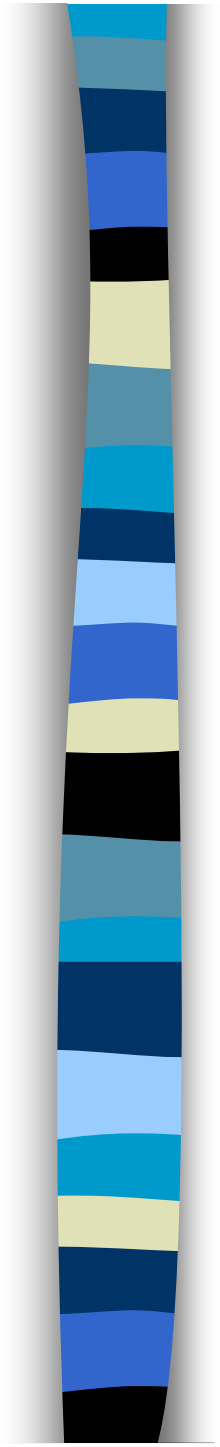


Razvoj vizuelnih aplikacija

Pregled Swing komponenti

- Najčešće korišćene specijalizovane komponente korisničkih interfejsa
- Mahom nasleđuju zajedničku osnovnu klasu **JComponent**
- Upotreba svake komponente obuhvata:
  - Kreiranje objekta komponente
  - Uključivanje objekta u neki kontejner
  - Komponenti se prijavljuje osluškivač koji će reagovati na odgovarajuće događaje

JLabel i ImageIcon

- Labele su komponente koje služe za prikaz tekstualnog ili grafičkog sadržaja na ekranu.
- Labele se, u vidu opisa ili uputstva, najčešće koriste uz druge komponente ulaznog tipa, kao što su, na primer, polja za unos teksta.
- Labele mogu da sadrže:
  - Samo tekst
  - Samo sliku
  - I tekst i sliku.
- U okviru prostora koji je labeli dodeljen od strane raspoređivača, njen sadržaj može biti poravnat po levoj ili desnoj ivici labele ili može biti centriran. Inicijalno se upotrebljava poravnanje po levoj ivici.

JLabel i ImageIcon

- Labele same po sebi ne generišu i ne opažaju događaje, osim onih koje nasleđuju od superklase JComponent.
- Korisnik ne može da menja sadržaj labele. Međutim, sadržaj labele je moguće promeniti iz samog programa, pozivom metoda labele **setText** ili **setIcon**, što se može učiniti, npr. u sklopu reakcije na događaj koji je proizvela neka druga komponenta.
- Za prikazivanje grafičkog sadržaja, konstruktori labele prihvataju objekte koji implementiraju interfejs **Icon**.
- **ImageIcon** je klasa koja implementira interfejs **Icon**, a služi za prikazivanje slika u GIF ili JPEG formatu. Konstruktoru ove klase se prosleđuje string sa putanjom i imenom odabranog fajla.

JLabel i ImageIcon

- Dodatnim parametrom **Jlabel.CENTER** koji je upotrebljen pri kreiranju sve tri labele iz primera, zadato je centralno horizontalno poravnanje celokupnog sadržaja labele.
- Metodi:
  - `setVerticalTextPosition i`
  - `setHorizontalTextPosition`
- određuju relativnu poziciju teksta u odnosu na sliku.
- Njihova upotreba ima smisla samo u slučaju labele koja sadrži i tekst i sliku.

JLabel i ImageIcon - primer

Primer – tri labele

```
Icon icon = new ImageIcon("slika.gif");
JLabel label1 = new JLabel("Samo tekst", JLabel.CENTER);
JLabel label2 = new JLabel(icon, Jlabel.CENTER); // Samo slika
JLabel label3 = new JLabel("Slika i tekst", icon, JLabel.CENTER);
label3.setVerticalTextPosition(JLabel.BOTTOM);
label3.setHorizontalTextPosition(JLabel.CENTER);
```

JToolTip

- **JToolTip** je string koji služi kao uputstvo ili dodatno objašnjenje neke komponente.
- Pojavljuje se automatski kada pokazivač miša miruje nekoliko sekundi iznad komponente, a nestaje čim se pokazivač miša pomeri.
- Da bismo JToolTip dodelili nekoj komponenti, potrebno je samo pozvati metod komponente **setToolTipText.**

Primer

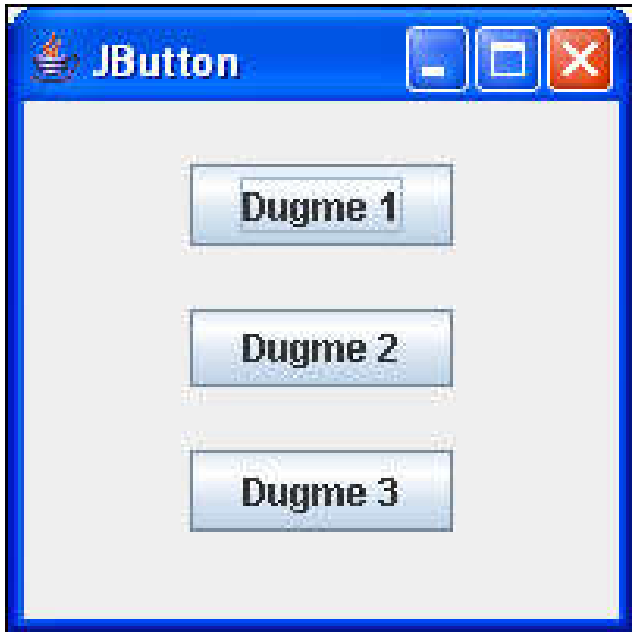
```
label1.setToolTipText("Samo tekst!");
label2.setToolTipText("Samo slika!");
label3.setToolTipText("I tekst i slika!");
```

Dugmad

- Dugme je komponenta koja generiše događaje tipa **ActionEvent** kada korisnik klikne mišem na njegovu površinu. Poput labela, i dugmad mogu da prikazuju tekst, sliku ili oboje.
- Bilo kom dugmetu se može dodeliti mnemonik pozivom metoda **setMnemonic** (kombinacija tastera sa tastature na koju reaguje).
- Pozivom metoda **setActionCommand**, svakom dugmetu se može dodeliti simbolični naziv koji može služiti za njegovu identifikaciju u programu.
- Pozivom metoda **setEnabled** dugme se može učiniti aktivnim ili neaktivnim u zavisnosti od vrednosti logičkog parametra koji se prosleđuje ovom metodu.
- U *Swing* biblioteci postoji nekoliko vrsta dugmadi koja su sva izvedena iz zajedničke osnovne klase **AbstractButton**. To su **JButton**, **JCheckBox**, **JRadioButton**, **JToggleButton**, **JMenuItem**, **JCheckBoxMenuItem** i **JRadioButtonMenuItem.**

Dugmad - JButton (1/3)

- Klasom **JButton** je predstavljeno uobičajeno dugme
- Kada korisnik klikne na njegovu površinu generiše se događaj tipa **ActionEvent** koji se prosleđuje prijavljenom osluškivaču tipa **ActionListener**
- Kada u programu postoji nekoliko dugmadi kojima je prijavljen isti osluškivač, unutar klase programiranog osluškivača veoma je važno odrediti koje dugme je generisalo događaj



Dugmad - JButton (2/3)

