Osnove programiranja 2 v Javi

[Za čiste začetnike]

Nadaljevanje Osnov programiranja 1 v Javi.
Delala sem po zapiskih Kako Java.

[Za dodatno razlago ali kontakt sem na voljo na emailu:
anchyr@gmail.com]

[Zavedam se, da so lahko v mojih zapiskih napake in se že v naprej opravičujem ter pozivam vse, da mi te napake sporočijo. Do napake je lahko prišlo zaradi zatipkanja ali pa zaradi tega, ker ne obvladam Javo 100%]

Anchy R
[28.4.2007]

Kazalo:

Zbirke podatkovnih struktur

- **ogrodje zbirk (collection framework)** - zajema celotno strukturo razredov (skupaj z vmesniki in abstraktnimi razredi), ki določajo delo z zbirkami podatkovnih struktur. Vse zbirke omogočajo:

```
//m tipa ene strukture
m.add(arg0);            //dodajanje elementa v zbirko
m.remove(arg0);         //odstranjevanje elementa iz zbirke
m.contains(arg0);       //preverjanje, ali zbirka vsebuje dolocen element
m.isEmpty();            //preverjanje, ali je zbirka prazna
m.size();               //preverjanje velikosti zbirke
m.toArray();            //pretvorbo zbirke v polje elementov
```

- **seznami** – so zbirke elementov v znanem zaporedju (vsak element ima točno določeno mesto (indeks) v seznamu)

```
import java.util.*;                         //potrebujemo za delo s seznami

public class Seznam
{
    private static final int MAX = 15;

    public static void main(String[] args)
    {
        //deklariramo seznam ArrayList, v <> je tip elementa
        ArrayList<Integer> sez = new ArrayList<Integer>();
                                                //ustvarimo prazen seznam z
                                                //velikostjo 10 elementov
        int meja = (int) (Math.random()*MAX);

        for(int i=0; i<=meja; i++)
        {
            sez.add(i);                     //i dodamo na seznam (na konec)
        }

        izpisiSeznam(sez);          //klicemo metodo, ki nam izpise seznam

        if(!sez.isEmpty())          //ce seznam ni prazen
        {
            sez.remove(0);          //iz njega odstranimo prvi element
        }

        izpisiSeznam(sez);

        sez.add(0, 715);            //na prvo mesto damo 715
        izpisiSeznam(sez);

        sez.clear();                //pobrisemo vse elemente iz seznama
        izpisiSeznam(sez);
    }

    //metoda za izpis seznama (sami ustvarimo)
    public static void izpisiSeznam(ArrayList<Integer> s)
    {
      //lahko izpisujemo element za elementom, ampak zaradi novosti Jave
      // 5.0 nam ni potrebno
      //ce bi vseeno zeleli:
      //for(int i=0; i<s.size(); i++)
```

```
        //    System.out.printf(" %d ", s.get(i));
        //ali
        //   for(int i: s)
        //     System.out.print(s.get(i).toString() + " ");

          //najbolj enostaven izpis vseh elementov
          System.out.println("Seznam vsebuje naslednje elemente: " + s);

          //izpis stevila elementov
          System.out.println("Število elementov v seznamu: " + s.size());
          System.out.println();
    }
}
```

- **množice**

```
import java.util.*;

public class Mnozica
{
    public static void main(String[] args)
    {
      HashSet<String> m = new HashSet<String>();//potrebujemo, ce delamo z
                                    //mnozicami (HashSet - zgoscena tabela)
                                    //ali pa TreeSet - uravnoveseno drevo

      for(int i=0; i<args.length; i++)//sprehodimo se po argumentih
      {
          if(!m.add(args[i]))       //dodajamo elemente na seznam. V primeru
                                    //neuspesnega dodajanja, vrne
          {                         //false - to pomeni, da je element ze v
                                    //mnozici, zato izpisemo:
            System.out.println("Ponovna pojavitev besede <"+args[i]+">");
          }
      }
      //izpisemo, koliko je bilo razlicnih besed in katere
     System.out.println("Skupaj je bilo "+m.size()+" razlicnih besed: "+m);
    }
}
```

grafika

- **hitra ponovitev od prejšnjega semestra** – komentarji in razlaga v prejšnjih zapiskih
  - preprosto okno

```
import javax.swing.*;

public class Okno
{
    public static void main(String[] args)
    {
        JFrame okno = new JFrame("Moje okno");
        okno.setSize(200, 100);
        okno.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okno.setVisible(true);
    }
}
```

  - grafične komponente

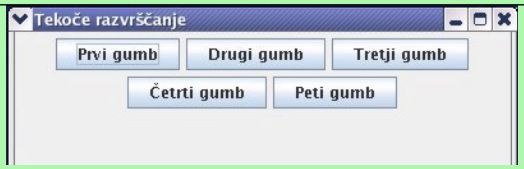
```
import javax.swing.*;
import java.awt.*;

public class Okno
{
    public static void main(String[] args)
    {
        Okvir okno = new Okvir();
        okno.setVisible(true);
    }
}

class Okvir extends JFrame
{
    private final int SIRINA = 200;
    private final int VISINA = 100;
    private String naslov = "Moje okno";
    private JLabel oznaka1 = new JLabel("Vpiši besedilo: ");
    private JLabel oznaka2 = new JLabel(" ..... ");
    private JTextField tekst = new JTextField(5);
    private JButton gumb1 = new JButton("Zbriši");
    private JButton gumb2 = new JButton("Potrdi");
    private JPanel ozadje = new JPanel();
    private Container vsebina = null;

    public Okvir()
    {
        setTitle(naslov);
        setSize(SIRINA, VISINA);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ozadje.add(oznaka1);
        ozadje.add(tekst);
        ozadje.add(oznaka2);
        ozadje.add(gumb1);
        ozadje.add(gumb2);
        vsebina = getContentPane();
        vsebina.add(ozadje);
    }
}
```

- razporejevalniki
  - robno razvrščanje (BorderLayout)

```
...
Container vsebina = getContentPane();
vsebina.setLayout(new BorderLayout());
vsebina.add(new JButton("Sever"), BorderLayout.NORTH);
vsebina.add(new JButton("Jug"), BorderLayout.SOUTH);
vsebina.add(new JButton("Vzhod"), BorderLayout.EAST);
vsebina.add(new JButton("Zahod"), BorderLayout.WEST);
vsebina.add(new JButton("Sredina"), BorderLayout.CENTER);
...
```

  - tekoče razvrščanje (FlowLayout)

```
...
Container vsebina = getContentPane();
vsebina.setLayout(new FlowLayout());
vsebina.add(new JButton("Prvi gumb"));
vsebina.add(new JButton("Drugi gumb"));
vsebina.add(new JButton("Tretji gumb"));
vsebina.add(new JButton("Četrti gumb"));
vsebina.add(new JButton("Peti gumb"));
...
```



  - razvrščanje v mrežo (GridLayout)

```
...
Container vsebina = getContentPane();
vsebina.setLayout(new GridLayout(2,3));
vsebina.add(new JButton("Prvi gumb"));
vsebina.add(new JButton("Drugi gumb"));
vsebina.add(new JButton("Tretji gumb"));
vsebina.add(new JButton("_etrti gumb"));
vsebina.add(new JButton("Peti gumb"));
...
```

