Java-concept och Swing

Swing low, sweet chariot


Alice "Babs" Nilson
SÅNGERSKA
Swing it magistern!
DEN NYA ADOLF JAHR-FILMEN
Regi: Schamyl Bauman

Javas Swing-API

- En del av Javas standard-API
- API - application programming interface
- Ett klassbibiliotek som följer med Java
- Är designat med många OO-concept och -principer
- Bygger på det äldre AWT (en del klasser ligger fortfarande i java.awt-paketet.

En massa paket

De mest centrala paketen är `javax.swing` och `java.awt`

Vad var paket?

En katalog med klasser/interface som hör ihop

What makes it Swing?

I AWT/Swing finns containers (behållare) och components (komponenter).

Containers är (genom arv) också komponenter

Olika LayoutManagers hanterar placering av komponenter.

AWT är grunden

I java.awt:

Component <- Container <- Window <- Frame

I Swing:

java.awt.Container <- javax.swing.JComponent

java.awt.Frame <- javax.swing.JFrame

<- javax.swing.JDialog

Swing-komponenter är containers

En JFrame är ett fönster. I en JFrame kan man lägga komponenter, t ex en JMenuBar. I en JMenuBar kan man lägga en (eller flera) JMenu och i en JMenu kan man lägga flera JMenuItem.

JFrame - Huvudfönstret

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

public class SimpleFrame {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setPreferredSize(new Dimension(480,200));
    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args){
    createAndShowGUI();
  }
}
```

Konstruktor - String med titel

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

public class SimpleFrame {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setPreferredSize(new Dimension(480,200));
    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args){
    createAndShowGUI();
  }
}
```

Stäng fönstret med x-knappen

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

public class SimpleFrame {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setPreferredSize(new Dimension(480,200));
    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args){
    createAndShowGUI();
  }
}
```

Fönstret har en storlek

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

public class SimpleFrame {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setPreferredSize(new Dimension(480,200));
    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args){
    createAndShowGUI();
  }
}
```

Packa ihop och visa fönstret

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

public class SimpleFrame {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setPreferredSize(new Dimension(480,200));
    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args){
    createAndShowGUI();
  }
}
```

Vi vill ha lite komponenter i fönstret

```
JFrame frame = new JFrame("Frame with menu");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setPreferredSize(new Dimension(480,200));

JMenuBar menuBar=new JMenuBar();
frame.add(menuBar, BorderLayout.NORTH);
JMenu menu = new JMenu("File");
menuBar.add(menu);
JMenuItem quit = new JMenuItem("Quit");
menu.add(quit);
quit.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent e){
    System.exit(0);
  }});
//Display the window.
frame.pack();
frame.setVisible(true);
```

JMenuBar - Lägg den högst upp

```
JFrame frame = new JFrame("Frame with menu");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setPreferredSize(new Dimension(480,200));

JMenuBar menuBar=new JMenuBar();
frame.add(menuBar, BorderLayout.NORTH);
JMenu menu = new JMenu("File");
menuBar.add(menu);
JMenuItem quit = new JMenuItem("Quit");
menu.add(quit);
quit.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent e){
    System.exit(0);
  }});
//Display the window.
frame.pack();
frame.setVisible(true);
```

JMenu

```
JFrame frame = new JFrame("Frame with menu");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setPreferredSize(new Dimension(480,200));

JMenuBar menuBar=new JMenuBar();
frame.add(menuBar, BorderLayout.NORTH);
JMenu menu = new JMenu("File");
menuBar.add(menu);
JMenuItem quit = new JMenuItem("Quit");
menu.add(quit);
quit.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent e){
    System.exit(0);
  }});
//Display the window.
frame.pack();
frame.setVisible(true);
```

JMenuItem

```
JFrame frame = new JFrame("Frame with menu");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setPreferredSize(new Dimension(480,200));

