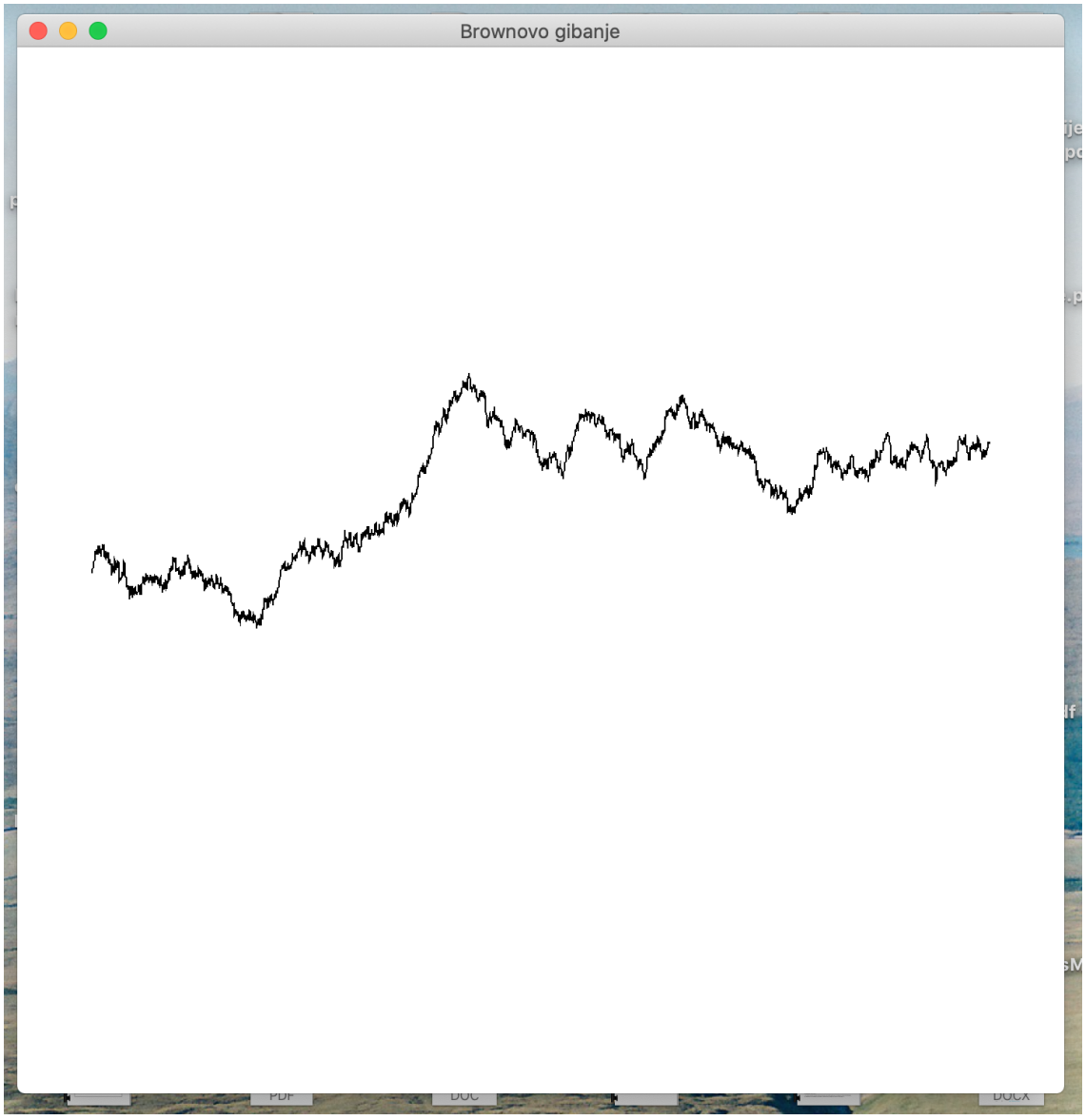
Risanje s Swingom

Brownovo gibanje



```
Random random = new Random ();

LinkedList<Double> pot = new LinkedList<Double>();
pot.addLast(0.0);
pot.addLast(random.nextGaussian());

final int DEPTH = 12;

for (int n = 0; n < DEPTH; n++) {
    ListIterator<Double> iterator = pot.listIterator();
    Double previous = iterator.next();
    Double next;
    while (iterator.hasNext()) {
        next = iterator.next();
        iterator.remove();
        Double between = (previous + next)/2.0 +
            random.nextGaussian()*Math.pow(2.0,-(n+2)/2.0);
        iterator.add(between);
        iterator.add(next);
        previous = next;
    }
}
```

Risanje poti

Uporabili bomo `Swing`, ki je API (application programming interface) za jezik Java.

