Języki i metody programowania – Java
INF302W
Wykład 5

wg
https://docs.oracle.com/javase/tutorial/uiswing/components/index.html

Autor

Dr inż. Zofia Kruczkiewicz

Struktura wykładu

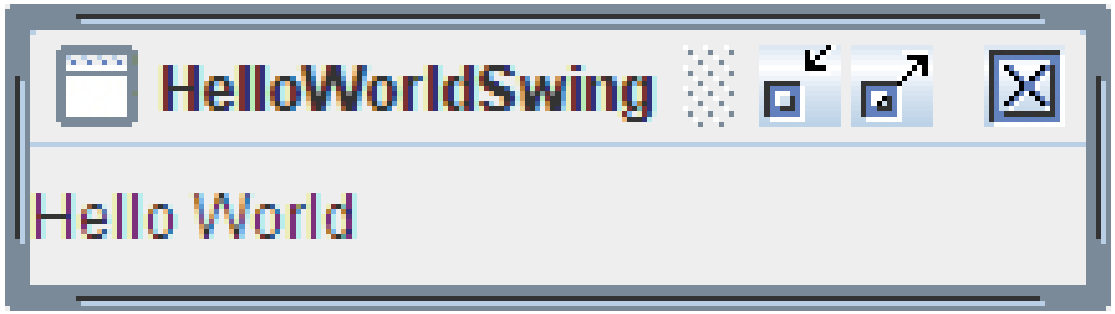
1. Zastosowanie w prostych programach składników pakietu Swing: ikony, etykiety, pola i obszary tekstowe, panele przesuwane, suwaki, przyciski opcji i wybory, listy rozwijane (1).
2. Interaktywność aplikacji Java: obsługa podstawowych zdarzeń- podstawowe procesy (i odpowiadające im klasy) obsługujące zdarzenia typu: Action, Adjustment, Focus, Item, Key, Mouse, Mouse Motion, Window. Tworzenie graficznych interfejsów użytkownika: okna dialogowe, suwaki, panele przewijane, paski narzędzi, wskaźniki zaawansowania wykonywanych operacji w programie (R-1, EL-1).
3. **Aranżacja elementów graficznego interfejsu użytkownika: podstawowe szablony rozmieszczenia składników interfejsu, zastosowanie kilku szablonów jednocześnie (EL-1).**
4. Programy z operacjami graficznymi, proste animacje (1).

JFC (Java Foundation Classes) i Swing – budowa interfejsu graficznego użytkownika(GUI)

Właściwość	Opis
Komponenty z biblioteki Swing do budowy GUI	Komponenty: od przycisków do podziału paneli na tabele. Wiele komponentów może sortować, drukować i przeciągać i upuszczać komponenty składowe itd
Wsparcie Look-and-Feel za pomocą dodatków	Wygląd i działanie aplikacji Swing można można modyfikować, umożliwiając wybór wyglądu i stylu. **Ten sam program może wykorzystywać zarówno wygląd Java jak i systemu Windows.** Dodatkowo, platforma Java obsługuje wygląd i styl GTK +, co zapewnia setki look-and-feel dla programów Swing. Wiele innych pakietów look-and-feel jest dostępnych z różnych źródeł.
Dostepność API	Włącza technologie wspomagające, takie jak czytniki ekranu i monitory brajlowskie, aby uzyskać informacje z interfejsu użytkownika.
Java 2D API	Umożliwia programistom łatwe wprowadzanie wysokiej jakości grafiki 2D, tekstu i obrazów w aplikacjach i apletach. Java 2D zawiera rozbudowane interfejsy API do generowania i wysyłania wysokiej jakości danych wyjściowych do urządzeń drukujących.
Internacjonalizacja	Umożliwia programistom tworzenie aplikacji, które mogą wchodzić w interakcje z użytkownikami na całym świecie w ich własnych językach i konwencjach kulturowych. W ramach metody wprowadzania danych programiści mogą tworzyć aplikacje akceptujące tekst w językach używających tysięcy różnych znaków, np japoński, chiński lub koreański.

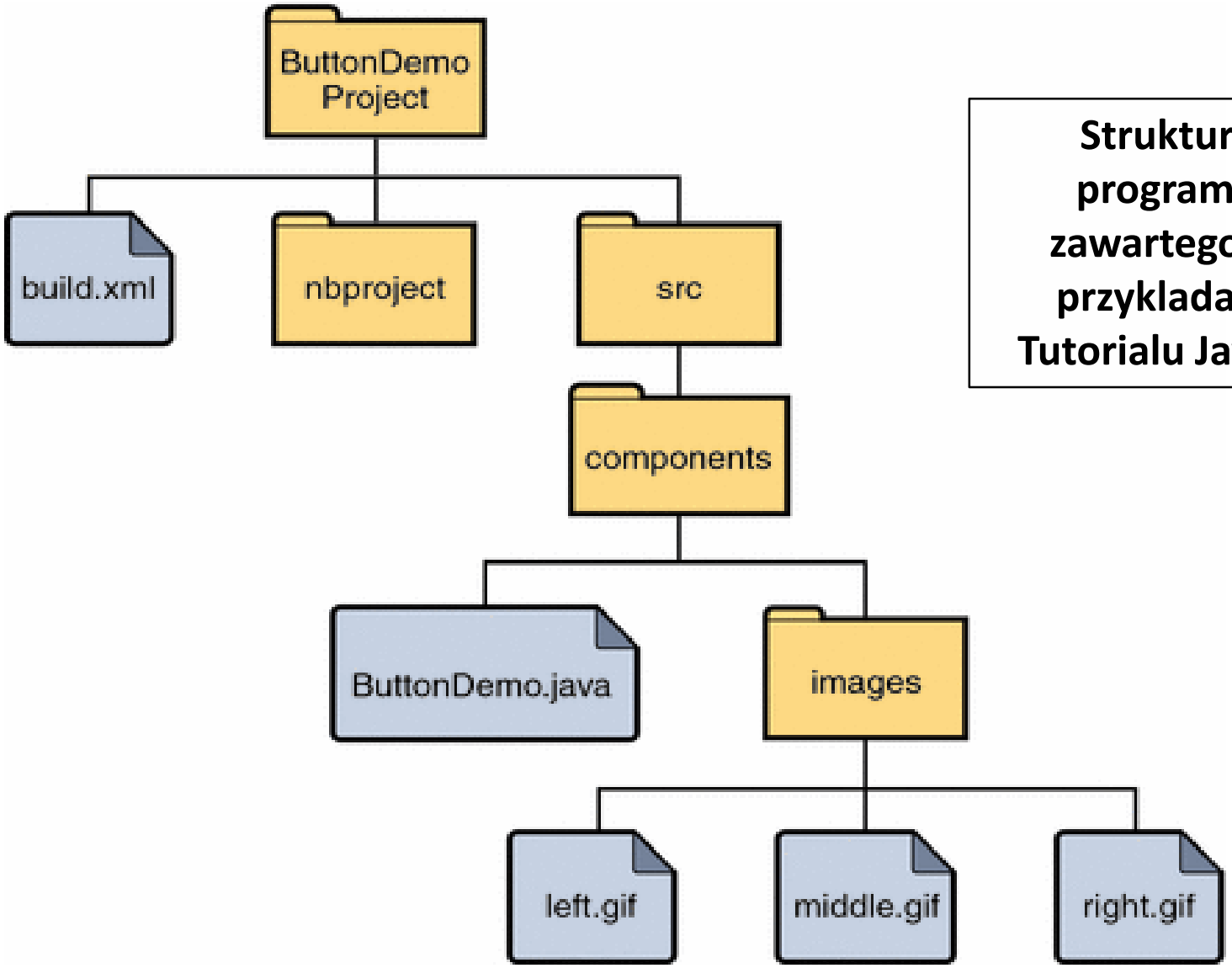
JFC (Java Foundation Classes) i Swing – 18 pakietów (najczęściej używane: javax.swing i java.swing.event)

javax.accessibility	javax.swing.plaf	javax.swing.text
javax.swing	javax.swing.plaf.basic	javax.swing.text.html
javax.swing.border	javax.swing.plaf.metal	javax.swing.text.html.parser
javax.swing.colorchooser	javax.swing.plaf.multi	javax.swing.text.rtf
javax.swing.event	javax.swing.plaf.synth	javax.swing.tree
javax.swing.filechooser	javax.swing.table	javax.swing.undo



Programy prezentujace budowę elementów (komponentów) GUI w Tutorial Java SE:

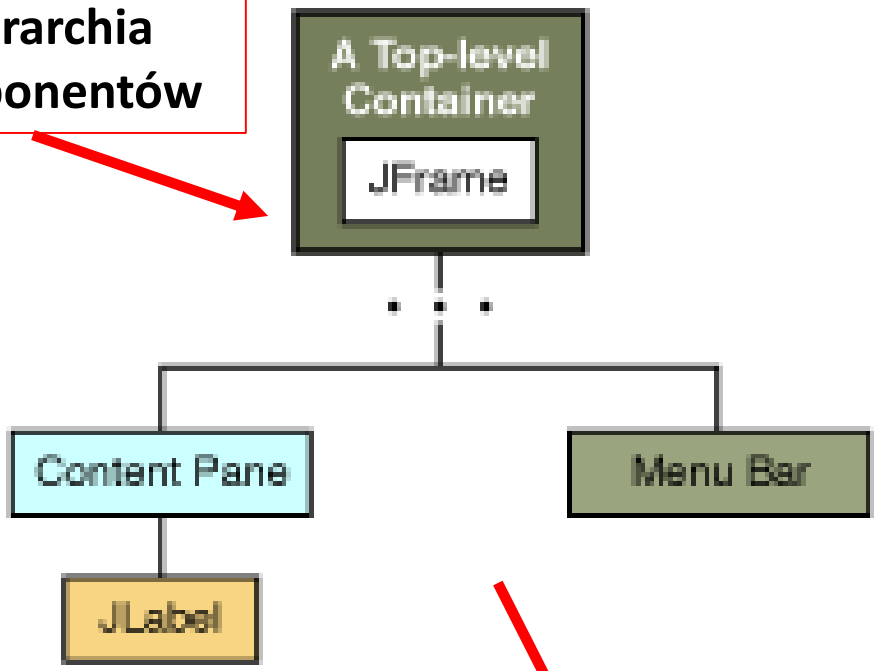
/Tutorial/tutorial/uiswing/examples/components/index.html

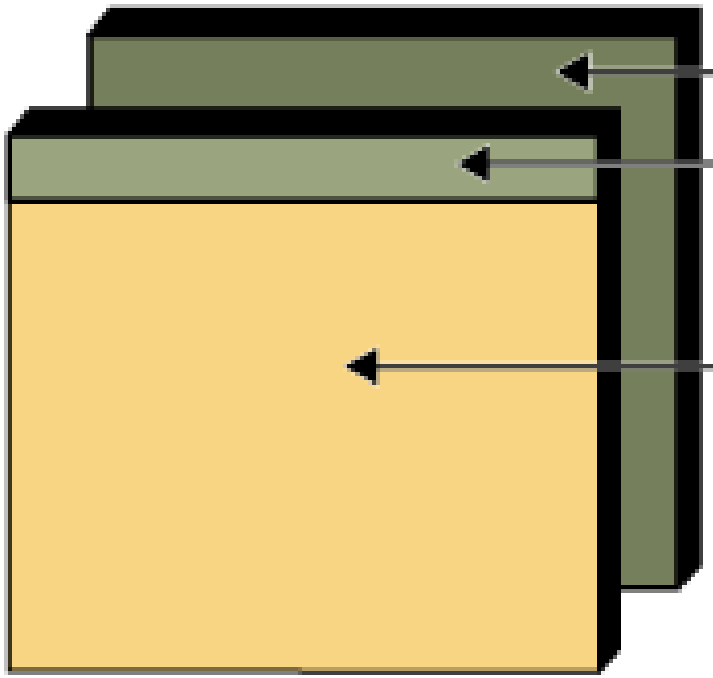


Struktura programu zawartego w przykladach Tutorialu JavaSE

Komponenty wysokiego poziomu: JFrame, JDialog, JApplet

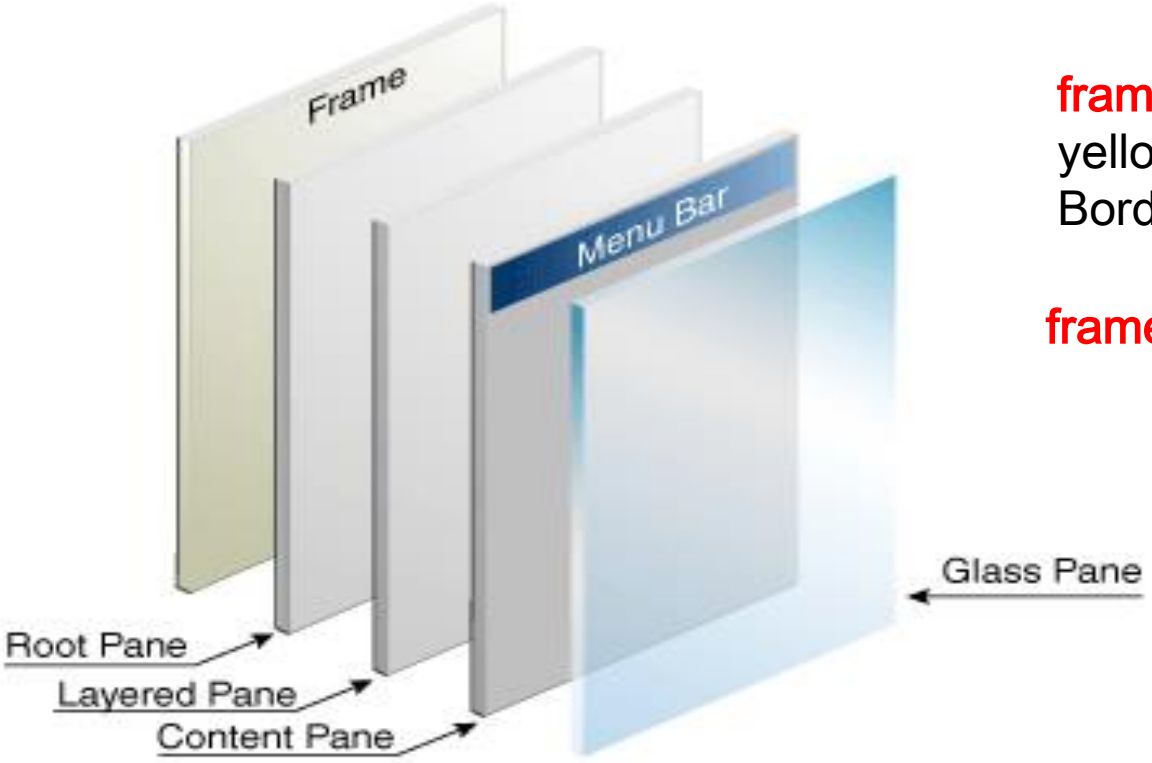
Hierarchia komponentów





Rezulat

Zbiór komponentów typu JPanel



frame.getContentPane().add(
yellowLabel,
BorderLayout.CENTER);

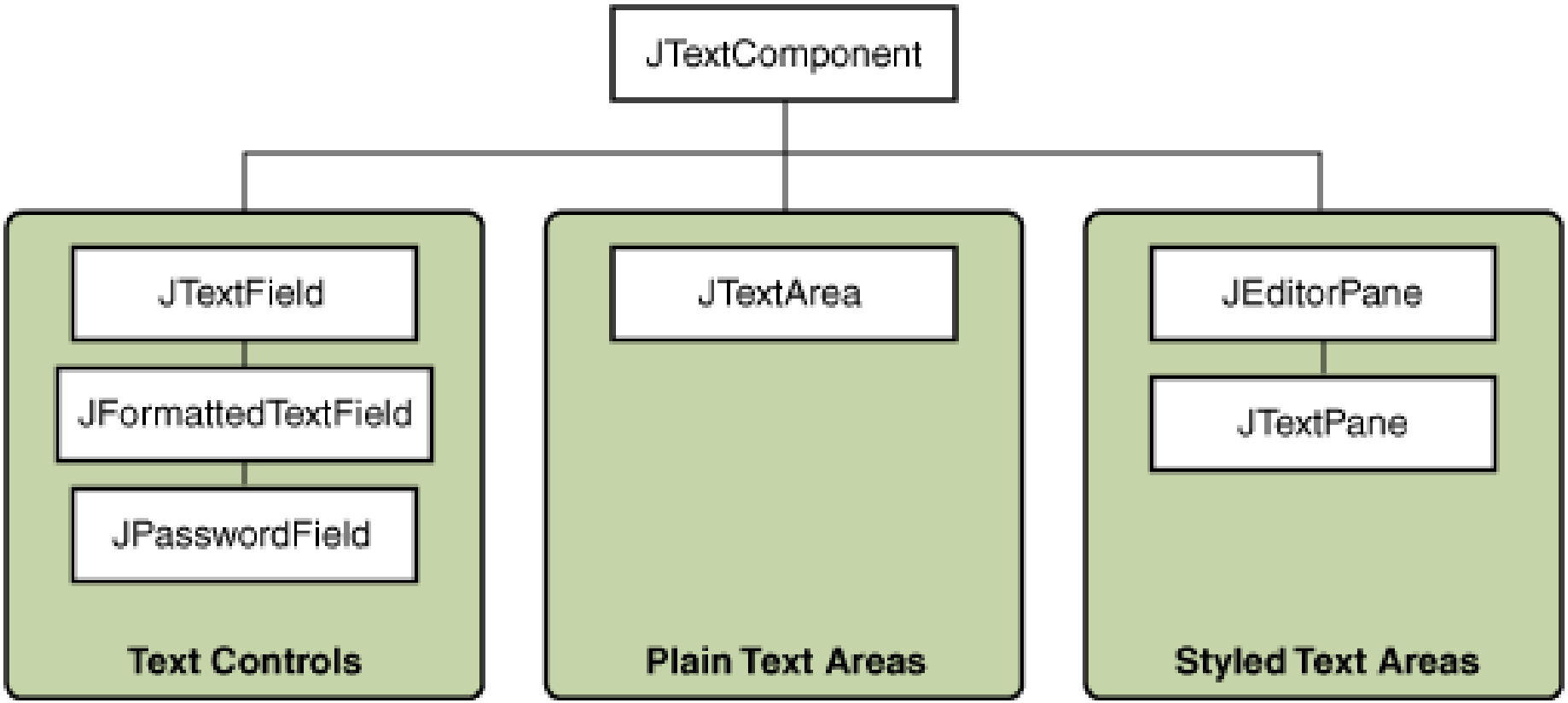
frame.setJMenuBar(greenMenuBar);

```
//Create a panel and add components to it.
JPanel contentPane = new JPanel(new BorderLayout());
contentPane.setBorder(someBorder);
contentPane.add(someComponent, BorderLayout.CENTER);
contentPane.add(anotherComponent, BorderLayout.PAGE_END);
topLevelContainer.setContentPane(contentPane);
```

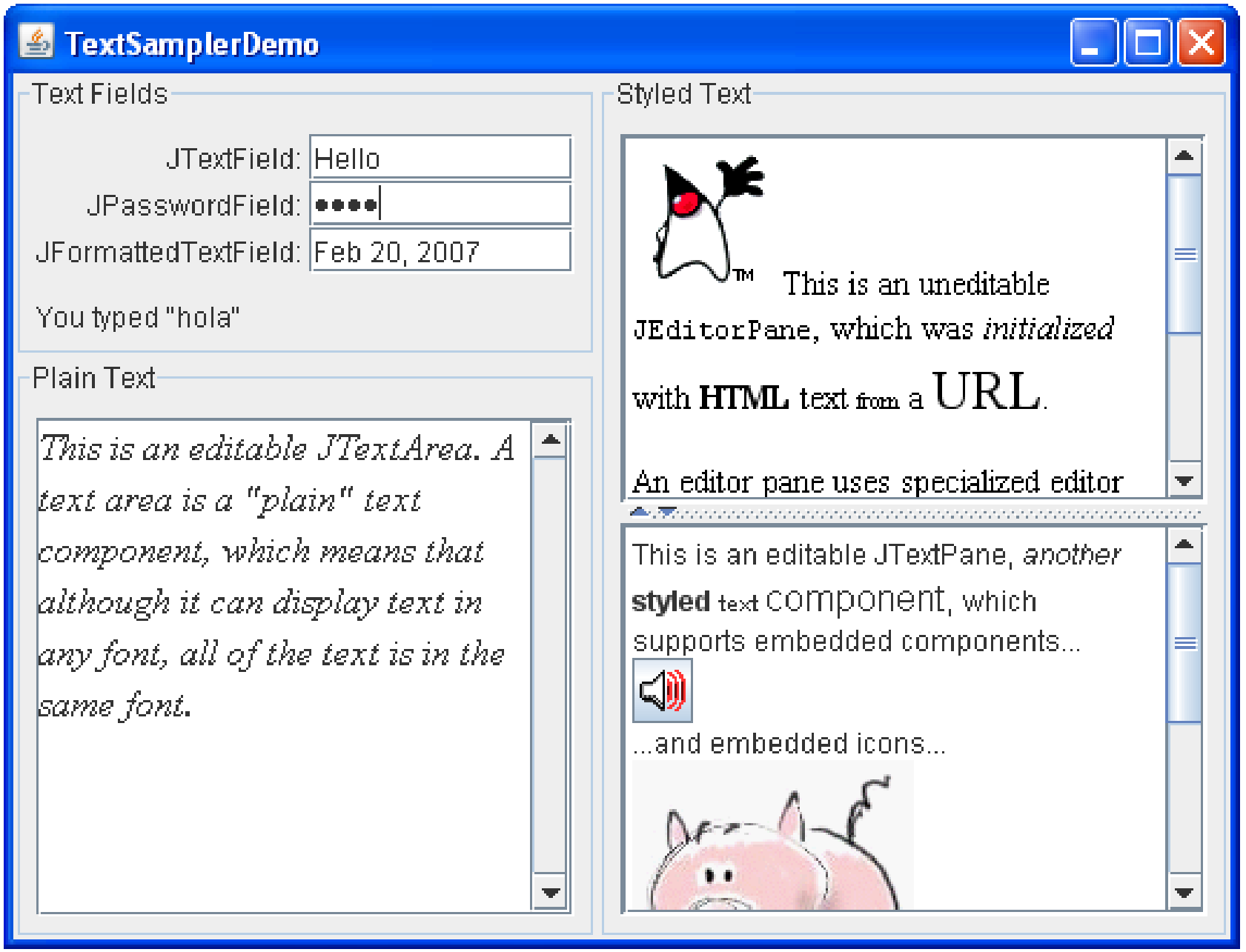
Rola komponentu JComponent

- **wyświetla informację** po umieszczeniu kursora na powierzchni komponentu, wprowadzoną do komponentu za pomocą metody **setToolTipText**
- **Wyświetla ramkę** ograniczającą powierzchnię komponentu za pomocą metody **setBorder** oraz **zawartośc ramki** za pomocą metody **paintComponent**
- Posiada obiekt **ComponentUI**, który rysuje, obsługuje zdarzenia komponentu. Sposób dzialania tego obiektu mozna ustawić za pomocą metody: **UIManager.setLookAndFeel**
- Możliwość ustawiania wartości atrybutów komponentu za pomocą metod **putClientProperty** i **getClientProperty**
- Obsługa układu komponentu za pomocą metod **getPreferredSize, getAlignmentX, setMinimumSize, setMaximumSize, setAlignmentX, setAlignmentY.**
- Metody komponentu wspierają funkcjonalność komponentu
- Metody komponentu wspierają funkcjonalnosć typu „przeciągnij i upuść” (drag and drop)
- Podwójne buforowanie
- Możliwość powiązania z klawiaturą za pomocą obiektów typu **InputMap** i **ActionMap**

