8. Grafické uživatelské rozhraní

Až dosud jsme pro výstupy a vstupy do našich programů využívali pouze konzoli nebo soubor. Java nám však poskytuje nástroje pro tvorbu grafického uživatelského rozhraní, tj. "okének", menu, tlačítek atd. Už od první verze Javy je její součástí balík **java.awt** (abstract window toolkit) poskytující třídy pro tvorbu rozhraní a od verze 1.2 také balík (package) **javax.swing**. V následujících příkladech budeme využívat především třídy z balíku swing, používání tříd z AWT je velmi podobné.

První aplikace

Pokud chceme vytvořit aplikaci s grafickým uživatelským rozhraním, prvním krokem je vytvořit rámec („okno“) ve kterém aplikace poběží. V jazyce Java se používá třída **JFrame** (třída Frame v awt). Následující program ukazuje, jak vytvořit základní rámec aplikace a na obrázku vidíte výsledek.

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

public class PrvniAplikace extends JFrame {

class Wl extends WindowAdapter {

      public void windowClosing (WindowEvent e) {
         System.exit(0);
      }
};

public PrvniAplikace(String nazev) {
      super(nazev);
      addWindowListener (new Wl());
}

public static void main (String []args) {
  PrvniAplikace apl=new PrvniAplikace("Moje prvni aplikace");
  apl.setLocation(100,100);
  apl.setSize(200,200);
  apl.setVisible(true);
}
}
```

Třída PrvniAplikace je potomkem třídy **JFrame** a obsahuje dvě metody: konstruktor PrvniAplikace a metodu main.
Dále v našem programu vytvoříme vnitřní třídu Wl, která je potomkem třídy **WindowAdapter** a bude obsluhovat události okna (tj. uživatelské akce jako je zmenšení na ikonu, změna velikosti okna nebo zavření okna). Zde je definováno, že při uzavření okna má skončit aplikace – tj. předefinujeme metodu **windowClosing()** předka, která se volá při uzavření okna.

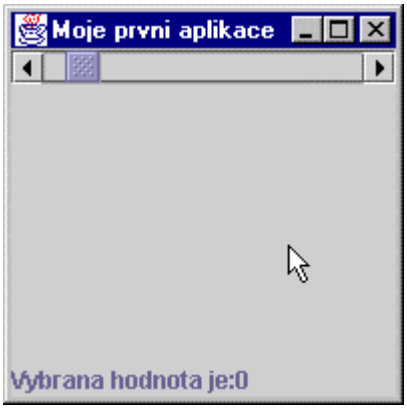
Ošetření ostatních události, které vznikají u okna (tzv. WindowEvent) nepřekrýváme – tj. bude se používat standardní ošetření nadefinované v předkovi. Všimněte si, že třída Wl je nadefinována uvnitř třídy PrvniAplikace – třídy vytvořené uvnitř jiných tříd se nazývají vnitřní třídy (inner class).
V konstruktoru pomocí vyvolání konstruktoru předka s parametrem tj. příkaz **super (String s)** přiřadíme našemu oknu titulek. Další část konstruktoru, metoda **addWindowListener**, zajistí, že naše okno bude reagovat na obsluhu uživatele prostřednictvím třídy **Wl** (tj. teprve nyní se přiřadí obsluha popsaná ve třídě Wl k našemu oknu).
V metodě main pak vytvoříme instanci třídy PrvniAplikace s názvem "Moje první aplikace", nastavíme jí výchozí umístění na obrazovce (metoda **setLocation**), výchozí velikost (metoda **setSize**) a zobrazíme ji na obrazovce (metoda **setVisible**).
Většina RAD nástrojů pro Javu používá pro popis zavření okna ještě o něco složitější konstrukci s využitím anonymní vnitřní třídy. Následující program je stejný jako předchozí, pouze pro obsluhu událostí okna používá anonymní vnitřní třídu. Tato konstrukce bude používána ve všech dalších programech.

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

public class PrvniAplikace2 extends JFrame {

public PrvniAplikace2(String nazev) {
     super(nazev);
     addWindowListener(new WindowAdapter () {
        public void windowClosing (WindowEvent e) {
          System.exit(0);
          }
     });
}

public static void main (String [] args) {
  PrvniAplikace2 apl=new PrvniAplikace2("Moje prvni aplikace");
  apl.setLocation(100,100);
  apl.setSize(200,200);
  apl.setVisible(true);
}
}
```

Vkládání dalších komponent

Na následující příkladě si ukážeme, jak do okna aplikace přidat další prvky. Zadání je jednoduché, přidat do aplikace posuvník a vypisovat, na jakou hodnotu je právě nastaven. Použijeme tedy instanci třídy **JScrollBar** a **JLabel**.

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

public class PrvniAplikace3 extends JFrame {

private int hodnota = 0;
//Definice a vytvoření instancí posuvníku a návěstí.
private JScrollBar posuvnik = new JScrollBar
                  (JScrollBar.HORIZONTAL,0,1,0,100);
private JLabel napis = new JLabel ("Vybrana hodnota je:"+hodnota);

public PrvniAplikace3(String nazev) {
      super(nazev);
      addWindowListener(new WindowAdapter () {
         public void windowClosing (WindowEvent e) {
           System.exit(0);
         }
      });
      getContentPane().add(posuvnik,BorderLayout.NORTH);
      getContentPane().add(napis,BorderLayout.SOUTH);
}

public static void main (String []args) {
   PrvniAplikace3 apl=new PrvniAplikace3("Moje prvni aplikace");
   apl.setLocation(100,100);
   apl.setSize(200,200);
   apl.setVisible(true);
}
}
```



Jak vidíme, základ aplikace zůstal stejný, metoda main se vůbec nezměnila. Použité komponenty (posuvník JScrollBar a popiska JLabel) jsou definovány jako proměnné instance této třídy. Dále definujeme proměnnou **hodnota** pro předávání hodnoty nastavené na posuvníku. Popisku a posuvník umístíme do okna pomocí metody **add.** (Knihovna Swing vyžaduje, aby všechny vkládané komponenty byly umístěny do "podokna" rámce – podokno získáte voláním metody **getContentPane()** – bližší popis je v dokumentaci na java.sun.com). V metodě add se pomocí druhého parametru specifikuje rozmístění komponent v okně v rámci aktuálního správce rozvržení.

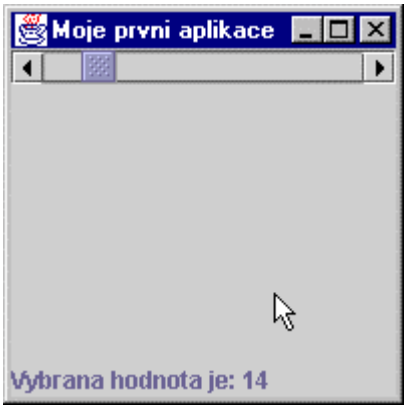
Posuvník je inicializován jako vodorovně orientovaný, nastavený na hodnotu nula, velikost posunovače je 1, minimální hodnota 0 a maximální 100 (protože posunovač má hodnotu 1, je skutečná nejvyšší nastavitelná hodnota 99, při velikosti posunovače 10 by byla 90 atd.). Tato aplikace sice zobrazí posuvník a nápis, ale na obsluhu posuvníku nereaguje. Je nutné obsloužit události posuvníku.

Obsluha událostí

Do naší aplikace je tedy nutné přidat obsluhu událostí generovaných instancí třídy JScrollBar.

Upravený kód vypadá následovně a vzhled aplikace ukazuje obrázek

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

public class PrvniAplikace4 extends JFrame {

private JScrollBar posuvnik;
private JLabel napis;
private int hodnota = 0;

private class Al implements AdjustmentListener {
      public void adjustmentValueChanged(AdjustmentEvent e) {
         hodnota = posuvnik.getValue();
         napis.setText("Vybrana hodnota je: "+hodnota);
      }
};

public PrvniAplikace4(String nazev) {
      super(nazev);
      addWindowListener(new WindowAdapter () {
         public void windowClosing (WindowEvent e) {
           System.exit(0);
         }
      });
      posuvnik= new JScrollBar(JScrollBar.HORIZONTAL,0,1,0,100);
      posuvnik.addAdjustmentListener (new Al());
      napis = new JLabel ("Vybrana hodnota je: "+hodnota);
      getContentPane().add(posuvnik,BorderLayout.NORTH);
      getContentPane().add(napis,BorderLayout.SOUTH);
}

public static void main (String []args) {
  PrvniAplikace4 apl= new PrvniAplikace4("Moje prvni aplikace");
  apl.setLocation(100,100);
  apl.setSize(200,200);
  apl.setVisible(true);
}
}
```



Do naší aplikace jsme přidali třídu **Al** implementující rozhraní **AdjustmentListener,** která definuje metodu **adjustmentValueChanged(AdjustmentEvent e).** Tato metoda má jako parametr AdjustmentEvent, tj. událost, kterou generují instance třídy JScrollBar. V rámci obsluhy události posuvníku pomocí metody **getValue( )** získáme hodnotu nastavenou na posuvníku a změníme text zobrazovaný popiskou pomocí metody **setText(String s)**. Ani toto však

ještě nestačí, posledním krokem je přiřadit v konstruktoru tento listener konkrétní instanci třídy JScrollBar pomocí metody **addAdjustmentListener( )**.