- **dogodki in poslušalci**
  - poslusalec kot notranji razred

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

public class Okno
{
    public static void main(String[] args)
    {
         Okvir okno = new Okvir();
         okno.setVisible(true);
    }
}

class Okvir extends JFrame
{
    private final int SIRINA = 200;
    private final int VISINA = 150;
    private String naslov = "Moje okno";
    private JLabel oznaka1 = new JLabel("Vpiši besedilo: ");
    private JLabel oznaka2 = new JLabel(" ..... ");
    private JTextField tekst = new JTextField(5);
    private JButton gumb1 = new JButton("Zbriši");
    private JButton gumb2 = new JButton("Potrdi");
    private JPanel vnos = new JPanel();
    private JPanel izpis = new JPanel();
    private JPanel gumbi = new JPanel();
    private Container vsebina = null;
    public Okvir()
    {
        setTitle(naslov);
        setSize(SIRINA, VISINA);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        vnos.add(oznaka1);
        vnos.add(tekst);
        izpis.add(oznaka2);
        gumb1.addActionListener(new GumbPoslusalec());
        gumb2.addActionListener(new GumbPoslusalec());
        gumbi.add(gumb1);
        gumbi.add(gumb2);
        vsebina = getContentPane();
        vsebina.setLayout(new GridLayout(3,1,20,5));
        vsebina.add(vnos);
        vsebina.add(izpis);
        vsebina.add(gumbi);
    }

    private class GumbPoslusalec implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            Object izvor = e.getSource();
            if (izvor == gumb1)
                tekst.setText("");
            if (izvor == gumb2)
                oznaka2.setText(tekst.getText());
        }
    }
}
```

- poslušalec, ki ni kot notranji rezred

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

public class Okno
{
    public static void main(String[] args)
    {
         Okvir okno = new Okvir();
         okno.setVisible(true);
    }
}

class Okvir extends JFrame implements ActionListener
{
    private final int SIRINA = 200;
    private final int VISINA = 150;
    private String naslov = "Moje okno";
    private JLabel oznaka1 = new JLabel("Vpiši besedilo: ");
    private JLabel oznaka2 = new JLabel(" ..... ");
    private JTextField tekst = new JTextField(5);
    private JButton gumb1 = new JButton("Zbriši");
    private JButton gumb2 = new JButton("Potrdi");
    private JPanel vnos = new JPanel();
    private JPanel izpis = new JPanel();
    private JPanel gumbi = new JPanel();
    private Container vsebina = null;

    public Okvir()
    {
        setTitle(naslov);
        setSize(SIRINA, VISINA);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        vnos.add(oznaka1);
        vnos.add(tekst);
        izpis.add(oznaka2);
        gumb1.addActionListener(this);
        gumb2.addActionListener(this);
        gumbi.add(gumb1);
        gumbi.add(gumb2);
        vsebina = getContentPane();
        vsebina.setLayout(new GridLayout(3,1,20,5));
        vsebina.add(vnos);
        vsebina.add(izpis);
        vsebina.add(gumbi);
    }

    public void actionPerformed(ActionEvent e)
    {
        Object izvor = e.getSource();

        if (izvor == gumb1)
            tekst.setText("");
        if (izvor == gumb2)
            oznaka2.setText(tekst.getText());
    }
}
```

- poslušalec kot anonimni razred

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

public class Okno{
    public static void main(String[] args{
         Okvir okno = new Okvir();
         okno.setVisible(true);
    }
}

class Okvir extends JFrame implements ActionListener{
    private final int SIRINA = 200;
    private final int VISINA = 150;
    private String naslov = "Moje okno";
    private JLabel oznaka1 = new JLabel("Vpiši besedilo: ");
    private JLabel oznaka2 = new JLabel(" ..... ");
    private JTextField tekst = new JTextField(5);
    private JButton gumb1 = new JButton("Zbriši");
    private JButton gumb2 = new JButton("Potrdi");
    private JPanel vnos = new JPanel();
    private JPanel izpis = new JPanel();
    private JPanel gumbi = new JPanel();
    private Container vsebina = null;

    public Okvir(){
        setTitle(naslov);
        setSize(SIRINA, VISINA);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        vnos.add(oznaka1);
        vnos.add(tekst);
        izpis.add(oznaka2);
        gumb1.addActionListener(this);
        gumb2.addActionListener(this);

        gumb1.addActionListener(
            new ActionListener(){
               public void actionPerformed(ActionEvent e){
                     tekst.setText("");
               }
            }
        );
        gumb2.addActionListener(
            new ActionListener(){
               public void actionPerformed(ActionEvent e){
                     oznaka2.setText(tekst.getText());
               }
            }
        );
        gumbi.add(gumb1);
        gumbi.add(gumb2);
        vsebina = getContentPane();
        vsebina.setLayout(new GridLayout(3,1,20,5));
        vsebina.add(vnos);
        vsebina.add(izpis);
        vsebina.add(gumbi);
    }
}
```

- poslušalec za okna

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

public class Okno
{
    public static void main(String[] args)
    {
        Okvir okno = new Okvir();
        okno.setVisible(true);
    }
}

class Okvir extends JFrame
{
    private final int SIRINA = 200;
    private final int VISINA = 150;
    private String naslov = "Moje okno";

    public Okvir()
    {
        setTitle(naslov);
        setSize(SIRINA, VISINA);
        addWindowListener(new OknoPoslusalec());
    }
}

class OknoPoslusalec extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {
         System.exit(0);
    }
}
```

```
import javax.swing.*;
import java.awt.*;
public class Okno
{
    public static void main(String[] args)
    {
        Okvir okno = new Okvir();
        okno.setVisible(true);
    }
}
class Okvir extends JFrame
{
    private final int SIRINA = 200;
    private final int VISINA = 150;
    private String naslov = "Moje okno";

    public Okvir()
    {
        setTitle(naslov);
        setSize(SIRINA, VISINA);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

- prenos reference na objekt – ko imamo več poslušalcev, moramo paziti. Temu se lahko izognemo, če za okno uporabimo setDefaultCloseOperation(Jframe.EXIT_ON_CLOSE). Če pa ne, je spodaj en primer, kako se še lahko naredi:

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

public class Okno
{
    public static void main(String[] args)
    {
         Okvir okno = new Okvir();
         okno.setVisible(true);
    }
}

class Okvir extends JFrame
{
    private final int SIRINA = 200;
    private final int VISINA = 150;
    private String naslov = "Moje okno";
    private JLabel oznaka1 = new JLabel("Vpiši besedilo: ");
    private JLabel oznaka2 = new JLabel(" ..... ");
    private JTextField tekst = new JTextField(5);
    private JButton gumb1 = new JButton("Zbriši");
    private JButton gumb2 = new JButton("Potrdi");
    private JPanel vnos = new JPanel();
    private JPanel izpis = new JPanel();
    private JPanel gumbi = new JPanel();
    private Container vsebina = null;

    public Okvir()
    {
        setTitle(naslov);
        setSize(SIRINA, VISINA);
        addWindowListener(new OknoPoslusalec());
        vnos.add(oznaka1);
        vnos.add(tekst);
        izpis.add(oznaka2);
        //ker imamo dva poslusalca, moramo paziti na vidnost elementov
        //zato jih podamo zunanjemu razredu
        gumb1.addActionListener(new GumbPoslusalec(gumb1, gumb2, tekst,
oznaka2));
        gumb2.addActionListener(new GumbPoslusalec(gumb1, gumb2, tekst,
oznaka2));
        gumbi.add(gumb1);
        gumbi.add(gumb2);
        vsebina = getContentPane();
        vsebina.setLayout(new GridLayout(3,1,20,5));
        vsebina.add(vnos);
        vsebina.add(izpis);
        vsebina.add(gumbi);
    }
}
```

```
//znotraj tega razreda uredimo poslusalca za gumbe
class GumbPoslusalec implements ActionListener
{
    private JButton gumb1, gumb2;
    private JTextField tekst;
    private JLabel oznaka;

    public GumbPoslusalec(JButton g1, JButton g2, JTextField t,JLabel o)
    {
        this.gumb1 = g1;
        this.gumb2 = g2;
        this.tekst = t;
        this.oznaka = o;
    }

    public void actionPerformed(ActionEvent e)
    {
        Object izvor = e.getSource();

        if (izvor == gumb1)
            tekst.setText("");
        if (izvor == gumb2)
            oznaka.setText(tekst.getText());
    }
}

//razred poslusalca okna
class OknoPoslusalec extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {
         System.exit(0);
    }
}
```