- Ako je klasa osluškivača realizovana kao unutrašnja klasa i imamo sačuvane reference na objekte dugmadi - identifikacija se može ostvariti poređenjem: **e.getSource() == mojeDugme1**

Primer - Identifikacija dugmadi u telu osluškivača poređenjem referenci

```
JButton jb1 = new JButton("Dugme 1");
JButton jb2 = new JButton("Dugme 2");
JButton jb3 = new JButton("Dugme 3");

ActionListener al = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    Object source = e.getSource();
    if (source == jb1)
      System.out.println("Pritisnuto dugme 1");
    else if (source == jb2)
      System.out.println("Pritisnuto dugme 2");
    else
      System.out.println("Pritisnuto dugme 3");
  }
};
jb1.addActionListener(al);
jb2.addActionListener(al);
jb3.addActionListener(al);
```

Dugmad - JButton (3/3)

Primer - Identifikacija pomoću simboličnih naziva

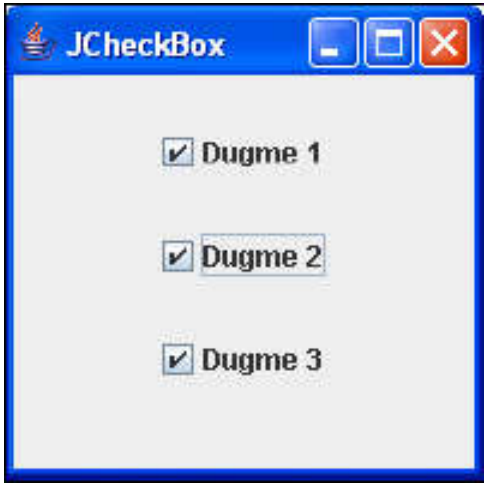
```
JButton jb1 = new JButton("Dugme 1");
// postavljanje simboličkog naziva dugmeta
jb1.setActionCommand("prvo");
JButton jb2 = new JButton("Dugme 2");
jb2.setActionCommand("drugo");
JButton jb3 = new JButton("Dugme 3");
jb3.setActionCommand("trece");

ActionListener al = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand(); // simbolički naziv
    if (command.equals("prvo"))
      System.out.println("Pritisnuto dugme 1");
    else if (command.equals("drugo"))
      System.out.println("Pritisnuto dugme 2");
    else
      System.out.println("Pritisnuto dugme 3");
  }
};
jb1.addActionListener(al);
jb2.addActionListener(al);
jb3.addActionListener(al);
```

- Ako prethodno za dugme nismo pozvali metod **`setActionCommand`**, simbolični naziv dugmeta će inicijalno biti string koji je ispisan na njegovoj površini – trebalo bi izbegavati ovu situaciju

Dugmad - JCheckBox (1/3)

- **JCheckBox** je dugme koje može biti selektovano ili ne - obično je korisnik taj koji stanje dugmeta postavlja na željenu vrednost
- Inicijalne vrednosti se mogu postaviti pozivom metoda **setSelected(boolean b)**, a stanje dugmeta se uvek može proveriti pozivom metoda **isSelected**
- **JCheckBox** generiše događaje dva tipa: **ItemEvent** i **ActionEvent** (generišu se uvek u paru - relativno nebitno koji od dva tipa se osluškuju)
- Događaj tipa **ItemEvent** se generiše neznatno pre događaja tipa **ActionEvent**, pa je utoliko bolje implementirati osluškivač tipa **ItemListener**



Dugmad - JCheckBox (2/3)

Primer - Korišćenje osluškivača tipa `ActionListener`

```
JCheckBox jcb1 = new JCheckBox("Dugme 1");
jcb1.setSelected(true);
JCheckBox jcb2 = new JCheckBox("Dugme 2");
jcb2.setSelected(true);
JCheckBox jcb3 = new JCheckBox("Dugme 3");
jcb3.setSelected(true);

ActionListener al = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    Object source = e.getSource();
    if (((JCheckBox)source).isSelected())
      System.out.println("Selektovano ");
    else
      System.out.println("Deselektovano ");

    if (source == jcb1)
      System.out.println("dugme 1");
    else if (source == jcb2)
      System.out.println("dugme 2");
    else
      System.out.println("dugme 3");
  }
};
jcb1.addActionListener(al);
jcb2.addActionListener(al);
jcb3.addActionListener(al);
```

Dugmad - JCheckBox (3/3)

Primer - Verzija sa osluškivačem tipa `ItemListener`

```
JCheckBox jcb1 = new JCheckBox("Dugme 1");
jcb1.setSelected(true);
JCheckBox jcb2 = new JCheckBox("Dugme 2");
jcb2.setSelected(true);
JCheckBox jcb3 = new JCheckBox("Dugme 3");
jcb3.setSelected(true);

ItemListener il = new ItemListener() {
  public void itemStateChanged(ItemEvent e) {
    if (e.getStateChange() == ItemEvent.SELECTED)
      System.out.println("Selektovano ");
    else
      System.out.println("Deselektovano ");

    Object source = e.getItemSelectable();
    if (source == jcb1)
      System.out.println("dugme 1");
    else if (source == jcb2)
      System.out.println("dugme 2");
    else
      System.out.println("dugme 3");
  }
};
jcb1.addItemListener(il);
jcb2.addItemListener(il);
jcb3.addItemListener(il);
```

Dugmad - `JRadioButton` (1/2)

- Dugmad tipa **`JRadioButton`** se koriste u grupama - u svakom trenutku samo jedno dugme iz grupe može biti selektovano
- Grupa dugmadi tipa **`JRadioButton`** je predstavljena objektom klase **`ButtonGroup`**
- **`ButtonGroup`** objekat nije komponenta i nema grafičku reprezentaciju na ekranu - služi samo za interno grupisanje dugmadi
- Da bi neko dugme bilo inicijalno selektovano, kao i kod dugmadi tipa **`JCheckBox`**, koristi se metod **`setSelected`**

Dugmad - JRadioButton (2/2)

Primer

```
JRadioButton jrb1 = new JRadioButton("Dugme 1");
JRadioButton jrb2 = new JRadioButton("Dugme 2");
JRadioButton jrb3 = new JRadioButton("Dugme 3");

jrb1.setSelected(true);

ButtonGroup group = new ButtonGroup();
group.add(jrb1);
group.add(jrb2);
group.add(jrb3);

ActionListener al = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    Object source = e.getSource();
    if (source == jrb1)
      System.out.println("Selektovano prvo dugme");
    else if (source == jrb2)
      System.out.println("Selektovano drugo dugme");
    else
      System.out.println("Selektovano trece dugme");
  }
};
jrb1.addActionListener(al);
jrb2.addActionListener(al);
jrb3.addActionListener(al);
```

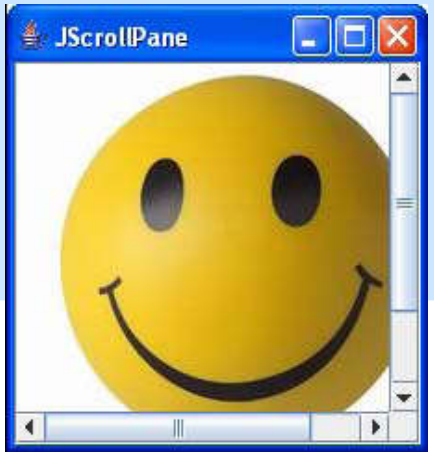
JScrollPane (1/2)

