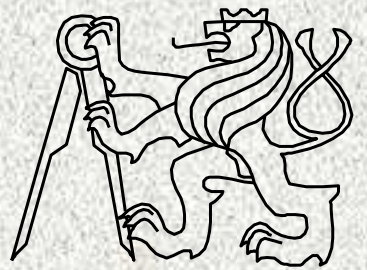
Grafické uživatelské rozhraní v Javě



A0B36PR2-Programování 2

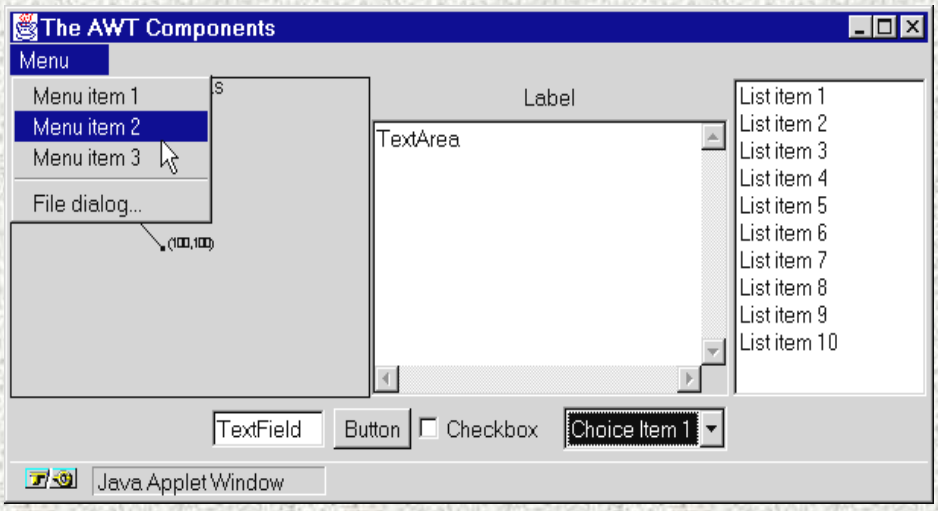
Fakulta elektrotechnická
České vysoké učení technické

GUI (Graphical User Interface)

Vizuální a interaktivní komunikaci počítač-člověk podporují balíčky:

- **`java.awt`** - obsahuje:
  - komponenty: knoflíky, textová pole, menu, posuvníky, grafiku ....
  - kontejnery: tj. komponenty, do kterých lze vkládat komponenty,
  - layout managery: rozmisťují komponenty v ploše kontejneru.
- **`java.awt.event`** - události a jejich zachytávání.
- **`javax.swing`** - podstatně vylepšuje GUI, nahrazuje plně java.awt.

Ukázka v awt :



Zásady návrhu GUI - 1

- Kvalita GUI postmaně ovlivňuje efektivitu práce uživatele (i negativně) .
- Uživatel podle GUI posuzuje kvalitu aplikace (hazuka zpochybňuje).
- Usilujte o jednoduše elegantní návrh s intuitivní a konzistentní funkcionalitou.
- Rozumně s rozměry, barvami a kontrasty - mají asociované významy.
- Respektujte styl a zvyklosti uživatele.
  - Poznejte zkušenosti a prostředí uživatelů (laik vs. expert).
  - Uvažte jak eventuálně hladce dále GUI rozšiřovat.
  - Jednoduchost bývá lepší než složitost - nepřeplácat komponentami.

Zásady návrhu GUI - 2

- Uživatel se nesmí ztratit – vyznačujte stopu jak se tam dostal.
- Nezahltit informacemi a vizuálními podněty – usability testy prototypů.
- Udržovat konzistenci použití komponent.
  - Konzistence mezi aplikacemi – look and feel.
  - Vnitřní konzistence aplikace.
- Komponenty mají váhu – navozují závažnost (velikost, font, barva).
- Pozor na ošidné layouty a resizing.
- Uvažte standardy a zvyklosti platforem.
- Uvažte i18n (i-nternationalizatio-n)

Knihovny pro GUI

- AWT - Abstract Window ToolKit
  - první, těžké(heavyweight),
  - vykreslení zajišťuje platforma – rychlejší, ne vždy vše funguje vše stejně
  - hierarchický model
- Swing
  - doporučené, nové komponenty (tree-view, list box,...),
  - delegační model pro události
  - mnoho nových vlastností, ikony, tool-tips
  - robustní
  - Look and Feel
  - důsledné oddělení modelu od pohledu a řadiče
- SWT-Standard Widget Toolkit, Eclipse IBM,
  - podobné AWT (platformově závislé vykreslení)
  - mnoho rozšiřujících vlastností

Knihovny pro GUI – které použít?

1. Vyberte si knihovnu
   1. Swing – moderní, pěkný
   2. SWT – také dobrá volba, nutno distribuovat s knihovnou, např. swt.dll
   3. AWT – zastaralé, některé třídy stejné i pro Swing: Color, Point, ...

2. vybrané knihovny se držte, v žádném případě nemíchejte vizuální komponenty z různých knihoven !!! – problémy při překreslování, výběru v menu, ...

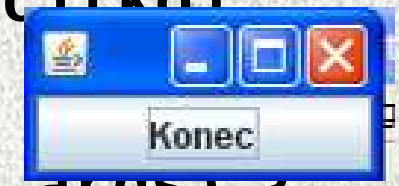
Vytvoření jednoduchého okna - Swing

```
package zaklady;
import javax.swing.JButton;
import javax.swing.JFrame;
public class PrvniOkno extends JFrame{
  JButton tlacitko = new JButton("Konec");
  public PrvniOkno(){
     this.getContentPane().add(tlacitko);
  }
  public static void main(String[] args) {
     PrvniOkno po = new PrvniOkno();
     po.pack();
     po.setVisible(true);
} }
```

Swing

Okno je potomkem JFrame

Vložení tlačítka do okna

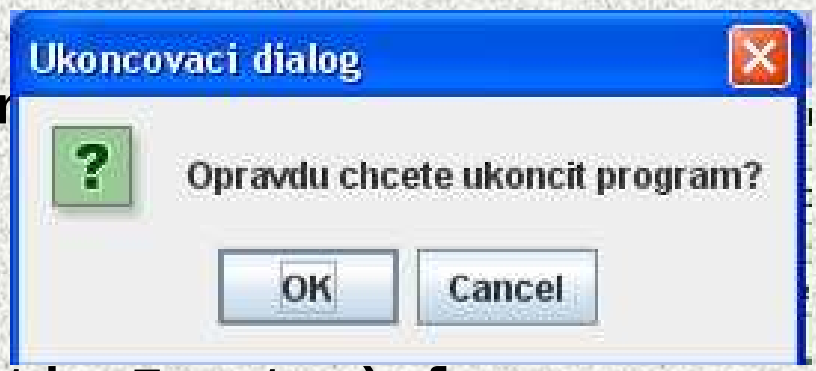


Zabal okno, vytvoř dostatek místa pro všechny komponenty

zobraz okno

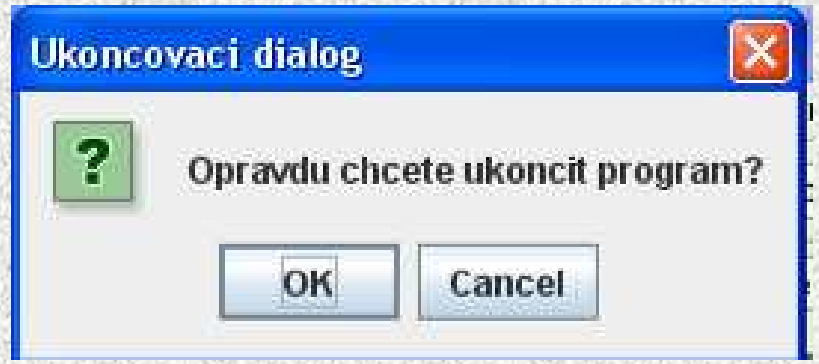
Jednoduché zpracování události od tlačítka

```
…
public class PrvniOkno extends JFrame implements
   ActionListener{
   JButton tlacitko = new JButton("Konec");
   public PrvniOkno(){
       this.getContentPane().add(tlacitko);
       tlacitko.addActionListener(this);
   }
   public static void main(String
       PrvniOkno po = new PrvniOkn
       po.pack();
       po.setVisible(true);
   }
   public void actionPerformed(ActionEvent e) {
//      obsluha události, viz další slide
 }
```

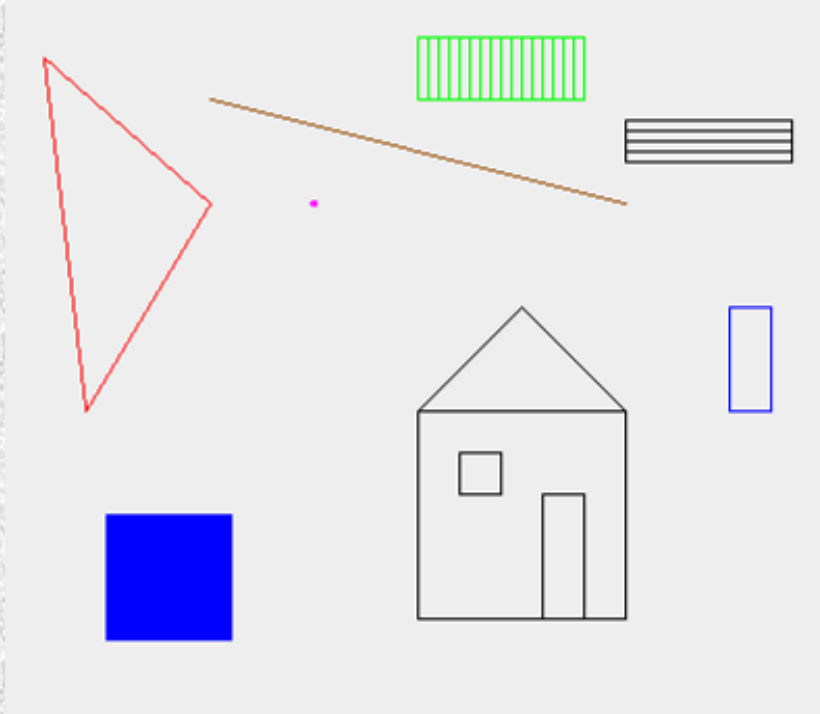


Obsluha dialogu (podrobně v následující přdnášce)

```
public void actionPerformed(ActionEvent e) {
switch (JOptionPane.showConfirmDialog(this,
 "Opravdu chcete ukoncit program?", "Ukoncovaci dialog",
                                JOptionPane.WARNING_MESSAGE)){
case JOptionPane.OK_OPTION: //ukonci program
   System.exit(0);break;
case JOptionPane.CANCEL_OPTION: //rozmyslel si to,nedelej nic.
   }
}
```

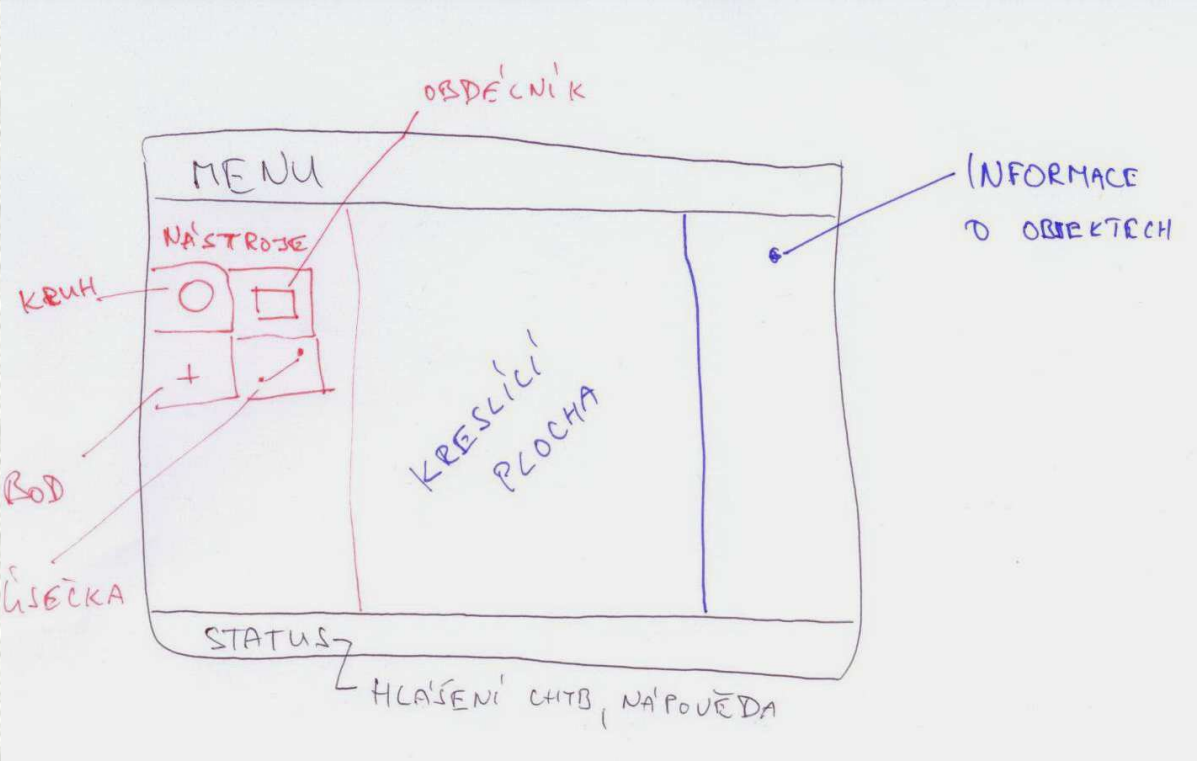


Komplexnější příklad GUI



- Minule jsme vytvořili sadu objektů pro vektorové kreslítko
- body,
- úsečky,
- obdélníky,
- ...

