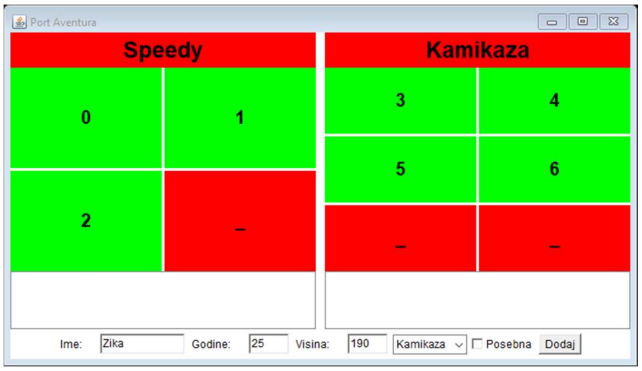
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек СИ Београд 10.06.2018.

Завршни колоквијум из Објектно оријентисаног програмирања II

1) *(укупно100 поена)* Саставити на језику *Java* следећи пакет класа:

- *(5 поена)* ***Посетилац*** има јединствени аутоматски генерисан и целобројни идентификатор, име, број година и целобројну висину који се задају приликом стварања. Сви параметри могу да се дохвате.
- *(25 поена)* Активна ***вожња*** се ствара са задатим називом, ценом вожње за једну особу, минималном висином и минималним бројем година посетиоца који смеју да се возе на датој вожњи, бројем врста и колона седишта на која је могуће сместити посетиоце и трајањем вожње. Сви параметри могу да се дохвате. Може да се провери да ли задати посетилац може да се вози на вожњи. Посетиоца је могуће сместити на прво слободно место при чему се попуњавање врши по редовима. У покушају додавања посетиоца у већ попуњену вожњу онај ко покушава да дода посетиоца се блокира док се не ослободи неко место. Након што се попуне сва места, вожња почиње и траје задато време. На крају вожње сви посетиоци се уклањају и вожња поново може да се попуњава. Могуће је проверити да ли је вожња у току. Вожњу је могуће трајно зауставити. Садржани панел приказује назив вожње и матрицу текстуалних ознака које представљају идентификаторе посетилаца на седиштима или слободна седишта (знак _), као што је приказано на слици. Заузето седиште има зелену позадину, а слободно црвену. Назив вожње има зелену позадину у току вожње, док у супротном има позадину црвене боје. Могуће је дохватити садржани панел са приказом вожње.
- *(10 поена)* ***Карта*** се ствара са задатим посетиоцем који ју је купио, вожњом на коју се односи и идентификатором да ли је посебна. Сви параметри могу да се дохвате. Могуће је дохватити цену карте која је једнака цени вожње за једну особу, при чему се цена код посебне карте увећава за 50%. Текстуални опис карте је `Karta[`*ид_посетиоца*`]`.
- *(10 поена)* ***Ред*** се састоји од произвољног броја карата. Ствара се празан након чега се карте додају једна по једна на крај реда. Могуће је узети карту са почетка реда. У случају да карта не постоји, блокира се онај који је потражује. Могуће је дохватити број карата у реду. Није дозвољено користити библиотечке класе за збирке података при реализацији реда.
- *(25 поена) Активан* ***радник*** се задаје именом и вожњом на којој ради и који могу да се дохвате. Приликом стварања радник ствара и ред у који се смештају карте за његову вожњу и који се може дохватити. Радња радника се састоји у дохватању карата из реда, провери да ли посетилац који је купио задату карту може да се вози на задатој вожњи и додавању таквог посетиоца у вожњу. Посетиоци који не испуњавају услове вожње се игноришу. Може се трајно зауставити рад радника. Може се дохватити укупна зарада радника као сума цена свих карата које је обрадио (узео из реда).
- *(25 поена)* ***Парк*** је главни прозор који се ствара са задатим низом радника који у њему раде. Кориснику је потребно приказати све вожње, као и дати могућност да створи нову карту са посетиоцем задатог имена, броја година и висине, као и са вожњом на којој посетилац жели да се вози. Вожња се бира из падајуће листе у којој су побројане све вожње које тренутно постоје у парку. Сматрати да на једној вожњи ради само један радник. При стварању карту је потребно сместити у ред радника одговарајуће вожње.