- **JScrollPane** je kontejner koji omogućava skrolovanje komponente koja je postavljena na njegovu površinu. Ako je dovoljno mala, komponenta se postavlja u centar kontejnera, inače se pojavljuju trake koje omogućavaju skrolovanje njegovog sadržaja
- Za kreiranje objekta tipa **JScrollPane** koriste se sledeći konstruktori:
  - **JScrollPane(Component view)**
  - **JScrollPane(Component view, int vsbPolicy, int hsbPolicy)** – za druga dva argumanta omogućavaju zadavanje pravila pojavljivanja traka za skroliranje i koriste se sledeće konstante:
    - **JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED**
    - **JScrollPane.HORIZONTAL_SCROLLBAR_NEVER**
    - **JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS**
    - **JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED**
    - **JScrollPane.VERTICAL_SCROLLBAR_NEVER**
    - **JScrollPane.VERTICAL_SCROLLBAR_ALWAYS**

JScrollPane (2/2)

Primer Kreiranje objekta tipa `JScrollPane` - omogućava skroliranje slike

```
//<applet code="ScrollDemo.class" width=320 height=320></applet>
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ScrollDemo extends JApplet {
  public void init() {
    ImageIcon smiley = new ImageIcon("smiley-big.jpg");
    JLabel picture = new JLabel(smiley);
    JScrollPane pictureScrollPane = new JScrollPane(picture);
    getContentPane().add(pictureScrollPane);
  }
  public static void main(String[] args) {
    JFrame frame = new JFrame("JScrollPane");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JApplet applet = new ScrollDemo();
    applet.init();
    applet.start();
    frame.getContentPane().add(applet);
    frame.setSize(320,320);
    frame.setVisible(true);
  }
}
```



Tekstualne komponente (1/4)

- **`JTextField`** i **`JTextArea`** su polja u koja korisnik može da upisuje tekst
- **`JTextField`** može sadržati samo jedan red teksta, dok **`JTextArea`** može sadržati i više redova, a obe klase podržavaju koncept selekcije
- Neki od korisnih metoda za rad sa tekstualnim komponentama:
  - `void setText(String newText)`
  - `String getText()`
  - `String getSelectedText()`
  - `void select(int start, int end)`
  - `void selectAll()`
  - `int getSelectionStart()`
  - `int getSelectionEnd()`
  - `void setEditable(boolean canBeEdited)`

Tekstualne komponente (2/4)

- Konstruktori klase **`JTextField`** su sledećeg oblika:
  - **`JTextField()`**
  - **`JTextField(int columns)`**
  - **`JTextField(String contents)`**
  - **`JTextField(String contents, int columns)`**
- Konstruktori klase **`JTextArea`** su sledećeg oblika:
  - **`JTextArea()`**
  - **`JTextArea(int lines, int columns)`**
  - **`JTextArea(String contents)`**
  - **`JTextArea(String contents, int lines, int columns)`**
- **`columns`** – zadaje se željena širina tekstualnog polja
- **`lines`** - zadaje se visina tekstualne površine

Tekstualne komponente (3/4)

- Klasa **`JTextArea`** poseduje još neke korisne metode:
  - **`void append(String text)`** - dodaje zadati tekst na kraj tekućeg sardžaja
  - **`void insert(String text, int pos)`** - zadati tekst umeće na zadatoj poziciji tekućeg sadržaja
  - **`void replaceRange(String text, int start, int end)`** - deo tekućeg sardžaja između zadatih pozicija zamenjuje zadatim tekstom

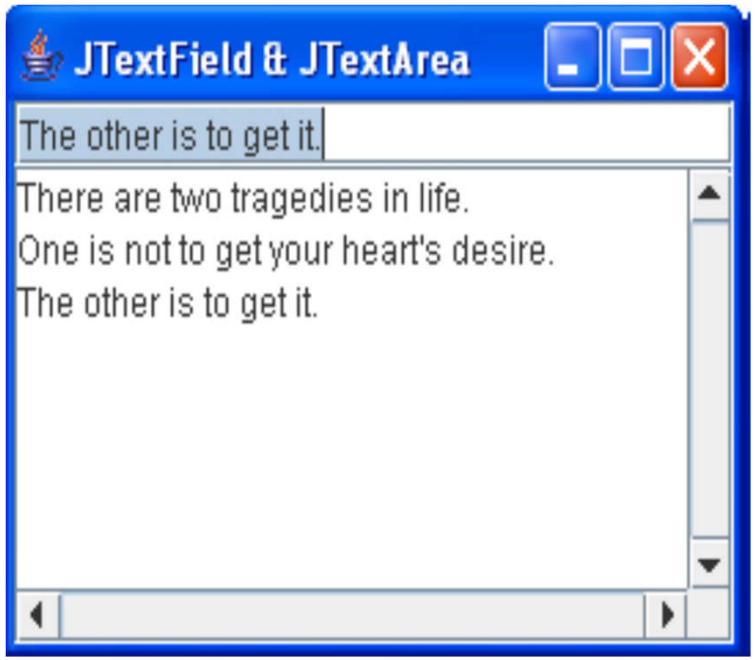
Tekstualne komponente (4/4)

- Objekti klase **`JTextField`** (za razliku od **`JTextArea`**) generišu događaje tipa **`ActionEvent`** kada korisnik tokom unosa teksta pritisne tipku 'Enter' na tastaturi

- Sve tekstualne komponente svoj sadržaj čuvaju u posebnom objektu klase **`Document`**

- Objekti ove klase generišu događaje tipa **`DocumentEvent`**, koji nastaju nakon svake izmene sadržaja tekstualne komponente. Odgovarajući osluškivač tipa **`DocumentListener`** se ne prijavljuje tekstualnoj komponenti direktno, već upravo odgovarajućem objektu klase **`Document`**

Tekstualne komponente - primer



Tekstualno polje dozvoljava unos teksta,
Tekstualna površina ne dozvoljava unos.

Kada se nakon unosa u tekstualno polje stisne ENTER:

- sadržaj tekstualnog polja se doda na kraj tekstualne površine.

- Nakon toga se selektuje sadržaj tekstualnog pollja.

```
//<applet code="TextDemo.class" width=270 height=200></applet>
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TextDemo extends JApplet {
  JTextField textField;
  JTextArea textArea;
  String newline = "\n";
```

