KAPITOLA

24

Seznámení s AWT: práce s okny, grafikou a textem

Témata kapitoly:

- **Třídy AWT**
- **Základy práce s okny**
- **Práce s okny založenými na třídě Frame**
- **Vytvoření okna typu Frame v apletu**
- **Vytvoření samostatné aplikace využívající okna**
- **Zobrazování informací v okně**
- **Práce s grafikou**
- **Práce s barvami**
- **Nastavení režimu kreslení**
- **Práce s písmy**
- **Řízení textového výstupu pomocí FontMetrics**

S knihovnou Abstract Window Toolkit (AWT) jsme se seznámili již v kapitole 22, v níž jsme ji využili v kódu několika ukázkových apletů. Tato kapitola je věnována jejímu podrobnému popisu. Součástí knihovny AWT je řada tříd a metod umožňujících vytváření a správu oken. Navíc se jedná o základ, na němž je vystavěna i knihovna Swing. Knihovna AWT je však poměrně rozsáhlá a její úplný popis by zabral celou samostatnou knihu. Proto zde nemůžeme popsat každičký detail každé třídy, metody či proměnné instance knihovny AWT. Nicméně v této a v následujících dvou kapitolách vám ukážeme základní techniky potřebné k efektivnímu využívání AWT při vytváření vlastních apletů či samostatných aplikací založených na grafickém uživatelském rozhraní. Získáte tak základ, díky němuž budete schopni prozkoumat zbývající části AWT již samostatně. A budete také připraveni přejít ke studiu knihovny Swing.

V této kapitole se dozvíte, jak se okna vytvářejí a spravují, jak se pracuje s fonty, s výstupním textem a jak se využívá grafika. Kapitola 25 pak je věnována popisu různých ovládacích prvků, například posuvníků a tlačítek, podporovaných AWT. Dále v ní najdete i podrobnější popis dalších aspektů mechanizmů zpracování událostí v Javě. Kapitola 26 se zabývá subsystémem pro práci s obrázky a videem, který je součástí AWT.

Ačkoliv knihovna AWT se velice často využívá zejména při tvorbě apletů, můžete ji použít i při psaní samostatných aplikací založených na grafickém uživatelském rozhraní, které vám nabízí například systémy Windows. Kvůli zjednodušení kódu je většina příkladů v této kapitole napsána ve formě apletů. Chcete-li je spustit, musíte mít k dispozici buď nějaký prohlížeč apletů, anebo webový prohlížeč podporující Javu. V několika příkladech vám také ukážeme tvorbu samostatných programů využívajících okna.

A ještě jedna důležitá poznámka. V současnosti většina programů Javy využívá uživatelská rozhraní založená na knihovně Swing. Jelikož ve srovnání s knihovnou AWT knihovna Swing

nabízí bohatší implementace některých často používaných grafických ovládacích prvků (například tlačítek, seznamů a zaškrtávacích políček), můžete vcelku snadno dospět k závěru, že knihovna AWT není nadále potřebná, neboť byla přece nahrazena knihovnou Swing. Takový závěr je však zcela nesprávný. Jak jsme si již řekli, knihovna Swing je postavena na základech vycházejících z AWT. To znamená, že mnoho aspektů AWT představuje i aspekty knihovny Swing. Navíc mnoho tříd knihovny AWT je ať už přímo či nepřímo využíváno i knihovnou Swing. V neposlední řadě je nutné zmínit i to, že pro některé typy krátkých programů (především pak krátkých apletů), využívajících grafické uživatelské rozhraní pouze v minimální míře je použití knihovny AWT namísto knihovny Swing stále smysluplné. Z toho všeho vyplývá, že ačkoliv většina uživatelských rozhraní bude v současné založena na knihovně Swing, dobrá znalost AWT je i nadále nezbytná. Zjednodušeně řečeno, neznáte-li AWT, nemůžete se stát výborným javovým programátorem.

POZNÁMKA

Pokud jste dosud nečetli kapitolu 23, měli byste tak učinit právě teď. V ní totiž najdete základní informace o mechanizmech zpracování událostí, které jsou využity v mnoha příkladech této kapitoly.