JMenuBar menuBar=new JMenuBar();
frame.add(menuBar, BorderLayout.NORTH);
JMenu menu = new JMenu("File");
menuBar.add(menu);
JMenuItem quit = new JMenuItem("Quit");
menu.add(quit);
quit.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent e){
    System.exit(0);
  }});
//Display the window.
frame.pack();
frame.setVisible(true);
```

Sammanfattning

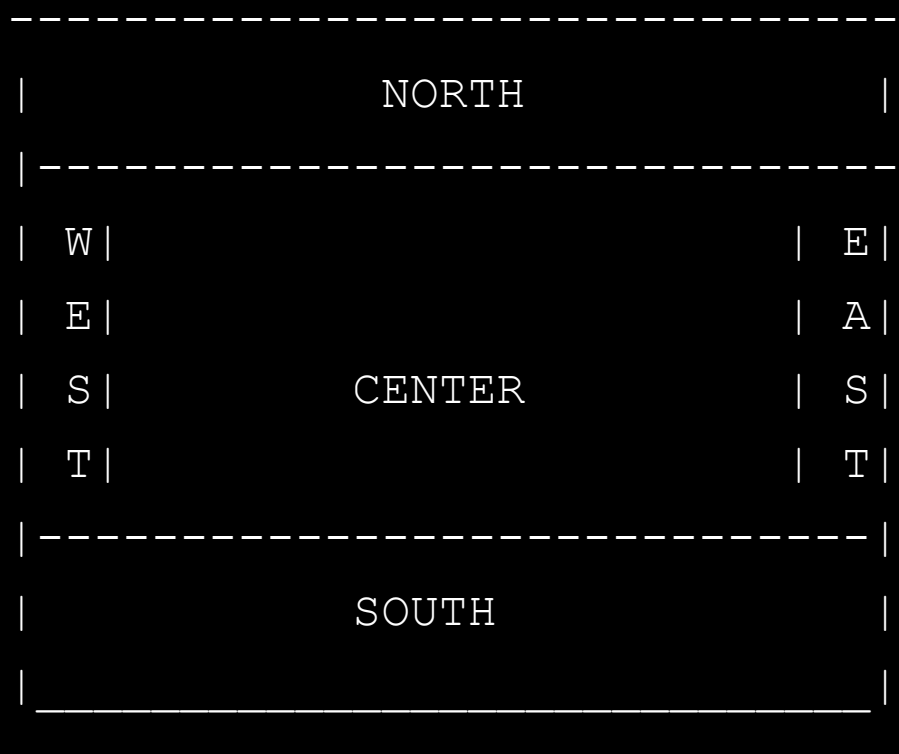
Huvudfönster - JFrame

JFrame är en container, i vilken vi lägger Components

JMenuBar	En menyrad
JMenu	En meny
JMenuItem	Ett menyelement

Sammanfattning, forts...

JFrame har en default layout manager som heter BorderLayout

