**TDA 545:** Objektorienterad programmering

# Föreläsning 13:

# Swing (GUI), händelser, timer

GUI = graphical user interface

Magnus Myréen

Chalmers, läsperiod 1, 2015-2016

# Idag

Idag: grafiska gränssnitt; läs kap 17

- swing
- att hantera händelser
- timers

Nästa gång fortsätter med liknande saker: grafik (också kap 17)

Idag kommer det *mycket kod* i presentationen.

Idén är att varje sida har all kod som behövs för att köra det exemplet.

*Kopiera koden, ändra, prova själv!*

# Begrepp

***Betyder samma sak:***

grafiskt användargränssnitt

=

grafiskt gränssnitt

=

graphical user-interface

=

GUI

=

användargränssnitt

=

user-interface

=

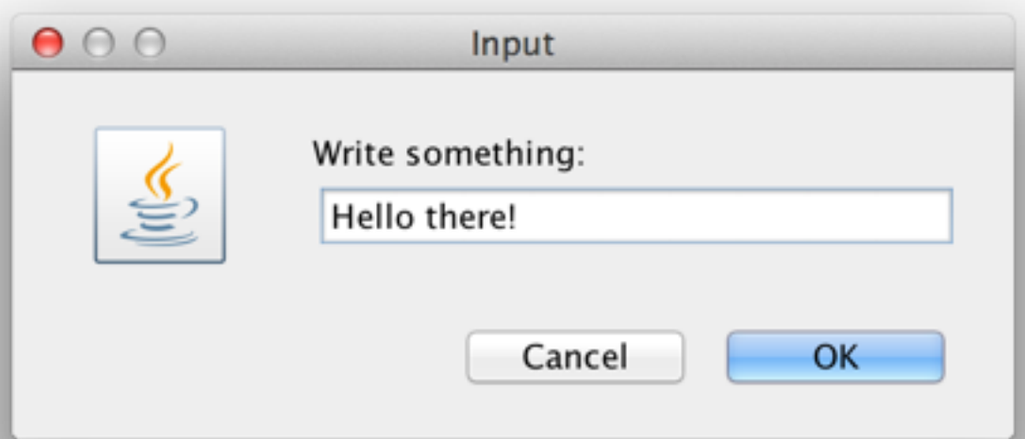
UI

# En `InputDialog`

Vi har ju sätt lite grafiska gränssnitt (dvs GUI) redan, t.ex.

```
import javax.swing.*;

public class Demo0 {
    public static void main(String[] args) {
        String ans = JOptionPane.showInputDialog("Write something:");
        System.out.println("You wrote: " + ans);
    }
}
```

får fönstret nedan att hoppa fram.




*Utskrift:*

```
$ java Demo0
You wrote: Hello there!
```

Nästan allt annat måste man implementera från enklare objekt själv…

# Att göra GUI själv

*Man behöver:*

**Komponenter (widgets)**

Själva fönstren och de grafiska element som man placerar ut i fönstren

**Layouthantering**

Placerar komponenter i förhållande till varandra i fönstren

**Händelsehantering**

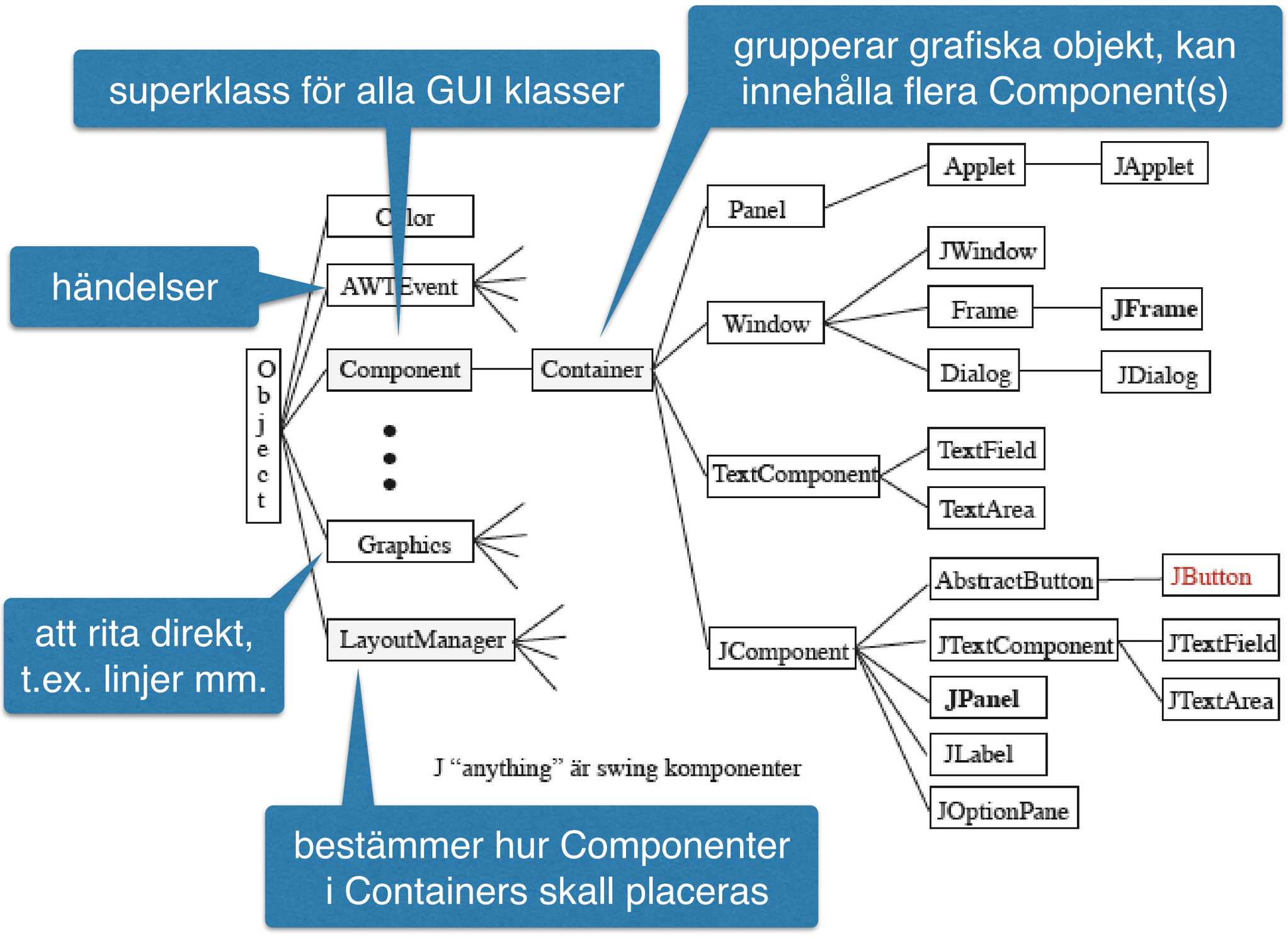
Hjälpmedel för "signaler" mellan de olika programdelarna

**Grafik**

Hjälpmedel för att rita tex linjer, rektanglar och andra mer komplexa objekt

*Obs.* Flera saker finns på två ställen `java.awt.*` och `javax.swing.*`.
Använd `javax.swing.*` det är nyare och bättre.

