Objektno orijentirano programiranje

15.1: Grafičko sučelje – Swing

Razmještaj kontrola i obrada događaja

Creative Commons

You are free to

- **Share** — copy and redistribute the material in any medium or format
- **Adapt** — remix, transform, and build upon the material

under the following terms

- **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **NonCommercial** — You may not use the material for commercial purposes.
- **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

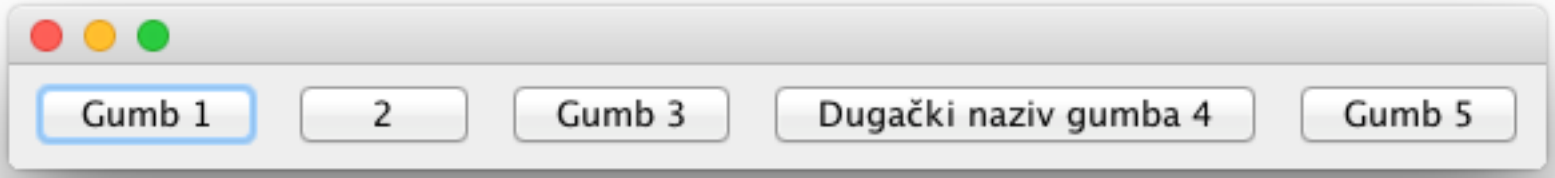
- https://creativecommons.org/licenses/by-nc-sa/4.0/

Upravljači razmještajem (*layout managers*)

- Objekti koji su zaduženi za proračun koordinata i veličina komponenti u nekoj sadržavajućoj komponenti (kontejneru)
- osnovni upravljači :
  - **FlowLayout** – postavlja komponente od lijeve strane na desno i kada se popuni jedan red onda se prelazi u drugi red
  - **BorderLayout** – postavlja komponente oko centralnog dijela *Containera*
  - **CardLayout** – postavlja komponente jednu iza druge, kao karte
  - **GridLayout** – postavlja komponente u tablicu čije su ćelije iste veličine
  - **GridBagLayout** – postavlja komponente u tablicu, a za ćelije se definiraju ograničenja (npr. zauzima dvije ćelije vodoravno ili okomito). Kod promjene veličine se neki stupac ili redak povećava u određenim proporcijama)
  - **BoxLayout** – stavlja komponente u jedan redak ili kolonu
  - **SpringLayout** – omogućuje definiranje relacija između komponenti (koristi se kod unosa podataka)
  - **GroupLayout** – omogućuje posebno definiranje vertikalnih i horizontalnih grupa (koristi se kod alata za izradu GUI-ja)

FlowLayout

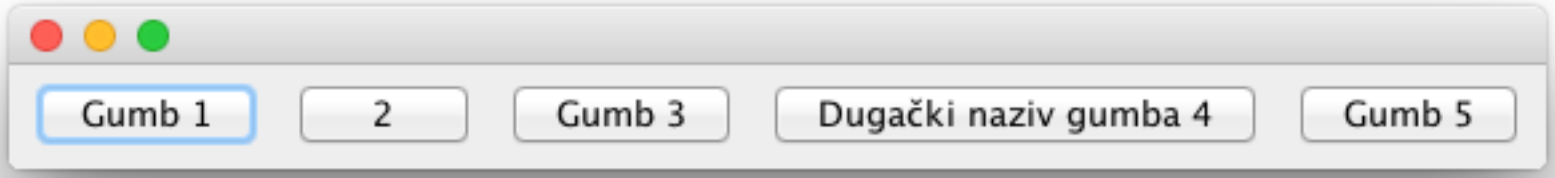
- bitan je redoslijed dodavanja komponenti u container
- koristi se kada je potrebno minimizirati prostor na ekranu kojeg zauzimaju komponente
- koristi se za centriranje komponenti
- ako komponente ne stanu u jedan red stavlja ih u novi
- **podrazumijevani razmještaj (eng. layout) za JPanel**
- neke metode:
  - `setAlignment(int align)`
  - `FlowLayout.LEFT, RIGHT i CENTER`
  - `setHgap(int hgap)`
  - `setVgap(int vgap)`



FlowLayout - primjer

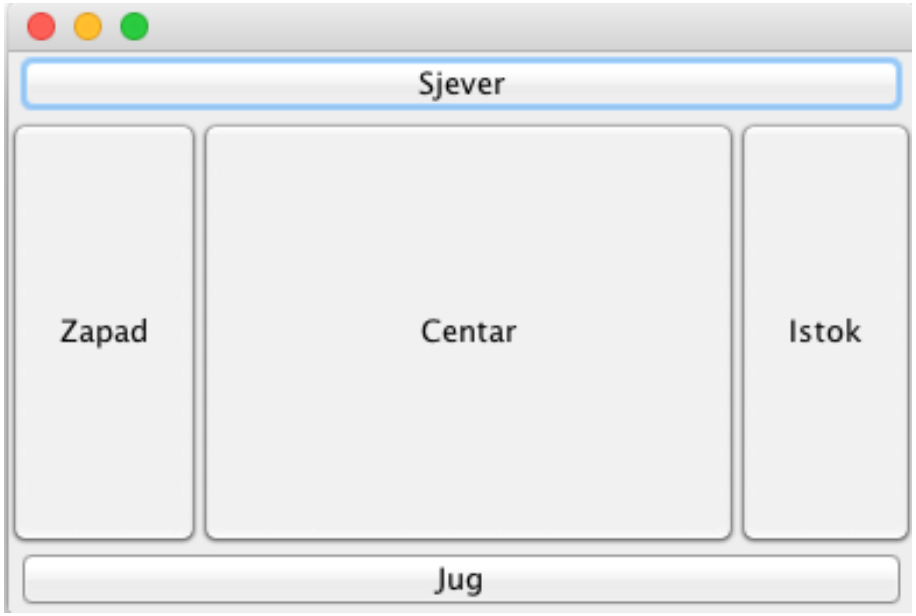
```
public class FlowLayoutExample extends JFrame {
  public FlowLayoutExample() {
    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    setLayout(new FlowLayout());

    add(new JButton("Gumb 1"));
    add(new JButton("2"));
    add(new JButton("Gumb 3"));
    add(new JButton("Dugački naziv gumba 4"));
    add(new JButton("Gumb 5"));
  }
...
```