- Ustvarili bomo okvir (frame) razreda `JFrame`.
- Okvir bo vseboval ploščo (panel) razreda `JPanel`, kjer lahko rišemo like.
- Imeli bomo razreda `Okno` in `Platno` za okvir in ploščo.

Na splošno lahko okviri vsebujejo mnoge plošče. Danes bomo imeli samo eno ploščo.

```
import javax.swing.*;


Okno okno = new Okno ();

okno.pack();
okno.setVisible(true);  // dokaži okno

okno.setPot(pot);  // dodamo pot na okno
```

`Okno` je posebni razred, ki ga pišemo za okno, kjer bomo narisali pot.

```
@SuppressWarnings("serial")
public class Okno extends JFrame {

    private Platno platno;

    public Okno() {
        // super() - Ni nam ga treba napisati!
        this.setTitle("Brownovo gibanje");
        platno = new Platno ();
        this.add(platno);
    }

    public void setPot(LinkedList<Double> pot) {
        platno.setPot(pot);
    }
}
```

```
public class Platno extends JPanel {

    private int numPoints;
    private int[] potX, potY;

    public Platno () {
        this.setPreferredSize(new Dimension(700,700));
        this.setBackground(Color.white);
        numPoints = 0;
        potX = new int[0]; potY = new int[0];
    }

    public void setPot(LinkedList<Double> pot) {
        numPoints = pot.size();
        potX = new int[numPoints]; potY = new int[numPoints];
        int xZero = 50; int yZero = 350;
        int i = 0;
        for (Double v : pot) {
            potX[i] = xZero + (int) Math.round(i * 600.0/(numPoints-1));
            potY[i] = yZero + (int) Math.round(v * 100);
            i++;
        }
    }

    @Override
    protected void paintComponent(Graphics g) { ... }
}
```

Konstruktori sta:

```
public Okno() {
    this.setTitle("Brownovo gibanje");
    platno = new Platno ();
    this.add(platno);
}
public Platno () {
    this.setPreferredSize(new Dimension(700,700));
    this.setBackground(Color.white);
    ...
}
```

Oba avtomatično izvajata `super()` na začetku.

`new Okno()` se ustvari nov okvir z naslovom "Brownovo gibanje". Nato se ustvari novo ploščo, ki jo doda okvirju.

`new Platno()` se ustvari novo ploščo z velikostjo 700×700 in belim ozadjem.

Da bi risali na platno, preglasimo metodo `paintComponent` iz razreda `JPanel`.

```
@Override
protected void paintComponent(Graphics g) {
    g.setColor(Color.BLACK);
    g.drawPolyline(potX, potY, numPoints);
    this.repaint();
}
```

Ta metoda uporablja metodo `drawPolyline` iz razreda `Graphics`. Potem `this.repaint()` ponovno nariše vse platno na ekran.

Metoda `paintComponent` je protektirana (protected) v razredu `Platno`, ker je delkarirana kot protektirana metoda v nadrazredu `JPanel`.

Objekt `g` razreda `Graphics` nam da Java.

Risanje likov

```
public abstract class Lik {
    protected double x, y, theta;
    protected Color barva;

    public Lik(double x, double y, double theta, Color barva) {
        this.x = x; this.y = y;
        this.theta = theta;
        this.barva = barva;
    }

    public void premakni(double dx, double dy) {
        this.x += dx;
        this.y += dy;
    }

    public void zavrti (double dtheta) {
        this.theta += dtheta;
    }

    public void setColor(Color barva) {
        this.barva = barva;
    }

    public abstract boolean vsebujeTocko(double x, double y);
    public abstract void narisiSe(Graphics g);
}
```

```
protected double x, y, theta;
protected Color barva;
```

Abstraktni razred `Lik` ima polja `x, y, theta, barva`, do katerih imajo dostop samo razred `Lik` in njegovi podrazredi (ter razredi v istem paketu).

```
Lik(double x, double y, double theta, Color barva) {
    this.x = x; this.y = y;
    this.theta = theta;
    this.barva = barva;
}
```

To je konstruktor razreda `Lik`. Ker je `Lik` abstraktni razred, lahko uporabljamo ta konstruktor samo v definicijah konkretnih razredov, ki razširjajo (extend) `Lik`.