Třídy AWT

Třídy knihovny AWT jsou obsaženy v balíčku **`java.awt`**. Jedná se o jeden z největších balíčků Javy. Avšak díky tomu, že je tento balíček logicky uspořádán do hierarchické struktury, je práce s ním jednodušší, než se na první pohled může zdát. V tabulce 24.1 najdete přehled některých důležitých tříd AWT.

Tabulka 24.1 Vybrané třídy knihovny AWT

Třída	Popis
`AWTEvent`	Zapouzdřuje události AWT.
`AWTEventMulticaster`	Rozesílá události více posluchačům.
`BorderLayout`	Správce rozvržení založený na využití ohraničených oblastí okna. BorderLayout využívá pět komponent či oblastí: sever, jih, východ, západ a střed.
`Button`	Vytváří ovládací prvek typu tlačítko.
`Canvas`	Prázdné okno.
`CardLayout`	Správce rozvržení založený na použití karet. CardLayout emuluje karty v kartotéce. Pouze ta, která je zobrazena v popředí, je zobrazena celá.
`Checkbox`	Vytváří ovládací prvek typu zaškrtávací políčko.
`CheckboxGroup`	Vytváří ovládací prvek typu skupina zaškrtávacích políček.
`CheckboxMenuItem`	Vytváří položku nabídky umožňující zapnutí/vypnutí.
`Choice`	Vytváří ovládací prvek typu výběr neboli rozbalovací seznam.
`Color`	Umožňuje správu barev způsobem, který je přenositelný a nezávislý na platformě.
`Component`	Abstraktní nadtřída různých komponent AWT.
`Container`	Podtřída třídy **`Component`**, do níž lze vložit další komponenty.
`Cursor`	Zapouzdřuje bitmapový ukazatel (kurzor).

Třída	Popis
`Dialog`	Vytváří dialogové okno.
`Dimension`	Specifikuje rozměry objektu. Šířka je uložena v proměnné **`width`** a výška v proměnné **`height`**.
`EventQueue`	Řadí události do fronty.
`FileDialog`	Vytváří dialogové okno umožňující výběr souboru.
`FlowLayout`	Správce rozvržení umísťující ovládací prvky do okna za sebou tak, jak jsou definovány. FlowLayout uspořádává komponenty ve směru zleva doprava a shora dolů.
`Font`	Zapouzdřuje typ písma.
`FontMetrics`	Zapouzdřuje různé informace týkající se písma. Tyto informace vám pomáhají při zobrazování textu v okně.
`Frame`	Vytváří standardní okno mající záhlaví, rohy pro změnu velikosti a lištu s nabídkou.
`Graphics`	Zapouzdřuje grafický kontext. Tento kontext je využíván různými výstupními metodami k zobrazení výstupu v okně.
`GraphicsDevice`	Popisuje grafické zařízení, například obrazovku či tiskárnu.
`GraphicsEnvironment`	Popisuje kolekci dostupných objektů typu **`Font`** a **`GraphicsDevice`**.
`GridBagConstraints`	Definuje různá omezení týkající se třídy **`GridBagLayout`**.
`GridBagLayout`	Správce rozvržení založený na použití tabulky. GridBagLayout zobrazuje komponenty v souladu s omezeními danými instancí třídy **`GridBagConstraints`**.
`GridLayout`	Správce rozvržení GridLayout zobrazuje komponenty ve dvoudimenzionální mřížce.
`Image`	Zapouzdřuje grafické obrázky.
`Insets`	Zapouzdřuje ohraničení kontejneru.
`Label`	Vytváří popis zobrazující nějaký řetězec.
`List`	Vytváří seznam umožňující uživateli výběr některé položky. Podobá se standardnímu seznamu ve Windows.
`MediaTracker`	Spravuje objekty médií.
`Menu`	Vytváří rozbalovací nabídku.
`MenuBar`	Vytváří lištu s nabídkou.
`MenuComponent`	Abstraktní třída implementovaná různými třídami určenými pro práci s nabídkou.
`MenuItem`	Vytváří položku nabídky.
`MenuShortcut`	Zapouzdřuje klávesovou zkratku pro položku nabídky.
`Panel`	Nejjednodušší konkrétní podtřída třídy **`Container`**.
`Point`	Zapouzdřuje kartézské souřadnice bodu uložené v proměnných **`x`** a **`y`**.
`Polygon`	Zapouzdřuje mnohoúhelník.
`PopupMenu`	Zapouzdřuje vyskakovací nabídku.
`PrintJob`	Abstraktní třída představující tiskovou úlohu.
`Rectangle`	Zapouzdřuje pravoúhelník.
`Robot`	Podporuje automatizované testování aplikací založených na využití knihovny AWT.
`Scrollbar`	Vytváří ovládací prvek typu posuvník.

