OBJEKTNO PROGRAMIRANJE 2

Oznaka predmeta: OP2

Predavanje broj: 08

Nastavna jedinica: JAVA

Nastavne teme:

Labela. Programsko dugme. Polja za potvrdu i radio dugmad. Liste i padajuće liste. Polje za tekst i površina za tekst. SWING. Osnovna struktura swing aplikacije. Lookandfeel. Dodavanje komponenti. Rokovanje događajima. Osluškivači preko anonimne klase. Korišćenje standardnih komponenti.

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

Eckel B., *Thinking in Java*, 2nd edition, Prentice-Hall, New Jersey 2000.

Cay S. Horstmann and Gary Cornell*: "Core Java, Advanced Features", Vol. 2, Prantice Hall, 2013.*

The Java Tutorial, Sun Microsystems 2001. *http://java.sun.com*

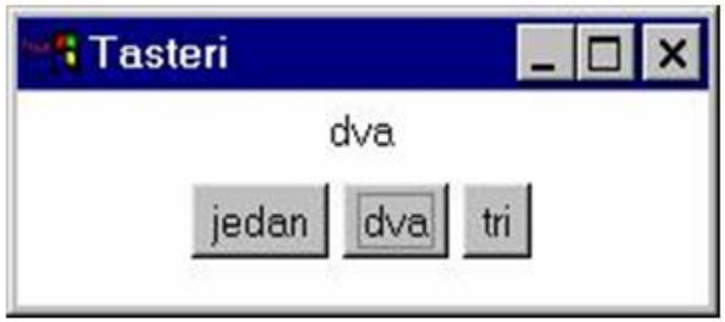
Branko Milosavljević, Vidaković M, *Java i Internet programiranje*, GInT, Novi Sad 2002.

Labela i programsko dugme

```
import java.awt.*;
import java.awt.event.*;
public class PrimerTastera extends Frame implements ActionListener {
  Label labela = new Label("Pocetni tekst");
  public PrimerTastera() {
    super("Tasteri");
    dodajKomponente();
    setSize(250,100);
    setVisible(true);
  }
  void dodajKomponente() {
    add("North",labela);
    labela.setAlignment(Label.CENTER);
    Button jedan=new Button("jedan"), dva=new Button("dva"),
    tri=new Button("tri");
    tri.setActionCommand("Kraj");
    jedan.addActionListener(this);
    dva.addActionListener(this);
    tri.addActionListener(this);
    Panel panel = new Panel();
    panel.add(jedan); panel.add(dva); panel.add(tri);
    add("Center",panel);
  }
```

Labela i programsko dugme

```
  public void actionPerformed(ActionEvent e) {
    labela.setText(e.getActionCommand());
    if(labela.getText().equals("Kraj")) dispose();
  }
  public static void main(String args[]){
    PrimerTastera prozor = new PrimerTastera();
  }
}
```




```
//dodajte windowClosing
```

Polja za potvrdu i radio-dugmad

- Klasa Checkbox omogućuje kreiranje
  - polja za potvrdu i
  - radio-dugmadi
- Objektu polja za potvrdu:
  - je pridružena jedna labela
  - stanje objekta je tipa boolean.
- Klasa sadrži metode za dohvatanje i modifikovanje stanja i labele.
- Promena stanja prouzrokuje ItemEvent događaj:
  - događaj se obrađuje metodom itemStateChanged()

- Klasa CheckboxGroup se koristi za grupisanje Checkbox objekata.
- Grupisani Checkbox objekti se ponašaju kao radio-dugmad.
  - Ako nije u grupi, objekat se ponaša kao obično polje za potvrdu.

Polja za potvrdu i radio-dugmad

```
import java.awt.*;
import java.awt.event.*;
public class PrimerPoljaZaPotvrdu extends Frame implements ItemListener
{
  Label labela = new Label("Pocetni tekst");
  Checkbox poljeZaPotvrdu[] = new Checkbox[4];
  public PrimerPoljaZaPotvrdu() {
    super("Polja za potvrdu");
    dodajKomponente(); setSize(250,120); setVisible(true);
  } //dodajte zatvaranje aplikacije
  void dodajKomponente() {
    add("North",labela); labela.setAlignment(Label.CENTER);
    Panel panel = new Panel();
    Panel panel1 = new Panel(); panel1.setLayout(new GridLayout(2,1));
    Panel panel2 = new Panel(); panel2.setLayout(new GridLayout(2,1));
    poljeZaPotvrdu[0] = new Checkbox("jedan");
    poljeZaPotvrdu[1] = new Checkbox("dva");
    CheckboxGroup grupa = new CheckboxGroup();
    poljeZaPotvrdu[2] = new Checkbox("tri",grupa,false);
    poljeZaPotvrdu[3] = new Checkbox("cetiri",grupa,false);
    for(int i=0;i<4;++i) poljeZaPotvrdu[i].addItemListener(this);
    for(int i=0;i<2;++i) panel1.add(poljeZaPotvrdu[i]);
    for(int i=2;i<4;++i) panel2.add(poljeZaPotvrdu[i]);
```

