Java v rytme Swingu

(vývoj okienkových aplikácií)

Róbert Novotný
robert.novotny@upjs.sk

Layout managery

- umiestňovanie komponentov do formulára na základe pixelovej polohy má mnoho nevýhod
- portabilita medzi prostrediami klesá:
  - GTK (Gnome/Linux) má implicitne oveľa väčšie ovládacie prvky než Windows
- problémy s prekladom a lokalizáciou
  - ak na "Box" stačí 20 pixelov, na "Škatuľa" nie
  - prostredia píšuce sprava doľava vyžadujú nové UI
- vytváranie rozhrania je štýlom "umiestni komponent-skontroluj-prispôsob polohu-skontroluj", čo je pomalé a neefektívne

Layout managery

- layout manager (správca rozloženia) je algoritmus, ktorý automaticky či poloautomaticky umiestňuje komponenty do kontajnera
- existuje mnoho algoritmov
  - niektoré "blbé",
  - niektoré hyperinteligentné

FlowLayout

- jednoduchý, ale jednoduchý layout manager
- analógia: sádzanie znakov do odseku
  - sádžeme znaky, kým máme miesto na riadku, ak nie, prejdeme na nový riadok
- komponenty ukladá zľava doprava, pokiaľ sa zmestia
  - alebo sprava doľava (arabské, izraelské... prostredie)
- v „odseku" nastaviť zarovnávanie (vľavo, stred, vpravo)
- komponent má veľkosť z nastavení `setPreferredSize()`

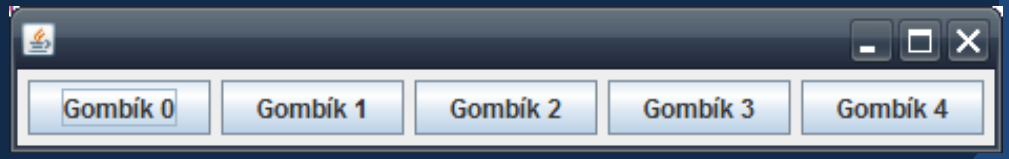
FlowLayout

- jednoduchý, ale jednoduchý layout manager
- analógia: sádzanie znakov do odseku
  - sádžeme znaky, kým máme miesto na riadku, ak nie, prejdeme na nový riadok
- komponenty ukladá zľava doprava, pokiaľ sa zmestia
  - alebo sprava doľava (arabské, izraelské... prostredie)
- v „odseku" nastaviť zarovnávanie (vľavo, stred, vpravo)
- komponent má veľkosť z nastavení `setPreferredSize()`

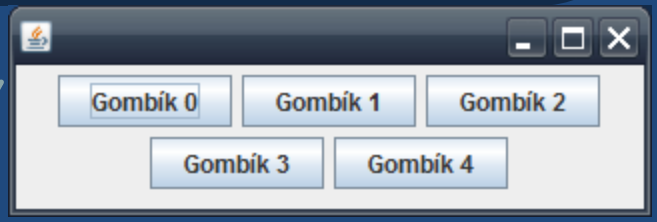
FlowLayout

```
public class FlowLayoutFrame extends JFrame {
  public FlowLayoutFrame() {
    setLayout(new FlowLayout());

    for(int i = 0; i < 5; i++) {
      add(new JButton("Gombík " + i));
    }

    pack();
}
```

