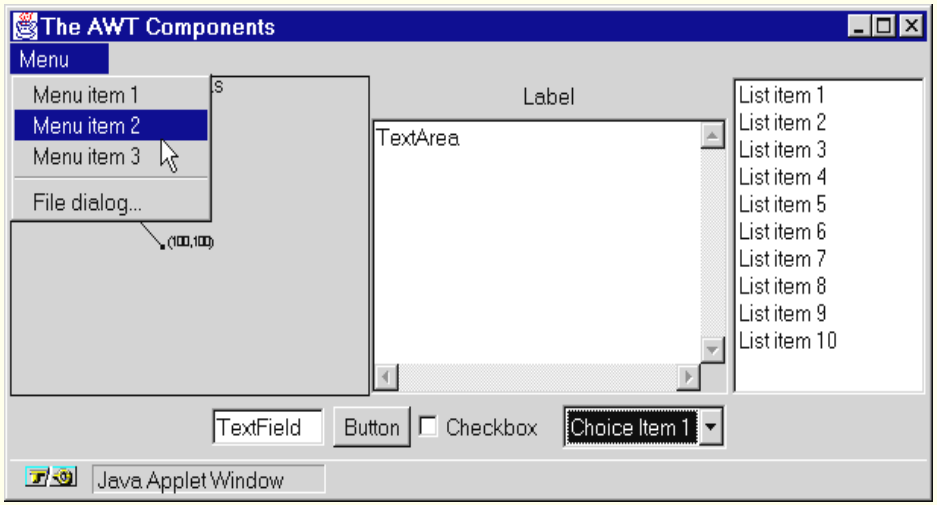
GUI (Graphical User Interface)

Vizuální a interaktivní komunikaci počítač-člověk podporují balíčky:

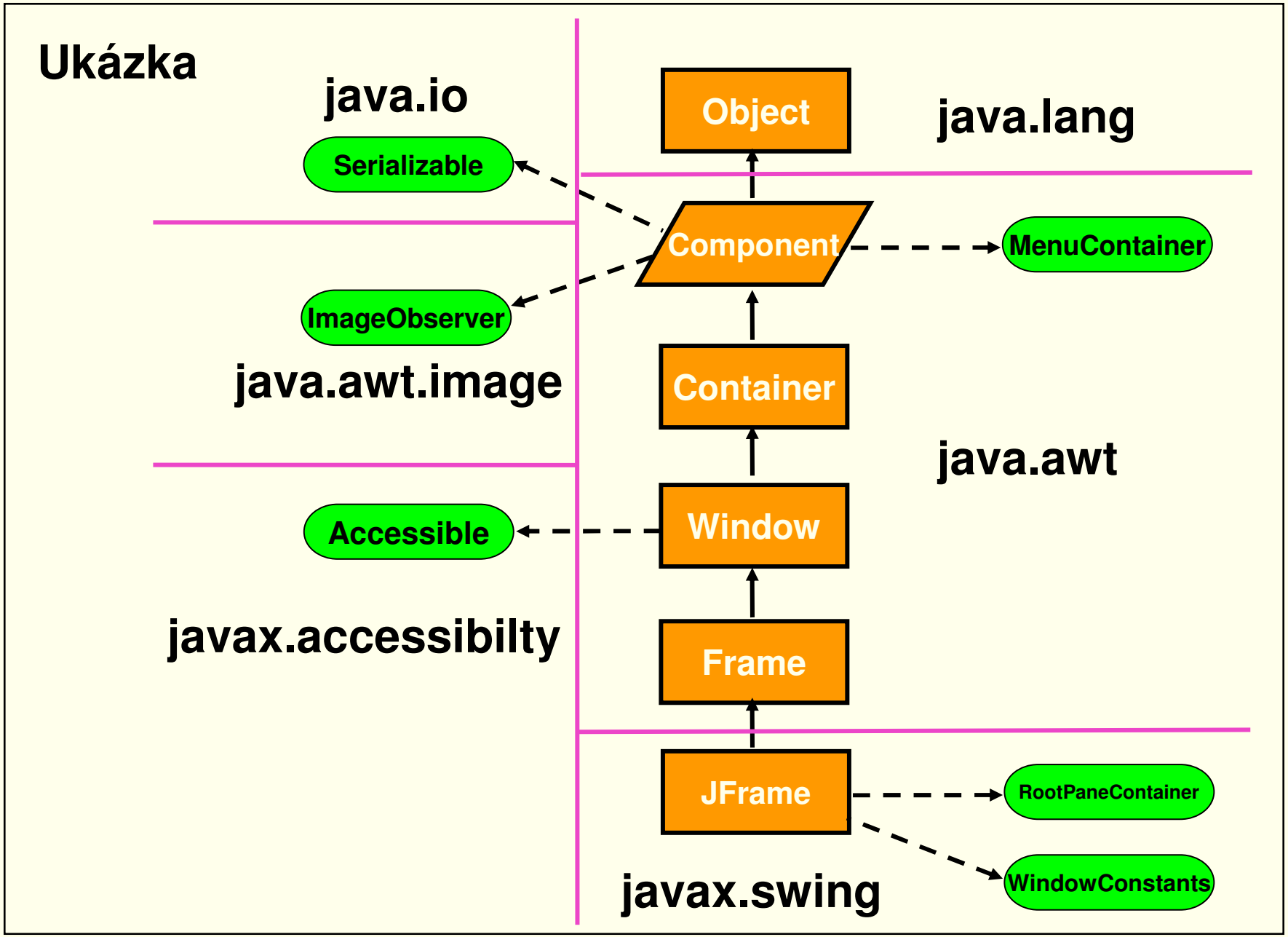
- **java.awt - obsahuje:**
  - **komponenty: knoflíky, textová pole, menu, posuvníky, grafiku ....**
  - **kontejnery: tj. komponenty do kterých lze vkládat komponenty,**
  - **layout managery: rozmisťují komponenty v plošekontejneru.**
- **java.awt.event - pojednává události a jejich zachytávání.**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

