Objektno Orijentisano Programiranje 1

Prostorni raspored i događaji

Upravljanje prostornim rasporedom

- Cilj: obezbediti konzistentan prikaz komponenti
  - pri promeni operativnog sistema
  - pri promeni fonta
  - pri promeni geometrije kontejnera
  - pri promeni jezika
  - na uređajima različite rezolucije

Upravljanje prostornim rasporedom

- U zavisnosti od vrste, upravljači mogu a i ne moraju poštovati preferirane, minimalne i maksimalne dimenzije komponenti.
- Navedene dimenzije komponenti se mogu podešavati metodama:

  void setMinimumSize(Dimension minimumSize)

  void setMaximumSize(Dimension maximumSize)

  void setPreferredSize(Dimension preferredSize)

  Primer:

```
JButton b = new JButton(“Test”);
b.setPreferredSize(new Dimension(30, 10));
```

  void pack() – za računanje optimalnih dimenzija prozora, u skladu sa navedenim dimenzijama komponenti

Vrste upravljača

- BorderLayout
- FlowLayout
- GridLayout
- GridBagLayout
- GroupLayout
- ...

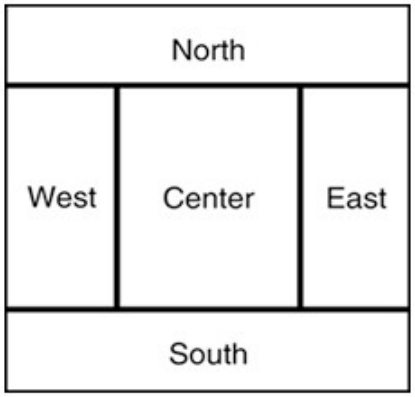
Ako nijedan ne odgovara

- Tada raditi bez upravljača.
- Primer:

```
....
//Postaviti Layout na null:
setLayout(null);
setResizable(false);
JButton ok = new JButton("Ok");
add(ok);
//Postavljati komponente direktno na koordinate:
ok.setBounds(10, 10, 30, 15);
....
```

BorderLayout

- Slaže komponente po stranama sveta (BorderLayout.NORTH, BorderLayout.SOUTH, BorderLayout.EAST, BorderLayout.WEST i BorderLayout.CENTER).
- Prvo podesi veličinu severne i južne komponente tako što uzme preferiranu visinu komponti, a za širinu uzima širinu kontejnera.
- Zatim postavi veličinu istočne i zapadne komponente tako što zadrži njihovu preferiranu širinu, dok visinu proširi na ceo preostali prostor.
- Ostatak u potpunosti dodeljuje centralnoj komponenti.



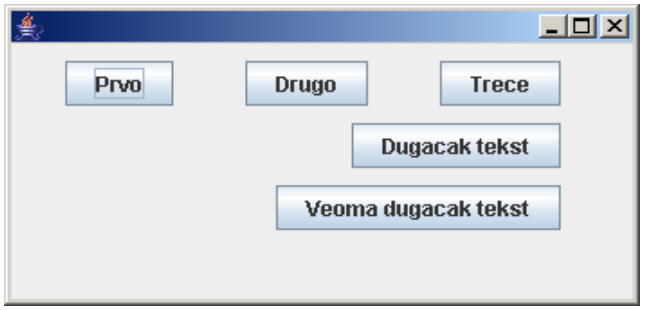
Primer 01 - BorderLayoutTest.java

BorderLayout

- Podrazumevani upravljač za JFrame je BorderLayout.
- Raspoloživi konstruktori:
  - BorderLayout()
  - BorderLayout(int hgap, int vgap) pravi razmak (hgap, vgap) u pikselima između zona)

FlowLayout

- Slaže komponente po horizontali poštujući preferiranu veličinu.
- Kada više nema mesta u tekućem redu prelazi u novi.
- Komponente može da slaže poravnato u levo, u desno i centrirano.