15_Swing/hr.fer.oop.swing1.layouts.FlowLayoutExample



BorderLayout

- Jedan od najkorisnijih upravljača i **podrazumijevani za JFrame**
- Dijeli područje na 5 cjelina
- Centralni dio se širi najviše moguće u svim smjerovima
  - ostali dijelovi zauzimaju onoliko mjesta koliko im je najmanje potrebno



- **Na svako mjesto se može staviti samo jedna komponenta**.
  - Za više njih na jednom dijelu koristimo kontejner s više komponenti
- za postavljanje komponenti se koristi metoda:

```
add(Component comp, Object constraints)
constraints: BorderLayout.NORTH, SOUTH, EAST, WEST, CENTER
ili:         BorderLayout.PAGE_START, PAGE_END, LINE_END,
                        LINE_START, CENTER
```

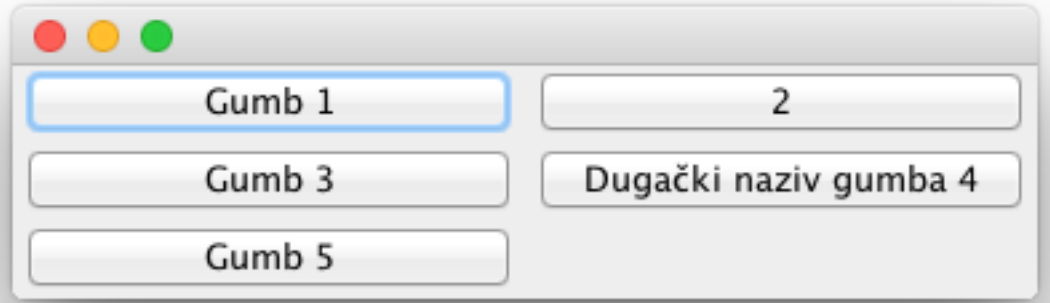
BorderLayout – jednostavni primjer

```
public class BorderLayoutExample extends JFrame {
 public BorderLayoutExample() {
    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    setLayout(new BorderLayout());

    add(new JButton("Sjever"), BorderLayout.NORTH);
    add(new JButton("Zapad"), BorderLayout.WEST);
    add(new JButton("Centar"), BorderLayout.CENTER);
    add(new JButton("Istok"), BorderLayout.EAST);
    add(new JButton("Jug"), BorderLayout.SOUTH);
  }
...
}
```

15_Swing/hr.fer.oop.swing1.layouts.BorderLayoutExample

GridLayout

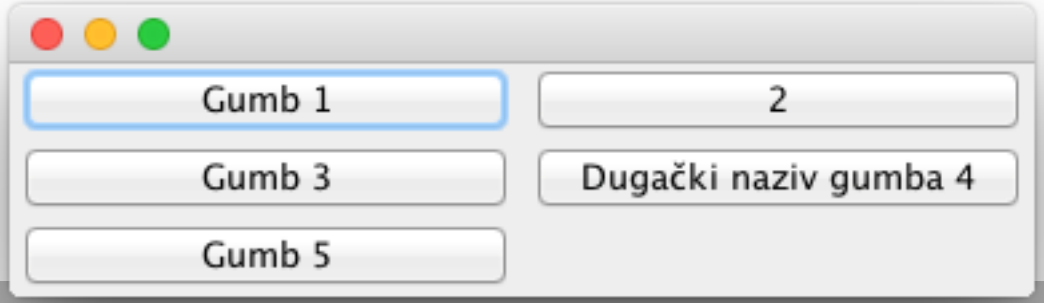
- služi za stavljanje komponenti u tablicu čije su ćelije iste veličine
- konstruktor:
  - `GridLayout(int rows, int cols)`
    - barem jedan od parametara mora biti različit od 0
    - ako su obje vrijednosti različite od nule, broj stupaca se ignorira te se automatski izračunava na osnovu broja redaka i broja elemenata
- važnije metode
  - `setColumns(int cols)`
  - `setRows(int rows)`



GridLayout – jednostavni primjer

```
public class GridLayoutExample extends JFrame {

  public GridLayoutExample() {
    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    setLayout(new GridLayout(3,0));

    add(new JButton("Gumb 1"));
    add(new JButton("2"));
    add(new JButton("Gumb 3"));
    add(new Jbutton("Dugački naziv gumba 4"));
    add(new JButton("Gumb 5"));
  }
...
```