Třída	Popis
`ScrollPane`	Kontejner obsahující horizontální a případně i vertikální posuvník nějaké jiné komponenty.
`SystemColor`	Obsahuje barvy widgetů grafického uživatelského rozhraní, jimiž jsou například okna, posuvníky, text a další.
`TextArea`	Vytváří ovládací prvek obsahující víceřádkovou textovou oblast.
`TextComponent`	Nadtřída tříd `TextArea` a `TextField`.
`TextField`	Vytváří ovládací prvek obsahující jednořádkové textové pole.
`Toolkit`	Abstraktní třída implementovaná knihovnou AWT.
`Window`	Vytváří okno nemající žádný rámeček, žádnou lištu s nabídkou a žádné záhlaví.

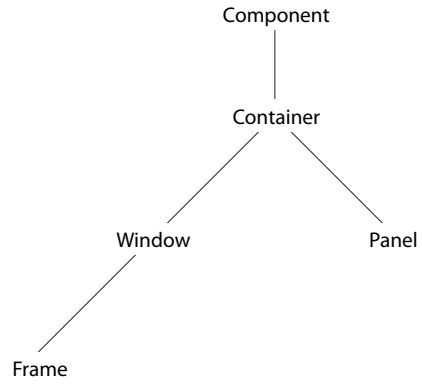
Ačkoliv základní struktura knihovny AWT se od doby uvedení Javy verze 1.0 nezměnila, některé z původních metod již byly prohlášeny za zastaralé a nahrazeny novějšími. Nicméně z důvodu zachování zpětné kompatibility Java stále podporuje všechny metody původní verze 1.0. Avšak protože tyto metody nemají být při tvorbě nových aplikací používány, v této knize se jimi vůbec nebudeme zabývat.

Základy práce s okny

Knihovna AWT definuje okna v souladu s hierarchií tříd, v níž každá další úroveň přidává funkcionalitu a zvyšuje specifičnost. Dvěma nejčastěji používanými typy oken jsou ty, které jsou odvozené od třídy `Panel`, využívané zejména aplety, a ty, které jsou odvozené od třídy `Frame`, představující okno pro standardní aplikace. Většina funkcionality těchto oken je odvozena od jejich rodičovských tříd. To znamená, že popis hierarchií tříd souvisejících s těmito dvěma třídami je základním předpokladem pro pochopení těchto tříd. Na obrázku 24.1 vidíte hierarchii tříd související s třídami `Panel` a `Frame`. V následujících částech se podíváme na každou ze zobrazených tříd poněkud podrobněji.

Třída Component

Na vrcholu celé hierarchie tříd AWT se nachází třída `Component`. Jedná se o abstraktní třídu zapouzdřující veškeré atributy vizuální komponenty. S výjimkou nabídek platí, že všechny ty prvky uživatelského rozhraní, které jsou zobrazeny na obrazovce a s nimiž uživatel nějakým způsobem pracuje, jsou podtřídami třídy `Component`. Tato třída definuje více než stovku veřejných metod odpovědných za správu událostí, například událostí souvisejících se vstupem z klávesnice či s pohyby myší, umísťováním oken a nastavováním jejich velikosti či s překreslováním. (Mnohé z těchto metod jste využívali již v kapitolách 22 a 23, a to při tvorbě apletů.) Objekt třídy `Component` je odpovědný za uchovávání informací o aktuální barvě popředí a pozadí a o aktuálně vybraném písmu.