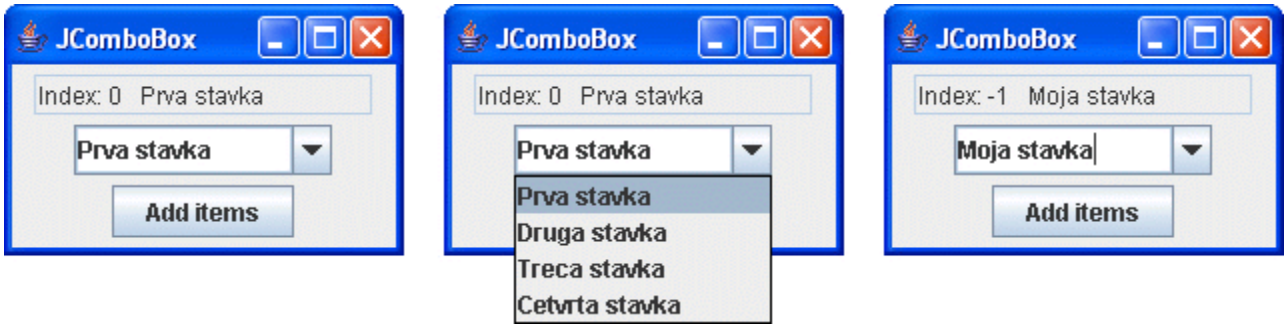
Tekstualne komponente - primer

```
  public void init() {
    textField = new JTextField(20);
    textField.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent evt) {
        String text = textField.getText();
        textArea.append(text + newline);
        textField.selectAll();
      }
    });
    textArea = new JTextArea(5, 20);
    textArea.setEditable(false);
    JScrollPane scrollPane = new JScrollPane(textArea,
      JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
      JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
    //dodavanje komponenti na povrsinu apleta
    Container cp = getContentPane();
    cp.add(textField, BorderLayout.NORTH);
    cp.add(scrollPane);
  }
  public static void main(String[] args) {
    JFrame frame = new JFrame("JTextField & JTextArea");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JApplet applet = new TextDemo();
    applet.init();
    applet.start();
    frame.getContentPane().add(applet);
    frame.setSize(270,200);
    frame.setVisible(true);
  }
}
```

Padajuće liste (`JComboBox`)

- Poput grupe dugmadi tipa **`JRadioButton`**, padajuća lista omogućava korisniku da odabere jednu od više ponuđenih mogućnosti, ali je izgled komponente puno kompaktniji
- Kada god korisnik odabere neku stavku iz menija, padajuća lista generiše događaj tipa **`ActionEvent`**
- Padajuća lista inicijalno ne dozvoljava unos teksta sa tastature – da bi se to postiglo potrebno je pozvati **`setEditable(true)`**
- Nove stavke se dodaju pozivom metoda **`addItem`** kojem se kao parametar prosleđuje string koji će biti dodat na kraj liste

Padajuće liste - primer (1/2)




```
//<applet code="ComboBoxDemo.class" width=200 height=125></applet>
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class ComboBoxDemo extends JApplet {
  private String[] description = {
    "Prva stavka", "Druga stavka", "Treca stavka", "Cetvrta stavka",
    "Peta stavka", "Sesta stavka", "Sedma stavka", "Osma stavka"
  };
  private JTextField jtf = new JTextField(15);
  private JComboBox jcb = new JComboBox();
  private JButton jb = new JButton("Add items");
  private int count = 0;
  public void init() {
    for(int i = 0; i < 4; i++)
      jcb.addItem(description[count++]);
    jtf.setEditable(false);
```

Padajuće liste - primer (2/2)

```
    jb.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        if(count < description.length)
          jcb.addItem(description[count++]);
      }
    });
    jcb.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jtf.setText("Index: "+ jcb.getSelectedIndex() + "    " +
          ((JComboBox)e.getSource()).getSelectedItem());
      }
    });
    jcb.setEditable(true);
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());
    cp.add(jtf);
    cp.add(jcb);
    cp.add(jb);
  }
  public static void main(String[] args) {
    JFrame frame = new JFrame("JComboBox");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JApplet applet = new ComboBoxDemo();
    applet.init();
    applet.start();
    frame.getContentPane().add(applet);
    frame.setSize(200,125);
    frame.setVisible(true);
  }
}
```

Liste (`JList`)

- Omogućava izbor prikazanih stavki, omogućavaju i višestruku selekciju (korišćenjem tipki 'Ctrl‘ ili 'Shift‘)
- Dozvoljeni načini selektovanja se mogu ograničiti pozivom metoda **`setSelectionMode`** kojem se kao parametar može proslediti jedna od sledećih konstanti:
  - **`ListSelectionModel.SINGLE_SELECTION`**
  - **`ListSelectionModel.SINGLE_INTERVAL_SELECTION`**
  - **`ListSelectionModel.MULTIPLE_INTERVAL_SELECTION`**
- Lista na svaku promenu selekcije generiše događaje tipa **`ListSelectionEvent`**

Meniji

- Svaki kontejner na najvišem nivou hijerarhije, kao što su **`JAplet`**, **`JFrame`**, **`JDialog`** i njihovi naslednici, može da sadrži najviše jednu traku sa menijima - metod **`setJMenuBar`** koji kao parametar prihvata objekat klase **`JMenuBar`**

- Svaki **`JMenuBar`** može da poseduje više menija predstavljenih klasom **`JMenu`**, a svaki meni može da sadrži više stavki koje su predstavljene klasama **`JMenuItem`**, **`JCheckBoxMenuItem`** i **`JRadioButtonMenuItem`**. Stavka u meniju može biti i drugi meni

- Sve klase kojima su predstavljene stavke u meniju nasleđuju klasu **`AbstractButton`**

- Bilo kojoj stavki može biti pridružen osluškivač tipa **`ActionListener`** (ili**`ItemListener`** za **`JCheckBoxMenuItem`**)

Meniji

- Mnemonici se koriste isto kao kod drugih vrsta dugmadi, dok su akceleratori specifični baš za stavke u meniju.
- **Mnemonik** može aktivirati stavku samo ako je ona trenutno prikazana na ekranu, odnosno ako je njen meni otvoren.
- Ako je nekoj stavci pridružen određeni karakter kao mnemonik, stavka se aktivira ako je vidljiva na ekranu i ako korisnik pritisne tipku odgovarajućeg karaktera na tastaturi (sa ili bez 'Alt').
- Stavka obično prikazuje mnemonik podvlačeći u svom tekstu prvo pojavljivanje pridruženog karaktera.
- **Akcelerator** može da aktivira stavku uvek, bez obzira na to da li su stavka i njen meni prikazani na ekranu.