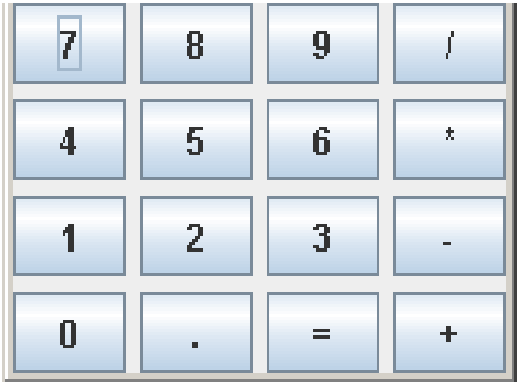


FlowLayout

- Podrazumevani upravljač za JPanel je FlowLayout.
- Kontruktori
  - FlowLayout()
  - FlowLayout(int align), gde je align poravnjanje komponenti: FlowLayout.LEFT, FlowLayout.CENTER, FlowLayout.RIGHT
  - FlowLayout(int align, int hgap, int vgap), gde je align poravnjanje, a hgap i vgap horizontalno i vertikalno rastojanje u pikselima između komponenti

GridLayout

- Raspoređuje komponente u mrežu sa zadatim brojem redova i kolona.
- Sve ćelije mreže su jednake veličine.
- Ne poštuje preferirane, minimalne ni maksimalne veličine komponenti.
- Pozicija komponente se određuje na osnovu redosleda ubacivanja u kontejner (raspored se vrši s leva na desno, odozgo na dole).



GridLayout

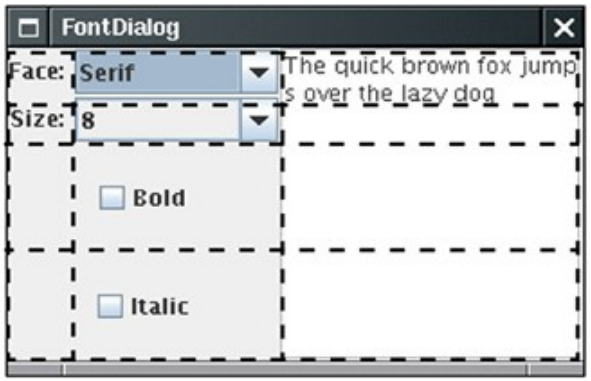
- Konstruktori
  - GridLayout()
  - GridLayout(int rows, int cols), gde su rows i cols broj redova i kolona
  - GridLayout(int rows, int cols, int hgap, int vgap), gde su rows i cols broj redova i kolona, a hgap i vgap horizontalno i vertikalno rastojanje u pikselima između ćelija

Primer 03 - GridLayoutTest.java

Primer 04 - CalculatorFrame.java

GridBagLayout

- Kompleksan upravljač koji koristi mrežu slično GridLayout-u s tom razlikom što komponente mogu zauzimati više ćelija.
- Kolone i vrste mogu imati različite dimenzije.



GridBagLayout – način unosa

- Kreirati GridBagLayout i pridružiti ga kontejneru.
- Za svaku komponentu
  - kreirati komponentu,
  - kreirati njen *constraint* objekat koji joj definiše položaj i ponašanje u okviru kontejnera i
  - uneti je u kontejner.

GridBagLayout

Primer:

```
  ..........
GridBagLayout layout = new GridBagLayout();
setLayout(layout);
GridBagConstraints constraints = new
  GridBagConstraints();
constraints.gridx = 0; //Kolona 0
constraints.gridy = 2; //Treci red
constraints.gridwidth = 2; //Zauzima 2 celije
constraints.gridheight = 1; // i jedan red
JButton button = new JButton("Ok");
getContentPane().add(button, constraints);
..............
```

GridBagLayout – parametri constraint-a

- int gridx, gridy – kolona i vrsta komponente (počevši od nule)
- int gridwidth, gridheight – u koliko kolona i redova se komponenta prostire (default 1)
- int weightx, weighty – da li se (i koliko) komponenta širi tokom promena dimenzija kontejnera po x i y osi. 0 – nema širenja
- int fill – da li se dimenzije komponente šire da zauzmu ceo rezervisani prostor ćelije. Moguće vrednosti:
  - GridBagConstraints.NONE – nema širenja (default), GridBagConstraints.HORIZONTAL – zauzima sav horizontalni prostor
  - GridBagConstraints.VERTICAL – zauzima sav vertikalni prostor
  - GridBagConstraints.BOTH – zauzima celu ćeliju po horizontali i vertikali

