

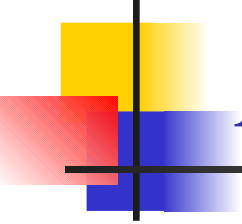


Grafický model



Swing vs. AWT

- Existují dvě základní sady komponent pro použití v Javě, které jsou standardně podporované:
 - **AWT (Abstract Window Toolkit):** Jedná se o prvotní návrh GUI pro Javu (verze 1). Pomalé, průměrný vzhled, obsahuje málo komponent. V současné době se používá pouze ve starých projektech, které jsou již implementovány a pouze se udržují
 - **Swing:** Nová sada komponent pro návrh GUI pro Javu (verze 2). Velké množství komponent, rychlejší než AWT, ...
- Můžete se ještě setkat s dalšími sadami komponent:
 - **SWT (Standard Widget Toolkit):** Jedná se o sadu komponent, na které je v současnosti postaven Eclipse. Je sice multiplatformní, ale je nutné SWT aplikaci kompilovat zvlášť pro každou platformu.
 - Další, většinou open-source sady komponent postavené na Swingu.



Applety, aplikace

■ Applet

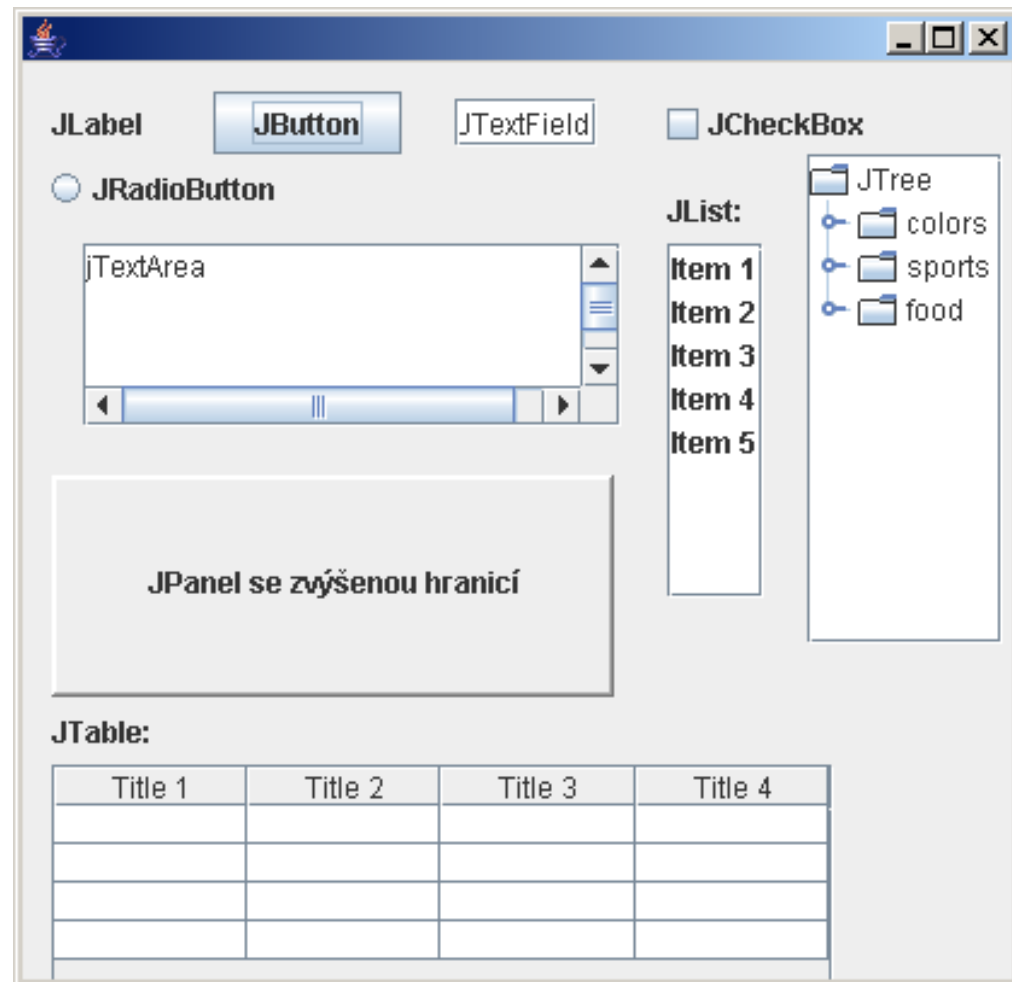
- Kreslí se na grafickou komponentu JApplet (příp. Applet při použití AWT).
- Applet je malá aplikace zakomponována do webové stránky. Applet se musí stáhnout z webu při každém načtení stránky, teprve poté se spustí.
- S applety se můžete v dnešní době setkat v intranetech různých společností, ale upouští se od nich, jedná se o mrtvou technologii.