Polja za potvrdu i radio-dugmad

```
    panel.add(panel1); panel.add(panel2); add("Center",panel);
  }
  public void itemStateChanged(ItemEvent e) {
    String tekst = "";
    for(int i=0;i<4;++i) {
      if(poljeZaPotvrdu[i].getState())
        tekst+=poljeZaPotvrdu[i].getLabel()+" ";
    }
    labela.setText(tekst);
    if(tekst.equals("cetiri ")) dispose();
  }
  public static void main(String args[]){
    PrimerPoljaZaPotvrdu prozor = new PrimerPoljaZaPotvrdu ();
  }
}
```

Liste i padajuće liste

- Klasa Choice realizuje padajuću listu
  - iz koje se može izabrati samo jedna stavka
  - Metodi klase Choice omogućuju
    - modifikaciju elemenata liste i
    - upit o njihovom statusu
- Klasa List realizuje listu
  - iz koje se može izabrati jedan ili više redova
  - Metodi klase List omogućuju
    - modifikaciju elemenata liste i
    - upit o njihovom statusu

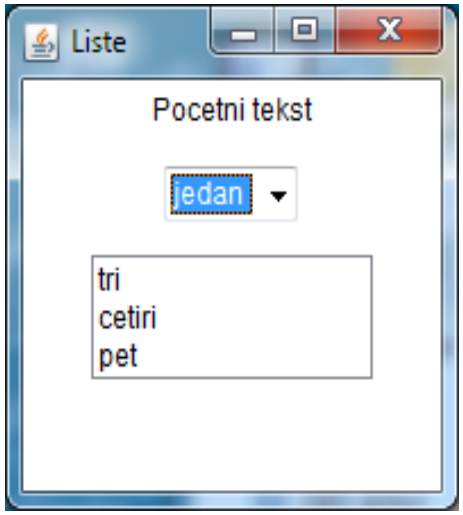
```
import java.awt.*;
import java.awt.event.*;
public class PrimerListe extends Frame implements ItemListener{
  Label labela = new Label("Pocetni tekst");
  Choice izbor = new Choice();
  List lista = new List(3,true);
  public PrimerListe() { super("Liste");
              dodajKomponente(); setSize(200,200); setVisible(true);   }
//dodajte zatvaranje aplikacije
```

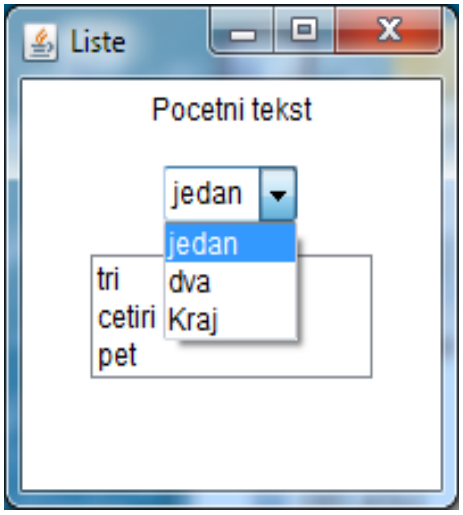
Primer liste i padajuće liste

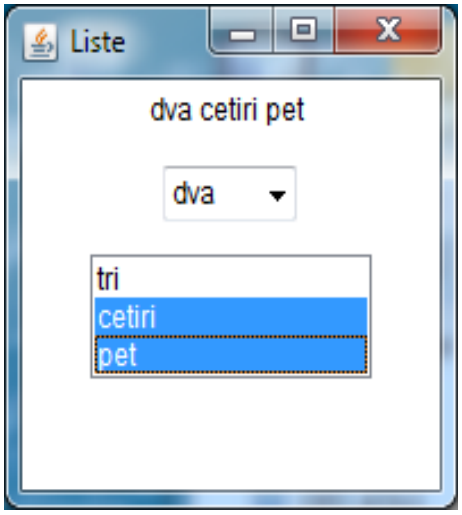
```
void dodajKomponente() {
  add("North",labela);
  labela.setAlignment(Label.CENTER);
  izbor.addItem("jedan");
  izbor.addItem("dva");
  izbor.addItem("Kraj");
  izbor.addItemListener(this);
  lista.add("tri");
  lista.add("cetiri");
  lista.add("pet");
  lista.addItemListener(this);
  Panel panel = new Panel(),
        panel1= new Panel(),
        panel2 = new Panel();
  panel1.add(izbor);
  panel2.add(lista);
  panel.add(panel1);
  panel.add(panel2);
  add("Center",panel);
}
```

