Programski jezik JAVA
PREDAVANJE 5

Prezentacija kreirana na osnovu sljedeće literature :
Dejan Živković: Osnove Java programiranja; Bruce Eckel: Misliti na Javi

Grafičko programiranje

- Svi proizvođači softverskih alata podržavaju izradu grafičkih aplikacija u njima - GUI programiranje (GUI=*graphic user interface*)
- Podrška za GUI u Javi nalazi se u paketima `java.awt.*` i `javax.swing.*` (AWT=Abstract Window Toolkit)
  - **java.awt** paket koji postoji od prve verzije programskog jezika Java;
  - Obezbjeđuje upotrebu minimalnog skupa komponenti grafičkog interfejsa, koje posjeduju sve platforme koje podržavaju Java-u
  - Za kreiranje grafičkih komponenti na ekranu, java.awt koristi biblioteke Java grafičkih klasa prisutnih na datoj platfomi. ***Program pisan sa awt klasama ima različit izgled na različitim platformama***.
  - **javax.swing** –od verzije Java 1.2 mnoge klase iz biblioteke AWT su napisane tako ***da ne zavise od konkretnog operativnog sistema, komponente rade na isti način nezavisno od operativnog sistema i izgledaju isto na svim platformama*** – biblioteka **Swing**.
- ***SWING nije potpuna zamjena za AWT i oba se paketa često koriste zajedno***.

Kreiranje prozora (JFrame)

- Prozor najvišeg nivoa u jednoj aplikaciji (prozor koji nije sadržan u drugom prozoru) naziva se ***frame*** u Javinoj terminologiji.
- Prozor se dijeli na okvire i dijaloge
- U AWT biblioteci prozor modelira klasa `Frame`, a u SWING biblioteci klasa `JFrame`

```
 package graf;
import javax.swing.*;

public class Graf extends JFrame  {
 public static final int DEFAULT_WIDTH  = 300;
 public static final int DEFAULT_HEIGHT = 200;

public Graf(){
          setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      }

    public static void main(String[] args) {
        Graf frame = new Graf();
           frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
           frame.setVisible(true);
    }
}
```

Kreiranje prozora (JFrame)

- Prozor konstruišemo proširivanjem klase `JFrame`.

Podrazumijevani konstruktor klase JFrame kreira frame dimenzija 0 × 0 piksela, pa u konstruktoru naše klase pozivamo metodu `setSize` koja daje dimenzije prozoru (300 × 200 piksela u našem slučaju).

- U glavnom programu kreiramo frame pomoću operatora `new`
- U liniji koda:

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

određujemo šta će se desiti kada korisnik zatvori prozor

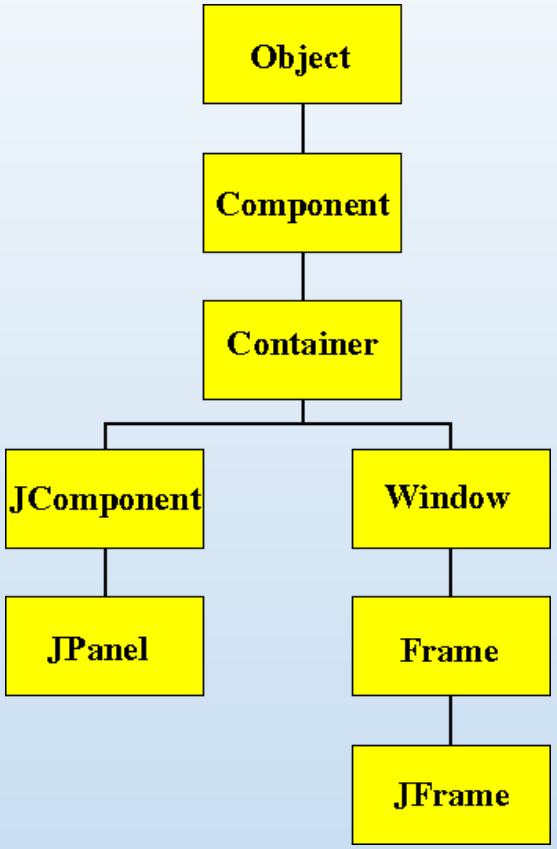
Naš izbor je da se program završi (*exit*).