Grafický návrh vektorového kreslítka



- určíme základní rozvržení
- určíme chování okna při zvětšování a zmenšování
- určíme prvky, které použijeme v jednotlivých oblastech

Základní součásti GUI

1. Komponenty (dialogové prvky) - v knihovně *javax.swing*

- tlačítka, seznamy, jezdci, textová pole, zatrhávací tlačítka, rádio tlačítka, …
- společné metody pro velikost, barvu, umístění textu, …

2. Kontejnery (v oknech) - v knihovně *javax.swing*

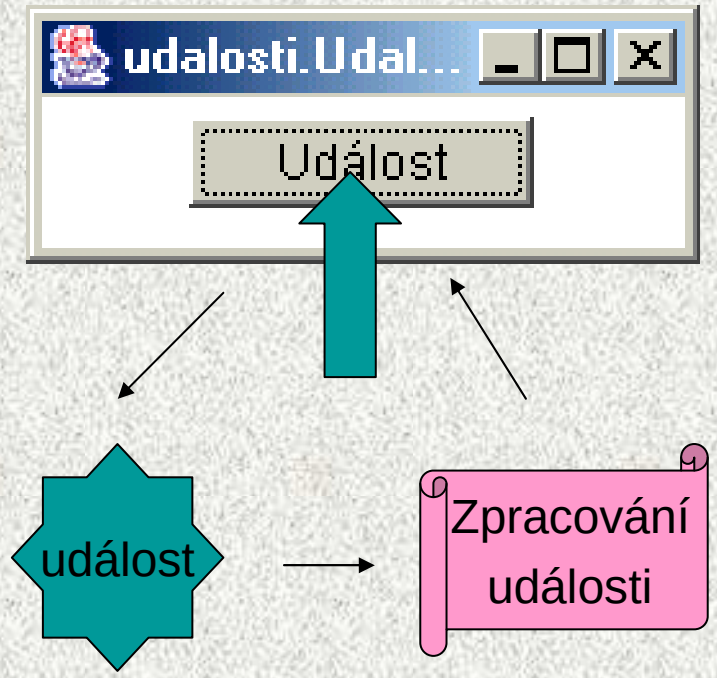
- okna obsahují kontejnery
- komponenty musí být umístěny v kontejnerech
- dva základní typy kontejnerů
  - `JPanel`– nejjednodušší, přidělí se komponenty (**javax.swing**)
  - `JFrame` – složitější, ale více možností (**java.awt**)

3. Správce rozmístění (Layout Manager) - v knihovně *javax.swing a java.awt*

- definuje pozici komponent v kontejneru
- postupně, pevná pozice, podle mřížky, sdružování, ..
- vzhled a chování celé aplikace, Windows, Motif nebo Metal (Look and Feel)

4. Obsluha událostí (events) - v knihovně *java.awt.event*

Zpracování událostí



Obsluha událostí

Mechanismus reakce na akci uživatele

- stisk tlačítka, zadání textu, stisk tlačítka myši, …

1. Pro každou komponentu je třeba:
   - deklarovat typ zachycované události, kterou je zájem zpracovat
   - určit „posluchače“, který má událost obsloužit
2. Akcí uživatele vznikne událost
   - událost je objektem Javy!
3. Události jsou zachyceny
   - události jsou zpracovány (obslouženy) „posluchači“ (listener)
     - třídami s **uživatelskými metodami pro reakci na událost**
   - „posluchači“ jsou třídy, které implementují rozhraní naslouchání
     - musejí mít schopnost „naslouchání“

Pozn.: O obsluze událostí bude speciální přednáška

Přehled základních prvků GUI

Komponenty

- JButton Tlačítko, událostí je kliknutí na tlačítko
- JCheckBox Zaškrtávací políčko, prvek je/není vybrán,událost po uzavření okna
- JComboBox Rozevírací seznam položek, klepnutím na položku se generuje událost
- JLabel Zobrazení popisku, bez generování události
- JPasswordField Zobrazení hesla, místo vložených znaků se zobrazí hvězdičky
- JRadioButton Přepínač, množina tlačítek, jen jedno lze zvolit, událost po uzavření okna
- JTextField Zadávání textu, událost se generuje po uzavření okna

Kontejnery

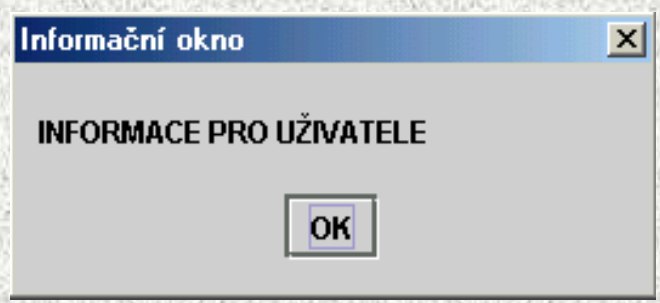
- JFrame Kontejner s ohraničením a záhlavím
- JPanel Kontejner bez ohraničení, implicitně rozmístění FlowLayout, jednodušší

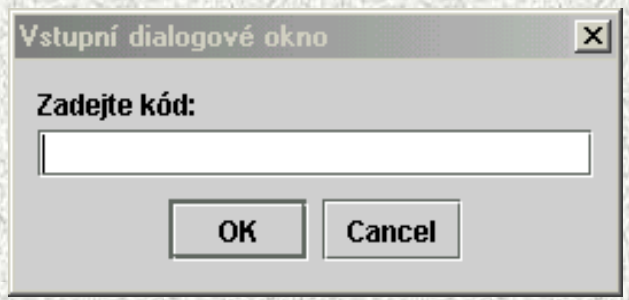
Rozvržení, správci - Layout Manager

- BorderLayout Rozmístění podle světových stran
- BoxLayout Rozmístění do podkontejnerů, sdružování komponent
- FlowLayout Rozmístění zleva doprava a shora dolů, nejjednodušší, implicitní
- GridLayout Rozmístění do pevné mřížky

Jednoduché GUI

```
import javax.swing.*;
class Demo1 {
public static void main(String[] args) {
String str;
JOptionPane.showMessageDialog(null,"INFORMACE PROUŽIVATELE",
            "Informační okno", JOptionPane.PLAIN_MESSAGE);
str = JOptionPane.showInputDialog(null, "Zadejte kód:   ",
    "Vstupní dialogové okno", JOptionPane.QUESTION_MESSAGE);

System.out.println("Kód je:  " + str);
System.exit(0);
  }
}
```





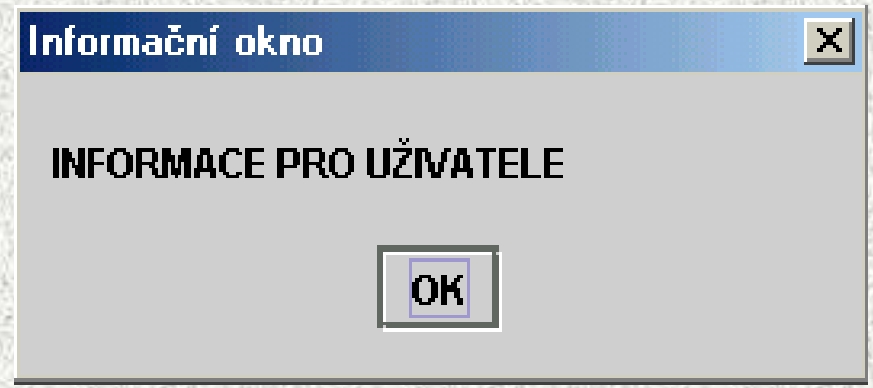
Jednoduché GUI – zobrazeni informace

```
JOptionPane.showMessageDialog(null,"INFORMACE PROUŽIVATELE",
                "Informační okno", JOptionPane.PLAIN_MESSAGE);
```

- metoda třídy `JOptionPane`, knihovny `javax.swing`:

```
+--java.awt.Component
        +--java.awt.Container
                +--javax.swing.JComponent
                        +--javax.swing.JOptionPane
```

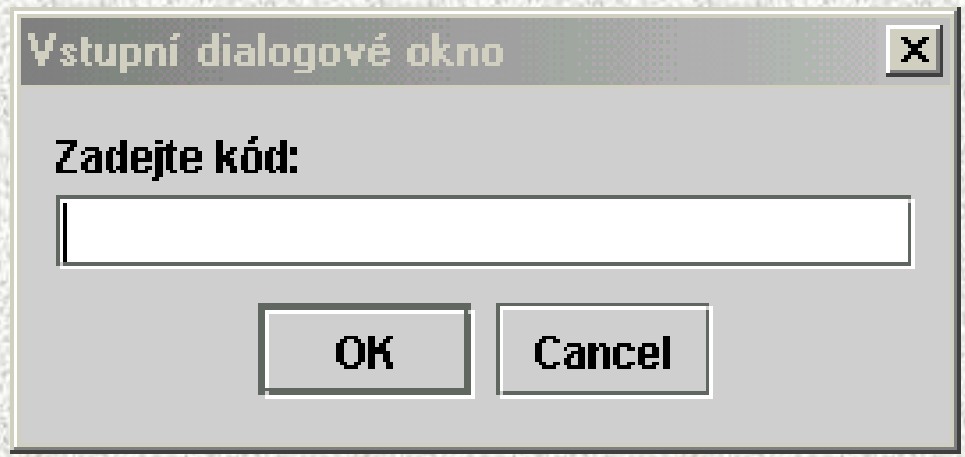
- jiné možné konstanty:
  - `ERROR_MESSAGE`
  - `INFORMATION_MESSAGE`
  - `WARNING_MESSAGE`
  - `QUESTION_MESSAGE`
  - `PLAIN_MESSAGE`



Jednoduché GUI – zobrazeni dialogu

```
str = JOptionPane.showInputDialog(null,
   "Zadejte kód:   ","Vstupní dialogové okno",
   JOptionPane.QUESTION_MESSAGE);
```

- **hodnota ze vstupního pole je hodnotou funkce**
- **nekontroluje povolené hodnoty**
- **kontrolní výstup na příkazový řádek**

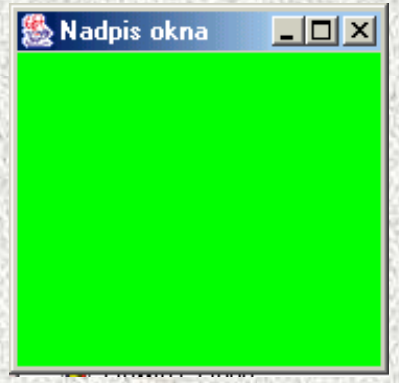


Zobrazení okna

```
class Demo2 {
  public static void main(String[] args) {
    Okno okno = new Okno();
  }
}
class Okno extends JFrame{
public Okno (){
super("Nadpis okna"); // konstruktor JFrame
setSize (300,100);    // nastavení velikosti
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                      // také DO_NOTHING_ON_CLOSE ☺
setVisible(true);} // zobrazení
}
```

Vytvoření kontejneru

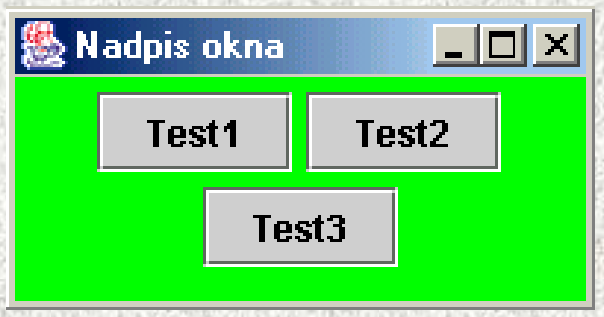
```
import java.awt.*;
import javax.swing.*;

class Okno3 extends JFrame{
public Okno3 (){
super ("Nadpis okna");
setSize (100,100);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
Container kon = getContentPane(); // vrací kontejner
    kon.setBackground(Color.green);
  }
}
```



Správci rozložení – Layout Manager - FlowLayout

- třídy určené k rozmisťování prvků v kontejneru
- nejjednodušší je třída FlowLayout
    - rozmisťuje zprava doleva a shora dolu a doprostřed

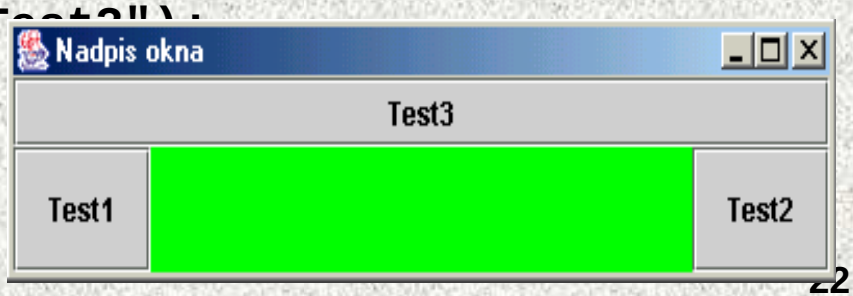
```
class Okno41 extends JFrame{
…
Container kon = getContentPane();
     kon.setBackground(Color.green);
     FlowLayout srb = new FlowLayout();
     kon.setLayout(srb);
     JButton tl1 = new JButton("Test1");
     kon.add(tl1);
     JButton tl2 = new JButton("Test2");
     kon.add(tl2);
     JButton tl3 = new JButton("Test3");
     kon.add(tl3);
     setContentPane(kon);
   }
}
```



Správci rozložení – Layout Manager - BorderLayout

- rozmisťuje do pěti oblastí podle „světových stran“ **BorderLayout**

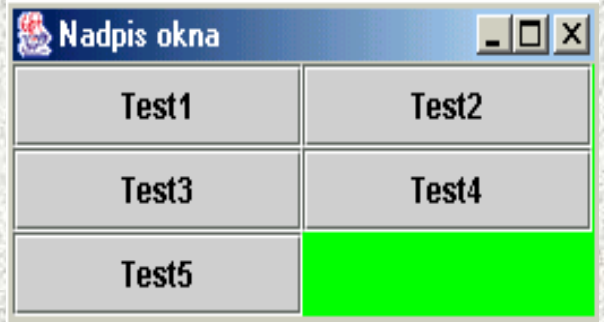
```
class Okno4_1 extends JFrame{
public Okno4_1 (){
…
Container kon = getContentPane();
     kon.setBackground(Color.green);
     BorderLayout srb = new BorderLayout();
     kon.setLayout(srb);
     JButton tl1 = new JButton("Test1");
     kon.add(tl1,srb.WEST);
     JButton tl2 = new JButton("Test2");
     kon.add(tl2,srb.EAST);
     JButton tl3 = new JButton("Test3");
     kon.add(tl3,srb.NORTH);
     setContentPane(kon);
   }
}
```