Primer liste i padajuće liste

```
  public void itemStateChanged(ItemEvent e) {
    String tekst = izbor.getSelectedItem() + " ";
    if(tekst.equals("Kraj ")) dispose();
    for(int i=0;i<3;++i)
      if(lista.isIndexSelected(i))
        tekst += lista.getItem(i) + " ";
    labela.setText(tekst);
  }
  public static void main(String args[]){
    PrimerListe prozor = new PrimerListe (); }
}
```







Polje za tekst i površina za tekst

- Klasa TextField omogućava prikaz i unos jedne linije teksta:
  - može se definisati karakter koji se pojavljuje umesto unesenog teksta (za lozinke)
  - za definisanje alternativnog karaktera koristi se metod setEchoCharacter()

- Klasa TextArea omogućuje prikaz i unos više linija teksta:
  - površina za tekst obezbeđuje horizontalan i vertikalan klizač za ”proklizavanje” teksta

- Klasa TextComponent je superklasa za tekst klase:
  - TextField i
  - TextArea

- Metod setEditable()
  - omogućava da tekst objekti budu definisani samo za čitanje

Polje za tekst i površina za tekst

```
import java.awt.*;
import java.awt.event.*;
public class PrimerTekst extends Frame implements ActionListener{
  TextField poljeZaTekst = new TextField("Uneti tekst ovde.");
  TextArea prostorZaTekst = new TextArea("I bice umetnut ovde!");
  public PrimerTekst() {
    super("Tekst");
    addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent we){
        System.exit(0);
      }
    });
    dodajKomponente();
    setSize(200,150);
    setVisible(true);
  }
  void dodajKomponente() {
    add("North",poljeZaTekst);
    add("Center",prostorZaTekst);
    poljeZaTekst.addActionListener(this);
  }
```

Polje za tekst i površina za tekst

```
  public void actionPerformed(ActionEvent e) {
    String tekst = poljeZaTekst.getText();
    if(tekst.equals("Kraj")) dispose();
    prostorZaTekst.insert(tekst+"\n",0);
  }
  public static void main(String args[]){
    PrimerTekst prozor = new PrimerTekst ();
  }
}
```

AWT, SWING

- Programski jezik Java je, u svojoj inicijalnoj verziji, posedovao biblioteku komponenti za izgradnju grafičkog korisničkog interfejsa (GUI) zvanu *Abstract Window Toolkit* (AWT).
- U pitanju je biblioteka koja se zasniva na korišćenju komponenti korisničkog interfejsa koje su dostupne na platformi na kojoj se program pokreće (*Windows*, *Motif*, *Macintosh*, itd).
- Implementacija AWT komponenti različita je za svaki operativni sistem.
- Java klase koje predstavljaju AWT komponente koristile su u velikoj meri *native* programski kod koji je vršio interakciju sa operativnim sistemom.
  - AWT klase u *Windows* distribuciji Java virtuelne mašine koriste *awt.dll* datoteku.
- Aplikacije koje koriste AWT komponente izgledaju različito na različitim operativnim sistemima.
- Ideja je bila da se napravi biblioteka koja je bazirna na Java-i, tako da se izgubi ograničenje zajedničkih komponenti operativnih sistema.

SWING

- Ovakav koncept ima za posledicu da je za skup GUI komponenti bilo neophodno izabrati samo one komponente koje postoje u svim operativnim sistemima na kojima će se Java programi izvršavati.
  - Ovakav skup komponenti je vrlo siromašan.
- Umesto da se postigne cilj da Java GUI aplikacije izgledaju "jednako kao i sve druge" aplikacije, postiglo se da one izgledaju "jednako osrednje" na svim platformama zbog siromašnog skupa komponenti od kojih mogu biti sačinjene.
- U vreme kada je bila aktuelna Java verzija 1.1, počet je razvoj na novoj biblioteci GUI koponenti koja je imala drugačiji koncept:
  - kompletna biblioteka je napisana u Javi, što znači da se komponente samostalno "iscrtavaju" na ekranu umesto da ih iscrtava operativni sistem.
  - posledica toga je da GUI aplikacije izgledaju isto na svim operativnim sistemima i da nema ograničenja na broj i tip GUI komponenti koje će ući u biblioteku.
- Naziv biblioteke u toku njenog razvoja bio je *Swing*, i to ime se zadržalo i kasnije.

