Föreläsning 17

- LayoutManager
  BorderLayout
  GridLayout
  BoxLayout
  null Layout
- MouseListener
- Border
- Grafiskt program med flera klasser
- Bild och ljud

JF 6.3
6.4 Mouse Events
6.5
6.6 Borders

LayoutManager

När man ska placera ut komponenter i en container så kan man ha hjälp av s.k. layout-managers. En layout-manager placerar komponenterna efter bestämda regler.

Med metoden **setLayout** anger man den layout-manager som ska användas.

```
container.setLayout( new BorderLayout() );
```

I en klass som ärver *JPanel* gör man anropet

```
setLayout( new BorderLayout() );  // this.setLayout( new BorderLayout() );
```

Containern i *JFrame* har *BorderLayout* som layout-manager om du inte anger något annat.

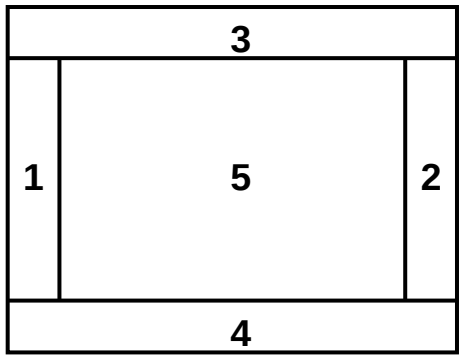
En *JPanel* har *FlowLayout* som layout-manager om du inte anger något annat.

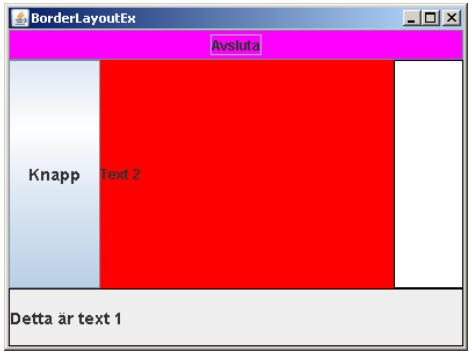
Det finns ett flertal layoutmanagers i Java. Vi kommer huvudsakligen använda oss av *FlowLayout*, *BorderLayout* och *GridLayout*.

BorderLayout

Komponenterna placeras in i en av följande positioner i containern:

1. West
2. East
3. North
4. South
5. Center





Komponenternas storlek ändras om nödvändigt. T.ex. anpassas komponenterna till containerns bredd (3+4) och till containerns höjd(1+2). Även höjden på 3+4 ändras och bredden på 1+2.
Det område som är kvar i mitten är Center och komponenten (5) anpassas till denna storlek.

BorderLayout

Komponenter placeras på containern med add-metoden

add(komponent, position); // Vid arv av JPanel

container.add(komponent, position); // I JPanel-komponent

position = BorderLayout.NORTH, BorderLayout.EAST, BorderLayout.SOUTH, BorderLayout.WEST eller BorderLayout.CENTER.

Om man utelämnar *position*

add(komponent); // Vid arv av JPanel

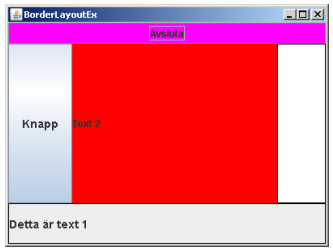
container.add(komponent); // I JPanel-komponent

placeras komponenen i BorderLayout.CENTER.

BorderLayoutPanel.java

BorderLayoutApp.java

JFrame använder *BorderLayout* om inget annat anges.

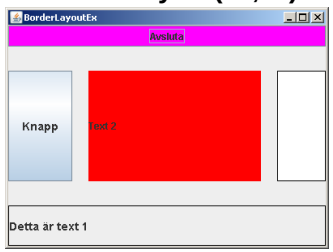


Konstruktorer:

public BorderLayout()
komponenterna placeras intill varandra

public BorderLayout(int hgap, int vgap)
hgap resp *vgap* anger avståndet mellan komponenterna

BorderLayout(20,30)



GridLayout

Containern delas in i ett antal lika höga rader och lika breda kolumner. Alla celler som därmed bildas är lika stora. I varje cell kan man placera en komponent. Komponenten anpassar sig till cellens storlek. Komponenter placeras på containern med metoden

add(komponent) ; // Vid arv av JPanel

container.add(komponent) ; // JPanel-komponent

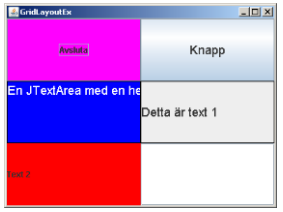
De placeras in radvis, från vänster till höger.
Först fylls raden högst upp osv.