vytvoríme 5 gombíkov



prispôsobí veľkosť okna komponentom

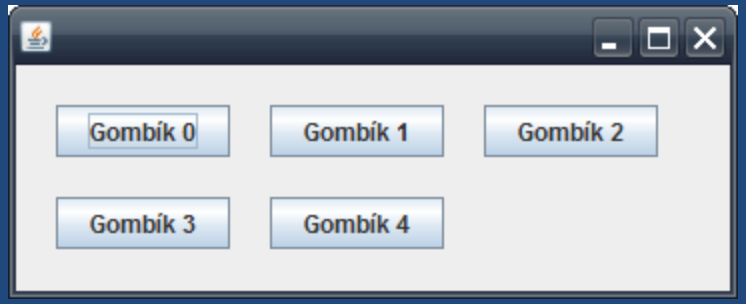


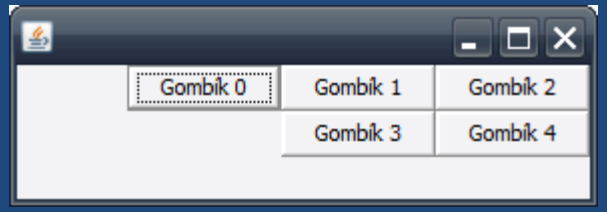
zmena veľkosti okna preusporiada komponenty

FlowLayout

- možno meniť rozostup medzi komponentami a zarovnanie

```
public FlowLayoutFrame() {
    setLayout(new FlowLayout(FlowLayout.LEFT, 20, 20));
}
```






```
public FlowLayoutFrame() {
    setLayout(new FlowLayout(FlowLayout.RIGHT, 0, 0));
}
```

FlowLayout

- možno meniť rozostup medzi komponentami a zarovnanie

```
public static void main(String[] args) {
  UIManager.setLookAndFeel(new NimbusLookAndFeel());

  DemoFrame okno = new DemoFrame();
  okno.setVisible(true);
}
```

BorderLayout

- kontajner je rozdelený na 5 sektorov

- každý sektor môže obsahovať najviac 1 komponent
- sever a juh sa naťahujú do šírky
- západ a východ do výšky
- stred v oboch smeroch

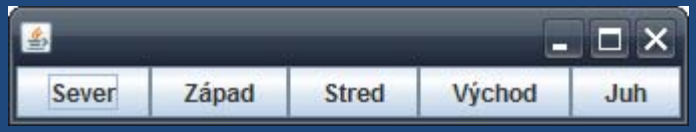
BorderLayout

```
setLayout(new BorderLayout());

add(new JButton("Sever"), BorderLayout.PAGE_START);
add(new JButton("Západ"), BorderLayout.LINE_START);
add(new JButton("Stred"), BorderLayout.CENTER);
add(new JButton("Východ"), BorderLayout.LINE_END);
add(new JButton("Juh"), BorderLayout.PAGE_END);

pack();
```

- metóda add() tu má 2 parametre:
  - komponent ukladaný do kontajnera
  - koordinát určujúci umiestnenie

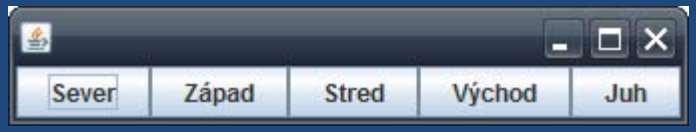
BoxLayout

- veľmi podobný flow layoutu
- komponenty ukladá
  - buď horizontálne - vedľa seba (LINE_AXIS)
  - alebo vertikálne – pod seba (PAGE_AXIS)
- v prípade nedostatku miesta však nezalamuje na "nový riadok"



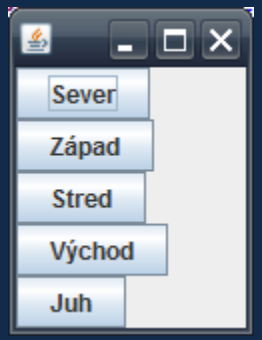
BoxLayout

- veľmi podobný flow layoutu
- komponenty ukladá
  - buď horizontálne - vedľa seba (LINE_AXIS)
  - alebo vertikálne – pod seba (PAGE_AXIS)
- v prípade nedostatku miesta však nezalamuje na "nový riadok"



BoxLayout

```
BoxLayout = new BoxLayout(getContentPane(),
                          BoxLayout.PAGE_AXIS)
setLayout(layout);

add(new JButton("Sever"));
add(new JButton("Západ"));
add(new JButton("Stred"));
add(new JButton("Východ"));
add(new JButton("Juh"));

pack();
```