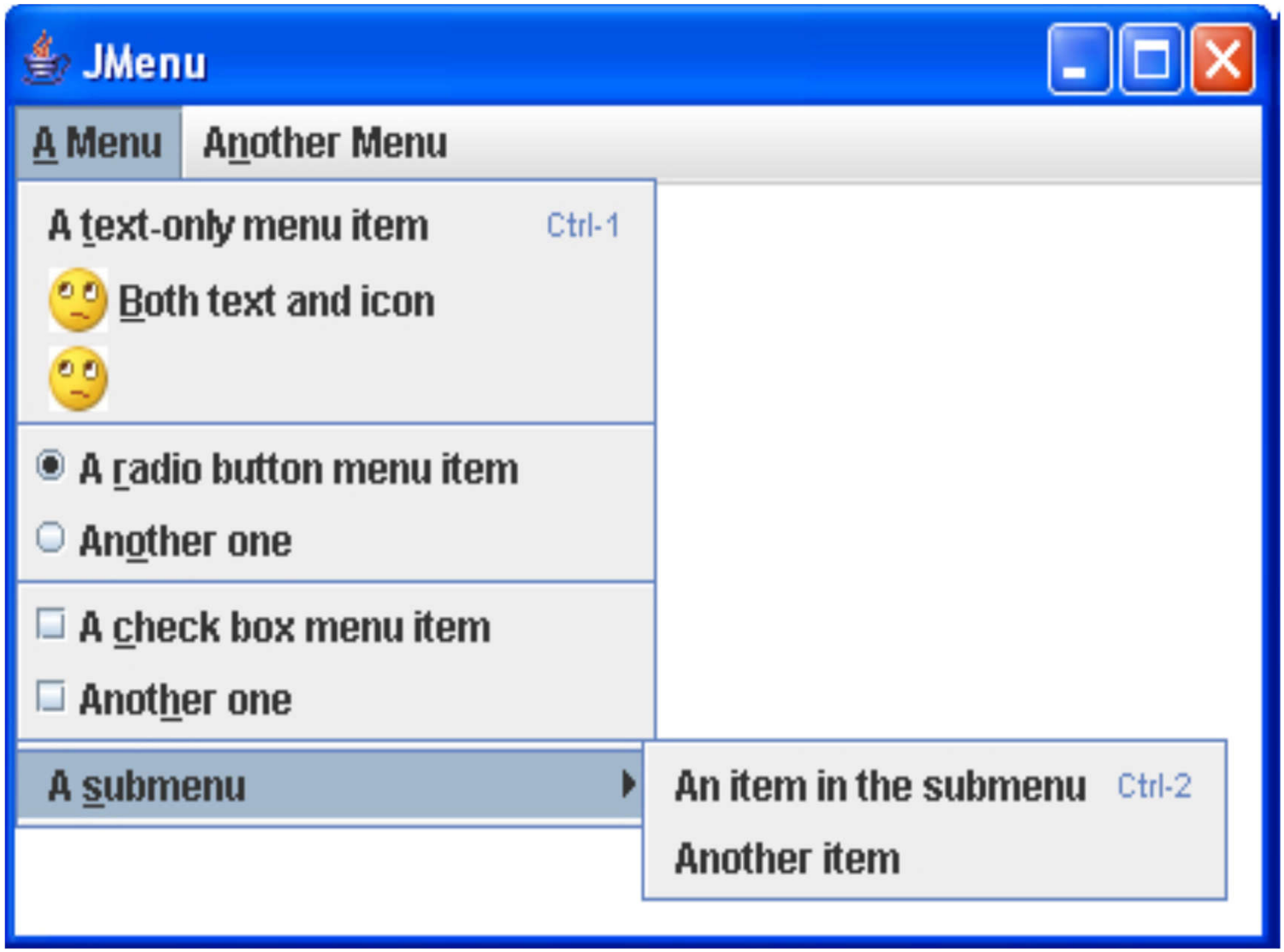
Meniji

- Mnemonik se zadaje pri kreiranju stavke, prosleđujući odgovarajući argument konstruktoru, ili pozivajući metod setMnemonic. Akcelerator se zadaje pozivajući metod setAccelerator. Na primer:

```
menuItem.setMnemonic(KeyEvent.VK_A);
menuItem.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_1,
      ActionEvent.CTRL_MASK));
```

- Pri zadavanju mnemonika koristi se konstanta klase **`KeyEvent`**, koja odgovara tipki koju korisnik treba da pritisne.
- Pri zadavanju akceleratora mora se koristiti objekat klase **`KeyStroke`** koji kombinuje tipku, zadatu preko konstante iz klase **`KeyEvent`** i modifikator 'Ctrl', 'Alt' ili 'Shift‘ koji se zadaje preko konstante iz klase **`ActionEvent`**.

Meniji - primer (1/5)



Meniji - primer (2/5)

```
//<applet code="MenuDemo.class" width=410 height=275></applet>
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class MenuDemo extends JApplet {

  public void init() {
    JMenuBar menuBar;
    JMenu menu, submenu;
    JMenuItem menuItem;
    JCheckBoxMenuItem cbMenuItem;
    JRadioButtonMenuItem rbMenuItem;


    //kreiranje trake sa menijima
    menuBar = new JMenuBar();
    setJMenuBar(menuBar);

     //kreiranje prvog menija
    menu = new JMenu("A Menu");
    menu.setMnemonic(KeyEvent.VK_A);
    menuBar.add(menu);
```

Meniji - primer (3/5)

```
//tri stavke tipa JMenuItem
menuItem = new JMenuItem("A text-only menu item");
setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_1,
  ActionEvent.CTRL_MASK));
menuItem.addActionListener(al);
menu.add(menuItem);

ImageIcon smiley = new ImageIcon("smiley.gif");
menuItem = new JMenuItem("Both text and icon", smiley);
menuItem.setMnemonic(KeyEvent.VK_B);
menuItem.addActionListener(al);
menu.add(menuItem);

menuItem = new JMenuItem(smiley);
menuItem.setMnemonic(KeyEvent.VK_D);
menuItem.addActionListener(al);
menuItem.setActionCommand("An image-only menu item");
menu.add(menuItem);

 //dve stavke tipa JRadioButtonMenuItem

menu.addSeparator();
ButtonGroup group = new ButtonGroup();
rbMenuItem = new JRadioButtonMenuItem("A radio button menu item");
rbMenuItem.setSelected(true);
rbMenuItem.setMnemonic(KeyEvent.VK_R);
rbMenuItem.addActionListener(al);
group.add(rbMenuItem);
menu.add(rbMenuItem);
```

Meniji - primer (4/5)

```
rbMenuItem = new JRadioButtonMenuItem("Another one");
rbMenuItem.setMnemonic(KeyEvent.VK_O);
rbMenuItem.addActionListener(al);
group.add(rbMenuItem);
menu.add(rbMenuItem);

//dve stavke tipa JCheckBoxMenuItem
menu.addSeparator();
cbMenuItem = new JCheckBoxMenuItem("A check box menu item");
cbMenuItem.setMnemonic(KeyEvent.VK_C);
cbMenuItem.addActionListener(al);
menu.add(cbMenuItem);
cbMenuItem = new JCheckBoxMenuItem("Another one");
cbMenuItem.setMnemonic(KeyEvent.VK_H);
cbMenuItem.addActionListener(al);
menu.add(cbMenuItem);

//podmeni
menu.addSeparator();
submenu = new JMenu("A submenu");
menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_2,
  ActionEvent.CTRL_MASK));
menuItem = new JMenuItem("An item in the submenu");
menuItem.addActionListener(al);
submenu.add(menuItem);
menuItem = new JMenuItem("Another item");
menuItem.addActionListener(al);
submenu.add(menuItem);
menu.add(submenu);
```

Meniji - primer (5/5)

```
    //drugi meni
    menu = new JMenu("Another Menu");
    menu.setMnemonic(KeyEvent.VK_N);
    menuBar.add(menu);

}
  public static void main(String[] args) {
    JFrame frame = new JFrame("JMenu");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JApplet applet = new MenuDemo();
    applet.init();
    applet.start();
    frame.getContentPane().add(applet);
    frame.setSize(410,275);
    frame.setVisible(true);
  }
}
```

JPopupMenu

- Objekat klase **`JPopupMenu`** može sadržavati stavke i podmenije na isti način kao objekat klase **`JMenu`**, ali se **`JPopupMenu`** ne dodaje traci sa menijima - on se prikazuje na ekranu kao rezultat pritiska odgovarajućeg tastera miša ili tastature.
- Da bi se **`JPopupMenu`** prikazao na ekranu, potrebno je prijaviti osluškivač za događaje generisane mišem svakoj komponenti koja bi trebalo da prikazuje ovu vrstu menija. Osluškivač treba da detektuje zahtev korisnika za prikazivanjem menija (što zavisi od operativnog sistema) i pozivom metoda **`show`** prikaže traženi meni.
- Da li je događaj generisan mišem za prikazivanje menija, možemo utvrditi pozivom metoda **`e.isPopupTrigger()`**, gde je **`e`** tipa **`MouseEvent.`**
- Metod menija **`show`** prihvata tri parametra - referencu na komponentu koja je izazvala prikazivanje menija i koordinate tačke koja će predstavljati gornji levi ugao menija.