# Släktträdet för Javas grafiska objekt



# Komponenter

Ett användargränssnitt består alltså av komponenter (widgets):

Det finns huvudsakligen två typer

- **containers** (behållare) som innehåller andra komponenter och består i sin tur av
  - "top-level containers" dvs det vi vanligen ser som ett fönster på skärmen (Jframe, JApplet) och
  - "intermediate containers" som används för att strukturera och positionera andra komponenter (tex JPanel)
- **grundläggande komponenter** som presenterar eller hämtar information som JButton, JtextField, Jlabel, JSlider  mm

# Det blir mycket arv!

java.lang.Object
java.awt.Component
java.awt.Container
javax.swing.JComponent
javax.swing.AbstractButton
javax.swing.JButton

java.lang.Object
java.awt.Component
java.awt.Container
java.awt.Window
java.awt.Frame
javax.swing.JFrame

JButton klassen ärver alla sina föräldrars metoder! och det är inte alltid som metoderna i "sista" klassen, dvs JButton här, är de mest intressanta.

*Vi börjar med:*

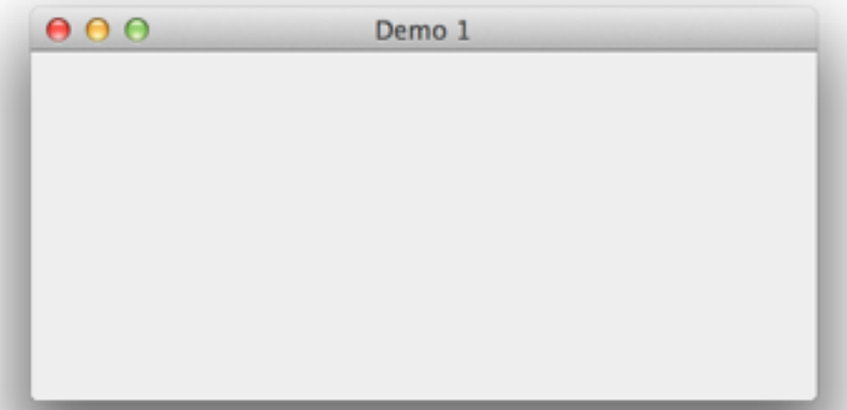
**Att skapa ett fönster (frame)** och att sätta saker på det.

# `JFrame`

## en “top-level” komponent

**JFrame** är ett fönster som inte är placerat i ett annat fönster.

Alla andra komponenter ligger i en JFrame (eller i en JApplet).

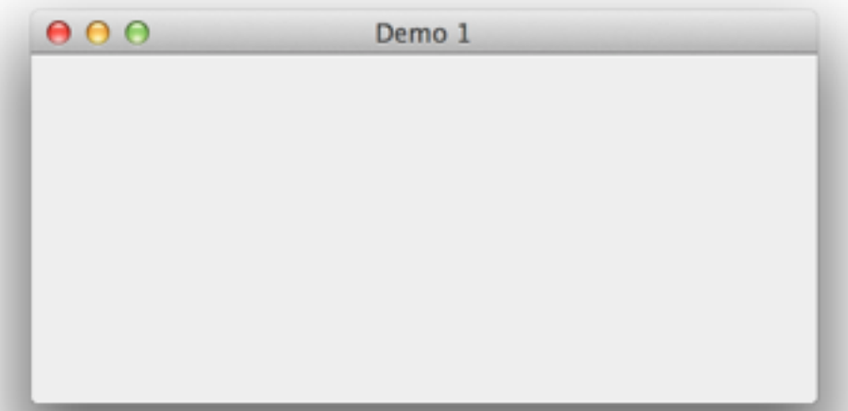



Man lägger alla komponenter i huvudytan (kallas ContentPane)

# Ett tomt JFrame fönster

```
import java.awt.*;
import javax.swing.*;

public class Demo1 {
    public static void main(String[] args) {
        JFrame f = new JFrame("Demo 1");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(400,200);
        f.setLocation(50,50);
        f.setVisible(true);
    }
}
```

skapar en ny JFrame



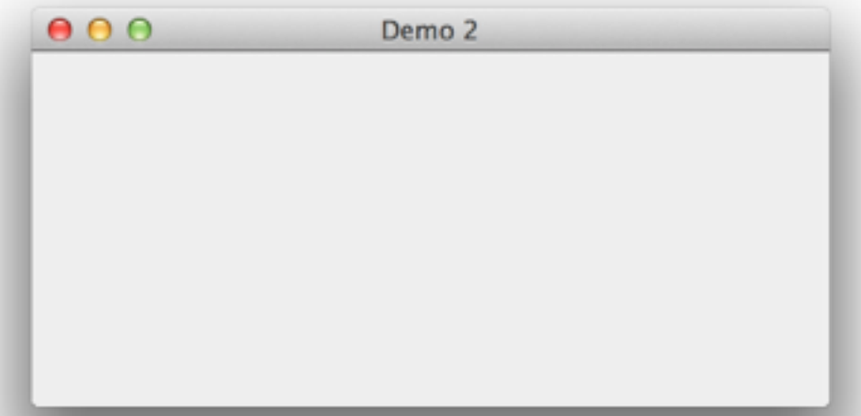

sätter storlek och position

visar den

# … på två olika sätt

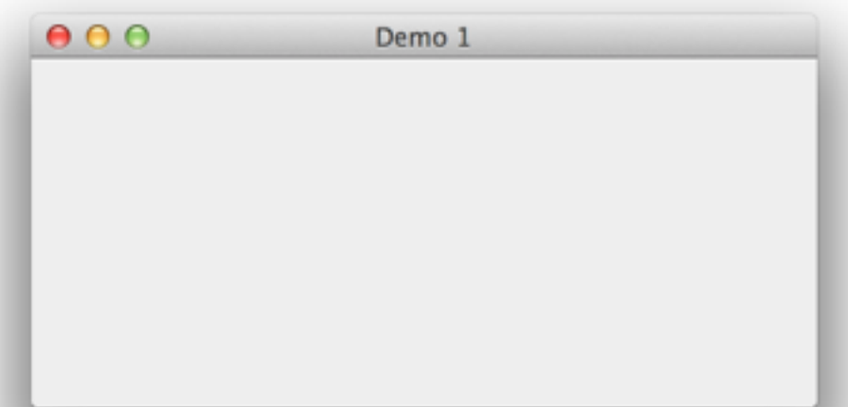
```
import java.awt.*;
import javax.swing.*;

public class Demo2 extends JFrame {
    public Demo2() {
        setTitle("Demo 2");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400,200);
        setLocation(50,50);
        setVisible(true);
    }
    public static void main(String[] args) {
        Demo2 f = new Demo2();
    }
}
```

ärver JFrame!




behöver inte “`f.`” t.ex. `f.setSize`

```
import java.awt.*;
import javax.swing.*;

public class Demo1 {
    public static void main(String[] args) {
        JFrame f = new JFrame("Demo 1");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(400,200);
        f.setLocation(50,50);
        f.setVisible(true);
    }
}
```



# JPanel

## komponent(er) för struktur

**JPanel:** en "osynlig" container som innehåller andra GUI komponenter, speciellt kan den innehålla andra **JPanel**s.

Man placerar en **JPanel** i en **JFrame**.