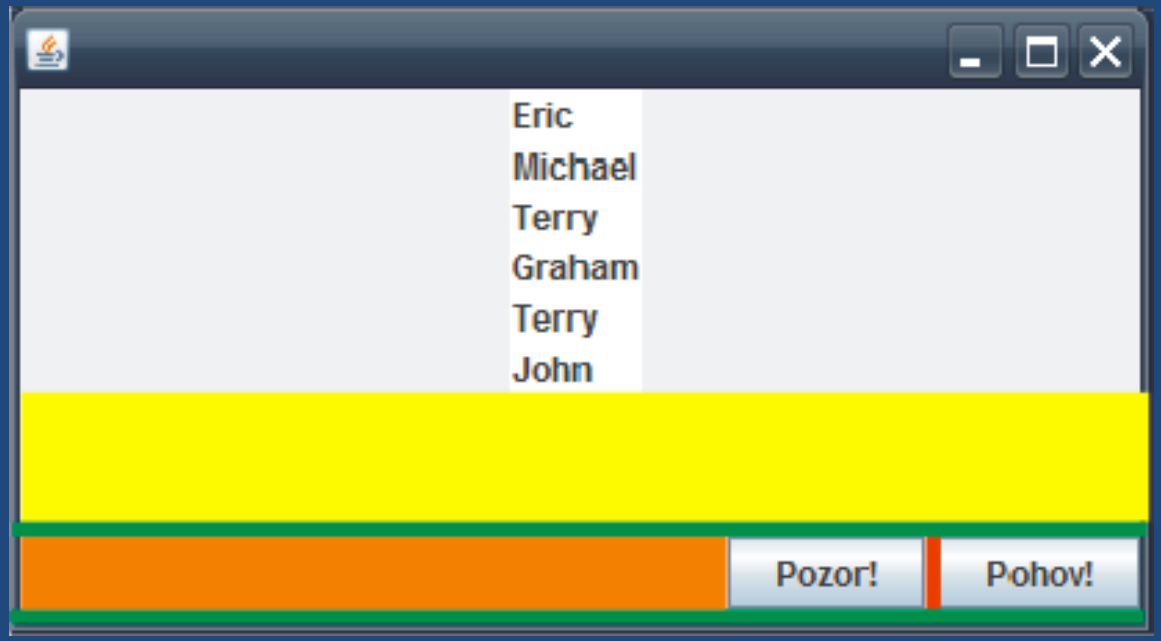


- šírka komponentov sa vezme z *preferredSize*
- komponenty sa hádžu pod seba

BoxLayout

```
setLayout(new BoxLayout(getContentPane(), BoxLayout.PAGE_AXIS));

add(new JList(new String[] {"Eric", "Michael", "Terry", "Graham",
                            "Terry", "John"}));
// guma, ktorá sa rozťahuje, aby vyplnila miesto
add(Box.createVerticalGlue());
// pevný box (oddeľovač)
add(Box.createRigidArea(new Dimension(0, 5)));

JPanel buttons = new JPanel();
buttons.setLayout(new BoxLayout(buttons, BoxLayout.LINE_AXIS));
add(buttons);

// guma drží gombíky pri pravom okraji okna
buttons.add(Box.createHorizontalGlue());
buttons.add(new JButton("Pozor!"));
// oddeľovač gombíkov
buttons.add(Box.createRigidArea(new Dimension(5, 0)));
buttons.add(new JButton("Pohov!"));
// oddeľuje gombíky od spodného okraja okna
add(Box.createRigidArea(new Dimension(0, 5)));
pack();
```