Obecně lze tedy říci, že komponentu, kterou chceme přidat do okna aplikace, nejprve deklarujeme jako datový atribut instance potomka třídy JFrame a přiřadíme mu počáteční hodnotu, tj. vytvoříme instanci komponenty pomocí konstruktoru. Pak definujeme třídu implementující rozhraní (případně třídu, která je potomkem příslušného XxxAdapteru) pro obsluhu událostí této komponenty.

Posledním krokem je přiřazení konkrétního "posluchače událostí" konkrétní komponentě pomocí metody addXxxListener().

Následující tabulky by vám měly umožnit základní orientaci v komponentách, událostech, rozhraních pro sledování událostí atd.

V tabulce je přehled všech událostí, rozhraní určených pro jejich obsluhu a metod pro přiřazení a odejmutí těchto ovladačů, v druhém sloupci naleznete seznam komponent, které generují tyto události.

Událost, rozhraní a add- a remove-metody	**Komponenty sledující tyto události**
ActionEvent ActionListener addActionListener( ) removeActionListener( )	JButton, JList, JTextField, JMenuItem a jeho potomci včetně JCheckBoxMenuItem, JMenu, and JPopupMenu
AdjustmentEvent AdjustmentListener addAdjustmentListener( ) removeAdjustmentListener( )	JScrollbar
ComponentEvent ComponentListener addComponentListener( ) removeComponentListener( )	Třída Component a její potomci,včetně tříd JButton, JCanvas, JCheckbox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileChooser, JFrame, JLabel, JList, JScrollbar, JTextArea, a JTextField
ContainerEvent ContainerListener addContainerListener( ) removeContainerListener( )	Třída Container a její potomci, včetně tříd JPanel, JApplet, JScrollPane, Window, JDialog, JFileChooser, a JFrame
FocusEvent FocusListener addFocusListener( ) removeFocusListener( )	Třída Component a její potomci,včetně tříd JButton, JCanvas, JCheckbox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileChooser, JFrame, JLabel, JList, JScrollbar, JTextArea, a JTextField
KeyEvent KeyListener addKeyListener( ) removeKeyListener( )	Třída Component a její potomci,včetně tříd JButton, JCanvas, JCheckbox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileChooser, JFrame, JLabel, JList, JScrollbar, JTextArea, a JTextField
MouseEvent MouseListener addMouseListener( ) removeMouseListener( )	Třída Component a její potomci,včetně tříd JButton, JCanvas, JCheckbox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileChooser, JFrame, JLabel, JList, JScrollbar, JTextArea, a JTextField

MouseEvent MouseMotionListener addMouseMotionListener( ) removeMouseMotionListener( )	Třída Component a její potomci,včetně tříd JButton, JCanvas, JCheckbox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileChooser, JFrame, JLabel, JList, JScrollbar, JTextArea, a JTextField
WindowEvent WindowListener addWindowListener( ) removeWindowListener( )	Window a jeho potomci JDialog, JFileChooser, a JFrame
ItemEvent ItemListener addItemListener( ) removeItemListener( )	JCheckBox, JCheckBoxMenuItem, JComboBox, JList
TextEvent TextListener addTextListener( ) removeTextListener( )	Potomci JTextComponent, včetně JTextArea a JTextField

tabulka 20 Vztah událostí a komponent

Následující tabulka obsahuje přehled metod definovaných v jednotlivých rozhraních pro sledování událostí. Pokud má rozhraní více jak jednu metodu, existuje třída implementující toto rozhraní standardním způsobem (jmenuje se vždy XxxAdapter, kde Xxx je shodné s první částí pojmenování Listeneru).

Listener - rozhraní a adapter - třída	**Metody v rozhraní**
ActionListener	actionPerformed(ActionEvent)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
ComponentListener ComponentAdapter	componentHidden(ComponentEvent) componentShown(ComponentEvent) componentMoved(ComponentEvent) componentResized(ComponentEvent)
ContainerListener ContainerAdapter	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
FocusListener FocusAdapter	focusGained(FocusEvent) focusLost(FocusEvent)
KeyListener KeyAdapter	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
MouseListener MouseAdapter	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)
MouseMotionListener MouseMotionAdapter	mouseDragged(MouseEvent) mouseMoved(MouseEvent)

WindowListener WindowAdapter	windowOpened(WindowEvent) windowClosing(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent)
ItemListener	itemStateChanged(ItemEvent)
TextListener	textValueChanged(TextEvent)

tabulka 21 Listenery a jejich metody

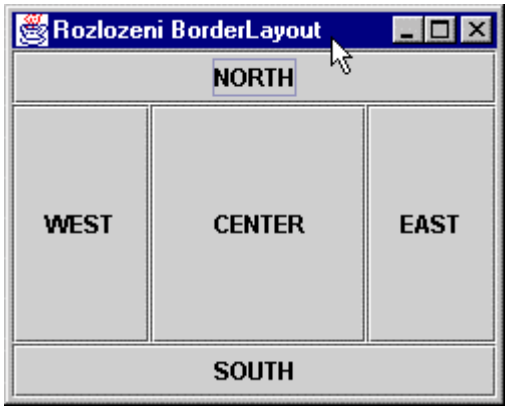
Rozložení komponent v rámci okna

V našem úvodním programu pro grafické rozhraní jsme použili několik konstrukcí, které ještě nebyly vysvětleny. Objevily se zde dva řádky kódu:

```
getContentPane().add(posuvnik,BorderLayout.NORTH);
getContentPane().add(napis,BorderLayout.SOUTH);
```

Zatím jsme si vysvětlili, že metoda add() slouží pro přidání komponenty do rámce, ale ne kam budou komponenty umístěny. Pro určení rozmístění komponent se používají správci rozložení (tzv. **layout manager**). Základní správce rozvržení si nyní popíšeme.

Rozložení BorderLayout



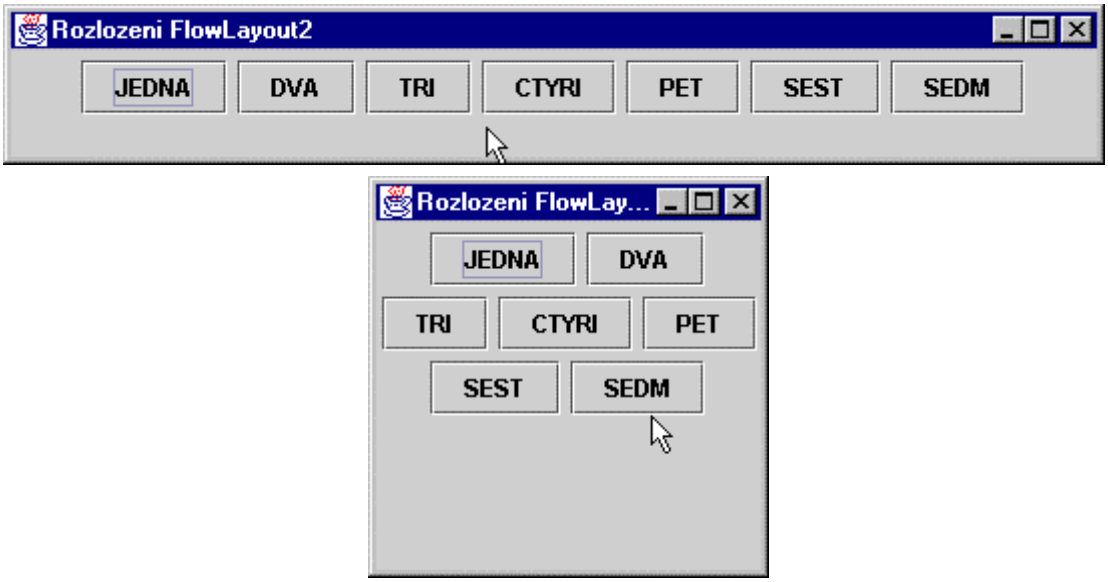
Toto rozložení je standardně nastaveným rozložením pro okno aplikace. Z tohoto důvodu jsme v úvodním příkladě žádné rozložení nenastavovali a jen jsme jej používali.