**JPanels** används framförallt för att gruppera och strukturera de fönster man konstruerar.

# JPanel exempel

```
import java.awt.*;
import javax.swing.*;

public class Demo3 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 3");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(400,200);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new BorderLayout());
        // sätt panelen in i f
        f.add(myPanel);
        // visa ramen
        f.setVisible(true);
    }
}
```

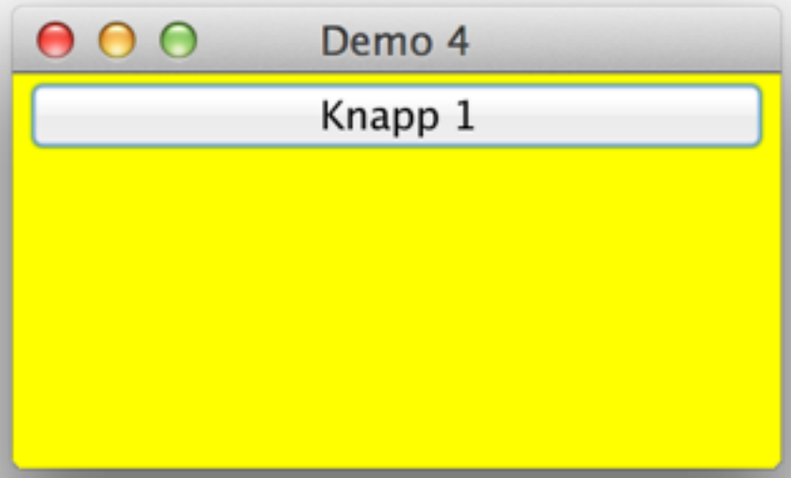

skapar panelen

gör den gul

ger den en layout

säger att panelen ska ligga i JFrame fönstret

# En knapp!




```
import java.awt.*;
import javax.swing.*;

public class Demo4 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 4");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,150);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new BorderLayout());
        // sätt en knapp in i panelen
        JButton button1 = new JButton("Knapp 1");
        myPanel.add(button1,BorderLayout.NORTH);
        // sätt panelen in i f
        f.add(myPanel);
        // visa ramen
        f.setVisible(true);
    }
}
```

skapar knappen

sätter den i övre- (dvs norra) delen av panelen.

# LayoutManager

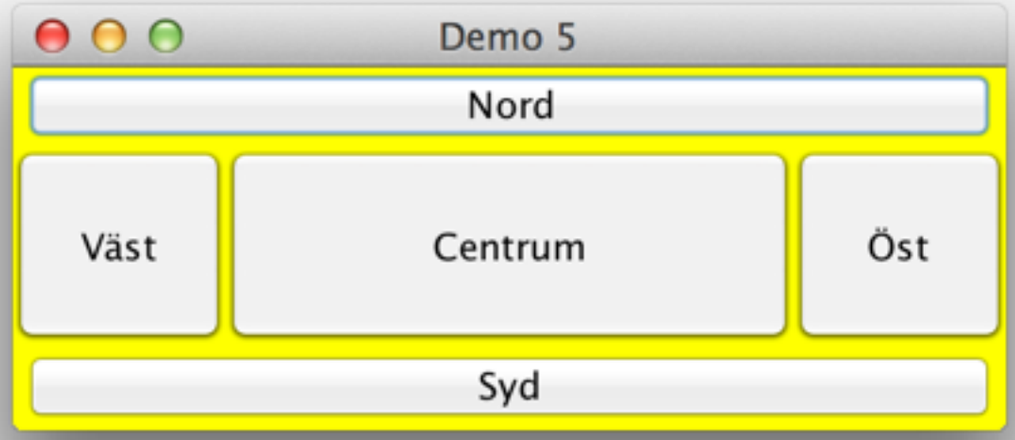
t.ex. BorderLayout

En layoutmanager hjälper oss att placera ut komponenterna i fönstret (JPanel).

Varje container har en egen layoutmanager.

```java
import java.awt.*;
import javax.swing.*;

public class Demo5 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 5");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(350,150);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new BorderLayout());
        // sätt knappar in i panelen
        JButton buttonN = new JButton("Nord");
        JButton buttonS = new JButton("Syd");
        JButton buttonE = new JButton("Öst");
        JButton buttonW = new JButton("Väst");
        JButton buttonC = new JButton("Centrum");
        myPanel.add(buttonN,BorderLayout.NORTH);
        myPanel.add(buttonS,BorderLayout.SOUTH);
        myPanel.add(buttonE,BorderLayout.EAST);
        myPanel.add(buttonW,BorderLayout.WEST);
        myPanel.add(buttonC,BorderLayout.CENTER);
        // sätt panelen in i f
        f.add(myPanel);
        // visa ramen
        f.setVisible(true);
    }
}
```

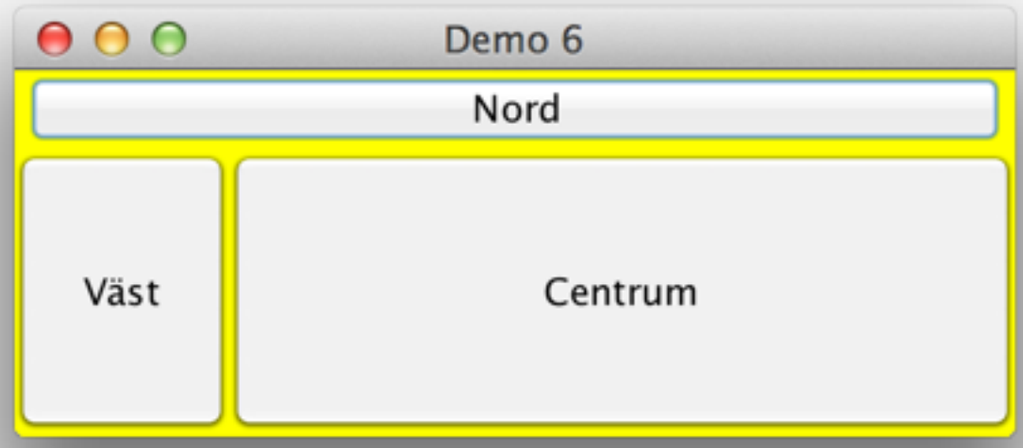
**BorderLayout**
har fem platser för objekt: norr, söder, öst, väst och centrum




Vi sätter en knapp in i varje plats.

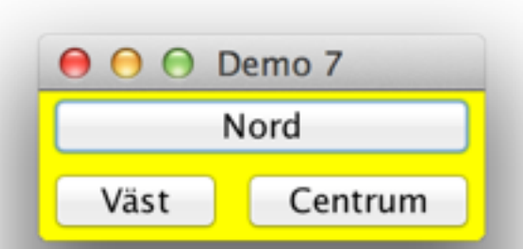
```java
import java.awt.*;
import javax.swing.*;

public class Demo6 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 6");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(350,150);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new BorderLayout());
        // sätt knappar in i panelen
        JButton buttonN = new JButton("Nord");


        JButton buttonW = new JButton("Väst");
        JButton buttonC = new JButton("Centrum");
        myPanel.add(buttonN,BorderLayout.NORTH);


        myPanel.add(buttonW,BorderLayout.WEST);
        myPanel.add(buttonC,BorderLayout.CENTER);
        // sätt panelen in i f
        f.add(myPanel);
        // visa ramen
        f.setVisible(true);
    }
}
```

**BorderLayout**
har fem platser för objekt: norr, söder, öst, väst och centrum




Man behöver inte sätt i varje plats.

