

Java tečaj

8. dio

Swing

Uvod

- ◆ Swing == grafičko korisničko sučelje
- ◆ Jezgra temeljena na starijoj implementaciji korisničkog sučelja: AWT (Abstract Windows Toolkit)
(vidljivo npr. Po stablu nasljeđivanja)
- ◆ Dodan niz mogućnosti i unapređenja
- ◆ Postoji još i JavaFX – nećemo ga razmatrati u okviru ove vještine

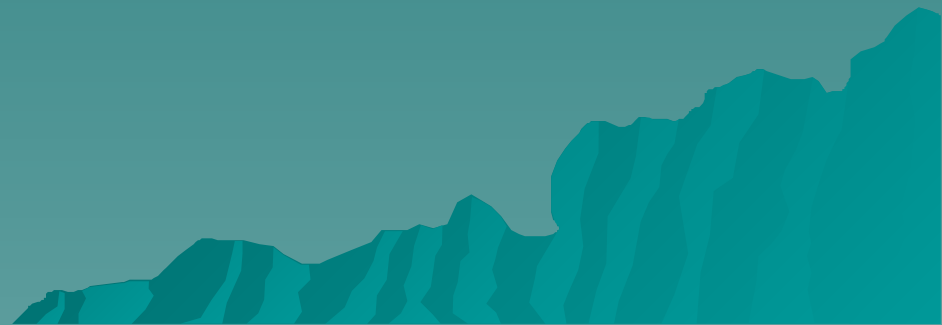
Vršni elementi

- ◆ Swing definira dva vršna elementa (engl. Top level containers):
 - **JFrame** – standardni prozor
 - **JDialog** – dijaloška kutija

Vršni elementi: JFrame

- ◆ Loš primjer otvaranja prozora (zašto loš – vidi slide 16)

OtvaranjaProzora.java



Vršni elementi : JFrame

P01

```
public class OtvaranjeProzora {  
  
    public static void main(String[] args) {  
        JFrame frame = new JFrame();  
  
        frame.setLocation(20, 20);  
        frame.setSize(500, 200);  
  
        frame.setVisible(true);  
    }  
}
```

Vršni elementi : JFrame

- ◆ Prikazani prozor prilikom “ubijanja” zapravo ne nestaje – samo se skriva
- ◆ Posljedica je da aplikacija (proces) i dalje radi, iako korisnik više ništa ne može napraviti

Vršni elementi : JFrame

- ◆ AWT `Frame` po defaultu ignorira korisnikov pokušaj zatvaranja prozora
- ◆ Swing za `JFrame` definira metodu

`setDefaultCloseOperation(int)`

```
setDefaultCloseOperation(  
    WindowConstants.DO_NOTHING_ON_CLOSE  
)
```

Vršni elementi : JFrame

P02

```
public class OtvaranjeProzora {  
  
    public static void main(String[] args) {  
        JFrame frame = new JFrame();  
  
        frame.setLocation(20, 20);  
        frame.setSize(500, 200);  
        frame.setDefaultCloseOperation(  
            WindowConstants.DISPOSE_ON_CLOSE);  
  
        frame.setVisible(true);  
    }  
}
```


Vršni elementi : JFrame

- ◆ **WindowConstants** navodi nekoliko vrijednosti:
 - **DO_NOTHING_ON_CLOSE** – ignorira zahtjev za zatvaranjem
 - **HIDE_ON_CLOSE** – samo sakrij prozor
 - **DISPOSE_ON_CLOSE** – sakrij prozor pa oslobodi sve njegove resurse
 - **EXIT_ON_CLOSE** – terminiraj samu aplikaciju (može izazvati iznimku!)

Vršni elementi : JFrame

- ◆ Preporučam uporabu:

`DISPOSE_ON_CLOSE`

ili

`EXIT_ON_CLOSE`

(ako je to stvarno nužno)

Višedretvenost?