Obrázek 24.1 Hierarchie tříd, v níž se nacházejí třídy Panel a Frame

Třída Container

Třída **`Container`** je podtřídou třídy **`Component`**. Její součástí jsou další metody, díky nimž je do ní možné vnořit další objekty typu **`Component`**. To ale znamená, že do instance třídy **`Container`** je možné uložit i další instance téže třídy (neboť ony samy jsou vlastně instancemi třídy **`Component`**). Díky tomu je možné vytváření kontejnerů majících více úrovní. Kontejner je odpovědný za rozvržení (tj. za umístění) veškerých komponent, které jsou jeho součástí. K tomuto účelu využívá různé správce rozvržení, s nimiž se podrobněji seznámíte v kapitole 25.

Třída Panel

Třída **`Panel`** je konkrétní podtřídou třídy **`Container`**. Objekt této třídy si můžete představit jako rekurzivně vnořitelnou konkrétní komponentu obrazovky. Třída **`Panel`** je nadtřídou třídy **`Applet`**. Když je výstup směřován do okna apletu, je ve skutečnosti vykreslován na povrchu objektu třídy **`Panel`**. V podstatě lze říci, že třída **`Panel`** představuje okno nemající žádné záhlaví, žádnou lištu s nabídkou ani žádné ohraničení. Spustíte-li nějaký aplet v okně webového prohlížeče, pak přesně z tohoto důvodu uvedené prvky nevidíte. Spustíte-li ovšem aplet v prohlížeči apletů, pak záhlaví a ohraničení uvidíte, neboť tyto prvky zajišťuje samotný prohlížeč apletů.

Pomocí metody **`add( )`** zděděné od třídy **`Container`** lze do instance třídy **`Panel`** přidávat další komponenty. Po přidání požadovaných komponent můžete pomocí metod **`setLocation( )`**, **`setSize( )`**, **`setPreferredSize( )`** či **`setBounds( )`** definovaných třídou **`Component`** měnit jejich umístění či velikost.

Třída Window

Třída **`Window`** vytváří vlastně okno nejvyšší úrovně. Přitom platí, že *okno nejvyšší úrovně* není obsaženo v žádném jiném objektu; toto okno je umístěno přímo na pracovní ploše. Obecně lze říci, že instance této třídy nebudete sami vytvářet. Namísto toho ve většině případů využijete podtřídu třídy **`Window`** nazvanou **`Frame`**. Ta je tématem další části.

Třída Frame

Třída **Frame** zapouzdřuje to, co je obecně chápáno jako „okno". Jedná se o podtřídu třídy **Window** umožňující tvorbu oken majících záhlaví, lištu s nabídkou a rohy pro změnu velikosti.

Třída Canvas

Ačkoliv se nejedná o součást hierarchie tříd souvisejících s okny apletů či s okny typu **Frame**, jedná se o další typ okna, jehož používání může být pro vás zajímavé. Třída **Canvas** zapouzdřuje prázdné okno, na němž můžete kreslit. Příklad použití třídy **Canvas** najdete v dalších částech této knihy.

Práce s okny založenými na třídě Frame

Lze říci, že ten typ oken, který budete hned po apletech vytvářet nejčastěji, je odvozen od třídy **Frame**. Tuto třídu budete využívat při tvorbě oken potomků v apletech a při tvorbě oken nejvyšší úrovně či oken potomků v samostatných aplikacích. Jak jsme si již řekli, tato třída vytváří standardní okno.