JPopupMenu - primer (1/2)

```
//<applet code="PopupDemo.class" width=300 height=150></applet>
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class PopupDemo extends JApplet {
  JPopupMenu popup;
  JTextArea jta;
  public void init() {
    ActionListener al = new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jta.append("Action [" + e.getActionCommand() + "] performed!\n");
      }
    };
    //kreiranje menija
    popup = new JPopupMenu();
    JMenuItem menuItem = new JMenuItem("A popup menu item");
    menuItem.addActionListener(al);
    popup.add(menuItem);
    menuItem = new JMenuItem("Another popup menu item");
    menuItem.addActionListener(al);
    popup.add(menuItem);

    //kreiranje osluskivaca
    MouseListener popupListener = new PopupListener();
    jta = new JTextArea();
    jta.addMouseListener(popupListener);
    jta.setEditable(false);
    getContentPane().add(new JScrollPane(jta));
  }
```

JPopupMenu - primer (2/2)

```
class PopupListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
      maybeShowPopup(e);
    }
    public void mouseReleased(MouseEvent e) {
      maybeShowPopup(e);
    }
    private void maybeShowPopup(MouseEvent e) {
      if (e.isPopupTrigger()) {
        popup.show(e.getComponent(), e.getX(), e.getY());
      }
    }
  }
  public static void main(String[] args) {
    JFrame frame = new JFrame("JPopupMenu");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JApplet applet = new PopupDemo();
    applet.init();
    applet.start();
    frame.getContentPane().add(applet);
    frame.setSize(300,150);
    frame.setVisible(true);
  }
}
```

Podprozori za prikaz kratkih poruka (`JOptionPane`)

- U *Swing* biblioteci postoji skup standardnih podprozora koji omogućavaju jednostavno i brzo prikazivanje informacija ili unos podataka
- Statički metodi klase **`JOptionPane`**: **`showMessageDialog`**, **`showConfirmDialog`**, **`showOptionDialog`** i **`showInputDialog`**

JOptionPane - primer (1/2)

```
//<applet code="MessageBoxDemo.class" width=200 height=170></applet>
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class MessageBoxDemo extends JApplet {
  private JButton[] b = {
    new JButton("Greska"), new Jbutton("Da/Ne"), new Jbutton("Boja"),
    new JButton("Unos"), new JButton("3 vrednosti")
  };
  private JTextField txt = new JTextField(15);
  private ActionListener al = new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      String id = e.getActionCommand();
      if(id.equals("Greska"))
        JOptionPane.showMessageDialog(null, "Dogodila se greska...",
          "Greska", JOptionPane.ERROR_MESSAGE);
      else if(id.equals("Da/Ne")) {
        int val = JoptionPane.showConfirmDialog(null,
          "Odaberi da ili ne!", "Da/Ne", JOptionPane.YES_NO_OPTION);
        txt.setText("Izabrana opcija: " +
          (val == JOptionPane.YES_OPTION ? "Da" : "Ne"));
      } else if(id.equals("Boja")) {
        Object[] options = {"Crvena", "Zelena"};
        int sel = JoptionPane.showOptionDialog(null, "Odaberi boju!",
          "Upozorenje", JOptionPane.DEFAULT_OPTION,
          JOptionPane.WARNING_MESSAGE, null,
          options, options[0]);
        if(sel != JOptionPane.CLOSED_OPTION)
          txt.setText("Izabrana boja: " + options[sel]);
```

JOptionPane - primer (2/2)

```
} else if(id.equals("Unos")) {
        String val = JOptionPane.showInputDialog("Napisi nesto...");
        txt.setText(val);
      } else if(id.equals("3 vrednosti")) {
        Object[] selections = {"Prva", "Druga", "Treca"};
        Object val = JoptionPane.showInputDialog(null,
          "Odaberi vrednost!", "Unos",
          JOptionPane.INFORMATION_MESSAGE,
          null, selections, selections[0]);
        if(val != null)
          txt.setText(val.toString());
      }
    }
  };
  public void init() {
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());
    for(int i = 0; i < b.length; i++) {
      b[i].addActionListener(al);
      cp.add(b[i]);
    }
    cp.add(txt);
  }
  public static void main(String[] args) {
    JFrame frame = new JFrame("JMessageBox");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JApplet applet = new MessageBoxDemo();
    applet.init();
    applet.start();
    frame.getContentPane().add(applet);
    frame.setSize(200,170);
    frame.setVisible(true);
  }
}
```

Podprozori (`JDialog`)

- Podprozor je prozor koji se pojavljuje iz drugog prozora
- Često se koriste u prozorskim aplikacijama, a puno ređe u apletima
- Da bi se implementirao podprozor, potrebno je naslediti klasu **`JDialog`**
- Podprozor ima sve karakteristike prozora po pitanju smeštanja i rasporedjivanja komponenti, a inicijalni rasporedjivač je tipa **`BorderLayout`**

JDialog - primer (1/2)

```
//<applet code="DialogDemo.class" width=125 height=75></applet>
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
class MyDialog extends JDialog {
  public MyDialog(JFrame parent) {
    super(parent, "My dialog", true);
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout(FlowLayout.CENTER,10,15));
    cp.add(new JLabel("Here is my dialog"));
    JButton ok = new JButton("OK");
    ok.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        dispose(); // zatvara podprozor
      }
    });
    cp.add(ok);
    setSize(150,125);
  }
}
public class DialogDemo extends JApplet {
  private JButton b1 = new JButton("Dialog Box");
  private MyDialog dlg = new MyDialog(null);
  public void init() {
    b1.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        dlg.setVisible(true);
      }
    });
```

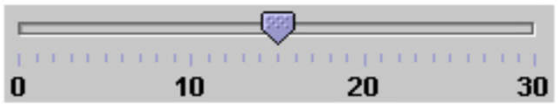
JDialog - primer (2/2)

```
    getContentPane().add(b1);
  }
  public static void main(String[] args) {
    JFrame frame = new JFrame("JDialog");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JApplet applet = new DialogDemo();
    applet.init();
    applet.start();
    frame.getContentPane().add(applet);
    frame.setSize(200,170);
    frame.setVisible(true);
  }
}
```

Ostale komponente (1/2)

- **`JSlider`** – Omogućava korisniku da zadaje numeričke vrednosti iz nekog opsega prevlačenjem markera
- **`JProgressBar`** – Obično se koristi da bi se korisniku prikazao napredak u obavljanju neke dugotrajne aktivnosti kao što su instalacija softvera ili kopiranje velike količine podataka
- **`JToolBar`** – Traka koja sadrži 'alate', poput ikona i dugmadi. Korisnik može da prevuče traku uz bilo koju ivicu prozora ili čak može da je odvoji od prozora pa da traka tako postane poseban prozor
- **`JTable`** – Prikazuje dvodimenzionalnu tabelu elemenata i eventualno omogućava korisniku editovanje
- **`JTree`** – Prikazuje hijerarhijski organizovane podatke u strukturi stabla
- **`JSplitPane`** – Kontejner koji prikazuje dve komponente. Korisnik može da pomera liniju koja razdvaja komponente da bi podesio njihove relativne veličine