15_Swing/hr.fer.oop.swing1.layouts.GridLayoutExample



SpringLayout

- definira odnose između rubova komponenti
  - konstruktor: `SpringLayout()`
  - dodavanje odnosa između komponenti:
    - `layout.putConstraints()`
    - konstante ruba: `SpringLayout.EAST, NORTH, SOUTH, WEST,VERTICAL_CENTER, HORIZONTAL_CENTER,BASELINE`

```
SpringLayout layout = new SpringLayout();
setLayout(layout);
JButton btn1 = new JButton("Gumb 1");
add(btn1);
JButton btn2 = new JButton("Gumb 2");
add(btn2);
JButton btn3 = new JButton("Gumb 3");
add(btn3); //vidi sljedeći slajd
```

SpringLayout – jednostavni primjer

```
layout.putConstraint(SpringLayout.WEST, btn1, 5,
  SpringLayout.WEST, getContentPane());
layout.putConstraint(SpringLayout.NORTH, btn1, 5,
  SpringLayout.NORTH, getContentPane());
layout.putConstraint(SpringLayout.WEST, btn2, 10,
  SpringLayout.EAST, btn1);
```

A: udaljenost lijevog ruba Gumba 1 od lijevog ruba kontejnera

B: udaljenost gornjeg ruba Gumba 1 od gornjeg ruba kontejnera

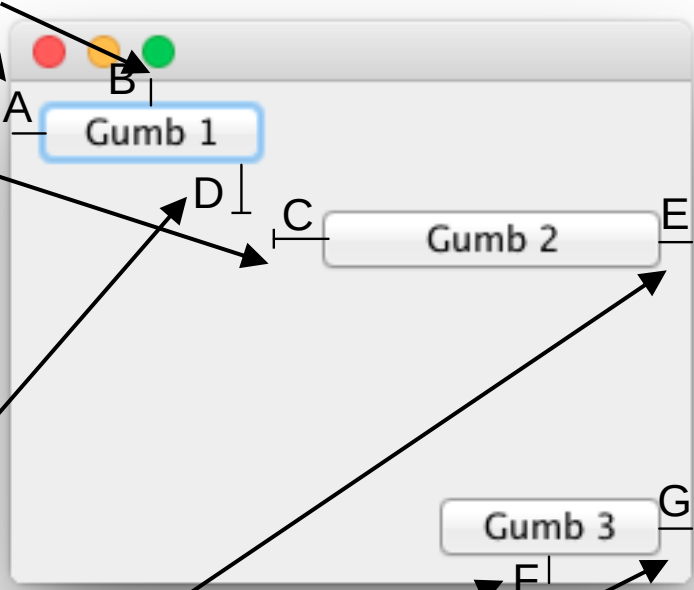
C: udaljenost lijevog rub Gumba 2 od desnog ruba Gumba 1

D: udaljenost gornjeg ruba Gumba 2 od donjeg ruba Gumba 1

E: udaljenost desnog ruba Gumba 2 od desnog ruba kontejnera

F: udaljenost donjeg ruba Gumba 3 od donjeg ruba kontejnera

G: udaljenost desnog ruba Gumba 3 od desnog ruba kontejnera




```
layout.putConstraint(SpringLayout.NORTH, btn2, 10,
  SpringLayout.SOUTH, btn1);
layout.putConstraint(SpringLayout.EAST, btn2, -5,
  SpringLayout.EAST, getContentPane());
layout.putConstraint(SpringLayout.SOUTH, btn3, -5,
  SpringLayout.SOUTH, getContentPane());
layout.putConstraint(SpringLayout.EAST, btn3, -5,
  SpringLayout.EAST, getContentPane());
```

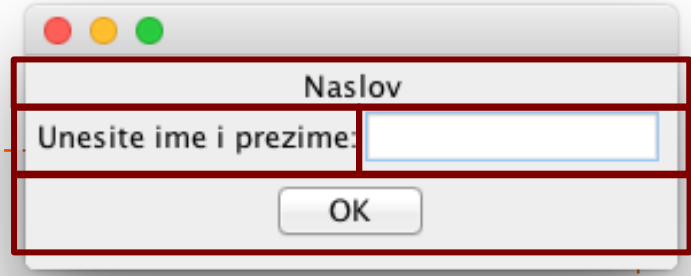
SpringLayout - napomena

- *SpringLayout* je uobičajeno namijenjen za korištenje pomoću nekog od alata za izradu grafičkog sučelja
  - kao alternative mogu se koristiti *GroupLayout* i *GridBagLayout*

- Ako je ipak potrebno ručno stvoriti formu za unos pomoću *SpringLayouta* preporuča se koristiti neku od metoda *makeGrid* and *makeCompactGrid* u klasi *SpringUtilities*
  - više na https://docs.oracle.com/javase/tutorial/uiswing/layout/spring.html

Primjer obrade klika na gumb

- Unos podataka o osobi – ime i prezime, te potvrda gumbom OK
  - klikom na gumb pojavljuje se novi ekran s porukom Hello uneseno ime i prezime
- Kôd je organiziran na način da je dodavanje mnoštva komponenti izveden konstruktorom klase *EnterNameFrame*.
  - korišten je *BorderLayout*



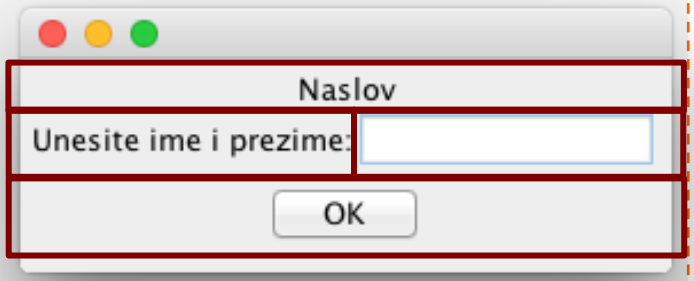

```
public class EnterNameFrame extends JFrame {
  public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        EnterNameFrame frame = new EnterNameFrame();
        frame.pack(); //slaganje komponenti na najmanje prostora
        frame.setVisible(true);
    });
  }
…
```