GridBagLayout – parametri constraint-a

- int anchor – ukoliko komponenta ne zauzima celu ćeliju, može se definisati njen položaj u okviru ćelije. Moguće vrednosti:
  - GridBagLayout.CENTER (default),
  - GridBagLayout.NORTH,
  - GridBagLayout.NORTHEAST,
  - GridBagLayout.EAST,
  - GridBagLayout.SOUTHEAST,
  - GridBagLayout.SOUTH,
  - GridBagLayout.SOUTHWEST,
  - GridBagLayout.WEST
  - GridBagLayout.NORTHWEST

GridBagLayout – parametri constraint-a

- Pravljenje praznog prostora oko komponente
  - Insets insets – minimalni razmak između ćelije i komponente
    - Insets je klasa sa atributima: int bottom, int left, int right, int top
  - int ipadx, ipady – vrednosti koje se dodaju na minimalnu veličinu komponente

GridBagLayout – preporuke za korišćenje

- Skicirati dijalog na papiru.
- Tako nacrtati matricu da najmanje komponente staju u jednu ćeliju, a veće komponente prekrivaju više redova/kolona.
- Numerisati redove i kolone na papiru i očitati gridx, gridy, gridwidth, gridheight.
- Definisati poravnanje komponenti u okviru ćelija, ukoliko komponente ne treba da zauzimaju celu ćeliju.
- Staviti weight svih komponenti na 100. Pokrenuti prozor. Ukoliko se tada utvrdi da neke komponente ne treba da se šire, staviti im weight 0.

Primer 05 - GridBagLayoutTest.java

Primer 05 - FontDialogFrame.java

Event driven model

- Program se ne izvršava linearno (od gore prema dole)
- Pišu se metode koje se automatski izvršavaju po pojavi nekog događaja korisničkog interfejsa (klik mišem, pritisak tastera)
- Program ima
  - inicijalizacioni blok i
  - blokove koda koji reaguju na događaje korisničkog interfejsa

Event driven model

- Svaka akcija nad komponentama korisničkog interfejsa izaziva generisanje događaja (instanci naslednika *EventObject* klase).
  - navedene komponente korisničkog interfejsa se u ovom kontekstu nazivaju izvori događaja (*event sources*)

- Događaji se prosleđuju svim “osluškivačima” događaja koji su se kod izvora događaja registrovali da ih dati događaj zanima. Registrovanje se obavlja u dva koraka:
  - implementacijom odgovarajućeg *EventListener* interfejsa (čime određena klasa postaje osluškivač) i
  - povezivanjem sa izvorom.

Realizacija Listener-a

- Varijanta 1:

```
JButton bOK = new JButton("OK");
MyListener listener = new MyListener();
bOK.addActionListener(listener);

...

public class MyListener implements ActionListener {
  //ActionPerformed je jedina metoda ActionListener interfejsa
  public void actionPerformed(ActionEvent ev) {
    // Ovde se smešta kod koji se izvršava kada se klikne
    System.exit(0);
  }
}
```

Realizacija Listener-a

- Varijanta 2:

```
JButton b = new JButton("Pritisni me");
ActionListener al = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    ...
  }
};
b.addActionListener(al);
```

- Varijanta 3:

```
JButton b = new JButton("Pritisni me");
b.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    ...
  }
});
```