- Nakon konstrukcije prozor nije vidljiv na ekranu. Da bi postao vidljiv potrebno je pozvati metodu `setVisible`:

```
frame.setVisible(true);
```

Kreiranje prozora (JFrame)

- Elementi na kojima se zasniva grafičko programiranje su *kontejneri* i *komponente*
- *Kontejneri služe za grupisanje komponenti i mogu se prikazati na ekranu*
- Komponente se mogu prikazati samo unutar nekog kontejnera (**dugme** je primjer komponente, dok je **okvir** primjer kontejnera)
- `JFrame` nasljeđuje brojne metode od klasa `Window` i `Container`. Među njima su i metode za pozicioniranje prozora na ekranu: `setSize` i `setLocation`
- Da bismo dobro pozicionirali prozor moramo znati rezoluciju ekrana
- Tu informaciju možemo dobiti od klase `java.awt.Toolkit`.



Kreiranje prozora (JFrame)

- Koordinate tačke na ekranu izražavaju se u *pikselima*
- Početak koordinatnog sistema (0,0) je u lijevom gornjem vrhu ekrana; osa *x* ide slijeva na desno, a osa *y* odozgo prema dolje
- U paketu `java.awt` postoji klasa `Dimesion` koja služi za predstavljanje širine i visine ekrana

- Dimenzije ekrana možemo dobiti:

```
Toolkit kit = Toolkit.getDefaultToolkit();  //Statička metoda
Dimension screen = kit.getScreenSize();
int height = screen.height;  //Javne promjenljive članice !!
int width  = screen.width;
```

Kreiranje prozora (JFrame)

```
// Koristimo java.awt.Toolkit i java.awt.Dimension
 import java.awt.*;
 import javax.swing.*;
  public class SimpleFrameTest   {
    public static void main(String[] args)      {
       CenteredFrame frame = new CenteredFrame();
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
     }
 }
 class CenteredFrame extends JFrame  {
     public CenteredFrame()      {
         // Uzmimo dimenzije ekrana
         Toolkit kit = Toolkit.getDefaultToolkit();
         Dimension screen = kit.getScreenSize();
         int height = screen.height;
         int width  = screen.width;

         // Centriramo prozor na ekranu
         setSize(width/2, height/2);
         setLocation(width/4, height/4);
         setTitle("Centrirani prozor");
     }
}
```

Kao rezultat izvršavanja otvara se prozor na sredini ekrana, sa naslovom "Centrirani prozor". Klikom na dugme "x" prekida se izvršavanje programa

JPanel

- Moguće je direktno prikazivati informacije u *frame*-u, ali to nije dobar način konstrukcije korisničkog interfejsa.
- Frame je komponenta koja je namijenjena držanju drugih, specijalizovanih komponenti
- Prostija verzija kontejnera opšte namjene je ***panel***
- `JPanel` klasa implementira *panel*. Panel je:
  - Površina po kojoj se može crtati;
  - Container koji može sadržati druge komponente
- Da bi bio vidljiv na ekranu, panel se mora dodati kontejneru višeg nivoa (npr. okviru)

  - U tom cilju treba od prozora dobiti *content pane* pomoću metode `getContentPane`.