15_Swing/hr.fer.oop.swing1.example1.EnterNameFrame

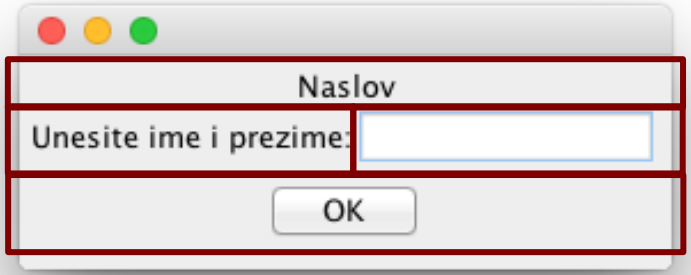
Kôd konstruktura *EnterNameFrame* – dodavanje komponenti

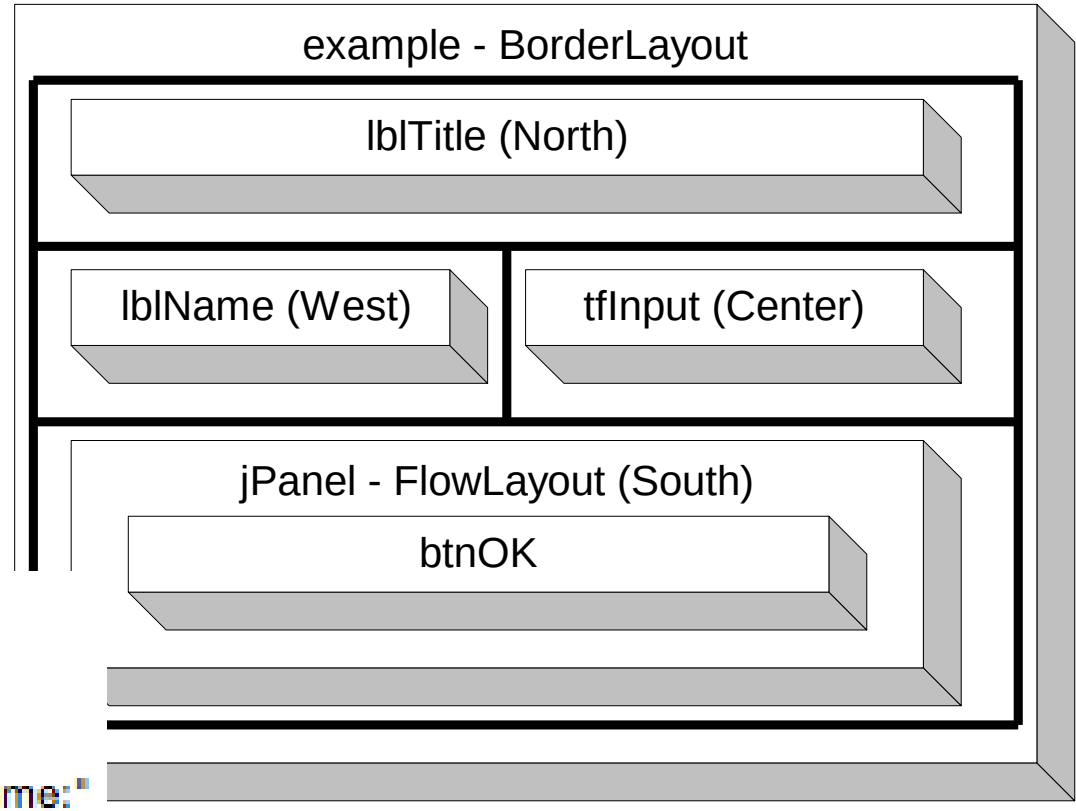
15_Swing/hr.fer.oop.swing1.example1.EnterNameFrame

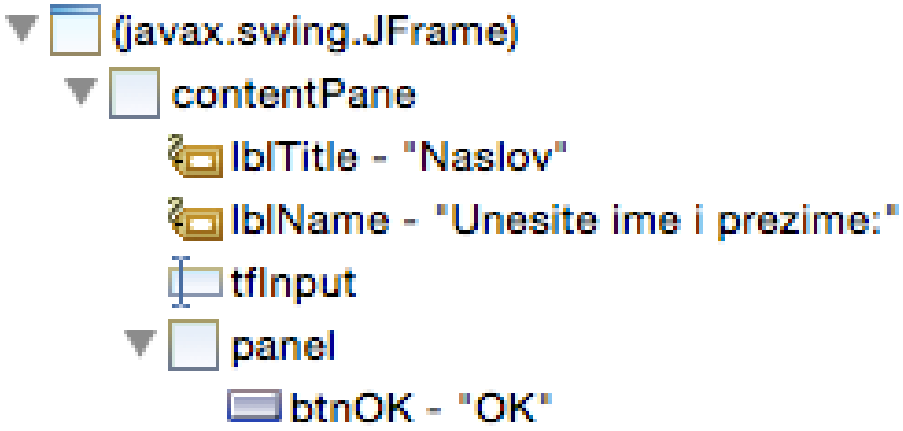
```
public EnterNameFrame() {
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setBounds(10, 10, 500, 200); // postavljanje lokacije i veličine

  JPanel panel = (JPanel) getContentPane(); // ovo je potrebno zbog okvira
  panel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));

  JLabel lblTitle = new JLabel("Naslov");
  lblTitle.setHorizontalAlignment(SwingConstants.CENTER);
  panel.add(lblTitle, BorderLayout.NORTH);

  JLabel lblName = new JLabel("Unesite ime i prezime:");
  panel.add(lblName, BorderLayout.WEST);

  JTextField tfInput = new JTextField();
  panel.add(tfInput, BorderLayout.CENTER);
  tfInput.setColumns(10);

  JPanel southPanel = new JPanel();
  panel.add(southPanel, BorderLayout.SOUTH);
  JButton btnOK = new JButton("OK");
  southPanel.add(btnOK);
...
```