- lastnosti sistema – npr. velikost ekrana

```
import java.awt.*;
import javax.swing.*;

class Okno1 extends JFrame
{
    private static final long serialVersionUID = 1L;

    public Okno1()
    {
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

   // podatki o velikosti zaslona
   Toolkit tk = Toolkit.getDefaultToolkit();
   Dimension zaslon = tk.getScreenSize();

   //izpisemo velikost zaslona
   System.out.printf("Velikost zaslona je %d x %d\n", zaslon.width,
zaslon.height);

   // inicializiramo sirino in visino
   int sirina = zaslon.width;
   int visina = zaslon.height;

   //lokacija okna
   setTitle("Centrirano okno");
   setSize(sirina / 4, visina / 4);
   setLocation(sirina / 4, visina / 4);
    }
}

public class Vaja
{
   public static void main(String[] args)
   {
       Okno1 okno = new Okno1();
       okno.setVisible(true);
   }
}
```

- risanje
  - risanje z grafičnimi primitivi – primer za sinx

```
import java.awt.*;
import javax.swing.*;

//razred za izris grafa od sinusa
public class Sinus extends JPanel
{
   private static final long serialVersionUID = 1L
   private final int SIRINA = 800;
   private final int VISINA = 400;
   private final int ROB = 10;

   public static void main(String[] args)
   {
         JFrame okno = new JFrame("Graf sin(x)");
         Sinus graf = new Sinus();     //ustvarimo nov graf
         okno.getContentPane().add(graf);//na delovno povrsino damo graf
         okno.setSize(graf.getSize().width, graf.getSize().height);
         okno.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         okno.setVisible(true);
   }

   public Sinus()
   {
         this.setSize(SIRINA, VISINA);  //nastavimo sirino in visino

   }

   //metoda, ki poskrbi, da se graf izrise
   public void paintComponent(Graphics g)
   {
         super.paintComponent(g);
         Dimension velikost = this.getSize();
         //dolocimo, kje je risanje dovoljeno
         g.setClip(ROB, ROB, velikost.width-2*ROB, velikost.height-
2*ROB);
         //kam postavimo koordinatni sistem
         g.translate(velikost.width/2, velikost.height/2);
         narisiOsi(g);     //klicemo metodo, ki narise osi
         narisiSinus(g);   //klicemo metodo, ki narise sinus
   }

   //metoda, ki narise osi
   private void narisiOsi(Graphics g)
   {
         Dimension velikost = this.getSize();
         int hMeja = velikost.width/2;
         int vMeja = velikost.height/2;
         int crtica = velikost.width/200;
         g.setColor(Color.black);
         g.drawLine(-hMeja, 0, hMeja, 0);    //narisemo linijo
         int korak = hMeja/6;    //korak, ki nam pomaga narisati osi

         for(int i=korak; i<=hMeja; i=i+korak)
         {
               //na vsak korak izrisemo linijo
               g.drawLine(0, vMeja, 0, -vMeja);
               //na vsak korak izrismo crtico
               g.drawLine(-i, crtica, -i, -crtica);
```

```
        }

        //se nekaj izrisa
        g.drawLine(0, vMeja, 0, -vMeja);
        g.drawLine(-crtica, -2*vMeja/3, crtica, -2*vMeja/3);
        g.drawLine(-crtica, 2*vMeja/3, crtica, 2*vMeja/3);
    }

    //metoda, ki nam narise sinus
    private void narisiSinus(Graphics g)
    {
        Dimension velikost = this.getSize();
        int hMeja = velikost.width/2;
        int vMeja = velikost.height/2;
        //razmerje med x in radiani (1/6 hMeja je PI/4)
        double razmerjeX = 3*Math.PI/(2*hMeja);
        //razmerje za y os (2/3 vMeja je 1)
        double razmerjeY = 2*vMeja/3;
        int korak = hMeja/6;
        g.setColor(Color.red);

        //postopno risemo sinue
        for(int x=-hMeja; x<=hMeja; x++)
        {
            int y = (int)(razmerjeY*Math.sin(x*razmerjeX));
            y = -y;     //obrnemo y koordinato (kartezicni sistem)
            g.fillOval(x-1, y-1, 5, 5);   //sirina krivulje
        }
    }
}
```

- bitne slike

```
import java.awt.*;
import javax.swing.*;

public class BitnaSlika extends JPanel
{
   private final int ROB = 10;   //rob slike
   private final int NASLOV = 34;//naslov
   private final int OKVIR = 8;  //okvir
   private int sirina;           //sirina
   private int visina;           //visina
   private Image bitna = null;

   //main metoda
   public static void main(String[] args)
   {
         JFrame okno = new JFrame("Prikaz bitne slike");//naslov okna
         BitnaSlika s = new BitnaSlika();    //ustvarimo novo bitno sliko
         okno.getContentPane().add(s);       //jo dodamo na povrsino
         //nastavimo velikosti
         okno.setSize(s.sirina + s.OKVIR, s.visina + s.NASLOV);
         okno.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         okno.setVisible(true);
   }

   //konstruktor
   public BitnaSlika()
   {
         int i = 1;
         //potrebujemo ustrezen izvod razreda, da lahko uporabimo
         //getImage()
         Toolkit slika = Toolkit.getDefaultToolkit();
         //preberemo sliko in jo nalozimo v pomnilnik
         Image bitna = slika.getImage("slika1.jpg");

         //prazna zanka, ki se izvaja tako dolgo, dokler ne dobimo
         //podatke o visini in sirini (ko getWidth in getHeith dobita
         //pozitivni vrednosti
         while(((bitna.getWidth(this)<0) ||
                     (bitna.getHeight(this)<0)) && (i++<10000000))
               ;

         //poskrbimo, da se visina in sirina prilegata prebrani sliki
         this.sirina = bitna.getWidth(this) + 2*ROB;
         this.visina = bitna.getHeight(this) + 2*ROB;
   }

   //metoda, ki poskrbi za izris bitne slike
   public void paintComponent(Graphics g)
   {
         super.paintComponent(g);

         if(bitna != null)                        //ce slika obstaja
         {
               g.drawImage(bitna, ROB, ROB, this); //jo narisemo
         }
   }
}
```