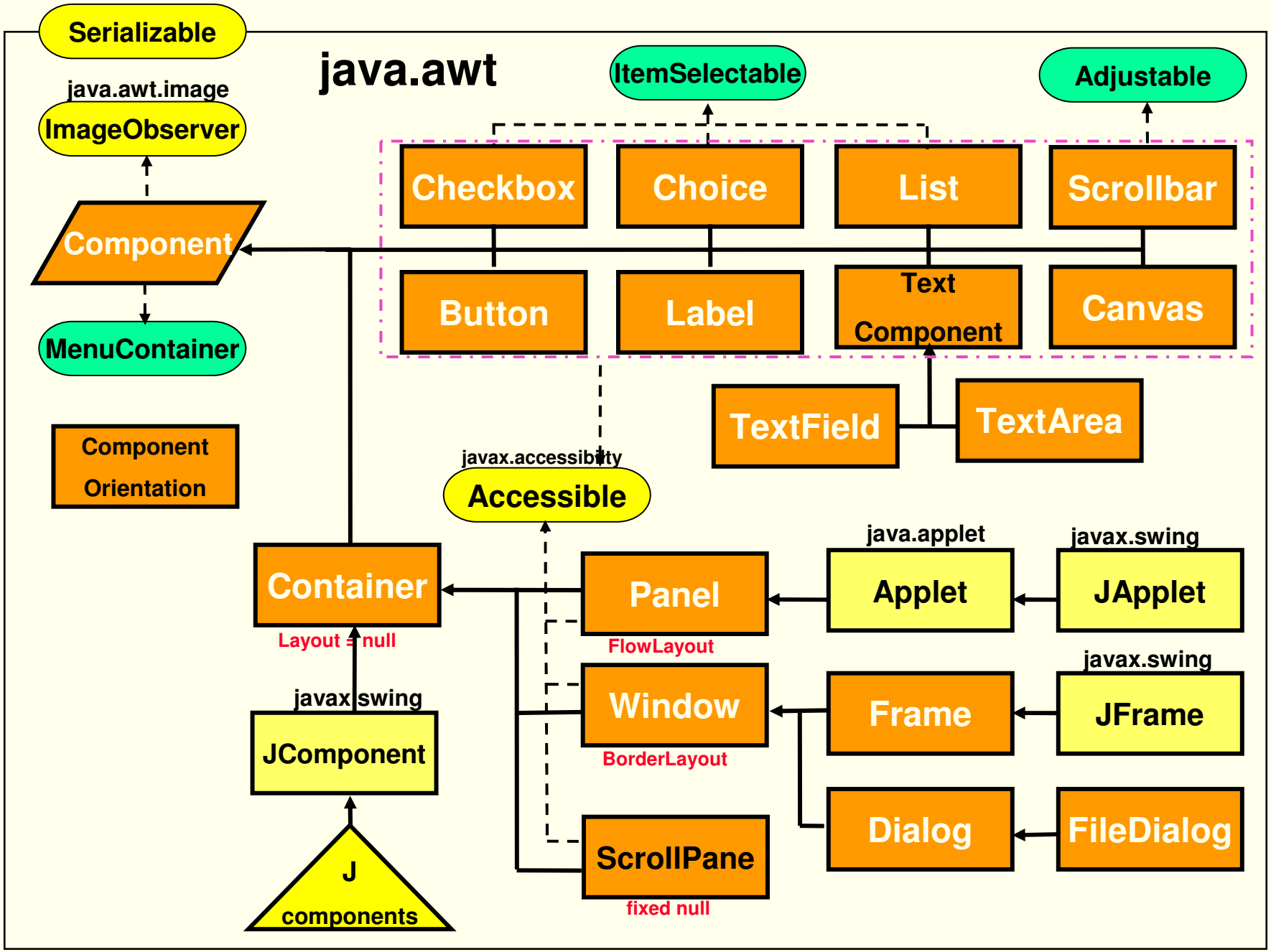
Ukázka v awt :