primer07 Listener2

primer08 Listerner3

Realizacija Listener-a

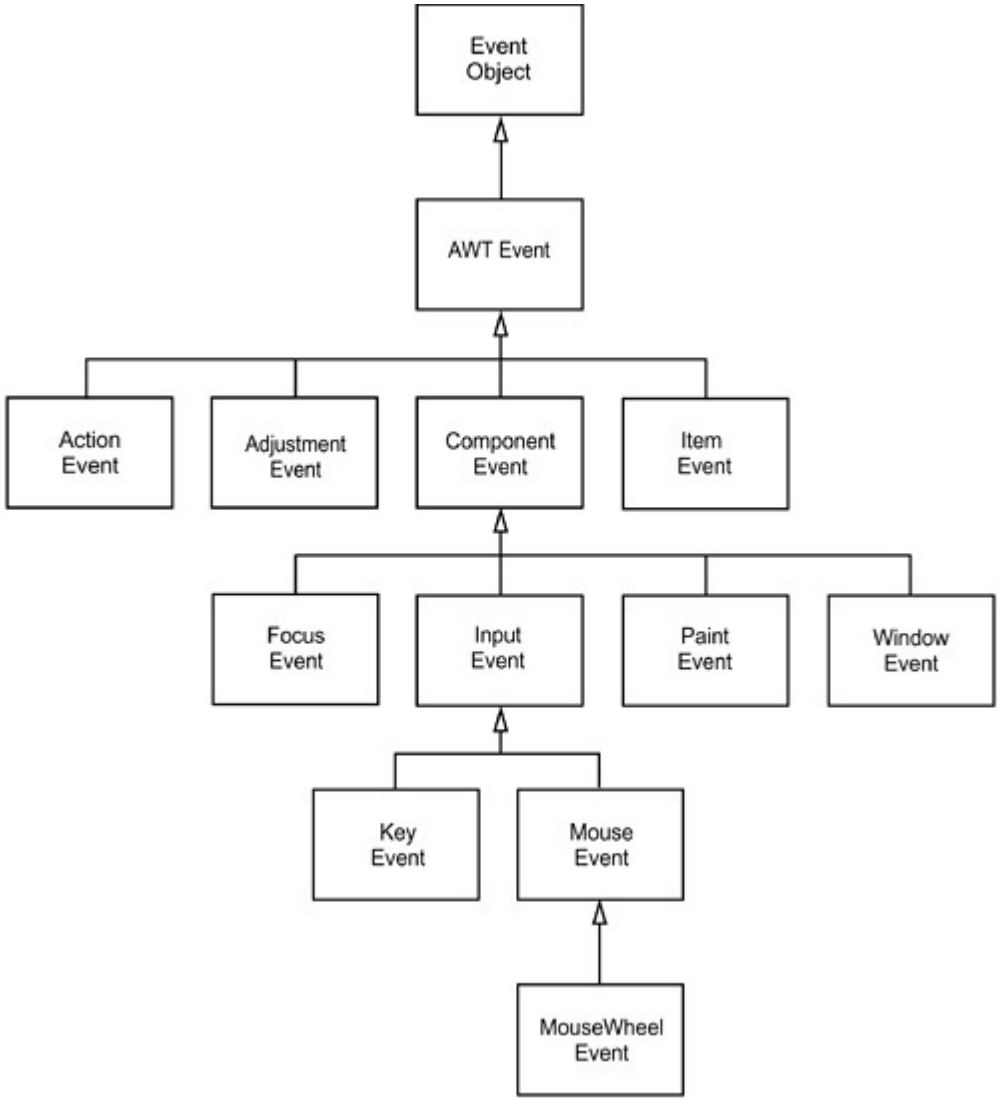
- Varijanta 4:

```
public class MainFrame extends JFrame implements
  ActionListener {
  ...
  @Override
  public void actionPerformed(ActionEvent ev) {...}
}
```

primer09 Listener4

Hijerarhija događaja

Za svaku vrstu događaja postoji odgovarajuća xxxEvent klasa:



Vrste događaja

A za svaku xxxEvent klasu postoji bar jedan xxxListener interfejs, npr:

Događaj	Dodeljeni Listener
ActionEvent	ActionListener
AdjustmentEvent	AdjustmentListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener
KeyEvent	KeyListener
MouseEvent	MouseListener, MouseMoveListener
WindowEvent	WindowListener, WindowStateListener
ItemEvent	ItemListener
TextEvent	TextListener

Adapter klase

- Adapter klase su uvedene za sve xxxListener interfejse koje imaju više od jedne metode, sa ciljem da olakšaju pisanje reakcija na događaje.
- One implementiraju *Listener interfejse i obezbeđuju podrazumevane reakcije na događaje (najčešće prazna tela metoda)

primer10 WindowListener

primer11 WindowAdapter

Adapter klase

- Primeri Adapter klasa:
  - FocusAdapter
  - MouseMotionAdapter
  - KeyAdapter
  - WindowAdapter
  - MouseAdapter

Šta se tačno dešava?

