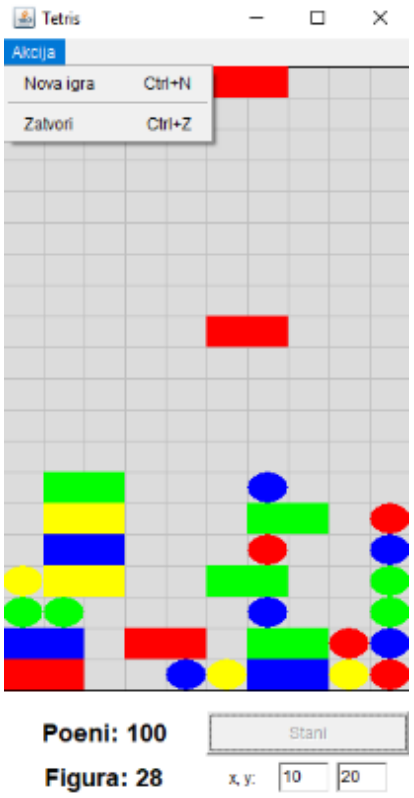
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек СИ

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 индексима врсте и колоне, које је могуће дохватити. Могуће је померити позицију за једно место у задатом смеру (*LEVO, DESNO, DOLE*). Могуће је направити позицију која се налази у задатом смеру једно место поред текуће позиције. Два позиције су једнаке уколико се налазе у истој врсти и истој колони.
- *(15 поена)* ***Фигура*** се ствара са задатом позицијом на којој се налази њен горњи леви угао и бојом. Фигуру је могуће исцртати на задатој табли. Могуће је дохватити позицију на којој се налази горњи леви угао фигуре. Могуће је проверити да ли се фигура простире преко задате позиције. Могуће је проверити да ли се фигура може померити у задатом смеру на задатој табли. Предвидети померање фигуре у задатом смеру на задатој табли. Грешка је уколико померање није могуће.
- *(10 поена)* ***Једноделни*** блок је фигура која се простире само на позицији на којој се налази њен горњи леви угао. Померање једноделног блока је могуће уколико је позиција на табли на којој би се блок након померања нашао слободна. Једноделни блок се исцртава као попуњена елипса задате боје на позицији на којој се блок налази.
- *(15 поена)* ***Дводелни*** блок је фигура која се простире на позицији на којој се налази њен горњи леви угао и на позицији десно. Померање дводелног блока је могуће уколико су позиције на табли на којима би се блок након померања нашао слободне (при чему блок сам себи не може сметати приликом померања). Дводелни блок се исцртава као правоугаоник који се простире преко позиција на којима се блок налази.
- *(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 произвољног броја фигура, које се додају појединачно. Могуће је проверити да ли је задата позиција на табли заузета. Позиција је заузета уколико се нека од фигура простире преко ње. Грешка је уколико је позиција ван опсега табле. У сваком тренутку је једна од фигура на табли покретна, а једна у приправности. Могуће је померити покретну фигуру у задатом смеру. Радња табле се састоји у померању покретне фигуре на доле на сваких 200ms. Уколико покретну фигуру није могуће померити на доле (позиција је заузета или је фигура достигла дно табле) потребно је све врсте табле које су у потпуности испуњене фигурама испразнити – уклонити све фигуре које се налазе у тим врстама, а затим све фигуре за које је то могуће померити на доле. За сваку уклоњену врсту играчу доделити 100 поена. Након тога фигура која је у приправности постаје покретна фигура, а генерише се нова фигура која је у приправности. Фигура која се генерише је блок случајне боје (црвена, зелена, плава, жута), налази се на врху табле у средини и једнака је вероватноћа да се састоји од једног или два дела. Активност табле је могуће зауставити и трајно прекинути.