- risanje na bitno sliko v ozadju

```
import java.awt.*;
import javax.swing.*;      //ima vgrajeno podporo za double buffering
import java.awt.event.*;
import java.awt.image.*;

//program je napisan tako, da se risanje ne izvaja neposredno na risalno
//povrsino, ampak na sliko risalne povrsine v pomnilniku. Po koncu
//risanja bomo pokazali sliko na zaslon tak postopek se imenuje dvojno
//polnenje (double buffering)
public class Krogi
{
   public static void main(String[] args)
   {
         Okvir11 okno = new Okvir11();
         okno.setVisible(true);
   }
}

class Okvir11 extends JFrame
{
   private static final long serialVersionUID = 1L;
   private final int SIRINA = 600;
   private final int VISINA = 400;
   private String naslov = "Risanje krogov z miško";
   private Container vsebina = null;
   private Risalnica platno = new Risalnica();    //platno (RIsalnica)
   private JButton brisi = new JButton("Brisi");

   //konstruktor
   public Okvir11()
   {
         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         this.setTitle(naslov);              //nastavimo naslov
         this.setSize(SIRINA, VISINA);       //nastavimo visino in sirino
         this.setResizable(false);           //da ne moremo povecati okna
         this.brisi.setBackground(Color.orange);  //ozadje gumba
         this.brisi.setForeground(Color.black);   //pred ozadje (barva
                                                  //pisave)

         //gumbu brisi dodamo poslusalca
         this.brisi.addActionListener
         (//navaden oklepaj!!!!!!!
               new ActionListener()
               {
                     //metoda, ki se izvede, ko je gumb pritisnjen
                     public void actionPerformed(ActionEvent d)
                     {
                           platno.brisiRisalnico();//pobrisemo risalnico
                           platno.repaint(); //repaintamo (karkoli
                                             //narisemo, zbrisemo, moramo
                                             //ponovno narisati
                     }                       //da se poznajo spremembe)
               }
         );//podpicje

         //platnu dodamo poslusalca miskinih klikov
         this.platno.addMouseListener
         (//navaden oklepaj!!!!!!!
               new MouseAdapter()
               {
```

```
                        //metoda, ki se zgodi, ko pritisnemo na misko
                        public void mouseClicked(MouseEvent e)
                        {
                              //klicemo metodo, ki narise krog (dobimo
                              //koordinate x in y)
                              platno.narisiKrog(e.getX(),e.getY());
                              platno.repaint();       //repaintamo
                        }
                  }
            );//podpicje!!!!!!!!!

            this.vsebina = this.getContentPane();
            this.vsebina.add(brisi, BorderLayout.SOUTH);
            this.vsebina.add(platno, BorderLayout.CENTER);
      }
}

//razred risalnica
class Risalnica extends JPanel
{
      private static final long serialVersionUID = 1L;     //Eclipse
      //barve, ki jih bomo potrebovali za izris krogov
      private Color[] barve = {Color.yellow, Color.orange, Color.red,
      Color.black, Color.blue, Color.green, Color.cyan, Color.magenta};

      //spremenljivka predstavlja sliko risalne povrsine v pomnilniku
      private BufferedImage drugaSlika = null;

      public void paintComponent(Graphics g)
      {
            super.paintComponent(g);

            if(drugaSlika != null)  //ce obstaja risalna povrsina
            {
                  g.drawImage(drugaSlika, 0, 0, this);//narisemo sliko
            }
      }

      //metoda, ki narise krog
      public void narisiKrog(int x, int y)
      {
          //ce druga slika ne obstaja, se najprej ustvari nova, enake
          //velikosti kot risalna povrsina
          if(drugaSlika == null)
          {
            drugaSlika = new BufferedImage(this.getSize().width,
                  this.getSize().height, BufferedImage.TYPE_4BYTE_ABGR);
          }

          //dobimo referenco na objekt Graphics (pomaga, da narisemo sliko)
          Graphics g = drugaSlika.getGraphics();
          //barvo izberemo nakljucno
          g.setColor(barve[(int)(Math.random()*barve.length)]);

          //izrisemo 5 krogov s srediscem v (x,y)
          for(int r=5; r<26; r=r+5)
          {
            //ce damo tukaj izbiro barv, se vsak krog obarva drugace ;-)
            //g.setColor(barve[(int)(Math.random()*barve.length)]);
            g.drawOval(x-r, y-r, 2*r, 2*r);     //izris kroga
          }
```

```
    }

    //metoda, ki pobrise risalnico (risalno povrsino postavimo na null)
    public void brisiRisalnico()
    {
        drugaSlika = null;
    }
}
```

- apleti
  - okno ter metode, ki lahko nastopajo v apletu

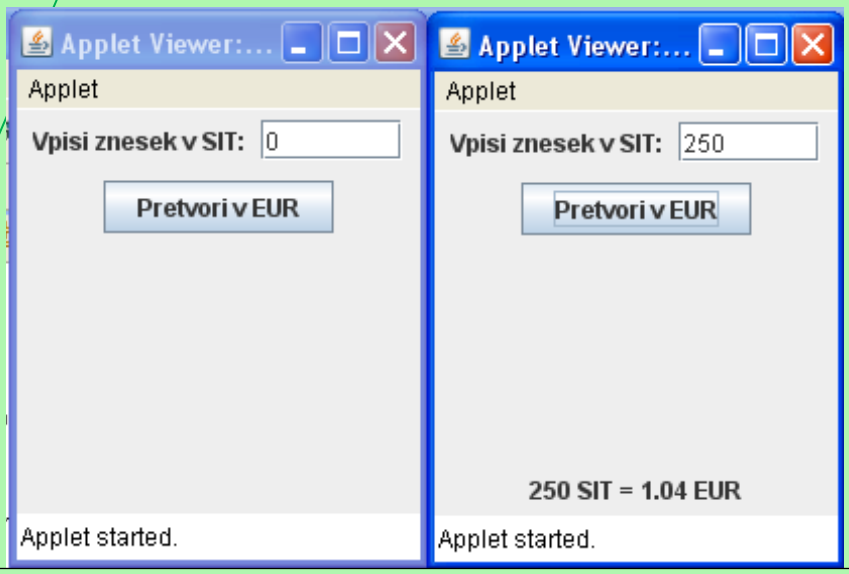
```
import javax.swing.*;
import java.awt.*;

public class Aplet extends JApplet
{
    private static final long serialVersionUID = 1L;//Eclipse

    //aplet se zažene
    //inicializacija spremenljivk
    //praviloma lahko samo to napišemo, ostale niso potrebne
    //enako kot main metoda
    public void init()
    {
        JLabel a = new JLabel ("Neki!");
        Container c = getContentPane();
        c.add(a);
    }

    //aplet postane aktiven
    public void start()
    {

    }

    //se izvede vsakokrat, ko uporabnik zapusti stran z apletom
    public void stop()
    {

    }

    //se izvede, ko uporabnik zapre brskalnik
    public void destroy()
    {

    }
}
```

  - html koda za aplet

```
<HTML>
<HEAD><TITLE>Moj aplet</TITLE></HEAD>
<BODY>
  <APPLET CODE="Aplet.class" WIDTH=300 HEIGHT=50></APPLET>
</BODY>
</HTML>
```

- primer za menjavo iz SIT v EUR

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

//razred menjava (iz SIT v EUR)
public class Menjava extends JApplet
{
   private static final long serialVersionUID = 1L;      //Eclipse
   final double tecaj = 239.64;                          //tecaj
   JLabel znesek = new JLabel("Vpisi znesek v SIT: ");
   JLabel rezultat = new JLabel("");
   JTextField vnos = new JTextField("0", 6);
   JButton gumb = new JButton("Pretvori v EUR");

   //zacetek apleta, inicializacija
   public void init()
   {
         JPanel izracun = new JPanel();
         izracun.add(znesek);
         izracun.add(vnos);
         JPanel gumbi = new JPanel();
         gumbi.add(gumb);
         JPanel izpis = new JPanel();
         izpis.add(rezultat);
         Container vsebina = getContentPane();
         vsebina.add(izracun, BorderLayout.NORTH);
         vsebina.add(gumbi, BorderLayout.CENTER);
         vsebina.add(izpis, BorderLayout.SOUTH);

         //gumbu dodamo poslušalca
         gumb.addActionListener
         (
               new ActionListener()
               {
                     public void actionPerformed(ActionEvent d)
                     {
                           rezultat.setText(vnos.getText() + " SIT = " +
                  Math.round(100*Double.parseDouble(vnos.getText())
                  /tecaj)/100.0 + " EUR");
                     }
               }
         );
   }
}
```




```
<HTML>
<HEAD><TITLE>Primer apleta</TITLE></HEAD>
<BODY>
  <H2>Pretvorba SIT v EUR</H2><BR>
  <APPLET CODE="Menjava.class" WIDTH=300 HEIGHT=100></APPLET>
</BODY>
</HTML>
```

- aplet kot program (z main metodo) – lahko poženemo kot normalen program: java ApletProgram

```
import javax.swing.*;
import java.awt.*;

public class ApletProgram extends JApplet
{
   private static final long serialVersionUID = 1L;     //Eclipse

