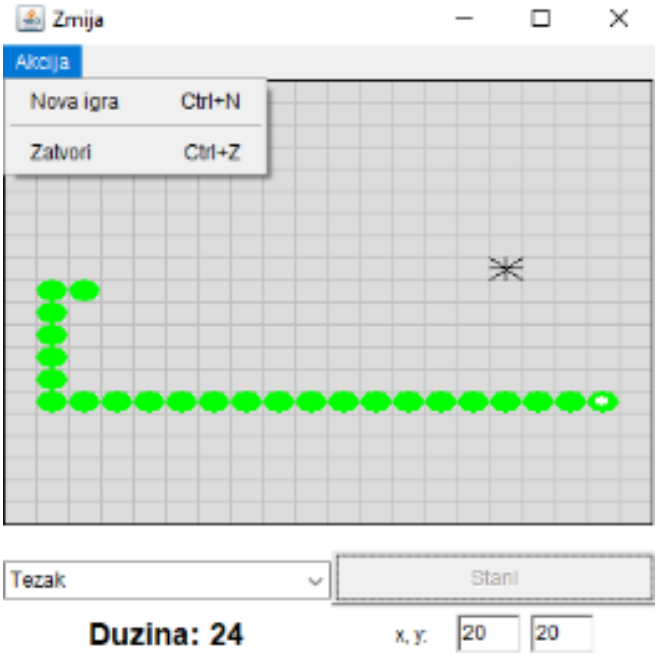
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсеци ИР, ОС, ОФ

Београд 09.06.2019.

Завршни колоквијум из Објектно оријентисаног програмирања II

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

- *(10 поена)* ***Позиција*** на правоугаоној табли се ствара са задатим целобројним индексом врсте и колоне, које је могуће дохватити. Могуће је померити позицију за једно место у задатом смеру (*LEVO, GORE, DESNO, DOLE*). Могуће је направити позицију која се налази у задатом смеру једно место поред текуће позиције. Две позиције су једнаке уколико се налазе у истој врсти и истој колони.
- *(10 поена)* ***Фигура*** се ствара са задатом позицијом на којој се налази њена глава и бојом. Фигуру је могуће исцртати на задатој табли. Могуће је дохватити позицију на којој се налази глава фигуре и поставити боју фигуре. Могуће је проверити да ли се фигура простире преко задате позиције. Предвидети померање фигуре у задатом смеру на задатој табли. Грешка је уколико померање није могуће.
- *(20 поена)* ***Змија*** је фигура која се простире на произвољном броју суседних позиција. Змија се приказује исцртавањем чланака, представљених зеленим попуњеним елипсама, на свакој од позиција преко којих се змија простире. Додатно, у центру позиције на којој се налази глава змије исцртава се и бела попуњена елипса пречника половине зелених елипси. Змија се може увећати додавањем чланка на реп. Могуће је дохватити дужину змије у броју позиција преко којих се простире. Змија се може померити у задатом смеру на задатој табли уколико је позиција на којој би се нашла глава змије у оквиру табле и на њој се не налази други део змије. При померању змије сваки њен чланак, осим главе, се помера на место претходног чланка.
- *(10 поена)* ***Мува*** је фигура која се исцртава црном бојом у виду звезде (видети слику). Померање муве је операција без ефекта.
- *(25 поена)* Активна ***табла*** је платно које се ствара са задатим бројем врста и колона. Број врста и колона је могуће дохватити и променити. Табла се састоји од једне змије и једне муве. На почетку игре змија је дужине 1, налази се у центру табле и креће се удесно. Мува се генерише на случајној позицији која није заузета. Могуће је проверити да ли је задата позиција на табли заузета. Позиција је заузета уколико се змија простире преко ње. Грешка је уколико је позиција ван опсега табле. Могуће је променити смер у ком се змија креће. Радња табле се састоји у померању змије у текућем смеру на постављени интервал времена. Интервал времена може да се постави. Уколико змију није могуће померити потребно је зауставити игру и променити боју змије на црвену. Уколико се након померања глава змије нађе на истој позицији као и мува потребно је увећати змију и изгенерисати нову муву. Активност табле је могуће зауставити и трајно прекинути.
- *(25 поена)* ***Игра*** је главни прозор апликације (са слике) који садржи једну таблу. Могуће је покренути нову игру са задатим димензијама табле. Интервал померања змије може бити лак (500ms), средњи (300ms) и тежак (100ms). Прозор приказује дужину змије. Смер кретања змије је могуће променити помоћу тастера на тастатури (стрелице лево, доле, десно, горе). Игру је могуће зауставити (помоћу дугмета) и затворити (из менија). У току игре није могуће променити интервал померања змије и димензије табле.