- *(25 поена)* ***Тетрис*** је главни прозор апликације (са слике) који садржи једну таблу. Могуће је покренути нову игру са задатим димензијама табле. Прозор приказује број фигура који се тренутно налазе на табли као и укупно освојени број поена. Активна фигура на табли помера се помоћу тастера на тастатури (стрелице лево, доле, десно). Игру је могуће зауставити (помоћу дугмета) и затворити (из менија). У току игре није могуће променити 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

```
public class Pozicija {
    public enum Smer{ LEVO, DOLE, DESNO };
    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 s) {
        x += s.ordinal()-1;
        y += s.ordinal()%2;}
    public Pozicija pom(Smer s) {
        return new Pozicija(x+s.ordinal()
            -1, y+s.ordinal()%2);}
    public boolean equals(Object o) {
        if(!(o instanceof Pozicija))
            return false;
        Pozicija p = (Pozicija)o;
        return x == p.x && y == p.y;}
}
```

```
public abstract class Figura {
    protected Pozicija p;
    protected Color boja;
    public Figura(Pozicija po, Color b) {
        p = po; boja = b; }
    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 abstract boolean preko(Pozicija p);
    public abstract boolean moze(
        Smer smer, Tabla t);
    public void pomeri(Smer smer, Tabla p)
        throws GNeMoze{
        if(!moze(smer,p)) throw new GNeMoze();
        this.p.pomeri(smer);}
    public Pozicija poz() { return p; }
}
```

```
public class Jednodelni extends Figura {
    public Jednodelni(Pozicija p, Color b)
    { super(p, b); }
    public boolean preko(Pozicija po) {
        return p.equals(p);}
    public boolean moze(Smer s, Tabla t) {
        try { return !t.zauzeto(p.pom(s));
        } catch (GNeMoze e){return false;}
    }
    protected void crt(Graphics g,int w,int h)
    { g.fillOval(p.x()*w, p.y()*h, w, h);}
}
```

```
public class Dvodelni extends Figura {
    public Dvodelni(Pozicija p, Color b) {
        super(p, b);}
    public boolean preko(Pozicija p) {
        return this.p.equals(p) ||
        this.p.pom(Smer.DESNO).equals(p);
    }
    public boolean moze(Smer s, Tabla t) {
        try {if(s == Smer.DOLE) {
         return !t.zauzeto(p.pom(Smer.DOLE)) &&
            !t.zauzeto(
            p.pom(Smer.DOLE).pom(Smer.DESNO));
        }else { if(s == Smer.LEVO)
            return !t.zauzeto(p.pom(Smer.LEVO));
            else return
            !t.zauzeto(p.pom(Smer.DESNO)
            .pom(Smer.DESNO));}}
            catch (GNeMoze e) { return false; }
    }
    protected void crt(Graphics g, int w,
        int h){g.fillRect(p.x()*w,p.y()*h w*2,h);
    }
}
```

```
public class Tabla extends Canvas implements
Runnable{
    private int x, y, poeni = 0;
    private List<Figura> figure=new ArrayList<>();
    private Figura tek, sled;
    private Thread nit = new Thread(this);
    private Color[] boje = new Color[]{Color.RED,
        Color.BLUE,Color.YELLOW, Color.GREEN};
    private boolean radi;
    private Label lpoeni, ldelova;
    public Tabla(int x, int y) {
        this.x = x; this.y = y; nit.start();
    }
    public void postaviLabele(Label pts,
        Label br) {
        lpoeni = pts; ldelova = br;
    }
    private void azurirajLabele() {
        if(ldelova==null||lpoeni==null) return;
        lpoeni.setText("Poeni: "+poeni);
        ldelova.setText("Figura:
            "+(figure.size()+(tek==null?0:1)));
    }
    public void dodaj(Figura f) { figure.add(f); }
    public boolean zauzeto(Pozicija p)
            throws GNeMoze {
        if(p.x()>=x||p.y()>=y||p.x()<0||p.y()<0)
            throw new GNeMoze();
        for(Figura f:figure)
            if(f.preko(p))return true;
        if(tek.preko(p)) return true;
        return false;
    }
    private Figura sledeci() {
        Pozicija p = new Pozicija(x/2, 0);
        Color boja =
            boje[(int)(Math.random()*boje.length)];
        double rnd = Math.random();
        if(rnd<0.5)return new Jednodelni(p,boja);