Správci rozložení – Layout Manager - GridLayout

- Rozložení v mřížce, rafinovanější je **GridLayout**

```
class Okno5 extends JFrame{
[…].
Container kon = getContentPane();
    kon.setBackground(Color.green);
    GridLayout srg = new GridLayout(3,3);
    kon.setLayout(srg);
    JButton tl1 = new JButton("Test1");
    kon.add(tl1);
    JButton tl2 = new JButton("Test2");
    kon.add(tl2);
    JButton tl3 = new JButton("Test3");
    kon.add(tl3);
    JButton tl4 = new JButton("Test4");
    kon.add(tl4);
    JButton tl5 = new JButton("Test5");
    kon.add(tl5);
    setContentPane(kon);
  }
}
```



Dialogové prvky - Tlačítka

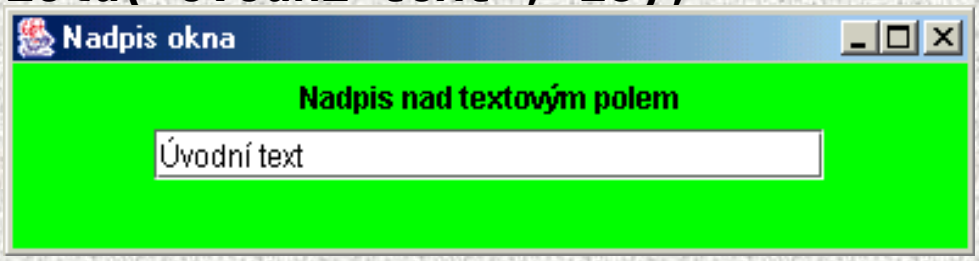
- komunikační komponenty jsou tlačítka - **JButton** – už jsme poznali

```
class Okno6 extends JFrame{
…
setVisible(true);

Container kon = getContentPane();
kon.setBackground(Color.green);
FlowLayout srf = new FlowLayout();
kon.setLayout(srf);
 JButton tl1 = new JButton("Start");
    kon.add(tl1);
    JButton tl2 = new JButton("Stop");
    kon.add(tl2);
    setContentPane(kon);
  }
}
```

Dialogové prvky - Textová pole,popisky

- komunikační komponenty jsou tlačítka - **JButton** – už jsme poznali
- především textová pole (aktivní) - **JTextField** a popisky (pasivní) - **JLabel**

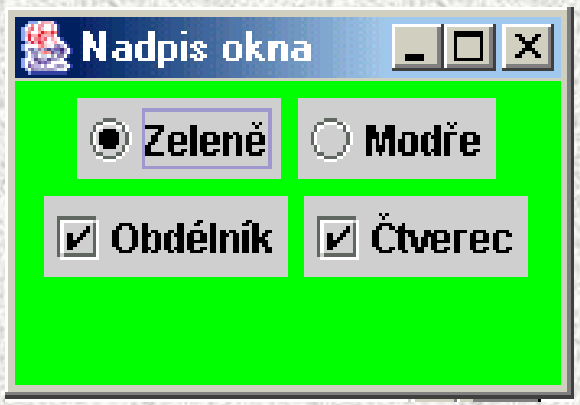
```
class Okno7 extends JFrame{
…
setVisible(true);

Container kon = getContentPane();
kon.setBackground(Color.green);
FlowLayout srf = new FlowLayout();
kon.setLayout(srf);
JLabel popisek = new JLabel("Nadpis nad textovým polem");
kon.add(popisek);
JTextField text = new JTextField("Úvodní text", 25);
kon.add(text);
setContentPane(kon);
  }
}
```



Přepínače (radio) a zaškrtávací tlačítka

- Přepínač umožní výběr jedné možnosti z více - **JRadioButton**
- Zaškrtávací tlačítka umožní zadávání dvouhodnotových parametrů **JCheckBox**

```
class Okno8 extends JFrame{
[…]
kon.setLayout(srf);
JCheckBox zt1 = new JCheckBox ("Obdélník");
JCheckBox zt2 = new JCheckBox ("Čtverec");
ButtonGroup vyhovelNevyhovel = new ButtonGroup();
JRadioButton rt1 = new JRadioButton ("Zeleně");
JRadioButton rt2 = new JRadioButton ("Modře");
vyhovelNevyhovel.add(rt1);
vyhovelNevyhovel.add(rt2);
kon.add(rt1);
kon.add(rt2);
kon.add(zt1);
kon.add(zt2);
setContentPane(kon);
  }
}
```



Seznamy

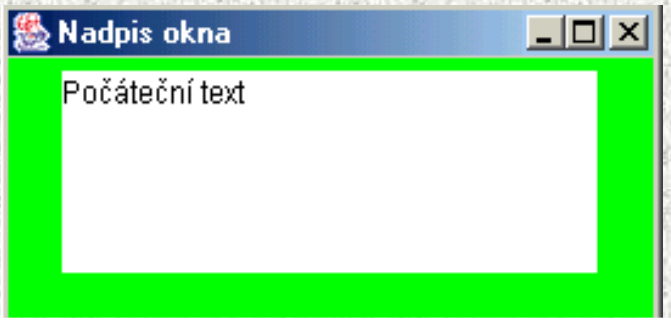
- výběr z více možností - **JComboBox**

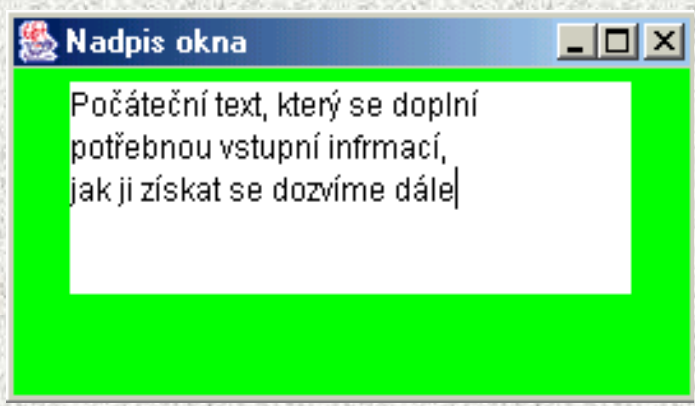
```
class Okno9 extends JFrame{
…
Container kon = getContentPane();
    kon.setBackground(Color.green);
    FlowLayout srf = new FlowLayout();
    kon.setLayout(srf);
    JComboBox rseznam1 = new JComboBox();
    rseznam1.addItem("První");
    rseznam1.addItem("Druhý");
    rseznam1.addItem("Třetí");
    kon.add(rseznam1);
    setContentPane(kon);
  }
}
```

Textová oblast + víceřádková

- Slouží ke vstupu textu s počáteční nápovědou – **JTextArea**
- **pro větší texty slouží JSrollPane, který umožní rolování textu**

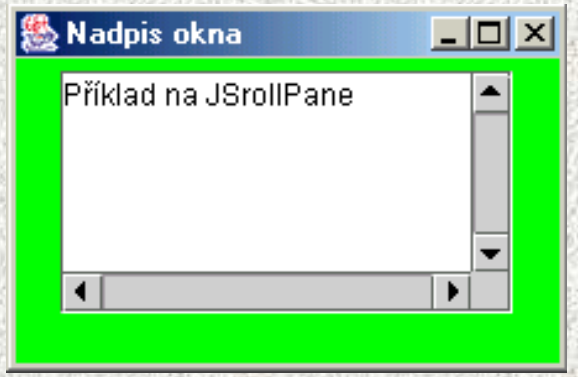
```
class Okno10 extends JFrame{
…
    Container kon = getContentPane();
    kon.setBackground(Color.green);
    FlowLayout srf = new FlowLayout();
    kon.setLayout(srf);
    JTextArea to = new JTextArea("Počáteční text", 5, 20);
    kon.add(to);
    setContentPane(kon);
  }

}
```





Textová oblast + víceřádková

```
class Okno10 extends JFrame{
…

Container kon = getContentPane();
kon.setBackground(Color.green);
FlowLayout srf = new FlowLayout();
kon.setLayout(srf);
JTextArea to = new JTextArea("Příklad na JSrollPane", 5, 15);
JScrollPane rp = new JScrollPane (to,
   JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
   JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
kon.add(rp)
setContentPane(kon);
}
```

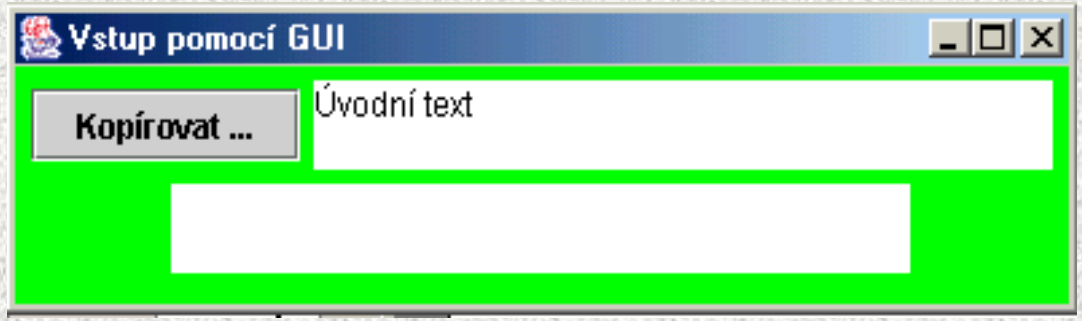


Obsluha událostí z GUI

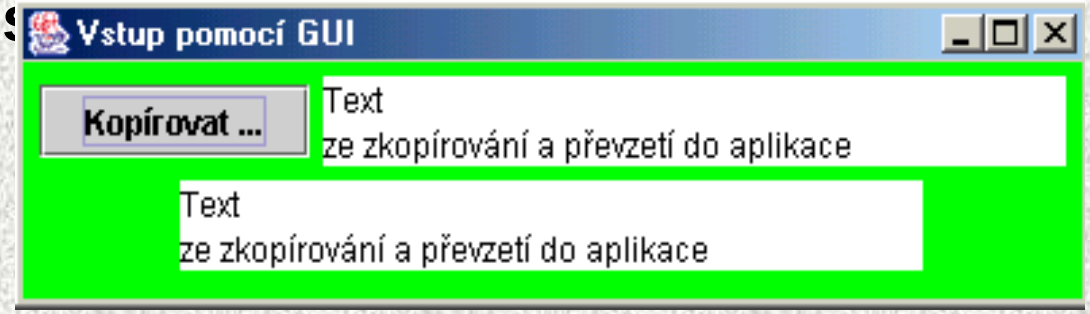
- Obsluha událostí předpokládá (viz speciální přednáška):
  - zdroj události (např. tlačítko)
  - objekt událost (zařídí JVM)
  - přiřazený posluchač (třída, která je schopna naslouchat a je připravena naslouchat)
  - metoda, která obslouží, zareaguje na událost

Obsluha události od tlačítka, příklad

- **Počáteční stav**



- **Stav po sti**



Obsluha události od tlačítka, příklad

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.lang.*;
class Demo121 {
  public static void main(String[] args) {
    Okno121 okno = new Okno121();
}
}
class Okno121 extends JFrame implements ActionListener{
  JTextArea to1 = new JTextArea ("Úvodní text", 2, 25);
  JTextArea to2 = new JTextArea (2, 25);
  JButton tl1 = new JButton ("Kopírovat ...");
```

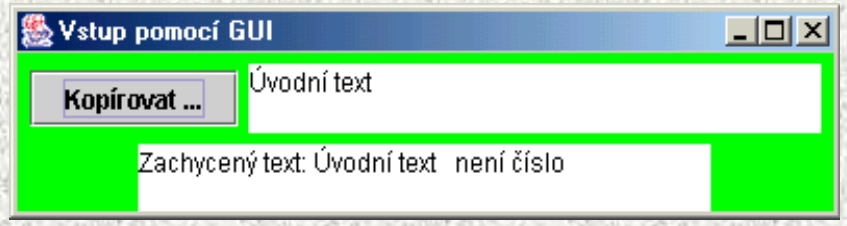
Obsluha události od tlačítka, příklad

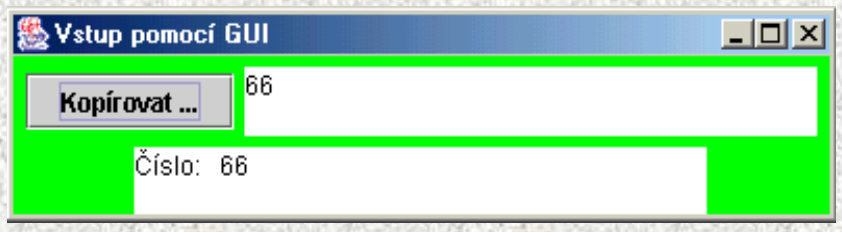
```
public Okno121(){
    super ("Vstup pomocí GUI");
    setSize(400, 100);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
    Container kon = getContentPane();
    kon.setBackground(Color.green);
    FlowLayout srf = new FlowLayout();
    kon.setLayout(srf);
    tl1.addActionListener(this);
    kon.add(tl1);
    kon.add(to1);
    kon.add(to2);
    setContentPane(kon);
      }
 public void actionPerformed (ActionEvent event){
 String t = to1.getText();
 to2.setText(t);
  }
}
```

Obsluha události od tlačítka, příklad zpracování

- u vstupních dat je třeba testování – např. pomocí výjimek, viz speciální přednáška