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 igra;
public class Pozicija {
    public enum Smer{LEVO, GORE, DESNO, DOLE};
    private int x, y;

    public Pozicija(int x, int y) {
        this.x = x; this.y = y;
    }
    public int x() { return x; }
    public int y() { return y; }
    public void pomeri(Smer smer) {
        switch (smer) {
        case LEVO: y-=1; break;
        case DESNO: y+=1; break;
        case DOLE: x+=1; break;
        case GORE: x-=1; break;
        }
    }
    public Pozicija pomereno(Smer smer) {
        Pozicija p = new Pozicija(x, y);
        p.pomeri(smer);
        return p;
    }
    @Override
    public boolean equals(Object obj) {
        if(!(obj instanceof Pozicija))
          return false;
        Pozicija p = (Pozicija)obj;
        return x == p.x && y == p.y;
    }
}
```

```java
public abstract class Figura {
    protected Pozicija p;
    protected Color boja;

    public Figura(Pozicija p, Color boja) {
        this.p = p; this.boja = boja;
    }
    protected abstract void crt(Graphics g, int w,
    int h);
    public void crtaj(Tabla t) {
         Graphics g = t.getGraphics();
         int w = t.getWidth() / t.x();
         int h = t.getHeight() / t.y();
         g.setColor(boja); crt(g, w, h);
    }
    public void postaviBoju(Color b) { this.boja = b;}
    public abstract boolean zauzimaPoziciju(Pozicija
    p);

    public abstract void pomeri(Smer smer, Tabla p)
    throws GNeMoze;
    public Pozicija pozicija() { return p; }
}
```

```java
public class Zmija extends Figura {
    private List<Pozicija> polja = new ArrayList<>();

    public Zmija(Pozicija p) {
        super(p, Color.GREEN); polja.add(p);
    }
    public void uvecaj() {
        polja.add(polja.get(polja.size()-1));
    }
    public int duzina() { return polja.size(); }
    @Override
    protected void crt(Graphics g, int w, int h) {
        for(Pozicija p:polja)
            g.fillOval(p.x()*w, p.y()*h, w, h);
        g.setColor(Color.WHITE);
        g.fillOval(p.x()*w+w/4, p.y()*h+h/4, w/2,h/2);
    }
    @Override
    public boolean zauzimaPoziciju(Pozicija p) {
        return polja.contains(p);
    }
    @Override
    public void pomeri(Smer smer, Tabla t) throws
GNeMoze {
        Pozicija pom = p.pomereno(smer);
        if(t.zauzetoPolje(pom) &&
!pom.equals(polja.get(polja.size()-1))) throw new GNeMoze();
        polja.remove(polja.size()-1); polja.add(0, p = pom);
    }
}
```

```java
public class Muva extends Figura {
    public Muva(Pozicija p) { super(p, Color.BLACK); }
    @Override
    protected void crt(Graphics g, int w, int h) {
        int x = p.x(); int y = p.y();
        g.drawLine(x*w, y*h, x*w+w, y*h+h);
        g.drawLine(x*w, y*h+h, x*w+w, y*h);
        g.drawLine(x*w+w/2, y*h, x*w+w/2, y*h+h);
        g.drawLine(x*w, y*h+h/2, x*w+w, y*h+h/2);
    }
    @Override
    public boolean zauzimaPoziciju(Pozicija p) {
        return this.p.equals(p);
    }
    @Override
    public void pomeri(Smer smer, Tabla p) throws GNeMoze {
        return;
    }
}
```

```java
public class Tabla extends Canvas implements Runnable{
    private int x, y, ms = 500;
    private Zmija zmija;
    private Muva muva;
    private Thread nit = new Thread(this);
    private boolean radi = false;
    private Label lduzina;
    private Smer smer, poslednji;
    private Igra igra;

