Možnosti zobrazení textu

266 Vykreslení prostého textu

Ukázka využití tříd z balíčku java.awt.

Postupujte takto:

```
public void paint(Graphics g) {
  // Chcete-li použít jiné než implicitní písmo, nastavte je.
  String family = "Serif";
  int style = Font.PLAIN;
  int size = 12;
  Font font = new Font(family, style, size);
  g.setFont(font);

  // Nakreslete řetězec tak, aby jeho účaří bylo na souřadnicích x, y
  int x = 10;
  int y = 10;
  g.drawString("řetězec", x, y);

  // Nakreslete řetězec tak, aby levý horní roh textu
  // byl na souřadnicích x, y.
  x = 10;
  y = 30;
  FontMetrics fontMetrics = g.getFontMetrics();
  g.drawString("řetězec", x, y+fontMetrics.getAscent());
}
```

267 Přirozenější proložení znaků

Uživatelé od dnešních aplikací očekávají, že si v nich budou moci nastavit vlastní atributy textu (typ písma, barva, velikost apod.). V tomto příkladu ukazujeme, jak nastavit proložení znaků.

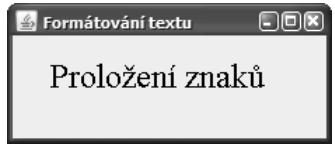
```
public void paint(Graphics g) {

  Hashtable<TextAttribute, Object> map =
    new Hashtable<TextAttribute, Object>();

  Font font = new Font(Font.SERIF, Font.PLAIN, 24);

  map.put(TextAttribute.KERNING, TextAttribute.KERNING_ON);
  font = font.deriveFont(map);
  g.setFont(font);
```

```
  String text = "Proložení znaků";
  g.drawString(text, 30, 60);
}
```



268 Podtržení textu

Chcete-li zdůraznit určitou pasáž textu, můžete jí přidat atribut **podtržení**:

```
public void paint(Graphics g) {

  Hashtable<TextAttribute, Object> map =
    new Hashtable<TextAttribute, Object>();

  Font font = new Font(Font.SERIF, Font.PLAIN, 24);

  map.put(TextAttribute.UNDERLINE, TextAttribute.UNDERLINE_ON);
  font = font.deriveFont(map);
  g.setFont(font);
  String text = "Podtržený text";
  g.drawString(text, 30, 60);
}
```

269 Přeškrtnutí textu

Přeškrtnutý text se výborně hodí v aplikacích se zabudovanou podporou sledovaných změn, v nichž chcete zobrazit již odstraněný text. Tento atribut písma lze však využít také v jiných oknech nebo formulářích, v nichž chcete uživateli grafickou formou sdělit, která část textu bude po aplikaci změn odstraněna nebo dále neplatná.

```
public void paint(Graphics g) {

  Hashtable<TextAttribute, Object> map =
    new Hashtable<TextAttribute, Object>();

  Font font = new Font(Font.SERIF, Font.PLAIN, 24);

  map.put(TextAttribute.STRIKETHROUGH, TextAttribute.STRIKETHROUGH_ON);
```

```
  font = font.deriveFont(map);
  g.setFont(font);
  String text = "Přeškrtnutý text";
  g.drawString(text, 30, 60);
}
```

270 Změna barvy textu

Možnost změny barvy textu je v dnešní době již zcela běžným požadavkem uživatelů při práci s textem. Text v různých barvách ovšem lze využít také při prezentaci různých typů formulářů, v nichž vyplnění určitých polí je volitelné nebo povinné. Poznámky a vysvětlivky také je někdy vhodné zobrazit raději odlišnou barvou než písmem jiné velikosti.


```
public void paint(Graphics g) {

  Hashtable<TextAttribute, Object> map =
    new Hashtable<TextAttribute, Object>();

  Font font = new Font(Font.SERIF, Font.PLAIN, 24);

  map.put(TextAttribute.FOREGROUND, Color.BLUE);
  font = font.deriveFont(map);
  g.setFont(font);
  String text = "Modrý text";
  g.drawString(text, 30, 60);
}
```

271 Změna atributu textu u části textu

Ukázka využití tříd z balíčku java.text.

V mnohých aplikacích se vám bude hodit schopnost změnit atributy jen u vybrané části textu. Zvládnete-li mechanismus naznačený v tomto příkladu, budete moci uživateli nabídnout například možnost vlastního formátování textu. Třída `AttributedString` je zapouzdřením formátovaného textu, v němž lze podřetězcům nastavovat libovolné atributy. Atribut se skládá z názvu, hodnoty a intervalu znaků, na něž má být použit.

V tomto příkladu změníme barvu prvního slova v textu.


```
public void paint(Graphics g) {
  Graphics2D g2d = (Graphics2D) g;

  AttributedString text = new AttributedString("Barevný text");
  text.addAttribute(TextAttribute.FOREGROUND, Color.red, 0, 7);

  AttributedCharacterIterator textIterator = text.getIterator();
  FontRenderContext fontContext =
    new FontRenderContext(null, false, false);
  TextLayout textLayout = new TextLayout(textIterator, fontContext);

  textLayout.draw(g2d, 100, 50);
}
```

272 Rozměry textu v nakresleném objektu

Ukázka využití tříd z balíčku java.awt.