SWING

- Biblioteka je zamišljena tako da izgled komponenti na ekranu bude promenljiv, zavisno od izabrane "teme".
- Najčešće korišćeni *look-and-feel* moduli:
    - *Windows* (sve komponente izgledaju kao odgovarajuće *Windows* komponente),
    - *Motif* (GUI okruženje na UNIX-u) i
    - *Metal* (izgled svojstven samo Java aplikacijama)
- Kasnije su se pojavili *look-and-feel* dodaci sa Macintosh izgledom, itd.
- Promena izgleda aplikacije može da se obavi čak i za vreme izvršavanja programa.
- Iako se *Swing* biblioteka može koristiti i sa Java verzijom 1.1 (uz dodavanje biblioteke u CLASSPATH), sve mogućnosti biblioteke su dostupne tek od verzije 1.2.
- Od verzije 1.2 ova biblioteka je proglašena za standard za razvoj korisničkog interfejsa u Java aplikacijama, dok je AWT zadržan zbog kompatibilnosti sa starijim programima.
- *Swing* je postao sastavni deo veće biblioteke nazvane *Java Foundation Classes* (JFC).

Osnovna struktura SWING aplikacije

- Svaka Java aplikacija počinje svoje izvršavanje metodom *main*. Tako i GUI aplikacija počinje svoje izvršavanje ovom metodom, ali se najčešće tom prilikom odmah inicijalizuje i glavni prozor aplikacije koji se potom prikaže na ekranu.

```
import javax.swing.*;
public class MainFrame extends JFrame {
  public MainFrame() {
    setSize(300, 200);
    setTitle("My First GUI App");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}

public class MyApp {
  public static void main(String[] args) {
    MainFrame mf = new MainFrame();
    mf.setVisible(true);
  }
}
```

Osnovna struktura SWING aplikacije

- Aplikacija se sastoji iz dve klase:
  - klasa MainFrame iscrtava JFrame
  - klasa *MyApp* pokreće aplikaciju.
- U okviru *main* metode kreira se objekat klase *MainFrame* (što predstavlja inicijalizaciju glavnog prozora aplikacije) i zatim se taj prozor prikaže na ekranu (poziv metode *setVisible*).
- Klasa *MainFrame* nasleđuje klasu *JFrame*, što je standardan način za definisanje novih prozora.
- Klasa *JFrame* je deo *Swing* biblioteke smeštene u paket *javax.swing*.
- Komponente *Swing* korisničkog interfejsa po pravilu počinju velikim slovom J.
- U okviru konstruktora klase *MainFrame* se postavlja veličina prozora u pikselima (*setSize*) i naslov prozora (*setTitle*).
- Ako se pritisne dugme za aztvaranje aplikacije, ona bi podrazumevano bila samo uklonjena sa ekrana (dispose), te je potrebno eksplicitno staviti da se aplikacija zatvori.

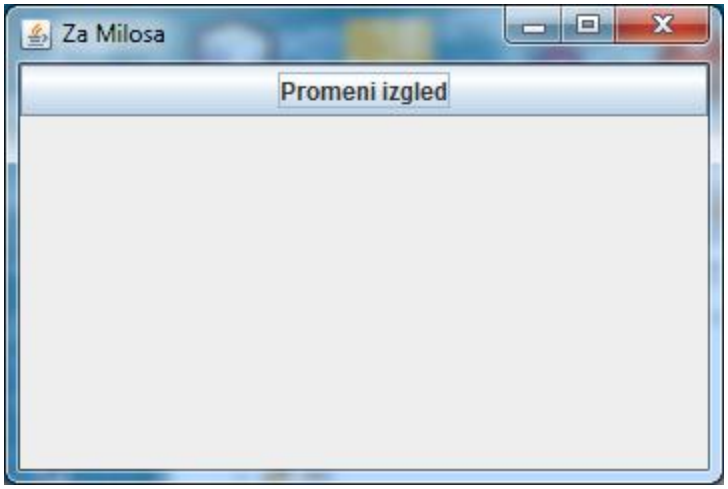
```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

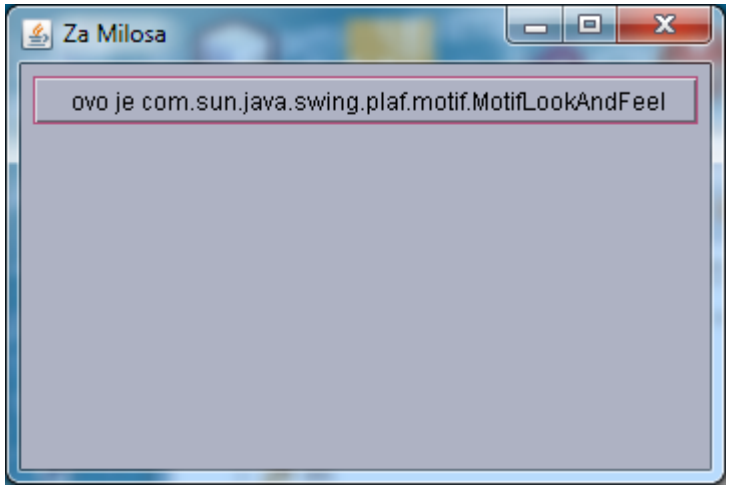
Primer lookandfeel

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ZaMilosa extends JFrame implements ActionListener {
  private int tipsminke;
  private String str;
  private JButton jdugme;
  public ZaMilosa() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    tipsminke = 0;
    str = "";
    jdugme = new JButton("Promeni izgled");
    getContentPane().setLayout(new BorderLayout());
    getContentPane().add("North",jdugme);
    jdugme.addActionListener(this);
    setSize(500,500);
    setTitle("Za Milosa");
    setVisible(true);
  }
  public void actionPerformed(ActionEvent e) {
    tipsminke++;
    tipsminke%=3;
```