- ◆ Pokretanjem korisničkog sučelja aplikacija automatski dobiva nove dretve:

- Java2D Disposer
- AWT-Shutdown
- AWT-Windows

- AWT-EventQueue-0

← Event Dispatching Thread (EDT)

Višedretvenost?

- ◆ S obzirom na višedretvenost, SVE poslove oko rada s korisničkim sučeljem **MORA** obavljati samo jedna dretva:
event dispatching thread (EDT)
- ◆ Potporu za ovo nudi razred `SwingUtilities`, u okviru metoda `invokeLater` te `invokeAndWait`

Višedretvenost?

- ◆ Posao koji treba odraditi EDT a koji se definira u nekoj drugoj dretvi modelira se kao `Runnable` objekt, i referenca na taj posao predaje se kroz `SwingUtilities`, metode `invokeLater` te `invokeAndWait` koje će ga predati EDT-u na izvođenje (OO *Strategija!*)

Vršni elementi : JFrame

- ◆ Umjesto da netko "izvana" mijenja sadržaj `JFrame`-a, obično se prozor izvodi kao novi razred koji nasljeđuje `JFrame`

Primjer: Prozor1.java

Vršni elementi : JFrame

```
public class Prozor1 extends JFrame {  
  
    public Prozor1() {  
        ...  
    }  
  
    protected void initGUI() {  
        ...  
    }  
  
    public static void main(String[] args) {  
        ...  
    }  
}
```

Vršni elementi : JFrame

```
public static void main(String[] args) {  
    SwingUtilities.invokeLater(  
        new Runnable() {  
            public void run() {  
                Prozor1 prozor = new Prozor1();  
                prozor.setVisible(true);  
            }  
        }  
    );  
}
```

Ili kraće lambdom:

```
SwingUtilities.invokeLater(  
    () -> new Prozor1().setVisible(true)  
);
```


Vršni elementi : JFrame

```
public Prozor1() {  
    super();  
    setDefaultCloseOperation(  
        WindowConstants.DISPOSE_ON_CLOSE);  
    setTitle("Prozor1");  
    setLocation(20, 20);  
    setSize(500, 200);  
    initGUI();  
}
```

Vršni elementi : JFrame

```
protected void initGUI() {  
    // kad krenemo s komponentama  
    // koje prozor prikazuje...  
}
```

Elementi korisničkog sučelja

Grafičke komponente koje se dodaju na površinu top-level komponente u Swingu izveden je iz razreda `JComponent`

<https://docs.oracle.com/javase/8/docs/api/javax/swing/JComponent.html>



Elementi korisničkog sučelja

Grafičke komponente se u vršnu komponentu (ili druge kontejnere) dodaju nekom od metoda `add()`.