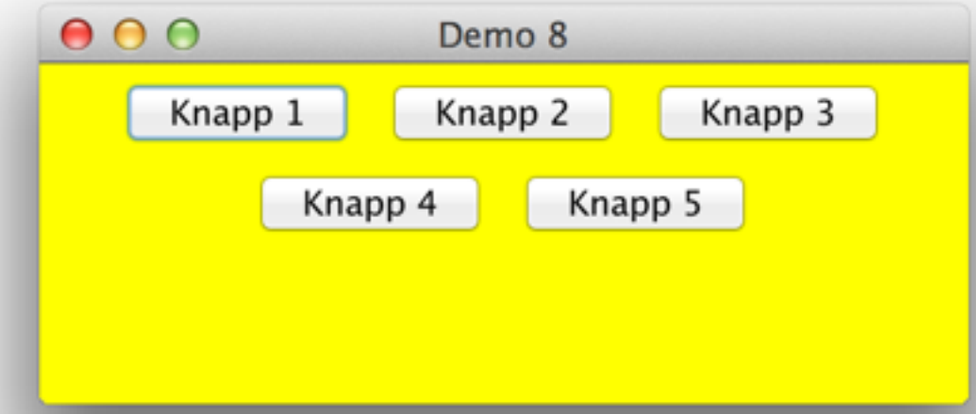
```java
import java.awt.*;
import javax.swing.*;

public class Demo7 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 7");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(350,150);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new BorderLayout());
        // sätt knappar in i panelen
        JButton buttonN = new JButton("Nord");
        JButton buttonW = new JButton("Väst");
        JButton buttonC = new JButton("Centrum");
        myPanel.add(buttonN,BorderLayout.NORTH);
        myPanel.add(buttonW,BorderLayout.WEST);
        myPanel.add(buttonC,BorderLayout.CENTER);
        // sätt panelen in i f
        f.add(myPanel);
        f.pack();
        // visa ramen
        f.setVisible(true);
    }
}
```




Ett anrop av “pack” innan visningen brukar ge lite *bättre storlek på komponenter*.

```java
import java.awt.*;
import javax.swing.*;

public class Demo8 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 8");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(350,150);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new FlowLayout());
        // sätt knappar in i panelen
        JButton button1 = new JButton("Knapp 1");
        JButton button2 = new JButton("Knapp 2");
        JButton button3 = new JButton("Knapp 3");
        JButton button4 = new JButton("Knapp 4");
        JButton button5 = new JButton("Knapp 5");
        myPanel.add(button1);
        myPanel.add(button2);
        myPanel.add(button3);
        myPanel.add(button4);
        myPanel.add(button5);
        // sätt panelen in i f
        f.add(myPanel);
        // visa ramen
        f.setVisible(true);
    }
}
```

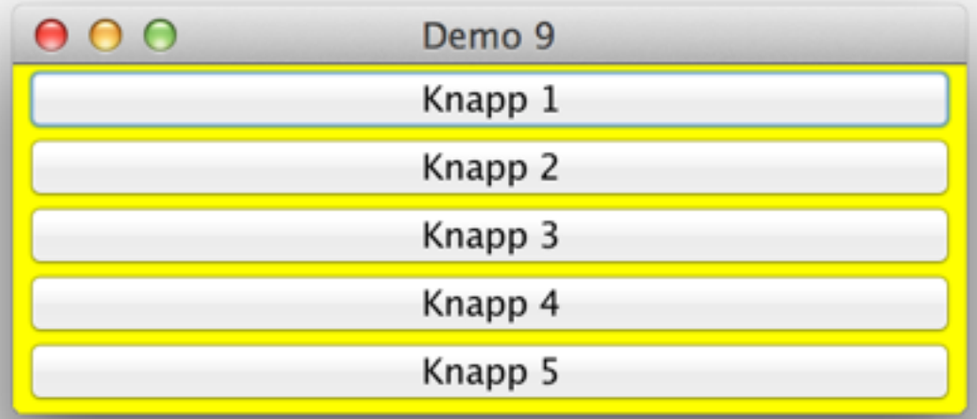
**FlowLayout**
Komponenterna placeras i tur och ordning radvis.



```java
import java.awt.*;
import javax.swing.*;

public class Demo9 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 9");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(350,150);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new GridLayout(5,1));
        // sätt knappar in i panelen
        JButton button1 = new JButton("Knapp 1");
        JButton button2 = new JButton("Knapp 2");
        JButton button3 = new JButton("Knapp 3");
        JButton button4 = new JButton("Knapp 4");
        JButton button5 = new JButton("Knapp 5");
        myPanel.add(button1);
        myPanel.add(button2);
        myPanel.add(button3);
        myPanel.add(button4);
        myPanel.add(button5);
        // sätt panelen in i f
        f.add(myPanel);
        // visa ramen
        f.setVisible(true);
    }
}
```

**GridLayout**

Komponenterna placeras i rader och kolumner.



```
import java.awt.*;
import javax.swing.*;

public class Demo10 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 10");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(350,150);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new GridLayout(5,1));
        // sätt knappar in i panelen
        JButton button1 = new JButton("Knapp 1");
        JButton button2 = new JButton("Knapp 2");
        JButton button3 = new JButton("Knapp 3");
        JButton button4 = new JButton("Knapp 4");
        JButton button5 = new JButton("Knapp 5");
        myPanel.add(button1);
        myPanel.add(button2);
        myPanel.add(button3);
        myPanel.add(button4);
        myPanel.add(button5);
        // sätt panelen in i f
        f.add(myPanel); f.pack();
        // visa ramen
        f.setVisible(true);
    }
}
```

**GridLayout**
Komponenterna placeras i rader och kolumner.


Ett anrop till “pack”.

```java
import java.awt.*;
import javax.swing.*;

public class Demo11 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 11");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(350,150);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new GridLayout(3,2));
        // sätt knappar in i panelen
        JButton button1 = new JButton("Knapp 1");
        JButton button2 = new JButton("Knapp 2");
        JButton button3 = new JButton("Knapp 3");
        JButton button4 = new JButton("Knapp 4");
        JButton button5 = new JButton("Knapp 5");
        myPanel.add(button1);
        myPanel.add(button2);
        myPanel.add(button3);
        myPanel.add(button4);
        myPanel.add(button5);
        // sätt panelen in i f
        f.add(myPanel); f.pack();
        // visa ramen
        f.setVisible(true);
    }
}
```

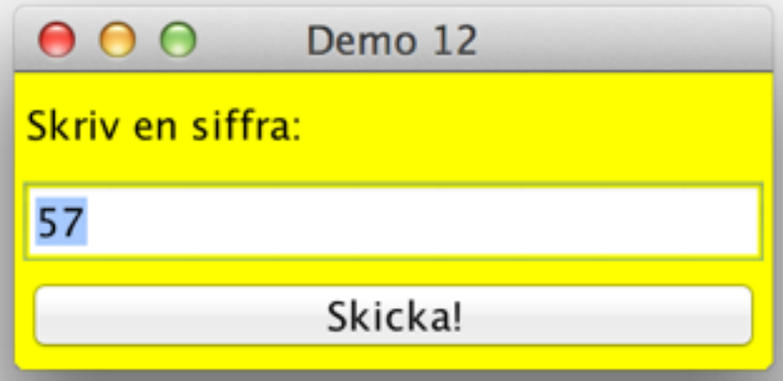
**GridLayout**
Komponenterna placeras i rader och kolumner.

Två kolmuner, tre rader.


Vi sätter in objekten i ordning.