1	2	3	4
5	6	7	8
9	10	11	12

GridLayout med 3 rader och 4 kolumner

Konstruktorer:

GridLayout(3,2)



public GridLayout(int rows, int cols)
rows anger antalet rader och *cols* antalet kolumner

GridLayout(3,2,10,20

public GridLayout(int rows, int cols ,int hgap, int vgap)
rows anger antalet rader och *cols* antalet kolumner
hgap resp *vgap* anger avståndet mellan komponenterna

GridLayoutPanel.java GridLayoutApp.java

BoxLayout

Komponenterna placeras ut verikalt eller horisontellt. I båda fallen i en följd.

När layouten sätts till BoxLayout ska man ge en referens till containern som argument och dessutom riktning i vilken komponenterna ska placeras.

setLayout(this, direction) ; // Vid arv av JPanel

container.add(container, direction) ; // JPanel-komponent

Komponenter placeras in med add-metoden:

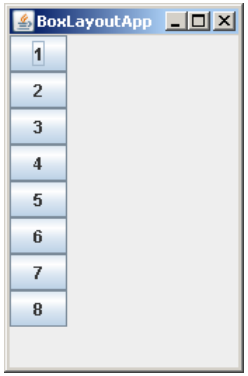
add(komponent) ; // Vid arv av JPanel

container.add(komponent) ; // JPanel-komponent

De placeras i samma ordning som de läggs till.

BoxLayout, - X_AXIS

BoxLayout – Y_AXIS



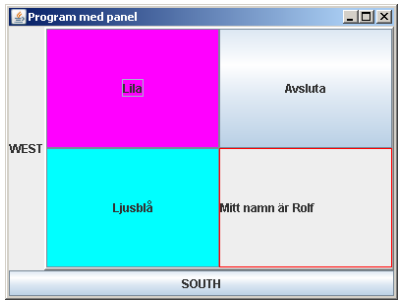
Konstruktorer:

public BoxLayout(Container target, int axis)
target är referens till panelen och *axis* riktningen

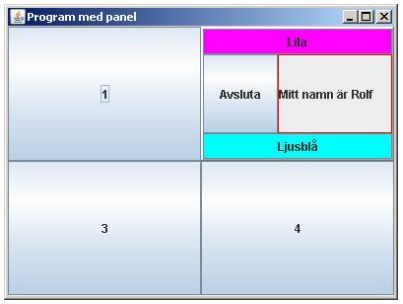
BoxLayoutPanel.java BoxLayoutApp.java

Design med hjälp av JPanel

Med hjälp av JPanel-komponenter kan man använda olika layout-managers i olika delar av fönstret.



GridLayout(2,2) i BorderLayout



BorderLayout i GridLayout(2,2)

Layout-manager i en JPanel är FlowLayout om inget annat anges. Man kan ange layout manager för panelen på två sätt:

Vid konstruktion:

```
private JPanel panel = new JPanel( new GridLayout(5,3) );
```

Med **metoden setLayout**:

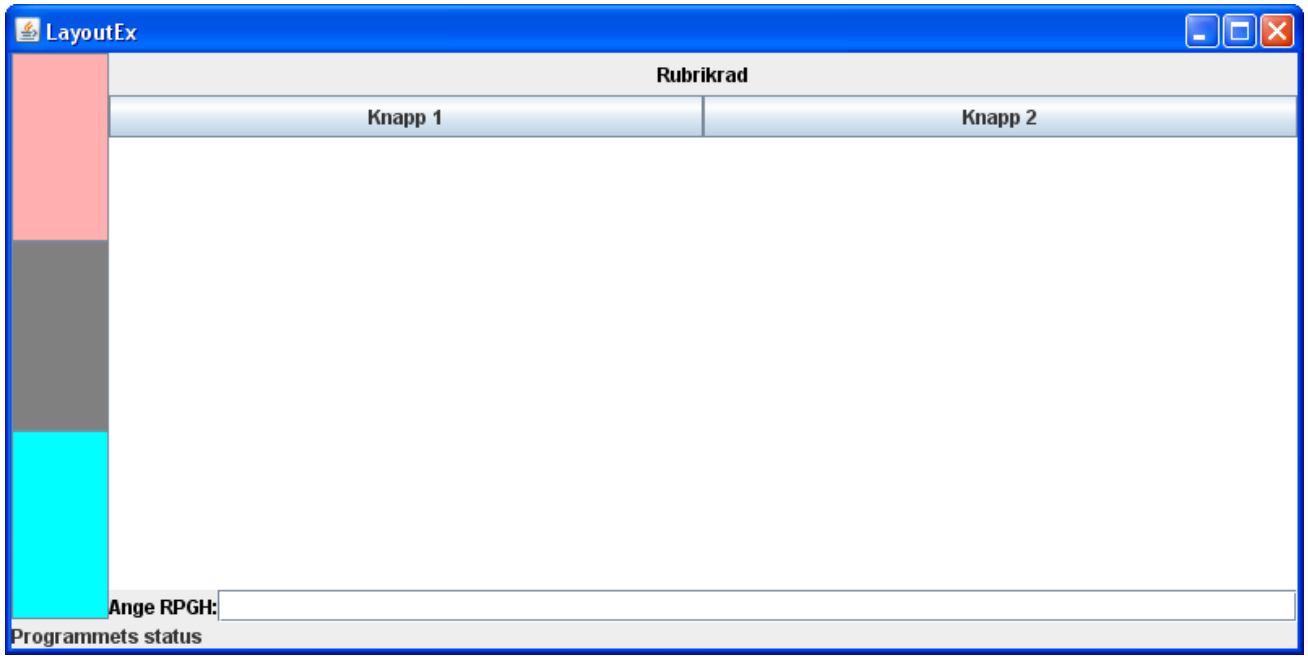
```
panel.setLayout(new GridLayout(5,3));
```

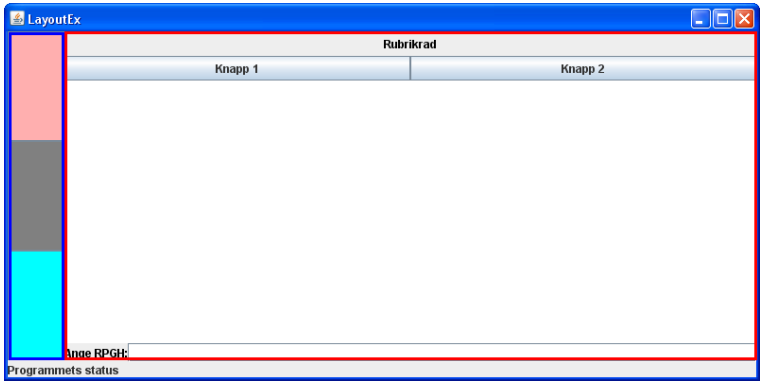
När komponenten ska placeras på en **JPanel** så använder man panelens add-metod:

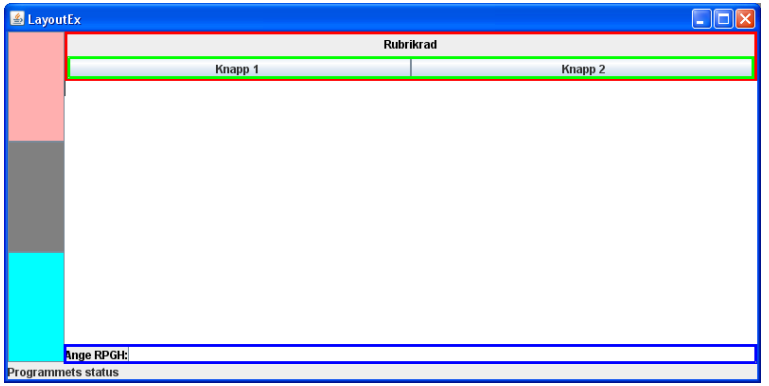
```
panel.add( label );     // komponenten som label refererar till placeras på
                        // panelen panel
```

Panel1	Panel2	Panel3

En layout – hitta paneler!