(0 поена) Приложена је класа са главном функцијом која испитује основне функционалности пакета класа уз исписивање резултата на стандардном излазу (конзоли).

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/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 lunapark.*;

public class Main {

    public static void main(String[] args) {
        Posetilac p = new Posetilac("Zika", 25, 190);
        System.out.println("ID: " + p.dohvId());
        System.out.println("Godine: " + p.dohvGodine());
        System.out.println("Visina: " + p.dohvVisina());

        Radnik []radnici = new Radnik[] { new Radnik("Pera", new Voznja("Speedy", 150, 120, 15, 2, 2, 5000)),
               new Radnik("Zile", new Voznja("Kamikaza", 200, 140, 18, 3, 2, 5000)) };

        Park park = new Park(radnici);
    }
}
```

```
package lunapark;

public class Posetilac {
    private static int sledId;
    private int id = sledId++;
    private String ime;
    private int godine, visina;

    public Posetilac(String ime, int g, int v) {
        this.ime = ime;
        godine = g;
        visina = v;
    }
    public int dohvId() { return id; }
    public String dohvIme() { return ime; }
    public int dohvGodine() { return godine; }
    public int dohvVisina() { return visina; }
}
```

```
import java.awt.*;

public class Voznja extends Thread {
    private double cena;
    private int minVisina, minGodine, trajanje;
    private String naziv;
    private int kap, n, brV, brK;
    private Posetilac posetioci[];
    private Label []labele;
    private Label nazivLab = new Label("",
                              Label.CENTER);
    private Panel panel = new Panel(new
                              BorderLayout());

    public Voznja(String naziv, double cena,
        int minVisina, int minGodine, int vrste,
                      int kolone, int trajanje) {
        this.cena = cena;
        this.minVisina = minVisina;
        this.minGodine = minGodine;
        this.naziv = naziv;
        brV = vrste;
        brK = kolone;
        this.trajanje = trajanje;
        this.kap = vrste * kolone;
        this.posetioci = new Posetilac [kap];
        this.labele = new Label [kap];
        nazivLab.setFont(new Font(null,
                              Font.BOLD, 25));
        nazivLab.setText(naziv);
        nazivLab.setBackground(Color.RED);
        panel.add(nazivLab, BorderLayout.NORTH);
        Panel p = new Panel(new
              GridLayout(vrste, kolone, 3, 3));
        for(int i=0; i<kap; i++) {
            labele[i] = new Label("_",
                              Label.CENTER);
            labele[i].setBackground(Color.RED);
            labele[i].setFont(new Font(null,
                              Font.BOLD, 20));
            p.add(labele[i]);
        }
        panel.add(p, BorderLayout.CENTER);
        start();
    }
    public boolean moze(Posetilacp) {
        return p.dohvGodine() >= minGodine &&
                    p.dohvVisina() >= minVisina;
    }
    public synchronized void dodaj(Posetilac p)
                    throws InterruptedException {
        while(n == kap) wait();
        posetioci[n] = p;
        labele[n].setText(p.dohvId()+"");
        labele[n++].setBackground(Color.GREEN);
        notify();
    }
    public synchronized boolean uToku() {
        return n == kap;
    }
    public void run() {
    try {
        while(!interrupted()) {
            synchronized (this) {
                while(n<kap) wait();
            }
            nazivLab.setBackground(Color.GREEN);
            sleep(trajanje);
            nazivLab.setBackground(Color.RED);
            synchronized (this) {
                n = 0;
                for(int i=0; i<kap; i++) {
                    labele[i].setText("_");
                    labele[i]
                    .setBackground(Color.RED);
                    }
                    notify();
                }
            }
        }catch(InterruptedExceptione) {}
        }
        public synchronized Panel dohvPanel() {
                                    return panel; }
        public String dohvNaziv() { return naziv; }
        public double dohvCenu() { return cena; }
        public double dohvMinVisinu() { return
                                    minVisina; }
        public double dohvMinGodine() { return
                                    minGodine; }
        public int dohvBrVrsta() { return brV; }
        public int dohvBrKolona() { return brK; }
        public double dohvTrajanje() { return
                                    trajanje; }
        public void prekini() { interrupt(); }
}
```