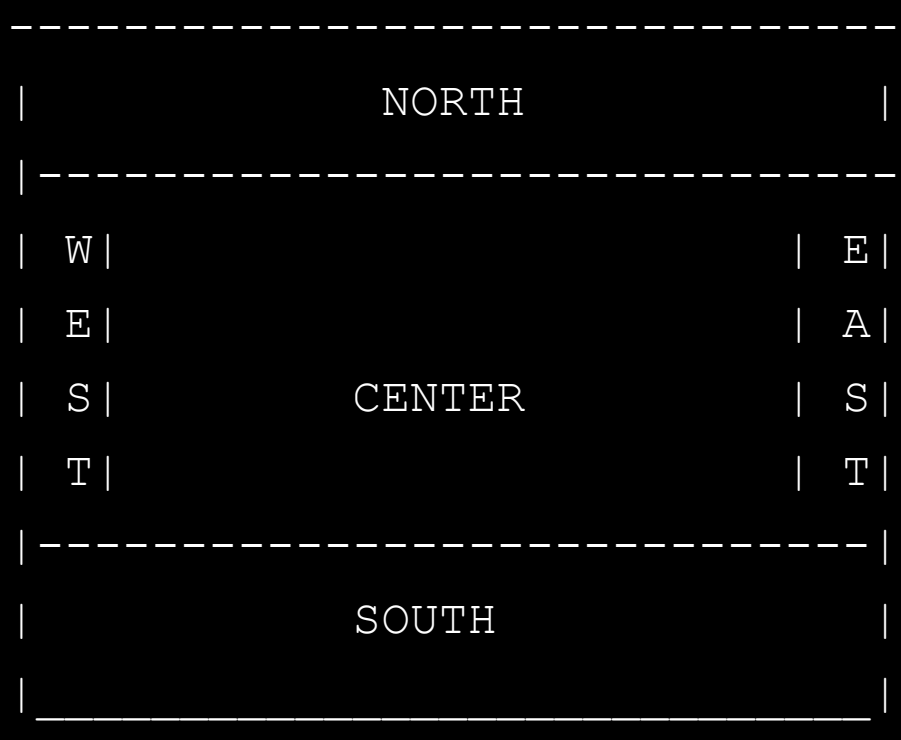


Java-concept och Swing 2

It Don't Mean a Thing If It Ain't Got That Swing

LayoutManager

JFrame har en default layout manager som heter BorderLayout



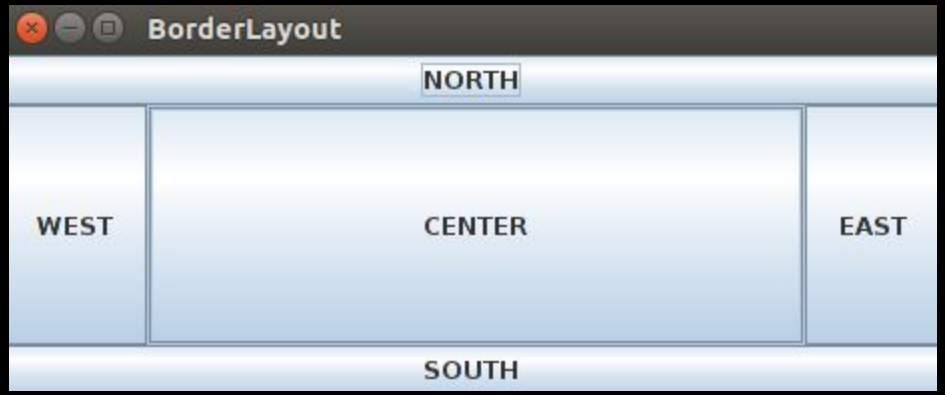
BorderLayout

I NORTH och SOUTH tar komponenter maximal bredd men behåller sin föredragna höjd.

I WEST och EAST tar komponenter maximal höjd men behåller sin föredragna bredd.

I CENTER växer komponenter på alla bredder

BorderLayout - example



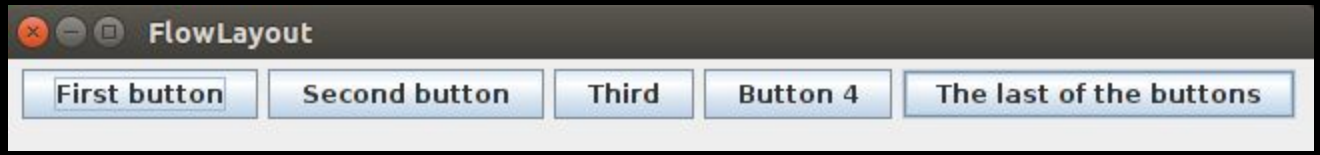
FlowLayout

Vi kan ändra layout för en frame:

```
frame.setLayout(new FlowLayout());
```

FlowLayout lägger ut komponenter i rad i den ordning vi anger. Från höger till vänster och om de inte får plats på en rad, så fortsätter man på en ny rad, centrerat.

FlowLayout - exempel



Komponenterna behåller sin föredragna storlek.

GridLayout - ett rutnät

Komponenter placeras ut i ordning, vänster till höger enligt kolumner och rader.

Komponenter växer åt alla håll om fönstret förstoras.

Kombinera olika layouter

Ofta vill man ha lite olika layout på olika delar.

Paneler är här användbara. En panel är bara en yta som kan innehålla komponenter.

Panelen kan ha en annan layout än fönstret.

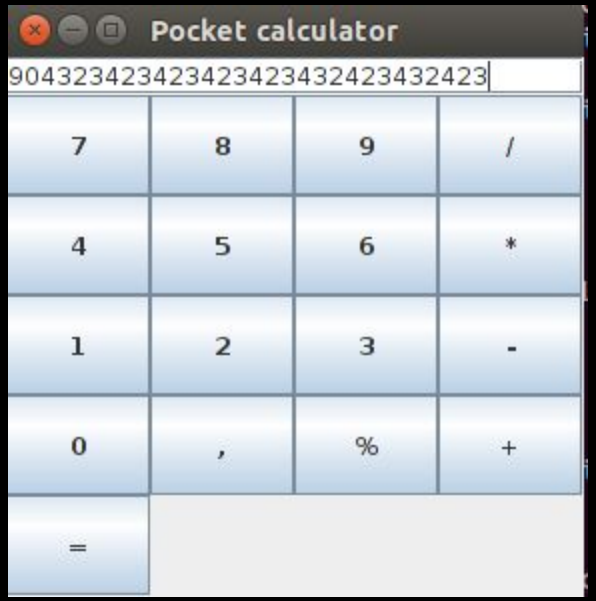
Exempel - miniräknare

Idé: Fönstret har BorderLayout och vi lägger displayen i NORTH.

Vi lägger knapparna i CENTER.

Ingen logik, bara exempel på layout här!

Calculator - example



Kodsnuttar från Calculator