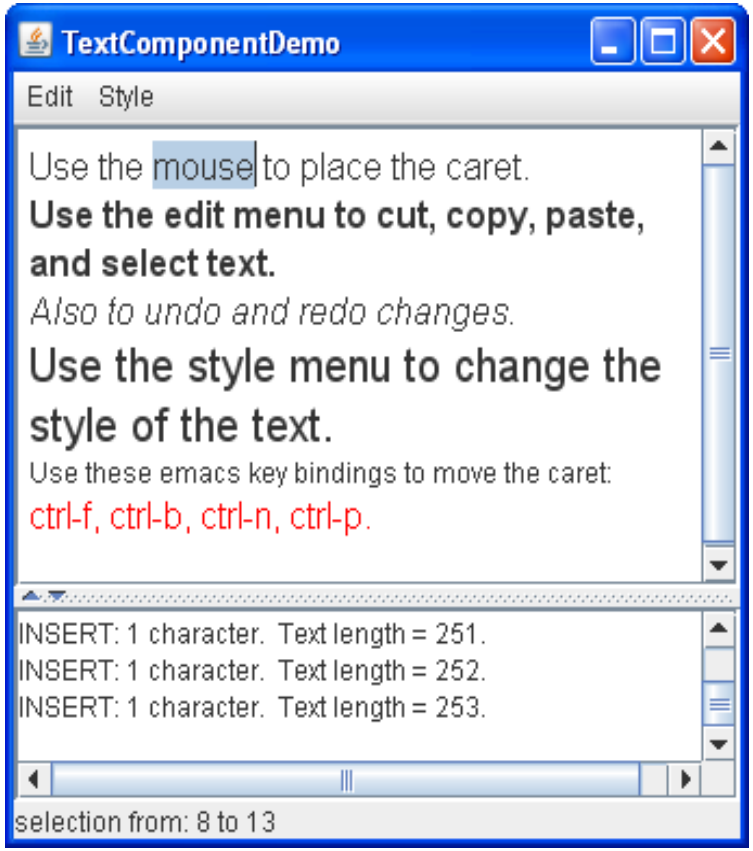
Rola komponentu tekstowego - JTextComponent



Rola komponentu tekstowego (cd)



Rola komponentu tekstowego - JTextComponent



- Associating Text Actions With Menus and Buttons
- Associating Text Actions With Key Strokes
- Implementing Undo and Redo
- Concepts: About Documents
- Implementing a Document Filter
- Listening for Changes on a Document
- Listening for Caret and Selection Changes
- Concepts: About Editor Kits

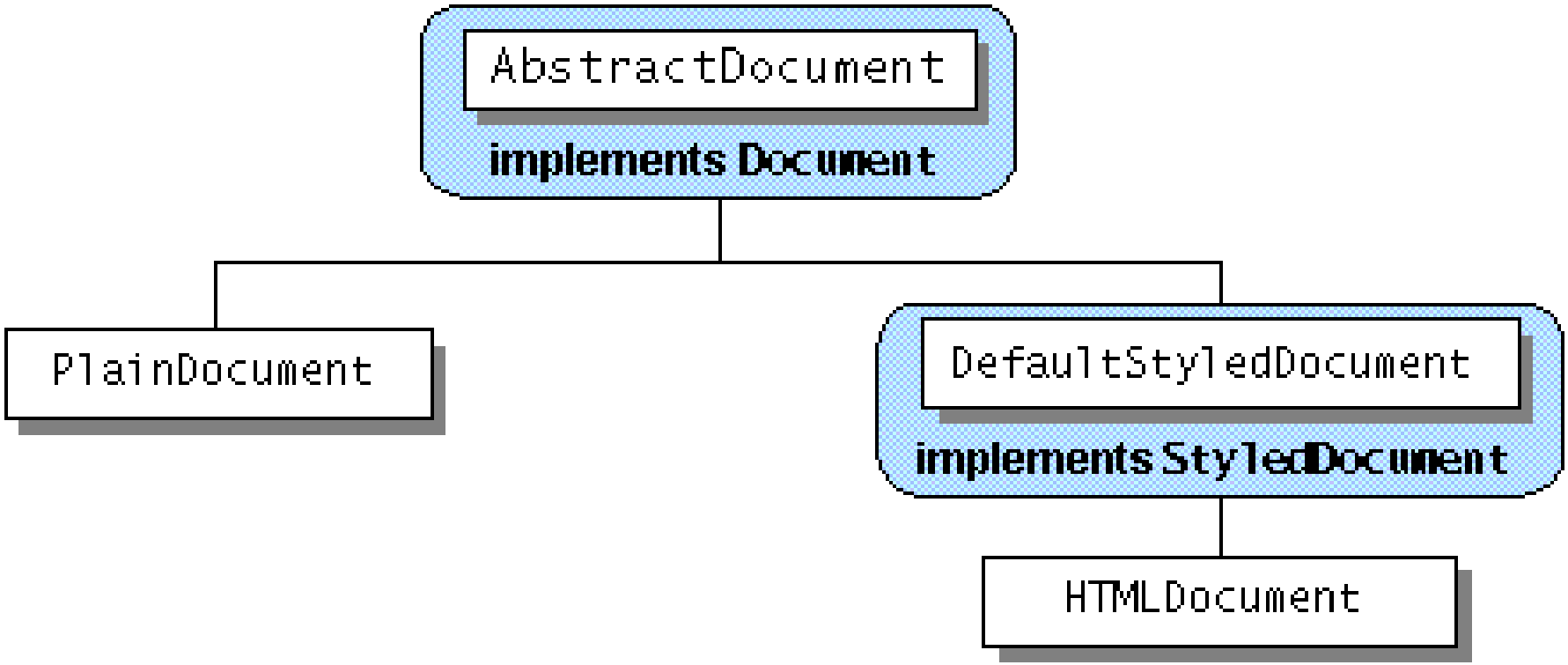
Rola komponentu tekstowego - JTextComponent

```
private HashMap<Object, Action> createActionTable(
                                        JTextComponent textComponent)
  {
  HashMap<Object, Action> actions = new HashMap<Object, Action>();
  Action[] actionsArray = textComponent.getActions();
      for (int i = 0; i < actionsArray.length; i++)
       { Action a = actionsArray[i];
         actions.put(a.getValue(Action.NAME), a); }
         return actions;
  }

 private Action getActionByName(String name)
           {   return actions.get(name);   }
```

Tworzenie tablicy akcji związanych z komponentem JTextComponent

```
protected JMenu createEditMenu()
  {  JMenu menu = new JMenu("Edit"); ...
     menu.add(getActionByName(DefaultEditorKit.cutAction)); ...
```

Rola komponentów implementujących interface Document w pakiecie javax.swing.text



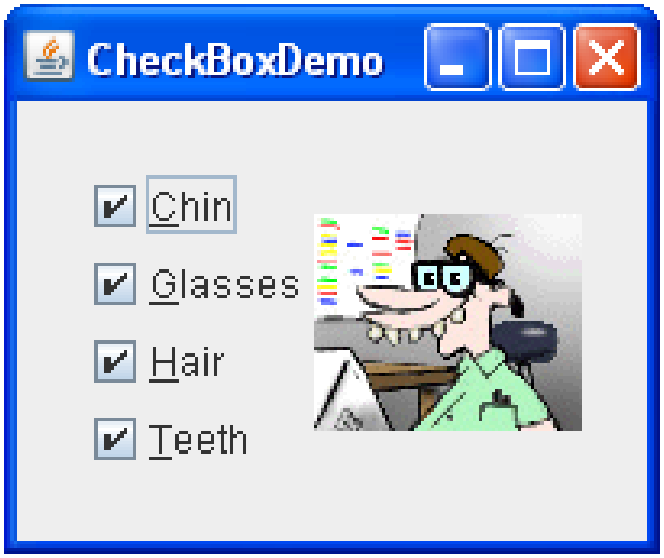
Lista pozostałych komponentów

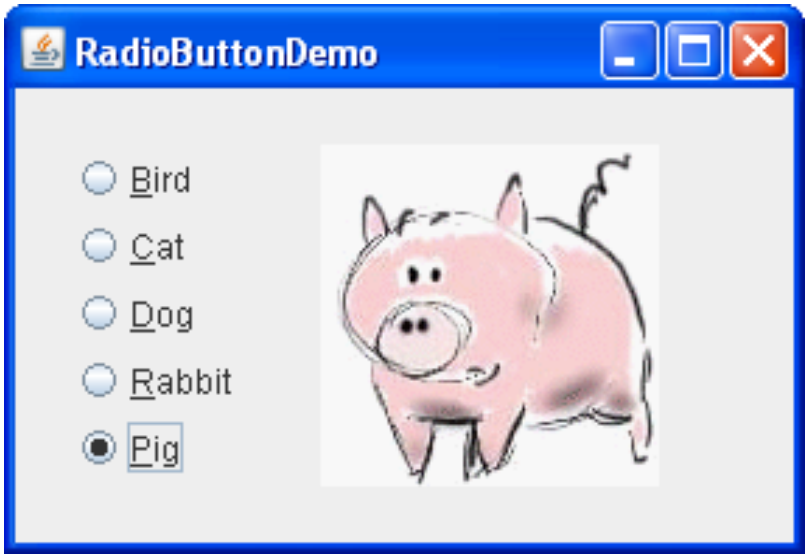
1. How to Make Applets
2. How to Use Buttons, Check Boxes, and Radio Buttons
3. How to Use Color Choosers
4. How to Use Combo Boxes
5. How to Make Dialogs
6. How to Use Editor Panes and Text Panes
7. How to Use File Choosers
8. How to Use Formatted Text Fields
9. How to Make Frames (Main Windows)
10. How to Use Internal Frames
11. How to Use Labels
12. How to Use Layered Panes
13. How to Use Lists
14. How to Use Menus
15. How to Use Panels

Lista pozostałych komponentów (cd)

16. How to Use Password Fields
17. How to Use Progress Bars
18. How to Use Root Panes
19. How to Use Scroll Panes
20. How to Use Separators
21. How to Use Sliders
22. How to Use Spinners
23. How to Use Split Panes
24. How to Use Tabbed Panes
25. How to Use Tables
26. How to Use Text Areas
27. How to Use Text Fields
28. How to Use Tool Bars
29. How to Use Tool Tips
30. How to Use Trees

Komponenty typu JButton, JCheckBox, JRadioButton, ButtonGroup

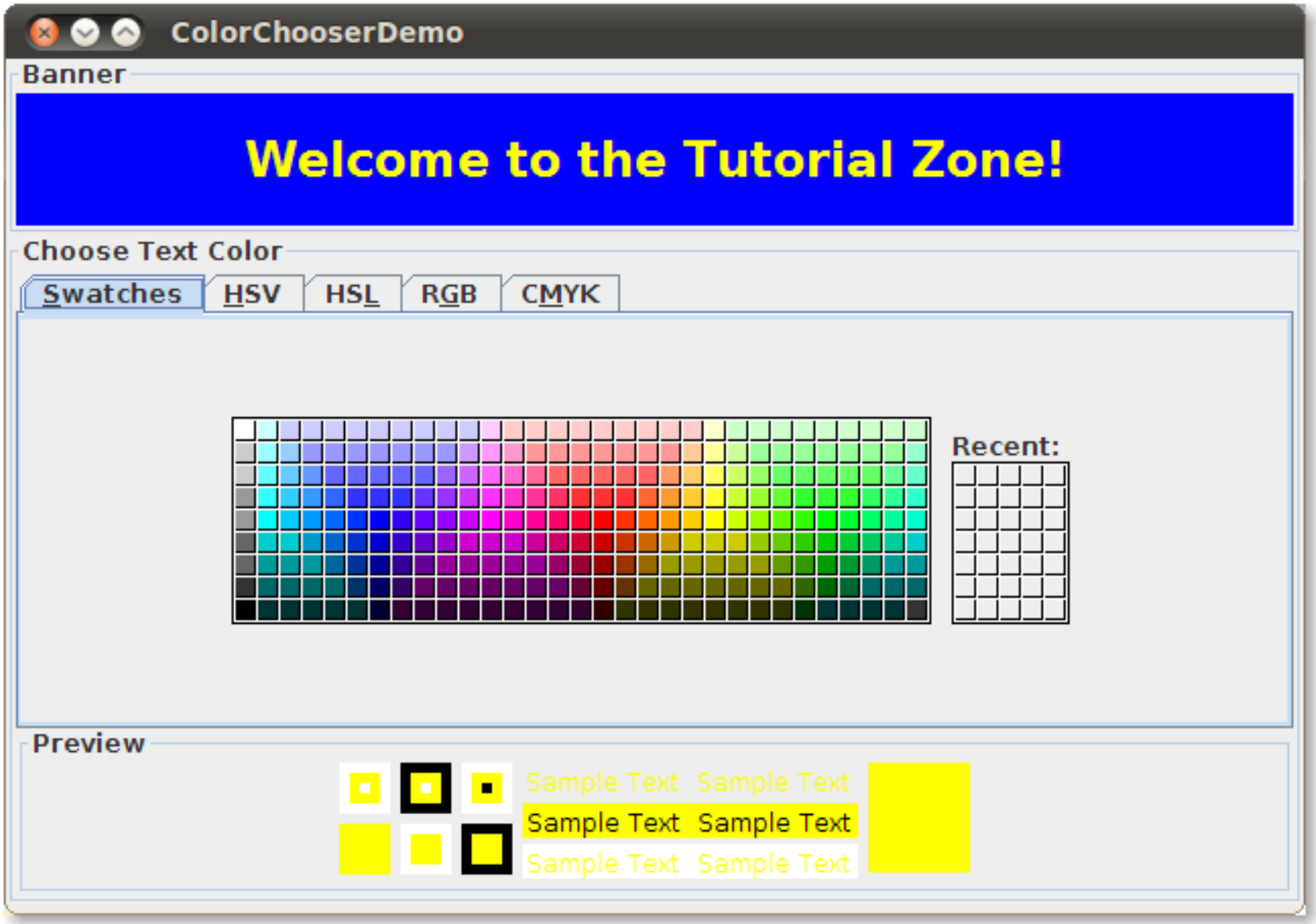




Komponenty typu JButton, JCheckBox, JRadioButton, ButtonGroup (cd)

Klasa	Przeznaczenie
JButton	Zwykły przycisk.
JCheckBox	Przycisk zaznaczany
JRadioButton	Jeden przycisk z grupy przycisków
JMenuItem	Pozycja z listy pozycji menu
JCheckBoxMenuItem	Przycisk reprezentujący pozycję z listy menu
JRadioButtonMenuItem	A menu item that has a radio button.
JToggleButton	Implementuje funkcjonalność przycisków JCheckBox i JRadioButton..
ButtonGroup	Grupa przycisków dziedziczących po AbstractButton. Najczęściej są to: **JRadioButton, JRadioButtonMenuItem, JToggleButton**

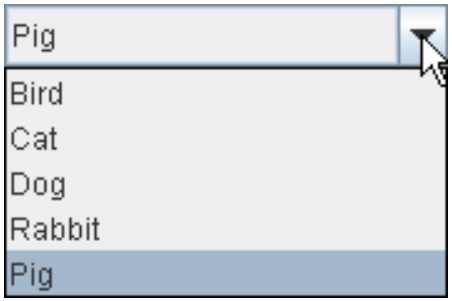
Komponenty typu JColorChooser

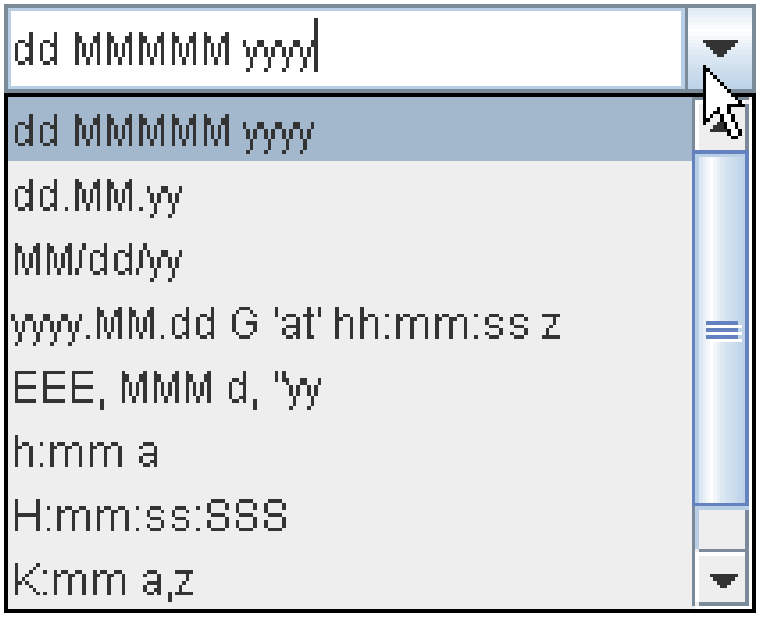


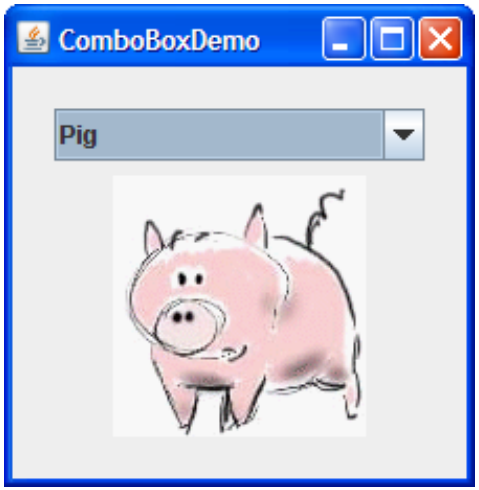
Komponenty typu JColorChooser

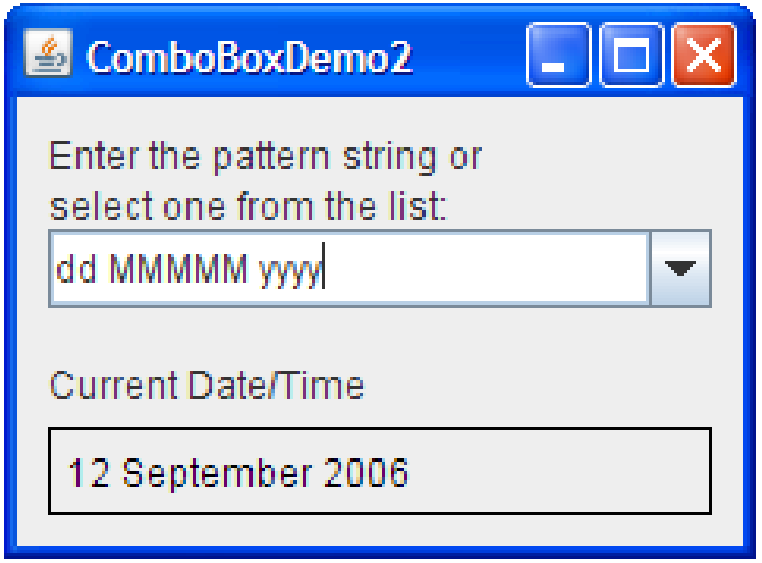
```
public class ColorChooserDemo extends JPanel ... {

   public ColorChooserDemo() {
      super(new BorderLayout());
      banner = new JLabel("Welcome to the Tutorial Zone!",
                                    JLabel.CENTER);
      banner.setForeground(Color.yellow);
       . . .
      tcc = new JColorChooser(banner.getForeground());
      . . .
     add(tcc, BorderLayout.PAGE_END); }

==============================
    tcc.getSelectionModel().addChangeListener(this); . . .
    //zmiana koloru banera
    public void stateChanged(ChangeEvent e) {
        Color newColor = tcc.getColor();
        banner.setForeground(newColor);
```