# Olika objekt!

```
import java.awt.*;
import javax.swing.*;

public class Demo12 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 12");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,120);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new GridLayout(3,1));
        // sätt olika saker i panelen
        JLabel label = new JLabel(" Skriv en siffra: ");
        JTextField text = new JTextField("57");
        JButton button = new JButton("Skicka!");
        myPanel.add(label);
        myPanel.add(text);
        myPanel.add(button);
        // sätt panelen in i f
        f.add(myPanel);
        // visa ramen
        f.setVisible(true);
    }
}
```




Lite text

… och en ruta där användaren kan skriva text

# JPanel i JPanel

```
import java.awt.*;
import javax.swing.*;

public class Demo13 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 13");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,120);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new GridLayout(3,1));
        // sätt olika saker i panelen
        JLabel label = new JLabel(" Skriv en siffra: ");
        JTextField text = new JTextField("57");
        myPanel.add(label);
        myPanel.add(text);

        JPanel buttonPanel = new JPanel();
        buttonPanel.setBackground(Color.green);
        buttonPanel.setLayout(new FlowLayout());
        JButton button1 = new JButton("Skicka!");
        JButton button2 = new JButton("Avbryt");
        buttonPanel.add(button1);
        buttonPanel.add(button2);

        myPanel.add(buttonPanel);
        // sätt panelen in i f
        f.add(myPanel);
        // visa ramen
        f.setVisible(true);
    }
}
```

Den yttre panelen är gul och har GridLayout


Här skapar vi en JPanel för knapparna.

Panelen har en FlowLayout

Vi sätter knapparna på knapp panelen…

… och knapp panelen på myPanel.

# Andra LayoutManagers

(inte viktigt)

**BoxLayout**:

Lägger ut **komponenter av olika storlek** horisontellt eller vertikalt, en rad eller en kolumn. Hänsyn tas till komponenternas önskade storlek och man har större kontroll.

Lite som en horisontell eller vertikal FlowLayout. En horisontell BoxLayout är tex väldigt lik en FlowLayout

**CardLayout**:

Komponenterna ligger ovanpå varandra ungefär som en kortlek och man kan bläddra mellan dem.

**GridBagLayout**:

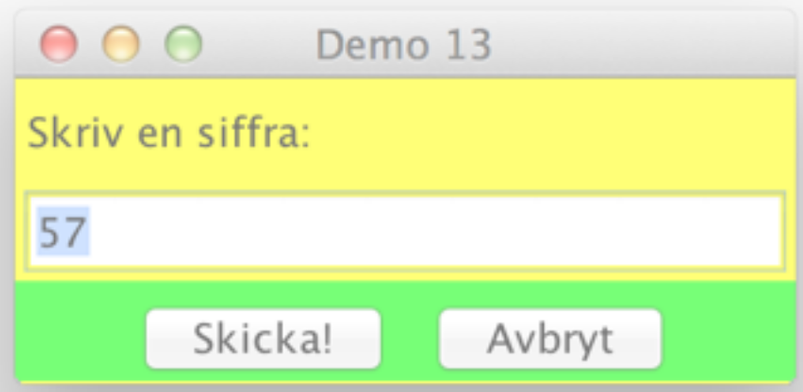
Liknar GridLayout men alla "rutor" behöver inte vara lika stora och en komponent kan spänna över flera rutor.

*Nästa sak:*

**Att hantera händelser,** t.ex. när användaren trycker på knappar.

# Varför fungerar det?

```
import java.awt.*;
import javax.swing.*;

public class Demo13 {
    public static void main(String[] args) {
        // skapar en 'frame' (dvs ett fönster)
        JFrame f = new JFrame("Demo 13");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,120);
        f.setLocation(50,50);
        // skapa en panel
        JPanel myPanel = new JPanel();
        myPanel.setBackground(Color.yellow);
        myPanel.setLayout(new GridLayout(3,1));
        // sätt olika saker i panelen
        JLabel label = new JLabel(" Skriv en siffra: ");
        JTextField text = new JTextField("57");
        myPanel.add(label);
        myPanel.add(text);

        JPanel buttonPanel = new JPanel();
        buttonPanel.setBackground(Color.green);
        buttonPanel.setLayout(new FlowLayout());
        JButton button1 = new JButton("Skicka!");
        JButton button2 = new JButton("Avbryt");
        buttonPanel.add(button1);
        buttonPanel.add(button2);

        myPanel.add(buttonPanel);
        // sätt panelen in i f
        f.add(myPanel);
        // visa ramen
        f.setVisible(true);
    }
}
```




***Svar:*** när man visar ett fönster så startar *en separat process* för **händelse hantering**.

Visar fönstret

Varför tar inte programmets körning slut här?

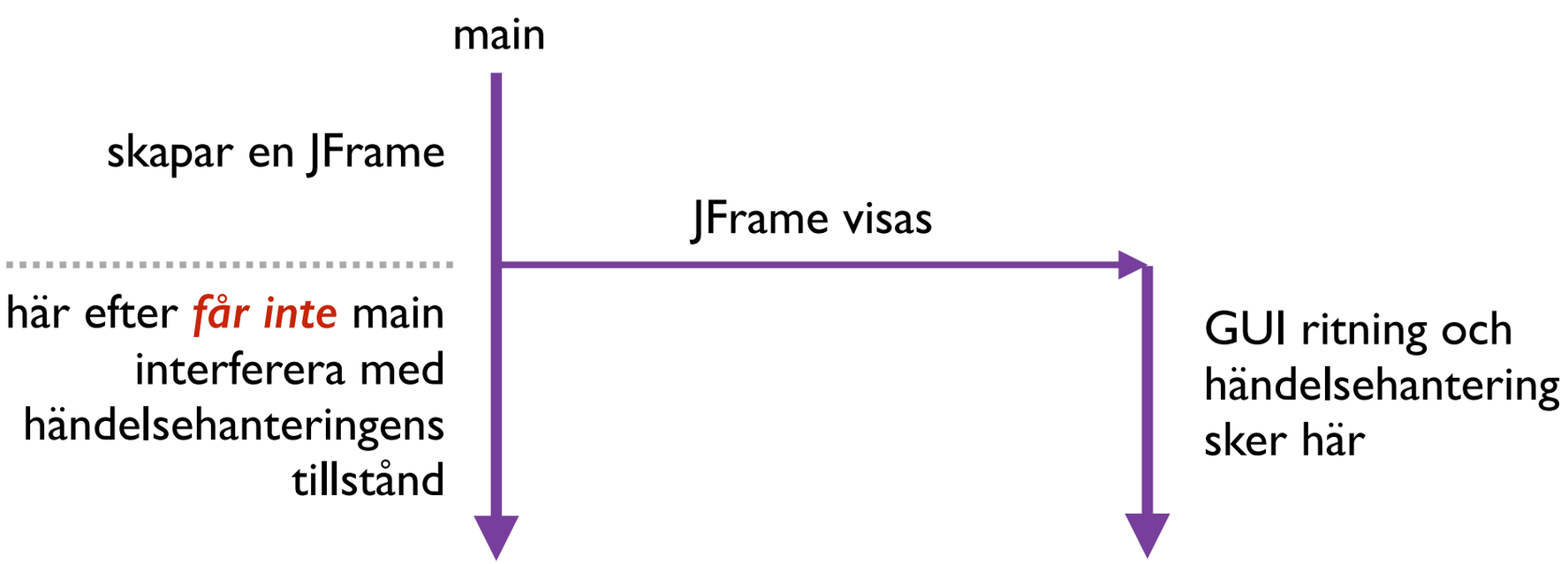
# Programstyrning kontra Händelsestyrning

**Programstyrning:**

`main` styr exekveringen och man "återvänder" alltid dit.

**Händelsestyrning:**

I `main` startar man bara upp programmet. Vanligen bara en rad eller två.
Programmet är sedan normalt passivt.
När man klickar på en knapp genereras en händelse som kör kod.



# Programstyrning kontra Händelsestyrning

**Programstyrning:**

`main` styr exekveringen och man "återvänder" alltid dit.

**Händelsestyrning:**

I `main` startar man bara upp programmet. Vanligen bara en rad eller två.
Programmet är sedan normalt passivt.
När man klickar på en knapp genereras en händelse som kör kod.