```
public void actionPerformed (ActionEvent event){
String t = to1.getText();
 try {
 int i = Integer.valueOf(t).intValue();
  to2.setText("Číslo:   " + i);
}
  catch (NumberFormatException e){
 to2.setText("Zachycený text:"+" " + t + " "+ "  není
  číslo!");
}
}
```





Grafika v Javě I

- Základní třída **`java.awt.Graphics, java.awt.Graphics2D (JDK1.2)`**
- Základní možnosti třídy **`Graphics`**:
  - kreslení základní 2D objektů, grafických primitiv
  - vykreslování textu a obrázků
  - nastavování a testování barev, fontu, ořezání, ploch
- Okamžik zobrazení není časově determinován
- Kreslit lze v komponentách **`Panel, Canvas`** (plátno) a **`Applet`**
- Vlastní kreslení je popsáno v metodě **`Container.paint(Graphics g)`**
  - „*vymaluj toto plátno, tento rámec …*“
  - parametr **`Graphics g`** je abstraktní, formální,
    - "automatický" objekt, o který se nestaráme
    - definuje počáteční vykreslení, nevolá se přímo! …
- **`Component`**.**`repaint()`** - vyvolá metodu **`update()`** - **“**jakmile to půjde“
- **`Container`**. **`update()`** - smaže plochu okna a volá metodu **`paint()`**

Grafika v Javě II

- Třída `Canvas`, reprezentuje komponentu pro kreslení
  - má jedinou vhodnou metodu **`void paint(Graphics g)`**
  - negeneruje žádné události
  - slouží pro vykreslování, tvorbu ikon
  - po reakci uživatele voláme metodu **`repaint()`**
- Souřadnicový systém
  - počátek souřadnic v levém horním rohu obrazovky
    - osa y zleva doprava, maximální hodnota y: **`getHeight()`**
    - osa x shora dolu, maximální hodnota x: **`getWidth()`**

Grafika v Javě, příklad

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class GrafikaSimple extends JFrame {
  public static void main(String[] args) {
    JFrame f = new GrafikaSimple();
  }
GrafikaSimple() {
  super ("Jednoduchá grafika");
  setSize(300, 200);
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setVisible(true);
  }
public void paint(Graphics g) {
    g.drawString("Hello, World",  175, 100);
    g.drawOval(70,100,140,70);
  }
 }
```

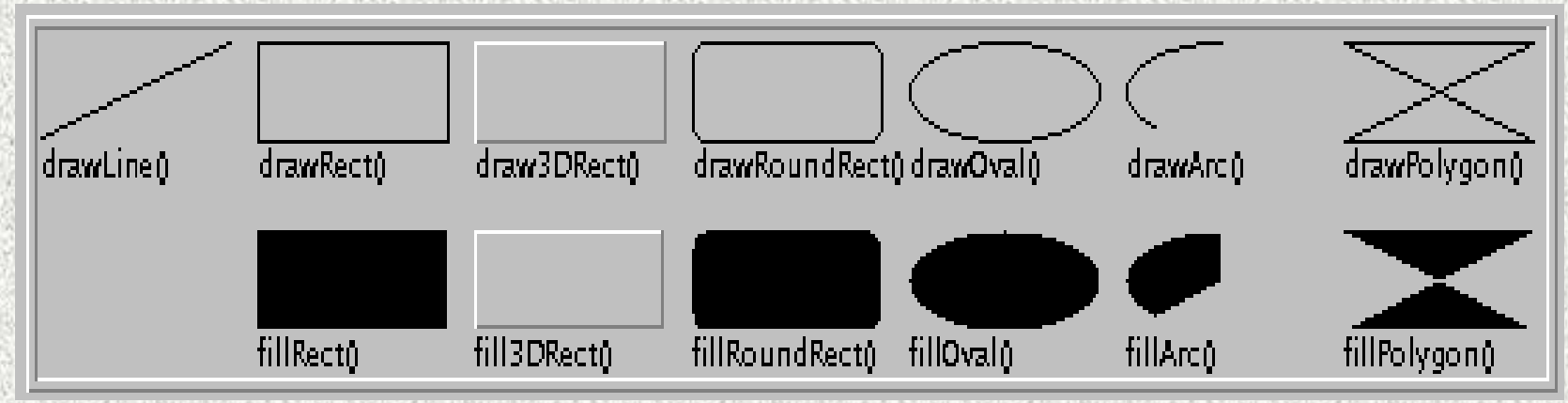
Grafická primitiva

Grafická primitiva:

- kreslení tvaru, obrysu – **`drawXXX()`**
- vyplnění tvaru – **`fillXXX()`**

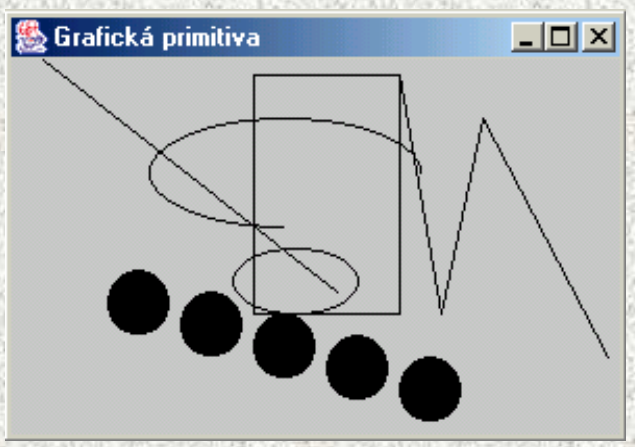
`XXX:`

- **`Line (jen draw)`**
- **`Rect, 3Drect, RoundRect`**
- **`Oval`**
- **`Arc`**
- **`Polyline`**



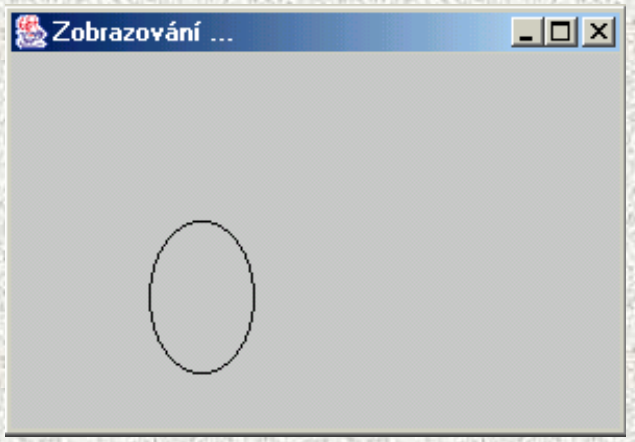
Příklad na grafiku

```
PrimitivaLine() {
super ("Jednoduchá grafika");
setSize(300, 200);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
}
public void paint(Graphics g) {
g.drawLine(15, 20, 160, 130);
g.drawRect(120,30, 70,110);
g.drawOval(110,110,60,30);
g.drawArc(70,50,130,50,0,270);
  int[] xp = {190, 210, 230, 290};
  int[] yp = {30, 140, 50, 160};
g.drawPolyline(xp,yp,4);
 for (int i = 0; i<5 ; i++) {
g.fillArc(50+i*35,120+i*10,30,30,0,360);
}
```



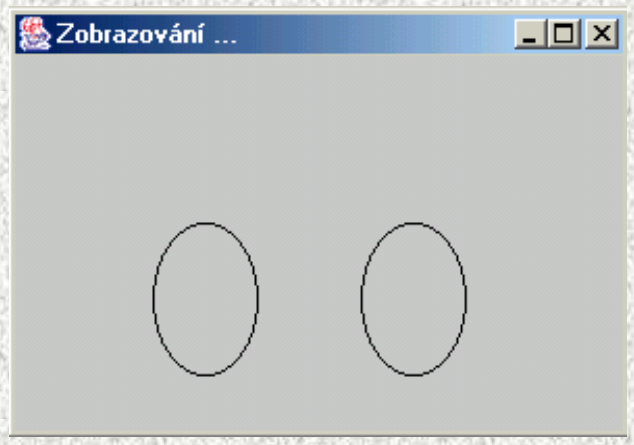
Zobrazování, paint, repaint, update

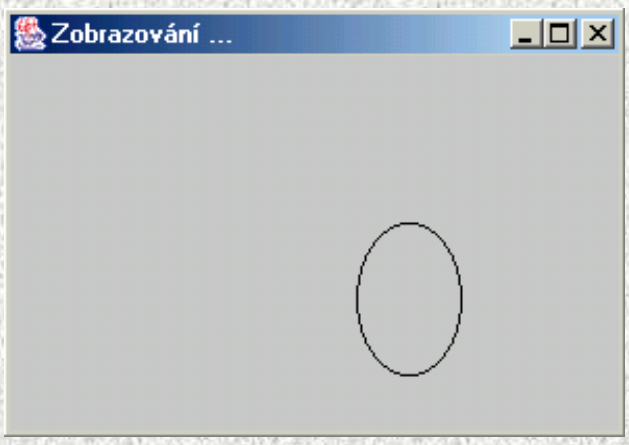
```
public class GrafikaSimplePokus extends JFrame {
  int x= 70; int y = 100; int w = 50; int h = 70;static int
   r=0;
  public static void main(String[] args) {
   GrafikaSimplePokus fr=new GrafikaSimplePokus();
   try { Thread.sleep(2000);}
   catch (InterruptedException ex) {}
   fr.x+=100;
   fr.repaint();
 }
  public GrafikaSimplePokus() {
    super ("Zobrazování ...");
    setSize(300, 200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
    }
```



Zobrazování, paint, repaint, update

```
//public void update(Graphics g){
 // paint(g); }

  public void paint(Graphics g) {
// if (r>0)g.clearRect(0, 0,300, 200);
  g.drawOval(x,y,w,h);
    System.out.println("paint" + r++);
  }}
```





Zobrazování, `paint, repaint, update`

- Metoda `paint` je volána automaticky při “změně” okna a vykreslí grafický kontext (Graphics) definovany obsahem `paint`
- Metoda `repaint`
  - volá `paint`
  - „přikreslí” přes stávající okno aktuální grafický kontext
  - volá `update`
- Je-li metoda `update` přetížena, pak smaže celé okno při manipulaci s oknem a volá `paint`

Přehled…

- **metoda třídy `JOptionPane`, knihovny `javax.swing`:**

```
+--java.awt.Component
        +--java.awt.Container
                +--javax.swing.JComponent
                        +--javax.swing.JOptionPane
```

Přehled

Třída - Component

Tato třída je velmi bohatá – obsahuje metody pro ovládání:

- velikosti, umístění a viditelnosti
- barvy pozadí a popředí
- událostí
- myši
- klávesnice
- kurzoru
- grafiky
- písma
- obrázky
- animace

JComponent

Důležité: JComponent je potomek třídy java.awt.Container a ten je potomkem Component

(kromě JFrame, JDialog, JWindow, JApplet) kterým zajišťuje:

- okraje (border)
- vysvětlivky (tooltip)
- vymalování (custom painting) – je rafinovanější
- L&F – v pozadí je ComponentUI dle výběru UIManageru
- vlastnosti požadované uživatelem
- podporu accessibility
- podporu DnD (Drag and Drop)
- double buffering – defaultně kreslí do skrytého bufru pro hladké animace
- key binding (vazba na klávesy)

Většina komponent podporuje grafiku a obrázky.

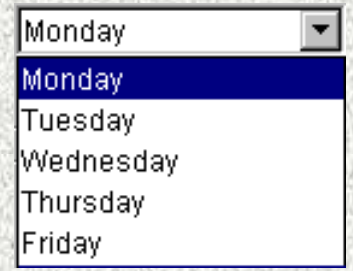
Samotná JComponent je transparentní.

Řídící komponenty

Tlačítka,

- **zvonková JButton,**
- **přepínací TogleButton**
- **zaškrtávací JCheckBox**
- **radio JRadioButton**

- pro spojení do sady se použije ButtonGroup

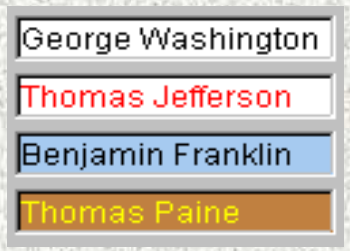


JComboBox

seznam, rozbalí se po kliknutí

JList – seznam, rozbalený

Řídící komponenty

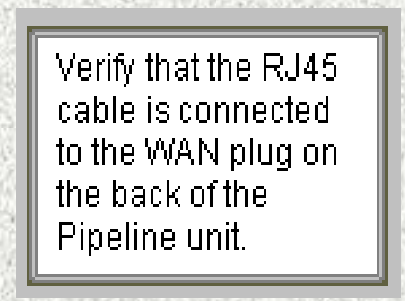


JTextField vstupní pole pro data
lze nastavit, zda má být editovatelné

JMenuBar, JMenu, JMenuItem

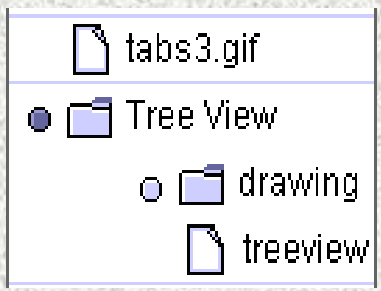
JSlider

Informační komponenty



JText

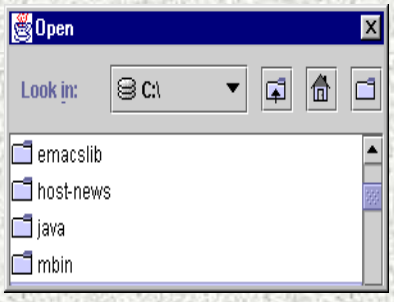
JTable



JTree

editable

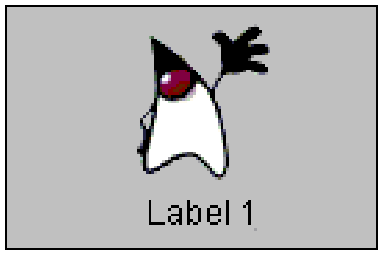
JColorchooser



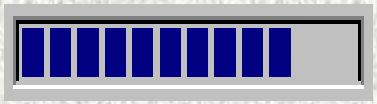
FileChooser

---

not editable



JLabel



Progress bar

Tool tip

Text

Třída javax.text.JTextComponent zajišťuje služby pro své textové potomky:

- Model: document
- View: pro visualizaci
- Controller: editor kit
- Podpora undo/redo
- Caret, listenery a filtry

Přímými potomky jsou třídy:

- JTextField
- JTextArea
- JEditorPane

Pomocí tzv. stylů a atributů lze vytvořit pěkné texty.
Navazují podbalíčky pro rtf a html.