Ostale komponente (2/2)



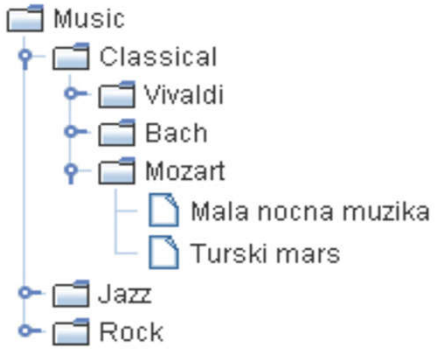
JSlider

JProgressBar

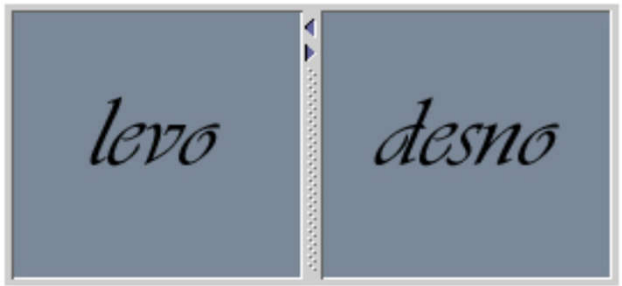
JToolBar

Ime	Prezime	Godine	Vegetarijanac
Petar	Petrovic	25	jeste
Marko	Markovic	38	nije
Milos	Milosevic	31	jeste
Zarko	Zarkovic	55	nije

JTable



JTree



JSplitPane

Animacije

- Animacije predstavljaju nizove slika koje se na ekranu prikazuju jedna nakon druge

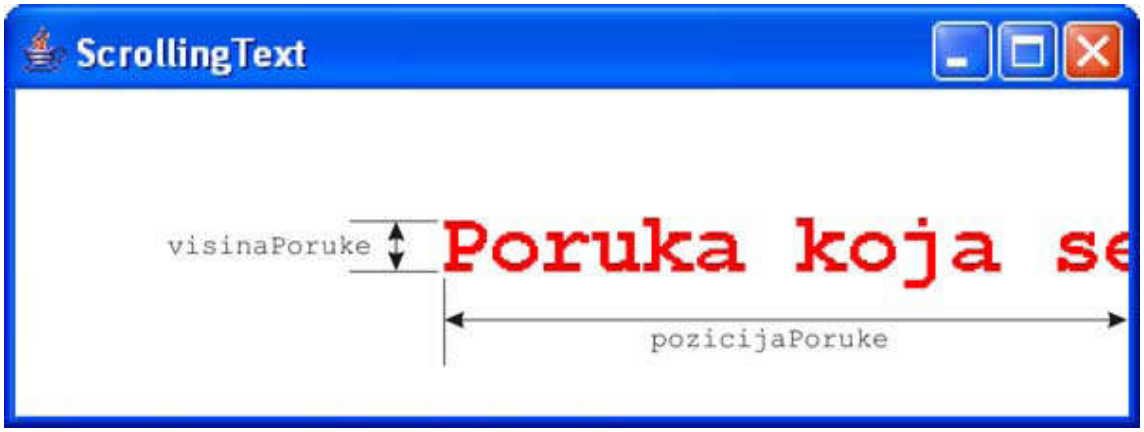
- Najprirodniji način za programiranje animacije da se kreira zasebna nit koja će animaciju izvršavati - nit možemo kreirati eksplicitno, nasleđivanjem klase **`Thread`** ili implementiranjem interfejsa **`Runnable`** i redefinisanjem odgovarajućeg metoda **`run`**

- Jednostavan mehanizam programiranja aplikacija – koristeći objekat klase **`javax.swing.Timer`**, koji nezavisno, bez uticaja korisnika, generiše dogadjaje u pravilnim vremenskim intervalima

Animacije

- Programiranje animacije - kreiramo brojač i reagujemo na svaki događaj koji on generiše tako što ćemo prikazati narednu sliku animacije.
- Događaji generisani brojačem su tipa **`ActionEvent.`**
- Brojač startujemo pozivanjem metoda **`start`** (po pravilu bi ga trebalo pozvati samo jednom za svo vreme postojanja brojača).
- Brojač takođe ima metod **`stop`** koji se poziva da bi se zaustavilo generisanje događaja. Ako smo zaustavili brojač i želimo ponovo da ga pokrenemo, moramo pozvati njegov metod **`restart.`**

Animacije - primer (1/3)




```
//<applet code="ScrollingText.class" width=400 height=150></applet>
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ScrollingText extends JApplet {
  Timer brojac;
  String poruka = "Poruka koja se skroluje...";
  int pozicijaPoruke = -1;
  int sirinaPoruke, visinaPoruke;
  int sirinaKaraktera;
```

Animacije - primer (2/3)

```
public void init() {
  JPanel display = new JPanel() {
    public void paintComponent(Graphics g) {
      super.paintComponent(g);
      g.drawString(poruka, getSize().width - pozicijaPoruke,
                           getSize().height/2 + visinaPoruke/2);
    }
  };
  getContentPane().add(display);
  display.setBackground(Color.white);
  display.setForeground(Color.red);
  Font fontPoruke = new Font("Monospaced", Font.BOLD, 30);
  display.setFont(fontPoruke);
  FontMetrics fm = getFontMetrics(fontPoruke);
  sirinaPoruke = fm.stringWidth(poruka);
  visinaPoruke = fm.getAscent();
  sirinaKaraktera = fm.charWidth('P');
}

public void start() {
  if (brojac == null) {
    ActionListener al = new ActionListener() {
      public void actionPerformed(ActionEvent evt) {
        pozicijaPoruke += sirinaKaraktera/2;
        if (getSize().width - pozicijaPoruke + sirinaPoruke < 0)
          pozicijaPoruke = -1;
        repaint();
      }
    };
```

Animacije - primer (3/3)

```
      brojac = new Timer(50, al);
      brojac.start();
    } else {
      brojac.restart();
    }
  }
  public void stop() {
    brojac.stop();
  }

  private static JApplet applet;
  public static void main(String[] args) {
    JFrame frame = new JFrame("ScrollingText");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    applet = new ScrollingText();
    applet.init();
    applet.start();

frame.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        applet.stop();
        applet.destroy();
      }
    });
    frame.getContentPane().add(applet);
    frame.setSize(400,150);
    frame.setVisible(true);
  }
}
```

Teorijske vežbe

Objektno-orjentisano programiranje

Zadatak 7

- Intefejs **URLAdresirano** predstavlja resurs na internetu koji ima jedinstvenu URL adresu i sadrzi metod String adresa().

```
public interface URLAdresirano {
    String adresa();
}
```