Níže vidíte dva z konstruktorů třídy **Frame**:

```
Frame( ) throws HeadlessException
Frame(String titulek) throws HeadlessException
```

První podoba konstruktoru umožňuje vytvoření standardního okna nemajícího žádný titulek. Druhá podoba vytváří standardní okno, v jehož záhlaví je zobrazen *titulek*. Všimněte si, že při vytváření těchto oken nemůžete zadat jejich velikost. To ovšem znamená, že velikost okna musíte definovat až po jeho vytvoření. Výjimka **HeadlessException** je vyvolávána tehdy, pokoušíte-li se vytvořit instanci typu **Frame** v prostředí nepodporujícím interakci ze strany uživatele.

Při práci s okny typu **Frame** budete nejčastěji používat několik klíčových metod, jejichž podrobnější popis najdete v následujících částech.

Nastavení rozměrů okna

Metoda **setSize( )** se používá k nastavení rozměrů okna. Definici této metody vidíte níže:

```
void setSize(int novaSirka, int novaVyska)
void setSize(Dimension novaVelikost)
```

Nová velikost okna je buď dána argumenty *novaSirka* a *novaVyska*, anebo je dána poli **width** a **height** objektu typu **Dimension** předávaného argumentem *novaVelikost*. V obou případech platí, že rozměry se zadávají v pixelech.

Chcete-li načíst aktuální velikost okna, zavolejte metodu **getSize( )**. Jedna z jejích podob je zobrazena níže:

```
Dimension getSize( )
```

Tato metoda vrací aktuální velikost okna, a to v polích **width** a **height** vráceného objektu typu **Dimension**.

Skrytí a zobrazení okna

Platí, že dokud po vytvoření okna typu **Frame** nezavoláte metodu **setVisible( )**, nebude toto okno viditelné. Definice uvedené metody vypadá takto:

```
void setVisible(boolean viditelne)
```

Daná komponenta bude viditelná pouze tehdy, bude-li argument *viditelne* této metody mít hodnotu **true**. V opačném případě bude komponenta skrytá.

Nastavení titulku okna

Pomocí metody **setTitle( )**, jejíž obecnou podobu najdete na následujícím řádku, můžete změnit titulek okna typu **Frame**:

```
void setTitle(String novyTitulek)
```

Argument *novyTitulek* zde představuje nový titulek, který má být zobrazen v záhlaví okna.

Zavření okna typu Frame

Při práci s okny typu **Frame** platí, že váš program musí při zavření okna zajistit jeho odstranění z obrazovky, a to zavoláním metody **setVisible(false)**. Pro zachycení události zavření okna musíte implementovat metodu **windowClosing( )** rozhraní **WindowListener**. Samotné odstranění okna z obrazovky pak musíte provést uvnitř metody **windowClosing( )**. Ukázku tohoto postupu najdete v příkladu, který je součástí následující části.

Vytvoření okna typu Frame v apletu

Ačkoliv je samozřejmě možné vytvořit okno jednoduchým vytvořením instance typu **Frame**, jedná se o postup, který zřejmě příliš často nevyužijete, neboť s takovým oknem nemůžete moc dělat. Například nebudete schopni přijímat či zpracovávat události, které v okně vzniknou, a ani nebudete schopni jednoduchým způsobem zajistit výstup do takového okna. Ve většině případů proto budete vytvářet podtřídy třídy **Frame**. Díky tomu budete moci překrýt metody třídy **Frame** a zajistit si zpracování událostí.

Vytvoření nového okna typu **Frame** v rámci apletu je ve skutečnosti docela snadné. Zaprvé musíte vytvořit podtřídu třídy **Frame**. Zadruhé musíte překrýt standardní metody apletu (například metody **init( )**, **start( )** či **stop( )**) tak, abyste byli schopni okno zobrazovat či skrývat dle potřeby. A nakonec musíte implementovat metodu **windowClosing( )** rozhraní **WindowListener**, a to tak, aby její implementace volala metodu **setVisible(false)** při zavírání okna.