```
JFrame frame = new JFrame("Pocket calculator");
JTextField display = new JTextField(20);
JPanel buttons=new JPanel();
buttons.setLayout(new GridLayout(0,4)); // 4 cols, as many rows as needed

JButton button9 = new JButton("9"); // and so on...

buttons.add(button7);
buttons.add(button8);
buttons.add(button9);
buttons.add(buttonDiv); // First row!

// and so on

frame.add(display, BorderLayout.NORTH);
frame.add(buttons, BorderLayout.CENTER);
```

Sammanfattning

BorderLayout - Smala ytor i N,W,S,E, bred yta i CENTER

FlowLayout - Allt i en rad, fyller på mer fler rader centrerat

GridLayout - kolumner och rader

Layouter går att kombinera, t ex med paneler

Sammanfattning komponenter

Vi har sett:

JButton - en knapp

Panel - en tom yta att lägga andra komponenter i

JTextField - ett textinmatningsfält

(Första filmen såg vi också menyrader, menyer och menyelement i)

Java-concept och Swing 3

Papa’s got a brand new bag
GridBagLayout

En lite krångligare/kraftfullare layout

GridBagLayout är betydligt kraftfullare än de vi sett hittills. Kraftfull betyder dock ofta krångligare att använda...

GridBagLayout - have it your way

Man specificerar precis hur komponenter ska placeras ut och var:

gridx, gridy - vilken cell? 0,0 är högst upp, tv

gridwidth, gridheight - hur stor plats?

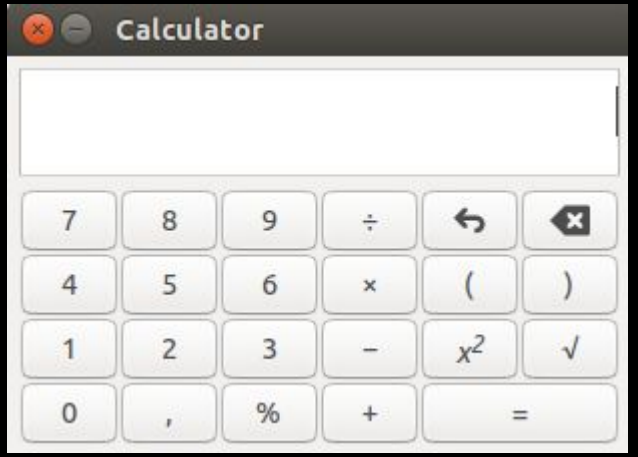
fill - om komponenten skalas om, hur göra?

(det finns mer men ovanstående är basic ;-))

GridBagLayout - en snyggare calc!

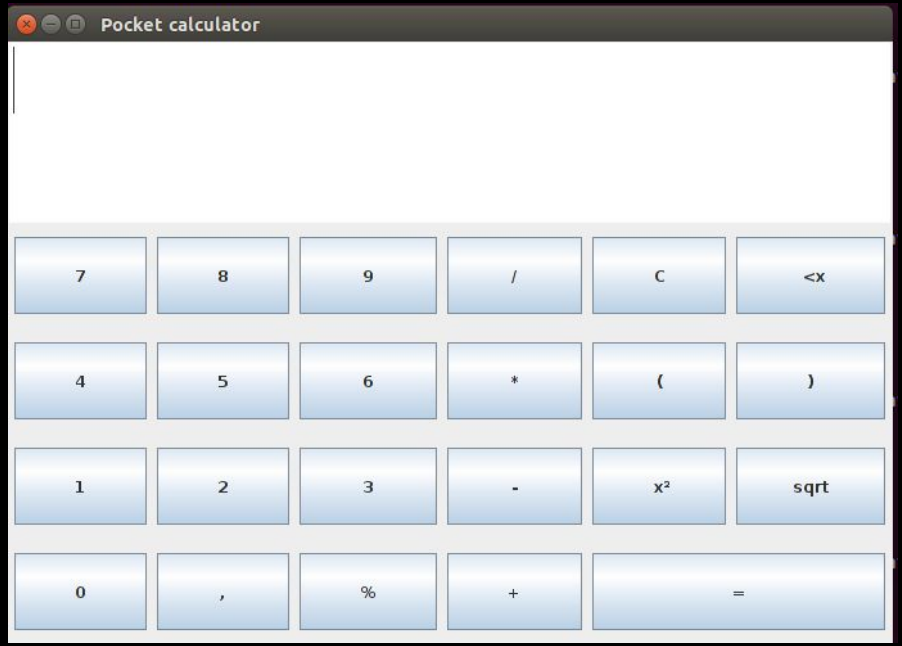
Vi försöker härma den här:

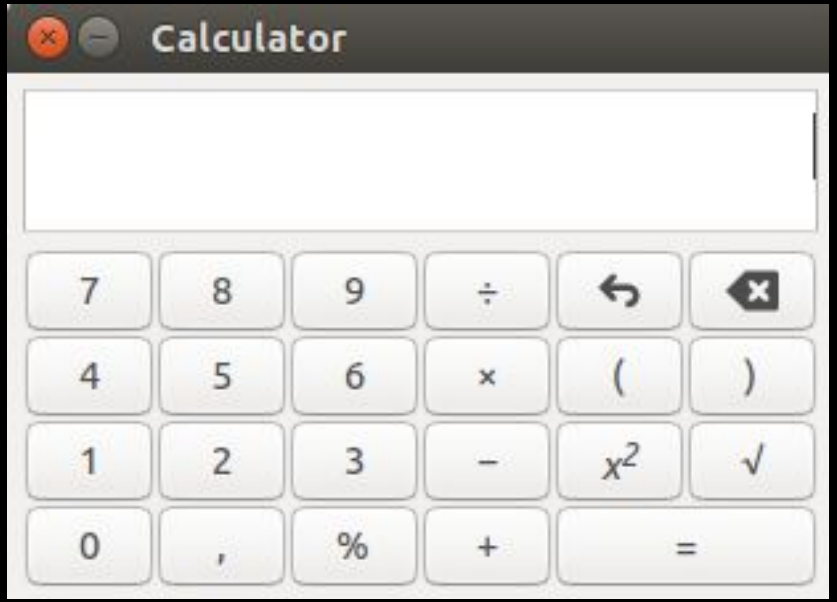
Display: 2 rader, 6 kolumner



“=” ska vara två knappar bred

Calc copycat





Calculator - kodfragment

```
// En JFrame har en ContentPane som är en Container
Container pane = frame.getContentPane();
pane.setLayout(new GridBagLayout());
// Skapa ett constraints-object
GridBagConstraints c = new GridBagConstraints();
// Sätt egenskaper för c:
c.weightx=1;       c.weighty=1;
c.gridwidth  = 6; c.gridheight = 2;
c.gridx = 0;       c.gridy = 0;
JTextPane display = new JTextPane();
pane.add(display, c); // add i klassen Container
```

Metoden add i Container

```
public void add(Component comp,
        Object constraints)
```

Adds the specified component to the end of this container. Also notifies the layout manager to add the component to this container's layout using the specified constraints object.

Knapparna i första raden