Zadatak 7

- Apstraktna klasa **WebSajt** implementira interfejs URLAdresirano i implementira odgovarajući metod tako što vraća vrednost svog polja u kom čuva adresu.
- Klasa osim toga definiše dva apstraktna metoda:
  - double **cenaReklameNaSajtu()** - vraća cenu koju vlasnici sajta mogu dobiti ukoliko neko želi da postavi reklamu na njihov sajt na period od godinu dana.
  - boolean **odobrenoKomentarisanje()** - vraća true ukoliko unutar sajta postoji opcija komentarisanja; u suprotnom vraća false.
- Pored navedenih metoda klasa treba da sadrži i konstruktor koji inicijalizuje adresu, kao i toString metod koji pored internet adrese sajta treba da iskoristi i cenu reklame na sajtu, kao i informaciju o tome da li je komentarisanje odobreno ili nije.

```java
public abstract class WebSajt implements URLAdresirano {

    private String adresa;

    public WebSajt(String adresa) {
        this.adresa = adresa;
    }

    @Override
    public String adresa() {
        return adresa;
    }

    @Override
    public String toString() {
        return "Web sajt sa adresom " + adresa() + ", "
            + "cenom reklame " + cenaReklameNaSajtu()
            + ", na kom " + (odobrenoKomentarisanje()
            ? "je" : "nije") + " odobreno komentarisanje";
    }

    public abstract double cenaReklameNaSajtu();

    public abstract boolean odobrenoKomentarisanje();
}
```

Zadatak 7

- Neapstraktna klasa **Blog** nasleđuje klasu WebSajt. Svaki blog sadrži niz postova, a postovi su opisani statičkom ugnježdenom klasom **Post**.
- Statička ugnježdena klasa **Post** sadrži sadržaj posta (String), broj lajkova (int) i broj komentara (int) na postu.
- Klasa sadrži konstruktor koji inicijalizuje sva polja klase, kao i sledeći metod:
  - **int popularnostPosta()** - Vraća broj koji govori o popularnosti posta. Svaki lajk vredi 5, dok svaki komentar vredi 12.
- Zbir vrednosti svih lajkova i komentara predstavlja popularnost posta i ujedno je i rezultat izvršavanja metoda.
- Konstruktor klase Blog prima internet adresu bloga, kao i niz postova od kojih se blog sadrži. Na blogu je komentarisanje odobreno ukoliko bar jedan post ima bar jedan komentar, a godišnja cena reklame se dobija tako što se zbir popularnosti svih postova pomnoži brojem 1,5.

```java
public class Blog extends WebSajt {

    private Post[] postovi;

    public Blog(String adresa, Post[] postovi) {
        super(adresa);
        this.postovi = postovi;
    }

    @Override
    public boolean odobrenoKomentarisanje() {
        for (Post post : postovi) {
            if (post.brojKomentara > 0) return true;
        }
        return false;
    }

    @Override
    public double cenaReklameNaSajtu() {
        int popularnost = 0;
        for (Post post : postovi) {
            popularnost += post.popularnostPosta();
        }
        return popularnost * 1.5;
    }
```

```
    public static class Post {
        private String sadrzaj;
        private int brojLajkova;
        private int brojKomentara;

        public Post(String sadrzaj, int brojLajkova, int brojKomentara) {
            this.sadrzaj = sadrzaj;
            this.brojLajkova = brojLajkova;
            this.brojKomentara = brojKomentara;
        }

        int popularnostPosta() {
            return brojLajkova * 5 + brojKomentara * 12;
        }
    }
}
```

Zadatak 7

- Klasa **OmiljeniSajtovi** čuva niz omiljenih sajtova (favorites). Niz sajtova treba da se učita u konstruktoru klase, a podaci o sajtovima se čuvaju u fajlu, i organizovani su na sledeći način:
  - U prvoj liniji fajla je broj sajtova.
  - Svaki blog je opisan sa k + 1 linijom, pri čemu je:
    - prva linija formata: adresa sajta?, broj postova na blogu (k),
    - svaka sledeća linija opisuje po jedan post na sledeći način: sadržaj posta, broj lajkova, broj komentara?.
- Klasa OmiljeniSajtovi treba da sadrzi i metod:
- **WebSajt gdeDaPostavimReklamu(double raspoloziviNovac) throws NemaDovoljnoNovca** - Vraća sajt čija je cena postavljanja reklame najveća moguća koja se uklapa u raspoloživi fond. Metod baca izuzetak NemaDovoljnoNovca ukoliko ne postoji sajt za koji se ima dovoljno sredstava, a izuzetak treba da sadrži odgovarajuću poruku.

```
public class OmiljeniSajtovi {

    private WebSajt[] sajtovi;
    private final String fajlSaPodacima = "sajtovi.txt";

    public OmiljeniSajtovi() throws IOException {
        BufferedReader in = new BufferedReader(new FileReader(fajlSaPodacima));
        sajtovi = new WebSajt[Integer.parseInt(in.readLine().trim())];
        for (int i = 0; i < sajtovi.length; i++) {
            String[] prvaLinija = in.readLine().split(",");
            String adresa = prvaLinija[0].trim();
            Blog.Post[] postovi = new Blog.Post
                    [Integer.parseInt(prvaLinija[1].trim())];
            for (int k = 0; k < postovi.length; k++) {
                String[] linija = in.readLine().split(",");
                postovi[k] = new Blog.Post(linija[0].trim(),
                    Integer.parseInt(linija[1].trim()),
                    Integer.parseInt(linija[2].trim()));
            }
            sajtovi[i] = new Blog(adresa, postovi);
        }
        in.close();
    }
```

```
    public WebSajt gdeDaPostavimReklamu(double raspoloziviNovac)
            throws NemaDovoljnoNovca {
        WebSajt rezultat = null;
        for (WebSajt sajt : sajtovi) {
            if (sajt.cenaReklameNaSajtu() <= raspoloziviNovac) {
                if (rezultat == null || raspoloziviNovac
                        - sajt.cenaReklameNaSajtu()
                        < raspoloziviNovac
                        - rezultat.cenaReklameNaSajtu()) {
                    rezultat = sajt;
                }
            }
        }
        if (rezultat == null) throw new NemaDovoljnoNovca(raspoloziviNovac);
        return rezultat;
    }
}
```

```java
import java.math.*;

public class NemaDovoljnoNovca extends Exception {

    /*
        ovaj deo:
            new BigDecimal(iznos).setScale(2, RoundingMode.HALF_UP).doubleValue()
        sluzi samo da zaokruzimo double na dve decimale
    */
    public NemaDovoljnoNovca(double iznos) {
        super("Ne postoji sajt za koji je dovoljno " +
            new BigDecimal(iznos).setScale(2, RoundingMode.HALF_UP).doubleValue()
                + " za reklamu");
    }
}
```

Zadatak 7

- U **glavnom programu** je potrebno napraviti instancu klase OmiljeniSajtovi.
- U while petlji je potrebno pozivati i ispisivati rezultat izvršavanja metoda gdeDaPostavimReklamu počevši od neke proizvoljne sume, pa sve dok ima novca.
- Ne pretpostavljamo da je fajl ispravno formatiran i potrebno je obraditi sve izuzetke do kojih može doći.

```java
import java.io.IOException;

public class Glavna {
    public static void main(String[] args) {
        try {
            OmiljeniSajtovi omiljeniSajtovi = new OmiljeniSajtovi();
            double iznos = 10000;
            while(true) {
                WebSajt sajtZaReklamu =
                omiljeniSajtovi.gdeDaPostavimReklamu(iznos);
                iznos -= sajtZaReklamu.cenaReklameNaSajtu();
                System.out.println(sajtZaReklamu);
            }
        } catch (IOException | NemaDovoljnoNovca e) {
            System.out.println(e.getMessage());
        }
    }
}
```