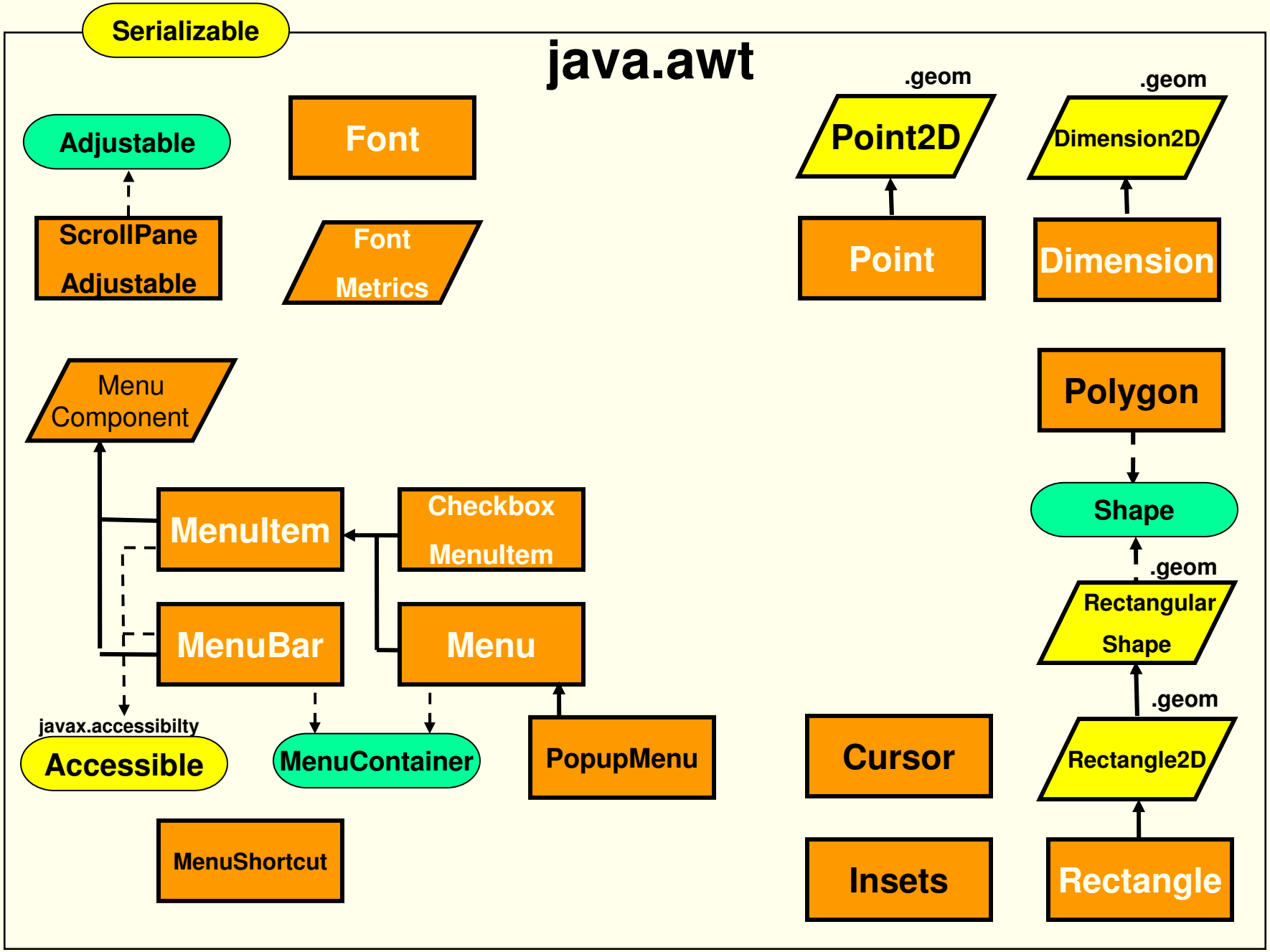


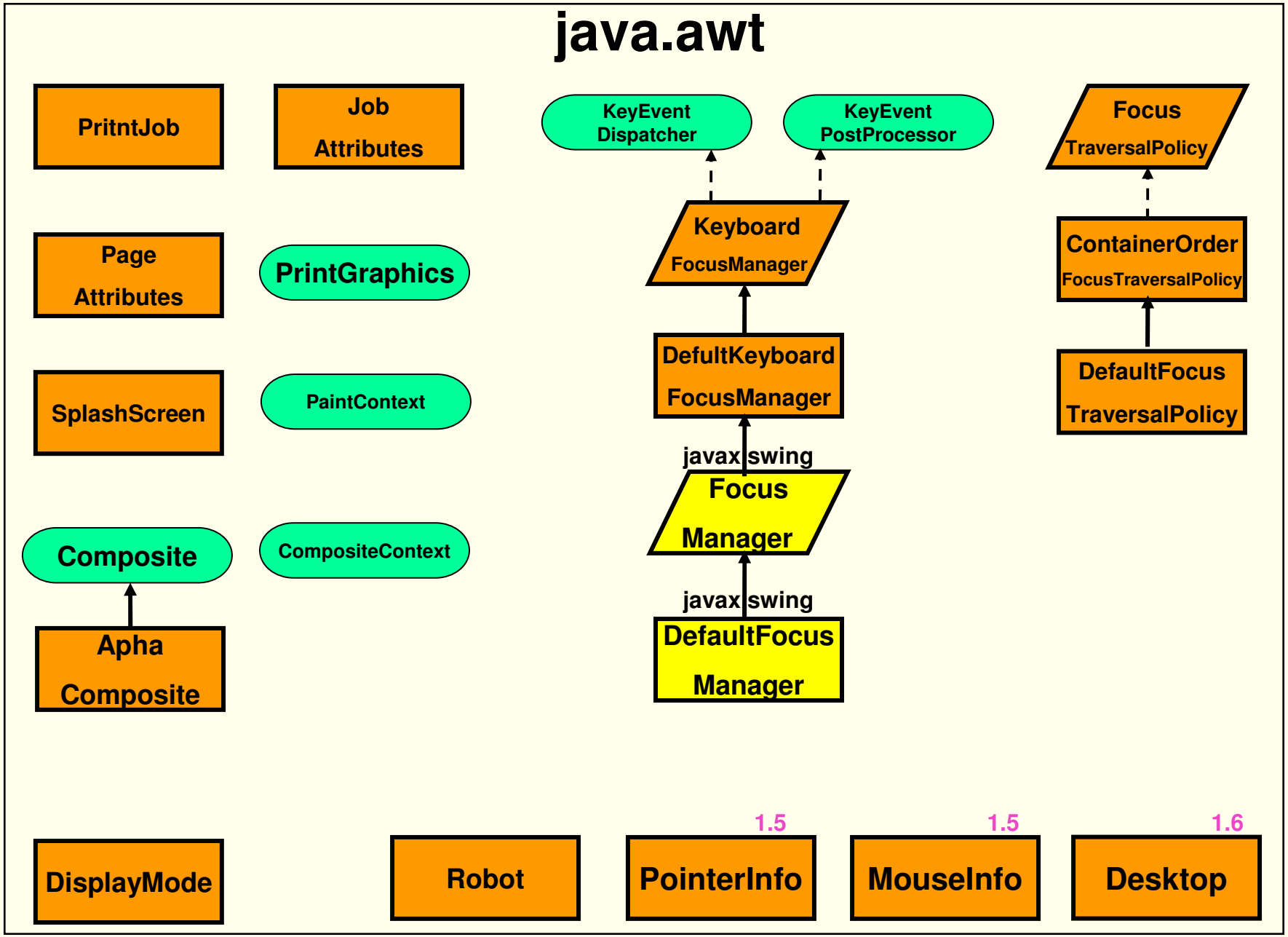
Zásady návrhu GUI

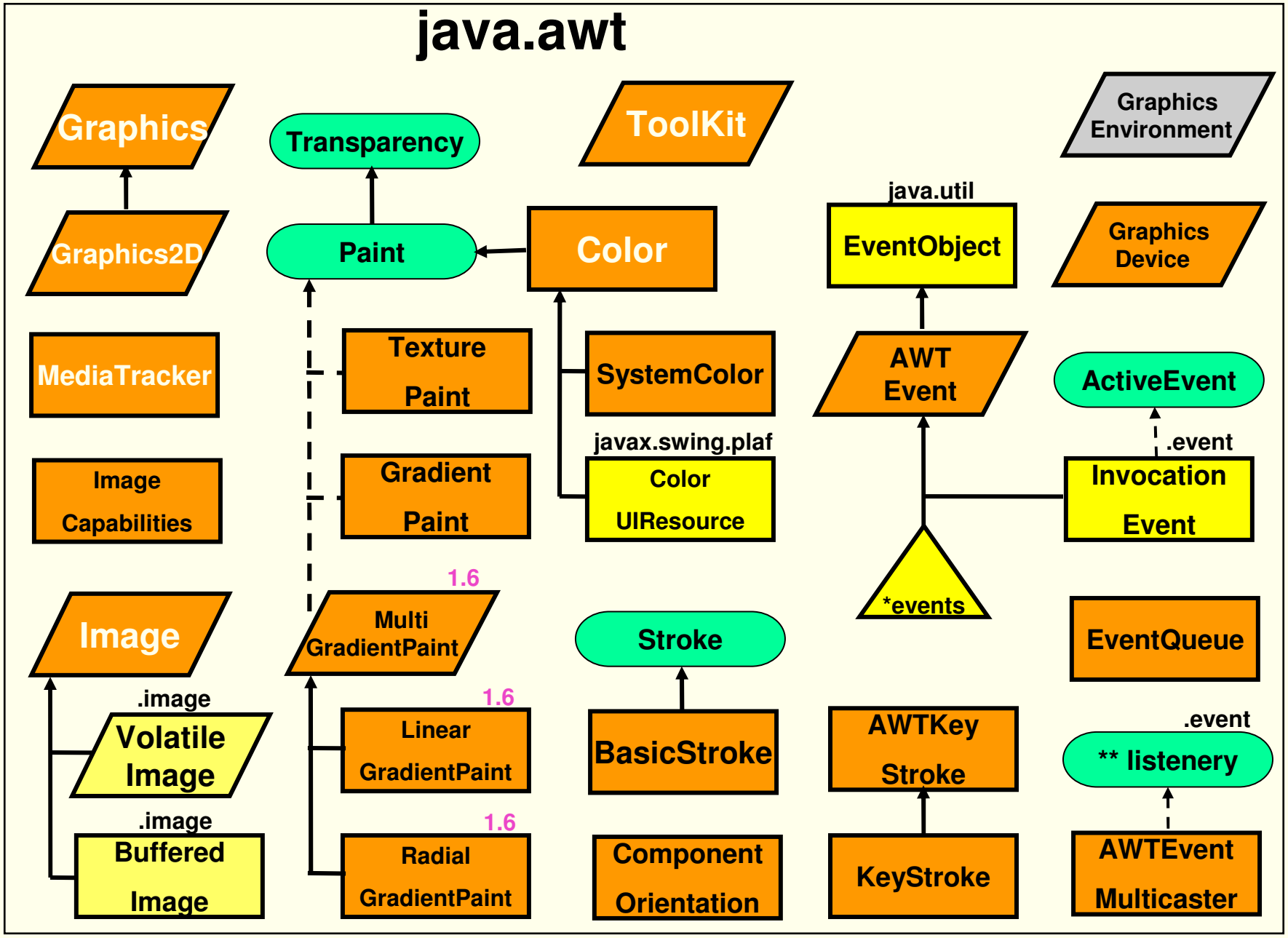
- Kvalita GUI podstatně ovlivňuje efektivitu práce uživatele (i negativně) .
- Uživatel podle GUI posuzuje kvalitu aplikace (hazuka zpochybňuje).
- Usilujte o jednoduše elegantní návrh s intuitivní a konzistentní funkcionalitou.
- Rozumně s rozměry, barvami a kontrasty - mají asociované významy.
- Respektujte 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.
- 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)


Ukázka
java.io
Serializable
ImageObserver
java.awt.image
Accessible
javax.accessibilty
Object
java.lang
Component
MenuContainer
Container
java.awt
Window
Frame
JFrame
RootPaneContainer
WindowConstants
javax.swing