```
public class Karta {
    private Posetilac posetilac;
    private Voznja voznja;
    private boolean posebna;

    public Karta(Posetilac posetilac,
              Voznja voznja, boolean posebna) {
        this.posetilac = posetilac;
        this.voznja = voznja;
        this.posebna = posebna;
    }
    public Posetilac dohvPosetilac() {
        return posetilac;
    }
    public Voznja dohvVoznja() {return voznja;}
    public double dohvCena() {
        return voznja.dohvCenu() *
                                    (posebna?1.5:1);
    }
    public boolean jePosebna() {return posebna;}
    public String toString() {
        return "Karta [" +
                        posetilac.dohvId() + "]";
    }
}
```

```
public class Red extends java.awt.List {
    class Elem {
        Karta k;
        Elem sled;
        Elem(Karta k) { this.k = k; }
    }
    private int br;
    private Elem prvi, posl;

    public synchronized void dodaj(Karta k) {
        Elem novi = new Elem(k);
        if(prvi==null) prvi = novi;
        else posl.sled = novi;
        posl = novi;
        br++;
        this.add(k.toString());
        notify();
    }
    public synchronized Karta sledeca() throws
                            InterruptedException {
        while(broj()==0) wait();
        this.remove(0);
        Karta k = prvi.k;
        prvi = prvi.sled;
        br--;
        return k;
    }
    public synchronized int broj() {
        return br;
    }
}
```

```
public class Radnik extends Thread {
    private String ime;
    private Voznja voznja;
    private Red red = new Red();
    private double zarada;

    public Radnik(String ime, Voznja voznja) {
        this.ime = ime;
        this.voznja = voznja;
        start();
    }
    public void run() {
    try {
        while(!interrupted()) {
            synchronized (voznja) {
                while(voznja.uToku())
                    voznja.wait();
            }
            Karta k = red.sledeca();
            if(!voznja.moze(k.dohvPosetilac()))
                continue;
            zarada += k.dohvCena();
            voznja.dodaj(k.dohvPosetilac());
        }
    }catch(InterruptedExceptione) {}
    }
    public String dohvIme() { return ime; }
    public double dohvZarada() {return zarada;}
    public Voznja dohvVoznju() {return voznja;}
    public Red dohvRed() { return red; }
    public void prekini() { voznja.prekini();
                                interrupt(); }
}
```

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

public class Park extends Frame {
    private Radnik radnici[];
    private TextField ime = new TextField(10),
                godine = new TextField(3),
                visina = new TextField(3);
    private Checkbox prioritetna = new
                    Checkbox("Posebna", false);
    private Choice tip = new Choice();

    public Park(Radnik []rad) {
        super("Port Aventura");
        setBounds(400, 400, 700, 400);
        this.radnici = rad;
        Panel centar = new Panel(new
                    GridLayout(1, 0, 10, 0));
        for(Radnik r : rad) {
            Panel pom = new Panel(new
                                BorderLayout());
            pom.add(r.dohvVoznju()
            .dohvPanel(), BorderLayout.CENTER);
            pom.add(r.dohvRed(),
            BorderLayout.SOUTH);
            tip.add(r.dohvVoznju().dohvNaziv());
            centar.add(pom);
        }
        add(centar, BorderLayout.CENTER);
        Panel jug = new Panel();
        jug.add(new Label("Ime: "));
        jug.add(ime);
        jug.add(new Label("Godine: "));
        jug.add(godine);
        jug.add(new Label("Visina: "));
        jug.add(visina);
        jug.add(tip);jug.add(prioritetna);
        Button dodaj = new Button("Dodaj");
        dodaj.addActionListener(new
                            ActionListener() {
            public void actionPerformed(
                            ActionEvente) {
                Posetilac p = new Posetilac(
                ime.getText(),
                Integer.parseInt(godine.getText(),
                Integer.parseInt(visina.getText())
                );
                Karta k = new Karta(
                p,radnici[tip.getSelectedIndex()]
                .dohvVoznju(),
                prioritetna.getState());
                radnici[tip.getSelectedIndex()]
                .dohvRed().dodaj(k);
            }
        });
        jug.add(dodaj);
        add(jug, BorderLayout.SOUTH);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent
                                        we) {
            for(Radnik r : radnici) r.prekini();
            dispose();
            }
        });
        this.setVisible(true);
    }
}
```