Lištu JMenuBar lze vložit do JFrame, JRootPane, JApplet, JDialog a JInternalFrame – nikoli však do JWindow. Lišta má být opacitní.

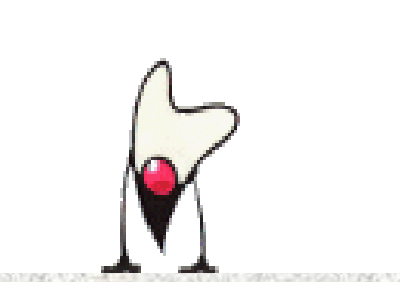
Třída Container

Do kontejnerů se vkládají komponenty a další kontejnery (mimo Window a jeho podtříd - tj. top level containers), čímž vznikne strom. Container vede seznam dle něhož LayoutManagery rozmísťují komponenty.

- Metody pro práci se seznamem komponent:
  - add(Component comp) – přidá na konec.
  - add(Component comp, Object constraints) – a navíc udává omezení.
  - add(Component comp, int index) – přidá na udanou pozici.
  - remove(Component comp), remove(int index), removeAll( ).
  - list(...) – výpis aktuální stavu seznamu.
- Dále lze nastavovat a zjišťovat typ rozmístění metodami:
  - setLayout(LayoutManager mgr) a LayoutManager getLayout( ).
- měnit rozmístění komponent pomocí metody:
  - invalidate( ) – zneplatní tento a všechny obalující kontejnery (parents).
  - validate( ) – znovu rozmístí všechny své komponenty.
- aktualizovat grafiku pomocí update(Graphics g) a paint(Graphics g).
- pracovat s fokusem.

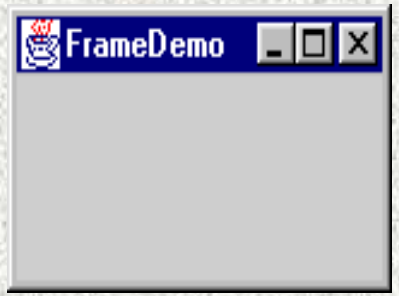
Kontejnery – top level

Okna, pro vkládání dalších prvků



JApplet, zobrazován v HTML prohlížeči

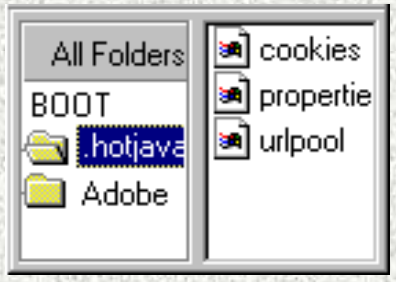
JDialog, dialogové okno, modální = dokud jej uživatel nezavře, nemůže přepnout do jiného okna aplikace



JFrame - okno

Kontejnery

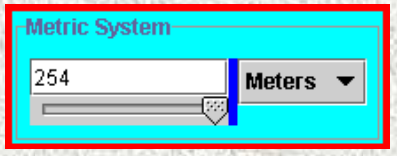
JScrollPane



JSplitPane

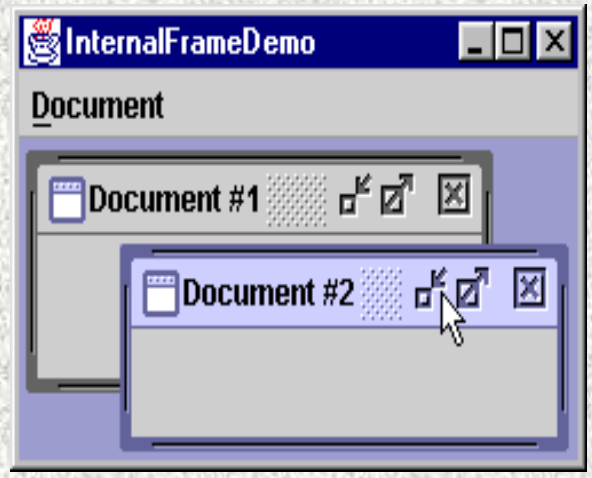
Tabbed pane

Tool bar

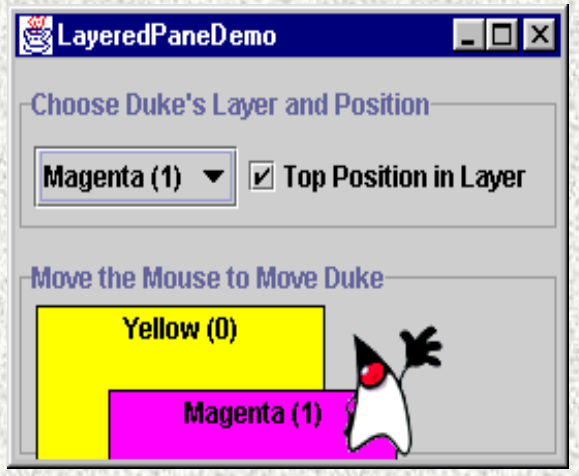


JPanel

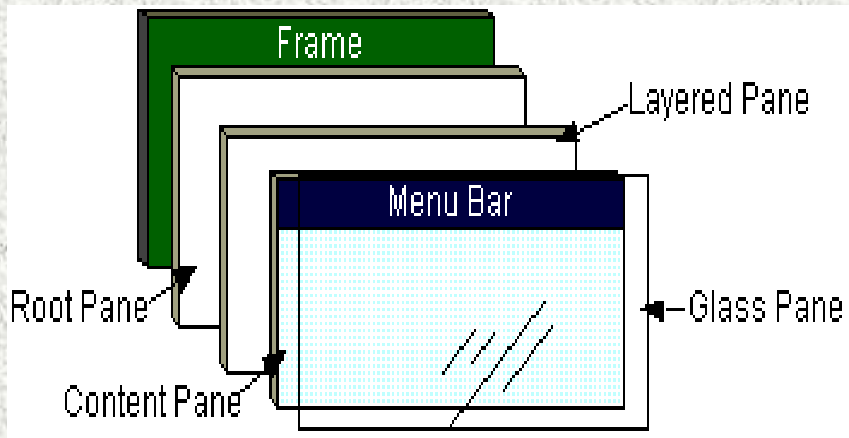
Speciální kontejnery



Internal frame

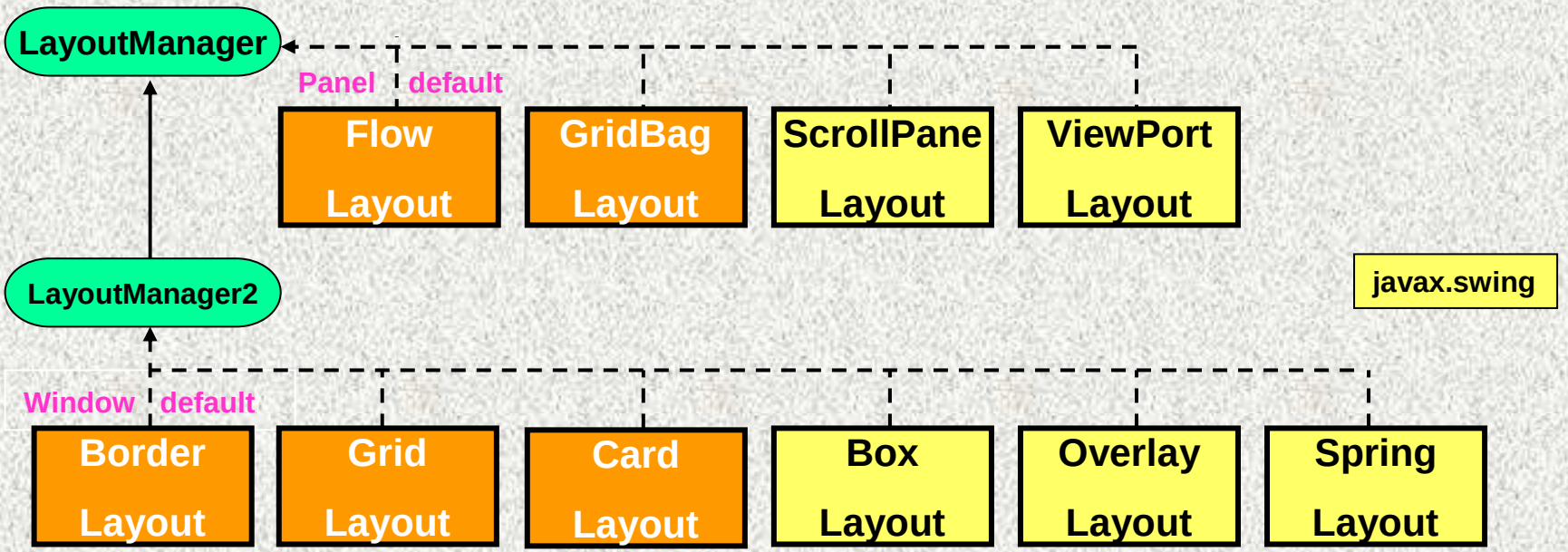


Layered pane



Root pane

LayoutManager ~ rozmisťovač



Flow - jako text přetékající na další řádky (alignment L/R a centrování)

Border - jako mapa s oblastmi (C,N,E,S,W) a jen pro pět komponent.

Card - jako balíček karet - vidět je jen vrchní karta.

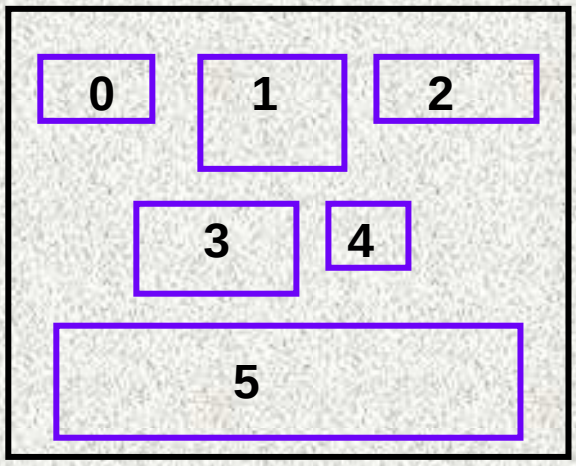
Grid - pravidelná mřížka - jedna komponenta zabere jen jedno k políčko.

GrigBag - nepravidelná mřížka - jedna komponenta zabere i více políček.

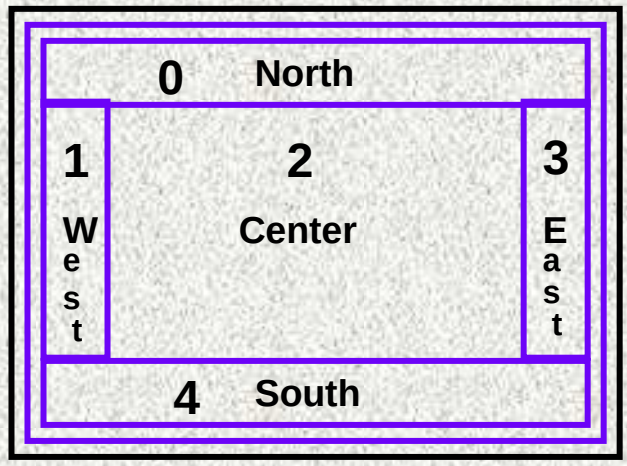
null - určí programátor pomocí setBounds(x, y, w, h).

java.awt: Layouts

Čísla vyjadřují index komponenty v seznamu.

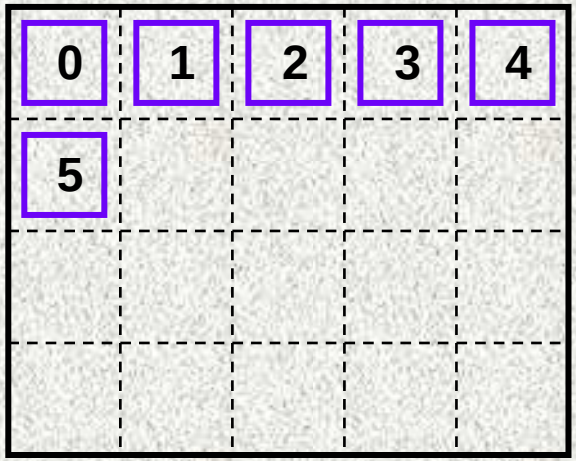


FlowLayout

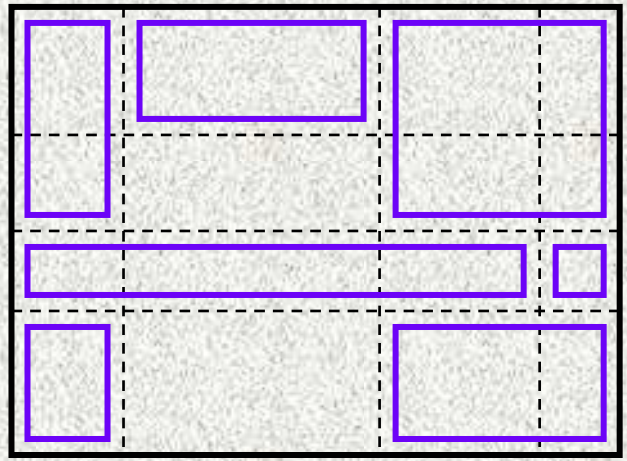


BorderLayout

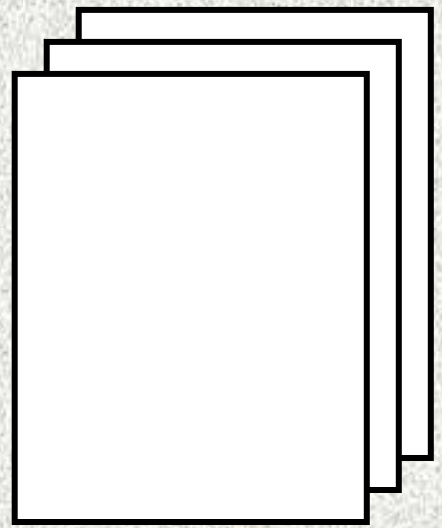
BorderLayout má pět oblastí. V každé zobrazí je jednu komponentu – poslední z přidělených do oblasti. Centrální oblast je vzadu – event. překrytá ostatními.