    public Tabla(int x, int y) {
        this.x = x; this.y = y; nit.start();
    }
    public void postaviIgru(Igra igra) { this.igra = igra; }
    public void postaviLabelu(Label delova) {
        this.lduzina = delova;
    }
    private void azurirajLabele() {
        if(lduzina == null || zmija == null) return;
        lduzina.setText("Duzina: "+zmija.duzina());
    }
    private void napraviMuvu() {
        if(zmija.duzina() == x*y) { muva = null; return; }
        while(true) {
            int x = (int)(Math.random()*this.x);
            int y = (int)(Math.random()*this.y);
            Pozicija p = new Pozicija(x,y);
            try {
                if(!zauzetoPolje(p)) {
                    muva = new Muva(p); break;
                }
            } catch (GNeMoze e) {}
        }
    }
    public boolean zauzetoPolje(Pozicija p) throws GNeMoze {
        if(p.x() >= x || p.y() >= y ||
            p.x() < 0 || p.y() < 0) throw new GNeMoze();
        return zmija.zauzimaPoziciju(p);
    }
    private synchronized void azuriraj() {
        if(zmija == null) {
            zmija = new Zmija(new Pozicija(x/2,y/2));
            napraviMuvu();
            smer = poslednji = Smer.DESNO;
            return;
        }
        try {
            zmija.pomeri(smer, this);
            poslednji = smer;
            if(zmija.pozicija().equals(muva.pozicija())){
                napraviMuvu();
                zmija.uvecaj();
            }
        } catch (GNeMoze e) {
            zmija.postaviBoju(Color.RED);
            stani();
            igra.azuriraj(false);
        }
    }
    public int x() {return x;}
    public int y() {return y;}
    public synchronized void pomeri(Smer smer) {
        if(zmija == null) return;
        if(Math.abs(poslednji.ordinal()-smer.ordinal()) == 2)
            return;
        this.smer = smer;
    }
    public synchronized void postaviNivo(int ms) {
        this.ms = ms;
    }
    @Override
    public void run() {
        try {
            while(!Thread.interrupted()) {
                synchronized (this) {
                    while(!radi) wait();
                }
                Thread.sleep(ms);
                azuriraj(); azurirajLabele();
                repaint();
            }
        }catch (InterruptedException e) {}
    }
    public synchronized void kreni(int x, int y) {
        this.x = x; this.y = y; zmija = null; muva = null;
        azurirajLabele(); repaint();
        radi = true; notify();
    }
    public synchronized void stani() { radi = false; }
    public void zavrsi() { nit.interrupt(); }
    @Override
    public void paint(Graphics g) {
        int w = getWidth()/x;
        int h = getHeight()/y;
        g.setColor(new Color(220, 220, 220));
        g.fillRect(0, 0, w*x, h*y);
        g.setColor(Color.LIGHT_GRAY);
        for(int i=0;i<=x;i++)
            g.drawLine(w*i, 0, w*i, h*y);
        for(int i=0;i<=y;i++)
            g.drawLine(0, h*i, w*x, h*i);
        g.setColor(Color.BLACK);
        g.drawRect(0, 0, w*x, h*y);
        if(zmija != null) zmija.crtaj(this);
        if(muva != null) muva.crtaj(this);
    }
}
```

```java
public class Igra extends Frame{
    private static final int W = 20;
    private static final int H = 20;
    private Tabla tabla = new Tabla(W, H);
    private Label duzina = new Label("Duzina: 0");
    private TextField sirina = new TextField("20"),
                      visina = new TextField("20");
    private Button stani = new Button("Stani");
    private Choice nivo = new Choice();

    public Igra() {
        super("Zmija");
        setSize(400, 400);
        add(tabla, BorderLayout.CENTER);
        add(bottom(), BorderLayout.SOUTH);
        dodajMeni(); dodajOsluskivace();
        setVisible(true);
    }
    public void azuriraj(boolean traje) {
        stani.setEnabled(traje);
        sirina.setEnabled(!traje);
        visina.setEnabled(!traje);
        nivo.setEnabled(!traje);
    }
    private void dodajMeni() {
        MenuBar bar = new MenuBar();
        setMenuBar(bar);
        Menu menu = new Menu("Akcija");
        bar.add(menu);
        MenuItem novaIgra = new MenuItem("Nova igra",
                                new MenuShortcut('N'));
        menu.add(novaIgra);
        novaIgra.addActionListener(e->{
            tabla.kreni(
                Integer.parseInt(sirina.getText()),
                Integer.parseInt(visina.getText())
            );
            azuriraj(true);
        });
        menu.addSeparator();
        MenuItem zatvori = new MenuItem("Zatvori",
                                new MenuShortcut('Z'));
        menu.add(zatvori);
        zatvori.addActionListener(e->{tabla.zavrsi();
                                      dispose();});
    }
    private Panel bottom() {
        Panel p = new Panel(new GridLayout(2,2));
        duzina.setAlignment(Label.CENTER);
        duzina.setFont(new Font(null, Font.BOLD, 18));
        tabla.postaviLabelu(duzina);
        tabla.postaviIgru(this);
        stani.setEnabled(false);
        nivo.add("Lak");
        nivo.add("Srednji");
        nivo.add("Tezak");
        Panel xy = new Panel();
        xy.add(new Label("x, y:"));
        xy.add(sirina);
        xy.add(visina);
        p.add(nivo); p.add(stani); p.add(duzina);
        p.add(xy);
        return p;
    }
    private void dodajOsluskivace() {
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                tabla.zavrsi();
                dispose();
            }
        });
        stani.addActionListener(e->{
            tabla.stani();
            azuriraj(false);
        });
        nivo.addItemListener(e->{
            tabla.postaviNivo(new int[] {500, 300, 100}
                              [nivo.getSelectedIndex()]);
        });
        tabla.addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                switch(e.getKeyCode()) {
                case KeyEvent.VK_LEFT:
                    tabla.pomeri(Smer.LEVO); break;
                case KeyEvent.VK_RIGHT:
                    tabla.pomeri(Smer.DESNO); break;
                case KeyEvent.VK_DOWN:
                    tabla.pomeri(Smer.DOLE); break;
                case KeyEvent.VK_UP:
                    tabla.pomeri(Smer.GORE); break;
                }
            }
        });
    }
    public static void main(String[] args) {
        new Igra();
    }
}
```