Swing (2. del)
Mouse listeners

```
public class NKotnik extends Lik {

    private int n; // število stranic
    private double stranica; // dolžina stranice
    private double[] robiX, robiY; // Koordinate robov

    public NKotnik(int n, double x, double y, double theta,
            Color barva, double stranica) {
        super(x, y, theta, barva);
        this.stranica = stranica;
        this.n = n;
        robiX = new double[n]; robiY = new double[n];
        izracunajRobe();
    }

    private void izracunajRobe () {
        double doRoba = stranica / (2 * Math.sin(Math.PI/n));
        for (int i = 0; i < n ; i++) {
            robiX[i] = x + doRoba * Math.cos(theta + i * 2 * Math.PI / n);
            robiY[i] = y + doRoba * Math.sin(theta + i * 2 * Math.PI / n);
    }    }

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

```
    @Override
    public boolean vsebujeTocko(double x, double y) {
        for (int i = 0; i < n ; i++) {
            double x0 = robiX[i] ; double x1 = robiX[(i+1)%n];
            double y0 = robiY[i] ; double y1 = robiY[(i+1)%n];
            if ((y0-y1) * (x-x0) + (x1-x0) * (y-y0) < 0) return false;
        }
        return true;
    }

    @Override
    public void narisiSe(Graphics g) {
        g.setColor(this.barva);
        int[] xPoints = new int[n];
        int[] yPoints = new int[n];
        for (int i = 0; i < n ; i++) {
            xPoints[i] = (int) Math.round(robiX[i]);
            yPoints[i] = (int) Math.round(robiY[i]);
        }
        g.fillPolygon(xPoints, yPoints, n);
    }
}
```

Metode vmesnika `MouseListener` iz API

`void mouseEntered(MouseEvent e)`

Invoked when the mouse enters a component.

`void mouseExited(MouseEvent e)`

Invoked when the mouse exits a component.

`void mousePressed(MouseEvent e)`

Invoked when a mouse button has been pressed on a component.

`void mouseReleased(MouseEvent e)`

Invoked when a mouse button has been released on a component.

`void mouseClicked(MouseEvent e)`

Invoked when the mouse button has been clicked (pressed and released) on a component.

```
public class Platno extends JPanel implements MouseListener {

    private LinkedList<Lik> liki;

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

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

    public boolean removeLik(Lik l) { return liki.remove(l); }

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

```
    @Override
    public void mouseClicked(MouseEvent e) {
            System.out.println("Mouse clicked"); }

    @Override
    public void mousePressed(MouseEvent e) {
            System.out.println("Mouse pressed"); }

    @Override
    public void mouseReleased(MouseEvent e) {
            System.out.println("Mouse released"); }

    @Override
    public void mouseEntered(MouseEvent e) {
            System.out.println("Mouse entered"); }

    @Override
         public void mouseExited(MouseEvent e) {
            System.out.println("Mouse exited"); }
}
```

```
@Override
public void mouseClicked(MouseEvent e) {
    int x = e.getX(); int y = e.getY();
    Lik izbraniLik = null;
    for (Lik l: this.liki) {
        if (l.vsebujeTocko(x, y)) izbraniLik = l;
    }
    if (izbraniLik != null) this.removeLik(izbraniLik);
}

@Override
public void mousePressed(MouseEvent e) { }

@Override
public void mouseReleased(MouseEvent e) { }

@Override
public void mouseEntered(MouseEvent e) { }

@Override
public void mouseExited(MouseEvent e) { }
```

Metode vmesnika `MouseMotionListener` iz API

`void mouseDragged(MouseEvent e)`

Invoked when a mouse button is pressed on a component and then dragged.