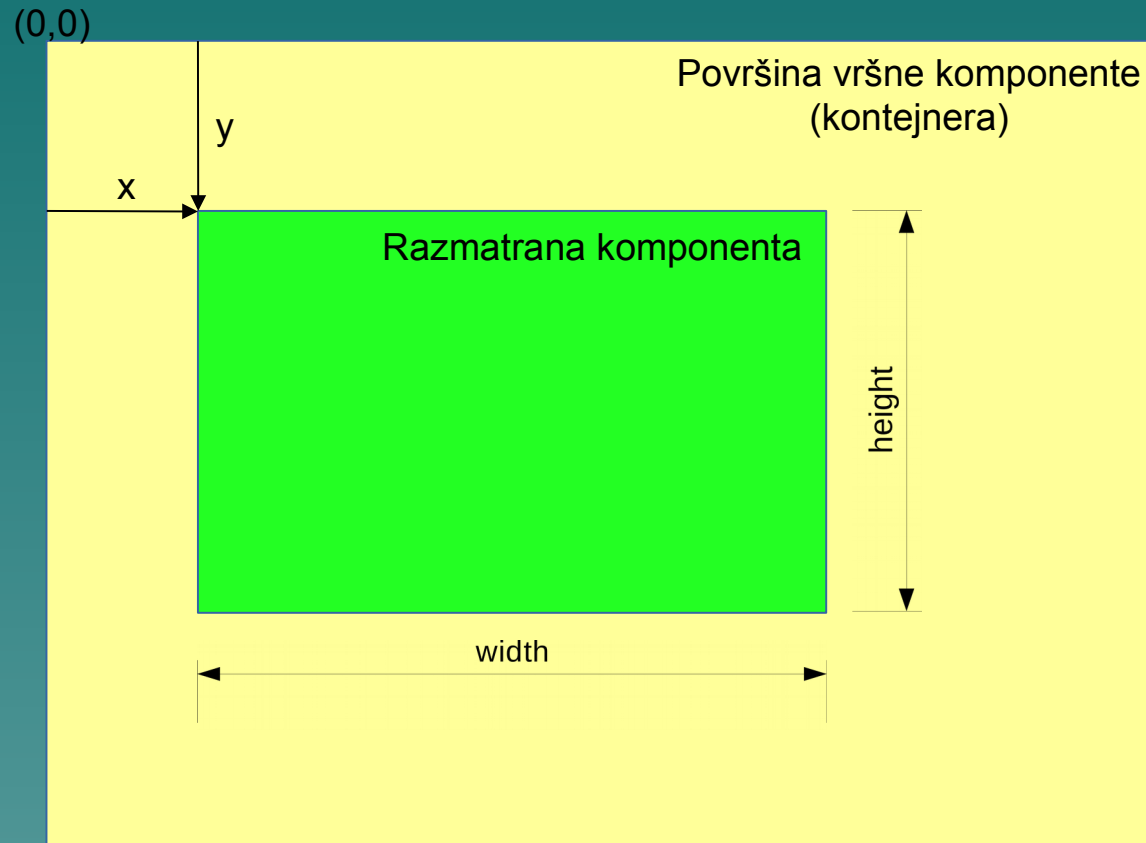
Unutar `JFrame`-a pozivamo
`getContentPane().add(...)`

Elementi korisničkog sučelja

◆ Koje nam metode nudi JComponent?

- `Point getLocation()` ;
`void setLocation(int x, int y)` ;
`void setLocation(Point p)` ;
`// Point: public int x, y;`
- `Dimension getSize()` ;
`void setSize(int w, int h)` ;
`void setSize(Dimension d)` ;
`// Dimension: public int width, height;`

Elementi korisničkog sučelja



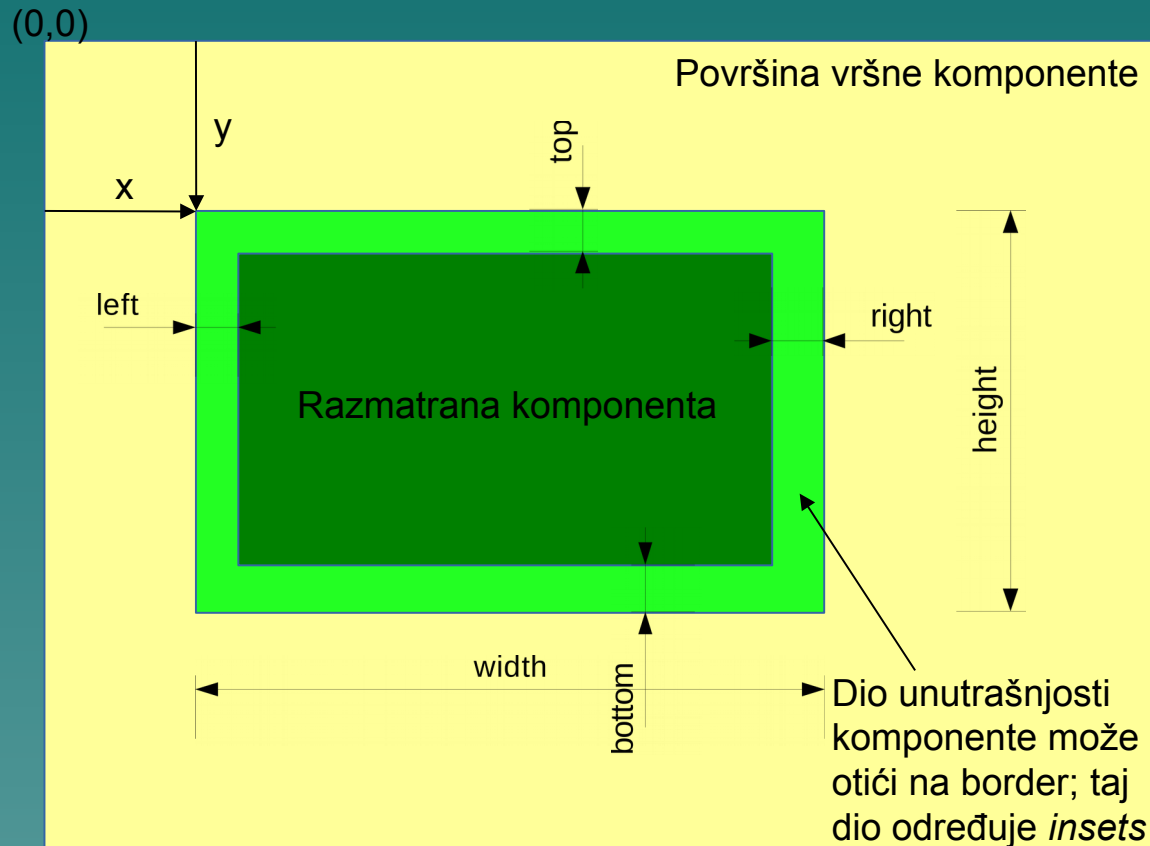
“Vršne” komponente još zovemo *kontejnerima*