Primer lookandfeel

```
    switch(tipsminke) {
    case 0:       str = "javax.swing.plaf.metal.MetalLookAndFeel";break;
    case 1:str= "com.sun.java.swing.plaf.windows.WindowsLookAndFeel";break;
    default:str="com.sun.java.swing.plaf.motif.MotifLookAndFeel";
    }
    jdugme.setText(" ovo je "+str);
    try {
UIManager.setLookAndFeel(str);SwingUtilities.updateComponentTreeUI(this);
    } catch (Exception excep) {}
  }
  public static void main(String[] args) {
    ZaMilosa zm = new ZaMilosa();
  }
}
```



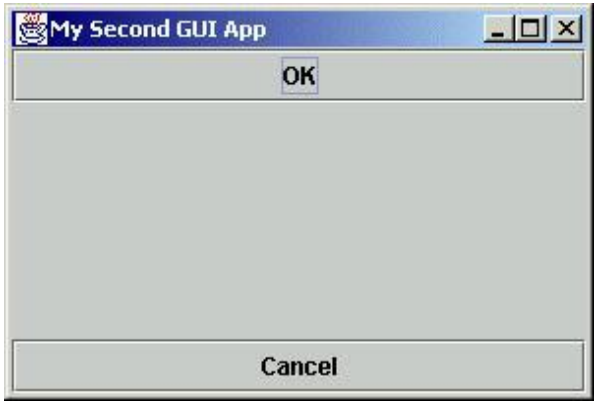


SWING: Dodavanje komponenti na prozor

```
import java.awt.*;
import javax.swing.*;
public class MainFrame extends JFrame {
  // elementi na formi su najčešće privatni atributi klase
  private JButton bOK     = new JButton("OK");
  private JButton bCancel = new JButton("Cancel");
  public MainFrame() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(300, 200);
    setTitle("My Second GUI App");
    // dodaju se komponente na formu:
    getContentPane().add(bOK, BorderLayout.NORTH);
    getContentPane().add(bCancel, BorderLayout.SOUTH);
  }
}
public class MyApp {
      public static void main(String[] args) {
        MainFrame mf = new MainFrame();
        mf.setVisible(true);
      }
}
```

SWING: Dodavanje komponenti na prozor

- Prozoru iz prethodnog primera dodata su dva dugmeta, predstavljena klasom *JButton*.
- Ovakve komponente korisničkog interfejsa najčešće se definišu kao atributi prozorske klase, u ovom slučaju *MainFrame*.
- U konstruktoru je dodato postavljanje komponenti na prozor – pozivi metode *add*.
- Komponente se mogu smestiti samo unutar nekog *kontejnera* – objekta koji je namenjen za prihvat komponenti.
  - Svaki kontejner ima sebi asociran *layout manager* (iz AWT biblioteke).
- Svaki prozor već ima svoj kontejner, koga možemo dobiti pozivom metode *getContentPane*.



SWING: Rukovanje događajima

- Osluškivači događaja implementira odgovarajuće *nazivListener* interfejse a objekt koji generiše događaj je predstavljen *nazivEvent* objektom.

```
import java.awt.*;
import java.awt.event.*;
public class MyListener  implements ActionListener {
  public void actionPerformed(ActionEvent ev) {
    System.exit(0);
  }
}...
```

- Interfejs *ActionListener* definiše jednu metodu, *actionPerformed*. Njen parametar je objekat klase *ActionEvent* koji bliže opisuje događaj.
- U prikazanom primeru osluškivač će, kada se događaj dogodi, zatvoriti aplikaciju (metoda *exit*).
- *Listener* mehanizam je preuzet iz AWT biblioteke tako da su svi događaji definisani u starijim paketima *java.awt* i *java.awt.event*.