BoxLayout

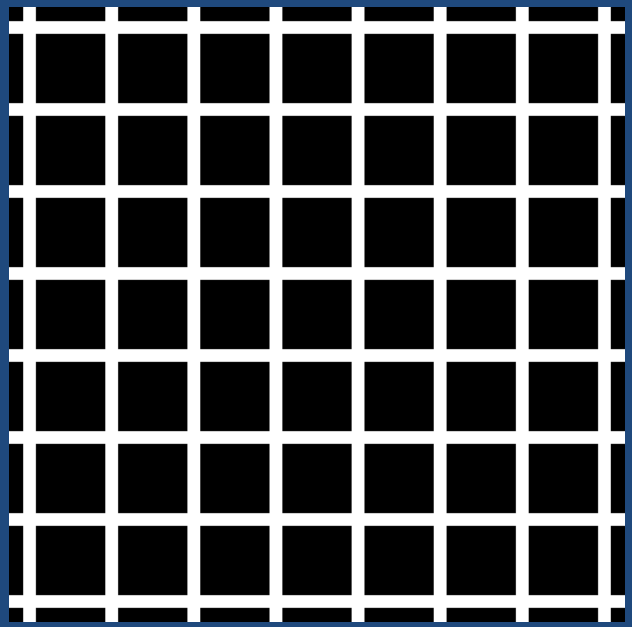


BoxLayout

- idea veľmi podobná LaTeXu
- rigid area – pevný box danej šírky a výšky
- glue – lepidlo. Rozťahuje sa tak, aby vyplnilo celú ostávajúcu šírku
- strut – rozpora. Box danej šírky alebo dĺžky.

GridLayout

- ukladá komponenty do mriežky
  - tá má daný počet riadkov a stĺpcov
  - zapĺňa celý kontajner
  - ignorujú sa preferované, minimálne i maximálne veľkosti komponentov



GridLayout

```
setLayout(new GridLayout(4, 3));
for (int i = 1; i <= 9; i++) {
  add(new JButton(Integer.toString(i)));
}
add(new JButton("*"));
add(new JButton("0"));
add(new JButton("#"));

pack();
```

- parametre konštruktora: 4 riadky, 3 stĺpce
- ak je viac komponentov než buniek mriežky, počet riadkov ostáva, doplnia sa nové stĺpce

GridLayout

- môžeme nadefinovať rozostupy medzi komponentami v konštruktore

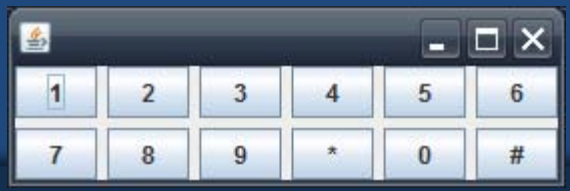
```
setLayout(new GridLayout(4, 3, 5, 5));
```

- 0 znamená ľubovoľný počet komponentov

```
setLayout(new GridLayout(0, 2, 5, 5));
```