BorderLayout rozděluje okno aplikace do pěti oblastí (viz obrázek), přičemž do každé z nich lze vložit pouze jednu komponentu (může to být instance třídy JPanel, do které lze vložit další komponenty). Velikost komponenty závisí na oblasti do které je vložena. Oblasti NORTH, SOUTH, WEST a EAST jsou jen tak velké, aby bylo možno zobrazit komponentu do nich vloženou. Oblast CENTER pak zabírá celý zbytek okna a komponenta do ní vložená je roztažena na celou tuto oblast.

V úvodním příkladu jsme umístili posuvník do oblasti North a návěstí do oblasti South.

Rozložení FlowLayout

Rozložení FlowLayout, jak již název napovídá, je velmi volné a uspořádání komponent závisí na velikosti okna. Komponenty se rovnají do řádků v pořadí v jakém je metodou add přidáváte do okna. Následující obrázky ukazují dva různé vzhledy téže aplikace v závislosti na tom, jak si uživatel nastaví velikost okna.



Rozložení GridLayout

Toto rozložení vytváří tabulku o zadaném počtu řádků a sloupců. Komponenty mají maximální velikost, při které se do okna ještě vejdou. Zvětší-li uživatel okno, zvětší se komponenty. Následující obrázek ukazuje aplikaci s GridLayoutem nastaveným na 3 řádky a 2 sloupce. Komponenty jsou řazeny do tabulky podle pořadí jejich vložení do okna metodou add a jejich velikost je určena velikostí okna (mají maximální velikost, při které se vejdou do okna).

Další rozvržení

V předcházejících podkapitolách jsme si ukázali pouze některé správce rozvržení. Dalším zajímavým rozložením s největší možností nastavení je GridBagLayout. Toto rozvržení je však příliš složité. Pro jednoduché aplikace stačí kombinace předcházejících a pro složitější aplikace pravděpodobně použijete RAD prostředí, které použití layoutu vyřeší za vás.
Dalším rozvržením je CardLayout, pro vytvoření tzv. karet. Ale toto rozložení neposkytuje možnost vytvořit kartičky s oušky, přepínání mezi kartičkami musíme řešit pomocí tlačítek nebo rozbalovacího seznamu. Balík javax.swig nám poskytuje třídu JTabbedPane, která tyto problémy řeší za nás.

Kombinování správců rozvržení

Jednotlivá rozvržení lze v rámci jedné aplikace kombinovat. Pro kombinování se používají instance třídy JPanel. Tato třída má stejně jako třída JFrame metody add() a setLayout(), standardním rozvržením pro panel je ale FlowLayout. Tato komponenta však nemá žádné viditelné ohraničení a lze ji vložit do instance třídy JFrame nebo opět do instance třídy JPanel. Příklad kombinování layoutů najdete např. v podkapitole Textová oblast.

Nastavení vlastností komponent

Pro každou komponentu je možné nastavit několik obecných vlastností pomocí metod jejich společného předka – třídy Component. Některé z nich vidíte v tabulce.

metoda třídy Component	popis metody
Color **getBackground**()	Vrátí nastavení barvy pozadí jako instanci třídy Color.
void **setBackground**(Color)	Nastaví barvu pozadí komponenty.
Color **getForeground**()	Vrátí nastavení barvy popředí jako instanci třídy Color.
void **setForeground**(Color)	Nastaví barvu popředí komponenty.
Font **getFont**()	Vrátí informace o nastaveném fontu.
void **setFont**(Font)	Nastaví font.
Dimension **size**()	Vrátí aktuální velikost komponenty. Lze použít i metody size().width() a size().height() pro získání každého rozměru zvlášť.
void **setVisible**(boolean)	Skryje nebo zobrazí komponentu na obrazovce.
boolean **isVisible**()	Vrátí true nebo false podle nastavení viditelnosti.
void **setEnabled** (boolean)	Zpřístupní či znepřístupní komponentu, při znepřístupnění je komponenta viditelná, ale uživatel ji nemůže použít.
boolean **isEnabled** ()	Zjistí stav zpřístupnění

tabulka 22 Vybrané metody třídy Component

Pro komponenty ze swingu existuje i metoda **setToolTipText**(String s), pomocí které můžeme pro komponentu nastavit plovoucí nápovědu. Použití těchto metod si ukážeme v následujících kapitolách.

Třída Font

Pro nastavení vzhledu písma slouží třída Font z knihovny AWT, kterou lze použít ve swingu. Umožňuje nastavit vzhled písma, řez písma a velikost písma.

Pro Javu je definováno 5 logických fontů, jejichž existence je zaručena na všech platformách. Pro jednotlivé platformy je pak v souboru **font.properties** uloženo, které fonty reprezentují (soubor font.properties je uložen v adresáři jre/lib). V následující tabulce jsou uvedena jména logických fontů a jejich reprezentace ve Windows.

Logický font	odpovídající font ve Windows
Serif	Times New Roman
Sans-serif	Arial
Monospaced	Curier New
Dialog	Arial
DialogInput	Curier New

tabulka 23 Logické fonty Javy a jejich nastavení ve Windows

V konstruktoru fontu je možné použít i název fontu z platformy, na které pracujete, tedy např. Arial nebo Georgia. Přenositelnost na jiné platformy je však poté problematická.

Pro nastavení řezu písma má třída Font tři konstanty: PLANT, BOLD (tučné) a ITALIC (kurzíva).

Konstruktor třídy Font má následující tvar: **Font(String name, int style, int size)**. Chceme-li tedy definovat pro náš program písmo s logickým fontem Dialog, tučným a o velikosti 20, vytvoříme ho takto:

```
Font f1 = new Font ("Dialog",Font.BOLD,20).
```

Font s písmem Georgia (pokud tento font máte ve svém operačním systému), tučnou kurzívou a velikostí 14 vytvoříme takto:

```
Font f2 = new Font ("Georgia",Font.BOLD + Font.ITALIC,14)
```

Třída Color

Třída Color slouží k nadefinování barvy. Najdeme v ní několik konstant, které můžeme použít. Pokud např. chceme v aplikaci použít modré tlačítko, nadefinujeme ho pomocí konstanty takto:

```
JButton tlacitko = new JButton ("Tlacitko");
tlacitko.setBackground (Color.blue);
```

Třída Color má také několik konstruktorů pro "namíchání" vlastních barev. Nejčastěji využijete dva následující:

Color (int r, int g, int b), kde každý z parametrů je z intervalu 0 až 255

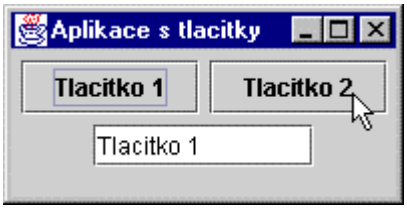
Color (int rgb)

Bílou barvu tedy můžeme "namíchat" takto:

```
Color barva1 = new Color(255,255,255);
Color barva2 = new Color(0xFF,0xFF,0xFF);
Color barva3 = new Color(0xFFFFFF);
```

Tlačítka a textové pole

Knihovna Swing nám poskytuje několik různých tlačítek, nejprve si ukážeme "obyčejné" tlačítko, tj. instanci třídy **JButton**. V následujícím programu použijeme dvě tlačítka a textové pole. Po stisknutí tlačítka se v textovém poli zobrazí text, který je na tlačítku.