Hijerahija u *BorderLayoutu* na primjeru *EnterNameFrame*







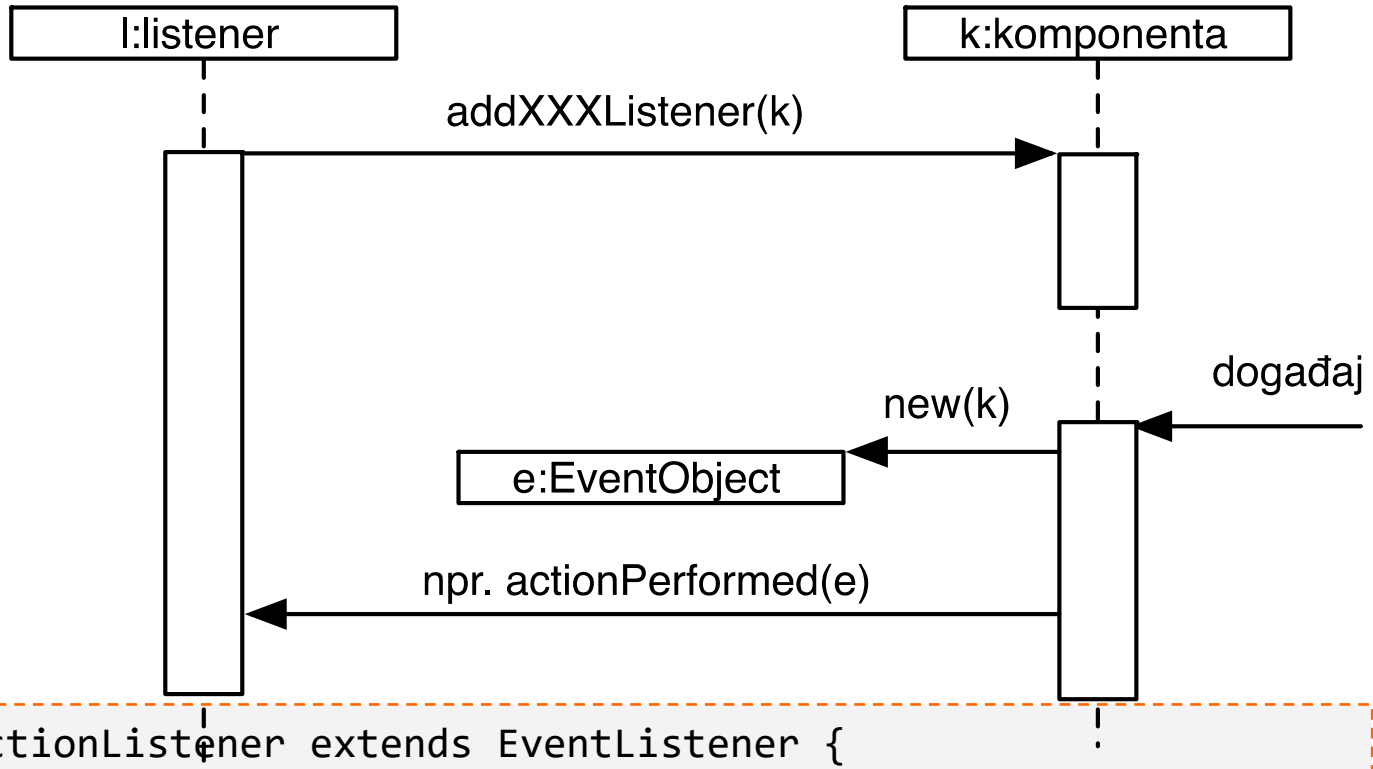
- Napomena: Gumb OK nije direktno dodan u „južni” dio, jer bi se u protivnom raširio od ruba do ruba

Model događaja (1)

- Događaj je općenito promjena stanja nekog objekta
  - npr. promjena pozicije miša na ekranu, promjena upisanog teksta, klik na gumb
- Osnovna ideja modela događaja:
  - komponente **okidaju** događaje
  - jedan ili više **promatrača** (engl. *observer*) su zainteresirani za te događaje (u *Swingu* se koristi i pojam *listener* umjesto promatrač)
  - promatrač je **pretplaćen** na događaje:
    - Uobičajeno pozivom metode oblika *addXYZListener(XYZListener)*
  - nakon okidanja događaja, sve pretplaćene komponente primaju informaciju o događaju
    - jedan od parametara poziva metode je objekt koji opisuje događaj i izveden je iz klase *java.util.EventObject*
    - *java.util.EventObject* u sebi ima referencu na objekt koji je izvor tog događaja

Model događaja (2)