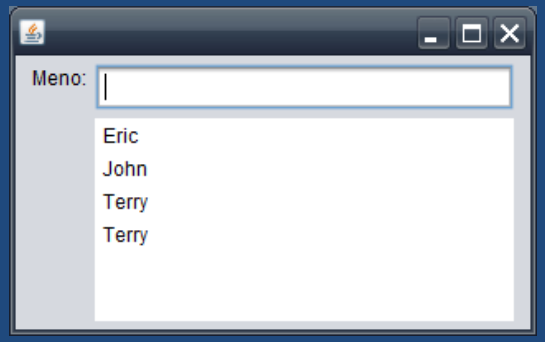
- 2 riadky krát X stĺpcov

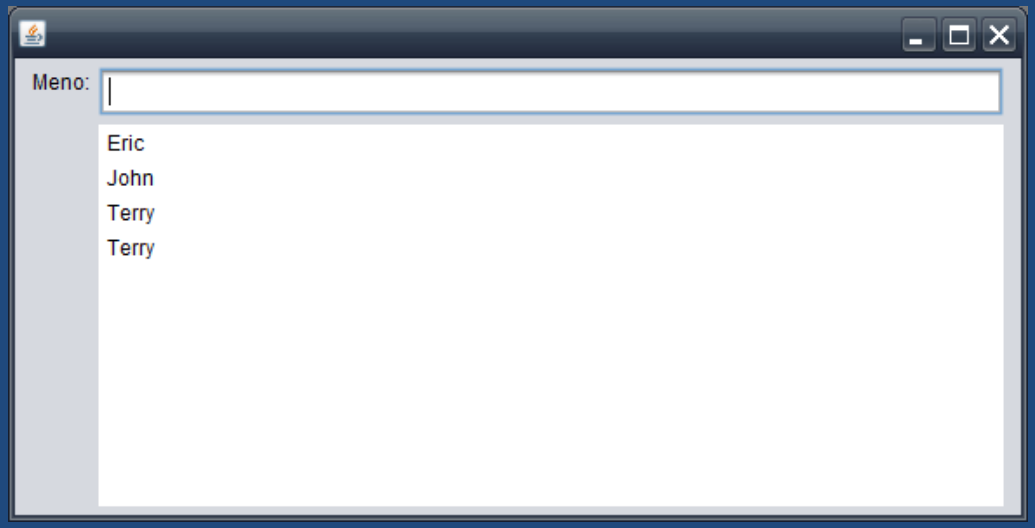
```
setLayout(new GridLayout(2, 0);
```



SpringLayout

- pomerne zložitý na ručné kódenie
- komponenty rozhadzuje na základe špecifikácie vzťahov medzi nimi
- príklad: horný okraj je zarovno s horným okrajom okna

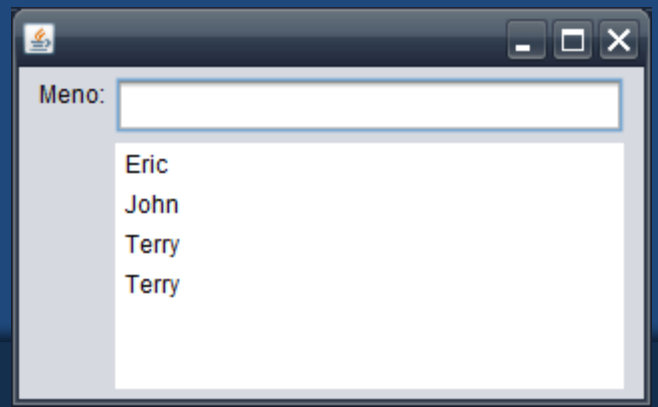




SpringLayout

```
SpringLayout layout = new SpringLayout();

JPanel panel = new JPanel();
panel.setLayout(layout);
panel.setBorder(
      BorderFactory.createEmptyBorder(5, 5, 5, 5));
add(panel);
```

- komponenty budeme ukladať do panelu
- panelu nastavíme okraj, aby komponenty mali "vzduch"



SpringLayout

```
SpringLayout layout = new SpringLayout();
setLayout(layout);

JLabel lblMeno = new JLabel("Meno:");
layout.putConstraint(SpringLayout.WEST, lblMeno,
                     5,
                     SpringLayout.WEST, getContentPane());
add(lblMeno);
```

- koordináty komponentov špecifikujeme cez putConstraint()
- kód: západná (*west*) strana labelu má byť vždy 5 pixelov vpravo od západnej (*west*) strany okna

SpringLayout

```
JTextField txtMeno = new JTextField();
layout.putConstraint(SpringLayout.WEST, txtMeno,
                     5, SpringLayout.EAST, lblMeno);
layout.putConstraint(SpringLayout.EAST, getContentPane(),
                     5, SpringLayout.EAST, txtMeno);
add(text);
```

- západný okraj textfieldu má byť 5 pixelov od východného okraja labelu
- východná strana textfieldu má byť 5 pixelov od východnej strany okna

SpringLayout

```
String[] mená = new String[] {"Eric", "Graham", "Michael",
                              "John", "Terry", "Terry"};
JList list = new JList();
layout.putConstraint(SpringLayout.NORTH, list, 5,
                     SpringLayout.SOUTH, text);
layout.putConstraint(SpringLayout.WEST, list, 0,
                     SpringLayout.WEST, text);
layout.putConstraint(SpringLayout.EAST, list, 0,
                     SpringLayout.EAST, text);
layout.putConstraint(SpringLayout.SOUTH, list, -5,
                     SpringLayout.SOUTH, getContentPane());
add(list);
```