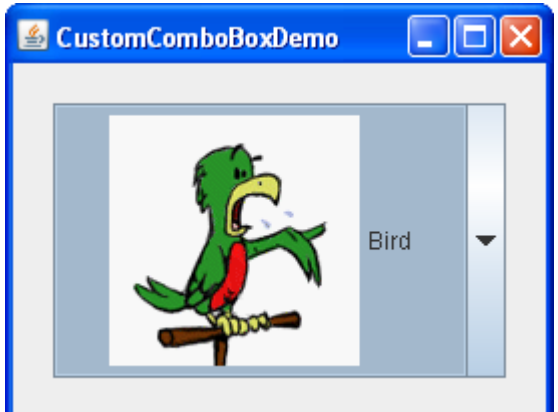
Komponenty typu JComboBox



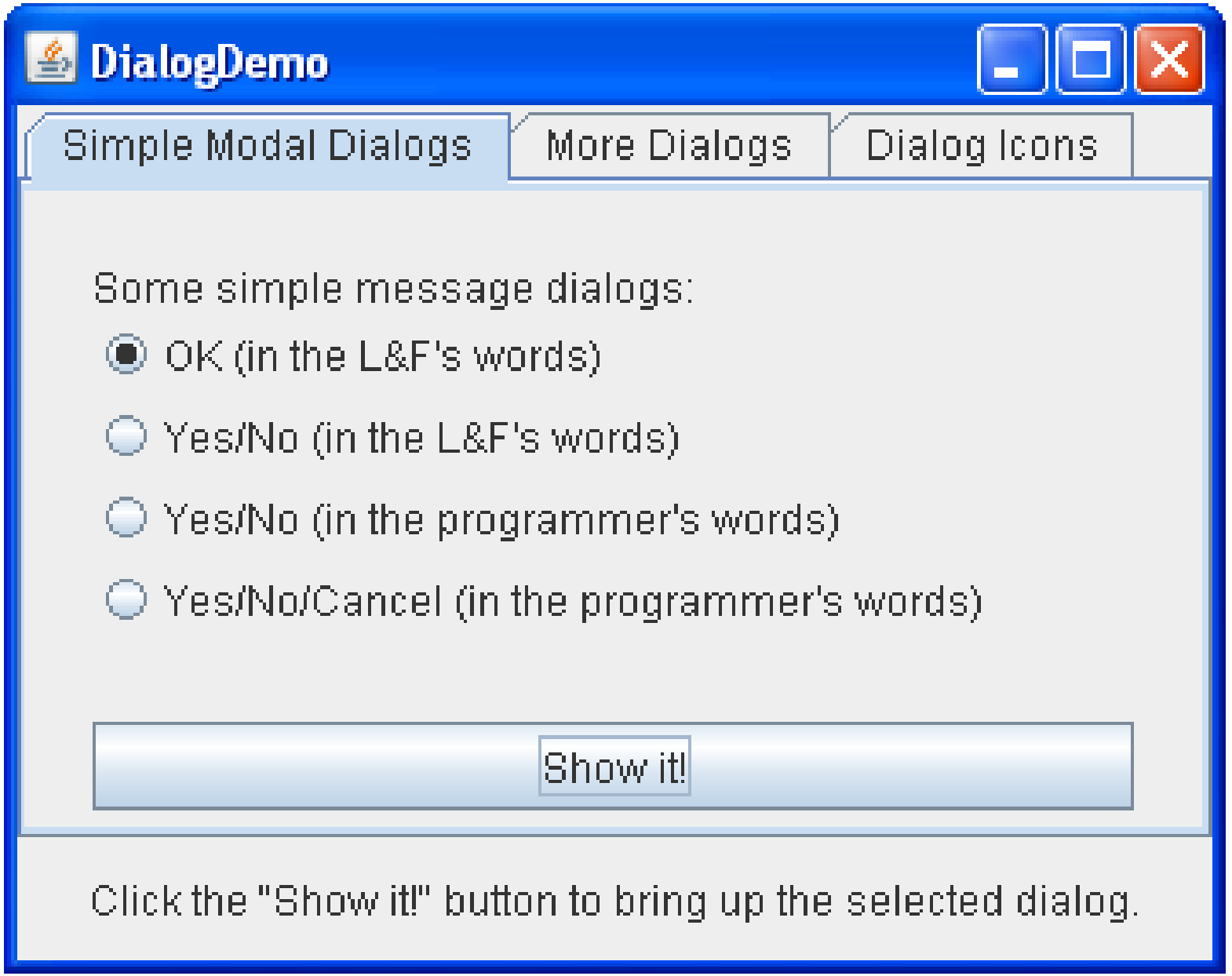








Dialog - zastosowanie komponentu typu JDialog



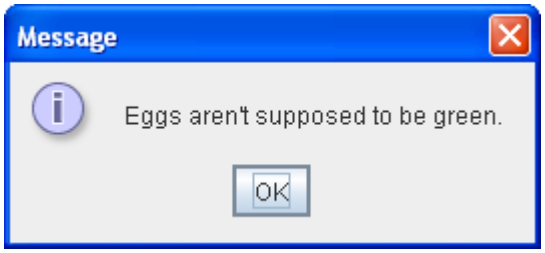
Dialog - zastosowanie komponentu typu JOptionPane


```
JOptionPane.showMessageDialog(
  frame,
"Eggs are not supposed to be green.");
```

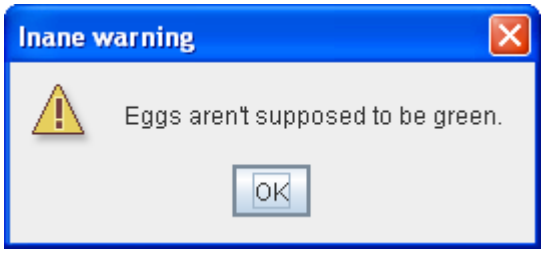
Icons used by JOptionPane

Icon description	Java look and feel	Windows look and feel
question		
information		
warning		
error		

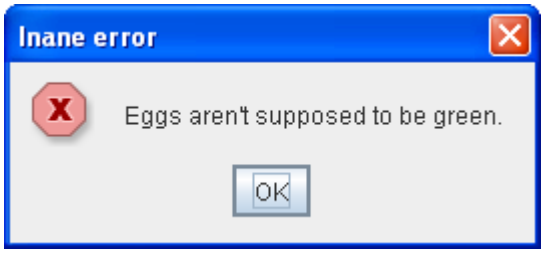
(cd) Dialog - zastosowanie komponentu typu JOptionPane: metoda showMessageDialog



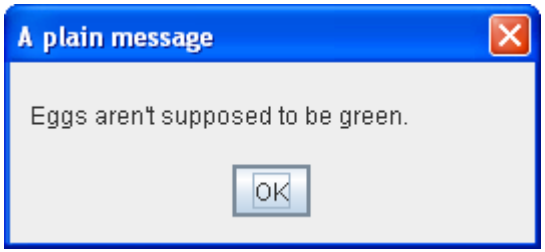

```
JOptionPane.showMessageDialog(frame,
            "Eggs are not supposed to be green.");
```



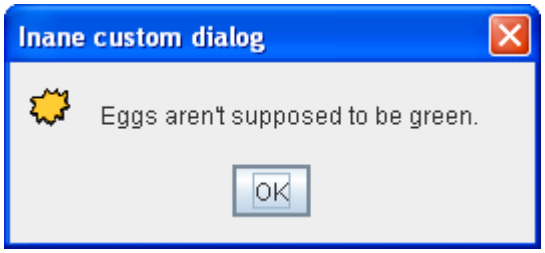

```
JOptionPane.showMessageDialog(frame,
"Eggs are not supposed to be green.", "Inane warning",
JOptionPane.WARNING_MESSAGE);
```




```
JOptionPane.showMessageDialog(frame,
"Eggs are not supposed to be green.",
"Inane error",
JOptionPane.ERROR_MESSAGE);
```

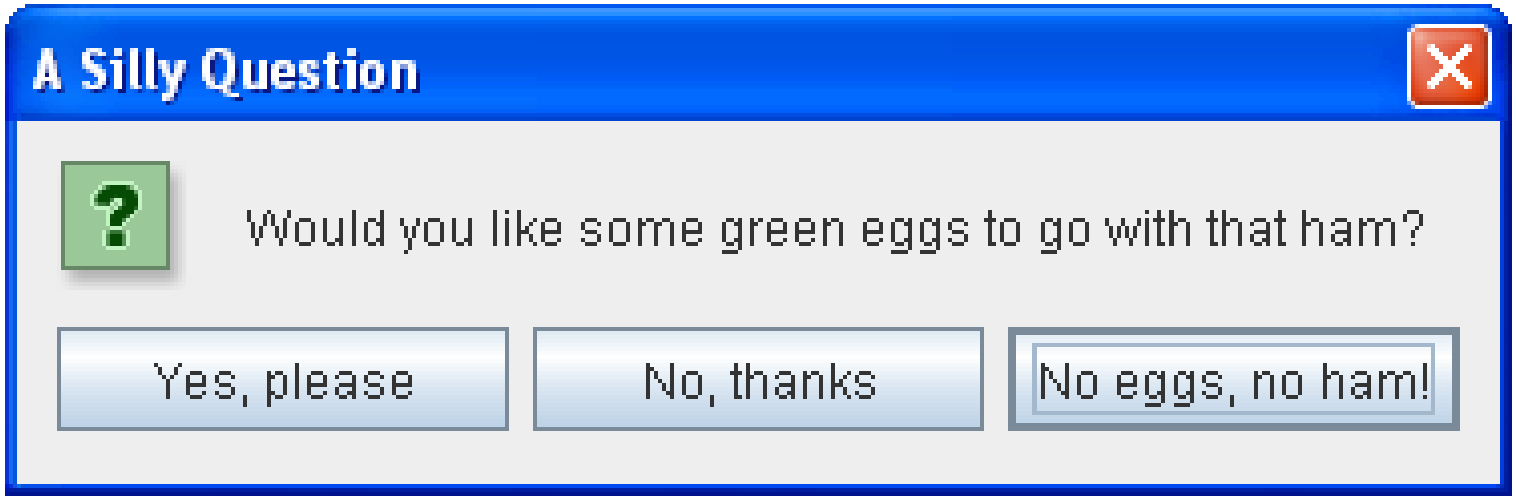



```
JOptionPane.showMessageDialog(frame,
"Eggs are not supposed to be green.",
"A plain message",
JOptionPane.PLAIN_MESSAGE);
```



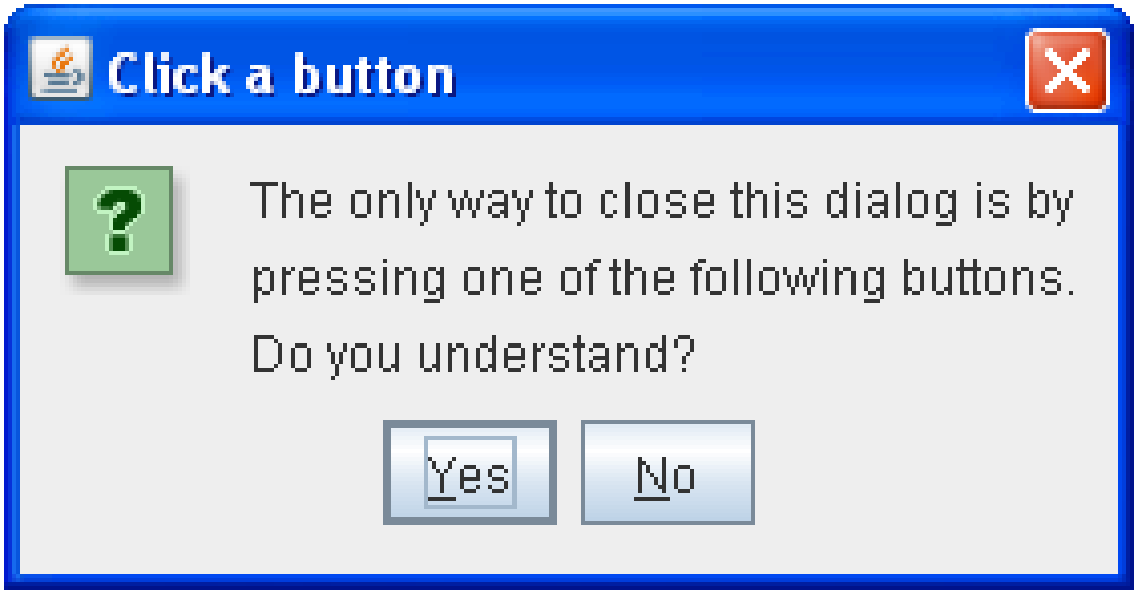

```
JOptionPane.showMessageDialog(frame,
 "Eggs are not supposed to be green.",
 "Inane custom dialog",
JOptionPane.INFORMATION_MESSAGE, icon);
```

(cd) Dialog - zastosowanie komponentu typu JOptionPane: metoda showOptionDialog



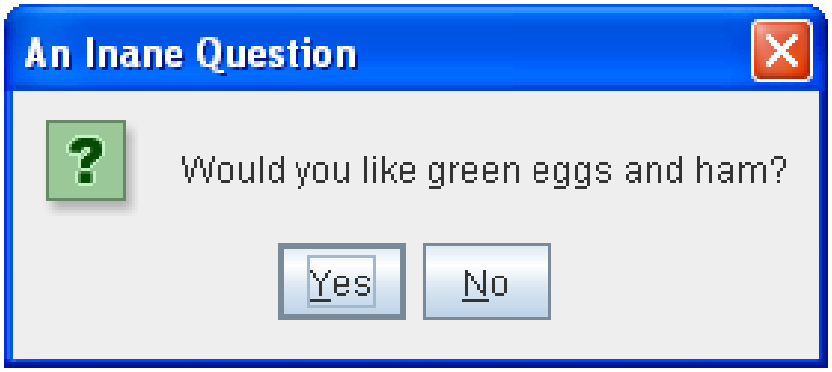

```
//tekst programisty
Object[] options = {"Yes, please", "No, thanks", "No eggs, no ham!"};
int n = JOptionPane.showOptionDialog(frame,
 "Would you like some green eggs to go " + "with that ham?",
"A Silly Question",
JOptionPane.YES_NO_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE,
null,
options,
options[2]);
```

(cd) Dialog - zastosowanie komponentu typu JOptionPane - konstruktor klasy JOptionPane

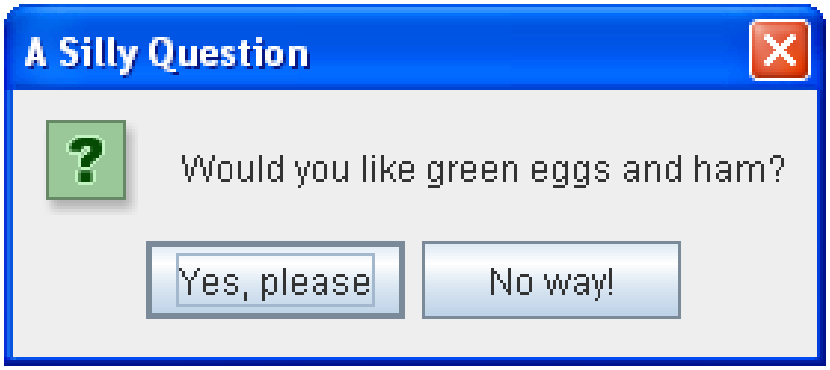



```
final JOptionPane optionPane = new JOptionPane(
"The only way to close this dialog is by\n" +
"pressing one of the following buttons.\n" + "Do you understand?",
JOptionPane.QUESTION_MESSAGE,
JOptionPane.YES_NO_OPTION);
```

(cd) Dialog - zastosowanie komponentu typu JOptionPane – dostosowanie tekstu na przyciskach

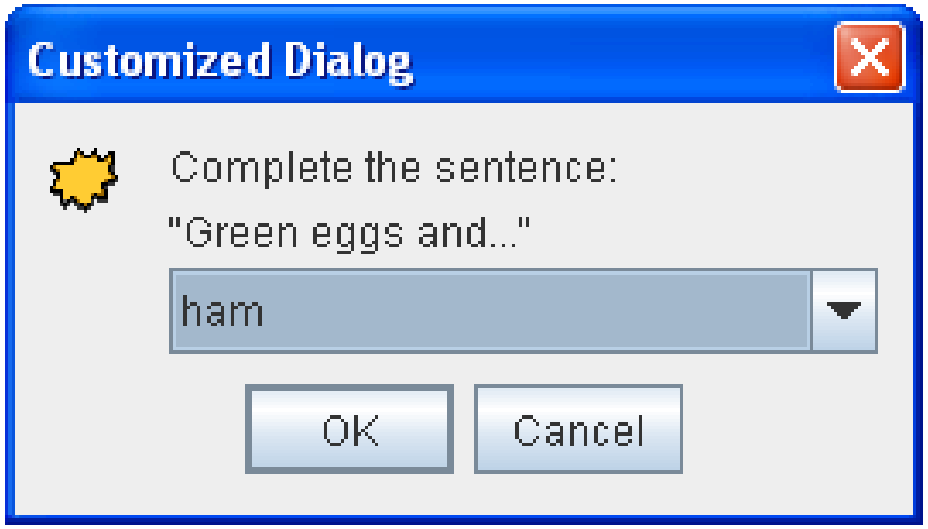



```
int n = JOptionPane.showConfirmDialog(
frame,
"Would you like green eggs and ham?",
 "An Inane Question",
JOptionPane.YES_NO_OPTION);
```



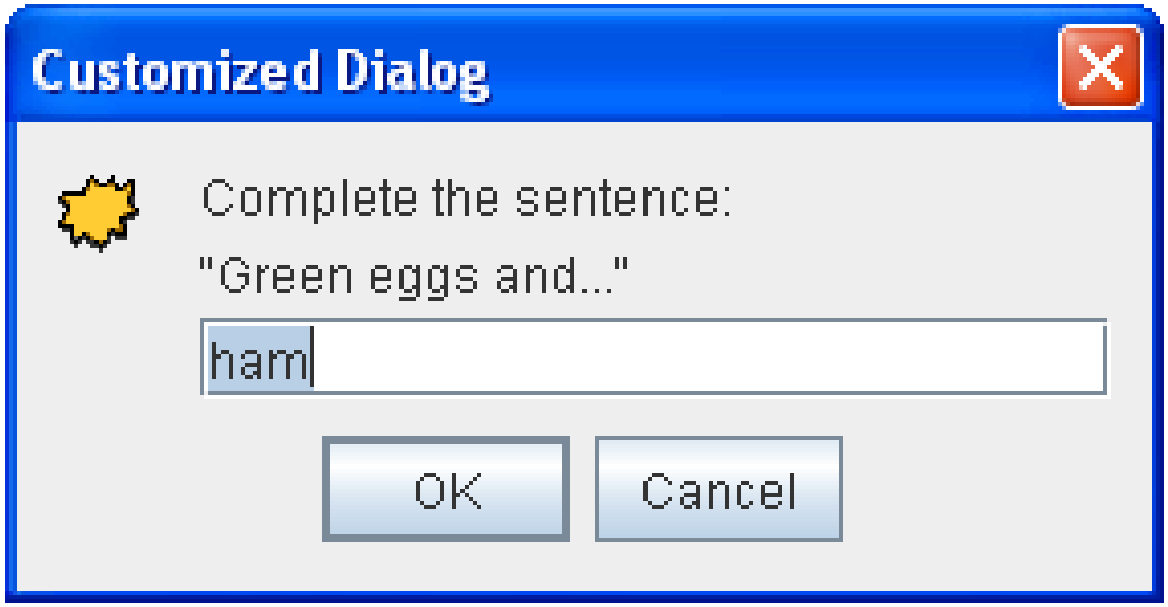

```
Object[] options = {"Yes, please", "No way!"};
int n = JOptionPane.showOptionDialog(
frame,
"Would you like green eggs and ham?",
 "A Silly Question",
JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE,
null,                    //do not use a custom Icon
options,                 //the titles of buttons
options[0]);             //default button title
```

(cd) Dialog - zastosowanie komponentu typu JOptionPane



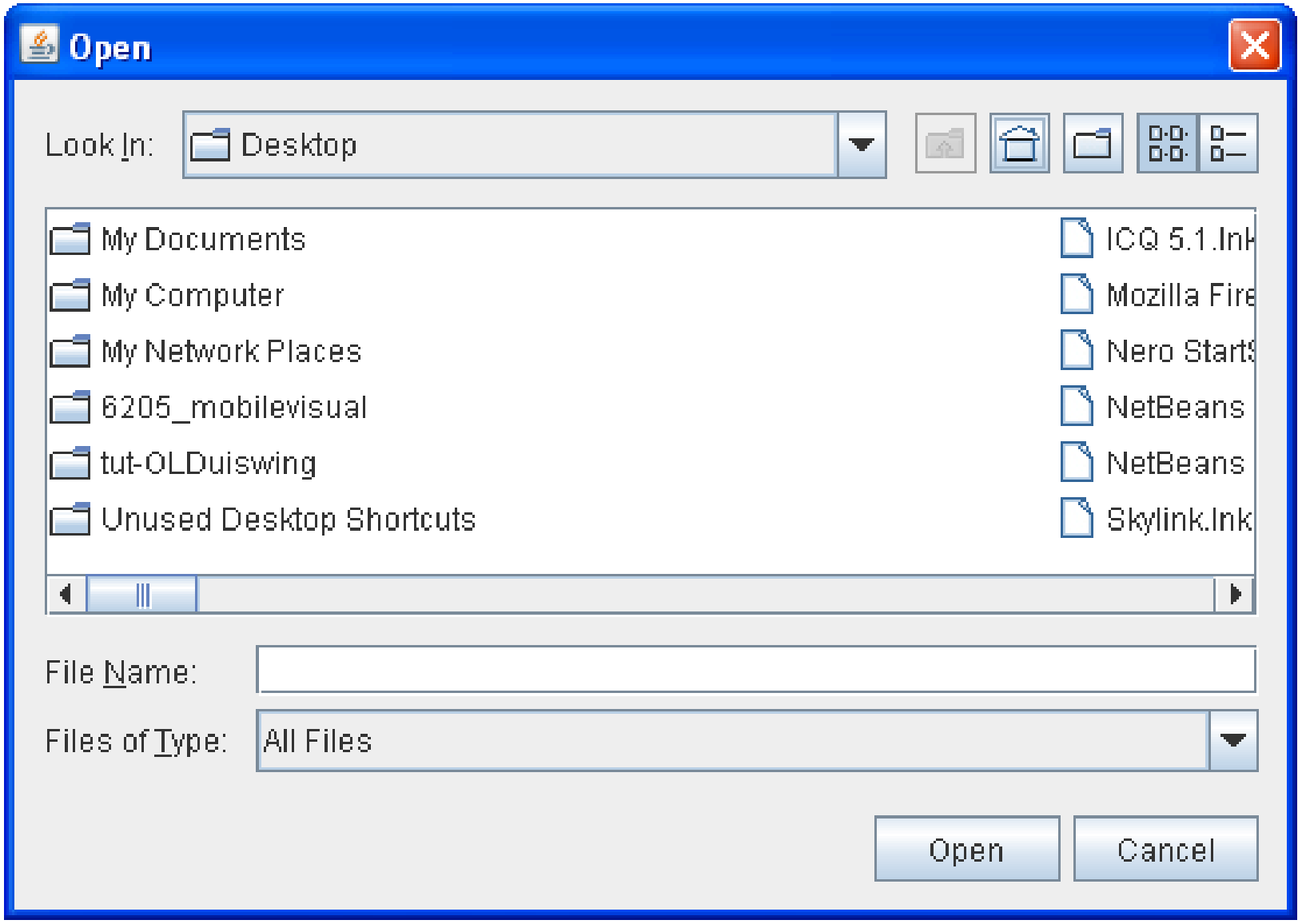

```
Object[] possibilities = {"ham", "spam", "yam"};
String s = (String)JOptionPane.showInputDialog(
frame,
 "Complete the sentence:\n" + "\"Green eggs and...\"",
 "Customized Dialog",
 JOptionPane.PLAIN_MESSAGE, icon, possibilities, "ham");

//reakcja na komunikat
if ((s != null) && (s.length() > 0))
{ setLabel("Green eggs and... " + s + "!");
return; }
//reakcja na brak komunikatu
setLabel("Come on, finish the sentence!");
```