    // s to metodo se aplet pozene
   public void init()
   {
     this.getContentPane().add(new JLabel("Program ali aplet? Oboje!"));
   }

   //main metoda
   public static void main(String[] args)
   {
         ApletProgram aplet = new ApletProgram(); //ustvarimo nov aplet
         JFrame okno = new JFrame("Program: ApletProgram.class");
         okno.setSize(400, 200);
         okno.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         okno.getContentPane().add(aplet, BorderLayout.CENTER);
         aplet.init();                  //poklicemo metodo init
         aplet.start();                 //zazenemo aplet
         okno.setVisible(true);         //okno postavimo vidnega
   }
}
```

Datoteke in tokovi

- **tokovi (streams)** – omogočajo branje oziroma zapisovanje podatkov
  - izjeme (deljenje z nič, branje iz neobstoječe datoteke, nedostopno omrežje, napačen URL naslov, dostop do elementa z indeksom izven meja tabele,…) – vsaka metoda lahko sproži izjemo (throws an exception), takrat se ustvari nov objekt, ki to izjemo opisuje
    - vrste izjem
      - Throwable (skupni predhodnik vseh izjemnih dogodkov)
        - Error (opis hudih napak (izčrpanje pomnilnika) – program ne more nadaljevati; ponavadi ne obravnavamo)
        - Exception
          - RuntimeException (izjeme, ki so posledica napak v programu – prekoračitev obsega, neprimerna pretvorba tipov) – to navadno sami odpravimo v programu, zato tega tudi ne obravnavamo
          - ostale potomce pa težko zaznamo in preprečimo – zanje poskrbimo s programsko obravnavo (razglašanjem ali prestrezanjem)
    - proženje, obravnava izjem, tray-catch-finally

```
import java.io.IOException;   //potrebujemo, ce obravnavamo izjeme

public class Program
{
      //kadar menimo, da lahko pride do izjeme, to oznacimo v glavi metode
      //ce ne pride do napake, metoda vraca int, drugace se predcasno konca
      //in sprozi izjemo tipa IOException (napaka pri branju, pisanju,...)
      public int pisiDat(String imeDat) throws IOException
      //ista metoda lahko razglasi tudi več izjem (locimo z vejico)
      //public int pisiDat(String imeDat) throws FileNotFoundException,
      //IOException
      {
            try
            {
                  //stavki, ki lahko sprozijo izjemo
            }
            catch (Exception e)
            {
                  //obravnava vseh izjem
                  System.out.println(e.getMessage()); //izpis podrobnega
                                                      //sporočila izjeme
                  e.printStackTrace();                //izpis sklada
            }
            finally
            {
                  //stavki za varen zakljucek dela
                  //zgodi v vsakem primeru, tudi ce pride do izjeme
                  //tudi ni nujno, da ga podamo
            }
            return stevilo;
    }
}
```

- binarni tokovi (tok bajtov)
  - za prenašanje binarne vsebine (slike, zvok, zaporedna predstavitev objektov)
  - FileInputStream (za branje)
  - FileOutputStream (za pisanje)

```
import java.io.*; //potrebujemo za delo s tokovi

public class Prepisi
{
    public static void main(String[] args) throws IOException
    {
        if(args.length < 2)
        {
            System.out.println("Uporaba: java Prepisi izvorDat ponorDat");
            System.exit(1);
        }

        InputStream vhTok = null;   //za branje binarnih podatkov (bajtov)
                                    //iz datoteke
        OutputStream izhTok = null; //za pisanje binarnih podatkov (bajtov)
                                    //v datoteko

        try
        {
            //ustvarimo nov tok za branje bajtov (args[0] - ime datoteke)
            vhTok = new FileInputStream(args[0]);
            //ustvarimo nov tok za pisanje bajtov (args[1] - ime datoteke)
            izhTok = new FileOutputStream(args[1]);
            int bajt;

            //zanko ponavljamo tako dolgo, dokler nam ne vrne -1 (ko
            //pridemo do konca datoteke)
            while ((bajt = vhTok.read()) != -1)//beremo bajt za bajtom iz
                                               //datoteke
            {
                  izhTok.write(bajt);     //pisemo bajt za bajtom v datoteko
            }
        }
        catch(FileNotFoundException e)
        {
            //dobimo sporocilo o napaki, jo izpisemo ter koncamo program
            System.out.println("Napaka pri odpiranju datoteke: " +
                                                          e.getMessage());
            System.exit(1);
        }
        catch(IOException e)
        {
            //dobimo sporocilo o napaki, jo izpisemo ter koncamo program
            System.out.println("Napaka: " + e.getMessage());
            System.exit(1);
        }
        finally
        {
            //tokova je potrebno zapreti
            vhTok.close();
            izhTok.close();
        }
    }
}
```

- znakovni tokovi
  - za znake
  - InputStreamReader
    - FileReader
  - OutputStreamWriter
    - FileWriter

```
import java.io.*;

//podobno kot branje bajtov, samo tokovi drugacni
public class PrepisiZnake
{
    public static void main(String[] args) throws IOException
    {
        if(args.length < 2)
        {
            System.out.println("Uporaba: java Prepisi izvorDat ponorDat");
            System.exit(1);
        }

        //tok za branje znakov iz datoteke
        FileReader vhTok = new FileReader(args[0]);
        //tok za pisanje znakov v datoteko
        FileWriter izhTok = new FileWriter(args[1]);
        int znak;

        while ((znak = vhTok.read()) != -1)
        {
            izhTok.write(znak);
        }

        vhTok.close();
        izhTok.close();
    }
}
```

- ovijanje tokov
  - če želimo prebrati druge tipe podatkov (npr. cela ali realna števila) – uporabimo druge razrede v paketu java.io in jih po potrebi povežemo z datotečnimi tokovi
  - npr. FileInputStream ovijemo z DataInputStream

```
import java.io.*;

public class PrepisiVrstice
{
    public static void main(String[] args) throws IOException
    {
        if(args.length < 2)
        {
            System.out.println("Uporaba: java Prepisi izvorDat ponorDat");
            System.exit(1);
        }

        //ce zelimo prebrati cela stevila:
        //
        //DataInputStream zna prebrane bajte združiti v tip double
        //
        //  FileInputStream tok = new FileInputStream(ime);
        //  DataInputStream podatki = new DataInputStream(tok);
        // ali
        //  DataInputStream podatki = new DataInputStream(new
        //                                      FileInputStream(ime));
        //
        //  double stevilo = podatki.readDouble();
        //
        //lahko uporabimo tudi medpomnenje
        //  DataInputStream podatki = new DataInputStream(new
        //              BufferedInputStream(new FileInputStream(ime)));


        //FileReader in FileWriter ovijemo z BufferedReader in
        //BufferedWriter
        BufferedReader vhTok = new BufferedReader(new FileReader(args[0]));
        BufferedWriter izhTok =new BufferedWriter(new FileWriter(args[1]));
        String vrstica;

        while ((vrstica = vhTok.readLine()) != null) //preberemo celo
                        //vrstico (ko smo na koncu, vrne .readLine() null)
        {
            izhTok.write(vrstica);  //zapisemo vrstico
            izhTok.newLine();       //nova vrstica
        }

        //oba tokova moramo zapreti
        vhTok.close();
        izhTok.close();
    }
}
```

- datoteke z naključnim dostopom

```
import java.io.*;

public class NakljucenDostop{
    public static void main(String[] args) throws IOException{
        String ime = "datoteka.dat";
        int max = (int) (Math.random()*100) + 1;
        int lokacija = (int) (Math.random()*max) + 1;