1. promatrač se dodaje komponenti pozivom metode *addXXXListener*
2. komponenta okida događaj i stvaraju informacije o događaju (*EventObject*)
3. poziva se odgovarajuća metoda nad promatračem (npr. *actionPerformed* za klik na gumb)




```
public interface ActionListener extends EventListener {
    public void actionPerformed(ActionEvent e);
}
```

Obrada događaja koji predstavlja klik na gumb

15_Swing/hr.fer.oop.swing1.example1.EnterNameFrame

```
public EnterNameFrame() {
   ...
  JTextField tfInput = new JTextField();
  panel.add(tfInput, BorderLayout.CENTER);
   ...
  JButton btnOK = new JButton("OK");
  btnOK.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(EnterNameFrame.this,
                                        "Hello " + tfInput.getText());
        }
   });
}
```

Umjesto anonimne klase može se upotrijebiti lambda izraz

```
btnOK.addActionListener(e -> JOptionPane.showMessageDialog(this,
                                    "Hello " + tfInput.getText()));
```

Važniji promatrači za Swing komponente

- *component listener*
  - prati promjene komponente (veličina, pozicija, vidljivost)
- *focus listener*
  - prati fokus komponente. Kada je komponenta u fokusu onda prima događaje sa tipkovnice
- *key listener*
  - prati koja je tipka pritisnuta (događa se jedino kada neka komponenta ima fokus)
- *mouse events*
  - Prati događaje vezane uz miša (klik, pomak – ulazak/izlazak u polje komponente)
- *mouse-motion events*
  - prati pomak miša nad komponentom

Pregled promatrača i komponenti (1)

Component	Listener							
	action	caret	change	document, undoable edit	item	list selection	window	other
button	X		X		X			
check box	X		X		X			
color chooser			X					
combo box	X				X			
dialog							X	
editor pane		X		X				hyperlink
file chooser	X							
frame							X	
internal frame								internal frame
list						X		list data
menu								menu
menu item	X		X		X			menu key, menu drag mouse
password field	X	X		X				
popup menu								popup menu
progress bar			X					

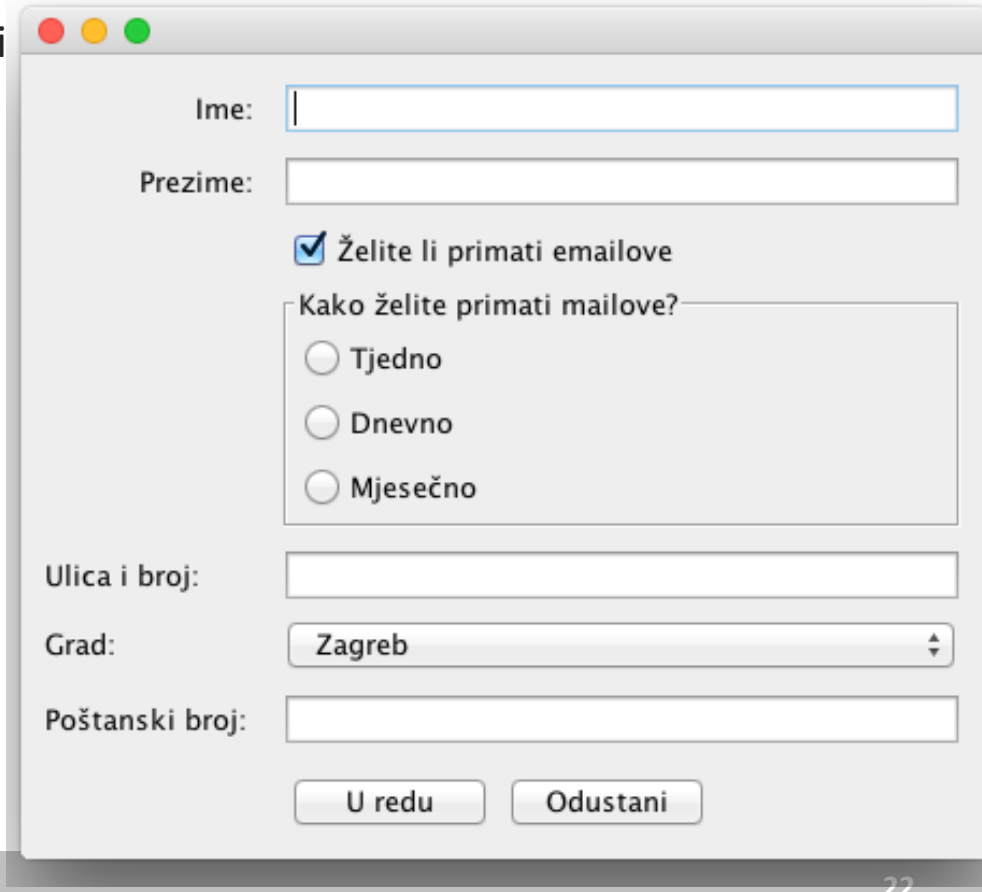
Pregled promatrača i komponenti (1)

Component	Listener							
	action	caret	change	document, undoable edit	item	list selection	window	other
radio button	X		X		X			
slider			X					
tabbed pane			X					
table						X		table model, table column model, cell editor
text area		X		X				
text field	X	X		X				
text pane		X		X				hyperlink
toggle button	X		X		X			
tree								tree expansion, tree will expand, tree model, tree selection
viewport (used by scrollpane)			X					