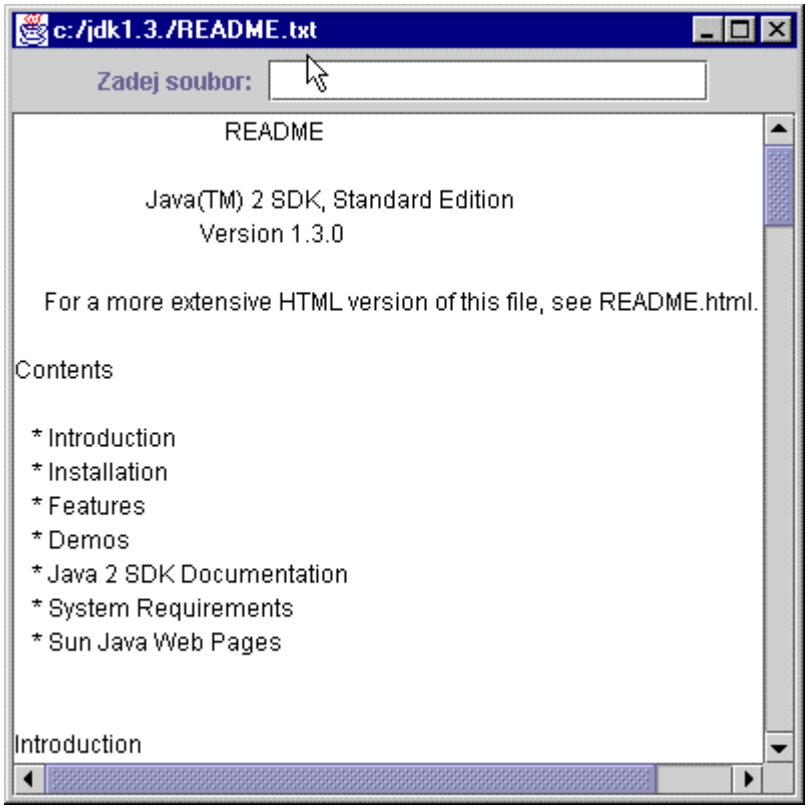


V konstruktoru tlačítka nastavíme text, který bude na tlačítku zobrazen, v konstruktoru textového pole (třída **JTextField**) nastavíme výchozí velikost (počet znaků, které lze zapsat). Použijeme FlowLayout a umístíme komponenty. Pro tlačítka musíme napsat ovladače a nastavit je jako "posluchače událostí". Tlačítko po stisknutí generuje událost **ActionEvent**, je tedy nutné napsat **vnitřní třídu AkceTlacitko**, která implementuje rozhraní **ActionListener**. ActionListener obsahuje jedinou metodu **actionPerformed (ActionEvent e)**, ve které musíme popsat činnost po stisknutí tlačítka. Pomocí metody události **e.getSource()** získáme zdroj události. Víme, že je to tlačítko, a tak ho přetypujeme **(JButton)e.getSource()**. Nakonec použijeme metodu **getText()**, která nám vrátí text zobrazovaný na tlačítku. Pomocí metody instance třídy JTextField **setText()** zobrazíme tento text v textovém poli. Pro každé tlačítko můžeme napsat jinou implementaci ActionListeneru. V našem příkladě je činnost obou tlačítek shodná, stačí tedy pouze jeden ovladač přiřazený oběma tlačítkům.

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

public class TlacitkaATextPole extends JFrame {
  JButton tlacitko1 = new JButton("Tlacitko 1"),
          tlacitko2 = new JButton("Tlacitko 2");
  JTextField txt = new JTextField(10);

  class AkceTlacitko implements ActionListener  {
    public void actionPerformed (ActionEvent e) {
      String jmenoTlacitka=((JButton)e.getSource()).getText();
      txt.setText(jmenoTlacitka);
    }
  };

  public TlacitkaATextPole (String nadpis) {

    super (nadpis);
    addWindowListener(new WindowAdapter () {
      public void windowClosing (WindowEvent e) {
        System.exit(0);
      }
    });
    getContentPane().setLayout(new FlowLayout());
    tlacitko1.addActionListener(new AkceTlacitko());
    tlacitko2.addActionListener(new AkceTlacitko());
    getContentPane().add(tlacitko1);
    getContentPane().add(tlacitko2);
    getContentPane().add(txt);
  };

  public static void main (String args []) {
     TlacitkaATextPole apl =
            new TlacitkaATextPole("Aplikace s tlacitky");
     apl.setSize(200,100);
     apl.setVisible(true);
  }
}
```

Textová oblast

V minulém příkladě jsme si ukázali textové pole, v tomto použijeme textovou oblast, v knihovně swing je to třída **JTextArea**. Textová oblast je vlastně víceřádkovým textovým polem a tudíž mnohé metody jsou stejné. Navíc obsahuje metodu **append()** pro přidávání textu za text, který už byl do textové oblasti vložen.



Našim úkolem je vytvořit program, který po zadání jména souboru zobrazí obsah tohoto souboru v textové oblasti. Horní řádek je složen z popisky (instance JLabel) a vstupního pole pro jméno souboru (instance JTextField). Obojí je vloženo do panelu (instance **JPanel**) – je to ukázka kombinování správců rozvržení (celé okno používá rozložení BorderLayout, v rámci panelu je použito rozložení FlowLayout). Obsah souboru se zobrazí v textové oblasti vytvořené instancí třídy JTextArea. Vzhledem k tomu, že obsah souboru může být větší než okno, je textová oblast „obalena“ posuvníkem (**JScrollPane**), který umožňuje posouvat se po souboru, který je větší než okno.

Další novinkou v tomto příkladě je obsluha JTextField – po zapsání jména souboru a stisknutí klávesy <Enter> chceme vypsat obsah souboru. Musíme tedy reagovat na události z klávesnice. Proto vytvoříme vnitřní třídu, která je potomkem třídy **KeyAdapter** a naimplementujeme metodu **keyPressed (KeyEvent e)**. Událost KeyEvent se generuje při každém stisknutí klávesy – proto musíme zjistit, jaká klávesa byla stisknuta (metoda **getKeyChar()**) a pouze v případě stisku klávesy <Enter> (konstanta **KeyEvent.VK_ENTER**) provedeme příslušnou akci, tj. zjistíme jméno souboru a spustíme výpis. Při jakékoli jiné klávese se neprovádíme nic (po stisknutí další klávesy se samozřejmě spustí obsluha automaticky znovu).

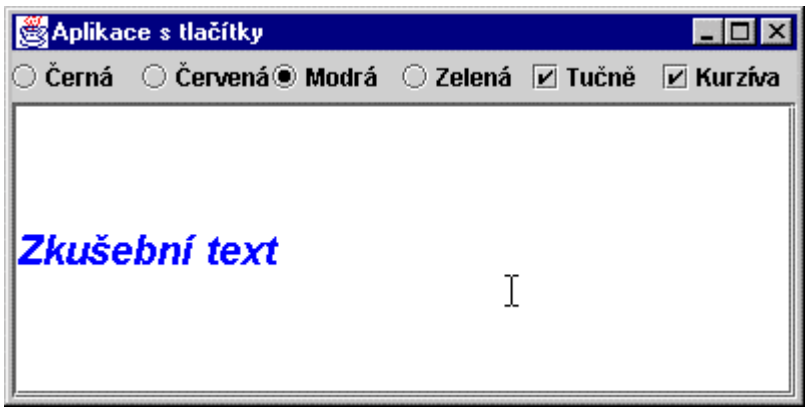
Metoda **zpracovani** otevře soubor, připraví oblast pro výpis (nastaví barvu textu na černou a vymaže předchozí výpis, lze vypisovat postupně několik souborů) a pomocí metody **append** přidává do textové oblasti jednotlivé řádky souboru. Pokud zadáte jméno neexistujícího souboru je odchycena výjimka a červeně zobrazeno chybové hlášení. Na závěr zpracování je uzavřen soubor.

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

public class TextovaOblast extends JFrame {
      JPanel zahlavi = new JPanel();
      JLabel textZadej = new JLabel ("Zadej soubor: ");
      JTextField zadejSoubor = new JTextField(20);
      JTextArea vypis = new JTextArea();

      class Kl extends KeyAdapter {
         public void keyPressed (KeyEvent e) {
           if (e.getKeyChar()==KeyEvent.VK_ENTER) {
             zpracovani(zadejSoubor.getText());
             zadejSoubor.setText("");
           }
         }
      };

      void zpracovani(String jmenoSouboru) {
         try {
           BufferedReader veta = new BufferedReader(new
                            FileReader(jmenoSouboru));
           String s = "";
           vypis.setForeground(Color.black);
           vypis.setText("");
           while ((s = veta.readLine()) != null){
             vypis.append(s+"\n");
           }
           veta.close();
           this.setTitle(jmenoSouboru);
         }
         catch (FileNotFoundException e) {
           vypis.setForeground(Color.red);
           vypis.setText("Soubor nebyl nalezen!!!!");
         }
         catch (IOException e ) {
           System.out.println("Chyba pri cteni");
         }
      }

      public TextovaOblast(String nadpis){
         super(nadpis);
         addWindowListener(
           new WindowAdapter() {
             public void windowClosing(WindowEvent e) {
               System.exit(0);
             }
           });
         zahlavi.add(textZadej);
         zahlavi.add(zadejSoubor);
         getContentPane().add(zahlavi,BorderLayout.NORTH);
         getContentPane().add(new JScrollPane(vypis));
         zadejSoubor.addKeyListener(new Kl());
      }
```

```
    public static void main(String[] args) {
       TextovaOblast apl = new TextovaOblast(
                           "Aplikace s textovou oblasti");
       apl.setLocation(100, 100);
       apl.setSize(400, 400);
       apl.setVisible(true);
    }
}
```