        //najprej preverimo, ce je v datoteki ze kaj napisano
        if(args.length > 0){
            //ce je, nastavimo ime, drugace se uposteva zgornje ime in se
            //ustvari nova datoteka
            ime = args[0];
        }
        //RandomAccessFile omogoca nakljucen dostop, ki implementira vmesik
        //DataInput in DataOutput
        //datoteko odpremo za branje in pisanje (rw)
        //branje podatkov: readDouble(), readInt(), readChar(),
        //readLine(),...
        //pisanje podatkov: writeDouble(), writeInt(), writeChar(),
        //writeChars(),...
        //ce datoteka ne obstaja, jo ustvari
        RandomAccessFile datoteka = new RandomAccessFile(ime,"rw");
        long dolzina = datoteka.length();
        //ce je v datoteki ze kaj napisano, opozorimo, da jo bomo prepisali
        if (dolzina > 0)
            System.out.println("Datoteka " + ime + " bo prepisana.");
        //zapisemo nekaj nakljucnih stevil v datoteko
        for(int i=0; i<=max; i++)
            datoteka.writeInt(i);  //pisemo stevila
        System.out.println("V datoteko smo vpisali števila od 0 do "+max);
        //z metodo seek se lahko postavimo na poljuben polozaj v datoteki
        datoteka.seek(lokacija*4);
        //getFilePointer() pa vrne trenutni polozaj datotecnega kazalca
        System.out.println("Datotecni kazalec je na lokaciji " +
                                        datoteka.getFilePointer());
        int stevilo = datoteka.readInt();//preberemo stevilo na oni
                                        //lokaciji
        System.out.println("Preberemo stevilo " + stevilo);
        System.out.println("Datotecni kazalec je na lokaciji " +
                                        datoteka.getFilePointer());
        datoteka.seek(0);
        try{
            while(true){
                  stevilo = datoteka.readInt();       //beremo stevila
                  System.out.print(stevilo + " ");
            }
        }
        catch(EOFException e){
            // obravnavamo izjemo "konec datoteke" (ce do nje pride)
            // in s tem prekinemo while zanko
        }
        Finally{
            datoteka.close();
        }
    }
}
```

- pisanje in branje objektov

```
import java.io.*;

public class Objekti
{
    static final int MAX = 5;

    public static void main(String[] args)
    {
        String imeDatoteke;

        if(args.length < 1)
        {
          imeDatoteke = "Objekti.dat";
        }
        else
        {
          imeDatoteke = args[0];
        }

        zapisiPodatke(imeDatoteke);
        preberiPodatke(imeDatoteke);
    }

    private static void zapisiPodatke(String imeDat)
    {
        Prijatelj[] prijatelji = new Prijatelj[MAX];
        prijatelji[0] = new Prijatelj("Janez", "Novak", 25, "123456");
        prijatelji[1] = new Prijatelj("Micka", "Kovaceva", 21, "123456");
        prijatelji[2] = new Prijatelj("Franci","Nabalanci",25,"121221");
        prijatelji[3] = new Prijatelj("Nana", "Študent", 19, "654321");
        prijatelji[4] = new Prijatelj("Miki", "Žmauc", 35, "566556");

        try
        {
          FileOutputStream izhTok = new FileOutputStream(imeDat);

          for(int i=0; i<MAX; i++)
          {
                prijatelji[i].pisiDat(izhTok);
          }
          izhTok.close();
        }
        catch(IOException e)
        {
          System.out.println("Napaka: " + e.getMessage());
        }
    }

    private static void preberiPodatke(String imeDat)
    {
          try
          {
                FileInputStream vhTok = new FileInputStream(imeDat);
               System.out.println("Prebrali smo naslednje prijatelje: ");

                try
                {
                      while(true)
                      {
                            Prijatelj p = new Prijatelj();
```

```
                              p.beriDat(vhTok);
                              System.out.println(p.toString());
                        }
                  }
                  catch(IOException e)
                  {
                        //ni potrebno nic napisati
                  }
                  vhTok.close();
            }
            catch(FileNotFoundException e)
            {
                  System.out.println("Napaka: ni datoteke " + imeDat);
            }
            catch(IOException e)
            {
                  System.out.println("Napaka: " + e.getMessage());
            }
            catch(ClassNotFoundException e)
            {
                  System.out.println("Napaka: ni razreda "+e.getMessage());
            }
      }
}

//razred mora imeti vmesnik Serializable, ker metoda writeObject() objekt
//najprej pretvori v zaporedje bajtov, zato morajo biti ti objekti take
//vrste, da omogocajo serializacijo - ce niso, metoda sprozi izjemo
//NotSerializableException; v primeru napake razreda pa
//InvalidClassException
class Prijatelj implements Serializable
{
      private String priimek;
      private String ime;
      private int starost;
      private String telefon;

      public Prijatelj()
      {
            this("", "", 0, "");
      }

      public Prijatelj(String i, String p, int s, String t)
      {
            this.priimek = p;
            this.ime = i;
            this.starost = s;
            this.telefon = t;
      }

      public void pisiDat(FileOutputStream izhodniTok) throws IOException
      {
            //tok za pisanje objektov
            ObjectOutputStream objIzhTok = new
                                          ObjectOutputStream(izhodniTok);
            objIzhTok.writeObject(this);  //pisemo objekt
            //na koncu zapisovanja, preden tok zapremo, moramo "splakniti"
            //tok (vsi bajti iz medpomnilnika se zapišejo v tok)
            objIzhTok.flush();
      }
```

```
	public void beriDat(FileInputStream vhodniTok) throws IOException,
                                                      ClassNotFoundException
	{
		//tok za branje objektov
		ObjectInputStream objVhTok = new ObjectInputStream(vhodniTok);
		//beremo objekt
		//(Prijatelj) mora biti, ker metoda readObject() vraca Objekt,
		//zato moramo napisati tip objekta
		//podobno je (int)(Math.random()*10) - prav tako tip v oklepaju
		Prijatelj stari = (Prijatelj)objVhTok.readObject();
		this.priimek = stari.priimek;
		this.ime = stari.ime;
		this.starost = stari.starost;
		this.telefon = stari.telefon;
	}
}
```

Delo z mrezo

- razred URL

- ustvaritev objekta URL

```
URL vir1 = new URL(URL naslov);
URL vir1 = new URL("http://lgm.fri.uni-lj.si/op2/index.html");

URL vir2 = new URL(protokol, gostitelj/host, ime datoteke);
URL vir2 = new URL("http", "lgm.fri.uni-lj.si/op2/", "index.html");
```

- metode, s katerimi dobimo posamezne elemente URL naslova

```
import java.net.*;         //potrebujemo za delo z mrezo

public class Vaja
{
   //ce je URL napacno sestavljen, konstruktor sprozi izjemo
   //MalformedURLException
   public static void main(String[] args) throws MalformedURLException
   {
         URL vir1 = new URL("http://lgm.fri.uni-lj.si/op2/index.html");

         vir1.getFile();   //vrne ime datoteke
         vir1.getHost();   //vrne ime gostitelja
         vir1.getPath();   //vrne tisti del URL naslova, ki doloca pot
         vir1.getPort();   //vrne številko vrat (port number)
         vir1.getProtocol(); //vrne ime protokola
         vir1.getRef();    //vrne sidro (anchor ali reference)
   }
}
```

- branje datoteke, podane z URL

- izpisovanje na zaslon

```
import java.io.*;
import java.net.*;

public class PreberiURL
{
   public static void main(String[] args)
   {
         if(args.length < 1)
         {
               System.out.println("Uporaba: java PreberiURL URL");
               System.exit(1);
         }

         //ker moramo ujeti izjemo IOException, ce do nje pride, moramo
         //narediti try/catch blok
         try
         {
               URL vir = new URL(args[0]);   //ustvarimo nov URL
               //datoteko bomo brali po vrsticah, zato moramo
               //InputStreamReader (bere znake) oviti z BufferedReader,
               //tako preberemo celo vrstico
               BufferedReader tok = new BufferedReader(new
InputStreamReader(vir.openStream()));
               String vrstica;