Primjer izrade jednog složenijeg prozora

- *BorderLayout* s panelom u centru (labele i unos) i panelom na jugu za dva gumba
- Gumbi u panelu s *GridLayoutom* (kako bi imali istu veličinu neovisno o tekstu), koji je u panelu koji koristi *FlowLayout* (da se ne raširе preko cijelog donjeg dijela) koji je onda dodan na jug glavnog prozora
  - klikom na gumb *U redu* na standardni izlaz se ispisuje podatak s forme
  - *Odustani* izlazi iz programa
- Središnji panel u zasebnoj datoteci *InputUserDataPanel* korištenjem *SpringLayouta*
  - pomoćna klasa postavlja ograničenja i širine kako bi se dobio pravilan izgled - Ne ulaziti u detalje…
- Alternativa GridLayout, ali onda bi labele zauzimale jednako mjesta kao i kontrole za unos

15_Swing/hr.fer.oop.swing1.example2.*

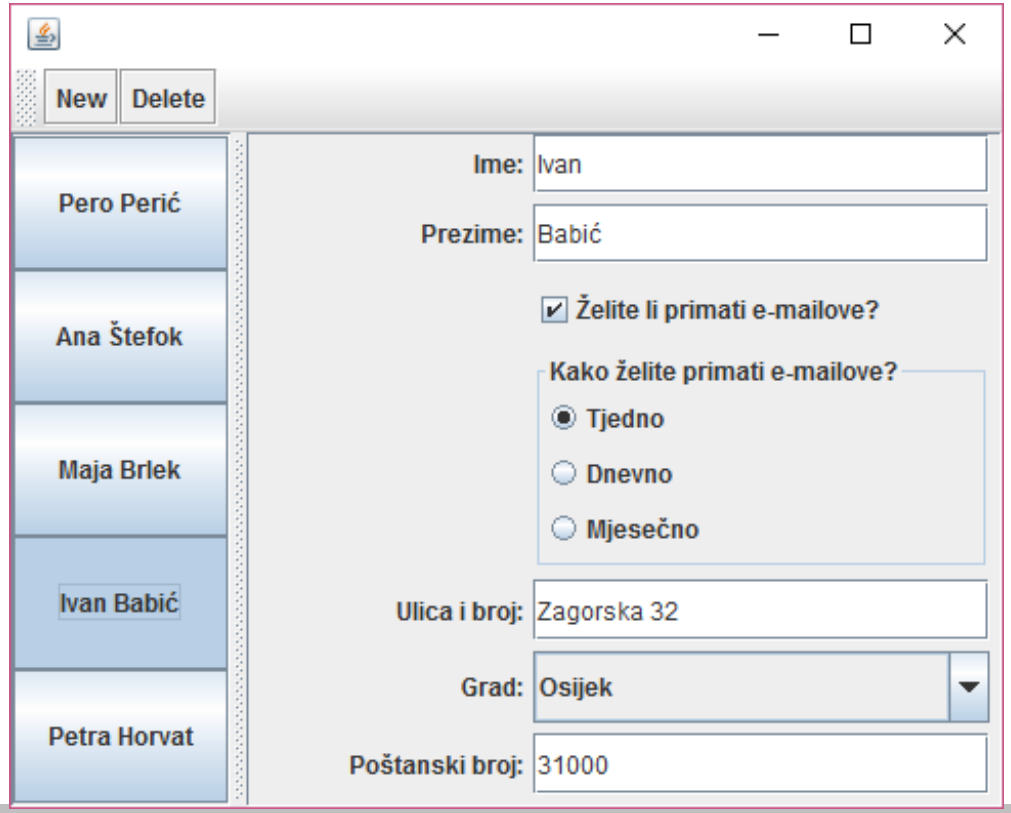


Paneli

- `JFrame` – prozor
- `JDialog` – modalni prozor
- `JPanel` – pomoćni kontejner
  - već smo ga koristili u primjerima
- `JSplitPane` – panel podijeljen na dva dijela
- `JScrollPane` – panel s klizačem (*scroll*)
- `JTabbedPane` – panel s tabovima
- `JToolBar` – panel za gumbe

Primjer korištenja više različitih panela

- Ekran podijeljen na centralni dio i toolbar u gornjem dijelu
- Centralni dio podijeljen na lijevi dio s gumbima kojima se aktivira prikaz podataka o korisniku (u centralnom dijelu)
  - Klikom na gumb *New* dodaje se novi gumb s lijeve strane i aktivira
  - *Delete* briše trenutno označeni gumb i prebacuje se na prvi
  - Promjenom imena ili prezimena i klikom na neki drugi gumb osim označenog, ažurira se tekst na gumbu



15_Swing/hr.fer.oop.swing1.example3.*

JSplitPane (1)

- Organizira komponente u 2 dijela
- Konstruktori:
  - `JSplitPane(int newOrientation)`
  - Orijentacija je `JSplitPane.HORIZONTAL_SPLIT` i `VERTICAL_SPLIT`
  - `JSplitPane(int newOrientation, boolean newContinuousLayout)`
    - `newContinuousLayout` – treba li kontinuirano iscrtavati komponente kod pomicanja granice
  - `JSplitPane(int newOrientation, Component newLeftComponent, Component newRightComponent)`
    - odmah postavlja i komponente
- Komponente se može postaviti i metodama:
  - `setLeftComponent, setTopComponent` – 1. komponenta
  - `setRightComponent, setBottomComponent` – 2. komponenta

JSplitPane (2)

- Ograničavanje minimuma veličine komponente:
  - `c.setMinimumSize(Dimension minimumSize)`
- Definiranje načina proširivanja kod promjene veličine *JSplitPanea*:
  - metoda `setResizeWeight`
- Postavljanje lokacije granice:
  - metoda: `setDividerLocation`