(cd) Dialog - zastosowanie komponentu typu JOptionPane

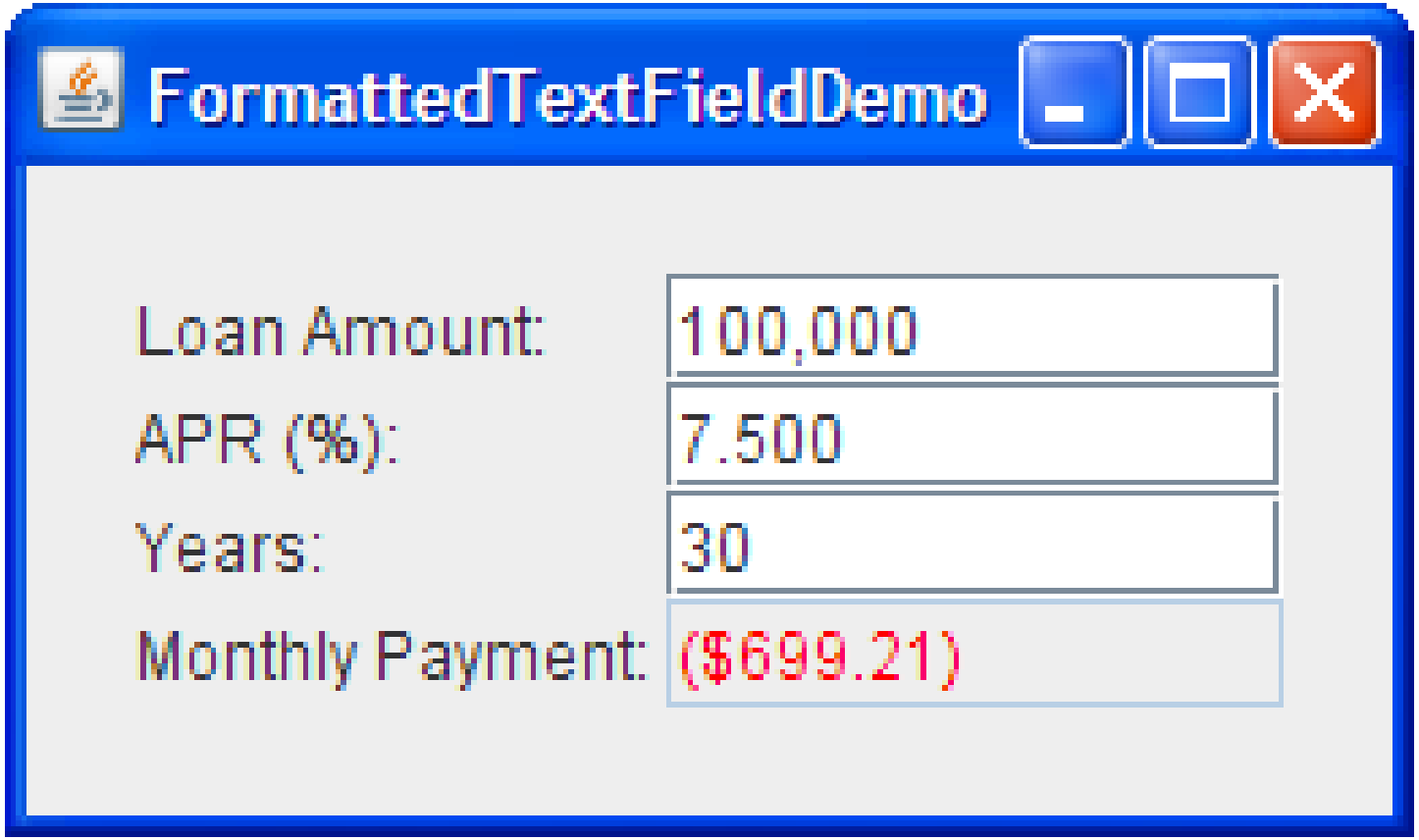



```
String s=JOptionPane.showInputDialog(
      frame,
     "Complete the sentence:\n" + "\"Green eggs and...\""));
```

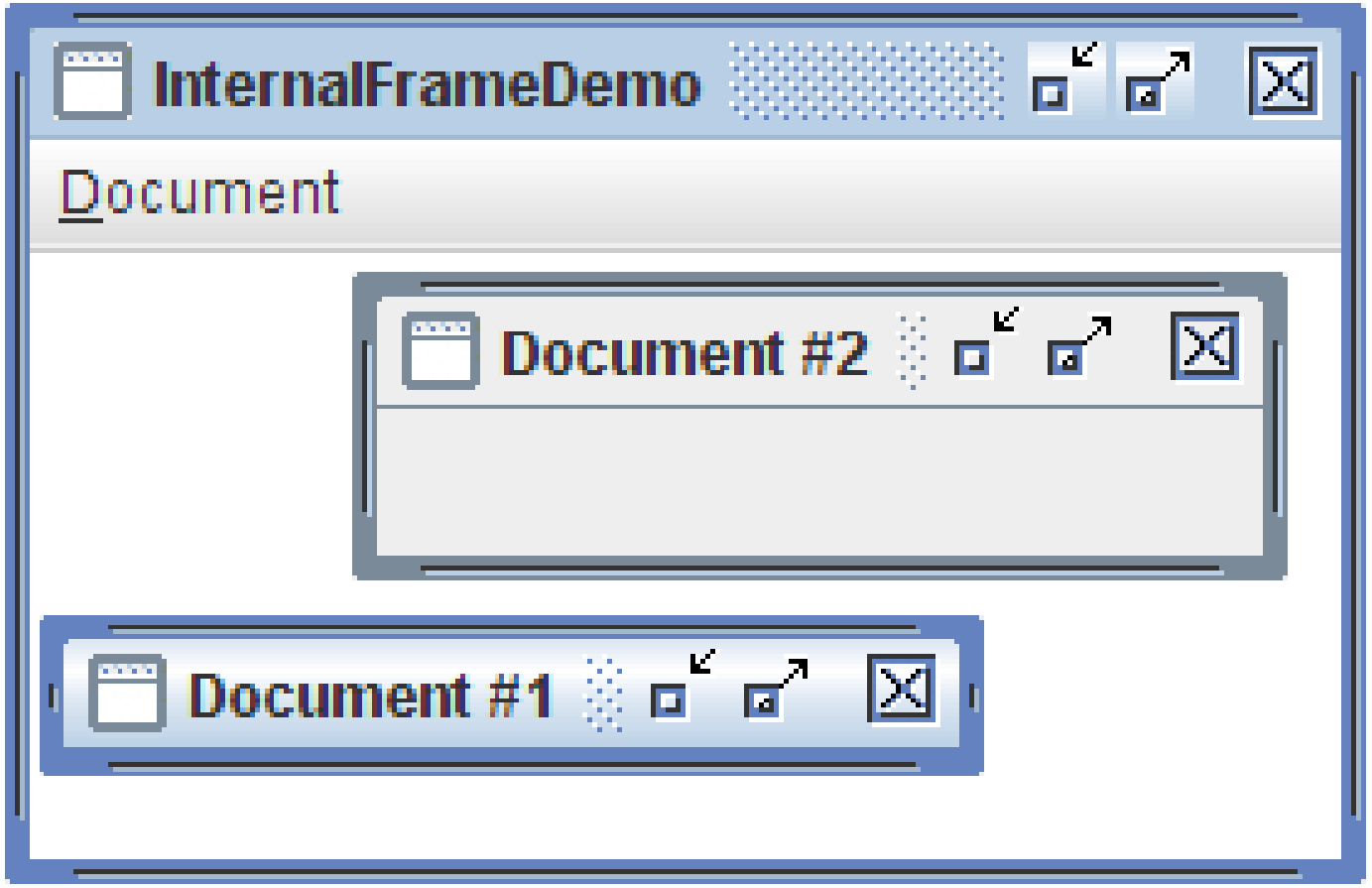
Komponent typu JFileChooser



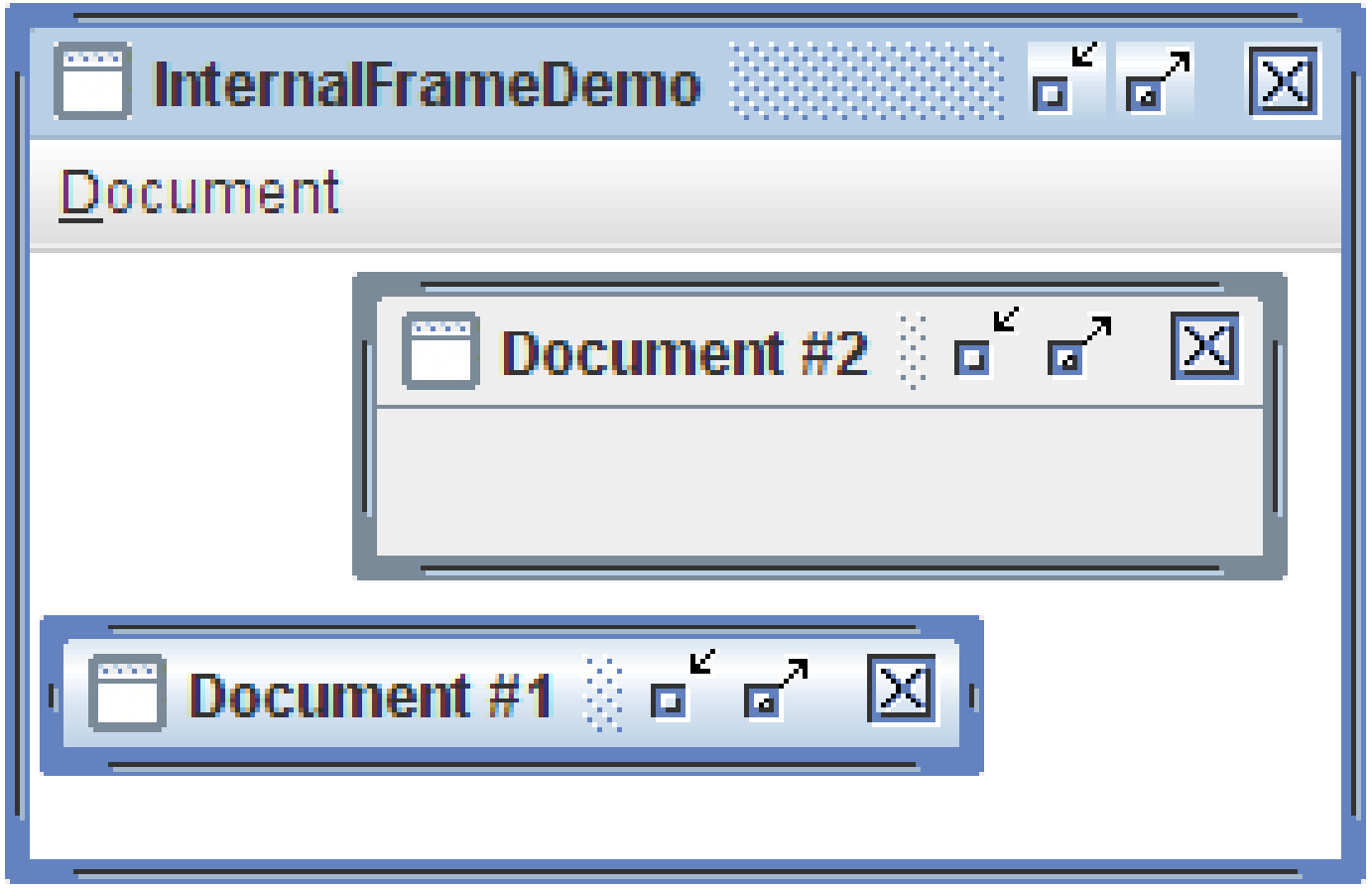
Komponent typu JFormattedTextField



Komponent typu JInternalFrame – okna wewnętrzne

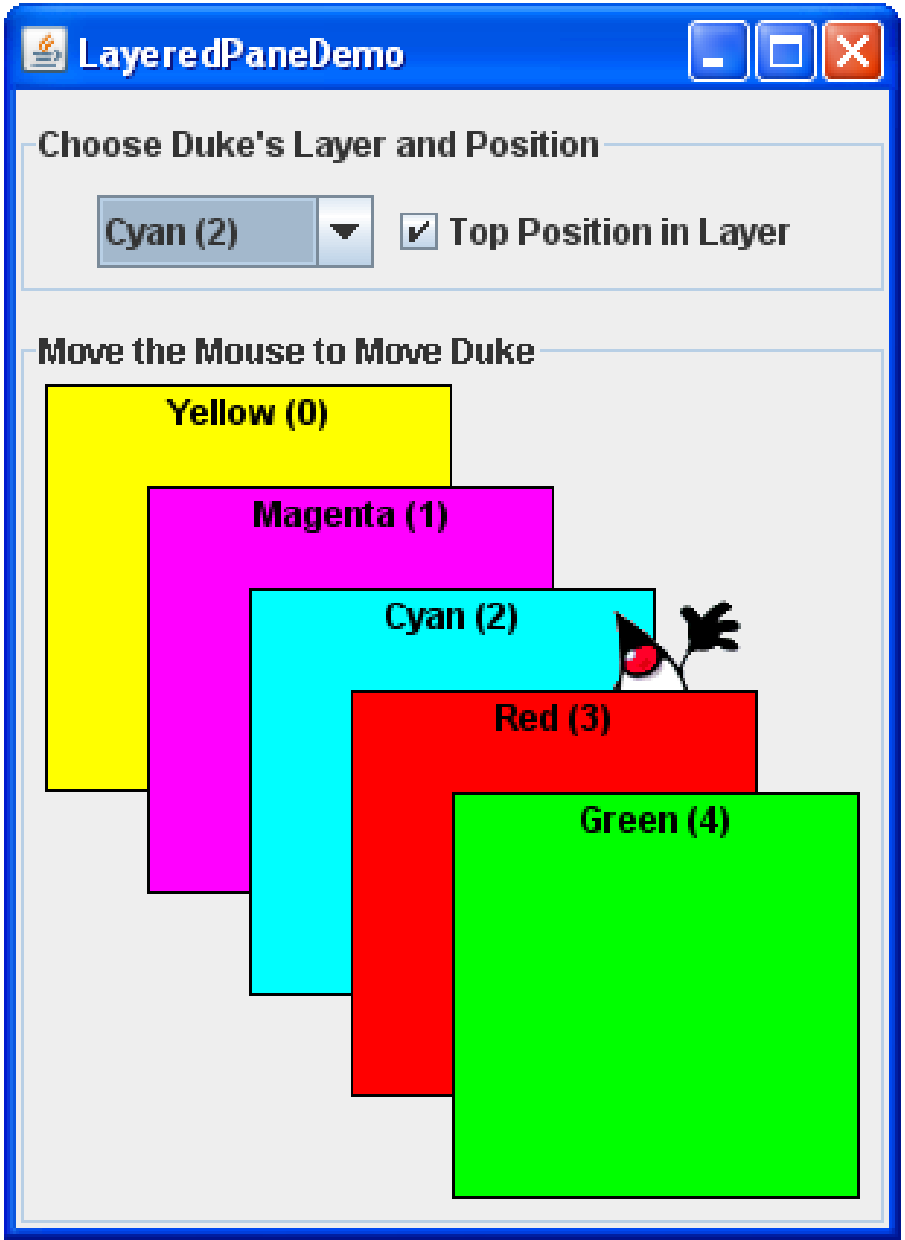


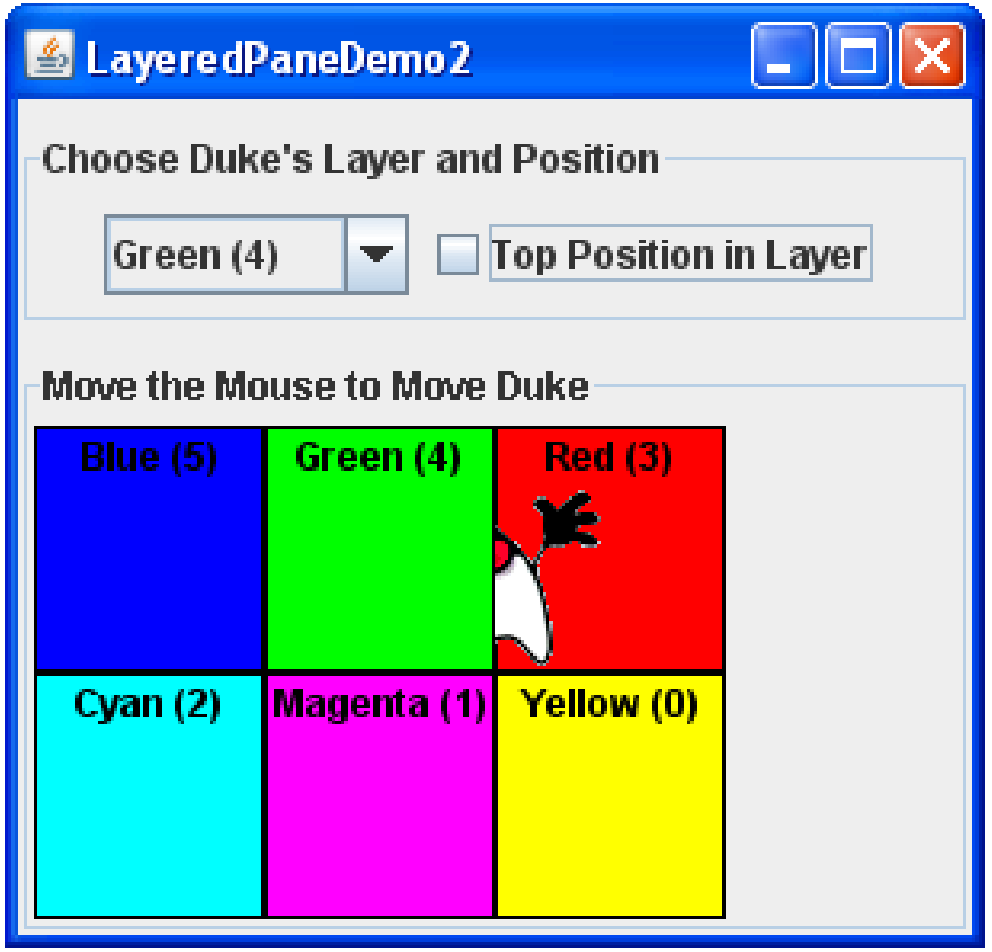
Komponent typu JInternalFrame – okna wewnętrzne



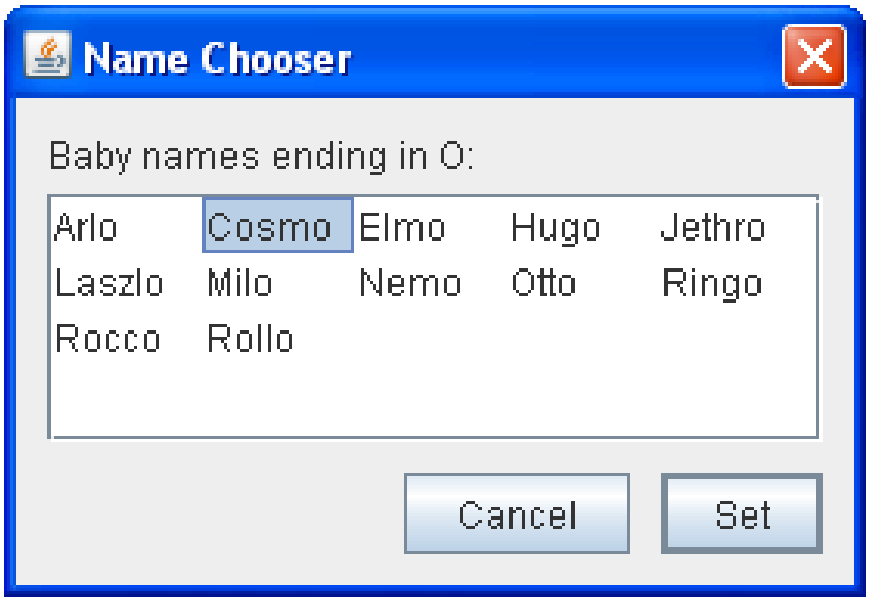
Komponent typu JLabel

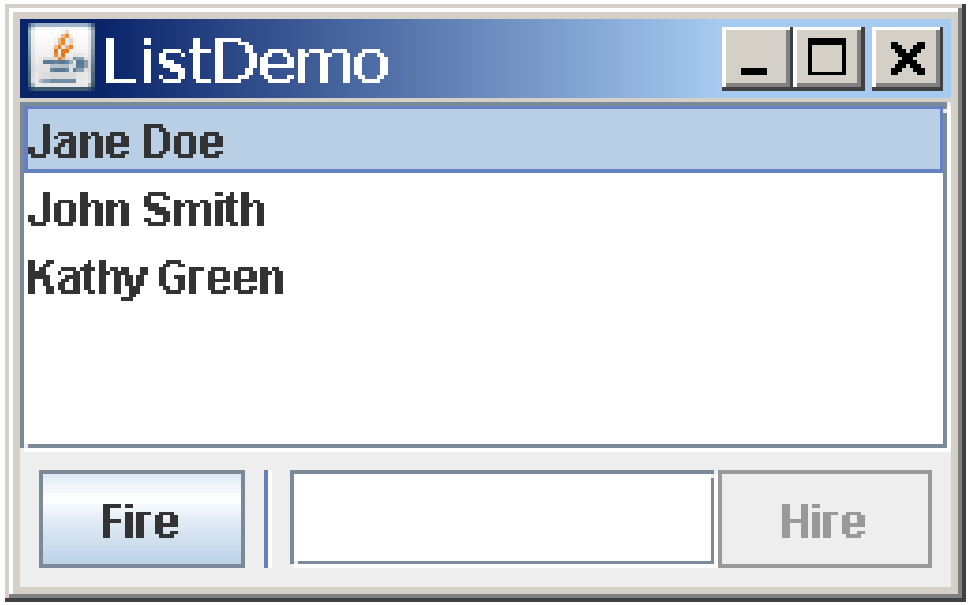
Komponent typu JLayeredPane – zestaw paneli