               //beremo in izpisujemo vrstice, dokler ne pridemo do
               //konca datoteke
```

```
                while ((vrstica = tok.readLine()) != null)
                {
                        System.out.println(vrstica);
                }

                tok.close();      //zapremo tok
        }
        catch(IOException e)//ce pride do izjeme(napake)koncamo program
        {
                System.out.println("Napaka: " + e.getMessage());
                System.exit(1);
        }
    }
}
```

Zagon programa

```
java PreberiURL http://lgm.fri.uni-lj.si/op2/index.html
```

- pisanje v datoteko

```
import java.io.*;
import java.net.*;
public class Shrani
{
   public static void main(String[] args) throws IOException
   {
        if(args.length < 1)
        {
                System.out.println("Uporaba: java Shrani URL");
                System.exit(1);
        }
        URL izvor = new URL(args[0]);
        String ime = izvor.getFile();       //dobimo ime datoteke
        //izlocimo predhodnje .../ pred imenom (uporabimo samo ime)
        ime = ime.substring(ime.lastIndexOf('/')+1);
        int bajt;
        //ker ne vemo, ali bo datoteka tekstovna ali binarna, bomo
        //uporabili BufferedInputStream, ki omogoca branje preko
        //medpomnilnika
        BufferedInputStream vhod = new
BufferedInputStream(izvor.openStream());
        BufferedOutputStream izhod = new BufferedOutputStream(new
FileOutputStream(ime));

        try
        {
                while ((bajt = vhod.read()) != -1) //beremo bajte
                {
                        izhod.write(bajt);      //pisemo bajte
                }
        }
        catch(IOException e)
        {
                System.out.println("Napaka: " + e.getMessage());
                System.exit(1);
        }
        finally
        {
                //zapremo tokova
                vhod.close();
                izhod.close();
        }
   }
```

```
}
```

- **prikaz slike, podane z URL**

```
import java.awt.*;
import javax.swing.*;
import java.net.*;

class Prikazi extends JPanel
{
   public final int SIRINA = 800;
   public final int VISINA = 600;
   public final int ROB = 10;
   public Image slika = null;
   private URL url = null;

   public static void main(String[] args)
   {
         if(args.length < 1)
         {
               System.out.println("Uporaba: java Prikazi URL_slike");
               System.exit(1);
         }

         JFrame okno = new JFrame("Prikaz bitne slike " + args[0]);
         Prikazi s = new Prikazi(args[0]);
         okno.getContentPane().add(s);
         okno.setSize(s.SIRINA, s.VISINA);
         okno.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         okno.setVisible(true);
   }

   public Prikazi(String naslov)
   {
         try
         {
               url = new URL(naslov);
               slika = Toolkit.getDefaultToolkit().getImage(url);
         }
         catch(MalformedURLException e)
         {
               System.out.println("Napaka: " + e.getMessage());
         }
   }

   public void paintComponent(Graphics g)
   {
         super.paintComponent(g);

         if (slika != null)
         {
               g.drawImage(slika, ROB, ROB, this);
         }
   }
}
```

niti

- razred Thread

```
//moramo extendat niti (Thread)
public class Niti extends Thread
{
   private int pocakaj;

   //konstruktor
   public Niti(String ime)
   {
         super(ime);
         pocakaj = (int) (Math.random()*5000);//cas, ki ga bo nit cakala
         System.out.println("Ime niti: " + getName() + ", cas pocivanja:
" + pocakaj);
   }

   //telo niti - doloca delovanje
   public void run()
   {
         for(int i=0; i<20; i++)
         {
               System.out.print(getName());

               try
               {
                     sleep(pocakaj);         //nit  caka (stoji na miru)
               }
               catch (InterruptedException e)
               {

               }
         }
         System.out.println("Konec niti: " + getName());//izpisemo ime
                                          //niti, ki se je koncala (umrla)
   }

   public static void main(String[] args)
   {
         final int MAX = 5;
         Niti[] niti = new Niti[MAX];

         for(int i=0; i<MAX; i++)
         {
               niti[i] = new Niti(" " + i + " ");  //ustvarimo novo nit z
                                                   //imenom i
         }

         System.out.println("Zagon vseh niti ...");

         for(int i=0; i<MAX; i++)
         {
               niti[i].start();  //zazenemo niti
         }
   }
}
```

- **vmesnik Runnable**

```
//implementiramo Runnable (ce recimo ne moremo extendat Thread)
public class Niti1 implements Runnable
{
   private Thread nit;     //nit tipa Thread
   private String ime;
   private int pocakaj;

   public Niti1(String ime)
   {
         this.ime = ime;
         this.pocakaj = (int) (Math.random()*5000);
         System.out.println("Ime niti: " + ime + ", cas pocivanja: " +
pocakaj);
   }

   public void pozeni()
   {
         if(nit == null)
         {
               nit = new Thread(this, ime);
               nit.start();
         }
   }

   public void run()
   {
         for(int i=0; i<20; i++)
         {
               System.out.print(nit.getName());

               try
               {
                     nit.sleep(pocakaj);     //nit caka
                     //Thread.sleep(pocakaj);//lahko tudi tako napisemo
               }
               catch (InterruptedException e)
               {
               }
         }
         System.out.println("Konec niti: " + nit.getName());
   }

   public static void main(String[] args)
   {
         final int MAX = 5;
         Niti1[] niti = new Niti1[MAX];

         for(int i=0; i<MAX; i++)
         {
               niti[i] = new Niti1(" " + i + " ");
         }
         System.out.println("Zagon vseh niti ...");

         for(int i=0; i<MAX; i++)
         {
               niti[i].pozeni();
         }
   }
}
```

- **stanja niti, prioriteta**
  - metode za prehode med posameznimi stanji

```
nit.start();   //pricne izvajanje niti in jo tako spravi v aktivno
stanje (klice se njena metoda run),
nit.sleep(i);  //zacasno zaustavi izvajanje niti,
nit.wait();    //blokira izvajanje niti, dokler ni izpolnjen dolocen
pogoj,
nit.notify();  //nadaljuje izvajanje niti, ko je izpolnjen dolocen
pogoj,
nit.yield();   //prepusti procesorski cas drugim nitim.
```

  - prioriteta

```
public class Niti2 extends Thread
{
   private int pocakaj;

   public Niti2(String ime)
   {
         super(ime);
         pocakaj = 500;
         //prioriteta niti, najmanjsa prioriteta (Thread.MIN_PRIORITY)
         //je 1, najvecja (Thread.MAX_PRIORITY) 10
         //in privzeta/normalna (Thread.NORM_PRIORITY) pa 5
         //tista nit z najvecjo prioriteto se bo koncala hitreje
         //nastavimo prioriteto s setPriority (dobimo pa z getPriority)
         this.setPriority((int) (Math.random()*Thread.MAX_PRIORITY)+1);
         System.out.println("Ime niti: " + this.getName() + ", s
prioriteto: " + this.getPriority());
   }