Na levi strani ukaza `this.x = x` je `this.x` polje razreda, ki ga konstruiramo. Na desni strani je `x` parameter konstruktorja.

Na primer, če izvajamo konstruktor `Lik` s parametri `(1.0,-2.0,Math.PI,Color.BLACK)` (z uporabo ukaza `super` v definiciji konstruktorja neposrednega podrazreda, kot v naslednji prosojnici), ukaz `this.x = x` pomeni `this.x = 1.0`.

Tri ekvivalentne možnosti

```
public Lik(double x, double y, double theta, Color barva) {
    this.x = x; this.y = y;
    this.theta = theta;
    this.barva = barva;
}
```

```
public Lik(double u, double v, double phi, Color c) {
    this.x = u; this.y = v;
    this.theta = phi;
    this.barva = c;
}
```

```
public Lik(double u, double v, double phi, Color c) {
    x = u; y = v;
    theta = phi;
    barva = c;
}
```

```
public class Krog extends Lik {

    private double polmer;

    public Krog(double x, double y, Color barva, double polmer) {
        super(x, y, 0.0, barva);
        this.polmer = polmer;
    }

    @Override
    public boolean vsebujeTocko(double x, double y) {
        double dx = x - this.x; double dy = y - this.y;
        return Math.sqrt(dx*dx + dy*dy) <= polmer;
    }

    @Override
    public void narisiSe(Graphics g) {
        g.setColor(this.barva);
        g.fillOval((int) Math.round(x - polmer), (int) Math.round(y - polmer),
            (int) Math.round(2 * polmer), (int) Math.round(2 * polmer));
    }
}
```

```
public Krog(double x, double y, Color barva, double polmer) {
    super(x, y, 0.0, barva);
    this.polmer = polmer;
}
```

To je konstruktor razreda `Krog`. Parametra `x,y` sta x-y koordinati središča kroga s polmerjem `polmer`, ki je barven z barvo `barva`.

`super(x,y,0.0,barva)` izvaja konstruktor nadrazreda `Lik` s parametri `(x,y,0.0,barva)`. Ker imamo deklaracijo `Krog extends Lik`, vsak konstruktor razreda `Krog` se mora začeti z uporabo `super` kot prvi ukaz. (Če se ne začne z ukazom `super`, Java avtomatično izvaja privzeti konstruktor nadrazreda s parametri `()`, to je brez parametrov. To smo že videli prejšnji teden, ko smo napisali `MyUrn`, `MyStack` in `MyQueue` brez ekspličitnih konstruktorjev.)