Zaškrtávací políčka a přepínače

Pro zaškrtávací políčka a přepínače nám knihovna swing poskytuje dvě třídy a to **JCheckBox** (zaškrtávací čtvereček) a **JRadioButton** (zaškrtávací kolečko). Instance obou těchto tříd lze použít k oběma účelům. Jestliže potřebujeme přepínače (tj. lze označit pouze jednu možnost ze skupiny), musíme použít instanci třídy **ButtonGroup** a jednotlivé možnosti do ní vložit.



Úkolem následujícího programu je pomocí zaškrtávacích políček nastavit vzhled písma v textovém poli. Nastavujeme kurzívu nebo tučné písmo (je tedy možné zaškrtnout jednu variantu, obě nebo žádnou) a barvu písma (je třeba vybrat právě jednu možnost). Pro barvy musíme tedy použít sloučení tlačítek do skupiny.

Připravíme si tedy čtyři instance třídy JCheckBox pro barvy, v konstruktoru určíme text u políčka. Obdobně vytvoříme dvě instance třídy JRadioButton. Připravíme si také fonty pro nastavování, skupinu tlačítek, panel pro umístění tlačítek a text pro nastavování vlastností.

V konstruktoru aplikace nastavíme správce rozmístění pro aplikaci a správce rozmístění pro panel tlačítek (použijeme GridLayout(1,6), tedy tabulkové rozložení s jedním řádkem a šesti sloupci). Do panelu a do skupiny tlačítek přidáme metodou add tlačítka pro volbu barev, do panelu přidáme i tlačítka pro volbu vzhledu písma. U tlačítek pro barvu určíme pomocí metody **setSelected()**, které bude zaškrtnuté při spuštění aplikace. U tlačítek pro vzhled písma neoznačíme žádné a nastavíme odpovídající font.

Instance obou druhů tlačítek generují při zaškrtnutí instanci třídy **ActionEvent**, pro ovládání napíšeme proto vnitřní třídy implementující rozhraní **ActionListener**. Pro každou barvu napíšeme vlastní ovladač, pro vzhled písma také jeden, abychom mohli zjistit kombinace zaškrtnutí.

Barvu textu v textové oblasti nastavíme pomocí metody **setForeground()** a font pomocí metody **setFont()**. Jestli je zaškrtávací políčko zaškrtnuto či ne zjistíme pomocí metody **isSelected()**.

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

public class RuznaTlacitka extends JFrame {

JCheckBox tucne = new JCheckBox ("Tučně");
JCheckBox kurziva = new JCheckBox ("Kurzíva");
JRadioButton cerna = new JRadioButton("Černá");
JRadioButton cervena = new JRadioButton("Červená");
JRadioButton modra = new JRadioButton("Modrá");
JRadioButton zelena = new JRadioButton("Zelená");
ButtonGroup skupinaTlacitek = new ButtonGroup();
JPanel panelTlacitek = new JPanel();
JTextField text = new JTextField("Zkušební text");
Font f1 = new Font ("Dialog",Font.PLAIN,20);
Font f2 = new Font ("Dialog",Font.BOLD,20);
Font f3 = new Font ("Dialog",Font.ITALIC,20);
Font f4 = new Font ("Dialog",Font.BOLD+Font.ITALIC,20);

      class NastavModrou implements ActionListener {
         public void actionPerformed (ActionEvent e) {
           text.setForeground(Color.blue);
         }
      };

      class NastavCernou implements ActionListener {
         public void actionPerformed (ActionEvent e) {
           text.setForeground(Color.black);
         }
      };

      class NastavCervenou implements ActionListener {
         public void actionPerformed (ActionEvent e) {
           text.setForeground(Color.red);
         }
      };

      class NastavZelenou implements ActionListener {
         public void actionPerformed (ActionEvent e) {
           text.setForeground(Color.green);
         }
      };

      class ZmenaPisma implements ActionListener {
         public void actionPerformed(ActionEvent e){
           if(tucne.isSelected() && kurziva.isSelected())
             text.setFont(f4);
           if(tucne.isSelected() && (!kurziva.isSelected()))
             text.setFont(f2);
           if((!tucne.isSelected()) && kurziva.isSelected())
             text.setFont(f3);
           if((!tucne.isSelected()) && (!kurziva.isSelected()))
             text.setFont(f1);
         }
      };
```

```
public RuznaTlacitka(String nadpis) {
      super (nadpis);
      addWindowListener(new WindowAdapter () {
         public void windowClosing (WindowEvent e) {
           System.exit(0);
         }
      });
      panelTlacitek.setLayout(new GridLayout(1,6));
      getContentPane().setLayout(new BorderLayout());
      skupinaTlacitek.add(cerna);
      skupinaTlacitek.add(cervena);
      skupinaTlacitek.add(modra);
      skupinaTlacitek.add(zelena);
      panelTlacitek.add(cerna);
      panelTlacitek.add(cervena);
      panelTlacitek.add(modra);
      panelTlacitek.add(zelena);
      panelTlacitek.add(tucne);
      panelTlacitek.add(kurziva);
      modra.addActionListener(new NastavModrou());
      cerna.addActionListener(new NastavCernou());
      cervena.addActionListener(new NastavCervenou());
      zelena.addActionListener(new NastavZelenou());
      tucne.addActionListener(new ZmenaPisma());
      kurziva.addActionListener(new ZmenaPisma());
      cerna.setSelected(true);
      text.setFont(f1);
      getContentPane().add( panelTlacitek,BorderLayout.NORTH);
      getContentPane().add(new JScrollPane(text));