`void mouseMoved(MouseEvent e)`

Invoked when the mouse cursor has been moved on a component but no buttons have been pushed.

```
public class Platno extends JPanel
        implements MouseListener, MouseMotionListener {

    ...

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

    ...

    @Override
    public void mouseDragged(MouseEvent e) {
        System.out.println("Mouse dragged"); }

    @Override
    public void mouseMoved(MouseEvent e) {
        System.out.println("Mouse moved"); }
}
```

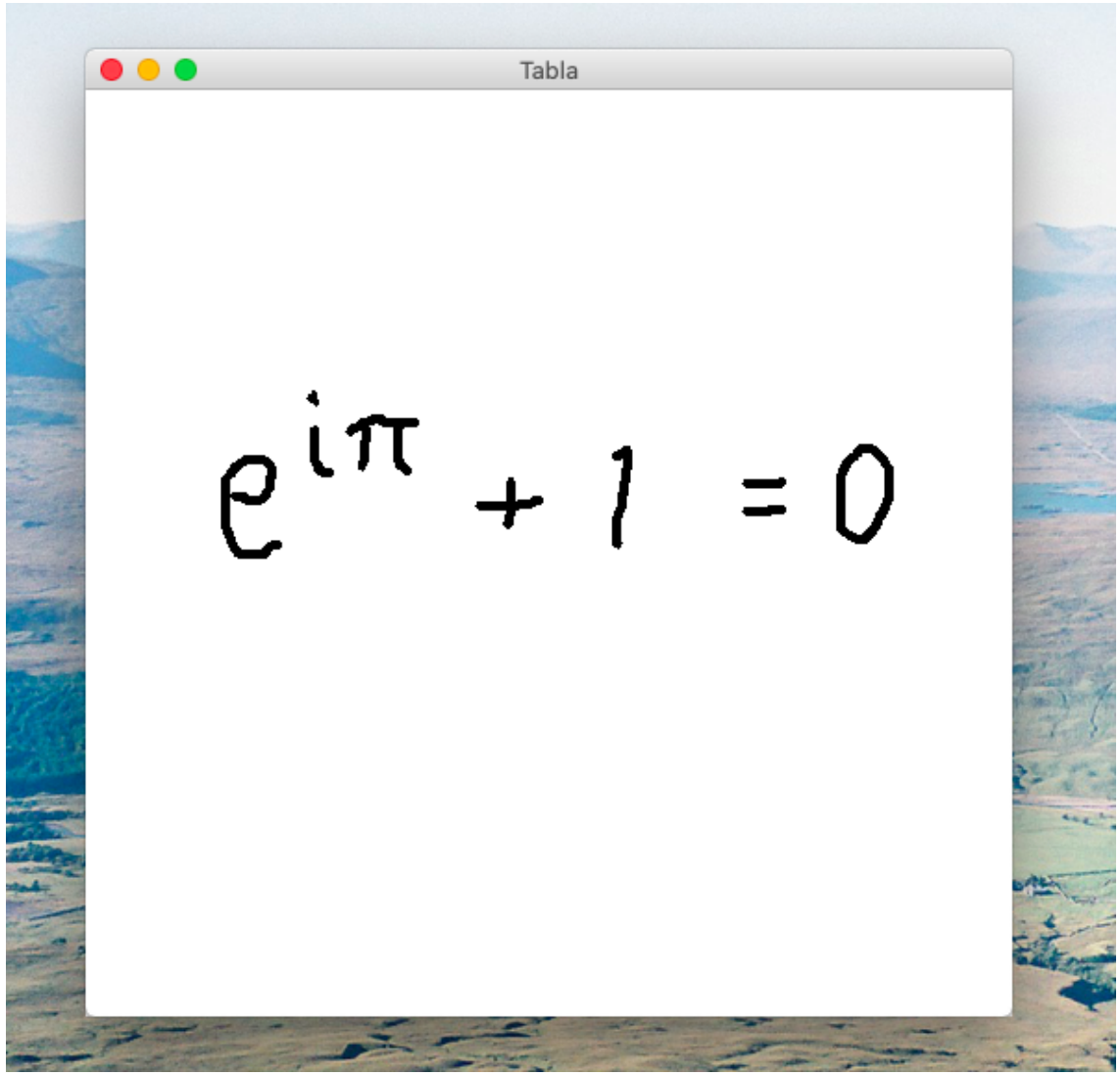
```
private int prejsnjiX, prejsnjiY;
private Lik premikajociLik;

@Override
public void mousePressed(MouseEvent e) {
    int x = e.getX(); int y = e.getY();
    Lik izbraniLik = null;
    for (Lik l: this.liki) {
        if (l.vsebujeTocko(x, y)) izbraniLik = l; }
    if (izbraniLik != null) {
        this.removeLik (izbraniLik);
        this.addLik (izbraniLik);
        premikajociLik = izbraniLik;
        prejsnjiX = x; prejsnjiY = y; }
}

@Override
public void mouseReleased(MouseEvent e) { premikajociLik = null; }

@Override
public void mouseDragged(MouseEvent e) {
    if (premikajociLik != null) {
        int x = e.getX(); int y = e.getY();
        premikajociLik.premakni(x - prejsnjiX, y - prejsnjiY);
        prejsnjiX = x; prejsnjiY = y; }
}
```