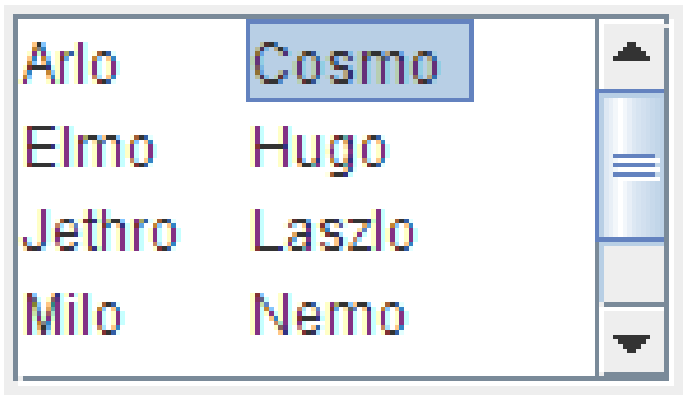




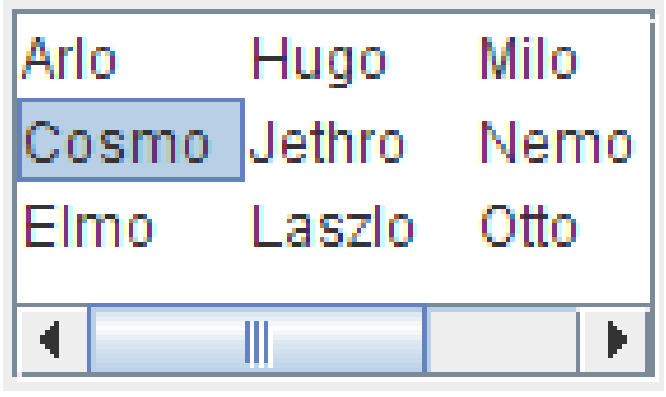
Komponent typu JList



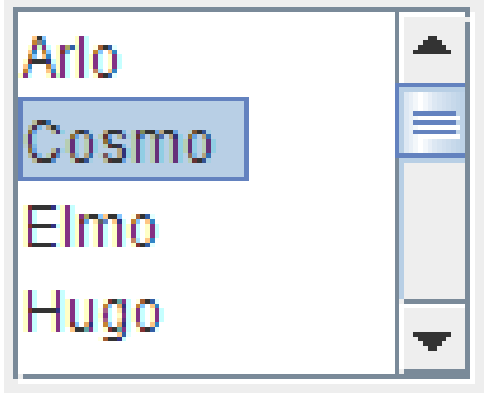




HORIZONTAL_WRAP

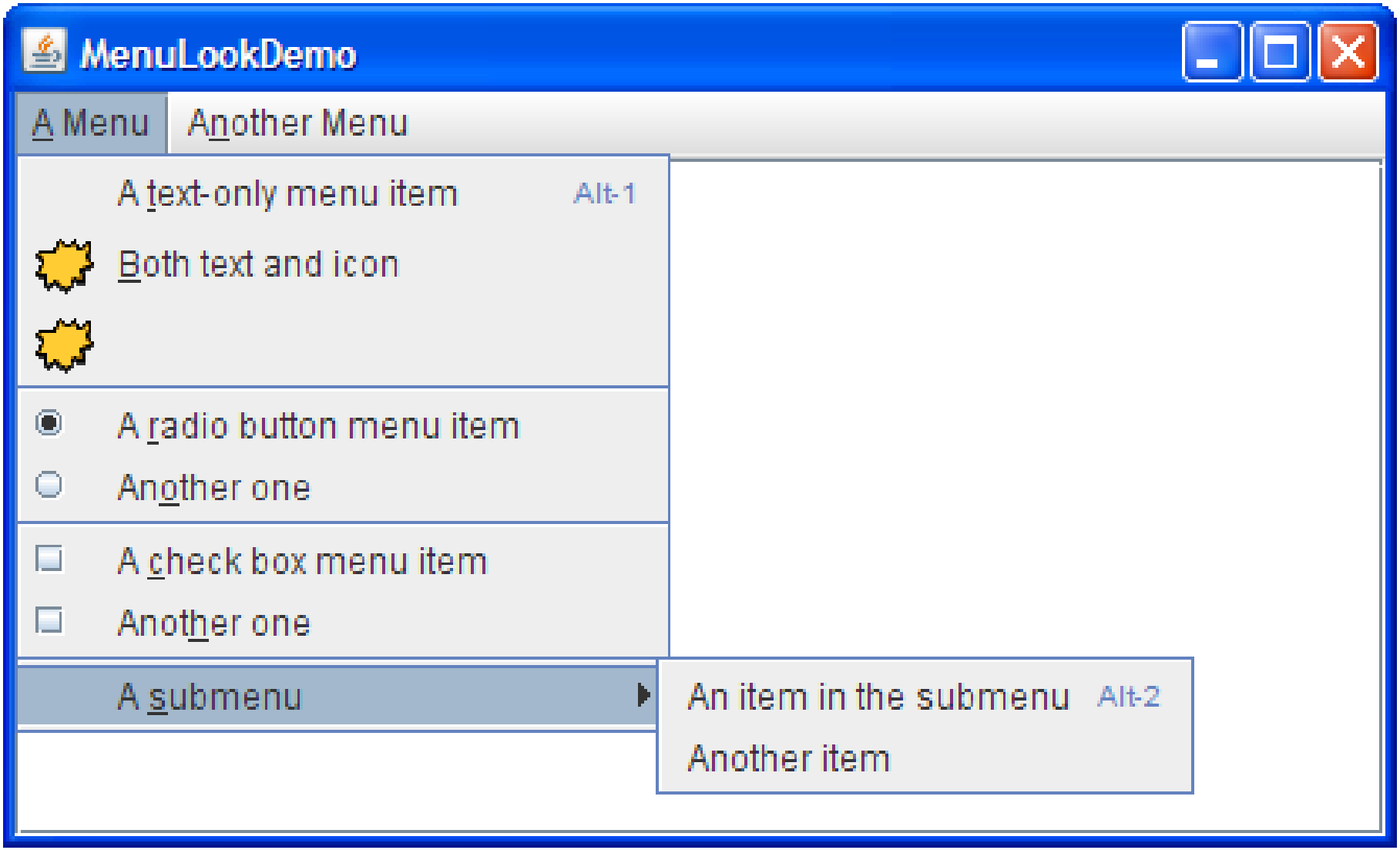


VERTICAL_WRAP

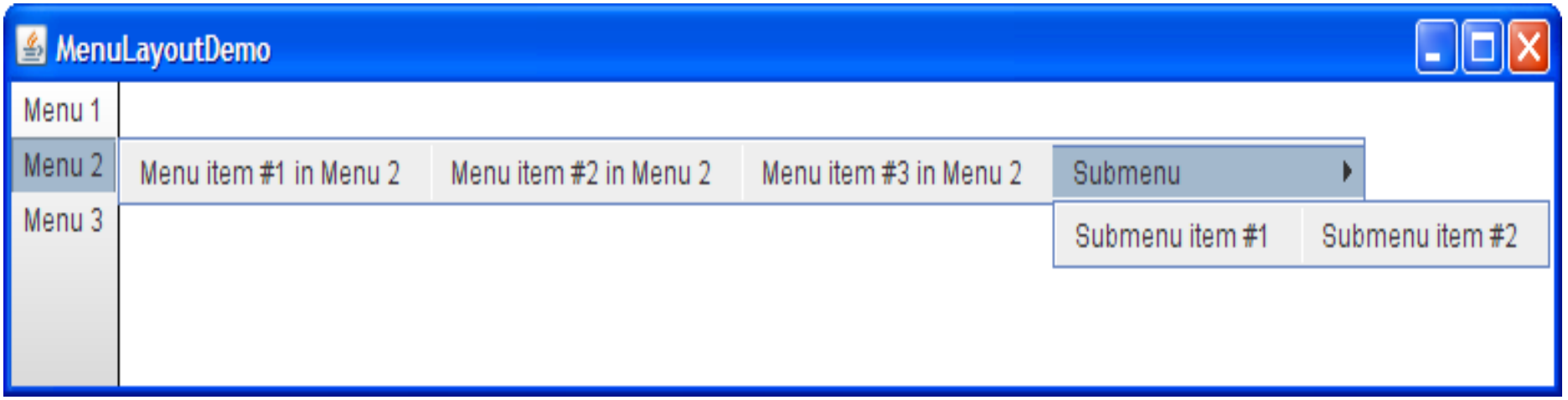


VERTICAL

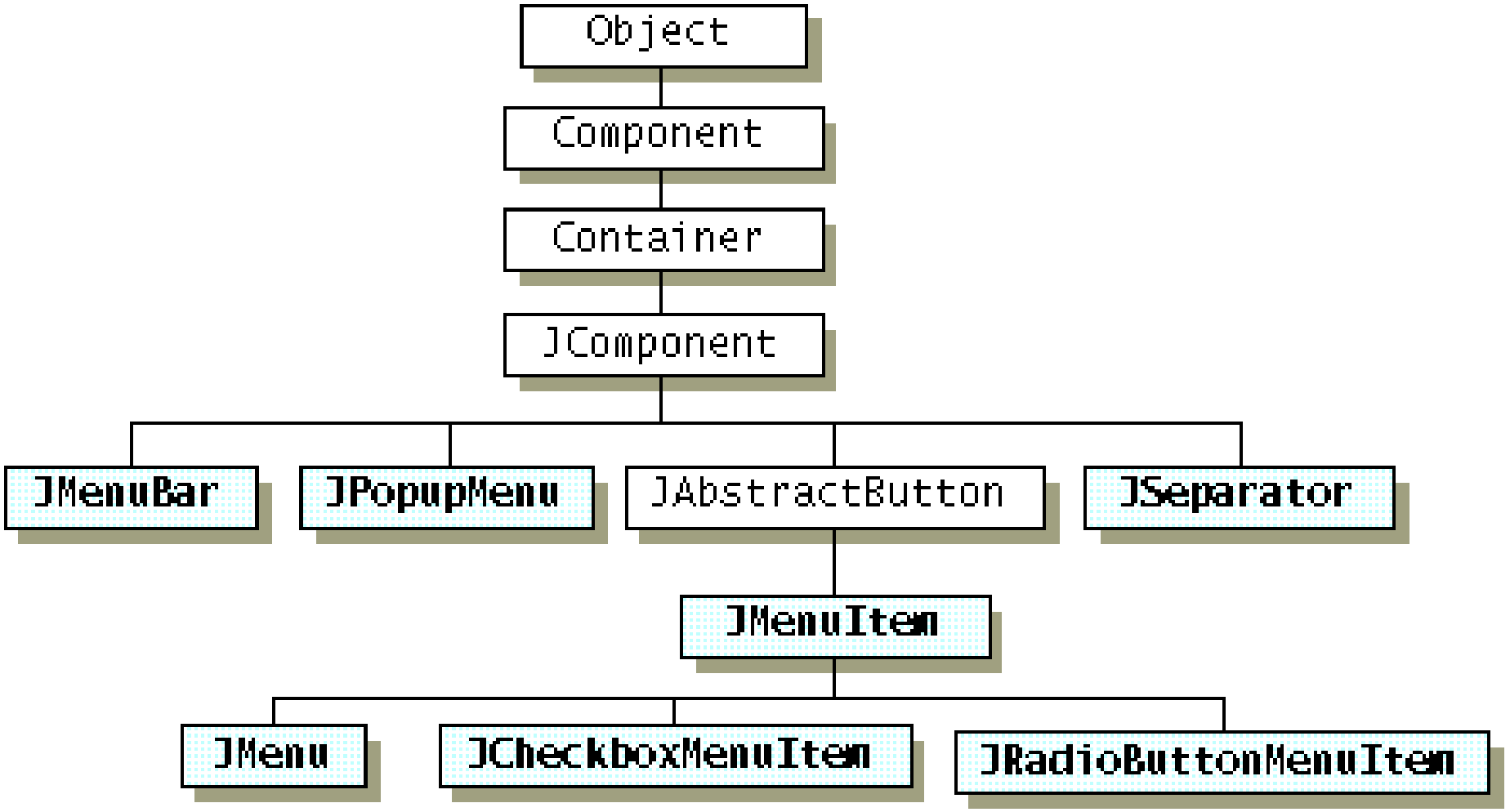
Komponenty do budowy menu



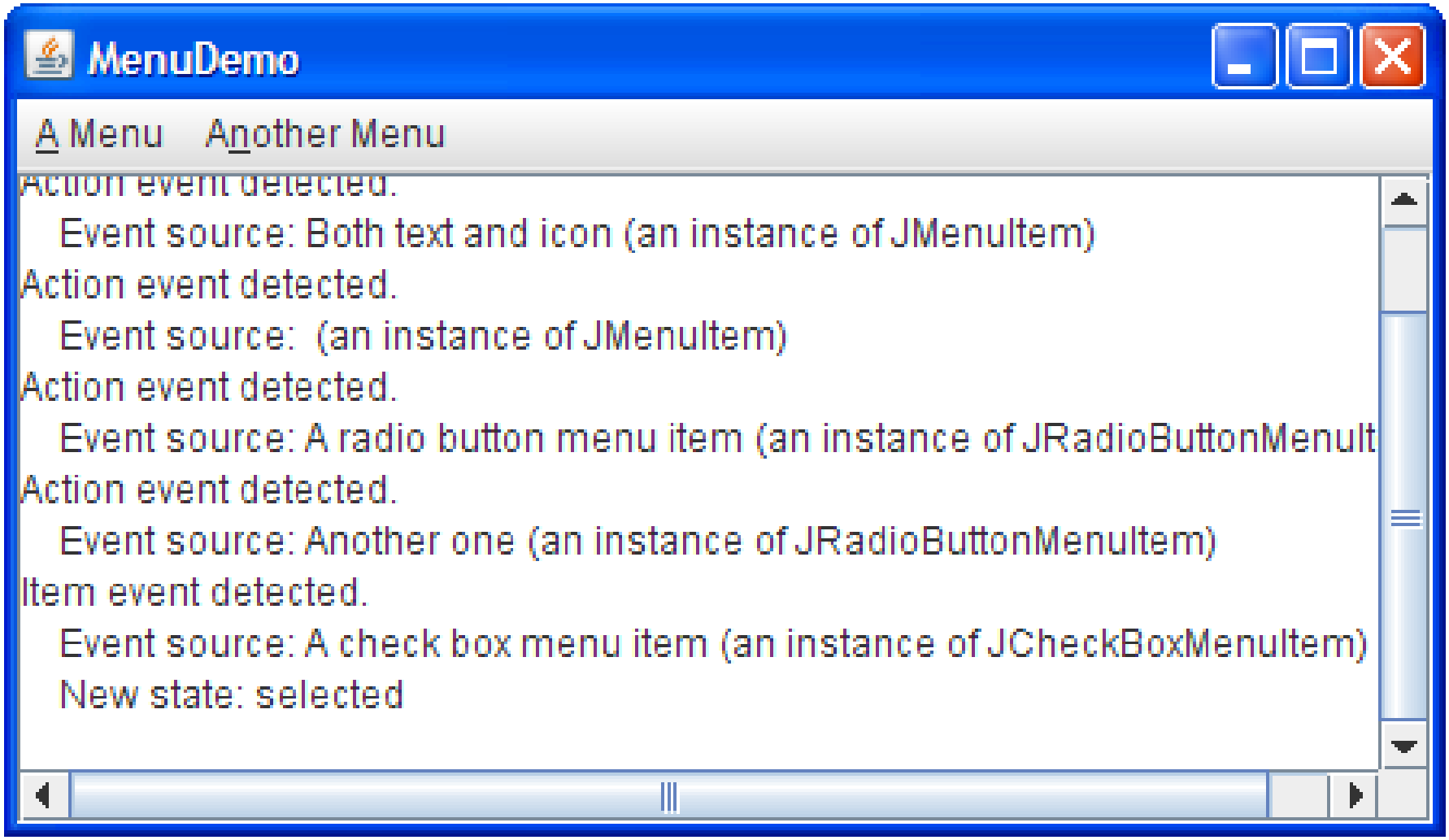
(cd) Komponenty do budowy menu



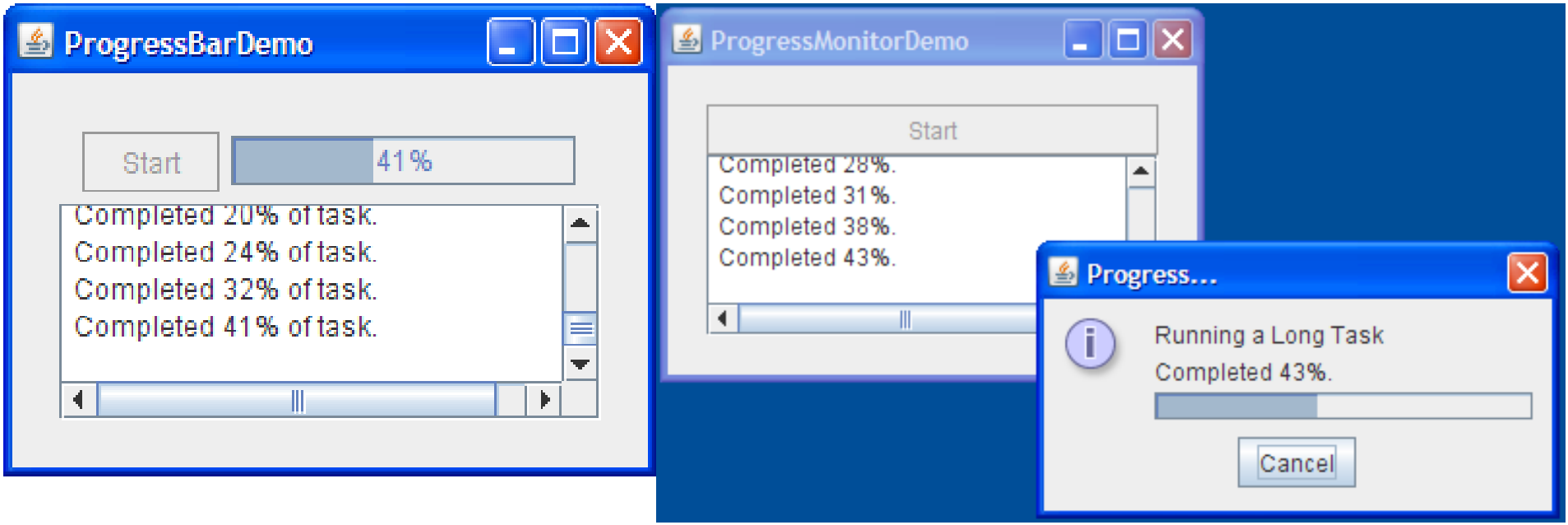
(cd) Komponenty do budowy menu



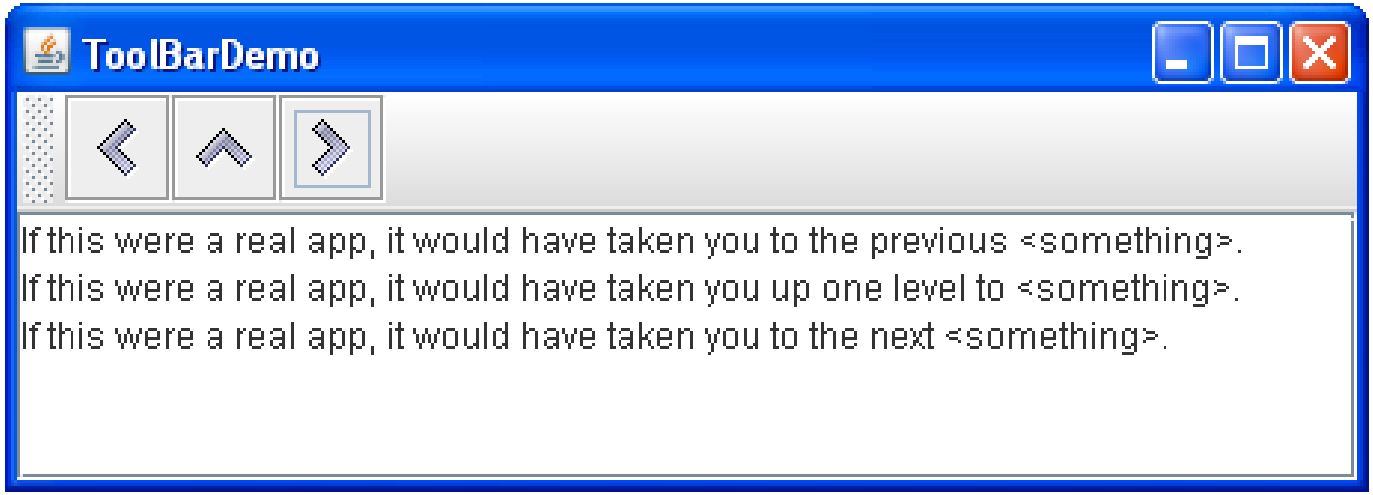
(cd) Komponenty do budowy menu - obsługa zdarzeń