}

public static void main (String args[]) {
      RuznaTlacitka a = new RuznaTlacitka("Aplikace s tlačítky");
      a.setLocation(100,100);
      a.setSize(400,200);
      a.setVisible(true);
}
}
```

Rozbalovací seznam

Další komponenta, jejíž použití si ukážeme, je rozbalovací seznam, pro který má knihovna swing třídu **JComboBox**. JComboBox umožňuje vytvořit seznam více položek a vybrat z nich právě jednu. Nabídka seznamu se rozvine po stisknutí ovládací šipky.

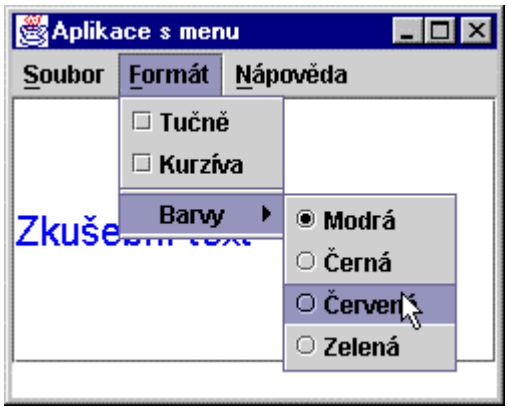
Následující program umožní změnit barvu pozadí aplikace podle výběru z rozbalovacího seznamu. Třída JComboBox má konstruktor, který nastaví jednotlivé možnosti nabídky z pole řetězců. Proto si připravíme pole s jednotlivými texty nabídek a pole s odpovídajícími barvami.

Vytvoříme instanci třídy JComboBox a přidáme ji do okna aplikace a nastavíme barvu pozadí podle seznamu (pokud neurčíte jinak bude vybrána první možnost).
Také instance třídy JComboBox generují **ActionEvent**. Je tedy třeba napsat ovladač implementující ActionListener a přiřadit ho rozbalovacímu seznamu pomocí **addActionListener**.
V ovladači je třeba zjistit, která volba byla vybrána, pomocí metody **getSelectedIndex()**. Tato metoda vrací index (pořadí) vybrané volby, první volba má index 0. Na základě indexu zvolíme položku z pole barev a použijeme ji k nastavení barvy aplikace. Odkaz na aplikaci se zapíše pomocí konstrukce **RozbalovaciSeznam.this**. Pouhé this nestačí, neboť jsme uvnitř vnitřní třídy a this tu označuje tuto vnitřní třídu. Nakonec musíme aplikaci znovu zobrazit s novou barvou.

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

public class RozbalovaciSeznam extends JFrame {

     String []barvy = {"lightgray","black","blue",
                       "cyan","gray","green","red","white"};
     Color [] konstanty = {Color.lightGray,Color.black,
                       Color.blue,Color.cyan,Color.gray,
                       Color.green,Color.red,Color.white};
     JComboBox vyber = new JComboBox(barvy);

     class Al implements ActionListener {
        public void actionPerformed (ActionEvent e){
          int index = vyber.getSelectedIndex();
          RozbalovaciSeznam.this.getContentPane().
                              setBackground(konstanty[index]);
          RozbalovaciSeznam.this.setVisible(true);
        }
     }

     RozbalovaciSeznam (String nazev){
        super(nazev);
        addWindowListener(new WindowAdapter () {
          public void windowClosing (WindowEvent e) {
            System.exit(0);
          }
        });
        getContentPane().add(vyber,BorderLayout.NORTH);
        vyber.addActionListener(new Al());
        vyber.setSelectedIndex(5);
        int index = vyber.getSelectedIndex();
        this.getContentPane().setBackground(konstanty[index]);
     }
```

```
    public static void main (String []args) {

        RozbalovaciSeznam apl = new RozbalovaciSeznam("Aplikace s
rozbalovacim seznamen");
        apl.setLocation(100,100);
        apl.setSize(180,250);
        apl.setVisible(true);
    }
}
```

Menu



Pro tvorbu menu nám knihovna swing poskytuje řadu tříd. Třída **JMenuBar** umožňuje vytvořit prázdnou lištu pro menu v horní části aplikace (bez závislosti na správci rozvržení). Pro tvorbu položek menu, které obsahují další položky, slouží třída **JMenu**. Pro vytvoření položek menu, které již vyvolají nějakou činnost programu, slouží třída **JMenuItem**. Třídy **JCheckBoxMenuItem** a **JRadioButtonMenuItem** nám poskytují možnost vytvořit zaškrtávací položky v menu – podobně jako u tlačítek se po zařazení do skupiny mohou chovat jako přepínače.

Vytvoříme program s menu, v horní liště budou volby Soubor, Formát a Nápověda. Volba Soubor bude obsahovat volby Otevřít, Uložit a Konec. Volba Nápověda volbu O aplikaci. Ve volbě Formát budou dvě zaškrtávací volby pro tučné písmo a kurzívu (mohou být zvoleny obě, jedna nebo nic) a další menu Barvy. Menu Barvy bude obsahovat čtyři zaškrtávací volby fungující jako přepínače (tj. musí být zvolena právě jedna možnost).

Jinak bude aplikace obsahovat pouze textové pole se zkušebním textem pro ověření funkce některých nabídek.

Na začátku si tedy musíme připravit jednotlivé položky menu, textové pole a fonty pro nastavování. Pro přepínání barev je nutné vytvořit skupinu tlačítek (instanci třídy **ButtonGroup**).

V konstruktoru musíme nastavit instanci třídy JMenuBar jako lištu naší aplikace pomocí metody **setJMenuBar()**. Pro každou položku menu je možné pomocí metody **setMnemonic(KeyEvent.*klávesa*)** nastavit možnost aktivovat položku menu stisknutím klávesy ALT a podtrženého písmene v menu. Konstanty pro jednotlivé klávesy najdete v nápovědě u třídy KeyEvent s balíku awt.event. Pomocí metody **add()** pak vytvoříme jednotlivá menu, položky vkládáme v pořadí, v jakém mají být zobrazeny. Pokud chceme mezi jednotlivé položky vložit oddělovač, použijeme metodu **addSeparator().** Pokud chceme pro některou položku, která je instanci JMenuItem (tj. spouští činnost neotevírá jiné menu), nastavit kombinaci kláves pro provedení akce (např. pro otevření souboru CTRL+O), použijeme metodu **setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,ActionEvent.CTRL_MASK))**.

Pokud mají volby pro barvy fungovat jako přepínače, musíme je pomocí metody **add()** vložit také do instance třídy **ButtonGroup**.

Je třeba přiřadit a napsat odpovídající ovladače. Instance tříd JMenuItem a JRadioMenuItem při zvolení nabídky generují události typu **ActionEvent** a instance třídy JCheckBoxMenuItem události **ItemEvent**. Pro obsluhu událostí je třeba napsat ovladače implementující odpovídající rozhraní. Rozhraní **ActioListener** má metodu **actionPerformed(ActionEvent e)** a rozhraní **ItemListener** metodu **itemStateChanged(ItemEvent e)**. Nastavení barev a fontů pak provedeme stejným způsobem jako v kapitole o zaškrtávacích polích.

Ovladače pro soubory a nápovědu si popíšeme v následujících kapitolách (zdrojový text je obsahuje)

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

public class RuznaMenu extends JFrame {

      JMenuBar lista = new JMenuBar();
      JMenu soubor = new JMenu("Soubor");
      JMenu  format = new JMenu("Formát");
      JMenu napoveda = new JMenu ("Nápověda");
      JMenu barvy = new JMenu("Barvy");
      JMenuItem otevrit = new JMenuItem("Otevřít");
      JMenuItem ulozit = new JMenuItem("Uložit");
      JMenuItem konec = new JMenuItem("Konec");
      JMenuItem oAplikaci = new JMenuItem ("O aplikaci");
      JCheckBoxMenuItem tucne = new JCheckBoxMenuItem("Tučné");
      JCheckBoxMenuItem kurziva = new
                                  JCheckBoxMenuItem("Kurzíva");
      JRadioButtonMenuItem modra = new
                                  JRadioButtonMenuItem("Modrá");
      JRadioButtonMenuItem cerna = new
                                  JRadioButtonMenuItem("Černá");
      JRadioButtonMenuItem cervena = new
                                  JRadioButtonMenuItem("Červená");
      JRadioButtonMenuItem zelena = new
                                  JRadioButtonMenuItem("Zelená");
      ButtonGroup skupina = new ButtonGroup();
      JTextField text = new JTextField("Zkušební text");
      JTextArea souborAAdresar = new JTextArea();
      Font f1 = new Font("DialogInput",Font.PLAIN,20);
      Font f2 = new Font ("DialogInput",Font.BOLD,20);
      Font f3 = new Font ("DialogInput",Font.ITALIC,20);
      Font f4 = new Font ("DialogInput",Font.BOLD+Font.ITALIC,20);

      class VyberSouboru implements ActionListener {
         public void actionPerformed (ActionEvent e) {
           JFileChooser fc = new JFileChooser();
           int stavVyberu = fc.showOpenDialog(RuznaMenu.this);
           if (stavVyberu ==JFileChooser.APPROVE_OPTION){
             souborAAdresar.setText("Vybrany soubor: "
                 +fc.getSelectedFile().getName()+"\n");
```