GridLayout



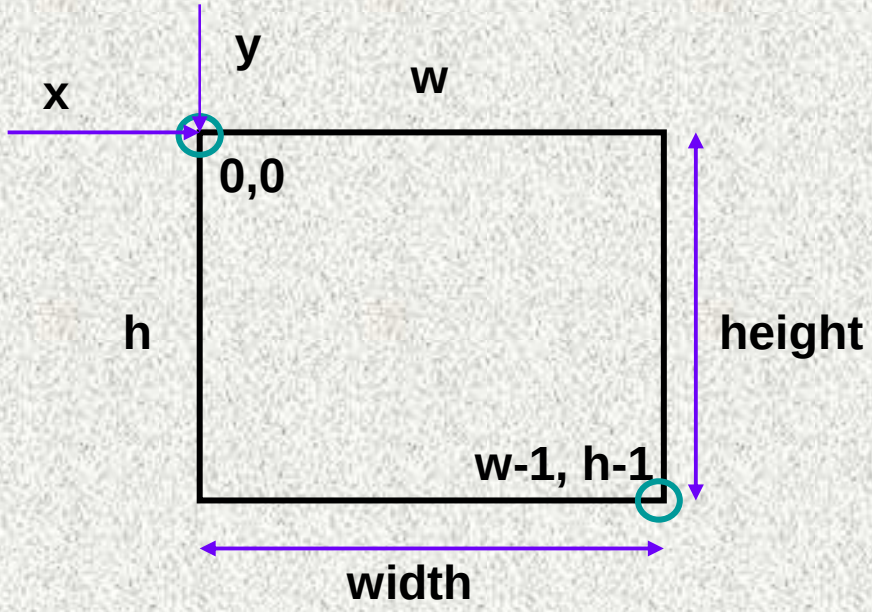
GridBagLayout



CardLayout

Metrika

Vizuální komponenty a displej se rozměřují v pixelech takto:



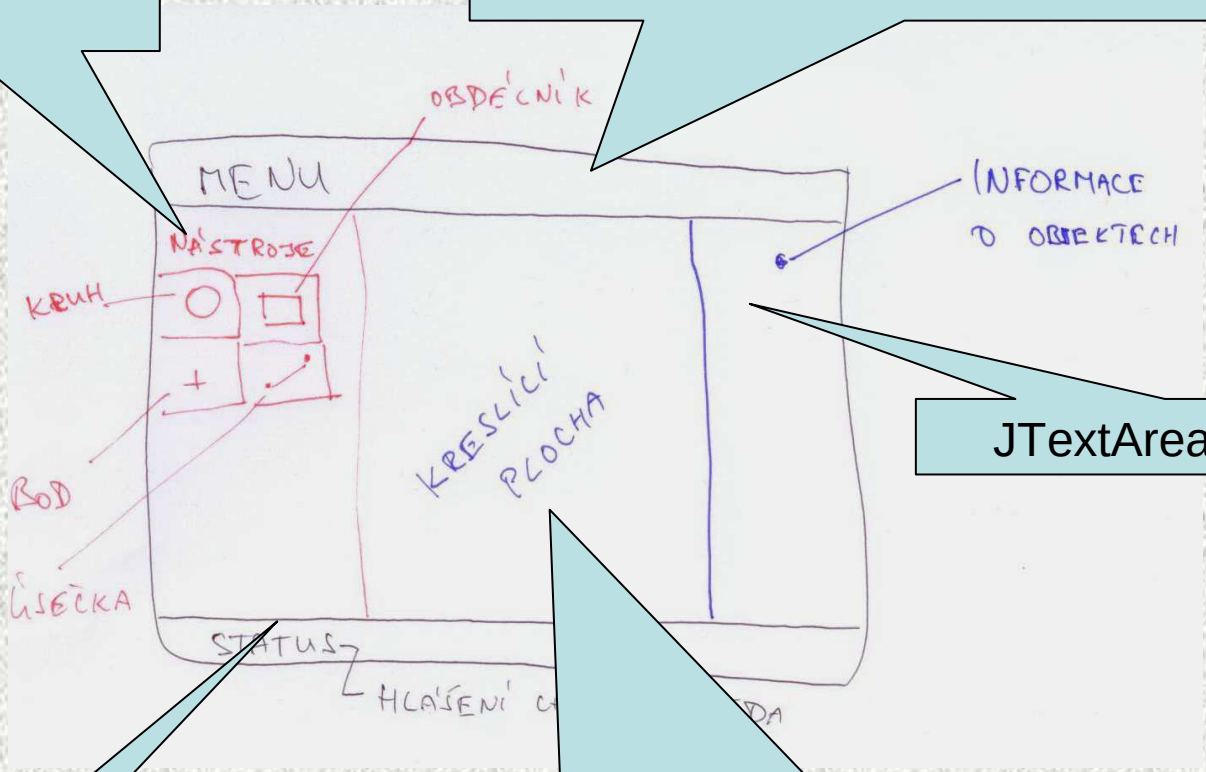
Tyto parametry se zadávají ve čtveřici či dvojici vždy takto (příklad):

setLocation(x, y)	new Point(x, y)	move (x, y)
setSize(w, h)	new Dimension(w, h)	
setBounds(x, y, w, h)	new Rectangle(x, y, w, h)	drawOval(x, y, w, h)

Grafický návrh vektorového kreslítka

Menu
soubor – nový, otevřít, uložit, exportovat, konec ...

ToolBar

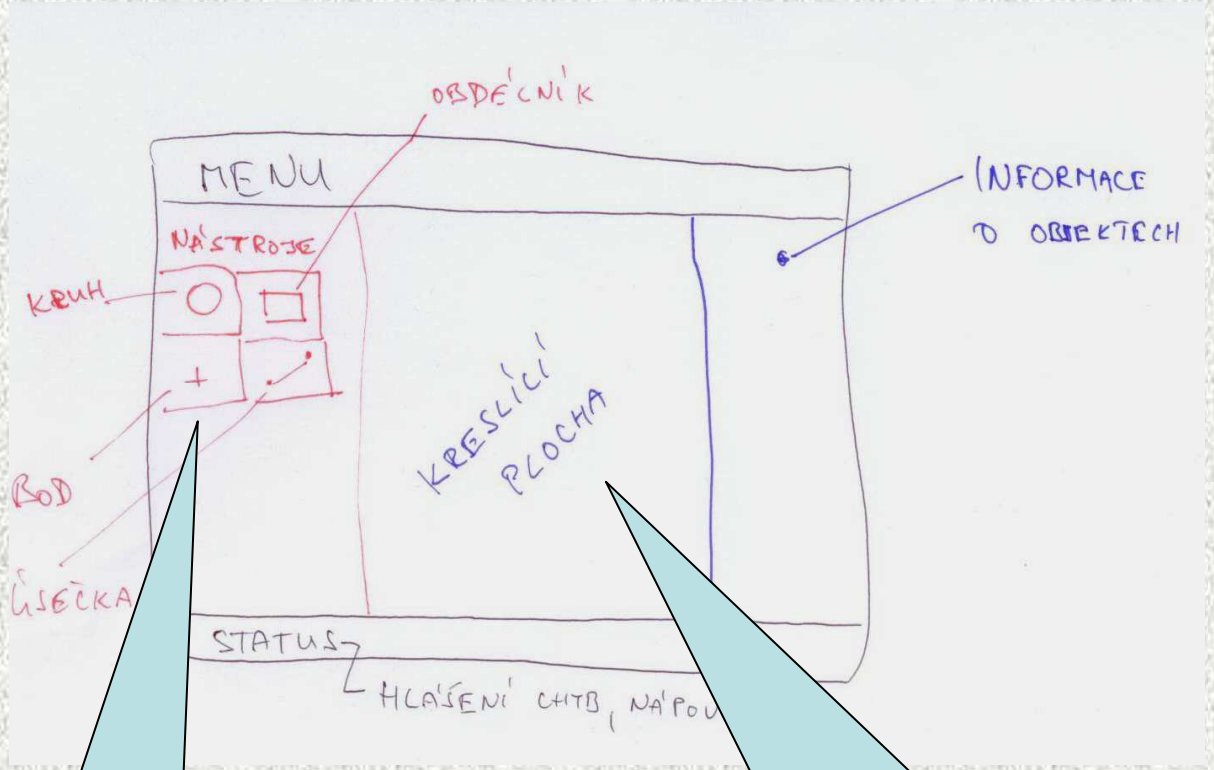


JTextArea ?? JTable

JLabel

Vlastní komponenta, vlastní chování na kliknutí, vlastní vykreslení, ...

Grafický návrh vektorového kreslítka



FlowLayout či
GridLayout

BorderLayout
zajistí vykreslení všech částí, největší
zbytek bude pro střed

Vytvoření a otevření hlavního okna

```
public class JFrameHlavniOkno extends JFrame {
public static void main(String args[]) {
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
   new JFrameHlavniOkno().setVisible(true);
                }
    });
   }
}
```

Otevření okna, anonymní třída, otevření zařazeno do fronty, Okno se umí zvětšit, zmenšit, posunovat, zavřít

Vložení komponent

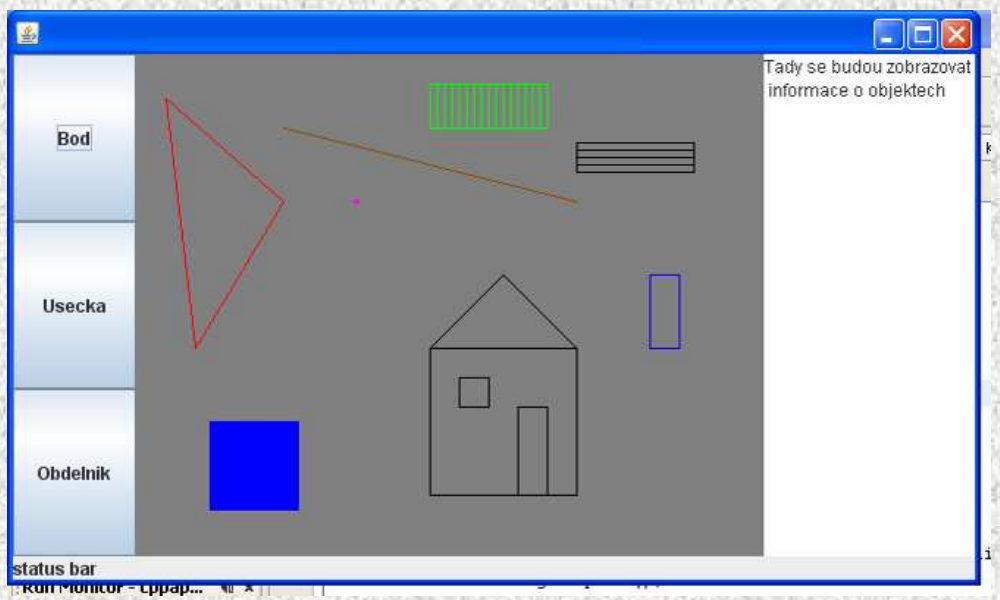
```
public class JFrameHlavniOkno extends JFrame {
    private JPanel nastroje = new JPanel();
    private KresliciPlocha plocha = new KresliciPlocha();
    private JLabel statusBar = new JLabel("status bar");
    private JTextArea text = new JTextArea("Tady se budou zobrazovat \n
    informace o objektech");
    public JFrameHlavniOkno(){
          Container c = this.getContentPane();
          nastroje.setLayout(new GridLayout(3, 1));
          c.setLayout(new BorderLayout());
          JToggleButton b = new JToggleButton("Bod"); nastroje.add(b);
           JToggleButton u = new JToggleButton("Usecka"); nastroje.add(u);
           JToggleButton o = new JToggleButton("Obdelnik"); nastroje.add(o);
           ButtonGroup tlacitka = new ButtonGroup();
           tlacitka.add(b); tlacitka.add(u); tlacitka.add(o);
           c.add(nastroje,BorderLayout.WEST);
          c.add(statusBar,BorderLayout.SOUTH); c.add(text,BorderLayout.EAST);
    c.add(plocha, BorderLayout.CENTER);
}
```

Tlačítka, vložena do panelu **nastroje,** přidány do ButtonGroup, aby bylo vybráno vždy právě jedno

Vytvoření a otevření hlavního okna II

```
public static void main(String args[]) {
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
JFrameHlavniOkno jfr = new JFrameHlavniOkno();
jfr.pack();
jfr.setVisible(true);
      }
  });
}
```

Zabalit = zmenšit na velikost, aby všechny komponenty byly vidět.