■ Aplikace

- Kreslí se na grafickou komponentu JFrame (příp. Frame při použití AWT)
- Spouští se pomocí programu „java“, který je součástí JRE (Java Runtime Edition). V moderních operačních systémech obvykle stačí „poklepat“ myší na JAR soubor a GUI aplikace se spustí.

Základní komponenty

- JLabel
- JButton
- JCheckBox
- JRadioButton
- JList
- JTextField
- JPanel
- JMenuBar
- JTable
- JTree
- JFrame
- JDialog
- a další





Komponenta, Kontejner

- **Komponenta**

- Grafický objekt, který má předka typu JComponent (např. JLabel, JButton, JTextField, ...)

- **Kontejner**

- Grafický objekt, který má potomka typu Container. Jedná se o komponentu, která může obsahovat jiné komponenty (např. JPanel, JFrame, ...)



Rozmístění komponent na obrazovce

- Způsob rozmístování komponent na formuláři není řízen absolutní polohou ovládacího prvku, ale „správcem rozvržení“ (Layout manager), který rozhoduje o tom, jak budou komponenty na formuláři rozmístěny.
 - Správce rozvržení automaticky přizpůsobuje komponenty rozměrům okna aplikace tak, aby po změně velikosti okna aplikace mohl odpovídajícím způsobem upravit také velikost, tvar a rozložení komponent na příslušném formuláři.
 - Existuje několik správců rozvržení.
 - Každý kontejner může mít v jednu chvíli nastaveného právě jednoho správce rozvržení. Na jednom formuláři je možné mít více kontejnerů s různými správci rozvržení.

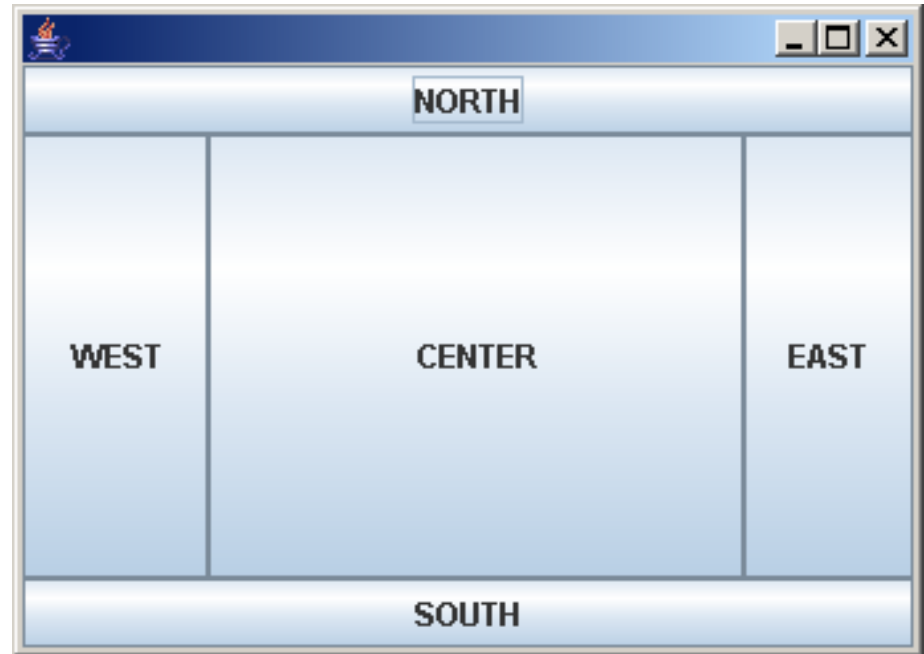


Null layout

- Nejedná se o čistokrevný layout, ale je důležité ho zmínit, protože se hodně využívá při tvorbě vlastních komponent
- Při nastavení layoutu kontejneru se nastaví:
`setLayout(null);`
- Při tomto layoutu se rozmístění komponent řídí podle levého horního rohu komponenty (location) a velikosti komponenty (size)