***Mera detaljer:***

När man klickar på en knapp genereras en händelse av typen ActionEvent.

Den fångas upp av en lyssnare av typen ActionListener (med en metod actionPerformed) som utför önskade åtgärder och sedan återgår programmet till det passiva läget (dvs den lyssnar igen).

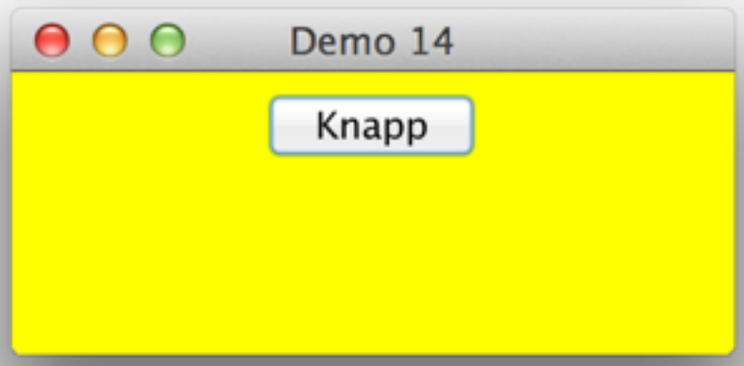
För att detta skall fungera så registrerar man lyssnaren hos komponenter.

# Lägg till en ActionListener

En klass som implementerar ActionListener…

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class DemoListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        System.out.println("Hej!");
    }

}

public class Demo14 {
    public static void main(String[] args) {
        JFrame f = new JFrame("Demo 14");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,120);
        JPanel myPanel = new JPanel();
        myPanel.setLayout(new FlowLayout());
        myPanel.setBackground(Color.yellow);
        JButton button1 = new JButton("Knapp");
        button1.addActionListener(new DemoListener());
        myPanel.add(button1);
        f.add(myPanel);
        f.setVisible(true);
    }
}
```

… dvs har en actionPerformed metod.




*Utskrift:*

```
Hej!
Hej!
Hej!
Hej!
```

om man trycker fyra gånger på knappen.

En ActionListener skapas…

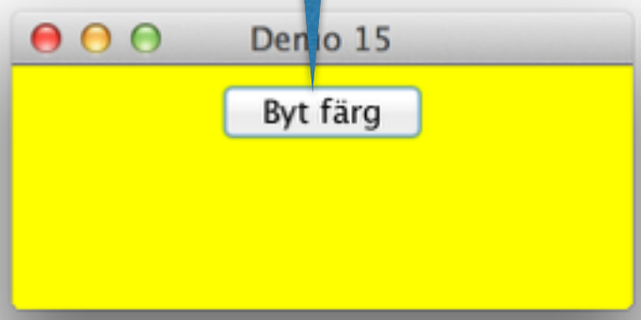
… och registreras med objektet som de ska lyssna på.

# att ändra tillstånd…

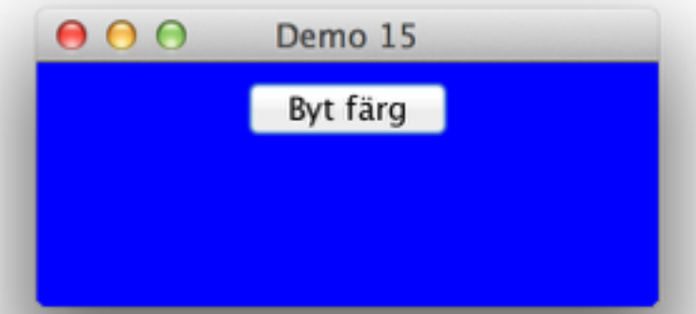
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class DemoListener implements ActionListener {
    JPanel myPanel;
    boolean isBlue = false;
    public DemoListener(JPanel myPanel) {
        this.myPanel = myPanel;
    }
    public void actionPerformed(ActionEvent e) {
        if (isBlue) {
            myPanel.setBackground(Color.yellow);
            isBlue = false;
        } else {
            myPanel.setBackground(Color.blue);
            isBlue = true;
        }
    }
}

public class Demo15 {
    public static void main(String[] args) {
        JFrame f = new JFrame("Demo 15");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,120);
        JPanel myPanel = new JPanel();
        myPanel.setLayout(new FlowLayout());
        myPanel.setBackground(Color.yellow);
        JButton button1 = new JButton("Byt färg");
        button1.addActionListener(new DemoListener(myPanel));
        myPanel.add(button1);
        f.add(myPanel);
        f.setVisible(true);
    }
}
```

Lyssnaren ändrar på färgen när man trycker på knappen.

Vi ändrar på färgen.




Färgen byts när man trycker på knappen.




Lyssnaren måste veta vilket objekt som skall uppdateras.

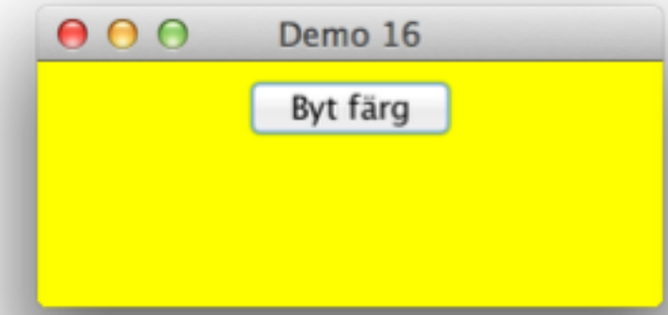
# … samma med arv

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class YellowBluePanel extends JPanel
                      implements ActionListener {
    boolean isBlue = false;
    public YellowBluePanel() {
        this.setBackground(Color.yellow);
    }
    public void actionPerformed(ActionEvent e) {
        if (isBlue) {
            this.setBackground(Color.yellow);
            isBlue = false;
        } else {
            this.setBackground(Color.blue);
            isBlue = true;
        }
    }
}