SWING: Rukovanje događajima

- Osluškivač događaja (instanca neke *Listener* klase) se pridružuje onoj komponenti korisničkog interfejsa za koju želimo da reaguje na taj događaj.
- Pridruživanje se obavlja metodom *add**naziv**Listener* koju ima komponenta.

  Primer *MainFrame* klase iz prethodnih primera koja je modifikovana tako što je dugmadima dodat prethodno prikazani osluškivač.

```
 import java.awt.*; import java.awt.event.*;
import javax.swing.*;
      public class MainFrame extends JFrame {
        private JButton bOK = new JButton("OK");
        private JButton bCancel = new JButton("Cancel");
        public MainFrame() {
          setSize(300, 200);    setTitle("My Second GUI App");
          getContentPane().setLayout(new FlowLayout());
          getContentPane().add(bOK);
          getContentPane().add(bCancel);
          // dodaju se reakcije na dogadjaje dugmadima
          bOK.addActionListener(new MyListener());
          bCancel.addActionListener(new MyListener());
        }
      }...
```

- Ovakav prozor sadrži dva dugmeta koja će, na klik mišem, reagovati na isti način, koriste isti *Listener* (MyListener).

Osluškivači preko anonimne klase

- Prozori često znaju biti pretrpani komponentama koje obrađuju više vrsta događaja (prozorska klasa sadrži desetke komponenti i *Listener* klasa.
- Definisati pedesetak *Listener* klasa samo za jedan prozor može učiniti program nepreglednim, te se *Listener* klase najčešće definišu kao unutrašnje klase u okviru prozorske klase ili neimenovane klase.

```
ActionListener a = (new ActionListener() {
  public void actionPerformed(ActionEvent ev) {
    System.exit(0);
  }
});
```

- U pitanju je definicija reference *a* na objekat klase koja implementira *ActionListener* interfejs. Ime klase nigde nije navedeno.
- Sve što je iz ove konstrukcije potrebno je referenca na objekat, koja se može iskoristiti na sledeći način:

```
bCancel.addActionListener(a);
```

  - definisana je *Listener* klasa i njena instanca (osluškivač) pridružen je dugmetu *bCancel* ili u jednom iskazu:

```
bCancel.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent ev) {
    System.exit(0);
  }
});
```

Osluškivači preko anonimne klase

Prethodna sintaksa je (uz malo navikavanja) preglednija od desetina odvojenih datoteka u kojima se nalaze definicije osluškivača.

```
import java.awt.*; import java.awt.event.*; import javax.swing.*;
public class SwingMenjajBoju extends JFrame {
  private JButton bOK     = new JButton("OK");
  private JButton bCancel = new JButton("Cancel");
  public SwingMenjajBoju() {
    setSize(300, 200);   setTitle("My GUI App");
    getContentPane().setLayout(new FlowLayout());
    getContentPane().add(bOK); getContentPane().add(bCancel);
    bOK.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent ev) {
        int r = (int)(Math.random()*256);
        int g = (int)(Math.random()*256);
        int b = (int)(Math.random()*256);
        bOK.setBackground(new Color(r, g, b));
      }
    });
    bCancel.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent ev) {
        System.exit(0);
      }
    });  } . . . //dodajte potreban kod
```

Neke od najčešće korišćenih komponenti SWINGa

Klasa	Opis
ButtonGroup	grupisanje radio button-a; nije vidljiva komponenta
JButton	dugme
JCheckBox	check box
JComboBox	combo box
JDialog	dijalog
JFrame	okvir
JLabel	labela
JList	list box
JMenu	meni
JMenuBar	linija menija
JMenuItem	stavka menija
JOptionPane	prozor poruke ili prompta
JPanel	komponenta koja je kontejner za druge komponente
JRadioButton	radio button
JTabbedPane	kartice (tabs); kartice se na ovu komponentu dodaju kao JPanel-i
JTextArea	višelinijsko polje za unos teksta (memo)
JTextField	jednolinijsko polje za unos teksta

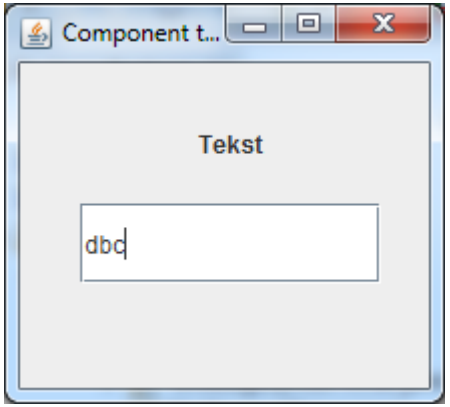
SWING: primeri korišćenja standardnih komponenti

