija: ").append(naziv)
        .append("\n").append("Duzina: ")
        .append(String.format("%.2f", duzina()))
        .append("\n").append("Stanice:\n");
        for(int i=0; i<putevi.size(); i++) {
            sb.append(putevi.get(i).dohvGradOd()
            .dohvNaziv()).append("\n");
            if(i==putevi.size()-1)
            sb.append(putevi.get(i).dohvGradDo()
            .dohvNaziv()).append("\n");
        }
        return sb.toString();
    }
}
```

```
public class Putnik {
    private Grad gOd, gDo;

    public Putnik(Grad gradOd, Grad gradDo) {
        gOd = gradOd; gDo = gradDo;
    }
    public Grad dohvGradOd() { return gOd; }
    public Grad dohvGradDo() { return gDo; }
}
```

```
import java.awt.*;

public class Autobus extends TextArea
                        implements Runnable {
    private Putanja p;
    private Putnik putnici[];
    private int r, vremeStajanja;
    private Thread nit;
    private Grad tGrOd;
    private enum Stanje {NASTANICI, VOZI, KRAJ};
    private Stanje stanje = Stanje.NASTANICI;

    public Autobus(Putanja put, int n, int v) {
        super("", 0, 0, SCROLLBARS_NONE);
        p = pu; tGrOd = p.dohvGrad(0);
        putnici = new Putnik[n];
        vremeStajanja = v; setEditable(false);
        setBackground(Color.YELLOW);
        setText(toString());
        nit = new Thread(this); nit.start();
    }
    public synchronized boolean vozi() {
                    return stanje==Stanje.VOZI; }
    public synchronized int brPutnika() {
                                    return br; }
    public synchronized boolean dodaj(Putnik p){
        if(br >= putnici.length ||
        stanje==Stanje.VOZI ||
        stanje==Stanje.KRAJ ||
        (stanje==Stanje.NASTANICI &&
        tGrOd!=p.dohvGradOd())) return false;
        putnici[br++] = p; setText(toString());
        if(br == putnici.length) notify();
        return true;
    }
    private synchronized void izbaci(Putnik p) {
        for(int i=0;i<br;i++) {
            if(putnici[i] == p) {
                putnici[i] = putnici[--br];break;
            }
        }
    }
    public synchronized Putanja dohvPutanja()
                                        {return p;}
    public void run() {
    try {
        synchronized (this) {
            while(br < putnici.length) wait();
        }
        while(true) {
            stanje = Stanje.VOZI;
            setBackground(Color.GREEN);
            setText(toString());
            Put put = p.izGrada(tGrOd);
            if(put==null) break;
            Thread.sleep((int)(put.duzina()/
            put.dohvBrzinu()*1000));
            tGrOd = put.dohvGradDo();
            for(int i=0;i<br;) {
                if(putnici[i].dohvGradDo() ==
                tGrOd) izbaci(putnici[i]);
                else i++;
            }
            stanje = Stanje.NASTANICI;
            setBackground(Color.ORANGE);
            setText(toString());
            Thread.sleep(vremeStajanja);
        }
    } catch (InterruptedException e) {}
        stanje = Stanje.KRAJ;
        setBackground(Color.RED);
        setText(toString());
    }
    public void prekini() {
        if(nit!=null) nit.interrupt();
    }
    public synchronized String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(p).append("Broj putnika: ")
        .append(br).append("/")
        .append(putnici.length);
        if(stanje==Stanje.VOZI) {
            sb.append("\nVozi: ")
            .append(tGrOd.dohvNaziv())
            .append(" - ");
            if(p.izGrada(tGrOd)!=null)
                sb.append(p.izGrada(tGrOd)
                .dohvGradDo().dohvNaziv());
            else sb.append("");
        }
        else sb.append("\nStoji: ")
            .append(tGrOd.dohvNaziv());
        return sb.toString();
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.util.List;

public class Prevoznik extends Frame {
    private List<Autobus> autobusi = new
                                ArrayList<>();
    private Panel buseviPan = new Panel(new
                        GridLayout(1, 0, 5, 0));
    private Label info = new Label("",
                                Label.CENTER);
    private Choice gradOd = new Choice(),
                 gradDo = new Choice();
    private Map<String, Grad> gradovi = new
                                HashMap<>();
    private double prihod;

    public synchronized void dodaj(Autobus a) {
        autobusi.add(a);
        for(int i=0; i<a.dohvPutanja()
                            .brGradova(); i++) {
            Grad g = a.dohvPutanja().dohvGrad(i);
            if(gradovi.get(g.dohvNaziv())==null){
                gradovi.put(g.dohvNaziv(), g);
                gradOd.add(g.dohvNaziv());
                gradDo.add(g.dohvNaziv());
            }
        }
        buseviPan.add(a); revalidate();
    }
    public synchronized boolean prijavi(Putnik
                                        p) {
        double cena = Double.MAX_VALUE;
        Autobus bus = null;
        for(Autobus a:autobusi) {
            double c = a.dohvPutanja().cena
            (p.dohvGradOd(), p.dohvGradDo());
            if(c < cena) { bus = a; cena = c; }
        }
        if(bus == null) return false;
        bus.dodaj(p); prihod += cena*1.1;
        info.setText(toString());
        return true;
    }
    public synchronized double dohvPrihod()
                                {return prihod;}
    public Prevoznik() {
        super("Lasta");
        setBounds(800, 300, 400, 300);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent
                                            e) {
            for(Autobus a:autobusi) a.prekini();
            dispose();
        }});
        info.setText(toString());
        info.setBackground(Color.GRAY);
        add(info, BorderLayout.NORTH);
        add(buseviPan, BorderLayout.CENTER);
        Panel jug = new Panel();
        jug.add(gradOd); jug.add(gradDo);
        Button dodajDugme = new Button("Dodaj
                                    putnika");
        dodajDugme.addActionListener(
                        new ActionListener() {
            public void actionPerformed(
                                ActionEvent e) {
            prijavi(new Putnik(
            gradovi.get(gradOd.getSelectedItem())
            ,gradovi.get(gradDo.getSelectedItem()
            )));}
        });
        jug.add(dodajDugme);
        add(jug, BorderLayout.SOUTH);
        setResizable(false); setVisible(true);
    }
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("Prevoznik: ")
        .append(String.format("%.2f", prihod));
        return sb.toString();
    }
}
```