```
        souborAAdresar.append("Vybrany adresar: "
            +fc.getCurrentDirectory().toString());
      }
      else
        souborAAdresar.setText(
                      "Akce vyberu souboru stornovana");
      RuznaMenu.this.repaint();
    }
};

class UlozeniSouboru implements ActionListener {
    public void actionPerformed (ActionEvent e) {
      JFileChooser fc = new JFileChooser();
      fc.showSaveDialog(RuznaMenu.this);
      RuznaMenu.this.repaint();
    }
};

class Ukonceni implements ActionListener {
    public void actionPerformed (ActionEvent e) {
      System.exit(0);
    }
};

class MujItemAdapter implements ItemListener {
    public void itemStateChanged(ItemEvent e){
      if(tucne.isSelected() && kurziva.isSelected())
        text.setFont(f4);
      if(tucne.isSelected() && (!kurziva.isSelected()))
        text.setFont(f2);
      if((!tucne.isSelected()) && kurziva.isSelected())
        text.setFont(f3);
      if((!tucne.isSelected()) && (!kurziva.isSelected()))
        text.setFont(f1);
    }
};

class NastavModrou implements ActionListener {
    public void actionPerformed (ActionEvent e) {
      text.setForeground(Color.blue);
    }
};

class NastavCernou implements ActionListener {
    public void actionPerformed (ActionEvent e) {
      text.setForeground(Color.black);
    }
};

class NastavCervenou implements ActionListener {
    public void actionPerformed (ActionEvent e) {
      text.setForeground(Color.red);
    }
};
```

```
class NastavZelenou implements ActionListener {
   public void actionPerformed (ActionEvent e) {
     text.setForeground(Color.green);
   }
};

class OtevreniDialogu implements ActionListener {
   public void actionPerformed (ActionEvent e) {
     JOptionPane.showMessageDialog(null,
           "informace o programu", "O Programu",
           JOptionPane.INFORMATION_MESSAGE);
   }
};

public RuznaMenu(String nadpis){
   super(nadpis);
   addWindowListener(
     new WindowAdapter() {
       public void windowClosing(WindowEvent e) {
         System.exit(0);
       }
     });
   setJMenuBar(lista);
   soubor.setMnemonic(KeyEvent.VK_S);
   format.setMnemonic(KeyEvent.VK_F);
   napoveda.setMnemonic(KeyEvent.VK_N);
   lista.add(soubor);
   lista.add(format);
   lista.add(napoveda);
   soubor.add(otevrit);
   otevrit.setAccelerator(KeyStroke.getKeyStroke(
                     KeyEvent.VK_O,ActionEvent.CTRL_MASK));
   otevrit.addActionListener(new VyberSouboru());
   soubor.add(ulozit);
   ulozit.addActionListener(new UlozeniSouboru());
   soubor.addSeparator();
   soubor.add(konec);
   konec.addActionListener(new Ukonceni());
   MujItemAdapter mia = new MujItemAdapter();
   tucne.addItemListener(mia);
   kurziva.addItemListener(mia);
   format.add(tucne);
   format.add(kurziva);
   modra.addActionListener(new NastavModrou());
   cerna.addActionListener(new NastavCernou());
   cervena.addActionListener(new NastavCervenou());
   zelena.addActionListener(new NastavZelenou());
   cerna.setSelected(true);
   format.addSeparator();
   format.add(barvy);
   barvy.add(modra);
   skupina.add(modra);
   barvy.add(cerna);
   skupina.add(cerna);
   barvy.add(cervena);
   skupina.add(cervena);
```

```
        barvy.add(zelena);
        skupina.add(zelena);
        napoveda.add(oAplikaci);
        oAplikaci.addActionListener(new OtevreniDialogu());
        getContentPane().add(text);
        getContentPane().add(souborAAdresar,
                                    BorderLayout.SOUTH);
        text.setFont(f1);
    }

    public static void main(String[] args) {
        RuznaMenu apl = new RuznaMenu("Aplikace s menu");
        apl.setLocation(100, 100);
        apl.setSize(400, 400);
        apl.setVisible(true);
    }
}
```

Okna pro otevírání a ukládání souborů

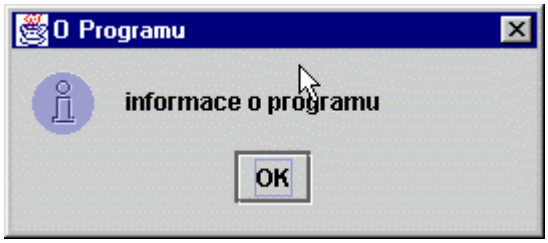
Pro otevírání a ukládání souboru poskytuje knihovna swing třídu **JFileChooser**. V minulém programu s menu byly volby Soubor-Otevrit a Soubor-Ulozit. Pro jejich obsluhu byly napsány ovladače implementující ActionListener. V jejich metodě actionPerformed je popsáno základní použití oken pro otevírání a ukládání souborů. Vytvoříme tedy instanci třídy JFileChooser a pomocí metody **showOpenDialog()** či **showSaveDialog()** otevřeme příslušné okno. Jako parametr vložíme odkaz na aplikaci, která okno otevírá (opět je nutná konstrukce *jménonadřízenétřídy*.this, pouhé this odkazuje na ovladač, ne na aplikaci). Do proměnné stavVyberu se vrátí informace o tom, jak uživatel dialog ukončil. Pokud je vrácena hodnota odpovídající konstantě **APPROVE_OPTION**, uživatel vybral soubor, pokud se rovná konstantě **CANCEL_OPTION**, uživatel akci stornoval. Pomocí metody **getSelectedFile()** získáme uživatelem vybraný soubor (jako instanci třídy File) metoda **getCurrentDirectory()** vrací vybraný adresář (jako instanci třídy File). Pokud nebyl soubor nebo adresář vybrán vracejí metody hodnotu null. Toto okno je standardně nastaveno jako modální tj. uživatel musí reagovat na toto okno, hlavní okno aplikace je neaktivní a nelze se do něj přepnout. V ovladači našeho programu pouze zobrazujeme jméno vybraného souboru a adresáře.

```
    class VyberSouboru implements ActionListener {
        public void actionPerformed (ActionEvent e) {
          JFileChooser fc = new JFileChooser();
          int stavVyberu = fc.showOpenDialog(RuznaMenu.this);
          if (stavVyberu ==JFileChooser.APPROVE_OPTION){
             souborAAdresar.setText("Vybrany soubor:
                    "+fc.getSelectedFile().getName()+"\n");
            souborAAdresar.append("Vybrany adresar:
                    "+fc.getCurrentDirectory().toString());
          }
```

```
            else
              souborAAdresar.setText("Akce
                              vyberu souboru stornovana");
            RuznaMenu.this.repaint();
          }
      };
```

Informační okna

V kapitole o menu jsme použili volbu O aplikaci. Po jejím vybrání se zobrazí jednoduché informační okno. Pro jeho vytvoření jsme použili třídu **JOptionPane** a její metodu **showMessageDialog**, která jako parametry používá odkaz na rodiče (může být i prázdný), text, který bude zobrazen v okně, titulek okna a konstantu označující ikonu, která se zobrazí. Na obrázku vidíte výsledné okno naší aplikace a následuje kód, který toto okno definuje.