   public void run()
   {
         for(int i=0; i<20; i++)
         {
               System.out.print(getName());
               try
               {
               sleep(pocakaj);
               }
               catch (InterruptedException e)
               {
               }
         }
         System.out.println("Konec niti: " + getName());
   }
   public static void main(String[] args)
   {
         final int MAX = 5;
         Niti2[] niti = new Niti2[MAX];
         for(int i=0; i<MAX; i++)
         {
               niti[i] = new Niti2(" " + i + " ");
         }
         System.out.println("Zagon vseh niti ...");
         for(int i=0; i<MAX; i++)
         {
               niti[i].start();
         }
   }
}
```

- **uporaba niti**

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

//razred za graficni primer uporabe niti
public class Stevec extends JFrame implements ActionListener, Runnable
{
   private static final long serialVersionUID = 1L;     //Eclipse
   private String naslov = "Števec";
   private Container vsebina = null;
   private JPanel kontrole = new JPanel();
   private JPanel plosca = new JPanel();
   private JButton start = new JButton("Start");
   private JButton stop = new JButton("Stop");
   private JButton reset = new JButton("Reset");
   private JLabel izpis = new JLabel("...");
   private int st = 0;
   //stevec (stevilo, ki ga bomo povecevali ob niti)
   private boolean pocakaj = false;
   private Thread stejNit = null;

   public Stevec()
   {
         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         setTitle(naslov);
         setSize(300,100);
         start.addActionListener(this);
         stop.addActionListener(this);
         reset.addActionListener(this);
         kontrole.add(start);
         kontrole.add(stop);
         kontrole.add(reset);
         plosca.add(izpis);
         vsebina = getContentPane();
         vsebina.add(kontrole, BorderLayout.NORTH);
         vsebina.add(plosca, BorderLayout.CENTER);
   }

   public void actionPerformed(ActionEvent e)
   {
         if(e.getSource() == start)
         {
               //ce pritisnemo start, ustvarimo novo nit, ce je ni,
               //drugace jo pozenemo
               pocakaj = false;
               if(stejNit == null)
               {
                     stejNit = new Thread(this);
                     stejNit.start();
               }
         }
         if(e.getSource() == stop)
         {
               //ce pritisnemo stop, cakamo
               pocakaj = true;
         }
         if(e.getSource() == reset)
         {
               //ce pritisnemo reset, se st postavi na 0
```

```
                st = 0;
                izpis.setText(Integer.toString(st));
          }
    }

    public void run()
    {
          while (true)
          {
                try
                {
                      //kako hitro se bodo cifre spreminjale na oknu
                      stejNit.sleep(100);
                }
                catch(InterruptedException e)
                {
                }
                if(!pocakaj)
                      izpis.setText(Integer.toString(st++));
          }
    }
    public static void main(String[] args)
    {
          Stevec stevec = new Stevec();
          stevec.setVisible(true);
    }
}
```

- **sinhronizacija niti**
  - ko uporabljamo več niti, lahko pride do težav

```
//pri vec nitih se lahko naredi, da prva preveri doloceno zalogo, potem
//caka pred nakupom, prav takrat pa druga nit preveri zalogo in opravi
//nakup (ko prva se caka), tako prva ne mora dokoncati nakup, ker zaloge
//ni vec. Ce pa niti sinhroniziramo, preprecimo drugi, da opravi nakup
//pred prvo (druga caka, da prva konca)
public class Trgovina1
{
   private int zaloga = 0;

   //konstruktor
   public Trgovina1()
   {
         zaloga = (int) (Math.random()*500) + 10;
   }

   //metoda, ki vrne zalogo
   public int zaloga()
   {
         return zaloga;
   }

   //sinhronizirana metoda - do nje lahko dostopa samo ena nit naenkrat
   public synchronized int nakup(int n)
   {
         if(zaloga >= n)
         {
               zaloga -= n;
               return n;
         }
         return -1;
   }

   //sinhronizirana metoda - do nje lahko dostopa samo ena nit naenkrat
   public synchronized boolean prosto(int n)
   {
         if(zaloga >= n)
         {
               return true;
         }
         else
         {
               return false;
         }
   }

   //main metoda
   public static void main(String[] args)
   {
         Trgovina1 trg = new Trgovina1();
         Kupec1[] kupci = new Kupec1[Integer.parseInt(args[0])];

         for(int i=0; i<kupci.length; i++)
         {
               kupci[i]=new Kupec1("Kupec "+Integer.toString(i+1),trg);
         }

         System.out.println("Zaloge je " + trg.zaloga() + ", trgovina je
odprta ...");
```

```
        for(int i=0; i<kupci.length; i++)
        {
            kupci[i].start();
        }
    }
}
//razred kupci
class Kupec1 extends Thread
{
    private Trgovina1 trgovina;

    //konstruktor
    public Kupec1(String i, Trgovina1 t)
    {
        super(i);
        trgovina = t;
    }
    //metoda, ki oberavnava, kako se kupci obnasanjo
    public void run()
    {
        while (trgovina.zaloga() > 0)
        {
            int stevilo = (int) (Math.random()*10) + 1;
            //sinhroniziramo blok znotraj metode (imenujemo kriticni
            //odsek)
            synchronized(trgovina)
            {
                if(trgovina.prosto(stevilo))
                {
                    try
                    {
                        sleep((int) (Math.random()*500+50));
                    }
                    catch(InterruptedException e)
                    {

                    }
                    if(trgovina.nakup(stevilo) == stevilo)
                    {
                        System.out.println("Uspešen nakup - " +
                        getName()+" -- število: "+stevilo +
                        " -- zaloga: " +trgovina.zaloga());
                    }
                    else
                    {
                        System.out.println("---> Napaka pri
                        nakupu ...");
                    }
                }
                else
                {
                    System.out.println("Ni dovolj zaloge - " +
                    getName()+" -- število: "+stevilo + " –
                    zaloga: " + trgovina.zaloga());
                }
            }
        }
        System.out.println(getName() + " je zakljucil nakupovanje");
    }
}
```

- proizvajalec in potrošnik
  - včasih moramo delo dveh niti tudi časovno uskladiti – primer: ena nit pripravlja podatke (proizvajalec/producer), druga pa te podatke uporablja (potrošnik/consumer) – obe ne moreta dostopati istočasno do podatkov in druga nit ne sme dobiti podatkov, ki še niso pripravljeni ter ne sme dvakrat prebrati istega podatka

```
public class PPSinhrono
{
   public static void main(String[] args)
   {
         Odlozisce1 odl = new Odlozisce1();
         Proizvajalec1 pro = new Proizvajalec1(odl);
         Potrosnik1 pot = new Potrosnik1(odl);
         pro.start();
         pot.start();
   }
}

class Odlozisce1
{
   private int vsebina;
   private boolean naVoljo = false;

   //spet si pomagamo s tem, da metodo sinhroniziramo
   public synchronized int vzemi()
   {
         //ce hocemo niti casovno uskladiti, moramo uorabiti wait() in
         //notify()
         while(naVoljo == false)
         {
               try
               {
                     //dokler je naVoljo false, nit caka
                     wait();
               }
               catch (InterruptedException e)
               {

               }
         }
         int v = vsebina;
         vsebina = 0;
         System.out.println("Vzeto: " + v);
         naVoljo = false;
         //obvestimo, da ne cakamo vec
         notify();
         return v;
   }

   //spet si pomagamo s tem, da metodo sinhroniziramo
   public synchronized void postavi(int v)
   {
         while(naVoljo == true)
         {
               try
               {
                     wait();
               }
               catch (InterruptedException e)
```

```
                {
                }
        }
        vsebina = v;
        System.out.println("Postavljeno: " + v);
        naVoljo = true;
        notify();
    }
}

class Proizvajalec1 extends Thread
{
    private Odlozisce1 polica;
    public Proizvajalec1(Odlozisce1 o)
    {
        this.polica = o;
    }

    public void run()
    {
        for (int i = 1; i <= 10; i++)
        {
            polica.postavi(i);
            try
            {
                sleep((int)(Math.random()*100));
            }
            catch (InterruptedException e)
            {
            }
        }
    }
}

class Potrosnik1 extends Thread
{
    private Odlozisce1 polica;
    public Potrosnik1(Odlozisce1 o)
    {
        this.polica = o;
    }

    public void run()
    {
        int vrednost = 0;
        do
        {
            try
            {
                sleep((int)(Math.random()*100));
            }
            catch (InterruptedException e)
            {

            }
            vrednost = polica.vzemi();
        } while(vrednost < 10);
    }
}
```