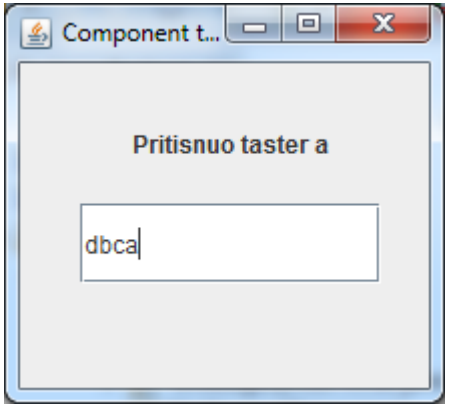
- Prvi primer ilustruje reagovanje na pritisnut taster (*KeyEvent*) u tekstualnom polju za unos (*JTextField*).
  - Komponenta za unos teksta *tf* ima svoj osluškivač događaja, objekat klase *Reakcija*.
  - Klasa *Reakcija* ne implementira interfejs *KeyListener*, već nasleđuje klasu *KeyAdapter*. Sada je dovoljno redefinisati samo metodu *keyReleased.*
  - Prilikom otpuštanja pritisnutog tastera dok je fokus na *tf* komponenti, poziva se osluškivač klase *Reakcija*. Ukoliko je pritisnut taster A, tekst labele *l* se menja u "Pritisnuo taster a", a inače se menja u "Tekst".

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

public class JTextField_Test extends JFrame {
  JTextField tf = new JTextField(30);
  JLabel lab = new JLabel("Tekst");

  public JTextField_Test() {
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(220, 100);
    setTitle("Component test");
```

SWING: primeri korišćenja standardnih komponenti

```
    getContentPane().setLayout(null);
    lab.setHorizontalAlignment(SwingConstants.CENTER);
    lab.setLocation(30,30);lab.setSize(150,20);
    getContentPane().add(lab);
    tf.setLocation(30, 70);tf.setSize(150, 40);
    getContentPane().add(tf);
    tf.addKeyListener(new Reakcija());
  }
  /** Rukovalac dogadjajima definisan kao inner klasa */
  class Reakcija extends KeyAdapter {
    public void keyReleased(KeyEvent e) {
      if (e.getKeyCode() == KeyEvent.VK_A)
        lab.setText("Pritisnuo taster a");
      else
      lab.setText("Tekst");
    }
  }
  public static void main(String[] args) {
    JTextField_Test jtf_t = new JTextField_Test();
    jtf_t.setVisible(true);
  }
}
```

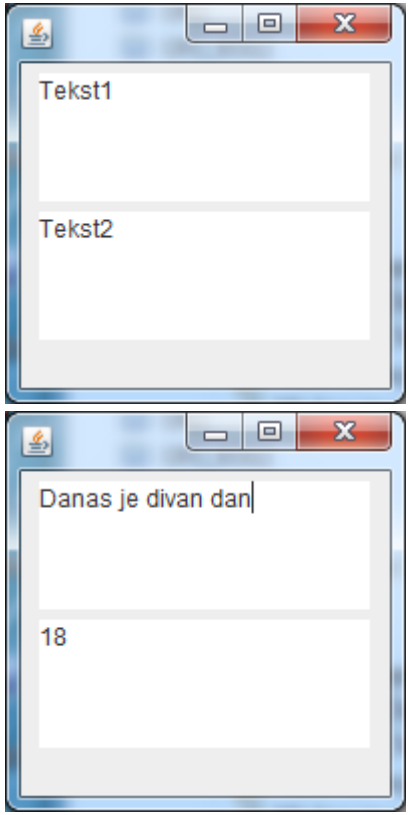




SWING: primeri korišćenja standardnih komponenti

- Događaj pomeranja kursora (*CaretEvent*) u gornjem *JTextArea* polju upisuje podatak o položaju kursora (getDot()) u donji *JTextArea*.
- Za osluškivanje ovakvog događaja implementira se *CaretListener* interfejs, a metoda koja se poziva prilikom pomeranja kursora je *caretUpdate*.