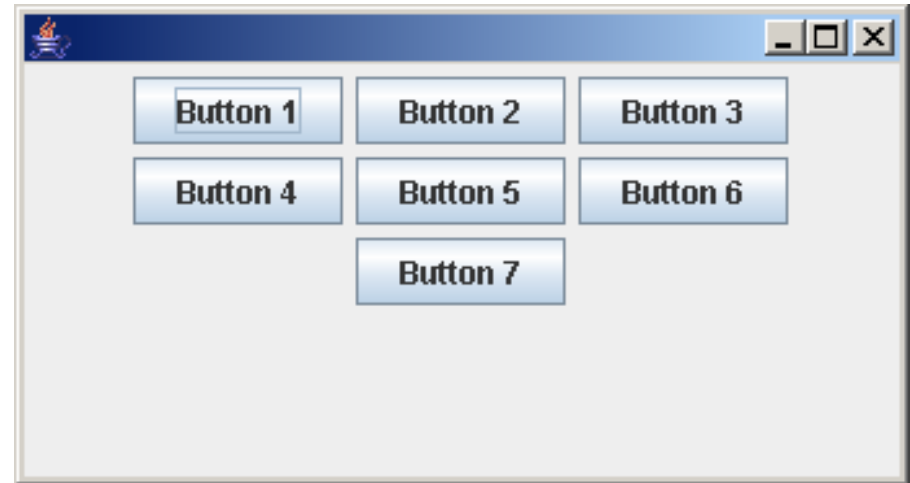
BorderLayout

- Rozpoznává čtyři hraniční a jednu středovou oblast:
 - NORTH, SOUTH, EAST, WEST, CENTER
 - NORTH a SOUTH má fixní výšku a dynamickou šířku. EAST, WEST právě naopak. CENTER vyplní vše ostatní.



FlowLayout

- Komponenty se vkládají zleva doprava a při zaplnění řádku se vkládání komponent přesune na další řádek.
- Komponenty se standardně centrují, ale je možné nastavit, aby se zarovnávaly vlevo nebo vpravo.



GridLayout

- Vytváří se tabulka komponent
- Komponenty se vkládají zleva doprava a shora dolů uvnitř mřížky
- V konstruktoru se určí počet řádků a sloupců
- Na obrázku vpravo je tabulka, která má 4 řádky. Pokud je počet komponent menší, pak jsou místa nevyužitá!



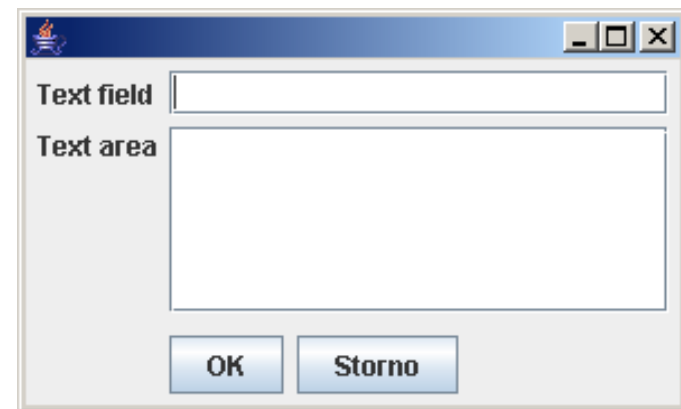
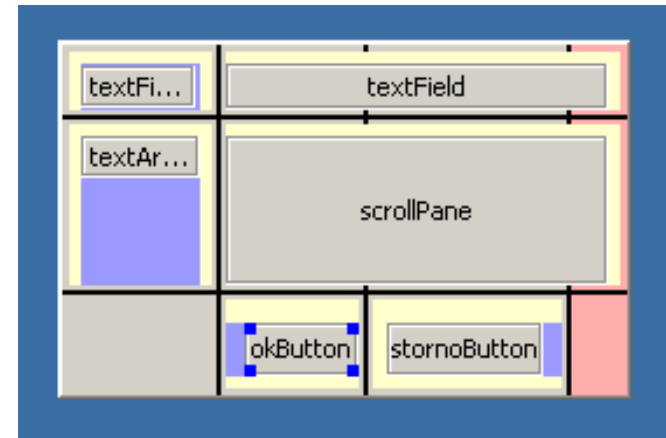


GridBagLayout I.

- Tento layout umožňuje nastavit každou komponentu tak, jak si to programátor přeje. Nevýhodou je značná složitost práce s tímto layoutem.
- Plocha je rozdělená na mřížku (jako GridLayout).
- S pozicí komponent, které jsou v mřížce obsaženy, je možné dělat některé pokročilejší úpravy.

GridBagLayout II.

- Na prvním obrázku vpravo je návrh GridBagLayoutu v NetBeans.
- Na druhém obrázku je výsledné zobrazení v aplikaci.





Ostatní layouty

- **BoxLayout:** Zjednodušený GridBagLayout – od Java SE 5.
- **CardLayout:** Podobné jako skládání karet na sebe. Vhodné při tvorbě Wizardů (průvodců), kde se kliká „Next“, „Previous“, ... a při kliknutí na takové tlačítko se jednoduchým způsobem zobrazí formulář s komponentami.
- **GroupLayout:** První layout navržený pro grafické návrháře (NetBeans Matisse nebo Eclipse WindowBuilder). V tomto layoutu se používá princip ukotvení komponent, přičemž každá komponenta může být ukotvena nahoru/dolů a doleva/doprava vůči nějaké komponentě nebo formuláři. Také je možné nastavit, jestli se bude komponenta „roztahovat“ do výšky anebo do šířky.