public class Demo16 {
    public static void main(String[] args) {
        JFrame f = new JFrame("Demo 16");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,120);
        YellowBluePanel myPanel = new YellowBluePanel();
        myPanel.setLayout(new FlowLayout());
        JButton button1 = new JButton("Byt färg");
        button1.addActionListener(myPanel);
        myPanel.add(button1);
        f.add(myPanel);
        f.setVisible(true);
    }
}
```

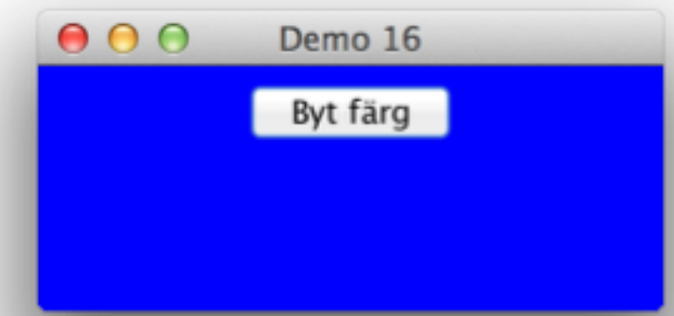
Kanske det är bättre att panelen själv lyssnar och regerar på knappar…

Istället för JPanel använder vi en YellowBluePanel

Nu är det panelen själv som lyssnar på uppdaterings signaler.



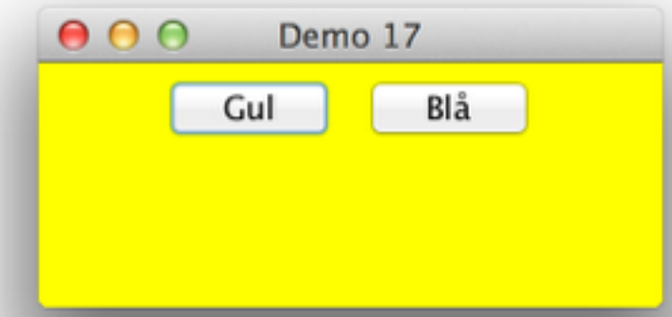

Färgen byts när man trycker på knappen.



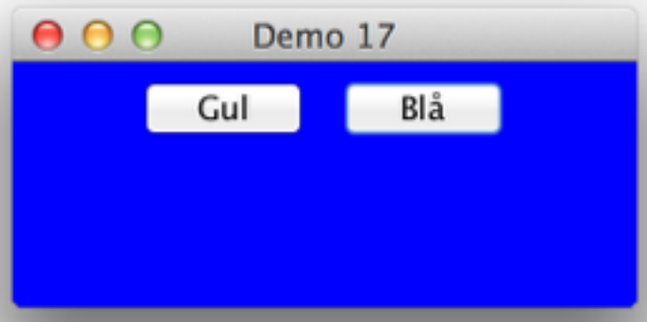
# att välja färg

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class YellowBluePanel extends JPanel
                      implements ActionListener {
    public YellowBluePanel() {
        this.setBackground(Color.yellow);
    }
    public void actionPerformed(ActionEvent e) {
        String str = e.getActionCommand();
        if (str.equals("yellow")) {
            this.setBackground(Color.yellow);
        } else if (str.equals("blue")) {
            this.setBackground(Color.blue);
        }
    }
}

public class Demo17 {
    public static void main(String[] args) {
        JFrame f = new JFrame("Demo 17");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,120);
        YellowBluePanel myPanel = new YellowBluePanel();
        myPanel.setLayout(new FlowLayout());
        JButton button1 = new JButton("Gul");
        button1.addActionListener(myPanel);
        button1.setActionCommand("yellow");
        myPanel.add(button1);
        JButton button2 = new JButton("Blå");
        button2.addActionListener(myPanel);
        button2.setActionCommand("blue");
        myPanel.add(button2);
        f.add(myPanel);
        f.setVisible(true);
    }
}
```

Lyssnaren läser meddelandet och uppdaterar färgen enligt meddelandet.




Färgen byts när man trycker på knapparna.



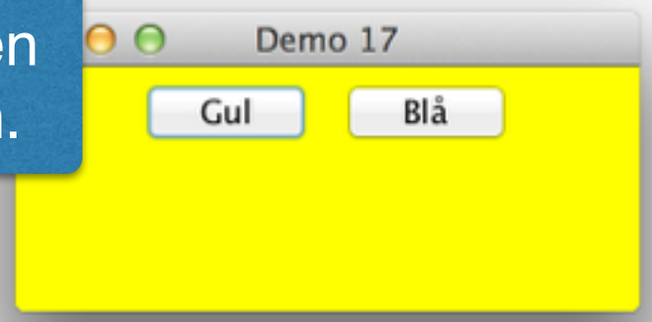

Om man trycker på button1 skickar den nu ett meddelande “yellow”…

# … samma på annat sätt

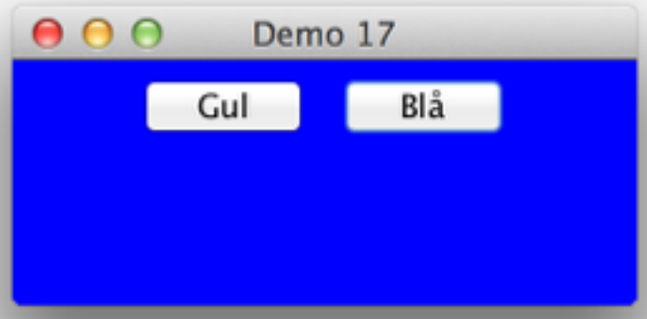
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class YellowBluePanel extends JPanel
                      implements ActionListener {
    public YellowBluePanel() {
        this.setBackground(Color.yellow);
    }
    public void actionPerformed(ActionEvent e) {
        Object obj = e.getSource();
        if (obj instanceof JButton) {
            JButton b = (JButton) obj;
            String str = b.getText();
            if (str.equals("Gul")) {
                this.setBackground(Color.yellow);
            } else if (str.equals("Blå")) {
                this.setBackground(Color.blue);
            }
        }
    }
}

public class Demo18 {
    public static void main(String[] args) {
        JFrame f = new JFrame("Demo 17");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,120);
        YellowBluePanel myPanel = new YellowBluePanel();
        myPanel.setLayout(new FlowLayout());
        JButton button1 = new JButton("Gul");
        button1.addActionListener(myPanel);
        myPanel.add(button1);
        JButton button2 = new JButton("Blå");
        button2.addActionListener(myPanel);
        myPanel.add(button2);
        f.add(myPanel);
        f.setVisible(true);
    }
}
```

Istället kan vi kolla vilken knapp som användes…

… och läsa texten på den knappen.

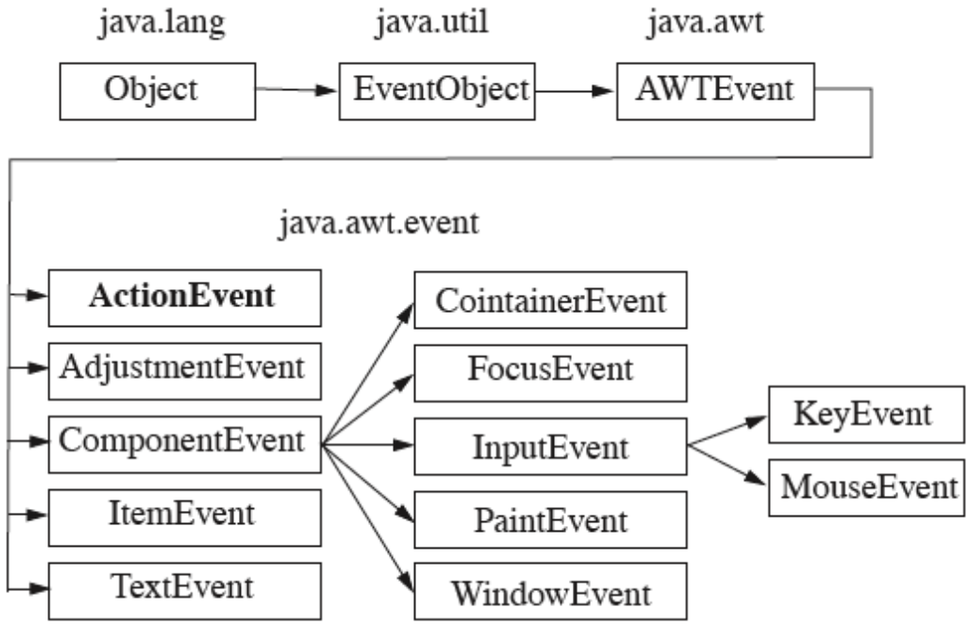



Färgen byts när man trycker på knapparna.




***Obs.*** Det finns flera *rätta* sätt att göra samma sak!

# Lyssnare och händelser




- Samma objekt kan vara källobjekt och lyssnarobjekt (som ovan).
- Ett källobjekt kan ha flera lyssnare.
- Lyssnare måste implementera händelsens standardinterface.
- I Java kan man lyssna på nästan vilka händelser som helst. Det är inte bara när en knapp har blivit nedtryckt, utan också tangentbordstryckningar, när ett fönster ändrat storlek, när musen har flyttats osv.

## Några av de vanligaste händelserna:

**Action** Vanliga händelser, som t ex att någon har tryckt på en knapp ( JButton )

**Mouse** Händelser med musen, som t ex att en musknapp har tryckts ned eller släppts

**MouseMotion** När muspekaren har rört på sig

**Key** Tangentbordshändelser. Man kan dels lyssna på när tangenter trycks ned och släpps upp och dels på när de faktiskt producerar ett tecken

**Window** Händelser för ett fönster, som t ex att det är på väg att stängas eller har ikonifierats

**Component** Förändringar av en komponent, t ex att dess storlek har ändrats (ofta som följd av att hela fönstrets storlek har ändrats)

Action**Event** - Action**Listener**,

Mouse**Event** - Mouse**Listener** ... osv

Ofta är ActionEvent ett resultat av någon annan händelse. Att någon trycker på en knapp är ju egentligen en MouseEvent. I knappens implementering ligger dock att den själv fångar upp mushändelsen och skickar iväg en ActionEvent.

*Nästa sak:*

**Timer** — behöver också en `ActionListener`

# Timer

I **javax.swing.Timer** finns

**Timer(int delay, ActionListener listener)**

Creates a Timer that will notify its listeners every delay milliseconds.

**void setRepeats(boolean flag)**

If flag is false, instructs the Timer to send only one action event to its listeners.

**void start()**

Starts the Timer, causing it to start sending action events to its listeners.

**void stop()**

Stops the Timer, causing it to stop sending action events to its listeners.

(Det finns 3 klasser som heter Timer, hitta den rätta i javax.swing)

# Skapa ett Launch program!