```
import java.awt.*;
import javax.swing.event.*;
import javax.swing.*;
public class JTextArea_Test extends JFrame {
  JTextArea ta1;
  JTextArea ta2;
  public JTextArea_Test() {
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(200, 200);
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());
    ta1 = new JTextArea("Tekst1", 4, 15);
    ta1.addCaretListener(new Reakcija());  cp.add(ta1);
    ta2 = new JTextArea("Tekst2", 4, 15);  cp.add(ta2);
  }
  class Reakcija implements CaretListener {
    public void caretUpdate(CaretEvent e) {
      ta2.setText("" + e.getDot());
    }   } } //DODAJTE POTREBAN KOD
```



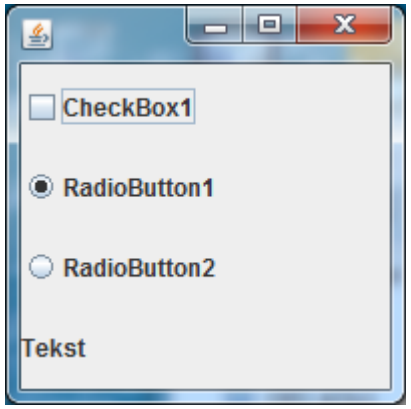
SWING: primeri korišćenja standardnih komponenti

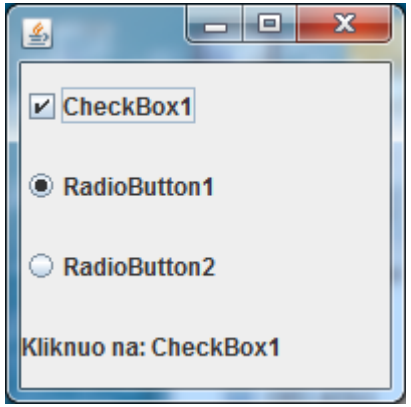
- Obrađuje se događaj izbora stavke (*ItemEvent)* u *check box* polju (*JCheckBox*) i *radio button* polju (*JRadioButton*).
- Metoda *getItem* klase *ItemEvent* vraća referencu na onaj objekat tj. komponentu koja je izabrana (stavka).
- Prilikom izbora stavke labeli se menja tekst u “Odabrao stavku: “ + <tekst komponente na koju je kliknuto>.
- Da bi dva *radio button*-a radila u paru, tj. da bi selekcija jednog izazvala deselekciju drugog, potrebno ih je staviti ih u istu grupu metodom add *ButtonGroup* objekta (nema prefiks J).

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class JCheckBox_Test extends JFrame {
  JCheckBox cb1 = new JCheckBox("CheckBox1");
  ButtonGroup group = new ButtonGroup();
  JRadioButton rb1 = new JRadioButton("RadioButton1", true);
  JRadioButton rb2 = new JRadioButton("RadioButton2", false);
  JLabel lab;
  public JCheckBox_Test() {
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(200, 200);
    Container cp = getContentPane();
```

SWING: primeri korišćenja standardnih komponenti

```
    cp.setLayout(new GridLayout(4,1));
    lab = new JLabel("Tekst");
    cp.add(cb1);
    group.add(rb1); // dodajemo radio button-e u grupu
    group.add(rb2); // kako bi radili u paru
    cp.add(rb1);
    cp.add(rb2);
    cb1.addItemListener(new Reakcija());
    Reakcija r = new Reakcija();
    rb1.addItemListener(r);
    rb2.addItemListener(r);
    cp.add(lab);
  }
  class Reakcija implements ItemListener {
    public void itemStateChanged(ItemEvent e) {
      lab.setText("Kliknuo na: " +
        ((AbstractButton) e.getItem()).getText());
    }
  }
  public static void main(String[] args) {
    JCheckBox_Test jcb_t = new JCheckBox_Test();
    jcb_t.setVisible(true);
}}
```





SWING: primeri korišćenja standardnih komponenti

- Događaj izbora stavke (*ItemEvent*) u *combo box*-u (*JComboBox*), gde će se metodom *getSource* klase *ItemEvent* vratiti objekat koji je izvor događaja, proveriti da li je to instanca klase *JComboBox*, i ako jeste tekst labele se postavlja na izabrani tekst u *combo box*-u dobijen metodom *getSelectedItem* klase *JComboBox*.

```
import java.awt.*; import java.awt.event.*; import javax.swing.*;
public class JComboBox_Test extends JFrame {
  String[] items = { "Prva opcija", "Druga opcija", "Treca opcija" };
  JComboBox c = new JComboBox();
  JLabel lab = new JLabel("Labela");
  public JComboBox_Test() {
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);setSize(200,200);
    for (int i = 0; i < items.length; i++) c.addItem(items[i]);
    c.addItemListener(new Reakcija());
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());
    cp.add(c);      cp.add(lab);
  }
  class Reakcija implements ItemListener {
    public void itemStateChanged(ItemEvent e) {
      if (e.getSource() instanceof JComboBox)
        lab.setText((String) c.getSelectedItem());
    }   }} //dodajte potreban kod
```