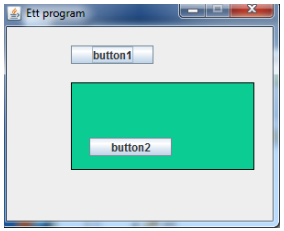
Ingen LayoutManager

Det går bra att ange att man ***inte*** vill använda en LayoutManager. Man gör detta genom anropet

```
container.setLayout( null );
```

När man använder "Null Layout" måste man för varje komponent som placeras på skärmen ange komponentens läge och storlek:

```
private JButton button = new JButton( ”button1” );
:
// positionen (x=70, y=20) i aktuell container. Komponenten blir 90 pixlar bred och
// 20 pixlar hög.
button.setBounds(70, 20, 90, 20);  // (x, y, bredd, höjd)
```



eller

```
button.setLocation(70, 20);  // (x, y)
label.setSize(90, 20);  // (bredd, höjd)
```

Användaren bör inte kunna ändra storleken på ett fönster som har "Null layout". Därför ska metoden setResizable anropas.

```
frame.setResizable( false );  // Fönstrets storlek kan inte ändras
```

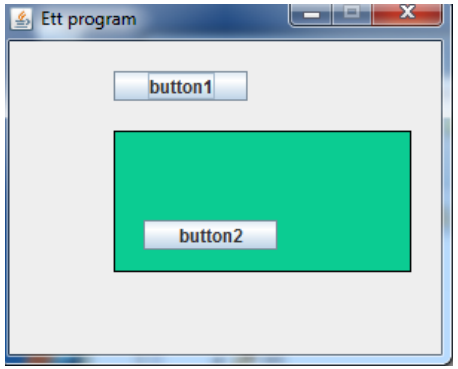
Om flera komponenter täcker varandra så syns den som placerades först i containern.

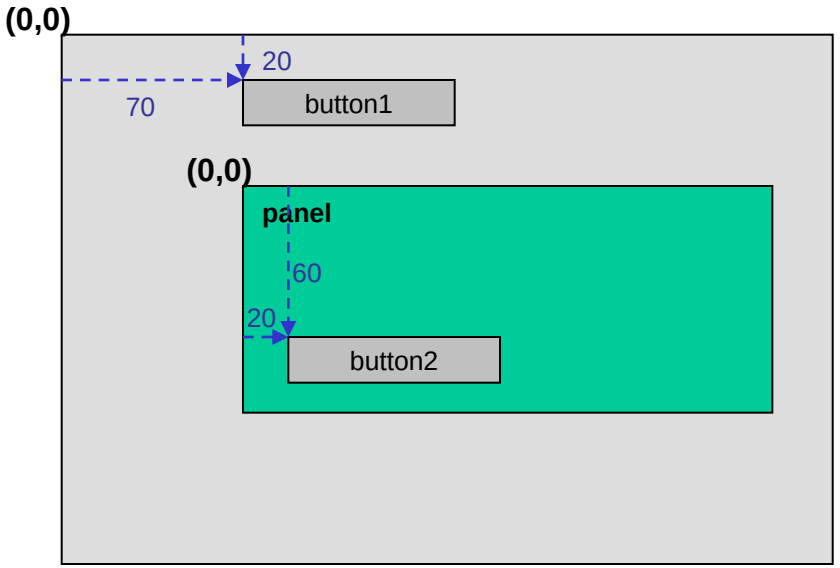
NullLayoutPanel.java NullLayoutApp.java

Null layout – läge och storlek

Den position du anger för komponenten är alltid relativt övre vänstra hörnet på den containern den är placerad i.

```
setLayout( null );
panel.setLayout( null );
:
button1.setBounds( 70, 20, 90, 20 );
button2.setBounds( 20, 60, 90, 20 );
panel.setBounds( 70, 60, 200, 95 );
panel.add( button2 );
add( button1 );
add( panel );
```





button1:
x = 70 (se pil)
y = 20 (se pil)
width = 90
height = 20

button2:
x = 20 (se pil)
y = 60 (se pil)
width = 90
height = 20

panel:
x = 70
y = 60
width = 200
height = 95

MouseListener

En ***MouseListener*** kan man använda bl.a. då man vill lyssna på om
* musknapp trycks ner / släpps då musmarkören är över komponent
* musmarkör rör sin i på komponent / från komponent