- za detalje korištenja pogledajte u API i *tutorial*:
  - https://docs.oracle.com/en/java/javase/13/docs/api/java.desktop/javax/swing/JSplitPane.html
  - http://docs.oracle.com/javase/tutorial/uiswing/components/splitpane.html

Primjer korištenja *JSplitPanea*

```
public SplitPaneWindow(){
  setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
  setLayout(new BorderLayout());

  JSplitPane splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
  add(splitPane, BorderLayout.CENTER);

  buttonsPanel = new JPanel();
  splitPane.setLeftComponent(buttonsPanel);

  userDataPanel = new InputUserDataPanel();
  splitPane.setRightComponent(userDataPanel);

...
```

15_Swing/hr.fer.oop.swing1.example3.SplitPaneWindow

Traka s alatima - *JToolBar*

- Može sadržavati bilo koje komponente
- Konstruktori:
  - `JToolBar(int orientation)`
    - `SwingConstants.HORIZONTAL i VERTICAL`
  - `JToolBar(int orientation, String name)`
    - ime je vidljivo kada se toolbar izdvoji u svoj prozor
- Važnije metode:
  - `addSeparator()` – dodaje razmak na kraj
  - `setFloatable(boolean b)` – da li se može izdvojiti u svoj prozor ili ne
  - `setRollover(boolean rollover)` – ne/crta okvir ako je miš nad komponentom

Primjer korištenja JToolBar-a

```
public SplitPainWindow(){
  ...
  private void createToolbar(){
    JToolBar toolbar = new JToolBar();
    add(toolbar, BorderLayout.NORTH);

    JButton newUserData = new JButton("New");
    toolBar.add(newUserData);
    newUserData.addActionListener((actionEvent) -> {
        ...
    JButton deleteUserData = new JButton("Delete");
    toolBar.add(deleteUserData);
    deleteUserData.addActionListener((actionEvent) -> {
        ...
```

15_Swing/hr.fer.oop.swing1.example3.SplitPaneWindow

Dinamičko kreiranja gumba i registracija promatrača

- Podaci o korisnicima zapisani u mapi gdje je svaki korisnik zaveden pod jedinstvenim identifikatorom, npr. pomoću klase UUID iz java.util: `UUID.randomUUID().toString()`
- Svi gumbi s lijeve strane trebaju imati isto ponašanje (prikazati podatke nekog korisnika), pa im je pridružen isti *ActionListener*
  - (vidi sljedeći slajd za programski kod listenera)
  - Gumb može imati proizvoljni string za *actionCommand* (u ovom slučaju će to biti korisnikom identifikator)

```
for (Map.Entry<String, UserData> entry : userData.entrySet()) {
        JToggleButton button = new JToggleButton(
                entry.getValue().getFirstName() + " " +
                entry.getValue().getLastName());
        button.setActionCommand(entry.getKey());
        button.addActionListener(toggleButtonListener);
        buttonsPanel.add(button);
}
```

15_Swing/hr.fer.oop.swing1.example3.SplitPaneWindow

Obrada događaja temeljem naredbi akcije (Action Command)

- Objekt u metodi za obradu događaja klika na gumb sadrži informaciju koji gumb je izazvao događaj (*getSource*) i o njegovoj naredbi (*getActionCommand*)

```
//Zajednički listener
toggleButtonListener = (actionEvent) -> {
        JToggleButton selectedButton =
                (JToggleButton) actionEvent.getSource();
        deselectOthers(selectedButton);
        String uid = actionEvent.getActionCommand();
        userDataPanel.setUserData(userData.get(uid));
        selectedButton.setSelected(true); //uvijek uključen
};
```

15_Swing/hr.fer.oop.swing1.example3.SplitPaneWindow

Traženje određene komponente u kontejneru

- Klikom na gumb za brisanje potrebno pronaći označeni gumb, ukloniti ga iz pripadajućeg panela te obrisati korisnika
  - Komponente nekog kontejnera mogu se dohvatiti s *getComponents*

```
deleteUserData.addActionListener((actionEvent) -> {
        // find selected button
        JToggleButton selectedToggleButon = null;
        for (Component c : buttonsPanel.getComponents()) {
                if (c instanceof JToggleButton) {
                        JToggleButton button = (JToggleButton) c;
                        if (button.isSelected()) {
                                selectedToggleButon = button;
                                break;
                        }
                }
        }
        buttonsPanel.remove(selectedToggleButon);
        userData.remove(selectedToggleButon.getActionCommand()); ...
```

15_Swing/hr.fer.oop.swing1.example3.SplitPaneWindow

Osvježavanje rasporeda komponenti

- Dodavanjem komponenti u kontejner (ili naknadno brisanje) nakon što je kontejner već bio iscrtan neće automatski presložiti njegove komponente
  - preslagivanje će se dogoditi npr. prilikom promjene veličine prozora
  - potrebno pozvati revalidate na kontejneru koji treba ponovo iscrtati

```
newUserData.addActionListener((actionEvent) -> {
        String uid = UUID.randomUUID().toString();
        userData.put(uid, new UserData());
        ...
        buttonsPanel.add(tb);
        System.out.println(userData.size());
        if (userData.size() == 1) {
                splitPane.getParent().revalidate();
                splitPane.setVisible(true);
        } else {
                buttonsPanel.revalidate();
        }
...
```

15_Swing/hr.fer.oop.swing1.example3.SplitPaneWindow