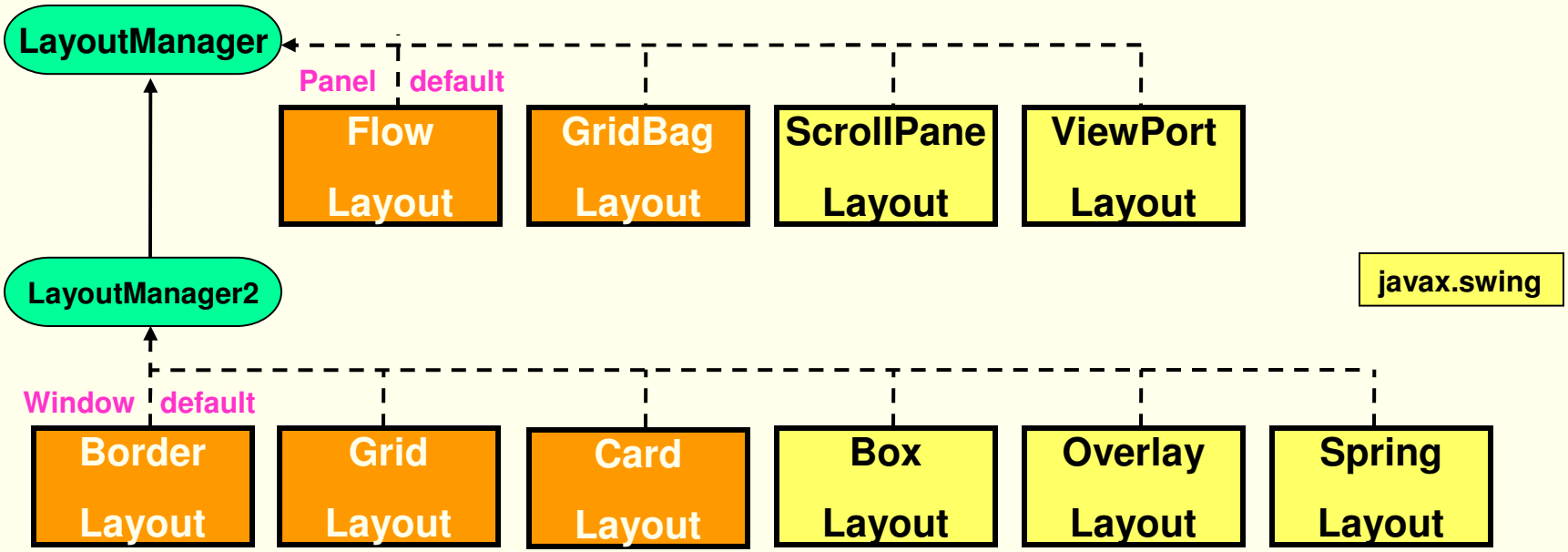

Serializable
java.awt
ItemSelectable
Adjustable
java.awt.image
ImageObserver
Checkbox
Choice
List
Scrollbar
Component
Button
Label
Text
Component
Canvas
MenuContainer
TextField
TextArea
Component
Orientation
javax.accessibility
Accessible
java.applet
javax.swing
Container
Panel
Applet
JApplet
Layout = null
FlowLayout
javax.swing
Window
Frame
JFrame
javax.swing
JComponent
BorderLayout
Dialog
FileDialog
J
components
ScrollPane
fixed null


Serializable
java.awt
Adjustable
Font
.geom
Point2D
.geom
Dimension2D
ScrollPane Adjustable
Font Metrics
Point
Dimension
Menu Component
Polygon
MenuItem
Checkbox MenuItem
Shape
.geom
Rectangular Shape
MenuBar
Menu
.geom
javax.accessibilty
Accessible
MenuContainer
PopupMenu
Cursor
Rectangle2D
MenuShortcut
Insets
Rectangle


java.awt
PritntJob
Job Attributes
KeyEvent Dispatcher
KeyEvent PostProcessor
Focus TraversalPolicy
Page Attributes
PrintGraphics
Keyboard FocusManager
ContainerOrder FocusTraversalPolicy
SplashScreen
PaintContext
DefultKeyboard FocusManager
DefaultFocus TraversalPolicy
javax.swing
Focus Manager
Composite
CompositeContext
javax.swing
Apha Composite
DefaultFocus Manager
1.5
1.5
1.6
DisplayMode
Robot
PointerInfo
MouseInfo
Desktop


java.awt
Graphics
Graphics2D
Transparency
Paint
ToolKit
Color
Graphics Environment
java.util
EventObject
Graphics Device
MediaTracker
Texture Paint
SystemColor
AWT Event
ActiveEvent
Image Capabilities
Gradient Paint
javax.swing.plaf
Color UIResource
.event
Invocation Event
*events
1.6
Image
Multi GradientPaint
Stroke
EventQueue
.image
Volatile Image
1.6
Linear GradientPaint
BasicStroke
AWTKey Stroke
.event
** listenery
.image
Buffered Image
1.6
Radial GradientPaint
Component Orientation
KeyStroke
AWTEvent Multicaster

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.

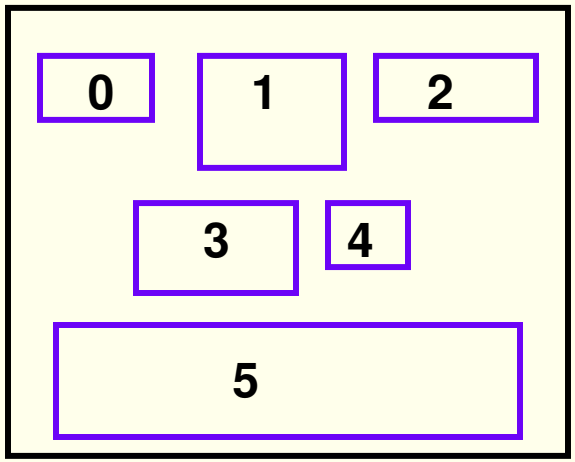
Card - jako balíček karet - vidět je jen vrchní karta.

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

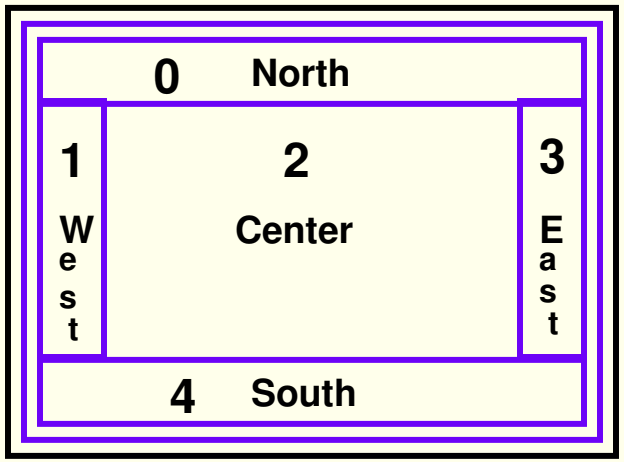
GridBag - nepravidelná mřížka - jedna komponenta zabere i více políček.