JPanel

- *Content pane* je `Container` u koji stavljamo druge komponente
- U tu svrhu on ima metodu `add`
- Kod za dodavanje komponente izgleda ovako:

```
Container contentPane = frame.getContentPane();
Component c = ....
contentPane.add(c);
```

JPanel

- Ako želimo crtati u JPanel-u možemo postupiti na sljedeći način:
  - Definišemo klasu koja proširuje `JPanel`;
  - Preradimo metodu `paintComponent` u kojoj se mora nalaziti kod za crtanje.
- Metoda `paintComponent` definisana je u klasi `JComponent` i uzima jedan argument tipa `Graphics`.
- `Graphics` je klasa koja sadrži objektne metode za crtanje, pisanje i prikazivanje slika. Crtanje u Java-i ide kroz `Graphics` objekat. Konačno, naša klasa za crtanje u panel imala bi ovaj oblik (ime klase je naravno proizvoljno):

```
class MojPanel extends JPanel
{
    public void paintComponent(Graphics g)
    {
 // kod za crtanje; treba nadjačati metod -
// paintComponent() iz klase JPanel kako bi se omogućilo -
// crtanje nove komponente
    }
}
```

JPanel

- Svaki put kada je potrebno iscrtati panel *event handler* će pozvati metodu `paintComponent` automatski
- Isto važi za `paintComponent` metode svih drugih komponenti. Programer nikad ne poziva metodu `paintComponent` sam!
  - Npr. minimizovanje i ponovno otvaranje prozora – poziva se `paintComponent` automatski
- Ako programer želi prisiliti ponovno iscrtavanje ekrana pozvat će `repaint` metodu iz `java.awt.Component` klase. Ova će metoda pozvati redom `paintComponent` metode za sve komponente sa konfigurisanim `Graphics` objektom.
  - Npr. hitno prikazati sadržaj komponenti u sredini drugog metoda

Nastavak...

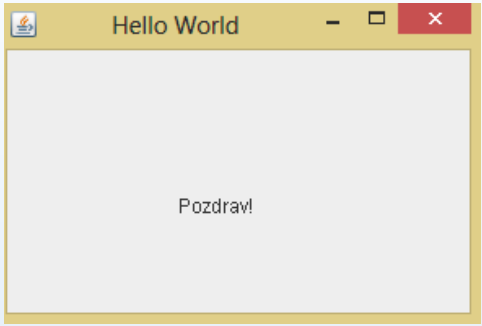
- Prikaz informacija u nekoj komponenti može se izvesti samo crtanjem u toj komponenti, čak i kada se želi prikazati tekst

- Metoda za ispis stringova u `Graphics` klasi naziva se `drawString` i poziva se na sljedeći način: `g.drawString(text,x,y);`
  - `g` je objekat tipa `Graphics`
  - `text` je tekst koji treba ispisati
  - `x` i `y` su koordinate prvog slova
- Koordinate se mjere u pikselima od gornjeg lijevog vrha komponente

- Za `paintComponent` metodu važi još i ovo pravilo:

  - Prva naredba u metodi mora biti `super.paintComponent(g)` (gdje je g objekt tipa `Graphics`) kako bi prvo komponenta sama izvršila svoj dio crtanja.

Primjer..




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

public class Graf extends JFrame  {
    public static void main(String[] args) {
        PozdravniFrame frame = new PozdravniFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
  class PozdravniFrame extends JFrame   {
public static final int DEFAULT_WIDTH  = 300;
 public static final int DEFAULT_HEIGHT = 200;
      public PozdravniFrame()       {
          setTitle("Hello World");
          setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
 // Dodajemo panel na frame. Uzmimo content pane na koji ćemo zalijepiti naš panel
          Container contentPane = getContentPane(); // metoda iz JFrame
          PozdravniPanel panel = new PozdravniPanel(); // Kreiramo naš panel
          contentPane.add(panel); // Dodajemo ga u content pane
      }
}
                    class PozdravniPanel extends JPanel   {
                          public void paintComponent(Graphics g)       {
                              super.paintComponent(g);
                              g.drawString("Pozdrav!",MESSAGE_X, MESSAGE_Y);
                          }
                          public static final int MESSAGE_X = 105;
                          public static final int MESSAGE_Y = 100;
                      }
```

Primjer

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

public class Dugmad {
public static void main(String[] args) {

// Konstruisanje okvira
JFrame okvir = new JFrame("Dugmad");
okvir.setSize(300, 200);
okvir.setLocation(100, 150);
okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

// Konstruisanje četiri dugmeta
JButton crvenoDugme = new JButton("Crveno");
JButton zelenoDugme = new JButton("Zeleno");
JButton plavoDugme = new JButton("Plavo");
JButton narandžastoDugme = new JButton("Narandžasto");
```

```
// Konstruisanje panela za
// dugmad
JPanel panel = new JPanel();

// Smiještanje dugmadi u panel
panel.add(crvenoDugme);
panel.add(zelenoDugme);
panel.add(plavoDugme);
panel.add(narandžastoDugme);

// Smijestanje panela u okvir
okvir.add(panel);
okvir.setVisible(true);
    }
  }
```