Konstruktor `Krog` nima parametra `theta` za kot vrtenja, ker vrtenje nima vpliva na krog.

```
public class Kvadrat extends Lik {

    private double stranica;
    private double x0, y0, x1, y1, x2, y2, x3, y3; // Koordinate robov

    public Kvadrat(double x, double y, double theta,
            Color barva, double stranica) {
        super(x, y, theta, barva);
        this.stranica = stranica;
        izracunajRobe();
    }

    private void izracunajRobe () {
        double doRoba = stranica / Math.sqrt(2);
        x0 = x + doRoba * Math.cos(theta + Math.PI/4);
        y0 = y + doRoba * Math.sin(theta + Math.PI/4);
        x1 = x + doRoba * Math.cos(theta + 3*Math.PI/4);
        y1 = y + doRoba * Math.sin(theta + 3*Math.PI/4);
        x2 = x + doRoba * Math.cos(theta + 5*Math.PI/4);
        y2 = y + doRoba * Math.sin(theta + 5*Math.PI/4);
        x3 = x + doRoba * Math.cos(theta + 7*Math.PI/4);
        y3 = y + doRoba * Math.sin(theta + 7*Math.PI/4);
    }
...
```

```
...
    @Override
    public void premakni(double dx, double dy) {
        super.premakni(dx, dy);
        izracunajRobe();
    }

    @Override
    public void zavrti (double dtheta) {
        super.zavrti(dtheta);
        izracunajRobe();
    }

    @Override
    public boolean vsebujeTocko(double x, double y) {
        return (x1-x0)*(x-x0) + (y1-y0)*(y-y0) >= 0 &&
            (x2-x1)*(x-x1) + (y2-y1)*(y-y1) >= 0 &&
            (x3-x2)*(x-x2) + (y3-y2)*(y-y2) >= 0 &&
            (x0-x3)*(x-x3) + (y0-y3)*(y-y3) >= 0;
    }
...
```

```
...
    @Override
    public void narisiSe(Graphics g) {
        g.setColor(this.barva);
        int[] xPoints = {(int) Math.round(x0), (int) Math.round(x1),
                (int) Math.round(x2), (int) Math.round(x3)};
        int[] yPoints = {(int) Math.round(y0), (int) Math.round(y1),
                (int) Math.round(y2), (int) Math.round(y3)};
        g.fillPolygon(xPoints, yPoints, 4);
    }
}
```

```
public Kvadrat(double x, double y, double theta, Color barva, double stranica)
    super(x, y, theta, barva);
    this.stranica = stranica;
    izracunajRobe();
}
```

To je konstruktor razreda `Kvadrat`. Parametra `x,y` sta x-y koordinati središča kvadrata z dolžino stranice `stranica` in kot vrtenja `theta`.

`izracunajRobe()` izračuna x-y koordinate štirih robov in da vrednosti poljem `x0, y0, x1, y1, x2, y2, x3, y3`.

Kot moramo, preglasimo abstraktni metodi `vsebujeTocko` in `narisiSe` iz abstraktne razrede `Lik`.

V razredu `Kvadrat` preglasimo tudi konkretni metodi `premakni` in `zavrti`, na primer:

```
@Override
public void premakni(double dx, double dy) {
    super.premakni(dx, dy);
    izracunajRobe();
}
```

`super.premakni(dx, dy)` izvaja metodo `premakni` iz nadrazreda `Lik`. Nato moramo spremeniti koordinate robov, ker so se spremenili vrednosti polj `x,y`.

Uporaba Swinga

```
public class Okno extends JFrame {

    private Platno platno;

    public Okno() {
        this.setTitle("Moja slika");
        platno = new Platno ();
        this.add(platno);
    }

    public void addLik(Lik l) {
        platno.addLik(l);
    }
}
```

```
public class Platno extends JPanel {

    private LinkedList<Lik> liki;

    public Platno () {
        this.setPreferredSize(new Dimension(750,750));
        this.setBackground(Color.white);
        this.liki = new LinkedList<Lik>();
    }

    public void addLik(Lik l) {
        liki.addLast(l);
    }

    @Override
    protected void paintComponent(Graphics g) {
        for (Lik l : this.liki) {
            l.narisiSe(g);
        }
        this.repaint();
    }
}
```