Nástroje lépe

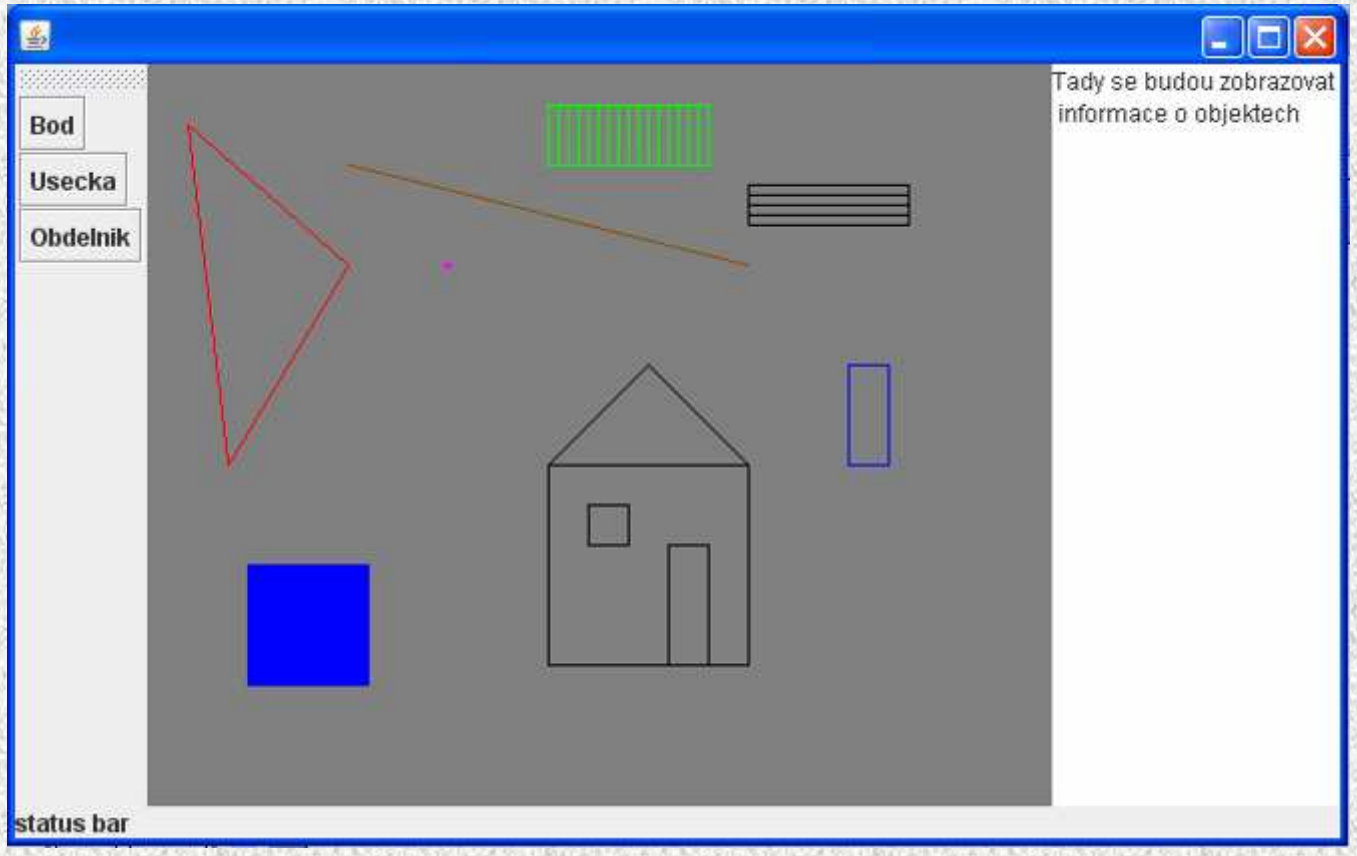
- Použijeme JToolbar

```
JToggleButton b = new JToggleButton("Bod");
  JToggleButton u = new JToggleButton("Usecka");
  JToggleButton o = new JToggleButton("Obdelnik");
ButtonGroup tlacitka = new ButtonGroup();
tlacitka.add(b); tlacitka.add(u); tlacitka.add(o);
JToolBar tb = new JToolBar("Nástroje pro malování");
tb.add(b);
tb.add(u);
tb.add(o);
```

...

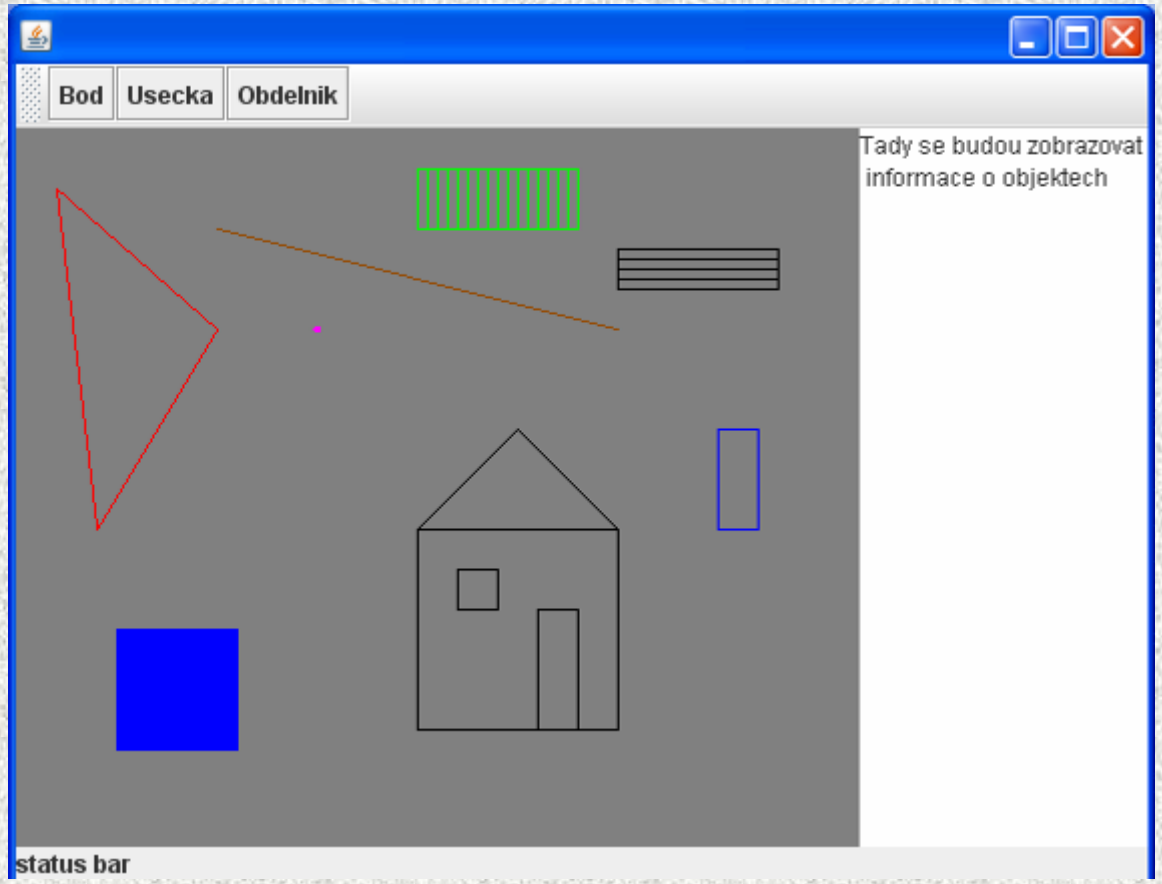
vytvořili jsme dokovatelné okno, které lze „vytáhnout z aplikace“

Nástroje lépe - ukázka



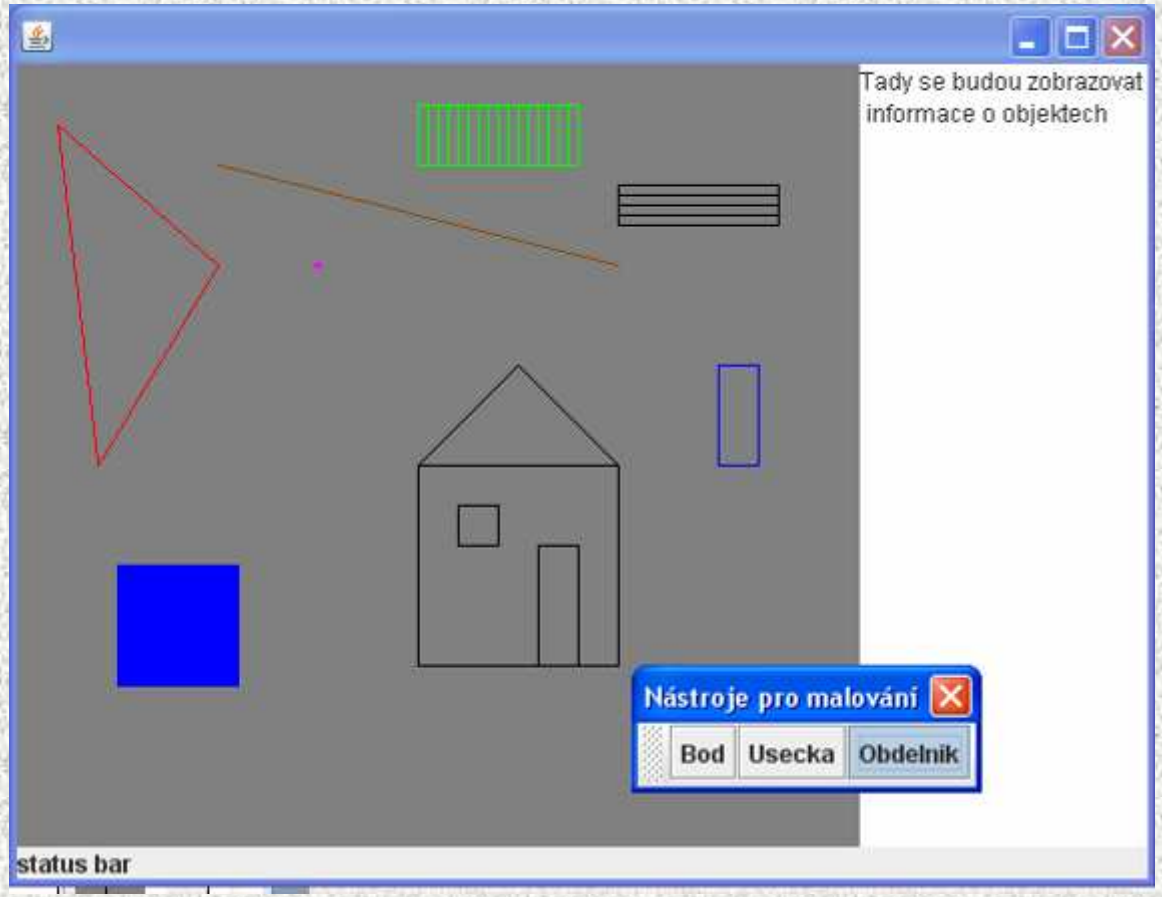
Základní pozice

Nástroje lépe - ukázka



Nástroje můžeme přesunout nahoru ...

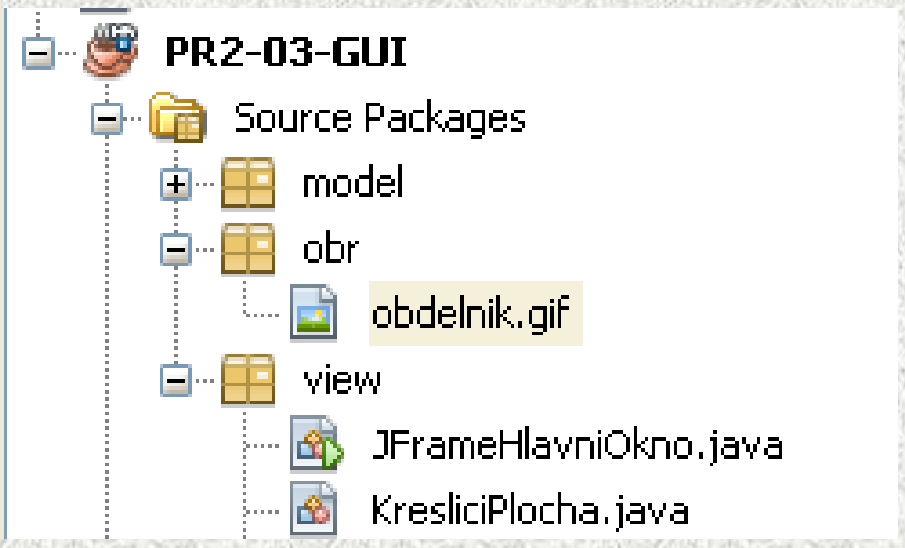
Nástroje lépe - ukázka

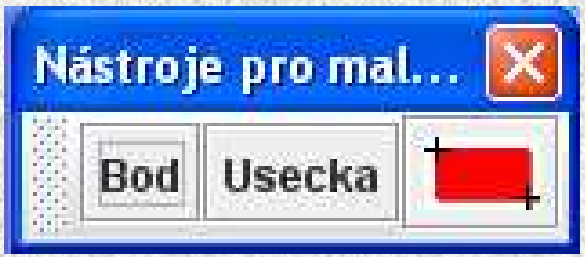


... nebo je vyndat z aplikace. a nemusíme nic dalšího programovat !!

Obrázek na tlačítko místo popisku

1. Připravíme si obrázek ve formátu ico, gif, png,... Použijeme například Gimp nebo malování. Velikost volíme s ohledem na požadované umístění
2. Obrázek vložíme do projektu, nejlépe do adresáře s obrázky
3. Dopíšeme vložení příslušné ikonky na obrázek, zde bez textu (lze text, text + ikonka, ikonka)





Obrázek na tlačítko místo popisku

Zdrojový kód:

```
Icon ikonka = new
    ImageIcon(getClass().getResource("/obr/obdelnik.gif"
    )); JToggleButton o = new JToggleButton(ikonka);
```

- lze zadat přímo cestu k souboru, ale způsobilo by to problémy s distribucí programu, takto vytvoříme jeden jar a je to (jar – bude později...)

JLabel a JToolTipText

- JLabel má mnohem významnější úlohu než java.awt.Label.

Do JLabel lze vložit text i ikonu a nastavit jejich vzájemnou horizontální a vertikální polohu.

```
Icon ic = new ImageIcon( "images/duke.gif" );
JLabel la = new JLabel( "This is Duke" , ic,
                        SwingConstants.LEADING );
```

JLabel má pozadí defaultně transparentní, tj. opacity je false.
To lze ovládat metodou setOpacity(boolean opacity).