```
import java.awt.event.*;
import javax.swing.Timer;

public class Launch implements ActionListener {

    private int i;

    public Launch(int i) {
        this.i = i;
    }

    public void actionPerformed(ActionEvent e) {
        System.out.println(i + " seconds left...");
        if (i == 0) {
            System.out.println("Launch!");
            System.exit(0);
        } else {
            i = i - 1;
        }
    }

    public static void main(String[] args) {
        int i = Integer.parseInt(args[0]);
        Launch l = new Launch(i);
        Timer t = new Timer(1000,l);
        t.start();
    }

}
```

*Utskrift:*

```
$ java Launch 5
4 seconds left...
3 seconds left...
2 seconds left...
1 seconds left...
0 seconds left...
Launch!
```

Startar händelsehanteringsprocessen.

# Problem 5 är relevant

http://www.cse.chalmers.se/edu/year/2013/course/tda545/courseMtrl/sampleExams/2013.10.24.pdf

# Räknare till tentan (Main.java)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Main implements ActionListener {

    public Main() {
       JFrame f = new JFrame();
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(250,120);
        f.setLocation(50,50);
        JPanel panel = new JPanel(new GridLayout(2,1));
        JLabel label1 = new JLabel("Vill du lära dig Java?", SwingConstants.CENTER);
        panel.add(label1);
        JPanel buttonPanel = new JPanel(new FlowLayout());
        JButton button1 = new JButton("Ja!");
        JButton button2 = new JButton("Kanske lite senare...");
        button1.addActionListener(this);
        button1.setActionCommand("yes");
        button2.addActionListener(this);
        button2.setActionCommand("no");
        buttonPanel.add(button1);
        buttonPanel.add(button2);
        panel.add(buttonPanel);
        f.add(panel);
        f.pack();
        f.setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        String str = e.getActionCommand();
        if (str.equals("yes")) {
            System.exit(0);
        } else if (str.equals("no")) {
            Stress s = new Stress();
        }
    }

    public static void main (String [] args) {
        Main q = new Main();
    }

}
```

# Räknare till tentan (Stress.java)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.time.*;
import java.time.temporal.ChronoUnit;

public class Stress implements ActionListener {

    JLabel timeLabel;

    private int mode = 2;

    public void toggle() {
        mode = (mode + 2) % 3;
    }

    public Stress() {

          FullScreenFrame ff = new FullScreenFrame();
        JPanel panel = new JPanel(new GridLayout(3,1));
        panel.setBackground(Color.BLACK);
        MaxLabel label1 = new MaxLabel(" Men det är ju bara ");
        MaxLabel label2 = new MaxLabel(" " + timeLeftSecs() + " ");
        MaxLabel label3 = new MaxLabel(" sekunder till tentan! ");
        panel.add(label1);
        panel.add(label2);
        panel.add(label3);
        ff.add(panel);

          ff.validate();
        ff.showFullScreen();

        label1.maximiseFont();
        label2.maximiseFont();
        label3.maximiseFont();

        timeLabel = label2;
        Timer t = new Timer(10,this);
        t.start();

        label1.setForeground(Color.WHITE);
        label2.setForeground(Color.YELLOW);
        label3.setForeground(Color.WHITE);

        label1.addMouseListener(new StressExit());
        label2.addMouseListener(new StressMode(this));
        label3.addMouseListener(new StressExit());

    }

    public String timeLeftSecs() {
        LocalDateTime tentan = LocalDateTime.of(2014,10,31,8,30,00);
        LocalDateTime nu = LocalDateTime.now();
        long left = nu.until(tentan,ChronoUnit.MILLIS);
        if (mode == 0) {
            return "" + (left / 1000);
        } else if (mode == 1) {
            return (left / 1000) + "," + ((left / 100) % 10);
        } else {
            if ((left / 10) % 100 < 10) {
                return (left / 1000) + ",0" + ((left / 10) % 100);
            } else {
                return (left / 1000) + "," + ((left / 10) % 100);
            }
        }
    }

    public void actionPerformed(ActionEvent e) {
        timeLabel.setText(" " + timeLeftSecs() + " ");
    }

    public static void main (String [] args) {
        Stress s = new Stress();
    }

}

class StressMode implements MouseListener {

    Stress s;

    public StressMode(Stress t) {
        s = t;
    }

    public void mouseClicked(MouseEvent e) {
        s.toggle();
    }

    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}

}

class StressExit implements MouseListener {

    public void mouseClicked(MouseEvent e) {
        System.exit(0);
    }

    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}

}
```

# Räknare till tentan (FullScreenFrame.java)

```
import java.awt.*;
import javax.swing.*;

public class FullScreenFrame extends JFrame {

    private GraphicsDevice device;

    public FullScreenFrame() {
        GraphicsEnvironment ge =
            GraphicsEnvironment.getLocalGraphicsEnvironment();
        device = ge.getDefaultScreenDevice();
    }

    public void showFullScreen() {
        if (device.isFullScreenSupported()) {
            setUndecorated(true);
            device.setFullScreenWindow(this);
        } else {
            System.err.println("Full screen not supported.");
            System.exit(1);
        }
    }

}
```

# Räknare till tentan (MaxLabel.java)

```
import java.awt.*;
import javax.swing.*;

public class MaxLabel extends JLabel {

    public MaxLabel(String text) {
        super(text,SwingConstants.CENTER);
    }

    public void maximiseFont() {
        Font font = this.getFont();
        String text = this.getText();
        int maxWidth = this.getWidth();
        int maxHeight = this.getHeight();
        int stringWidth = this.getFontMetrics(font).stringWidth(text);
        int size = 3;
        while (true) {
            Font newFont = new Font(font.getName(), Font.PLAIN, size+1);
            int newWidth = this.getFontMetrics(newFont).stringWidth(text);
            if (maxWidth < newWidth) {
                break;
            }
            font = newFont;
            size++;
        }
        while (maxHeight < this.getFontMetrics(font).getHeight()) {
            size--;
            font = new Font(font.getName(), Font.PLAIN, size);
        }
        this.setFont(font);
    }

}
```