Prvi poskus

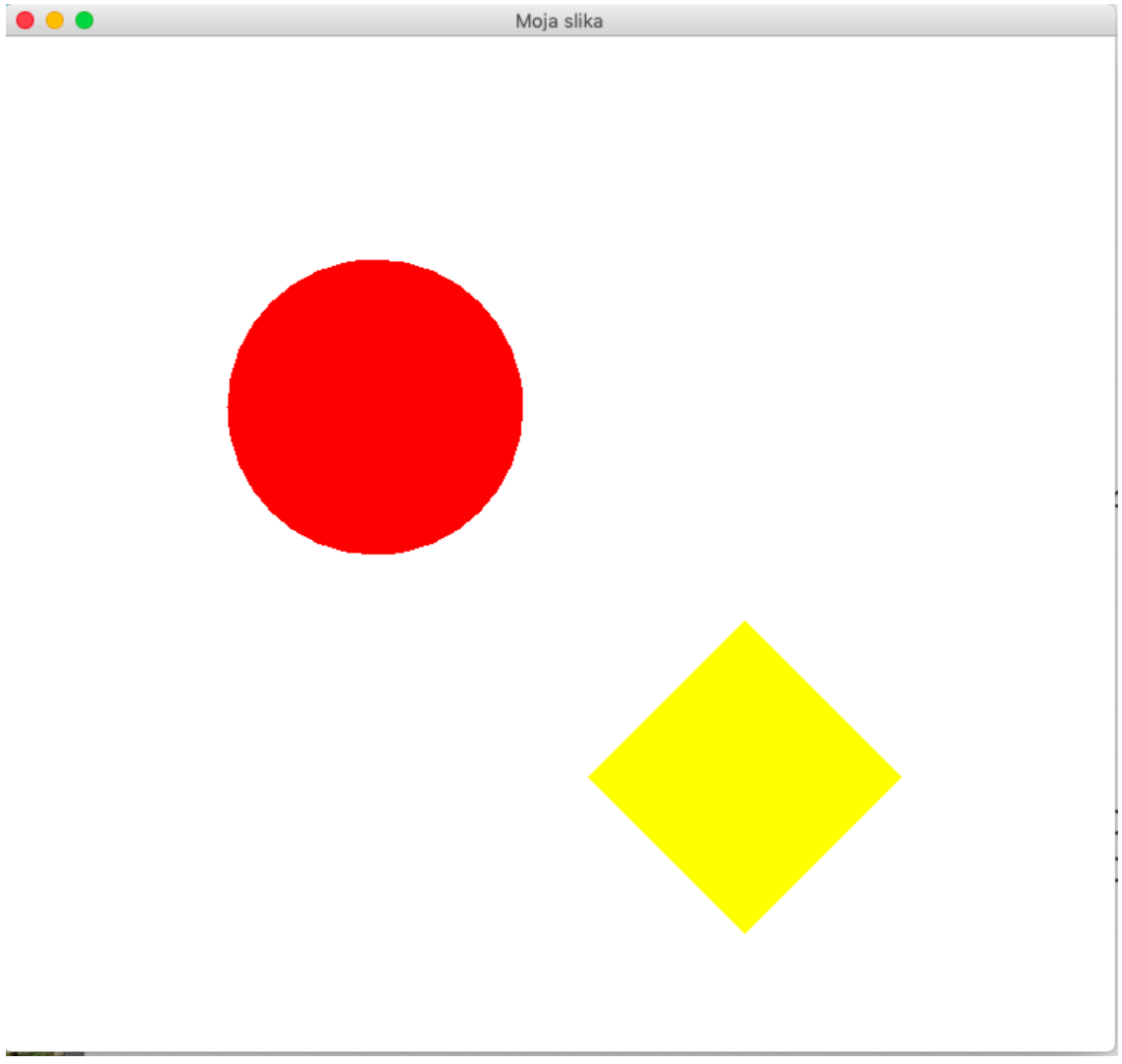
```
public class Slika {

    public static void main(String[] args) {

        Okno okno1 = new Okno ();

        okno1.pack();
        okno1.setVisible(true);

        okno1.addLik(new Krog(250.0, 250.0, Color.RED, 100.0));
        okno1.addLik(new Kvadrat(500.0, 500.0, Math.PI/4,
                        Color.YELLOW, 150.0));
    }
}
```