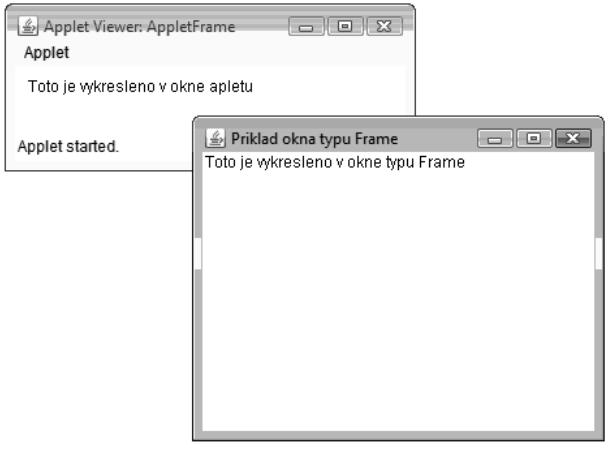
Jakmile máte nadefinovanou podtřídu třídy **Frame**, můžete vytvořit její instanci. Díky tomu sice vznikne okno typu **Frame**, avšak toto okno nebude zpočátku viditelné. Proto si okno musíte zviditelnit, a to zavoláním metody **setVisible(true)**. Kromě toho platí, že při vytváření je oknu přiřazena nějaká výchozí šířka a výška. Rozměry okna můžete explicitně nastavit zavoláním metody **setSize( )**.

Následující aplet vytváří podtřídu třídy **Frame** nazvanou **SampleFrame**. V rámci metody **init( )** třídy **AppletFrame** pak vzniká instance této třídy. Všimněte si, že třída **SampleFrame** volá konstruktor třídy **Frame**. Díky tomu program vytváří standardní okno typu **Frame** mající titulek předaný parametrem **title**. Jak vidíte, v našem ukázkovém kódu jsou také překryty metody **start( )** a **stop( )** apletu, a to proto, aby bylo možné zobrazit, resp. skrýt, vytvořené okno potomka. V důsledku těchto úprav bude ve chvíli ukončení apletu spuštěného v prohlížeči apletů či při přechodu na jinou stránku v případě apletu spuštěného ve webovém prohlížeči okno automaticky odstraněno. Navíc tyto úpravy povedou k tomu, že v případě návratu na původní stránku ve webovém prohlížeči se okno znovu zobrazí.

```
// Vytvoreni okna typu Frame, ktere je potomkem okna apletu.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="AppletFrame" width=300 height=50>
</applet>
*/

// Vytvoreni podtridy tridy Frame.
class SampleFrame extends Frame {
  SampleFrame(String title) {
    super(title);

    // Vytvoreni objektu pro zpracovani udalosti okna.
    MyWindowAdapter adapter = new MyWindowAdapter(this);

    // Registrace tohoto objektu, aby mohl udalosti okna prijimat.
    addWindowListener(adapter);
  }

  // Vystup v okne potomka.
  public void paint(Graphics g) {
    g.drawString("Toto je vykresleno v okne typu Frame", 10, 40);
  }
}

class MyWindowAdapter extends WindowAdapter {
  SampleFrame sampleFrame;

  public MyWindowAdapter(SampleFrame sampleFrame) {
    this.sampleFrame = sampleFrame;
  }

  public void windowClosing(WindowEvent we) {
    sampleFrame.setVisible(false);
  }
}

// Vytvoreni okna typu Frame.
public class AppletFrame extends Applet {
  Frame f;

  // Vytvoreni okna, nastaveni jeho velikosti a zviditelneni.
  public void init() {
    f = new SampleFrame("Priklad okna typu Frame");
    f.setSize(250, 250);
    f.setVisible(true);
  }

  // Zviditelneni okna.
```

```
  public void start() {
    f.setVisible(true);
  }

  // Skryti okna.
  public void stop() {
    f.setVisible(false);
  }

  // Vystup v okne apletu.
  public void paint(Graphics g) {
    g.drawString("Toto je vykresleno v okne apletu", 10, 20);
  }
}
```

Po spuštění tohoto programu se vám zobrazí následující výstup:



Zpracování událostí v okně typu Frame

Jelikož třída **`Frame`** je podtřídou třídy **`Component`**, dědí všechny možnosti a funkce definované touto třídou. To ovšem znamená, že okno typu **`Frame`** můžete používat a spravovat úplně stejným způsobem, jakým spravujete hlavní okno apletu. Například můžete překrýt metodu **`paint( )`** a zajistit si tak zobrazování požadovaného výstupu, volat metodu **`repaint( )`**, kdykoliv budete chtít okno překreslit, či přidávat procedury pro zpracování událostí. Kdykoliv pak v okně vznikne nějaká událost, budou zavolány ty procedury pro zpracování událostí, které jsou definované daným oknem. Přitom platí, že každé okno zpracovává svoje události samostatně. Například následující program vytváří okno reagující na události myši. Hlavní okno apletu však také zpracovává události myši. Budete-li s tímto programem chvíli experimentovat, zjistíte, že události okna jsou odesílány do toho okna, v němž událost nastala.

```
// Zpracovani udalosti mysi jak v okne apletu, tak v okne potomka.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
```

```
<applet code="WindowEvents" width=300 height=50>
</applet>
*/

// Vytvoreni podtridy tridy Frame.
class SampleFrame2 extends Frame
  implements MouseListener, MouseMotionListener {

  String msg = "";
  int mouseX=10, mouseY=40;
  int movX=0, movY=0;

  SampleFrame2(String title) {
    super(title);

    // Registrace tohoto objektu, aby mohl prijimat
    // vlastni udalosti mysi.
    addMouseListener(this);
    addMouseMotionListener(this);

    // Vytvoreni objektu pro zpracovani udalosti okna.
    MyWindowAdapter adapter = new MyWindowAdapter(this);

    // Vytvoreni objektu pro zpracovani udalosti okna.
    addWindowListener(adapter);
  }

  // Zpracovani udalosti klepnuti na tlacitko mysi.
  public void mouseClicked(MouseEvent me) {
  }

  // Zpracovani udalosti mys vstoupila do komponenty.
  public void mouseEntered(MouseEvent evtObj) {
    // Ulozeni souradnic.
    mouseX = 10;
    mouseY = 54;
    msg = "Ukazatel mysi prave vstoupil do okna potomka.";
    repaint();
  }

  // Zpracovani udalosti mys opustila komponentu.
  public void mouseExited(MouseEvent evtObj) {
    // Ulozeni souradnic.
    mouseX = 10;
    mouseY = 54;
    msg = "Ukazatel mysi prave opustil okno potomka.";
    repaint();
  }

  // Zpracovani udalosti stisknuti tlacitka mysi.
  public void mousePressed(MouseEvent me) {
```

```
    // Ulozeni souradnic.
    mouseX = me.getX();
    mouseY = me.getY();
    msg = "Tlacitko mysi stisknuto.";
    repaint();
  }

  // Zpracovani udalosti uvolneni tlacitka mysi.
  public void mouseReleased(MouseEvent me) {
    // Ulozeni souradnic.
    mouseX = me.getX();
    mouseY = me.getY();
    msg = "Tlacitko mysi uvolneno.";
    repaint();
  }

  // Zpracovani udalosti pretazeni mysi.
  public void mouseDragged(MouseEvent me) {
    // Ulozeni souradnic.
    mouseX = me.getX();
    mouseY = me.getY();
    movX = me.getX();
    movY = me.getY();
    msg = "*";
    repaint();
  }

  // Zpracovani udalosti posunu mysi.
  public void mouseMoved(MouseEvent me) {
    // Ulozeni souradnic.
    movX = me.getX();
    movY = me.getY();
    repaint(0, 0, 100, 60);
  }

  // Zobrazeni zpravy v okne potomka.
  public void paint(Graphics g) {
    g.drawString(msg, mouseX, mouseY);
    g.drawString("Ukazatel mysi ma polohu: " + movX + ", "
      + movY, 10, 40);
  }
}

class MyWindowAdapter extends WindowAdapter {
  SampleFrame2 sampleFrame;

  public MyWindowAdapter(SampleFrame2 sampleFrame) {
    this.sampleFrame = sampleFrame;
  }

  public void windowClosing(WindowEvent we) {
```