Metode za crtanje. Graphics2D

- Klasa `Graphics` za crtanje
  - Nije moguće mijenjati debljinu linija ili rotirati oblike
  - Zbog toga je dodata klasa `Graphics2D` sa mnogo većim grafičkim mogućnostima
- Klasa `Graphics2D` je podklasa klase `Graphics` (nasljeđuje klasu `Graphics`)
- Za korišćenje svih grafičkih mogućnosti klase `Graphics2D` u metodu `paintComponent()` ***mora se izvršiti eksplicitna konverzija*** tipa njegovog parametra na sljedeći način:

```
public void paintComponent(Graphics g)
{
        Graphics2D g2 = (Graphics2D) g;
        ...
}
```

Metode za crtanje. Graphics2D

- Crtanje u biblioteci Java 2D zasniva se na objektima klasa iz paketa java.awt.geom, u kom se nalaze klase:
  - `Point2D, Line2D, Rectangle2D, Shape2D,`
  - Ove klase implementiraju interfejs Shape
- Iscrtavamo ih pomoću metode `draw` klase `Graphics2D`:

```
Rectangle2D rect=...
   g2.draw(rect);
```

Klase u `java.awt.geom` paketu koriste realne koordinate umjesto cjelobrojnih. Java 2D biblioteka koristi koordinate tipa **float** u mnogim svojim unutrašnjim *floating-point* računima, što stvara probleme kad se koriste realne (double) konstante

Naime, realne konstante (kao 1.3) su tipa `double`, a kompajler neće dozvoliti da se `double` pretvori u `float` bez eksplicitne konverzije. To nas tjera da sve konstante pišemo kao float-konstante (npr `1.3F`) ili da koristimo eksplicitne konverzije (`float x = (float) 1.3;`).

Nastavak...

- Da bi korisnika oslobodili potrebe da radi sa konstantama tipa `float` (ako to ne želi) dizajneri Java-inih biblioteka kreirali su dvije subklase u svakoj *shape* - klasi, dok su *shape* - klase apstraktne. Na primjer, apstraktna klasa `Rectangle2D` ima dvije podklase:

- `Rectangle2D.Float`
- `Rectangle2D.Double`

- Prva klasa radi s koordinatama tipa `float`, a druga s koordinatama tipa `double`. Te su klase, ustvari, statičke unutrašnje klase u `Rectangle2D` koje proširuju `Rectangle2D`.

Nastavak...

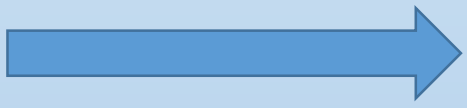
- Kreiranje pravougaonika tipa *float* i tipa *double* – prvi način

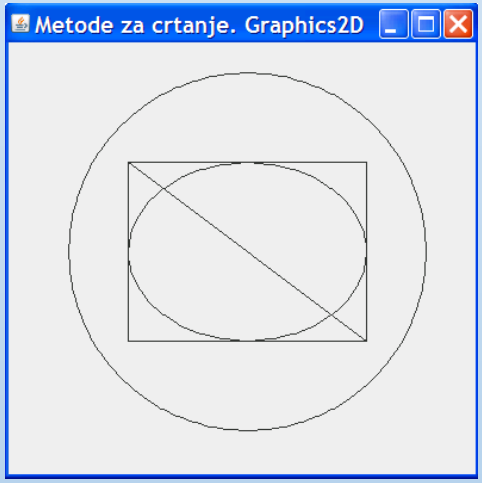
```
Rectangle2D.Float floRect=new Rectangle2D.Float(10.0F,25.0F,22.5F,20.0F);
Rectangle2D.Double douRect=new Rectangle2D.Double(10.0,25.0,22.5,20.0);
```