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

java.awt: Layouts



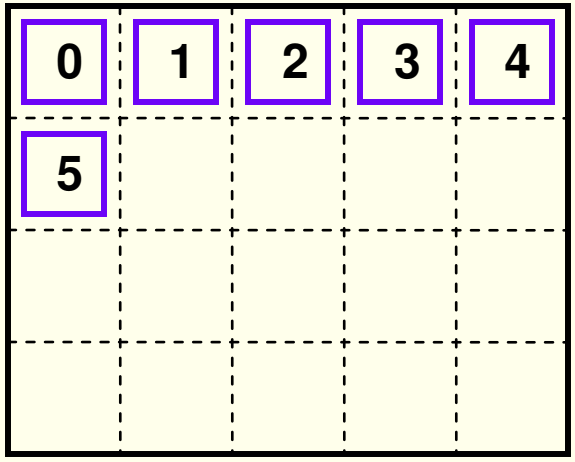
FlowLayout



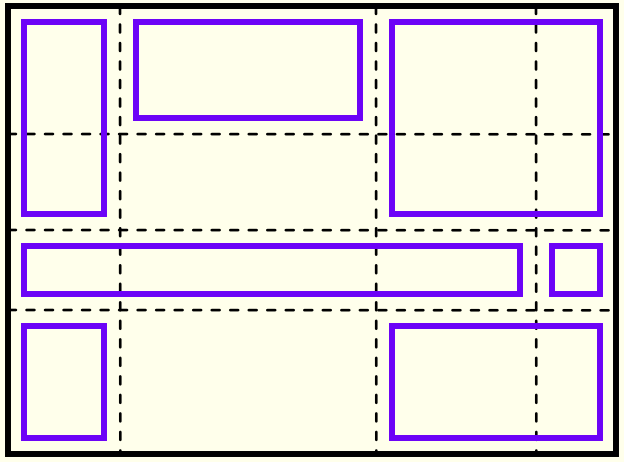
BorderLayout

Čísla vyjadřují index komponenty v seznamu.

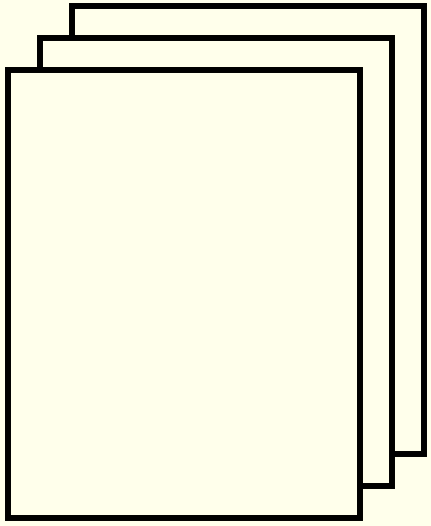
BorderLayout má pět oblastí. V každé zobrazí je jednu komponentu – poslední z přidělených do oblasti. Centrální oblast je vzadu – event. překrytá ostatními. Indexy oblastí jsou dle JBuilder – designeru.