```
JButton button7 = new JButton(" 7 ");
c.gridx = 0; c.gridy = 3;
pane.add(button7, c);
JButton button8 = new JButton(" 8 ");
c.gridx = 1; c.gridy = 3;
pane.add(button8, c);
JButton button9 = new JButton(" 9 ");
c.gridx = 2; c.gridy = 3;
pane.add(button9, c);
JButton buttonDiv = new JButton(" / ");
c.gridx = 3; c.gridy = 3;
pane.add(buttonDiv, c);
JButton buttonClear = new JButton(" C ");
c.gridx = 4; c.gridy = 3;
pane.add(buttonClear, c);
JButton buttonDel = new JButton(" <x");
c.gridx = 5; c.gridy = 3;
pane.add(buttonDel, c);
```

Likamedknappen är bredare

```
JButton buttonEquals = new JButton(" = ");
c.gridwidth  = 2; // Två knappar bred
c.gridheight = 1;
c.gridx = 4;
c.gridy = 6;
pane.add(buttonEquals, c);
```

Java-concept och Swing 4

Refactoring

Upprepning av liknande kod

```
// " sym " [ x ] [ y ] pane.add
JButton button7 = new JButton(" 7 ");
c.gridx = 0; c.gridy = 3; pane.add(button7, c);
JButton button8 = new JButton(" 8 "); c.gridx = 1; c.gridy = 3;
pane.add(button8, c);
JButton button9 = new JButton(" 9 ");
c.gridx = 2; c.gridy = 3; pane.add(button9, c);
JButton buttonDiv = new JButton(" / ");
c.gridx = 3; c.gridy = 3; pane.add(buttonDiv, c);
JButton buttonClear = new JButton(" C ");
c.gridx = 4; c.gridy = 3; pane.add(buttonClear, c);
JButton buttonDel = new JButton(" <x");
c.gridx = 5; c.gridy = 3; pane.add(buttonDel, c);
// OSV (detta vara bara första raden med 6 knappar...)
```

Refactoring - lyft ut upprepning

```
private static GridBagConstraints c
  = new GridBagConstraints();
private static JFrame frame
  = new JFrame("Pocket calculator");

private static void addButton(String sym, int x, int y){
  Container pane = frame.getContentPane();
  c.gridx = x;
  c.gridy = y;
  pane.add(new JButton(sym), c);
}
```

Vinsten? Mer kompakt kod

```
// First row:
addButton(" 7 ",0,3); addButton(" 8 ",1,3); addButton(" 9 ",2,3);
addButton(" / ",3,3); addButton(" C ",4,3); addButton(" <x",5,3);
// Second row:
addButton(" 4 ",0,4); addButton(" 5 ",1,4); addButton(" 6 ",2,4);
addButton(" * ",3,4); addButton(" ( ",4,4); addButton(" ) ",5,4);
// Third row:
addButton(" 1 ",0,5); addButton(" 2 ",1,5); addButton(" 3 ",2,5);
addButton(" - ",3,5); addButton(" x²",4,5); addButton("sqrt",5,5);
// Last row:
addButton(" 0 ",0,6); addButton(" , ",1,6); addButton(" % ",2,6);
addButton(" + ",3,6);

JButton buttonEquals = new JButton(" = "); // = är speciell...
c.gridwidth  = 2; c.gridheight = 1;        // vi kör gamla sättet
c.gridx = 4; c.gridy = 6;
pane.add(buttonEquals, c);

// Jämför med tidigare kod för bara första raden!
```

Java-concept och Swing 5

Listen and take action

Registrera en lyssnare på en knapp

```
// ActionListener är ett interface med blott en metod


// Vi använder en anonym inre klass för att registrera en lyssnare:
b.addActionListener(
  // Anonym implementation av interfacet ActionListener
  new ActionListener(){
    public void actionPerformed(ActionEvent e){
      display.setText(e.getActionCommand());
    }
  });
```

Registrera en lyssnare på en knapp

```
// ActionListener är ett interface med blott en metod


// Vi använder en anonym inre klass för att registrera en lyssnare:
b.addActionListener(
  // Anonym implementation av interfacet ActionListener
  new ActionListener(){
    public void actionPerformed(ActionEvent e){
      display.setText(e.getActionCommand());
    }
  });

// Ett ActionEvent från en JButton har knappens text som “actionCommand()”
```