- Drugi način (uočite da obje klase proširuju klasu `Rectangle2D`)

```
Rectangle2D floRect = new Rectangle2D.Float(10.0F, 25.0F,22.5F,20.0F);
Rectangle2D douRect = new Rectangle2D.Double(10.0, 25.0,22.5,20.0);
```

- Specificiranje `Float` ili `Double` treba nam samo kod kreiranja objekata.
- Sljedeći program kreira ovaj crtež:





Program

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

public class DrawTest
{
    public static void main(String[] args)
    {
        DrawFrame frame = new DrawFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

class DrawFrame extends JFrame
{
    public DrawFrame()
    {
        setTitle("Metode za crtanje. Graphics2D");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

        Container contentPane = getContentPane();
        DrawPanel panel = new DrawPanel();
        contentPane.add(panel);
    }

    public static final int DEFAULT_WIDTH  = 400;
    public static final int DEFAULT_HEIGHT = 400;
}
```

```
class DrawPanel extends JPanel
{
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;

        // crtamo pravougaonik
        double leftX = 100;
        double topY  = 100;
        double width = 200;
        double height = 150;

        Rectangle2D rect = new Rectangle2D.Double(leftX, topY, width, height);
        g2.draw(rect);

        // crtamo elipsu unutar pravougaonika rect
        Ellipse2D elipsa = new Ellipse2D.Double();
        elipsa.setFrame(rect);
        g2.draw(elipsa);

        // crtamo dijagonalnu liniju
        g2.draw(new Line2D.Double(leftX, topY, leftX+width, topY+height));

        // crtamo krug sa istim centrom
        double centerX = rect.getCenterX();
        double centerY = rect.getCenterY();
        double radius  = 150;

        Ellipse2D krug = new Ellipse2D.Double();
        krug.setFrameFromCenter(centerX, centerY, centerX+radius, centerY+radius);
        g2.draw(krug);
    }
}
```

- Za crtanje u različitim bojama koristi se `setPaint` metoda iz `Graphics2D` klase. Nakon poziva `setPaint` svo dalje crtanje vrši se u postavljenoj boji. Boje su definisane u klasi `java.awt.Color`. Na primjer:

```
g2.setPaint(Color.RED);
g2.drawString("Upozorenje!", 100,100);
```

- Promjena boje pozadine panela - pomoću metode `setBackground` iz klase `java.awt.Component`.
- Zatvorene figure (pravougaonici, elipse) mogu se ispuniti bojom. Za to je potrebno izabrati boju sa **`setPaint`** metodom i zatim umjestio **`draw`** metode zvati **`fill`** metodu.
- Fontovi su modelirani klasom java.awt.Font. Za crtanje stringova sa odabranim fontovima koristimo ovakav kod:

```
Font mojFont = new Font("SansSerif", Font.BOLD, 20);
g2.setFont(mojFont);
g2.drawString("Hello World!",..,..);
```

- Konstruktor objekta `Font` uzima ime fonta, njegov tip i veličinu u pikselima. Modifikujte klasu `HelloWorld` tako da ispisuje poruku u novom fontu.

Grafičko programiranje: Interfejs I - Događaji

- Programiranje grafičkog interfejsa - određeno događajima (*events*) izvan njega
- Pri tome *događaji* dolaze od tastature (**pritisci na tipke**) i miša (**kretanje miša i *klik* određenom tipkom miša**).
- Svaki GUI (*graphical user interface*) program strukturiran je kao jedna beskonačna petlja u kojoj se:
  - skupljaju informacije o događajima koji su se desili
  - obavještavaju svi zainteresovani objekti.
- Objekt koji je zainteresovan za neki tip događaja reaguje na njega izvršavanjem određenog dijela koda (neke svoje metode).

Grafičko programiranje: Interfejs I - Događaji

- U Javinom programu koji implementira grafičko okruženje imaćemo tri vrste objekata: *izvore* događaja, *osluškivače* i same *događaje*
- **Izvori događaja** su komponente grafičkog interfejsa (npr. paneli, dugmad, klizači, itd.).
- **Osluškivači (*listeners*) događaja** su objekti koji reaguju na neku vrstu događaja.
- Sami događaji (*events*) su, naravno, **instance određenih klasa**.

Osluškivači – Registracija i Implementacija

- Komponenta koja je izvor događaja nekog tipa ima metodu kojom registruje osluškivače.
- Na taj način se ostvaruje **veza između izvora i osluškivača** i samo će osluškivači koji su registrovani biti obaviješteni o događaju. Opšti oblik metode ovog tipa je:

```
izvorDogađaja.addDogađajListener(objektSlušač)
```

- Klasa koja modeluje osluškivač nekog događaja mora da implementira interfejs listener odgovarajućeg tipa.
- Interfejs deklariše metodu koja će automatski biti pozvana kad se događaj dogodi.
- Pošto je generisanje događaja u potpunosti pod kontrolom korisnika programa ne možemo znati kada će tačno metoda koju objekt-osluškivač definiše biti pozvana.
- Ono što znamo je da će automatski biti pozvane metode svih registrovanih osluškivača.

Primjer 1: JButton

- Klasa `javax.swing.JButton` modeluje grafičku komponentu koja predstavlja dugme.
- **Ta komponenta generiše `ActionEvent`** (klasa `java.awt.event.ActionEvent`) **kojim se definiše *akcija* koju komponenta realizuje.**
- U slučaju `JButton` komponente `ActionEvent` će biti generisan svaki put kada kliknemo na dugme.
- **Osluškivač za `ActionEvent` mora da implementira interfejs `ActionListener`** (u paketu `java.awt.event`) koje definiše smo jednu metodu: `actionPerformed(ActionEvent e)`.
- Ta će metoda automatski biti pozvana kad se klikne na dugme.
- Registracija osluškivača:

```
ActionListener listener=....;   //osluškivč ActionEvent-a
JButton button = new JButton("OK"); // izvor ActionEvent-a
button.addActionListener(listener); // registracija osluš.
```

Nastavak...

- Implementacija osluškivača

```
class MyListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        // reakcija na klik na dugme ide ovdje
    }
}
```

- Sada će svaki put kad korisnik klikne na dugme sa labelom "OK" biti pozvana metoda `listener.actionPerformed`.
- Metoda kao argument automatski dobija objekt klase `ActionEvent` koji reprezentuje događaj.
- Programer samo kreira i registruje osluškivač, dok se poziv odgovarajućoj metodi (`actionPerformed`) dešava automatski.

Kompletan Primjer

- Napravimo sada kompletan primjer: program će otvoriti prozor u kome se nalaze tri dugmeta, označena jednom bojom. Klikom na dugme mijenja se boja pozadine prozora.
- Konstruktor za `JButton` uzima kao argument labelu u obliku stringa ili ikonu ili oboje. Na primjer:

```
JButton yellow = new JButton("Yellow");
```

- Nakon toga button se mora smjestiti na panel pomoću metode add koju JPanel nasljeđuje iz klase java.awt.Container. Na primjer:

```
class ButtonPanel extends JPanel
{
        public ButtonPanel()
        {
            JButton yellow = new JButton("Yellow");
            add(yellow);
        }
 }
```

Nastavak...

- Nakon što smo button stavili na panel moramo registrovati njegov osluškivač (`ActionListener`).
- Taj će osluškivač promijeniti boju panela za šta treba imati pristup metodi `setBackground` iz `JPanel` klase.
- Prema tome, treba staviti klasu koja implementira `ActionListener` interfejs unutar klase `ButtonPanel`, jer će ona tada moći dohvatiti sve njene metode.
- Kako klasu nećemo koristiti izvan `ButtonPanel` klase možemo je deklarisati kao privatnu.
- Konačno, registracija osluškivača ima ovaj oblik:

```
// ColorAction implementira ActionListener
 ColorAction yellowAction = new ColorAction(Color.YELLOW);
 yellow.addActionListener(yellowAction);
```

- Ovdje smo iskoristili klasu `java.awt.Color` koja implementira neke temeljne boje u obliku konstanti.

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

 public class Dogadjaj1 {
    public static void main(String[] args){
       ButtonFrame frame = new ButtonFrame();
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class ButtonFrame extends JFrame   {
   public ButtonFrame()       {
      setTitle("Test sa bojom pozadine");
      setSize(300,200);
      Container cp = getContentPane();
      ButtonPanel panel = new ButtonPanel();
      cp.add(panel);
   }
}
```

```
class ButtonPanel extends JPanel   {
    public ButtonPanel()      {
       // Tri buttona
       JButton yellow = new JButton("Zuta");
       JButton blue   = new JButton("Plava");
       JButton red    = new JButton("Crvena");

       // Dodajemo ih na panel
       add(yellow);
       add(blue);
       add(red);
```

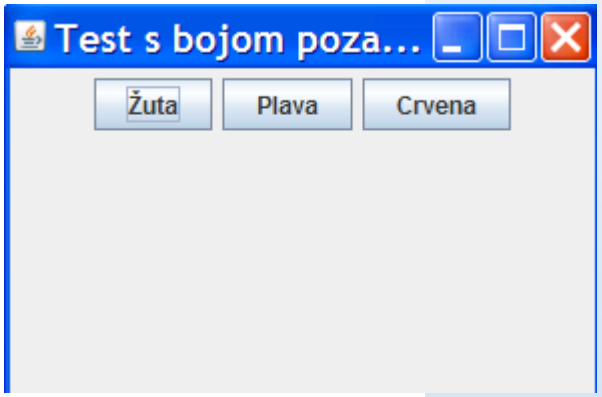
```
// Kreiramo osluškivače ...
        ColorAction yellowAction = new ColorAction(Color.YELLOW);
        ColorAction blueAction   = new ColorAction(Color.BLUE);
        ColorAction redAction    = new ColorAction(Color.RED);

        // registrujemo osluškivače
        yellow.addActionListener(yellowAction);
        blue.addActionListener(blueAction);
        red.addActionListener(redAction);
    }

    private class ColorAction implements ActionListener
    {
        public ColorAction(Color c) { backgroundColor=c; }

        public void actionPerformed(ActionEvent e) {
            // metoda iz JComponent klase
            setBackground(backgroundColor);
        }

        private Color backgroundColor;
    }
 }
```



Nastavak...

- Prethodni kod možemo pojednostaviti, jer se kreiranje svakog dugmeta sastoji od četiri akcije:
  - Instanciranje JButtona;
  - Dodavanje na panel (add);
  - Konstrukcija ActionListener objekta;
  - Registracija ActionListener objekta.
- Sve to možemo obaviti u jednoj metodi. Nova metoda neka se zove `makeButton`:

```
void makeButton(String labela, final Color bojaPozadine)
    {
        JButton baton = new JButton(labela);
        add(baton);
        baton.addActionListener(new ActionListener()
                {
                     public void actionPerformed(ActionEvent e) {
                     // metoda iz JComponent klase
                     setBackground(bojaPozadine);
                }
                });
        // Referenca baton sada nestaje, ali to nije bitno jer je čuva ButtonPanel
    }
```

Nastavak...

- Konstruktor klase `ButtonPanel` sada je vrlo jednostavan:

```
public ButtonPanel()
    {
        // Tri buttona
        makeButton("Yellow", Color.YELLOW);
        makeButton("Blue", Color.BLUE);
        makeButton("Red", Color.RED);
    }
```