GridLayout



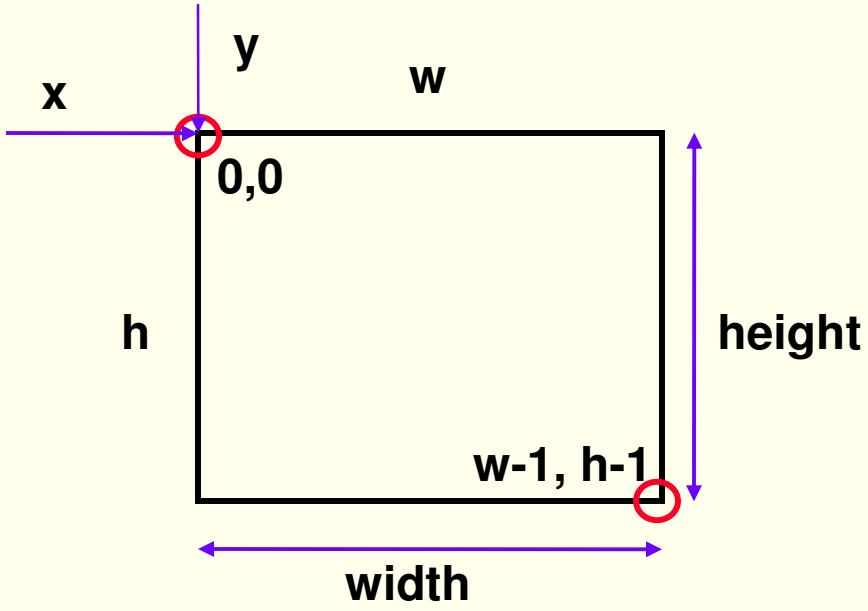
GridBagLayout



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)**

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**

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.**

Graphics

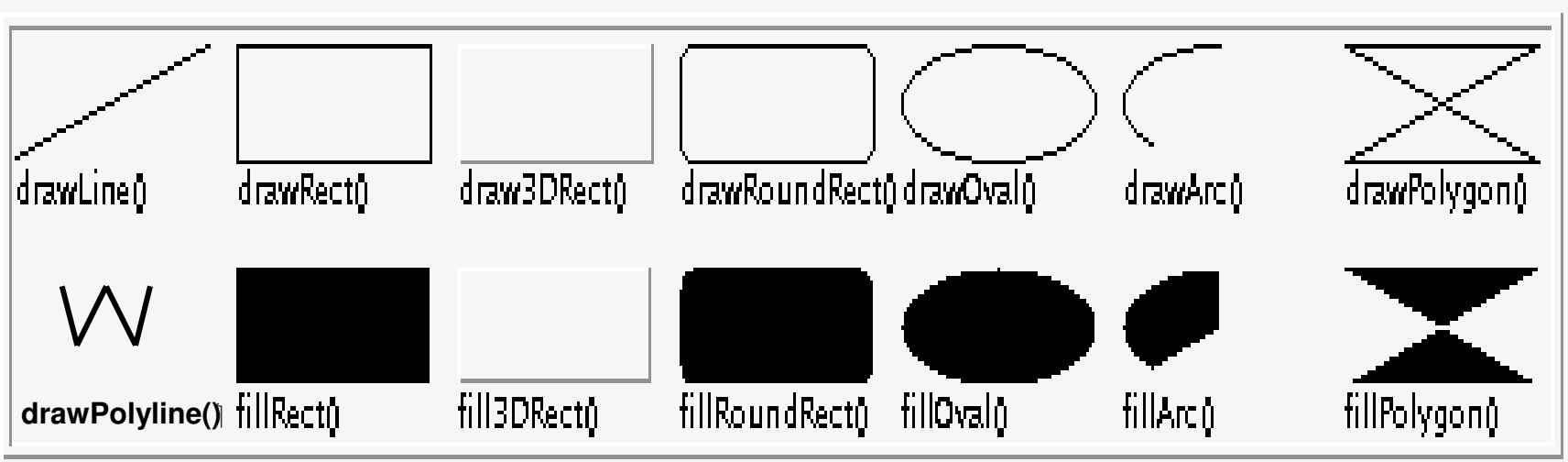
Tento objekt umožňuje v objektech typu Component a tedy i Canvas, Panel, Applet, Window, Frame atd. vytvořit a upravovat kresby, texty a obrázky.
Přístup k němu se získá přepsáním zděděné prázdné metody např. takto:

```
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:



- **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**

Font

Tato třídy podporuje rozmanitá vykreslení textů. Font má konstruktor:

```
public Font ( String name, int style, int size )
```

kde: name - courier, helvetica, dialog, inputdialog, sanserif, monospaced ...

style - tvar znaků zadaný konstantami PLAIN, BOLD, ITALIC

size - bodová velikost např. 8 .. 96

Příklad:

```
public void paint( Graphics g ) {
    g.setFont( new Font( "courier", Font.ITALIC + Font.BOLD, 18 ) );
    g.drawString( "Hello, World",  75, 400 );
}
```

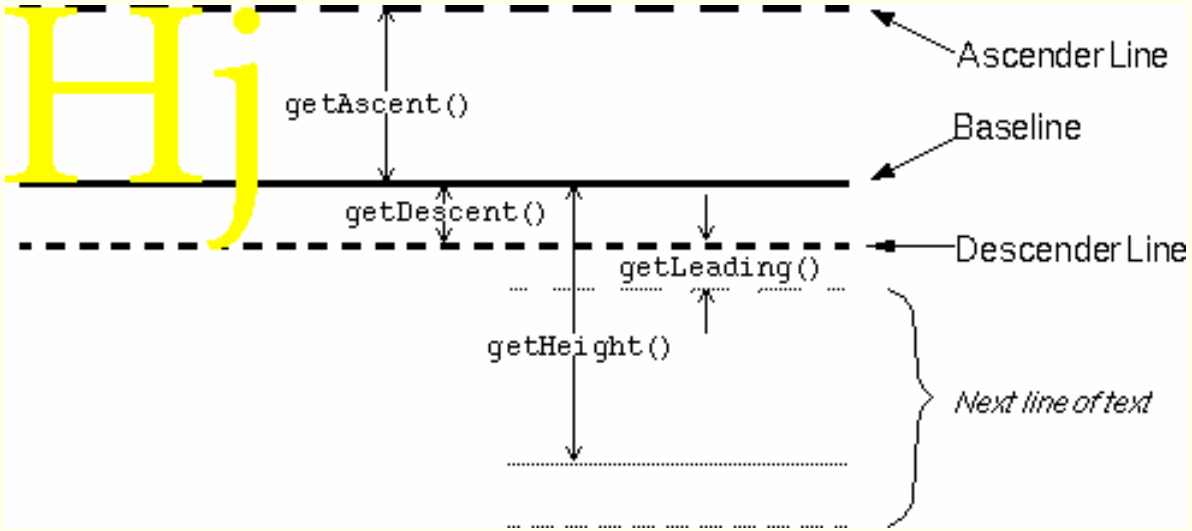
Fonty jsou uspořádány v rodinách. Zjistit dostupné fonty lze takto:

```
GraphicsEnvironment ge =
            GraphicsEnvironment.getLocalGraphicsEnvironment( );
Font[ ] ff = ge.getAllFonts( );
```

FontMetrics

Zvláště proporcionální písmomalířství je obtížné. FontMetrics umožňuje písmo daného fontu měřit a zkusmo pak zvolit vhodný font.

Metrika písma:



K měření pro daný font slouží metody: getLeading, getAscent, getDescent, getHeight, getMaxAscent, getMaxDescent, getMaxAdvance ... a zejména pratické:

- **int charWidth(char c) – vrací odstup znaku od dalšího v řádce.**
- **int stringWidth(String s) – vrací délku řetězu.**