- JToolTipText je vysvětlivkou, kterou lze vybavit každou instanci třídy JComponent a to metodou:

```
setToolTipText( String text )
```

instanci JTabbedPane pak metodou:

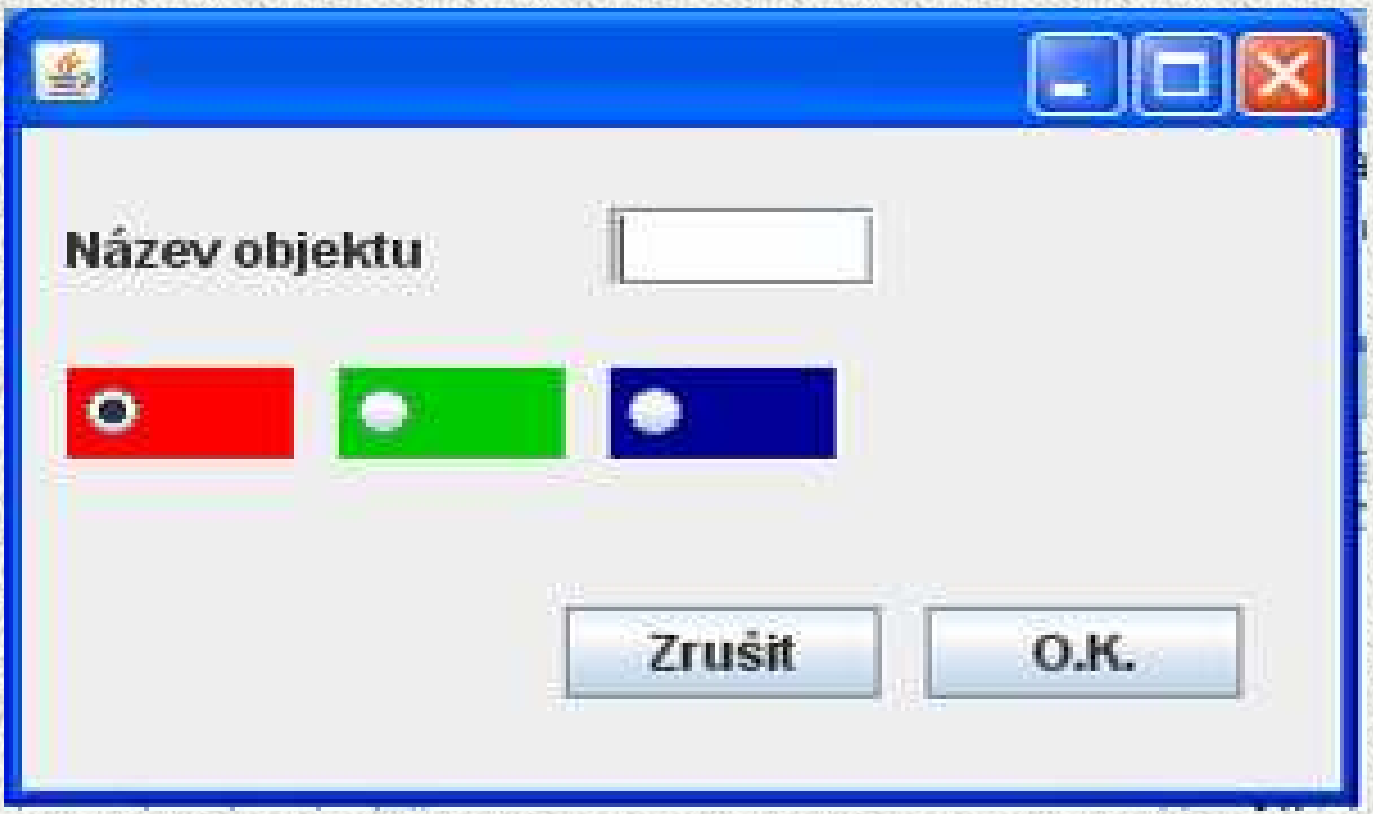
```
setToolTipText( String text, int index )
```

JToolTipText - příklad

- Přidáme bublinkovou nápovědu k nástroji obdélník:

```
JToggleButton o = new JToggleButton(ikonka);
o.setToolTipText("Tento nástroj umožňuje
kreslení                    obdélníků");
```

Okno pro nastavení vlastností GrO



Okno pro nastavení vlastností GrO

```
public class JFrameEditObjs extends JFrame{
   JTextField jTextField1 = new JTextField();
   JLabel jLabel1 = new JLabel("Název objektu");
   JButton jButtonOK = new JButton("O.K.");
   JButton jButtonZrus = new JButton("Zrušit");
   JRadioButton jRadioButton1 = new JRadioButton();
   JRadioButton jRadioButton2 = new JRadioButton();
   JRadioButton jRadioButton3 = new JRadioButton();
```

... deklarace složek

Okno pro nastavení vlastností GrO

```
public JFrameEditObjs() {
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);
        getContentPane().add(jTextField1);
        jTextField1.setBounds(130, 20, 59, 20);
        getContentPane().add(jLabel1);
        jLabel1.setBounds(10, 20, 80, 20);
        getContentPane().add(jButtonOK); ...
        jRadioButton1.setBackground(Color.RED);
        getContentPane().add(jRadioButton1); ...
        jRadioButton1.setBounds(10, 60, 50, 23);
        ButtonGroup bg = new ButtonGroup();
bg.add(jRadioButton1);bg.add(jRadioButton2);bg.add(jRadioButton3);
        jRadioButton1.setSelected(true);//implicitní barva
        this.setSize(300,200);
}
```

absolutní pozicování

panel pro kreslení

vložení do okna

x,y,šířka, výška

tento bude vybrán

nastavení velikosti okna

Kreslení na plátno – třída Graphics

Tento objekt umožňuje v objektech typu Component a tedy i Canvas, Panel, Applet, Window, Frame atd. vytvořit a upravovat kresby, texty a obrázky.

Přístup k němu se získá přepsáním zděděné prázdné metody např. takto:

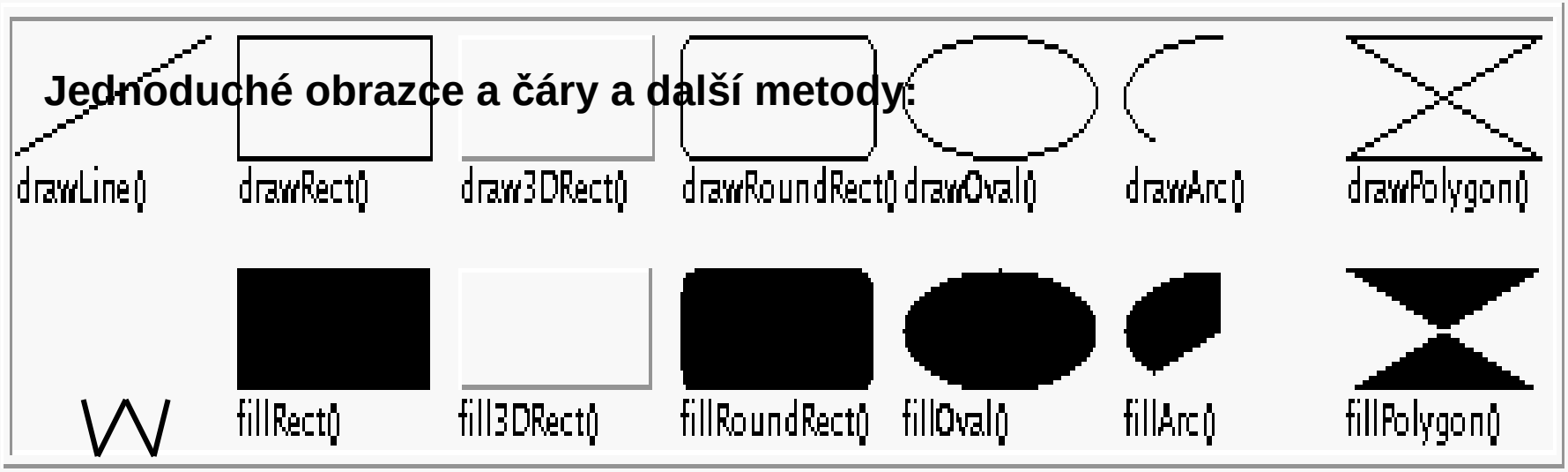
```
public void paint( Graphics g ) {
g.drawXYZ( ... );  // kreslí obrys obrazce v barvě BLACK
g.setColor( c1 );                  // nastaví pero na barvu c1
g.fillXYZ( ... );                     // kreslí plný obrazec
g.setFont( f );                     // nastaví font písma
g.setColor( c2 );                  // nastaví pero na barvu c2
g.drawString( s, ... );// nakreslí text dle fontu barvy c2
g.drawImage( im,...);// vykreslí obrázek {.gif,.jpg,.png }
      }
```

Případně též metodou **`Graphics getGraphics( ),`** která vytvoří grafický kontext pro skrytý bufer (off-screen image).

To využívá technika tzv. double–bufferingu pro hladší animace.

Graphics

Jednoduché obrazce a čáry a další metody:



drawPolyline()

- clearRect(…) - přemalování na barvu pozadí dle setBackground(…)
- clipRect(…), getClip( ), setClip( ), ... - vytřihovánky a nalepovánky
- copyArea(…) - kopírování plošky
- setFont(…), getFont( ) - práce s fonty
- getFontMetrics( ) - měření nápisů
- dispose( ) - uvolnění zdrojů

Kreslení grafických objektů, potomků GrO

```
public class Obdelnik extends O2D {
...
public void kresliSe(Graphics g) {
Color c = g.getColor(); //uloz si barvu pera
g.setColor(barva); //nastav svou barvu
g.drawRect(pozice.x, pozice.y, sirka, vyska);
switch (getSrafovani()) {
 case VODOROVNEPRUHY: for (int y = pozice.y; y < pozice.y + vyska; y
   += 5) { g.drawLine(pozice.x, y, pozice.x + sirka, y); } break;
 case SVISLEPRUHY: for (int x = pozice.x; x < pozice.x + sirka; x +=
   5) { g.drawLine(x, pozice.y, x, pozice.y + vyska); } break;
 case PLNE: g.fillRect(pozice.x, pozice.y, sirka, vyska); break; }
   g.setColor(c); //obnov barvu
    }
}
```

- metoda kresliSe je definována v každém potomkovi GrO

Seznam kreslených obrázků

- Reprezentován třídou Model
- zajišťuje vykreslení a evidenci objektů typu GrO – využíváme polymorfismu

```
public class Model {
                /*objekty budeme vkládat do kolekce*/
private List<GrO> seznamObjektu = new ArrayList<GrO>();

public void addGrO(GrO o){ seznamObjektu.add(o); }

public void kresliGrObjekty(Graphics g){
 for (GrO grO : seznamObjektu) {
            grO.kresliSe(g);
      }
  }
}
```

Každý vložený objekt ví, jak se vykreslit

Color

Barvířství je kumšt, neb lidské oko rozeznává asi 6000 barevných odstínů.
Třída Color poskytuje teoreticky 16777216 barev a řiditelnou průhlednost.

Konstanty definují 13 standardních barev:

```
{ BLACK, BLUE, CYAN, DARK_GRAY, GREEN, GRAY, LIGHT_GRAY,
MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW  }.
```

Všechny barvy v modelu RGB vytvoří konstruktory:

```
Color (int red, int green, int blue, int alpha )
   int  = 0 .. 255
Color (float red, float green, float blue, float alpha )
   float = 0.0 .. 1.0
```

kde: red, green, blue je síla barevných složek,
alpha = 255 je úplná opacita (neprůhlednost),
alpha = 0 je úplná transparence (průhlednost).

Metody darker( ) resp. brighter( ) vytvářejí novou barvu tmavějšího resp. světlejšího odstínu zadané barvy – avšak s úplnou opacitou(=neprůhlednost).

Informace

AbstractButton

Koncept knoflíků a příbuzných součástek je nově strukturován.

Strom třídy AbstractButton obsahuje (viz schémata):

- JButton,
- JToggleButton,
- JCheckBox,
- JRadioButton – spolupracuje s ButtonGroup,
- JMenuItem,
- JMenu,
- JCheckBoxMenuItem,
- JRadioButtonMenuItem.

Tyto třídy mají i konstruktor s parametrem Action, který podporuje elegantní použití handleru.

Metody paint a print

Ve třídě JComponent jsou přepsány a připsány metody pro kreslení a tisk.

`public void paint( Graphics g )` – postupně volá následující tyto tři metody:

```
public void paintComponent( Graphics g )
public void paintBorder( Graphics g )
public void paintChildren( Graphics g )
```

Defaultně se tedy vykresluje naspod grafika, pak okraje a navrch přes sebe komponenty v pořadí udaném v java.awt.Containeru.
Kterákoliv metoda může být přepsána a v metodě paint může být změněno pořadí vykreslování.

Obdobně je řešen tisk metodami:
`printComponent, printBorder a printChildren.`

Text

Třída javax.text.JTextComponent zajišťuje služby pro své textové potomky:

- Model: document
- View: pro visualizaci
- Controller: editor kit
- Podpora undo/redo
- Caret, listenery a filtry

Přímými potomky jsou třídy:

- JTextField
- JTextArea
- JEditorPane

Pomocí tzv. stylů a atributů lze vytvořit pěkné texty.
Navazují podbalíčky pro rtf a html.

Lištu JMenuBar lze vložit do JFrame, JRootPane, JApplet, JDialog a JInternalFrame – nikoli však do JWindow. Lišta má být opacitní.

javax.swing.table

```
Object arr [ ][ ] = new Object [8] [3]  ;
String [ ] names = { "AAA", "BBB", "CCC", };          //
  column names
TableModel tm = new DefaultTableModel( arr,  names  );
JTable  ta = new  JTable( );
ta.setModel( tm );
JScrollPane sp = new JScrollPane( );
sp.setViewportView( ta );
Container co = this.getContentPane( );
co.add( sp );
...
Object x = tm.getValueAt( i, j );
tm.setValueAt( "XYZ" , i, j );
```