En klass som implementerar ***MouseListener*** måste innehålla metoderna
mousePressed, *mouseReleased*, *mouseEntered*, *mouseExited*, *mouseClicked*

MouseListenerPanel demonstrerar hur man kan använda en MouseListener

```
public class MouseListenerPanel extends JPanel {
  :
   private class ML implements MouseListener {
      Random rand = new Random();
      public void mouseEntered(MouseEvent arg0) {
         button1.setBackground( new Color(rand.nextInt(256), rand.nextInt(256), rand.nextInt(256)));
      }

      public void mouseExited(MouseEvent arg0) {
         button2.setBackground( new Color(rand.nextInt(256), rand.nextInt(256), rand.nextInt(256)));
      }

      public void mousePressed(MouseEvent arg0) {
         setBackground( new Color( rand.nextInt(256), rand.nextInt(256), rand.nextInt(256)));
      }

      public void mouseReleased(MouseEvent arg0) {
         panel.setBackground( new Color( rand.nextInt(256), rand.nextInt(256), rand.nextInt(256)));
      }

      public void mouseClicked(MouseEvent arg0) {}
   }
   :
}
```

MouseListenerPanel.java

Bilder och ljud i applikation

Klassen ***ImageIcon*** har ett flertal konstruktorer varav några är väldigt användbara.

I en **applikation** kan man t.ex. hämta en bild så här::

```
ImageIcon im1 = new ImageIcon( ”C:/bilder/rolig.jpg” );
ImageIcon im2 = new ImageIcon( new URL( ”file:C:/bilder/rolig.jpg” ) );
```

Bilden ska ha något av formaten ***gif***, ***jpg*** eller ***png*** (sedan 1.3). Klassen URL används vid nätverkskommunikation, när bilden ska hämtas från en annan dator, men kan även användas för att komma åt filer på hårddisken.

Ljud hämtar man in i en applikation med klassmetoden ***newAudioClip*** i klassen ***JApplet***. Klassen ***AudioClip*** som representerar ljudfilen finns i paketet java.applet.

```
AudioClip sound = Applet.newAudioClip( new URL( ”file:C:/sound/gong.au” ) );
```

Med metoderna ***play*** (spela ljudet en gång), ***stop*** (avsluta uppspelning) och ***loop*** (spela ljudet upprepat) kan du styra användningen av ljudet.

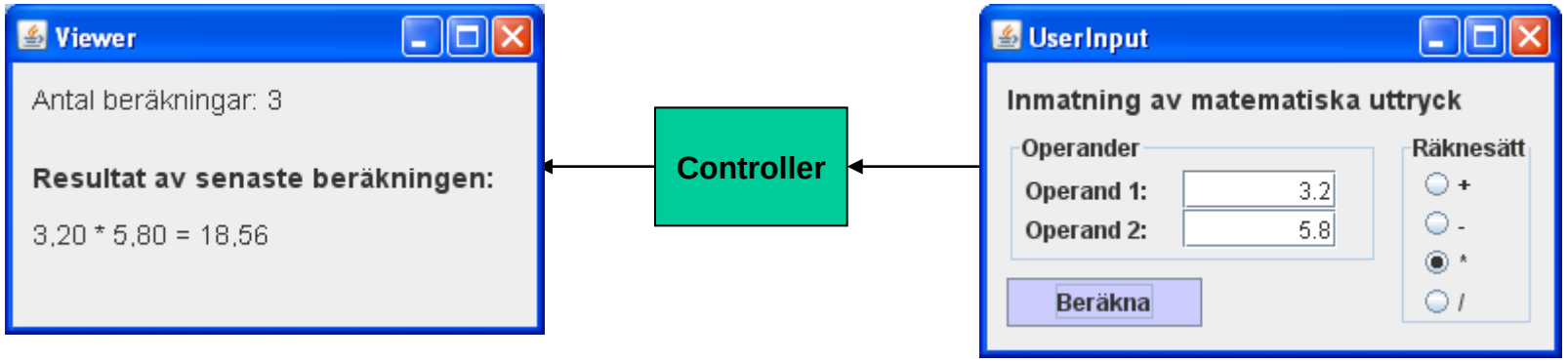
Ljudfilen ska ha något av formaten ***aiff***, ***au***, ***wav***, ***mid*** eller ***rmf***.