```
FontMetrics fm = g.getFontMetrics( );
int w = fm.stringWidth( "áčďéěíňóřšťúůýžÁČĎÉĚÍŇÓŘŠŤÚŮÝŽ" );
```

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.

Modifikace kreseb

- Při objevení (zpočátku, deiconifikaci, odkrytí) volá awt vlákno metodu g.clearRect(...), která přebarví pozadí a pak metodu paint(Graphics g).
- Při programovém volání metody repaint(...), awt vlákno zavolá metodu update(Graphics g), která defaultně obsahuje:

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

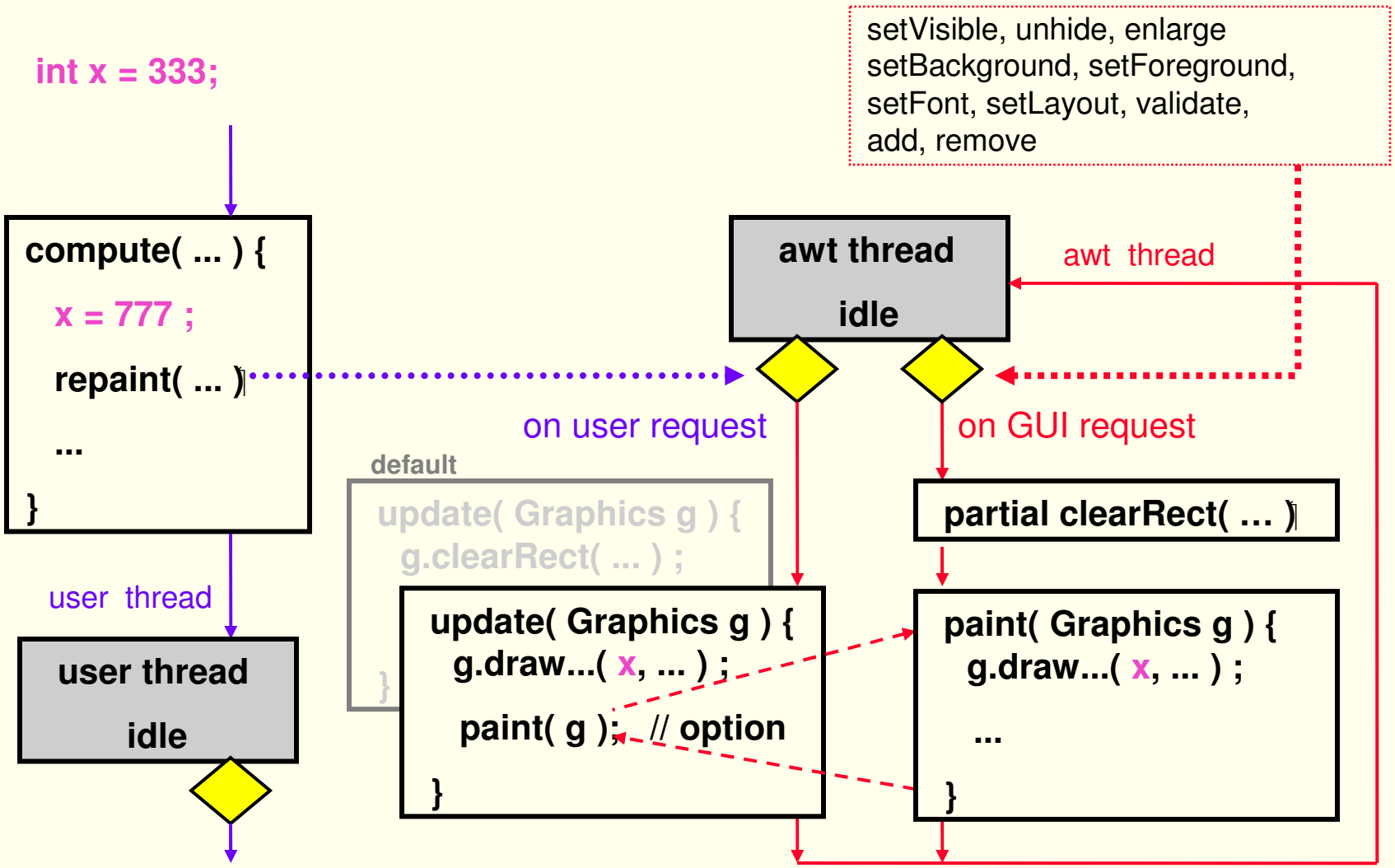
- Metodu update je možné přepsat, např. tak aby se nepřemalovávalo, čímž se kreslí na předchozí kresbu:

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

Vnější změnou atributů užitých v kreslících metodách (tj. g.draw( ) apod.) a následném zavolání repaint(...) se kresba změní či animuje.

Přepočítání atributů či pospávání by mělo konat samostatné vlákno – nikoli awt – thread (je dost zaměstnáno a sleduje chování uživatele).

repaint( ), update( ), paint( )



Frame s komponentami

```
public class MyFrame extends Frame {
   Button b1 = new Button( " OK " );
   Button b2 = new Button( " Cancel " );
   TextField tf = new TextField( " ********* " );
   public MyFrame( ) {
      super( " This is MyFrame " );                    // titulek
      this.setLayout( new FlowLayout( ) );             // zmena rozmisteni
      this.add( b1 );    b1.setForeground( Color.GREEN );
      this.add( b2 );    b2.setForeground( Color.RED );
      this.add( tf );
      this.setBounds( 100, 100, 400, 400 );
      this.setVisible( true );                         // aktivace awt vlakna
   }
   public static void main( String [ ] args )  {
      MyFrame mf = new MyFrame( );
      mf.b1.setBackground( Color.YELLOW );             // zmena zvnejsku
   }
}
```

Vykreslení obrázku

```
public class ImageLoad extends Frame {
   static Toolkit tk = Toolkit.getDefaultToolkit( );
   static Image img =  tk.getImage( "C:\\...\\star.gif" );
   int width, height ;
   public ImageLoad( ) throws Exception {
      MediaTracker mt = new MediaTracker( this );
      mt.addImage( img, 0 );
      mt.waitForAll( );
      width = img.getWidth( this );
      height = img.getHeight( this );
      this.setBounds( 100, 100, 200, 200);
      this.setVisible( true );
   }

   public void paint( Graphics g ) {
      g.drawImage( img, 100, 100, this );
   }
}
```