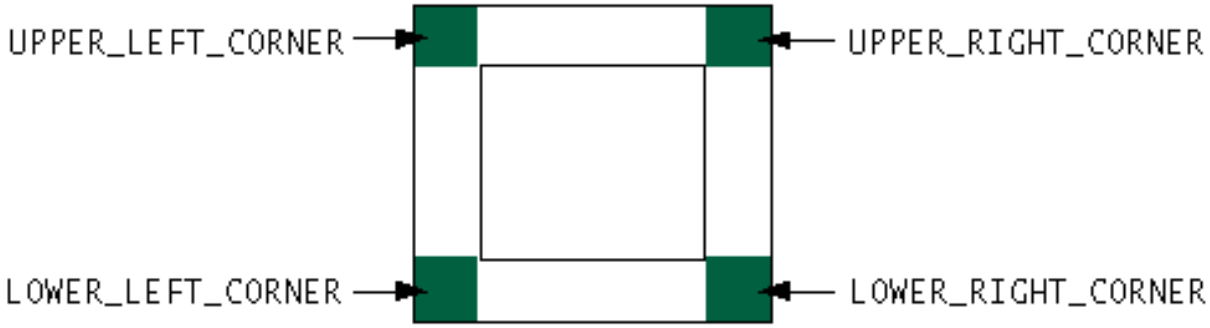
Komponent do obsługi stanu przebiegu JProgressBar