        else return new Dvodelni(p, boja);
    }
    private synchronized void azuriraj() {
        if(tek == null) {tek = sledeci();
            sled = sledeci();return;}
        try { tek.pomeri(Smer.DOLE, this);
        } catch (GNeMoze e) {
            int r = tek.poz().y();
            boolean rowCompleted = true;
            for(int i=0;i<x;i++) {
                try {
                if(!zauzeto(new Pozicija(i, r))) {
                    rowCompleted = false;break;}
                } catch (GNeMoze e1) {}
            }
            if(rowCompleted) {
            figure.removeIf(x->x.poz().y()==r);
                poeni += 100;boolean imaJos = true;
                while(imaJos) {
                    imaJos = false;
                for(Figura f:figure) {
                    if(f.moze(Smer.DOLE, this)) {
                        imaJos = true;
                        try {
                            f.pomeri(Smer.DOLE, this);
                        } catch (GNeMoze e1) {}}}}
            }else figure.add(tek);
            tek = sled; sled = sledeci();}
    }
    public int x() {return x;}
    public int y() {return y;}
    public synchronized void pomeri(Smer smer) {
        if(tek == null) return;
        try {tek.pomeri(smer, this);}
        catch (GNeMoze e) {}
    }
    public void run() {
        try {
            while(!Thread.interrupted()) {
                synchronized (this) {
                    while(!radi) wait();}
                Thread.sleep(200); azuriraj();
                azurirajLabele(); repaint();}
        }catch (InterruptedException e) {}
    }
    public synchronized void kreni(int x,int y) {
        this.x = x;this.y = y;
        figure.clear();tek = sled = null;
        poeni = 0;azurirajLabele();
        repaint();radi = true; notify();
    }
    public synchronized void stani() {radi = false;}
    public void zavrsi() {nit.interrupt();}
    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);
        for(Figura f:figure) f.crtaj(this);
        if(tek != null) tek.crtaj(this);
        if(sled!=null&&!tek.poz().equals(sled.poz()))
            sled.crtaj(this);
    }
}
```

```
public class Tetris extends Frame{
    private static final int W = 10, H = 20;
    private Tabla tabla = new Tabla(W, H);
    private Label poeni = new Label("Poeni: 0");
    private Label delova = new Label("Figura: 0");
    private TextField sirina = new TextField("10"),
            visina = new TextField("20");
    private Button stani = new Button("Stani");
    public Tetris() {
        super("Tetris");
        setSize(317, 600); dodajMeni();
        add(tabla, BorderLayout.CENTER);
        add(bottom(), BorderLayout.SOUTH);
        dodajOsluskivace();setVisible(true);
    }
    private void azuriraj(boolean traje) {
        stani.setEnabled(traje);
        sirina.setEnabled(!traje);
        visina.setEnabled(!traje);
    }
    private void dodajMeni() {
        MenuBar bar = new MenuBar();
        Menu menu = new Menu("Akcija");
        setMenuBar(bar); 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));
        poeni.setAlignment(Label.CENTER);
        delova.setAlignment(Label.CENTER);
        poeni.setFont(new Font(
            null,Font.BOLD, 18));
        delova.setFont(new Font(
            null,Font.BOLD, 18));
        tabla.postaviLabele(poeni, delova);
        stani.setEnabled(false);
        stani.addActionListener(e->{
            tabla.stani();azuriraj(false);
        });
        Panel xy = new Panel();
        xy.add(new Label("x, y:"));
        xy.add(sirina);xy.add(visina);
        p.add(poeni);p.add(stani);
        p.add(delova);p.add(xy);return p;
    }
    private void dodajOsluskivace() {
        addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e)
            {tabla.zavrsi();dispose();}
        });
        tabla.addKeyListener(new KeyAdapter() {
            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;
                }}});
    }
    public static void main(String[] args) {
        new Tetris();
    }
}
```