- sever zoznamu je 5 px od juhu textboxu
- západ zoznamu sa drží zarovno so západom textboxu
- východ zoznamu zarovno s východom textboxu
- juh zoznamu je 5 px nad juhom okna

GridBagLayout

- layout do flexibilnej mriežky
- podobný GridLayout-u
- jednotlivé riadky [stĺpce] môžu mať nezávislé šírky [výšky]
- pre každú bunku vieme nastaviť:
  - horizontálne a vertikálne naťahovanie
  - padding: rozostup od ostatných komponentov
  - koľko riadkov [stĺpcov] zaberá (span)
  - umiestnenie komponentu do bunky v duchu BorderLayoutu

GridBagLayout

- koordináty určované v triede GridBagConstraints
- inštancia sa používa pri pridávaní v add()
- atribúty:
  - gridx, gridy: koordináty v mriežke. [0, 0] = vľavo hore
  - gridwidth, gridheight: počet buniek, ktoré komponent zaberá (štandardne 1)
    - hodnota GridBagConstraints.REMAINDER pre bunku tiahnúcu sa po poslednú bunku v riadku [stĺpci]
    - použite GridBagConstraints.RELATIVE pre bunku tiahnúcu sa po preposlednú bunku v riadku [stĺpci]
  - fill: ako sa správať ku komponentu, ktorý je menší než bunka
    - GridBagConstants.HORIZONTAL = rozťahovať do šírky
    - GridBagConstants.VERTICAL = rozťahovať do výšky
    - GridBagConstants.BOTH = rozťahovať do šírky i do výšky
    - GridBagConstants.NONE = nerobiť nič

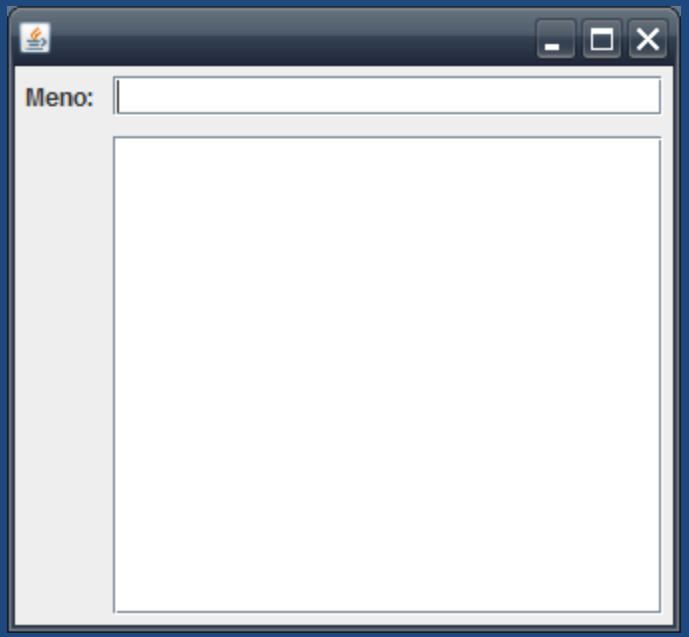
GridBagLayout – atribúty layoutu

- ipadx, ipady: interný rozostup. Šírka [výška] komponentu je aspoň minimálna šírka [výška] + ipadx [ipady].
- insets: minimálna veľkosť medzery medzi komponentom a okrajom bunky, v ktorej je.
- anchor: ukotvenie v prípade komponentov menších než bunka
  - konštanty v GridBagConstraints podobné ako v prípade BorderLayoutu.
- weightx, weighty: ako prerozdeľovať miesto v prípade naťahovania kontajnera. Číslo medzi 0-1.
  - Komponent s **weightx = 1** dostane 100% šírky medzery (polovicu vľavo, polovicu vpravo).
  - Komponent s **weightx = 0** nedostane žiadnu medzeru => bunka si zachováva rovnakú šírku.
  - pozor! aspoň 1 komponent v riadku musí mať nenulovú weightx a jeden v stĺpci nenulovú weighty. Inak budú komponenty centrované.