Aplikacija `Tabla`



```
public class Tabla {
    public static void main(String[] args) {
        Okno glavnoOkno = new Okno(500, 500);
        glavnoOkno.pack();
        glavnoOkno.setVisible(true);
    }
}


public class Okno extends JFrame {
    private Platno platno;

    public Okno(int width, int height) {
        setTitle("Tabla");

        // Platno, na katero rišemo
        platno = new Platno(width, height);
        this.add(platno);
    }
}
```

```
public class Platno extends JPanel
        implements MouseListener, MouseMotionListener {

    private List<Poteza> poteze;
    private Poteza aktivnaPoteza;

    public Platno(int sirina, int visina) {
        this.setBackground(Color.white);
        this.setPreferredSize(new Dimension(sirina, visina));
        this.addMouseListener(this);
        this.addMouseMotionListener(this);
        this.poteze = new LinkedList<Poteza>();
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D)g;
        for (Poteza p : poteze) {
            p.narisi(g2);
        }
        if (aktivnaPoteza != null) {
            aktivnaPoteza.narisi(g2);
        }
    }
```

```
    @Override
    public void mousePressed(MouseEvent e) {
        aktivnaPoteza = new Poteza(Color.BLACK,5,e.getX(),e.getY());
        this.repaint();
    }
    @Override
    public void mouseReleased(MouseEvent e) {
        aktivnaPoteza.dodajTocko(e.getX(), e.getY());
        poteze.add(aktivnaPoteza);
        aktivnaPoteza = null;
        this.repaint();
    }
    @Override
    public void mouseDragged(MouseEvent e) {
        aktivnaPoteza.dodajTocko(e.getX(), e.getY());
        this.repaint();
    }
    @Override
    public void mouseClicked(MouseEvent e) { }
    @Override
    public void mouseMoved(MouseEvent e) { }
    @Override
    public void mouseEntered(MouseEvent e) { }
    @Override
    public void mouseExited(MouseEvent e) { }
}
```

```
public class Poteza {
    private Color barva;
    private int sirina; // Širina, s katero rišemo potezo
    private Path2D pot;

    public Poteza(Color barva, int sirina, int x, int y) {
        this.barva = barva;
        this.sirina = sirina;
        this.pot = new Path2D.Float();
        pot.moveTo(x, y);
    }

    public void dodajTocko(int x, int y) {
        pot.lineTo(x, y);
    }

    public void narisi(Graphics2D g) {
        g.setColor(barva);
        g.setStroke(new BasicStroke(sirina,
                        BasicStroke.CAP_ROUND,
                        BasicStroke.JOIN_ROUND));
        g.draw(pot);
    }
}
```

Potrebno iz AWT (Abstract Windows Toolkit):

```
import java.awt.Color;
import java.awt.geom.Path2D;
import java.awt.Graphics2D;
import java.awt.BasicStroke;
```

Iz `BasicStroke`:

`static int CAP_ROUND`

> Ends unclosed subpaths and dash segments with a round decoration that has a radius equal to half of the width of the pen.

`static int JOIN_ROUND`

> Joins path segments by rounding off the corner at a radius of half the line width.

Statike metode in polja so direktno povezani z radredom ni z objekti v razredu. Na primer, za dostop do vrednosti polja `CAP_ROUND` napišemo `BasicStroke.CAP_ROUND`.