Elementi korisničkog sučelja

◆ Koje nam metode nudi JComponent?

- `Rectangle getBounds() ;`
`void setBounds(int x, int y, int w, int h) ;`
`void setBounds(Rectangle r) ;`
`// Rectangle:`
`public int x, y, width, height;`
- `Insets getInsets() ;`
`// Insets:`
`public int left, right, top, bottom;`

Elementi korisničkog sučelja



Definiranjem bordera za komponentu, komponenti ostaje manje površine za prikaz korisnog sadržaja.

Elementi korisničkog sučelja

- ◆ Koje nam metode nudi `JComponent`?
 - Prilikom crtanja slike komponente, komponenta bi trebala “slikati” samo po dijelu svoje površine koji je umanjen za trenutno postavljene *insete*

Elementi korisničkog sučelja

◆ Koje nam metode nudi `JComponent`?

- `Border` `getBorder()` ;
`void setBorder(Border border)` ;
`// Border je sučelje (sljedeći slide)`
- Stvaranje bordera:
 - izvođenjem razreda (uvijek opcija!)
 - Uporabom metoda tvornica:
`BorderFactory.createXXX(...)`

Elementi korisničkog sučelja

◆ Sučelje Border:

- Insets `getBorderInsets(Component c);`
`boolean isBorderOpaque();`
`paintBorder(Component c, Graphics g, int x, int y, int w, int h);`
- `Graphics` je objekt koji nudi primitive za crtanje

Elementi korisničkog sučelja

P04

- ◆ PRIMJER: apsolutno pozicionirana komponenta zadanih dimenzija s borderom

dodajemo primjerak komponente izvedene iz `JComponent` jer je `JComponent` apstraktan

omogućiti apsolutno pozicioniranje naredbom:
`getContentPane().setLayout(null);`

Elementi korisničkog sučelja

- ◆ Koje nam metode nudi `JComponent`?
 - Upravljanje prozirnošću pozadine
`setOpaque(boolean opaque);`
`boolean isOpaque();`
 - `Color` `getBackground();`
`void setBackground(Color color);`
`// JComponent ne poštuje opaque...`
 - `Color` `getForeground();`
`void setForeground(Color color);`

Elementi korisničkog sučelja

◆ Koje nam metode nudi `JComponent`?