GridBagLayout

```
GridBagLayout layout = new GridBagLayout();
setLayout(layout);

GridBagConstraints c = new GridBagConstraints();
```

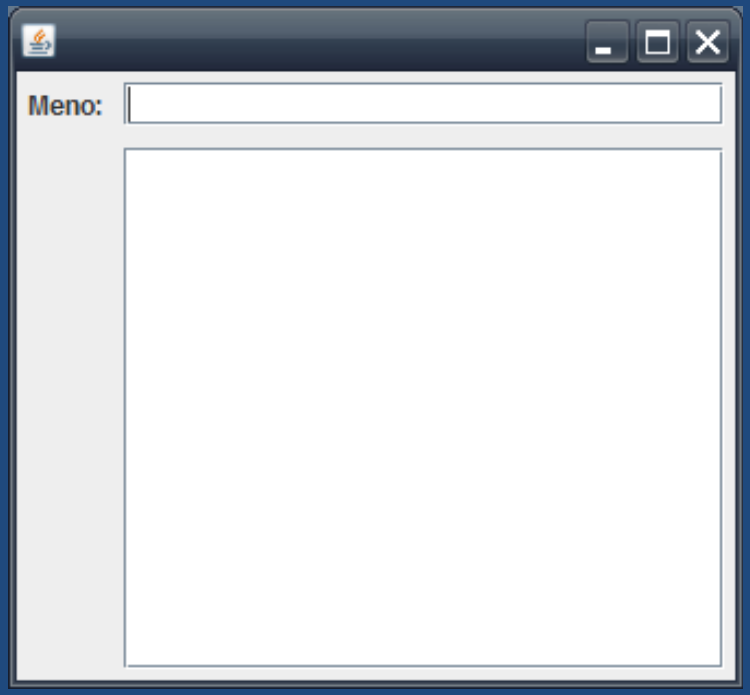
- špecifikácia obmedzení sa rieši v premennej typu GridBagConstraints
- stačí nám jediná inštancia, tú môžeme zrecyklovať pre viacero komponentov



GridBagLayout

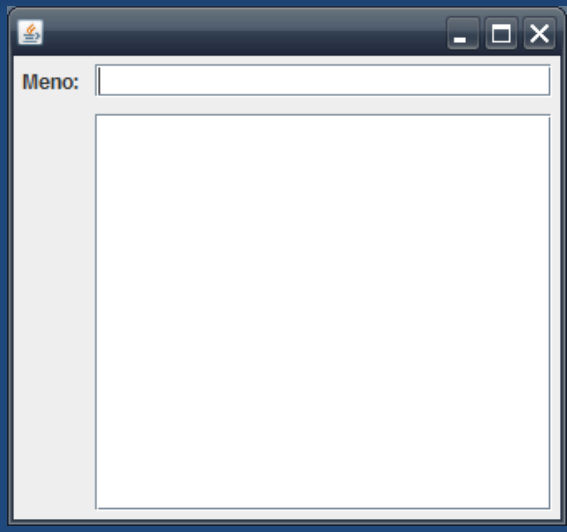
```
c.gridx = 0; c.gridy = 0;
c.fill = GridBagConstraints.NONE;
c.insets = new Insets(5, 5, 5, 5);
add(new JLabel("Meno:"), c);
```

- label *Meno* umiestnime do mriežky vľavo hore (0, 0)
- zakážeme mu meniť svoju veľkosť (fill = NONE)
- dodáme okolo neho 5px "vzduchu", aby nebol natlačený na okraj mriežky