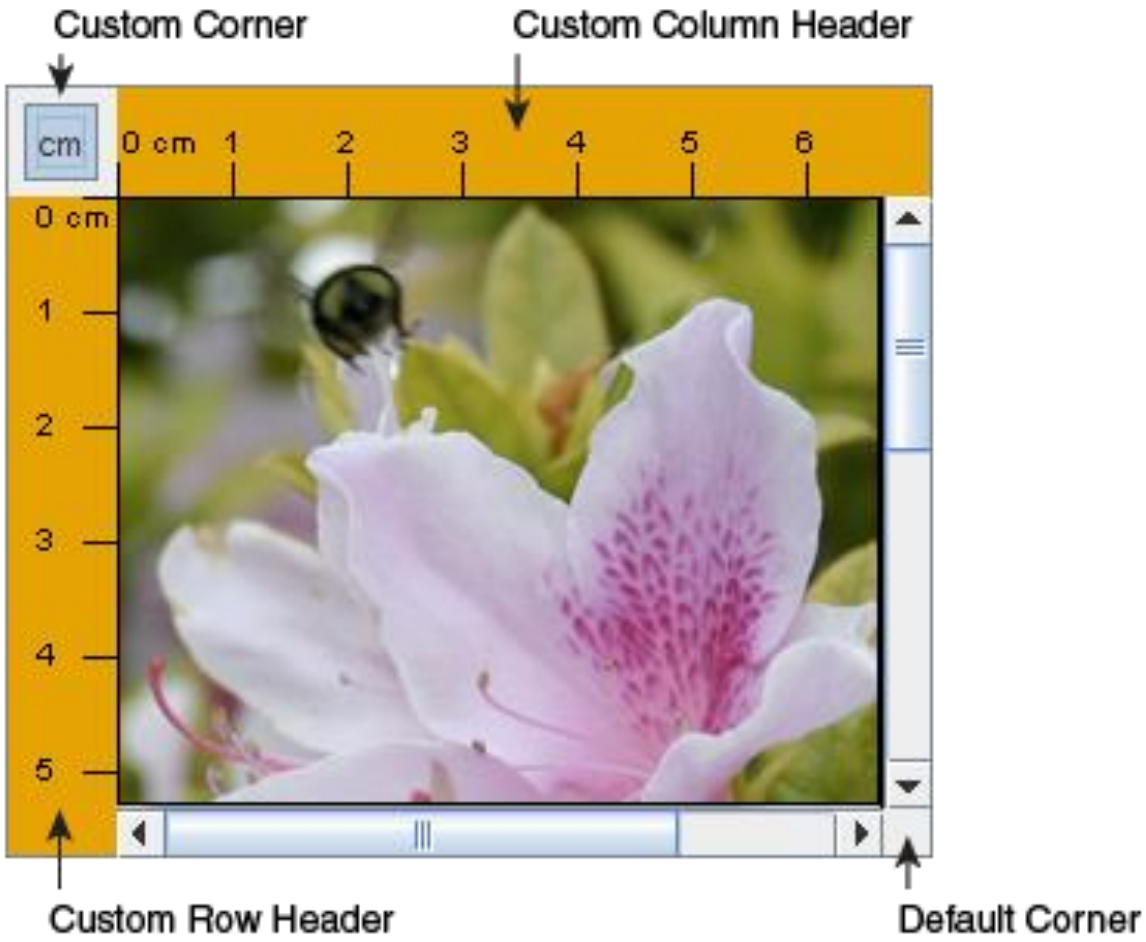


Komponent JScrollPane

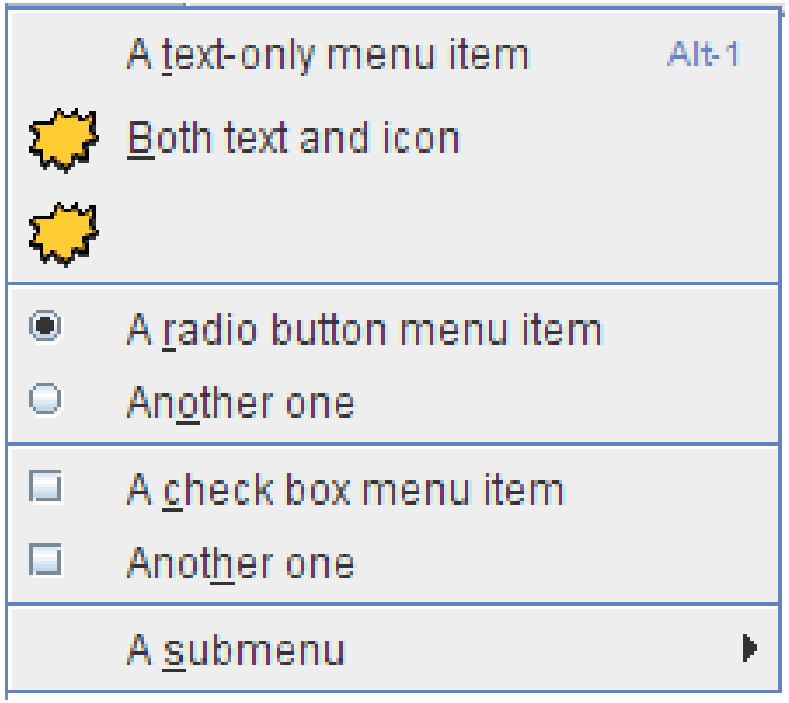


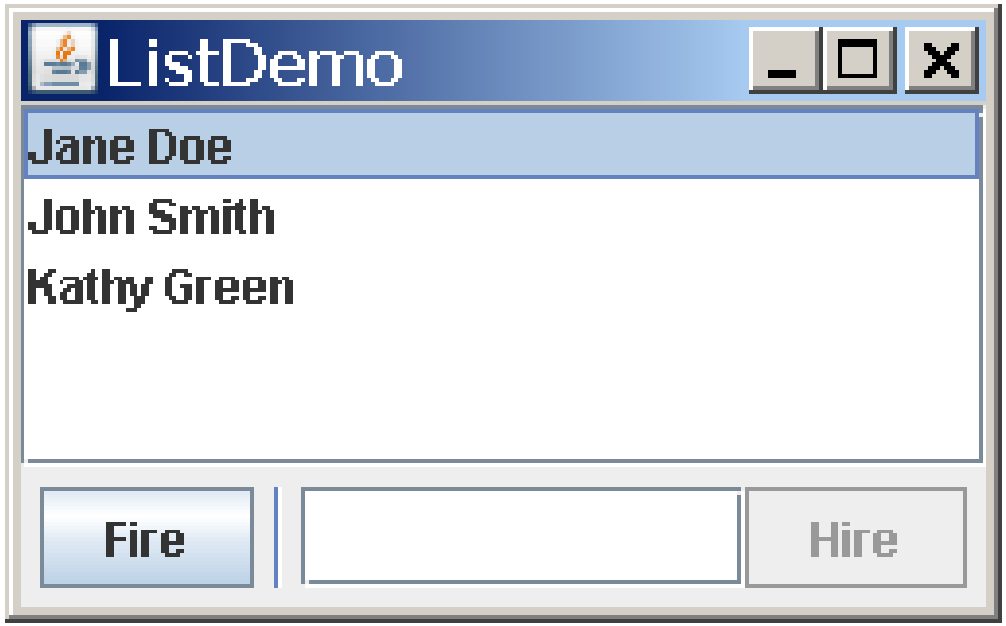
(cd) Komponent typu JScrollPane

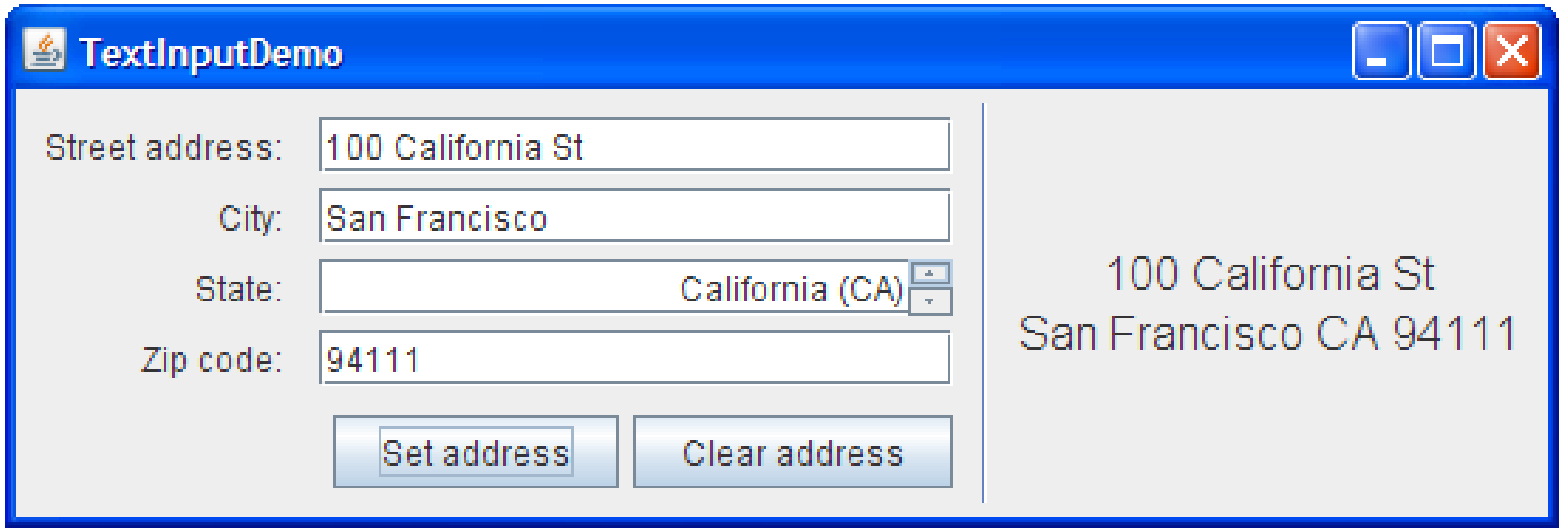




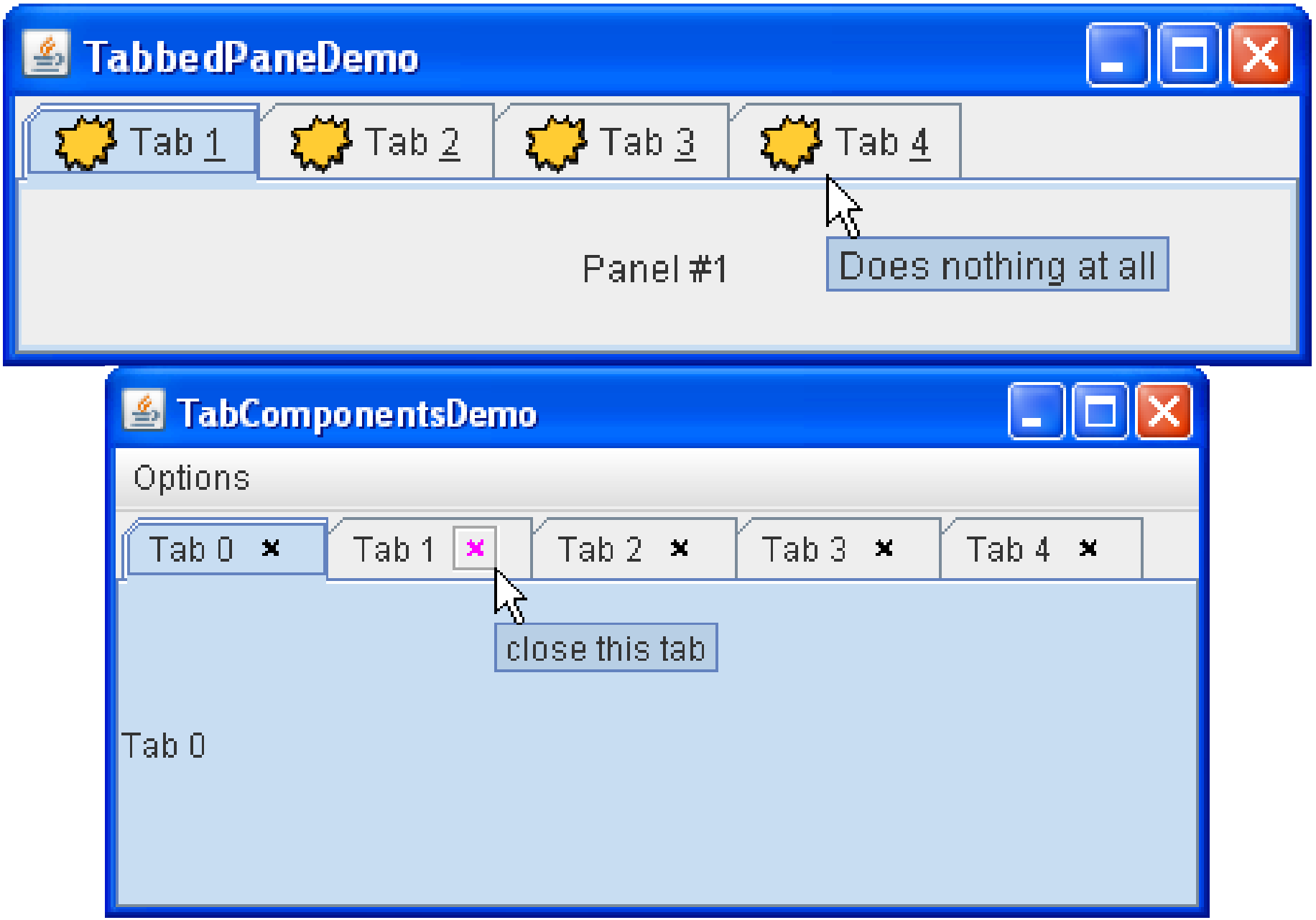
(cd) Komponent JSeparator



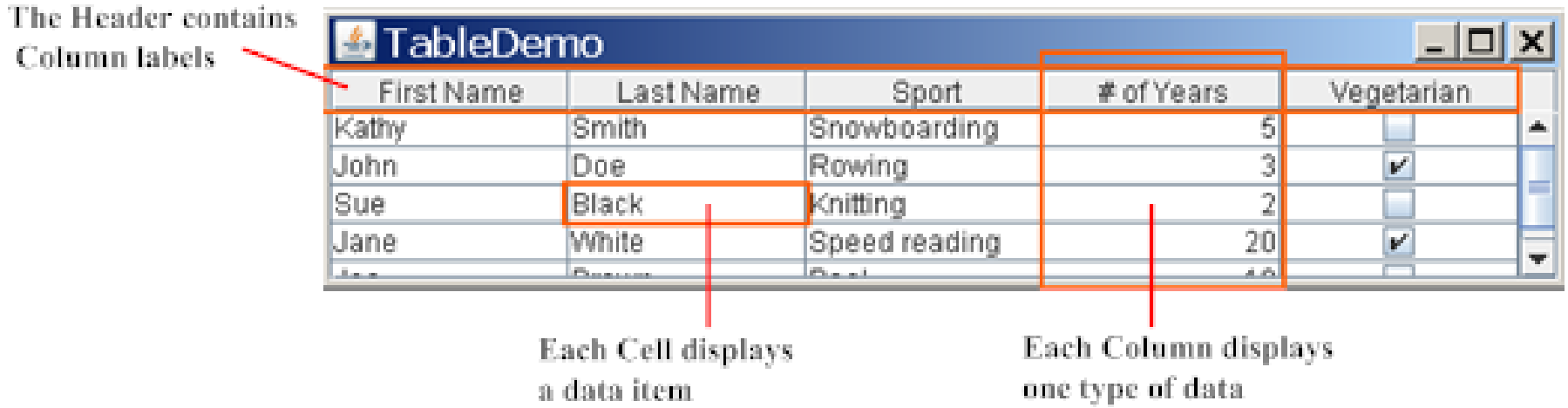


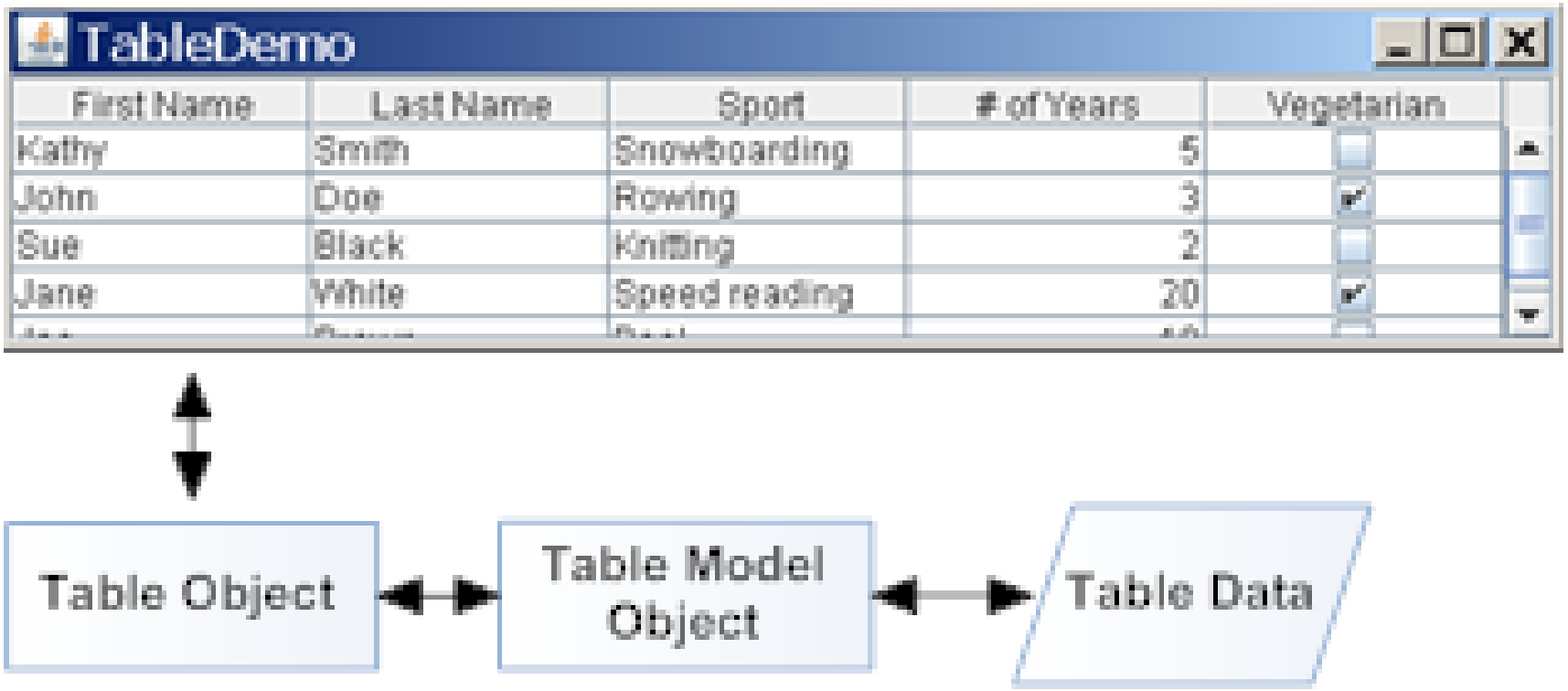


Komponent JTabbedPane

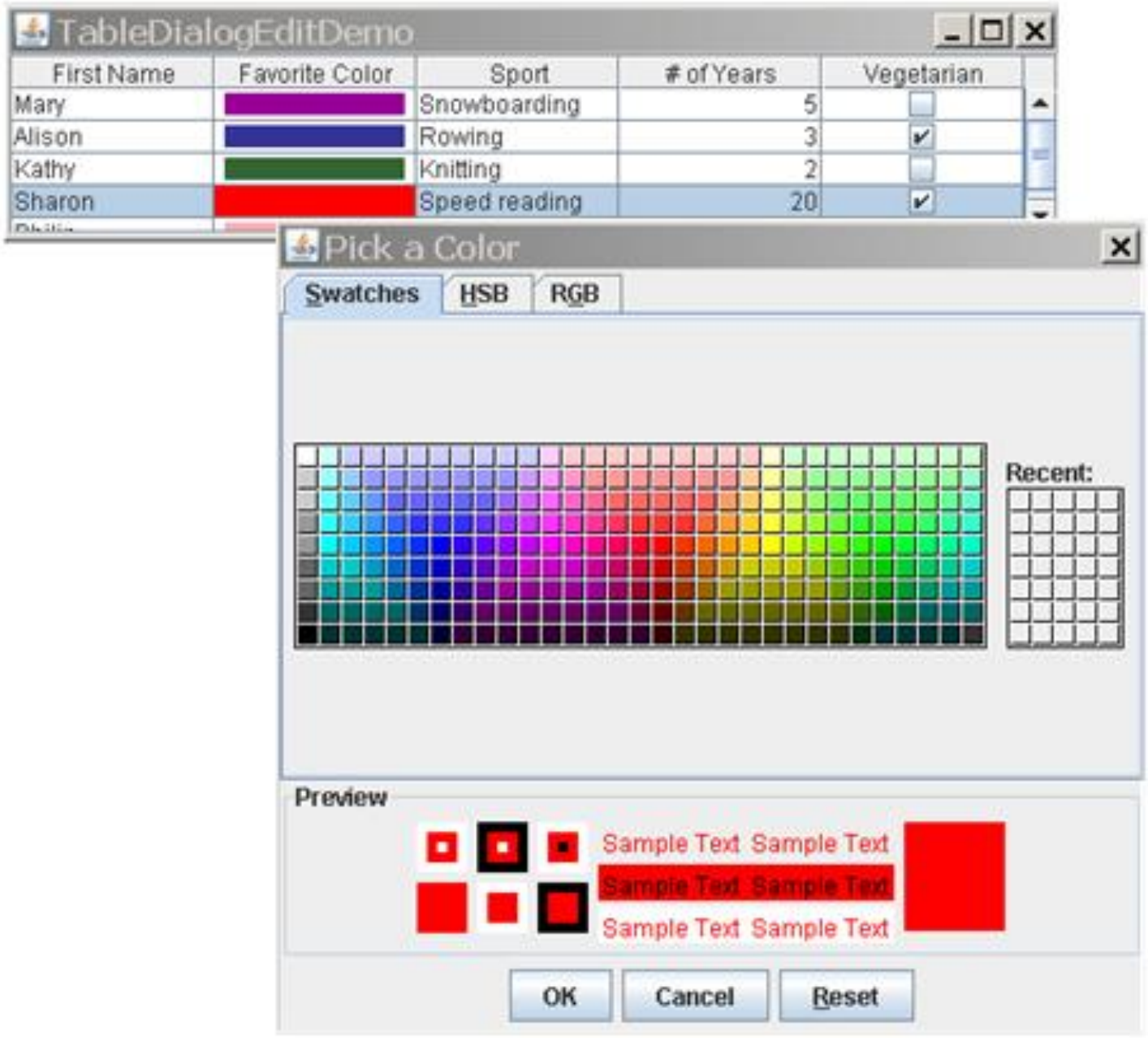


Komponent JTable

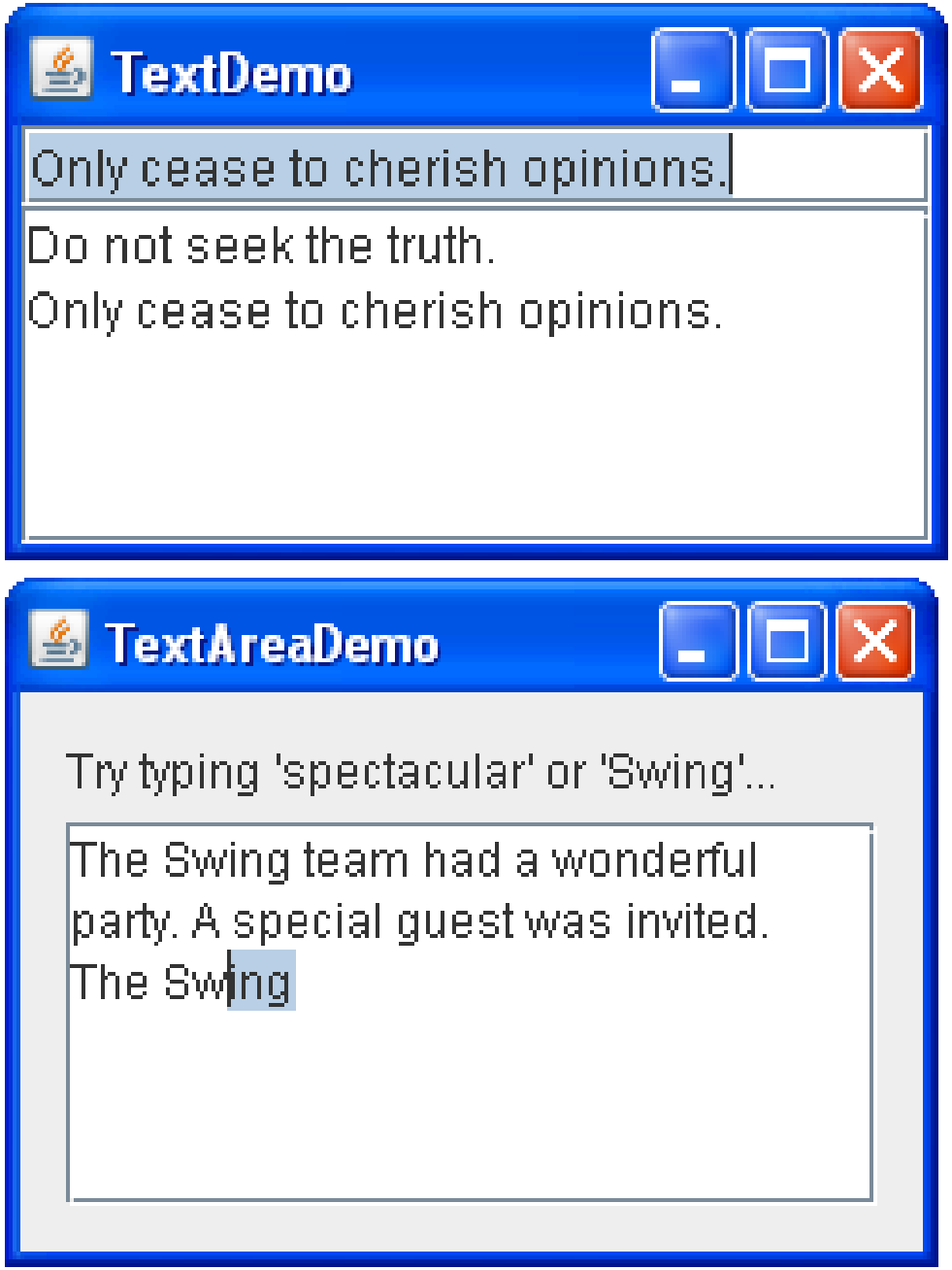




(cd) Komponenty JTable

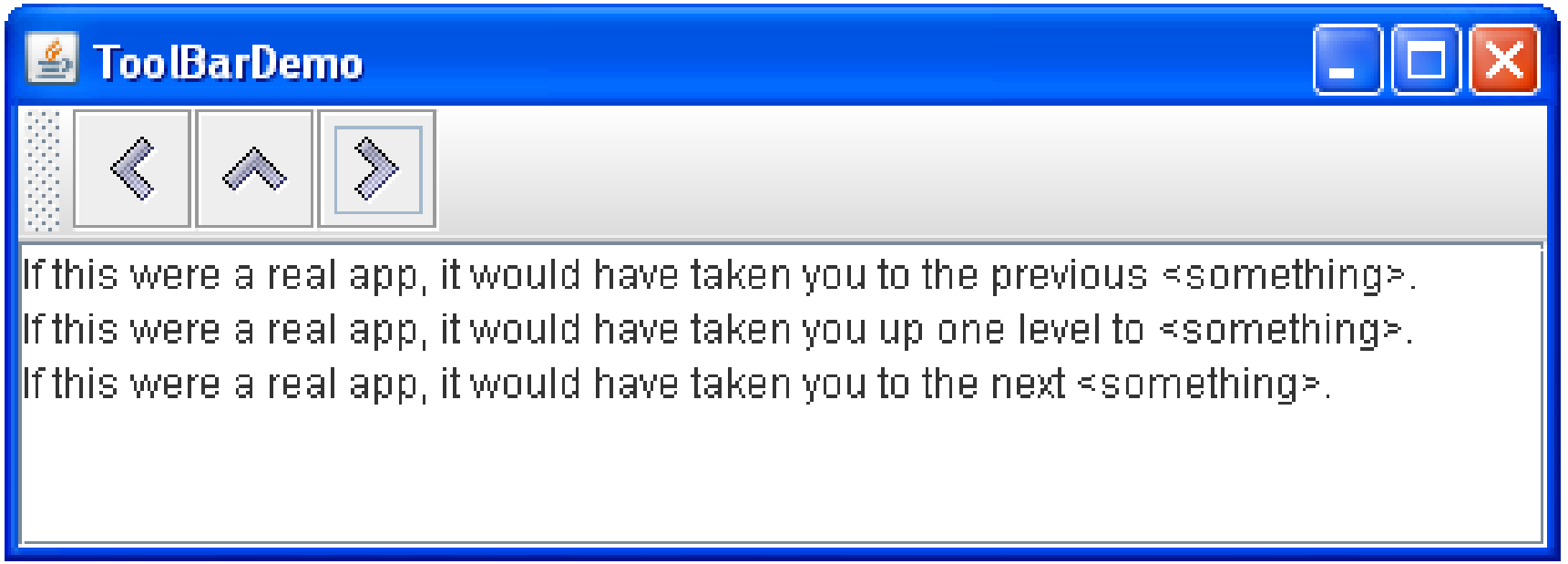


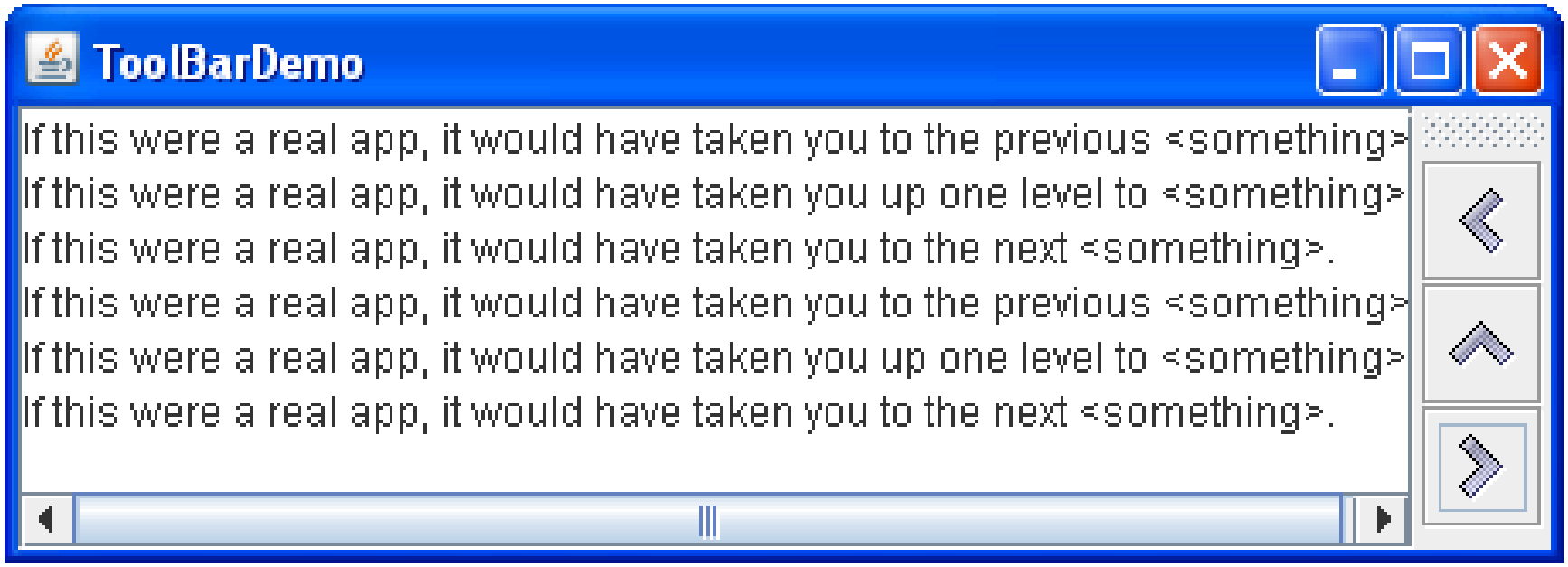
Komponent JTextArea



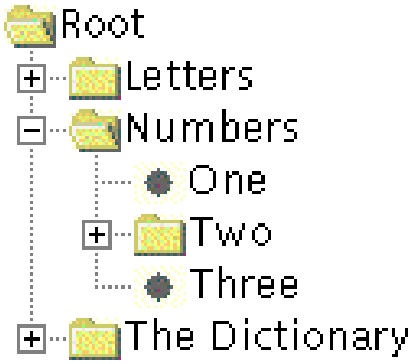
Komponent JTextField

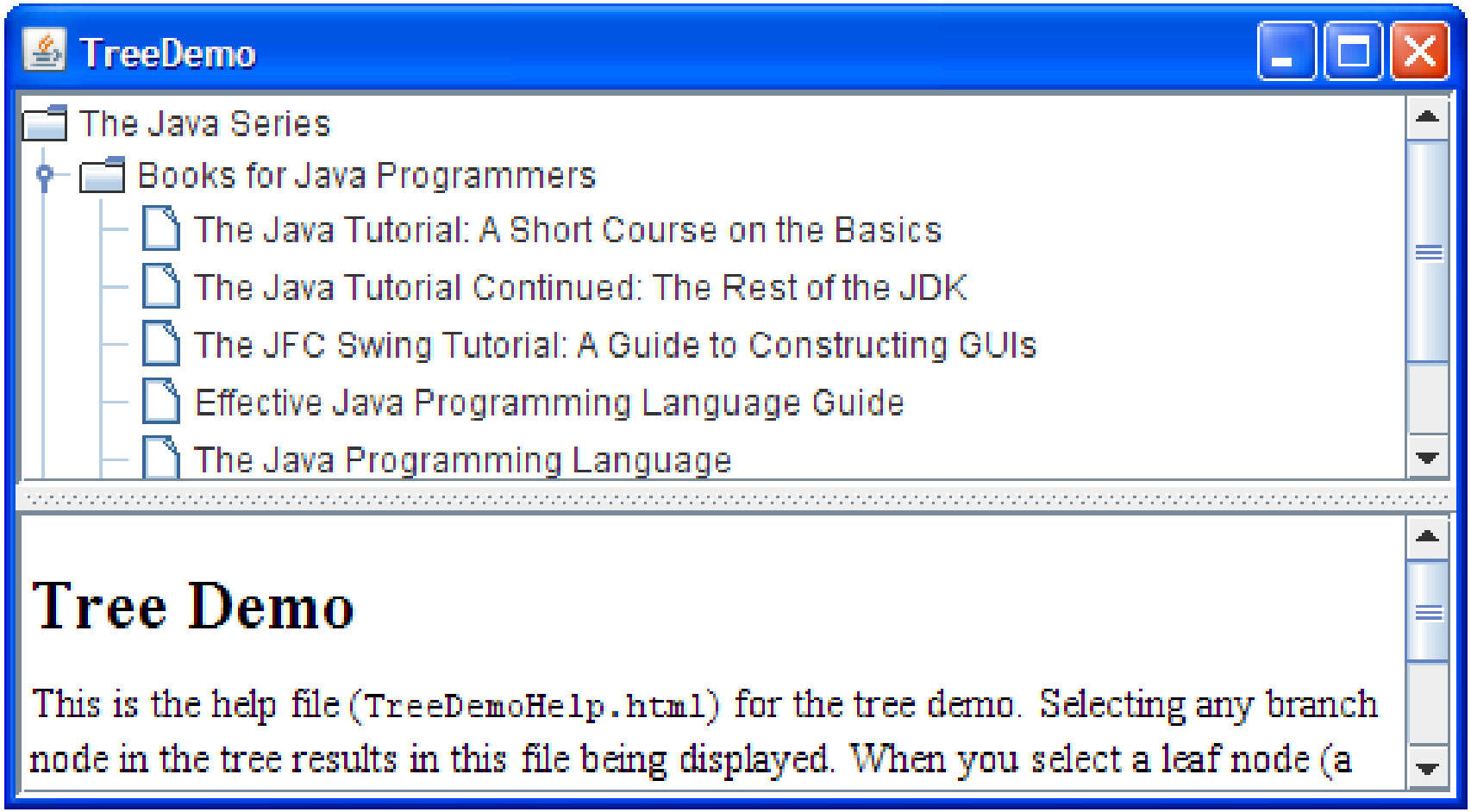
Komponent JToolBar





Komponent JTree





Rodzaje słuchaczy zdarzeń

Wydarzenia można podzielić na dwie grupy:

- zdarzenia o niskim poziomie – zdarzenia myszy i klawiatury
- zdarzenia semantyczne – zdarzenia i akcje komponentów Swing (działania użytkownika lub innych programów np bazy danych)

Zaleca się używać zdarzeń semantycznych, bo sprzyja to przenośności programu

Lista komponentów Swing z wykazem obsługujących ich słuchaczy zdarzeń (zdarzenia semantyczne)

Component	Action Listener	Caret Listener	Change Listener	Document Listener, Undoable Edit Listener	Item Listener	List Selection Listener	Window Listener	Other Types of Listeners
button	✔		✔		✔			
check box	✔		✔		✔			
color chooser			✔					
combo box	✔				✔			
dialog							✔	
editor pane		✔		✔				hyperlink
file chooser	✔							
formatted text field	✔	✔		✔				

(cd) Lista komponentów Swing z wykazem obsługujących ich słuchaczy zdarzeń

frame							✔	
internal frame								internal frame
list						✔		list data
menu								menu
menu item	✔		✔		✔			menu key menu drag mouse
option pane								
password field	✔	✔		✔				
popup menu								popup menu

(cd) Lista komponentów Swing z wykazem obsługujących ich słuchaczy zdarzeń

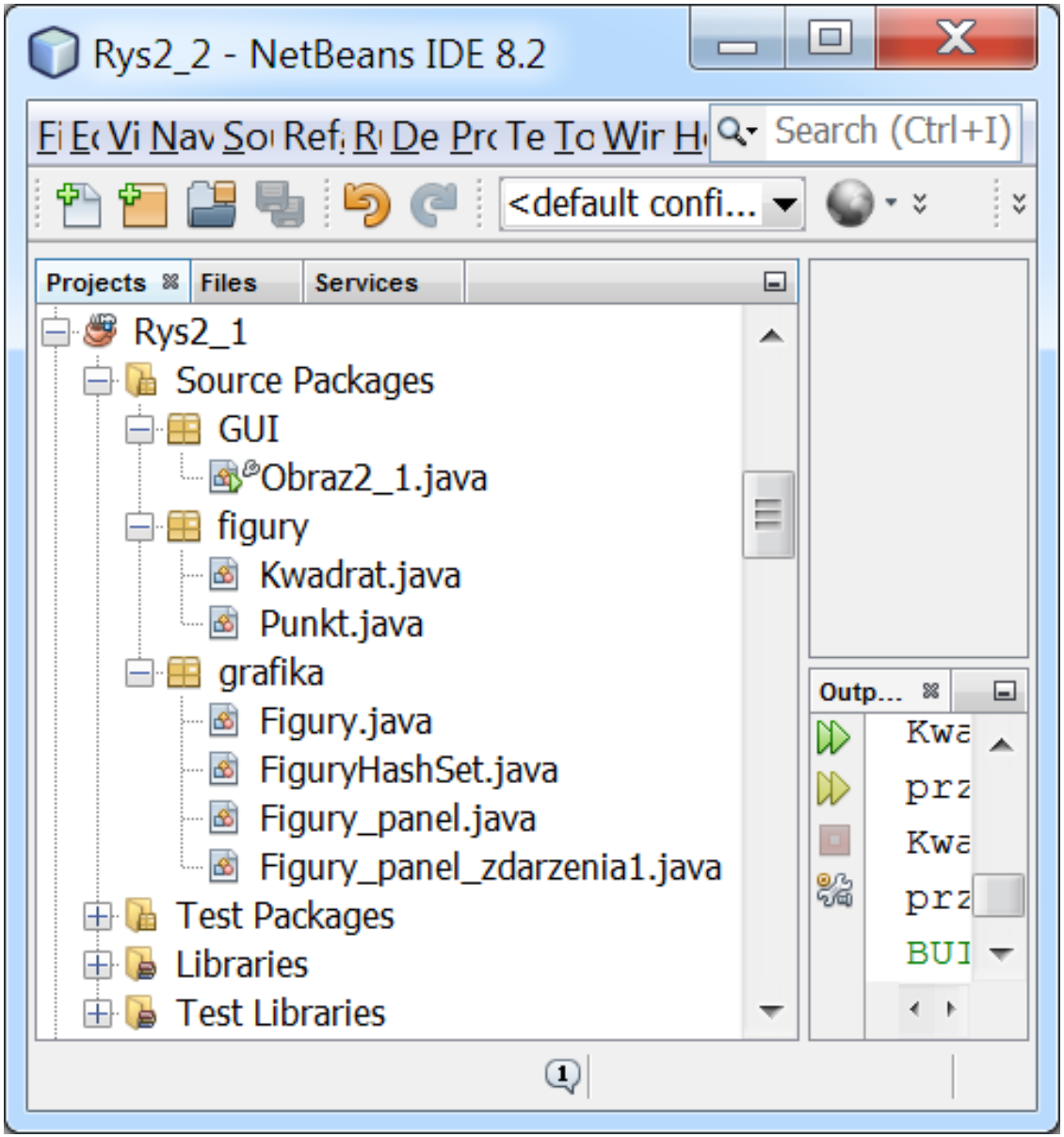
progress bar			✔					
radio button	✔		✔		✔			
slider			✔					
spinner			✔					
tabbed pane			✔					
table						✔		table model table column model cell editor

(cd) Lista komponentów Swing z wykazem obsługujących ich słuchaczy zdarzeń

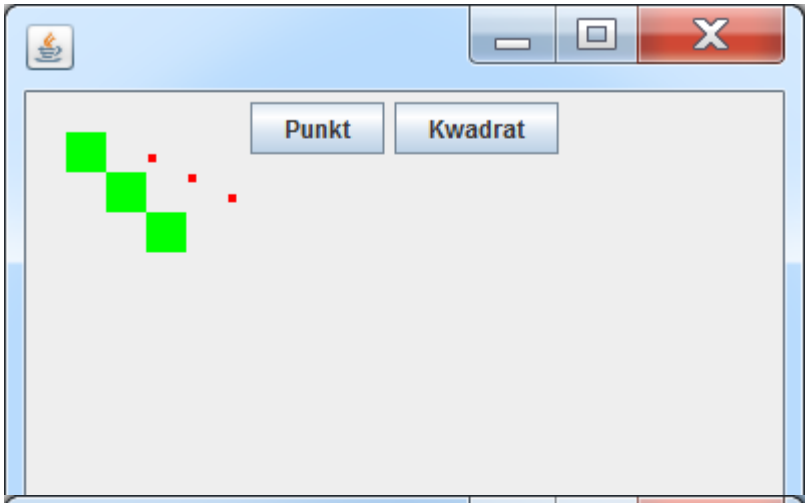
text area		✔		✔				
text field	✔	✔		✔				
text pane		✔		✔				hyperlink
toggle button	✔		✔		✔			
tree								tree expansion tree will expand tree model tree selection
viewport (used by scrollpane)			✔					