– Omogućenost

```
setEnabled(boolean enabled);  
boolean isEnabled();  
// različit prikaz komponente?
```

– Može li imati fokus

```
setFocusable(boolean focusable);  
boolean isFocusable();  
// primanje ulaza od korisnika...
```

Elementi korisničkog sučelja

- ◆ Koje nam metode nudi `JComponent`?
 - Za izradu "slike" komponente zadužena je metoda `void paint(Graphics g);`
 - Ona poziva lanac
`paintComponent(g);`
`paintBorder(g);`
`paintChildren(g);`
 - Ovaj posljednji poziv koristi se kada je komponenta kontejner za druge komponente
 - Specijalizirane komponente trebaju nadjačati `paintComponent(...)`

Elementi korisničkog sučelja

- ◆ Koje nam metode nudi `JComponent`?
 - Metode `paintXXX(Graphics g)` dobivaju grafički objekt koji nudi primitive za crtanje (linija, ovala, lukova, teksta, ...)
 - Objekt koji dobivamo sigurno će biti primjerak razreda `Graphics2D` pa ga možemo ukalupiti
 - ◆ On nudi još bogatiji API

Elementi korisničkog sučelja

P05

- ◆ PRIMJER: komponenta koja na površini ima nacrtanu elipsu

`Graphics` je objekt sa stanjem. Trenutna boja se postavlja sa `setColor(Color.xxx)` i nju koriste primitivi za crtanje sve dok se ne promijeni. Isto vrijedi i za trenutni font.

Nova grafička komponenta: sat

P06

◆ Općenito

- Svaka komponenta zadužena je za održavanje svojeg stanja (ne slike!)
- Kad god se stanje promijeni, potrebno je nad komponentom pozvati `repaint()`: to je signal EDT-u da bi trebalo ponovno osvježiti prikaz te komponente (kad stigne!)
- EDT će nad komponentom pozvati `paint()` koji dalje zove `paintComponent()` → to pregazimo!

Nova grafička komponenta: sat

◆ Općenito

- Trenutno vrijeme čini stanje
- `paintComponent(g)` crta to stanje
- imamo pomoćnu dretvu koja periodički osvježava stanje
- promjena stanja zakazuje novo crtanje pozivom `repaint()`
- Problem: ako je osvježavanje stanja prečesto, EDT neće umrijeti...