```
JOptionPane.showMessageDialog(null,
                "informace o programu", "O Programu",
                JOptionPane.INFORMATION_MESSAGE);
```

Třída JOptionPane umožňuje vytvářet i složitější informační okna. Poskytuje 4 metody pro jejich konfiguraci:

showConfirmDialog – poskytuje uživateli možnost vybrat jednu z variant typu yes/no/cancel.

showInputDialog – prompt pro vstup

showMessageDialog – sděluje uživateli zprávu (viz náš příklad)

showOptionDialog – kombinace předcházejících možností

Všechny zde uvedené metody jsou několikrát přetíženy a umožňují vytvořit i informační okna v češtině, protože i nápisy na tlačítkách jsou parametrizovány.

V této třídě je také definováno 5 konstant pro ikony, které se mohou v informačním okně zobrazit (INFORMATION_MESSAGE, ERROR_MESSAGE, WARNING_MESSAGE, QUESTION_MESSAGE, PLAIN_MESSAGE).

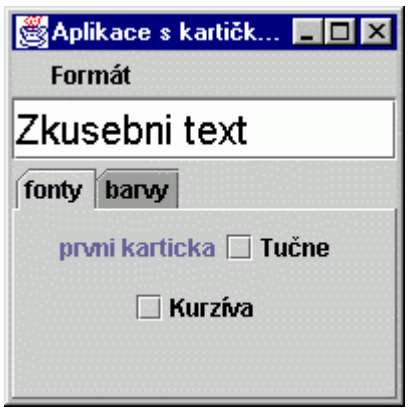
Ukážeme si jak vytvořit a obsloužit okno poskytující možnosti volby ano/ne/zrušit. Použijeme metodu **showOptionDialog**, prvním parametrem je odkaz na rodičovské okno. Dalšími parametry jsou text uvnitř okna a text v záhlaví okna. Pak následuje konstanta určující zda jde o dialog se dvěma tlačítky (hodnota YES_NO_OPTION) nebo se třemi tlačítky (YES_NO_CANCEL_OPTION). Další parametr určuje použitou ikonu. Další parametr je typu icon a umožňuje použít jiný obrázek než předdefinované ikony, pokud použijeme

standardní uvedeme prázdnou hodnotu null. Předposlední parametr určuje pole prvků typu Object, ze kterého se berou texty na tlačítka. Pokud pole neosahuje stringy ale instance jiné třídy použije se metoda toString. Poslední parametr určuje, která volba má přidělen fokus (tj. je nastavena jako aktivní) při otevření okna.

Po uzavření okna do proměnné navrat získáme hodnotu tlačítka, které uživatel stisknul. Pro návratové hodnoty jsou ve třídě JOptionPane definovány konstanty YES_OPTION, NO_OPTION, CANCEL_OPTION, CLOSED_OPTION a OK_OPTION. V našem příkladě jen vypíšeme výběr v okně. Tuto část kódu si můžete vyzkoušet tak, že třídu OtevreniDialogu v příkladě u menu nahradíte tou následující.

```
class OtevreniDialogu implements ActionListener {
      public void actionPerformed (ActionEvent e) {
         Object[] moznosti = {"Ano","Ne","Zrušit"};
         int navrat = JOptionPane.showOptionDialog(RuznaMenu.this,
           "Uložit soubor?",
           "Dotaz",
           JOptionPane.YES_NO_CANCEL_OPTION,
           JOptionPane.WARNING_MESSAGE,
           null,
           moznosti,
           moznosti[0]);
         if (navrat == JOptionPane.YES_OPTION)
                text.setText("vybrano Ano");
         if (navrat == JOptionPane.NO_OPTION)
                text.setText("vybrano Ne");
         if (navrat == JOptionPane.CANCEL_OPTION ||
                                navrat == JOptionPane.CLOSED_OPTION)
                text.setText("akce zrušena ");
      }
};
```

Kartičky



Jak už bylo zmíněno v kapitole o rozložení, poskytuje package javax.swing třídu JTabbedPane pro tvorbu kartiček. Obměníme příklad s tlačítky a tlačítka pro nastavení řezu písma umístíme na jednu kartičku a tlačítka pro barvy na druhou. Kartičky se zobrazí po volbě z menu.

Kartičky vytvoříme pomocí konstruktoru **JTabbedPane()**. Jednotlivé záložky pak vytvoříme pomocí metody **addTab(String napisNaZalozce ,JComponent c)**, první parametr udává nápis na záložce, druhý komponentu do kartičky umístěnou. Chceme-li na jednu kartičku umístit více prvků, použijeme panel. Do panelu umístíme komponenty a panel pak vložíme do kartičky. Podle počtu použití metody addTab se vytvoří odpovídající počet záložek. Aktivní kartičku tj. kartičku, která je při zobrazení v popředí, určíme

pomocí metody **setSelectedIndex(int poradi)**, kde parametr určuje index karty. Karty jsou indexovány podle pořadí vytvoření metodou addTab, první karta má index 0.

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

public class Karticky extends JFrame {

JMenuBar mb = new JMenuBar();
JMenuItem m = new JMenuItem("Formát");
JLabel napis1 = new JLabel ("prvni karticka");
JCheckBox tucne = new JCheckBox ("Tučne");
JCheckBox kurziva = new JCheckBox ("Kurzíva");
JPanel panel1 = new JPanel();
Font f1 = new Font("Dialog",Font.PLAIN,20);
Font f2 = new Font ("Dialog",Font.BOLD,20);
Font f3 = new Font ("Dialog",Font.ITALIC,20);
Font f4 = new Font ("Dialog",Font.BOLD+Font.ITALIC,20);
JLabel napis2 = new JLabel ("druha karticka");
JRadioButton cerna = new JRadioButton("Černá");
JRadioButton cervena = new JRadioButton("Červená");
JRadioButton modra = new JRadioButton("Modrá");
JRadioButton zelena = new JRadioButton("Zelená");
ButtonGroup skupinaTlacitek = new ButtonGroup();
JPanel panel2 = new JPanel();
JTabbedPane tabbedPane = new JTabbedPane();
JTextField text = new JTextField("Zkusebni text");

     class NastavModrou implements ActionListener {
        public void actionPerformed (ActionEvent e) {
          text.setForeground(Color.blue);
        }
     };

     class NastavCernou implements ActionListener {
        public void actionPerformed (ActionEvent e) {
          text.setForeground(Color.black);
        }
     };

     class NastavCervenou implements ActionListener {
        public void actionPerformed (ActionEvent e) {
          text.setForeground(Color.red);
        }
     };

     class NastavZelenou implements ActionListener {
        public void actionPerformed (ActionEvent e) {
          text.setForeground(Color.green);
        }
     };
```

```
      class ZmenaPisma implements ActionListener {
         public void actionPerformed(ActionEvent e){
           if(tucne.isSelected() && kurziva.isSelected())
             text.setFont(f4);
           if(tucne.isSelected() && (!kurziva.isSelected()))
             text.setFont(f2);
           if((!tucne.isSelected()) && kurziva.isSelected())
             text.setFont(f3);
           if((!tucne.isSelected()) && (!kurziva.isSelected()))
             text.setFont(f1);
         }
      };

      class ZobrazKarticky implements ActionListener {
         public void actionPerformed (ActionEvent e) {
           Karticky.this.getContentPane().add(tabbedPane,
                      BorderLayout.CENTER);
           Karticky.this.setVisible(true);
         }
      };

public Karticky(String nazev) {
      super(nazev);
      addWindowListener(new WindowAdapter () {
         public void windowClosing (WindowEvent e) {
           System.exit(0);
         }
});
      setJMenuBar(mb);
      mb.add(m);
      m.addActionListener(new ZobrazKarticky());
      panel1.setLayout(new FlowLayout());
      tucne.addActionListener(new ZmenaPisma());
      kurziva.addActionListener(new ZmenaPisma());
      panel1.add(napis1);
      panel1.add(tucne);
      panel1.add(kurziva);
      tabbedPane.addTab("fonty",panel1);
      tabbedPane.setSelectedIndex(0);
      panel2.setLayout(new FlowLayout());
      skupinaTlacitek.add(cerna);
      skupinaTlacitek.add(cervena);
      skupinaTlacitek.add(modra);
      skupinaTlacitek.add(zelena);
      panel2.add(napis2);
      panel2.add(cerna);
      panel2.add(cervena);
      panel2.add(modra);
      panel2.add(zelena);
      modra.addActionListener(new NastavModrou());
      cerna.addActionListener(new NastavCernou());
      cervena.addActionListener(new NastavCervenou());
      zelena.addActionListener(new NastavZelenou());
      cerna.setSelected(true);
      tabbedPane.addTab("barvy",panel2);
```

```
      text.setFont(f1);
      getContentPane().add(text,BorderLayout.NORTH);
}

public static void main (String []args) {
      Karticky apl = new Karticky("Aplikace s kartičkami");
      apl.setLocation(100,100);
      apl.setSize(200,200);
      apl.setVisible(true);
}}
```

Další komponenty GUI

Ukázali jsme si použití pouze některých komponent, které nám Java v knihovnách AWT a swing nabízí. Další zajímavé komponenty jsou např.

- JToolBar – pro tvorbu panelů nástrojů
- JProgressBar – pro sledování průběhu akce např. načítání souboru
- JDialog – pro vytvoření dalšího okna v aplikaci s libovolným obsahem

Pro tvorbu rozsáhlého uživatelského rozhraní je výhodné použít tzv. RAD (rapid aplication development) nástroje jako jsou pro Javu např. NetBeans, Forte, JBuilder nebo Visual Java.