Před vykreslením textu na povrch ovládacího prvku je často dobré zjistit, kolik místa bude třeba k tomu, aby nedošlo k oseknutí textu.

```
// Takto lze rozměry textu zjistit z těla metody paint().
public void paint(Graphics g) {
  Graphics2D g2d = (Graphics2D)g;
  Font font = new Font("Serif", Font.PLAIN, 12);
  FontMetrics fontMetrics = g2d.getFontMetrics();

  int width = fontMetrics.stringWidth("text");
  int height = fontMetrics.getHeight();
}

// Takto lze rozměry textu zjistit uvnitř komponenty.
class NovaKomponenta extends JComponent {
  NovaKomponenta() {
    Font font = new Font("Serif", Font.PLAIN, 12);
    FontMetrics fontMetrics = getFontMetrics(font);

    int width = fontMetrics.stringWidth("text");
    int height = fontMetrics.getHeight();
  }
}
```

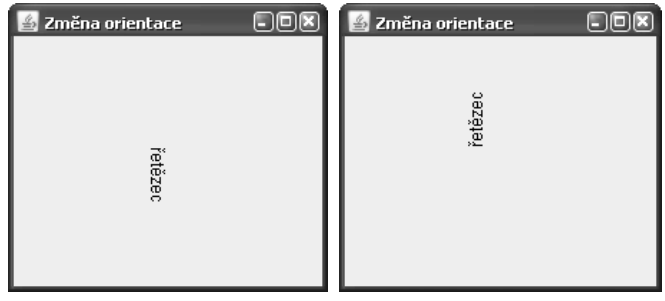
273 Změna orientace textu v nakreslených objektech

Ukázka využití tříd z balíčku java.awt.

Orientaci textu v nakreslených objektech lze změnit tak, aby byl text zobrazen svisle, vodorovně nebo pod určitým úhlem. Potřebujete-li změnit orientaci textu, použijte následující postup.

```
// Nakreslete řetězec otočený o 90 stupňů doprava.
AffineTransform at = new AffineTransform();
at.setToRotation(Math.toRadians(90));
g2d.setTransform(at);
g2d.drawString("řetězec", x, y);

// Nakreslete řetězec otočený o 90 stupňů doleva.
at = new AffineTransform();
at.setToRotation(Math.toRadians(-90));
g2d.setTransform(at);
g2d.drawString("řetězec", x, y);
```



274 Změna orientace textu v nakreslených objektech

Ukázka využití tříd z balíčku java.awt.

Pozici textu na nakresleném objektu (formuláři) můžete určit pomocí souřadnic `x` a `y`. Absolutní souřadnice jsou dobrá věc, pokud se dobře orientujete v geometrii. Chcete-li využít možnosti posunu počátku soustavy souřadnic, použijte metodu `translate()` definovanou ve třídě `AffineTransform`:

```
// Nakreslete řetězec otočený o 90 stupňů doprava.
AffineTransform at = new AffineTransform();
at.setToRotation(Math.toRadians(90));
at.translate(-100, 100);
g2d.setTransform(at);
g2d.drawString("řetězec", 0, 0);

// Nakreslete řetězec otočený o 90 stupňů doleva.
at = new AffineTransform();
at.setToRotation(Math.toRadians(-90));
```

```
at.translate(100, -100);
g2d.setTransform(at);
g2d.drawString("řetězec", x, y);
```

275 Změna velikosti a typu písma v nakreslených objektech

Při vykreslování textu se vám bude hodit, když budete moci snadno a rychle použít některé z písem, jež jsou v hostitelském systému dostupná. V tomto příkladu ukazujeme, jak lze vykreslovat text pomocí různých písem a jejich velikosti:

```
Graphics2D g2d = (Graphics2D) g;
g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
  RenderingHints.VALUE_ANTIALIAS_ON);
Font font = new Font("Serif", Font.PLAIN, 36);
g2d.setFont(font);
```


```
Graphics2D g2d = (Graphics2D) g;
g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
  RenderingHints.VALUE_ANTIALIAS_ON);
Font font = new Font("Tahoma", Font.PLAIN, 40);
g2d.setFont(font);
```

276 Paprskovité zobrazení textu

Ukázka využití tříd z balíčku java.awt.geom.

Tento program je variací na různé možnosti grafického zobrazení a změny orientace textu v nakreslených objektech. Ukazuje, jak lze text zobrazit na formuláři paprskovitě obtočený kolem středové osy.

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

public class PIText extends JFrame {
  PIText() {
    setTitle("Textové paprsky");
    setSize(200,200);
    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
  }
  public void paint(Graphics g) {
    Graphics2D g2D = (Graphics2D) g;
    AffineTransform aft = new AffineTransform();
    aft.setToTranslation(100.0, 100.0);
    g2D.transform(aft);
    aft.setToRotation(Math.PI / 8.0);
    String s = "Otočený text.";
    for (int i = 0; i < 16; i++) {
      g2D.drawString(s, 0.0f, 0.0f);
      g2D.transform(aft);
    }
  }
  public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
      public void run() {
        new PIText().setVisible(true);
      }
    });
  }
}
```