Nova grafička komponenta: sat

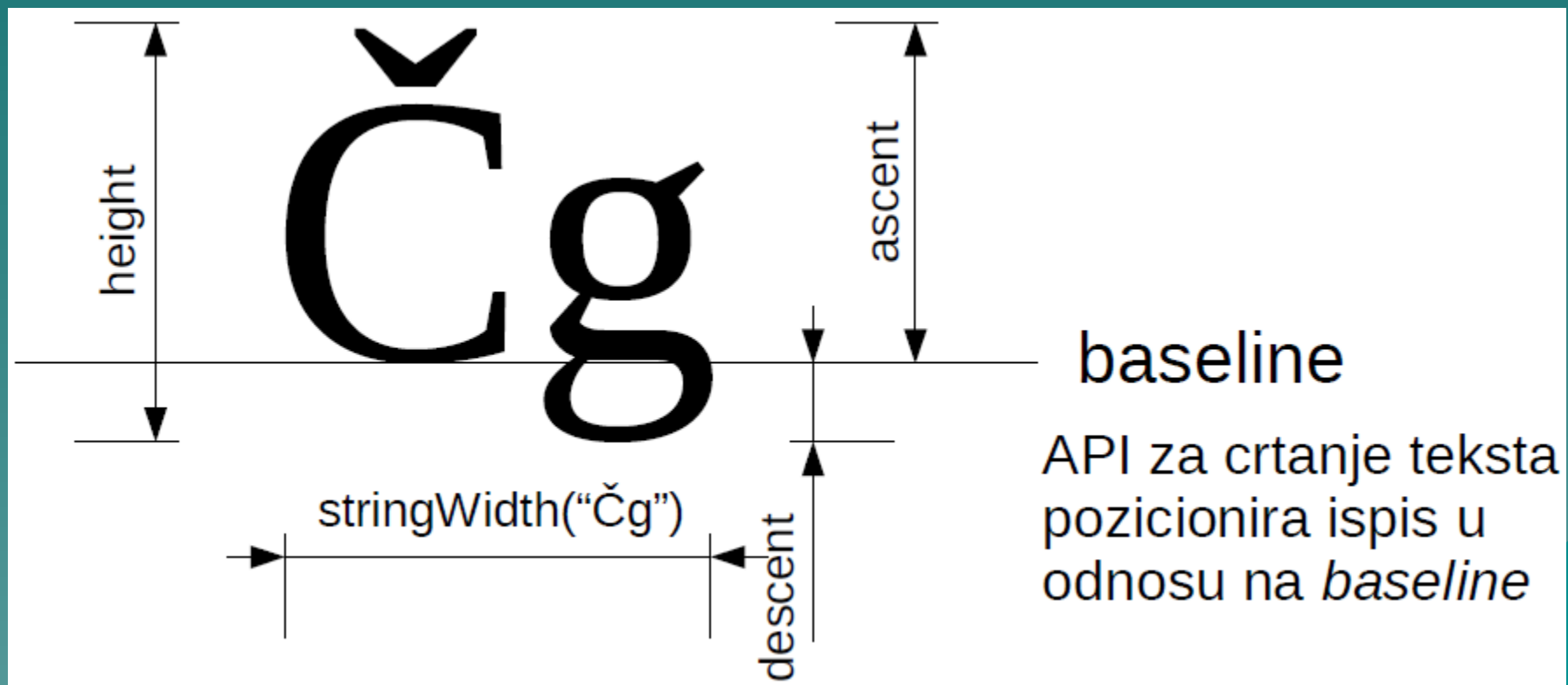
◆ Informacije o trenutnom fontu:

```
FontMetrics fm = g.getFontMetrics();
```

- `fm.getAscent()`
- `fm.getDescent()`
- `fm.getHeight()`
- `fm.stringWidth(text)`

Nova grafička komponenta: sat

◆ Informacije o trenutnom fontu:



Obrada događaja

- ◆ Obilna uporaba oblikovnog obrasca **Promatrač**: komponenta je subjekt
 - Promatrači (definirani kroz **sučelja**)
 - ◆ `MouseListener`
 - ◆ `MouseMotionListener`
 - ◆ `MouseWheelListener`
 - ◆ `KeyListener`
 - ◆ `FocusListener`
 - ◆ `ComponentListener`
 - ◆ ...

Obrada događaja

- ◆ Obilna uporaba oblikovnog obrasca
Promatrač: komponenta je subjekt
 - Komponenta nudi metode za prijavu i odjavu promatrača
npr. `addKeyListener(...)`
 - Obično postoje i prazne implementacije tih sučelja (**razredi**) koji omogućavaju pisanje manje koda (metode koje ne trebate ne spominjete); adapteri
 - `MouseAdapter`, `MouseMotionAdapter`, `KeyAdapter`, ...

Obrada događaja

P07

- ◆ Popravimo sat iz prethodnog primjera tako da se zaustavi kada se prozor gasi
 - Implementirati `WindowListener` (odnosno `WindowAdapter`) koji će dretvi sata signalizirati da se ugasi

Obrada događaja

P08

- ◆ PRIMJER: crtanje linija klikom miša
 - Za vježbu
 - Dodati `MouseListener` koji prati klikove miša
 - Dodati model linije (novi razred) i takve objekte stavljati u internu kolekciju linija
 - Komponenta to crta...

Elementi korisničkog sučelja

Swing nudi niz gotovih elemenata (komponenti, kontrola) koje se mogu koristiti (i prilagođavati!)

<https://docs.oracle.com/javase/tutorial/uiswing/components/index.html>

A stylized silhouette of a mountain range in shades of teal, located in the bottom right corner of the slide.