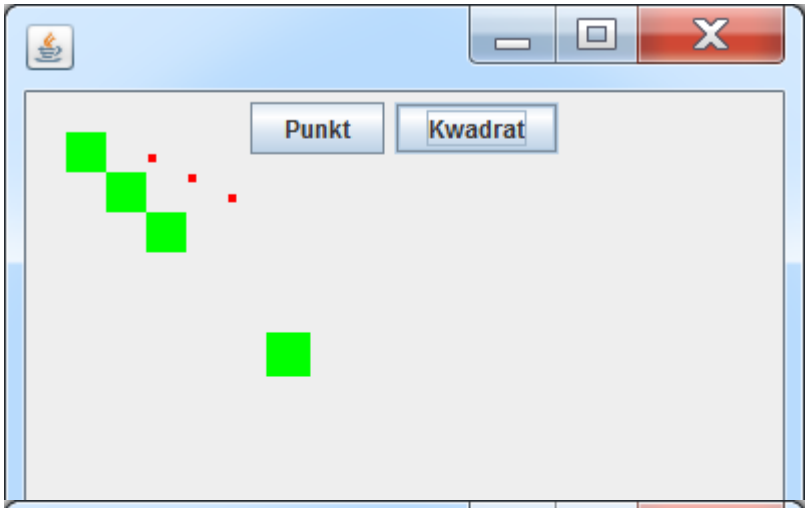
- How to Write an Action Listener
- How to Write a Caret Listener
- How to Write a Change Listener
- How to Write a Component Listener
- How to Write a Container Listener
- How to Write a Document Listener
- How to Write a Focus Listener
- How to Write an Internal Frame Listener
- How to Write an Item Listener
- How to Write a Key Listener
- How to Write a List Data Listener
- How to Write a List Selection Listener
- How to Write a Mouse Listener
- How to Write a Mouse-Motion Listener
- How to Write a Mouse-Wheel Listener
- How to Write a Property Change Listener
- How to Write a Table Model Listener
- How to Write a Tree Expansion Listener
- How to Write a Tree Model Listener
- How to Write a Tree Selection Listener
- How to Write a Tree-Will-Expand Listener
- How to Write an Undoable Edit Listener
- How to Write Window Listeners

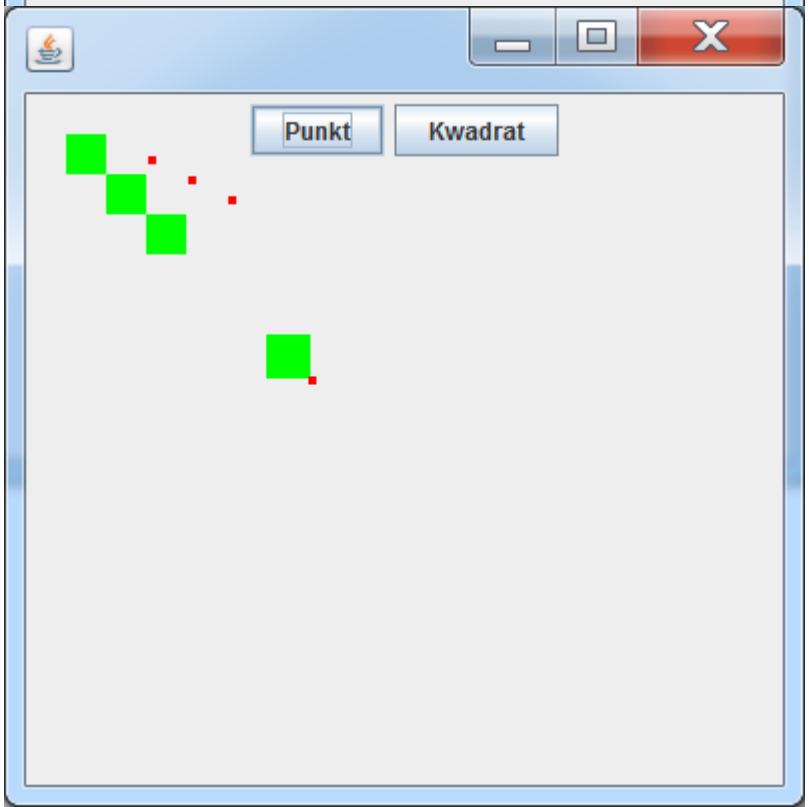
Przykład 1 obsługi zdarzeń – projekt Rys2_1

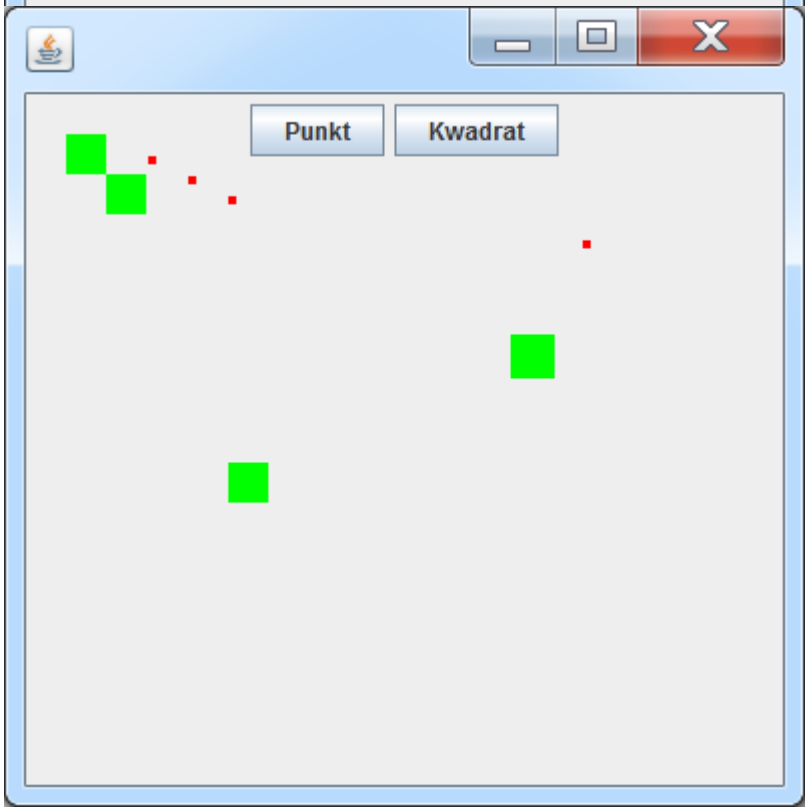


(cd) Przykład 1 obsługi zdarzeń – działanie programu









(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
package figury;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
public class Punkt implements Comparable {
   protected int x, y;

   public Punkt(int wspX, int wspY) {
      x = wspX;
      y = wspY;    }

   @Override
   public int compareTo(Object o) {
      Punkt p = (Punkt) o;
      if ((x == p.x) && (y == p.y)) {
         return 0;
      } else if ((x < p.x) && (y < p.y)) {
         return -1;
      }
      return 1;  }
```

```
public int getX() {    return x;    }

public int getY() {   return y;    }

   @Override
   public int hashCode() {
      int hash = 7;
      hash = 17 * hash + this.x;
      hash = 17 * hash + this.y;
      return hash;  }

   @Override
   public boolean equals(Object obj) {
      if (this == obj) {
         return true;    }
      if (obj == null) {
         return false;    }
      if (getClass() != obj.getClass()) {
         return false;  }
      return this.compareTo(obj) == 0;
   }
```

(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
public double odleglosc(Punkt p) {
    return Math.sqrt((x - p.x) * (x - p.x) + (y - p.y) * (y - p.y));  }

@Override
   public String toString() {
      return "Punkt{" + "x=" + x + ", y=" + y + '}';  }

   public int getDl() {       return 5;    }

   public void przesun(int dx, int dy, int a, int b) {
      x += dx;
      y += dy;
      if (x > a || x < 1)
         x = 5;
      if (y > b || y < 1)
         y = 2;
   }

   public void rysuj(Graphics g) {
      Graphics2D g2D = (Graphics2D) g;
      Color pedzel = new Color(255, 0, 0);
      g2D.setColor(pedzel);
      g2D.fillOval(x, y, 5, 5);}
}
```

(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
package figury;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;

public class Kwadrat extends Punkt {

   protected int dlugosc;

 public Kwadrat(int wspX, int wspY, int dlugosc_) {
      super(wspX, wspY);
      dlugosc = dlugosc_;  }

   public double odleglosc() {
      return Math.sqrt(x * x + y * y);
   }

   @Override
   public double odleglosc(Punkt p) {
      return odleglosc() + super.odleglosc(p); }

   public int getDl() {
      return dlugosc;  }
```

```
   @Override
   public int hashCode() {
      int hash = 5;
      hash = 97 * hash + this.dlugosc;
      return hash;  }

   @Override
   public boolean equals(Object obj) {
      if (this == obj) {
         return true;   }
      if (obj == null) {
         return false;   }
      if (getClass() != obj.getClass()) {
         return false;
       }
      final Kwadrat other = (Kwadrat) obj;
      if (this.dlugosc != other.dlugosc) {
         return false; }
     return this.compareTo(obj)==0;
   }
```

(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
   @Override
    public String toString() {
        String s = super.toString();
        return "Kwadrat{" + "dlugosc=" + dlugosc + '}' + " i dziedzicze od " + s;
    }

  public void rysuj(Graphics g) {
        Graphics2D g2D = (Graphics2D) g;
        Color pedzel = new Color(0, 255, 0);
        g2D.setColor(pedzel);
        g2D.fillRect(x, y, dlugosc, dlugosc);
      }
}
```

(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
package grafika;

import figury.Kwadrat;
import figury.Punkt;
import java.awt.Graphics;
import java.util.Collection;

public class Figury {

  protected int N = 3;
  public Collection<Punkt> figury;

  protected Punkt biezacy;

  public void polozenie() {
    for (Punkt figura : figury) {
      boolean p = figura instanceof Kwadrat;
     System.out.println(  p +
     ", ze jestem kwadratem, bo jestem " +
      figura.toString()
     + ", X=" + figura.getX()
     + ", Y=" + figura.getY()
     + ", odleglosc=" + figura.odleglosc(figura)); }
   }
 public Punkt getBiezacy() {   return biezacy;  }
```

```
public boolean Clicked(int x_, int y_) {
   for (Punkt figura : figury) {
     if (figura.getX() + figura.getDl() >= x_
        && figura.getX() <= x_
        && figura.getY() + figura.getDl() >= y_
        && figura.getY() <= y_) {
      /*if(figura.lezy_na(x_, y_)) */
         biezacy = figura;
        return true;   } }
      return false;   }

 public void rysuj_figury(Graphics g) {
      for (Punkt figura : figury) {
         figura.rysuj(g);  }
   }
public boolean przesun(int x, int y, int dl,
                                    int szer) {
      if (biezacy != null) {
         biezacy.przesun(x, y, dl, szer);
         return true;   }
      return false;   }
}
```

(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
package grafika;

import figury.Kwadrat;
import figury.Punkt;
import java.util.ArrayList;

public class FiguryHashSet extends Figury {

  public void pojemnik() {
      figury = new HashSet();  }

  public boolean wyszukaj(Punkt p) {
      return figury.contains(p);
   }

 public boolean wyszukaj(int dane[]) {
    Punkt p;
    if (dane[0] == 0) {
      p = new Punkt(dane[1], dane[2]);
    } else {
    p = new Kwadrat(dane[1], dane[2], dane[3]);
   }
   return wyszukaj(p);  }
```

```
 public void wypelnij() {
   for (int i = 0; i < N; i++) {
      figury.add(new Punkt(20 * (N + i),
                           10 * (N + i)));
     figury.add(new Kwadrat((i + 1) * 20,
                           (i + 1) * 20, 20));  }
   }

 public void Dodaj_punkt() {
       int a = figury.size();
       figury.add(new Punkt(a * 20, a * 20)); }

 public void Dodaj_kwadrat() {
   int a = figury.size();
   figury.add(new Kwadrat(a * 20, a * 20, 22));
  }

 public void init() {
       pojemnik();
       wypelnij();
       polozenie(); }
}
```

(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
package grafika;

import java.awt.Graphics;
import javax.swing.JPanel;

public class Figury_panel extends JPanel{
   FiguryHashSet kontroler;

  @Override
   protected void paintComponent(Graphics g) {
      super.paintComponent(g);
      kontroler.rysuj_figury(g);
   }

   public void init()
   {   kontroler=new FiguryArrayList();
      kontroler.pojemnik();
      kontroler.wypelnij();
   }

   public FiguryArrayList getKontroler() {
      return kontroler;  }
}
```

(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
package grafika;

import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.Random;
import javax.swing.JButton;

public class Figury_panel_zdarzenia1 extends Figury_panel
implements KeyListener, MouseListener, ActionListener{

   protected JButton punkt = new JButton("Punkt");
   protected JButton kwadrat = new JButton("Kwadrat");

@Override
   public void mouseClicked(MouseEvent e)
   {   int x_ = e.getX();
       int y_ = e.getY();
       kontroler.Clicked(x_, y_);
       requestFocus();
       repaint(); }
```

```
public void mouseEntered(
        MouseEvent e)
  {  }

   public void mouseExited(
        MouseEvent e)
   {  }

   public void mousePressed(
        MouseEvent e)
   {  }

   public void mouseReleased(
         MouseEvent e)
   {  }
```

(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
@Override
  public void keyPressed(KeyEvent evt) {
     int dx = 0, dy = 0, zmiana;
     Random r = new Random();
     if (evt.isShiftDown()) {
        zmiana = r.nextInt(100);
     } else {
        zmiana = r.nextInt(10);  }
     switch (evt.getKeyCode()) {
        case KeyEvent.VK_LEFT:
           dx -= zmiana;    break;
        case KeyEvent.VK_RIGHT:
           dx += zmiana;    break;
        case KeyEvent.VK_UP:
           dy -= zmiana;    break;
        case KeyEvent.VK_DOWN:
           dy += zmiana;    break;
     }
     kontroler.przesun(dx, dy,
                    getWidth(), getHeight());
     repaint();
}
```

```
@Override
  public void keyReleased(KeyEvent evt) {   }

  @Override
  public void keyTyped(KeyEvent evt)
    /*obsługa klawiszy (a, A, #, ...)*/ {     }


@Override
  public void actionPerformed(ActionEvent e)
{
     Object zrodlo = e.getSource();
     if (zrodlo == punkt) {
        kontroler.Dodaj_punkt();
     } else if (zrodlo == kwadrat) {
        kontroler.Dodaj_kwadrat();
     }
     repaint();
  }
```

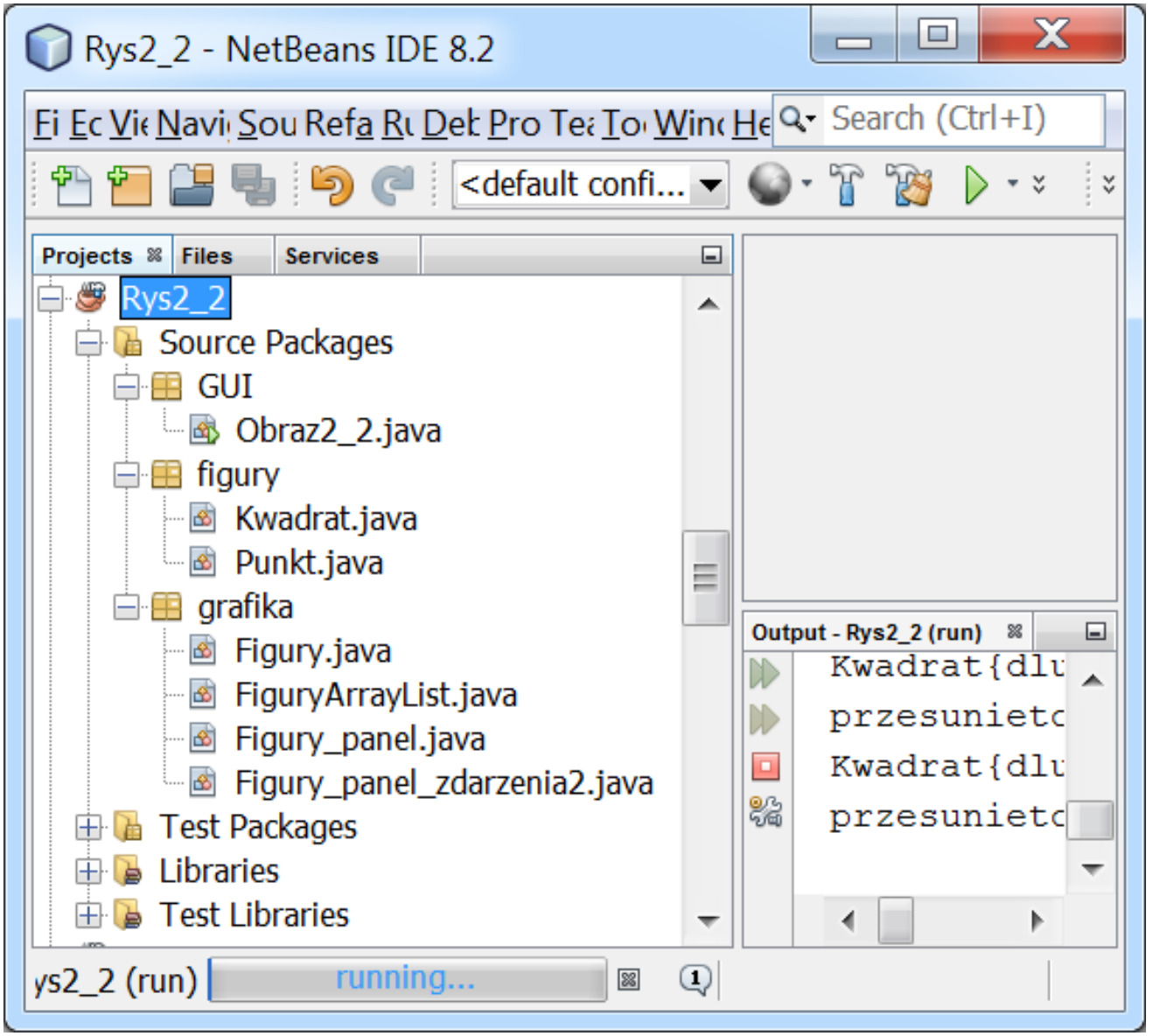
(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
@Override
   public void init() {
      super.init();
      setLayout(new FlowLayout());
      addKeyListener(this);
      setFocusable(true);
      addMouseListener(this);
      punkt.addActionListener(this);
      kwadrat.addActionListener(this);
      add(punkt);
      add(kwadrat);
   }
}
```

(cd) Przykład 1 obsługi zdarzeń – projekt Rys2_1

```
package GUI;

import grafika.FiguryArrayList;
import grafika.Figury_panel_zdarzenia1;
import javax.swing.JFrame;

public class Obraz2_1 {
 void rysunek_Swing() {
   JFrame ramka = new JFrame();
   Figury_panel_zdarzenia1 panel = new Figury_panel_zdarzenia1();
   panel.init();
   ramka.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
   ramka.setSize(400, 400);
   ramka.setContentPane(panel);
   ramka.setVisible(true); }

 public static void main(String args[]) {
     Obraz2_1 obraz = new Obraz2_1();
     java.awt.EventQueue.invokeLater(new Runnable() {
       @Override
        public void run() {
           obraz.rysunek_Swing();   }
     });
  } }
```

Przykład 2 obsługi zdarzeń – projekt Rys2_2



(cd) Przykład 2 obsługi zdarzeń – projekt Rys2_2

```
package grafika;

import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.Random;
import javax.swing.JButton;

public class Figury_panel_zdarzenia2
    extends Figury_panel implements ActionListener{

protected JButton punkt = new JButton("Punkt");
protected JButton kwadrat = new JButton("Kwadrat");

 public void mouseClicked_(MouseEvent e) {
      int x_ = e.getX();
      int y_ = e.getY();
      kontroler.Clicked(x_, y_);
      requestFocus();
      repaint(); }
```

```
public void keyPressed_(KeyEvent evt)
{
  int dx = 0, dy = 0, zmiana;
  Random r = new Random();
  if (evt.isShiftDown()) {
        zmiana = r.nextInt(100);
 } else {
    zmiana = r.nextInt(10);  }
  switch (evt.getKeyCode()) {
    case KeyEvent.VK_LEFT:
       dx -= zmiana;      break;
    case KeyEvent.VK_RIGHT:
       dx += zmiana;      break;
    case KeyEvent.VK_UP:
       dy -= zmiana;     break;
    case KeyEvent.VK_DOWN:
       dy += zmiana;     break;
     }
 kontroler.przesun(dx, dy,
          getWidth(), getHeight());
 repaint(); }
```

(cd) Przykład 2 obsługi zdarzeń – projekt Rys2_2

```
@Override
public void init() {
  super.init();
  setLayout(new FlowLayout());
  addKeyListener(new KeyAdapter() {
   @Override
   public void keyPressed(KeyEvent evt) {
         keyPressed_(evt); }
   });
  setFocusable(true);
  addMouseListener(new MouseAdapter() {
   @Override
     public void mouseClicked(MouseEvent e) {
          mouseClicked_(e); }
  });
  punkt.addActionListener(this);
  kwadrat.addActionListener(this);
  add(punkt);
  add(kwadrat);
}
```

```
@Override
 public void actionPerformed(ActionEvent e) {
      Object zrodlo = e.getSource();
      if (zrodlo == punkt) {
         kontroler.Dodaj_punkt();
      } else if (zrodlo == kwadrat) {
         kontroler.Dodaj_kwadrat();
      }
      repaint();
   }
}
```

(cd) Przykład 2 obsługi zdarzeń – projekt Rys2_2

```
package GUI;

import grafika.FiguryArrayList;
import grafika.Figury_panel_zdarzenia2;
import javax.swing.JFrame;

public class Obraz2_2 {
  void rysunek_Swing() {
    JFrame ramka = new JFrame();
    Figury_panel_zdarzenia2 panel = new Figury_panel_zdarzenia2();
    panel.init();
    ramka.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    ramka.setSize(400, 400);
    ramka.setContentPane(panel);
    ramka.setVisible(true); }

  public static void main(String args[]) {
      Obraz2_2 obraz = new Obraz2_2();
      java.awt.EventQueue.invokeLater(new Runnable() {
        @Override
         public void run() {
            obraz.rysunek_Swing();   }
      });
   } }
```