- Klasa koja generiše događaje ima add*Listener metode i remove*Listener metode putem kojih se objekti zainteresovani za odgovarajuće događaje mogu registrovati/deregistrovati.
- Klase koje generišu događaje, za tu svrhu imaju metode fire* kojima se registrovanim osluškivačima javlja da se događaj desio. Metode za pristup registrovanim osluškivačima su:
  - getListeners - generička metoda koja prima kao parametar *class* objekat traženog listenera,
  - get*Listeners - specijalizovane metode za konkretne vrste osluškivača.

Šta se tačno dešava?

- java.swing.event.EventListenerList - predstavlja pomoćnu klasu koja u nizu čuva parove objekata (class objekat osluškivača, osluškivač).
- Koristi se za podršku metodama add*Listener i fire*.
- Ukoliko se pozove metoda fire* prolazi se kroz listu osluškivača i za svaki osluškivač koji pripada odgovarajućoj klasi kreira se event i prosleđuje odgovarajućoj metodi osluškivača.

java.awt.EventDispatchThread

- Svaka awt ili swing aplikacija počinje sa dve niti. U jednoj se izvršava main metoda a u drugoj se obrađuju događaji(**E**vent-**D**ispatching **T**hread).
- Događaji prolaze kroz FIFO koji je instanca java.awt.EventQueue klase.
- Zbog konzistentnosti događaji se obrađuju serijski i bitno je da se slanje svih događaja obavlja iz EDT
  - zgodna statička metoda je SwingUtilities.isEventDispatchThread

java.awt.EventDispatchThread

- Problem je ako se iz listenera startuje dugotrajan posao sa potrebom da se ažurira stanje GUI komponenti.
  - "zamrzava" ekran
- Jedno rešenje je uvođenje niti.
- Najbolje rešenje je upotreba SwingWorker klase

primer12 Blokiran EDT

primer13 Resenje upotrebom niti

SwingWorker

- Klasa napravljena za rešavanje problema blokiranja EDT.
- Nasledi se klasa SwingWorker i redefinišu se metode doInBackground, process i finished.
- Startuje se pozivom metode execute(), a zaustavlja pozivom metode cancel()
  - metoda isCancelled() vraća true ako je rad prekinut

Rezime1/3

Interfejs	**Metode interfejsa**	**Događaj i njegove metode**	**Komponenta koja generiše događaj**
ActionListener	actionPerformed	**ActionEvent** getActionCommand getModifiers	AbstractButton JComboBox JTextField Timer
AdjustmentListener	adjustmentValueChanged	**AdjustmentEvent** getAdjustable getAdjustmentType getValue	JScrollbar
ItemListener	itemStateChanged	**ItemEvent** getItem getItemSelectable getStateChange	AbstractButton JComboBox
FocusListener	focusGained focusLost	**FocusEvent** isTemporary	Component

Rezime 2/3

Interfejs	**Metode interfejsa**	**Događaj i njegove metode**	**Komponenta koja generiše događaj**
KeyListener	keyPressed keyReleased keyTyped	**KeyEvent** getKeyChar getKeyCode getKeyModifiers getKeyText isActionKey isShiftDown	Component
MouseListener	mousePressed mouseReleased mouseEntered mouseExited mouseClicked	**MouseEvent** getClickCount getX getY getPoint TranslatePoint	Component
MouseMotionListener	mouseDragged mouseMoved	**MouseEvent**	Component
MouseWheelListener	mouseWheelMoved	**MouseWheelEvent** getWheelRotation getScrollAmount	Component

Rezime 3/3

Interfejs	**Metode interfejsa**	**Događaj i njegove metode**	**Komponenta koja generiše događaj**
WindowListener	windowClosing windowOpened windowIconified windowDeiconified windowClosed windowActivated windowDeactivated	**WindowEvent** getWindow	Window
WindowFocusListener	windowGainedFocus windowLostFocus	**WindowEvent** getOppositeWindow	Window
WindowStateListener	WindowStateChanged	**WindowEvent** getOldState getNewState	Window