Elementi korisničkog sučelja

P09

- ◆ Dodajmo u prozor jednu labelu i jedan gumb (`JLabel`, `JButton`)

Primjer: `Prozor2.java`

Elementi korisničkog sučelja

- ◆ Ograničenje ovakvog pristupa:
 - Pozicioniranje je apsolutno, širina prozora obično nije → izgleda loše pri promjeni veličine prozora!
- ◆ Kako bi riješili problem, koriste se komponente čija je zadaća automatsko pozicioniranje komponenti → **Layout Manager**

Layout manager

- ◆ Komponenta koja sama prema određenim pravilima razmješta druge komponente po raspoloživom prostoru
- ◆ Za potrebe izrade razmještaja sve komponente Layout Manageru nude:
 - `Dimension getMinimumSize()`
 - `Dimension getMaximumSize()`
 - `Dimension getPreferredSize()`

Layout manager

- ◆ Java nudi nekoliko gotovih Layout Managera
 - **Border Layout**
 - Box Layout
 - Card Layout
 - **Flow Layout**
 - GridBag Layout
 - **Grid Layout**
 - Spring Layout

Layout manager

- ◆ Layout Manager je definiran kroz sučelje `java.awt.LayoutManager` te njegovo proširenje `java.awt.LayoutManager2`
- ◆ U nedostatku prikladnog Layout Managera, ponekad je jednostavnije napisati svoj vlastiti izvođenjem prikladnog sučelja

Elementi korisničkog sučelja

P10

- ◆ Pozicioniranje labele i gumba uporabom `BorderLayout-a`

Primjer: `Prozor3.java`



Elementi korisničkog sučelja

P11

- ◆ Složenije pozicioniranje – koristiti `JPanel` kao kontejner drugih komponenti koji ima svoj vlastiti Layout Manager
- ◆ Dodajmo još tri gumba!

Primjer: `Prozor4.java`



Elementi korisničkog sučelja

- ◆ Kada se pritisne gumb, generira se *dogadjaj* (engl. *event*) koji opisuje što se točno dogodilo
- ◆ Slično kao i za niz drugih situacija za koje se generira *dogadjaje*:
 - Pomak miša, pritisak/otpuštanje tipke miša, klik miša, ...
 - Pritisak/otpuštanje tipke na tipkovnici, klik neke tipke, ...

Elementi korisničkog sučelja

- ◆ I dalje:
 - oblikovni obrazac **Promatrač**
- ◆ Na "klik" gumba (ili razmaknicu dok je gumb fokusiran) generira se događaj kojim se dojavljuje da je zatraženo izvođenje akcije
- ◆ Promatrača dodajemo pozivom:
 - `addActionListener`

Elementi korisničkog sučelja

- ◆ Promatrač za akciju: `ActionListener`

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

- ◆ `ActionEvent` objekt enkapsulira različite informacije o događaju: izvor događaja, stanje tipki CTRL/ALT/SHIFT, trenutak kada je događaj izazvan, ...

Elementi korisničkog sučelja

P12

- ◆ Primjer: Kada korisnik pritisne gumb, u labeli treba ispisati koji je gumb bio pritisnut

Primjer: Prozor5.java

Elementi korisničkog sučelja

P13


- ◆ Napraviti prozor koji ima `JTextField`, lijevo od njega "Unesi broj:", desno od njega gumb "Izračunaj" a iznad njega labelu. Kada korisnik pritisne gumb, u labeli treba ispisati kvadrat broja koji je korisnik upisao u `JTextField`! Ako nastupi problem, dojaviti to korisniku
(`JOptionPane.showMessageDialog(...)`)

Elementi korisničkog sučelja

P14

- ◆ Za vježbu:
TextReader
- ◆ Program ima dva gumba i jedan tekstualni editor (JTextArea)
- ◆ Gumbi su učitaj (učitava neku preddefiniranu datoteku) i sortiraj (čijim se aktiviranjem retci prikazani u editoru sortiraju)
- ◆ Proširiti s dijalogom za odabir datoteke (JFileChooser#showOpenDialog)

Prikaz podataka i modeli

- ◆ Swing obilato koristi obrazac MVC
 - ◆ Komponente koje prikazuju podatke (poput lista, tablica, stabala) su *pogledi*, i one su razdvojene od samih podataka i upravljanja podacima (koji su enkapsulirani u *model*)
 - ◆ Tako ista komponenta može prikazivati podatke iz datoteke, baze, memorije, ...
- 
- A stylized, layered mountain range graphic in shades of teal and blue, located in the bottom right corner of the slide.