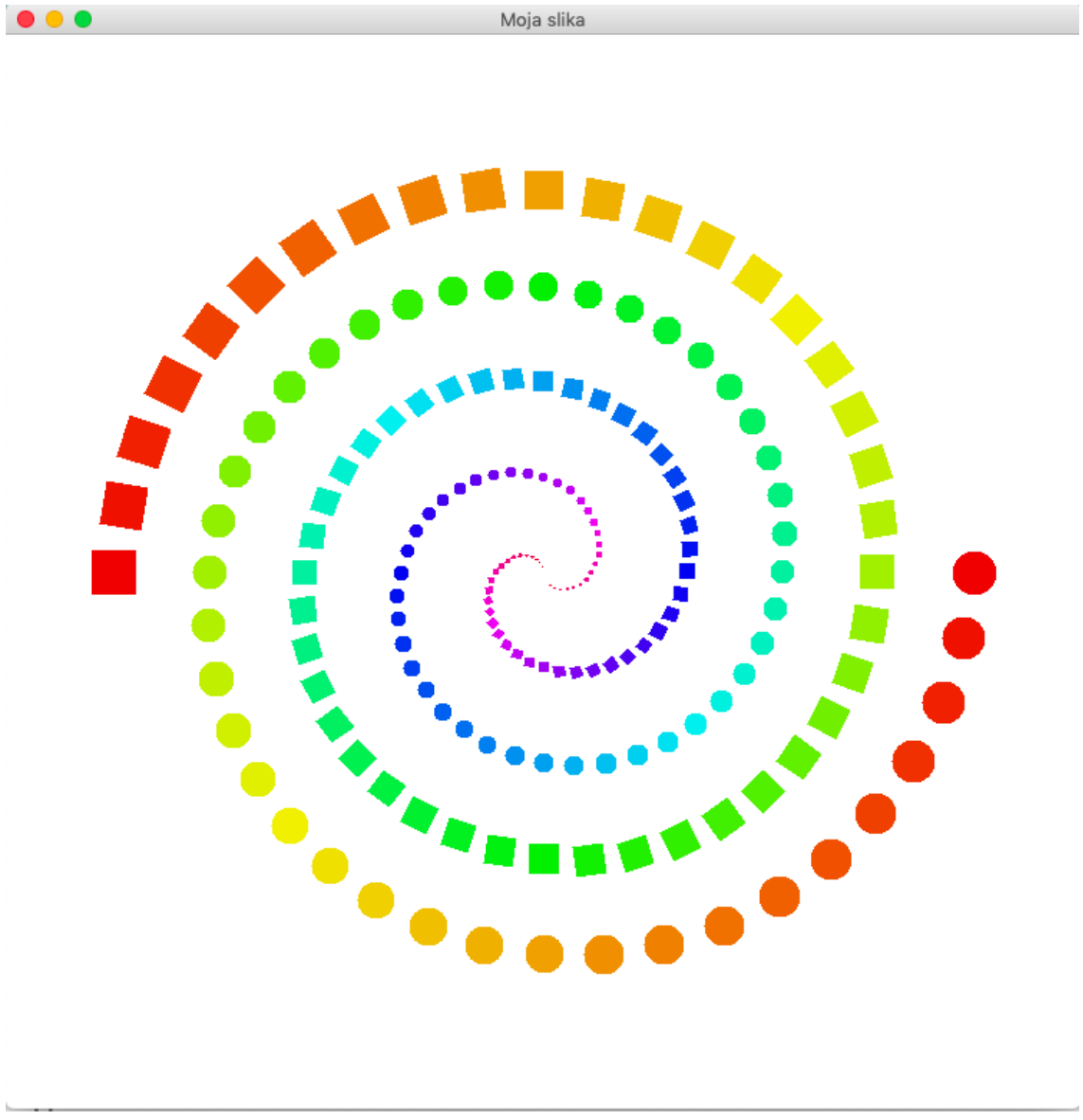

Moja slika

Drugi poskus

```
Okno okno1 = new Okno ();

okno1.pack();
okno1.setVisible(true);

for (int i = 0; i < 90; i++) {
    double theta = Math.PI * i / 20;
    double x = 375 + (300 * Math.cos(theta) * (90 - i) / 90);
    double y = 375 + (300 * Math.sin(theta) * (90 - i) / 90);
    int r = 0, g = 0 , b = 0;
    if (i < 15) {r = 240 ; g = 16 * i ; b = 0;}
    else if (i < 30) {r = 240 - 16 * (i - 15); g = 240 ; b = 0;}
    else if (i < 45) {r = 0; g = 240; b = 16 * (i - 30);}
    else if (i < 60) {r = 0; g = 240 - 16 * (i - 45); b = 240;}
    else if (i < 75) {r = 16 * (i - 60); g = 0 ; b = 240;}
    else if (i < 90) {r = 240 ; g = 0 ; b = 240 - 16 * (i - 75);}
    Color c = new Color(r, g, b);
    double size = (93-i)/3;
    okno1.addLik(new Krog(x, y, c, size/2));
    okno1.addLik(new Kvadrat(750 - x, 750 - y, theta, c, size));
}
```


Moja slika