Použití layoutů

1. Vhodná kombinace layoutů a kontejnerů
 - Př.: Vytvoření dialogového okna:
 - A. Vytvoří se nový JDialog a nastaví se jeho layout na BorderLayout
 - B. Na CENTER se do framu vloží JTextArea a napíše se do něj text „Autor: ...“
 - C. Na SOUTH se do framu vloží JPanel (ten má standardně FlowLayout – ten zůstane nezměněn)
 - D. Do panelu se vloží tlačítko a jeho text se změní na „Potvrdit“
2. Nebo použití GridBagLayoutu
3. Nebo použití GroupLayoutu (vhodné od Java SE 6)



Události I.

- V Javě je odděleno rozhraní (grafické komponenty) od implementace (kód, který chceme spustit, když v souvislosti s komponentou dojde k určité události)
- Výhodou je jednotný přístup ke všem typům událostí a flexibilita
- Nevýhodou je pro začátečníka na první pohled větší složitost



Události II.

- Příklad zachycení události tlačítka
 - Třída JButton obsahuje metodu `addActionListener()`
 - Do této metody se předá objekt, který implementuje rozhraní `ActionListener`, které obsahuje jedinou metodu: `actionPerformed()`
 - Příslušná metoda pak bude vykonána při každém stisknutí tlačítka



Nejčastěji používané události

- ActionListener – kliknutí na tlačítko
- KeyListener – stisk klávesy
- MouseListener – stisknutí tlačítka myši
- MouseMotionListener – pohyb myši
- ...
- Všechny události mají příponu Listener



Modely a posuvníky



Modely

- V Javě mají komponenty architekturu model/view/controller, která důsledně odděluje vizuální komponentu (např. instance třídy JTable) od jejích dat (implementace rozhraní TableModel) a také od implementace událostí (např. implementace rozhraní ActionListener)
- Tato architektura poskytuje velkou flexibilitu
- Na druhou stranu je práce např. s instancí třídy JTable komplikovanější



Modely

- Rozhraní TableModel přidružené ke třídě JTable je odpovědné za zaplnění tabulky daty
- Model je také zodpovědný za poskytnutí dalších informací:
 - Rozměry tabulky
 - Typ dat uložených v jednotlivých sloupcích tabulky
 - Informace o záhlaví
 - Informace o tom, zda lze obsah příslušné buňky upravovat



TableModel

- Rozhraní TableModel obsahuje následující metody, které se musí předefinovat:
 - `getRowCount() : int`
 - `getColumnCount() : int`
 - `getValueAt(řádek : int, sloupec : int) : Object`
 - `setValueAt(hodnota : Object, řádek : int, sloupec : int)`
 - `getColumnName(sloupec : int) : String`
 - `getColumnClass(sloupec : int) : Class`
 - `isCellEditable(řádek : int, sloupec : int) : boolean`
 - `addTableModelListner(listener : TableModelListener)`
 - `removeTableModelListener(listener : TableModelListener)`
- Naštěstí Java poskytuje ještě třídy `AbstractTableModel` a `DefaultTableModel`, které obě implementují rozhraní `TableModel`. Ve většině případů stačí použití těchto tříd a úsilí programátora se tak sníží na minimum



Třída JScrollPane

- V komponentách není automatická podpora posuvníků
- Řešením je třída JScrollPane, která podporu posuvníků obsahuje. Jedná se o kontejner, který má standardně nastaven border layout a objekt, ke kterému je nutné přidat posuvník, se do tohoto kontejneru vloží na CENTER
- Postup přidání podpory posuvníků k tabulce:
 1. Vytvořit instanci třídy JScrollPane
 2. Přidat do tohoto kontejneru instanci tabulky



Look and Feel

Java aplikace může změnit kompletně svůj vzhled bez jakéhokoliv kódování!

Standardně jsou k dispozici následující vzhledy:

System

Windows: Vzhled Windows nebo WindowsXP

Linux: Vzhled GTK nebo Motif

Mac: Vzhled nativní Mac aplikace

CrossPlatform ~ Metal

Další vzhledy od komunity



Změna vzhledu

Při startu aplikace parametrem -D

```
java -Dswing.defaultlaf=... MyApp
```

Při startu aplikace v metodě main

```
UIManager.setLookAndFeel(String);
```

V průběhu běhu aplikace

```
UIManager.setLookAndFeel(String);
```

```
SwingUtilities.updateComponentTreeUI(myJFrame);
```

```
myJFrame.pack();
```

`UIManager.getSystemLookAndFeelClassName()` vrátí Look and Feel nativní pro daný OS.