Prikaz podataka i modeli

- ◆ Kako bi prikazi postali svjesni da su se podatci promijenili, oni se registriraju kao promatrači nad model (subjekt)
→ opet oblikovni obrazac ***Promatrač***

Prikaz podataka i modeli

- ◆ Za modele se obično nudi:
 - Sučelje koje propisuje traženu funkcionalnost
(ListModel, TableModel, ...)
 - Apstraktna implementacija modela (apstraktni razred) koja već nudi prijavu/odjavu promatrača te metode za generiranje i slanje obavijesti
(AbstractListModel, AbstractTableModel, ...)

Prikaz podataka i modeli

- ◆ Za modele se obično nudi:
 - Potpuna implementacija modela koja se može koristiti za neke tipične zadatke (`DefaultListModel`, `DefaultTableModel`, ...)
 - ◆ Implementacija već na neki način interno pamti podatke te nudi dodatne metode za dodavanje, brisanje i modificiranje podataka uz automatsko slanje obavijest prijavljenim promatračima

Prikaz podataka i modeli

- ◆ To ćemo pogledati na primjeru komponente `JList`
- ◆ Važni razredi/sučelja:
 - `JList` (pogled / konkretan promatrač)
 - `ListModel` (sučelje koje opisuje model podataka / subjekt)
 - `ListDataListener` (apstraktni promatrač)
 - `ListDataEvent` (opis događaja: kako su podatci u modelu promijenjeni)

Prikaz podataka i modeli

P15

◆ Primjer:

- Napravite vlastiti model liste i potom aplikaciju koja prikazuje dva gumba i listu
- Na klik prvog gumba u listu se dodaje slučajno generirani cijeli broj
- Na klik drugog gumba briše se stavka odabrana u listi (ako takva postoji)
- Dekorirati listu u `JScrollPane`

Rad sa slikama

- ◆ Slika (bitmapa) je u Javi modelirana razredom **BufferedImage**
 - Sliku možemo učitati s diska

```
BufferedImage bim =  
    ImageIO.read(new File("jabuka.png")) ;
```
 - Nudi se i čitanje iz `InputStream`-a, `URL`-a (dohvat s Interneta)
 - `ImageIO` nudi i `write` kojom sliku može pohraniti na disk u zadanom formatu (png, jpg, ...)

Rad sa slikama

- ◆ Slika (bitmapa) je u Javi modelirana razredom `BufferedImage`

- Sliku možemo stvoriti u memoriji:

```
BufferedImage bim =  
    new BufferedImage(  
        500, 300, // dimenzije  
        BufferedImage.TYPE_3BYTE_BGR  
    );
```

- `TYPE_4BYTE_ABGR`,
`TYPE_BYTE_GRAY`,
...

Rad sa slikama

- ◆ Slika (bitmapa) je u Javi modelirana razredom `BufferedImage`
 - Jednom kada imamo sliku, možemo:
 - Zatražiti objekt za crtanje
`Graphics2D g = bim.createGraphics();`
 - Crtati
`g.drawLine(10,10,20,20);`
 - Uništiti objekt za crtanje
`g.dispose();`

Rad sa slikama

- ◆ Objekt `Graphics` nudi primitive kojima može po objektu za koji je stvoren nacrtati drugu sliku (uz mogućnost crtanja samo dijela slike, skaliranja slike, ...)

```
g.drawImage(  
    Image img, int x, int y,  
    ImageObserver observer  
);
```

i varijante...