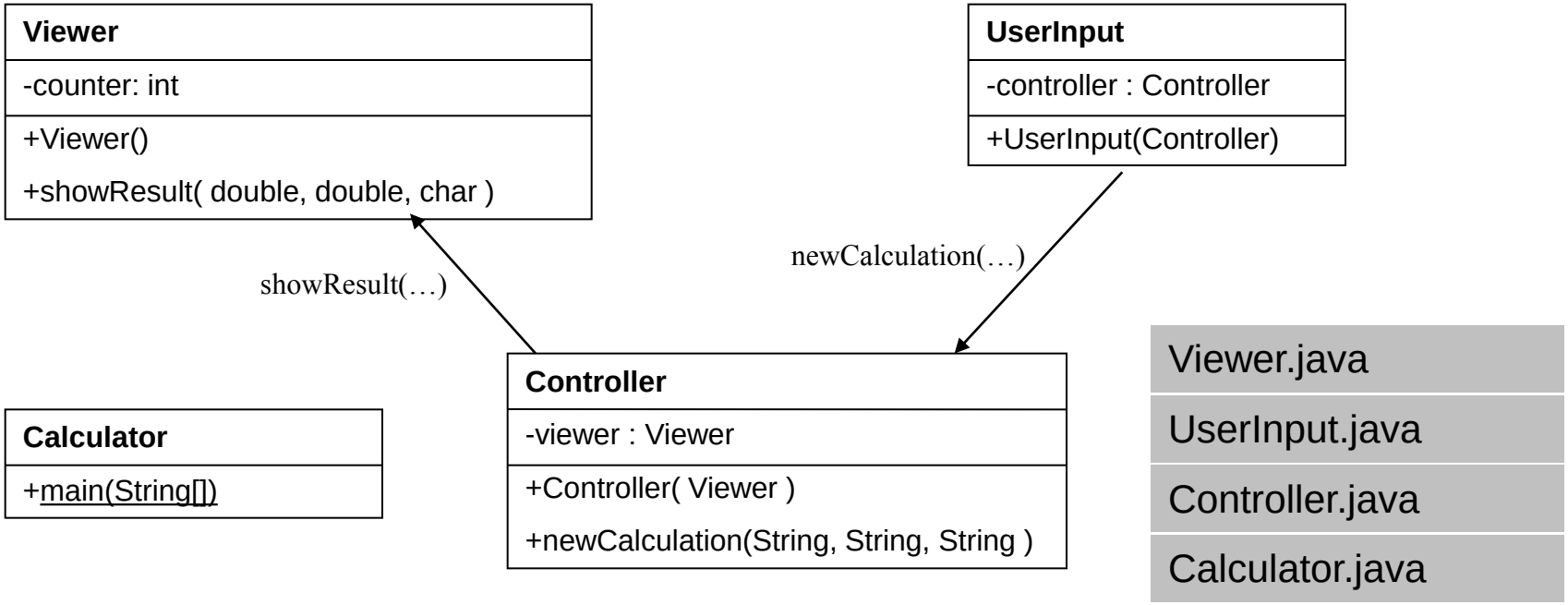
OBS! Packa upp bildfilerna i i katalogen M:\bilder och gong.au i M:\sound innan du testkör SlideShow

SlideShowPanel.java

Ett program som använder flera fönster

Programmet består av två paneler, en Controller-klass och startklass





- Viewer.java
- UserInput.java
- Controller.java
- Calculator.java