GridBagLayout

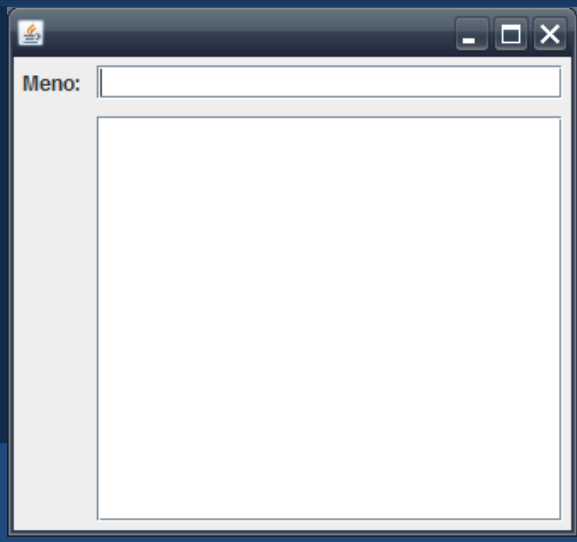
```
c.gridx = 1; c.gridy = 0;
c.fill = GridBagConstraints.HORIZONTAL;
c.weightx = 1;
add(new JTextField(), c);
```



- textové pole umiestnime do mriežky napravo od labelu (1, 0)
- veľkosť sa bude meniť horizontálne (fill = HORIZONTAL)
- 5px vzduch ostane z predošlého komponentu (premenná *c* si pamätá *insets*)
- weightx = 1: pri rozťahovaní do šírky dostane táto bunka všetok nový priestor.
- textfield sa však rozťahuje do nového priestoru
- label v prvom stĺpci má weightx = 0, nedostáva žiadny priestor.

GridBagLayout

```
c.gridx = 1; c.gridy = 1;
c.fill = GridBagConstraints.BOTH;
c.weightx = 1; c.weighty = 1;
add(new JTextField(), c);

setSize(new Dimension(640, 480));
```



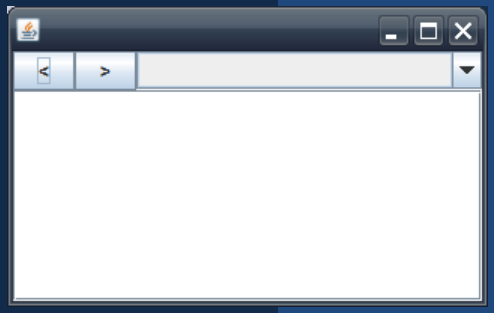
- listbox umiestnime do mriežky na [1, 1]
- rozťahuje sa v oboch smeroch (fill = BOTH)
- weightx = 1: v horizontálnom smere dostane jej bunka všetok nový priestor
- weighty = 1: vo vertikálnom smere dostane jej bunka všetok nový priestor => textbox nad ňou bude mať pevnú veľkosť
- insets sa prenesie v premennej ***c*** z labelu
- setSize() = nastaví veľkosť celého okna na príslušný rozmer

Kombinácia layout managerov

- layout managery možno aj kombinovať
- využívame hierachiu komponentov
- základný kontajner v okne je JPanel
- panelu vieme priradiť nezávislý layout manager
  - preddefinovaný je FlowLayout
- okno tak vieme vyskladať z panelov
- ak máme panely v samostatných triedach, môžeme ich znovupoužívať vo viacerých oknách (kompozícia z OOP!)

Kombinácia layout managerov

```
setLayout(new BorderLayout());

JPanel toolbar = new JPanel(new GridBagLayout());
add(toolbar, BorderLayout.NORTH);
GridBagConstraints c = new GridBagConstraints();

c.gridx = 0; c.gridy = 0; c.weightx = 0;
toolbar.add(new JButton("<"), c);

c.gridx = 1; c.gridy = 0; c.weightx = 0;
toolbar.add(new JButton(">"), c);

c.gridx = 3; c.gridy = 0; c.weightx = 1;
c.fill = GridBagConstraints.HORIZONTAL;
toolbar.add(new JComboBox(), c);

add(new JTextField());
```



panel s gombíkmi je na severe BorderLayoutu, má GridBagLayout, biela plocha je v strede BorderLayoutu