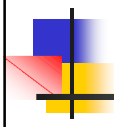


Programowanie w języku Java

Wykład 10: JFC/Swing – logika aplikacji

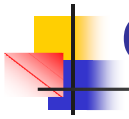
Obsługa zdarzeń

- Interfejs ActionListener

```
public class Beeper ... implements ActionListener {
  ... //inicjalizacja
   button.addActionListener(this);
  ... public void actionPerformed(ActionEvent e) {
   ...// Akcja
  }
}
```

- klasa ActionEvent

  String getActionCommand()
  int getModifiers()
  Object getSource()



Obsługa klawiatury

Interfejs	Klasa implementująca	Metody
KeyListener	KeyAdapter	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)

np. element.addKeyListener(new KeyAdapter());
element.removeKeyListener(new KeyAdapter());

Obsługa myszki

Interfejs	Klasa implementująca	Metody
MouseListener	MouseAdapter	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)
MouseMotionListener	MouseMotionAdapter	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
MouseInputListener	MouseInputAdapter	-II-

np. element.addMouseListener(new MouseAdapter());
element.removeMouseListener(new MouseAdapter());

Inne zdarzenia

Interfejs	Klasa implementująca	Metody
ComponentListener	ComponentAdapter	componentHidden(ComponentEvent) componentMoved(ComponentEvent) componentResized(ComponentEvent) componentShown(ComponentEvent)
FocusListener	FocusAdapter	focusGained(FocusEvent) focusLost(FocusEvent)
WindowListener	WindowAdapter	windowActivated(WindowEvent) windowClosed(WindowEvent) windowClosing(WindowEvent) windowDeactivated(WindowEvent) windowDeiconified(WindowEvent) windowIconified(WindowEvent) windowOpened(WindowEvent)

Przykład (1)

```
import javax.swing.*;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.Insets;
import java.awt.Dimension;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class MultiListener extends JPanel
                implements ActionListener {
   JTextArea topTextArea;
   JTextArea bottomTextArea;
   JButton button1, button2;
   final static String newline = "\n";

public MultiListener() {
      super(new GridBagLayout());
      GridBagLayout gridbag = (GridBagLayout)getLayout();
      GridBagConstraints c = new GridBagConstraints();

      JLabel l = null;
      c.fill = GridBagConstraints.BOTH;
      c.gridwidth = GridBagConstraints.REMAINDER;
      l = new JLabel("What MultiListener hears:");
      gridbag.setConstraints(l, c);
      add(l);
```

```
      c.weighty = 1.0;
      topTextArea = new JTextArea();
      topTextArea.setEditable(false);
      JScrollPane topScrollPane = new JScrollPane(topTextArea);
      Dimension preferredSize = new Dimension(200, 75);
      topScrollPane.setPreferredSize(preferredSize);
      gridbag.setConstraints(topScrollPane, c);
      add(topScrollPane);
      c.weightx = 0.0;
      c.weighty = 0.0;
      l = new JLabel("What Eavesdropper hears:");
      gridbag.setConstraints(l, c);
      add(l);
      c.weighty = 1.0;
      bottomTextArea = new JTextArea();
      bottomTextArea.setEditable(false);
      JScrollPane bottomScrollPane = new
            JScrollPane(bottomTextArea);
      bottomScrollPane.setPreferredSize(preferredSize);
      gridbag.setConstraints(bottomScrollPane, c);
      add(bottomScrollPane);
      c.weightx = 1.0;
      c.weighty = 0.0;
      c.gridwidth = 1;
      c.insets = new Insets(10, 10, 0, 10);
      button1 = new JButton("Blah blah blah");
      gridbag.setConstraints(button1, c);
      add(button1);
```

Przykład (2)

```
 c.gridwidth = GridBagConstraints.REMAINDER;
 button2 = new JButton("You don't say!");
 gridbag.setConstraints(button2, c);
 add(button2);

 button1.addActionListener(this);
 button2.addActionListener(this);
 button2.addActionListener(new
     Eavesdropper(bottomTextArea));

 setPreferredSize(new Dimension(450, 450));
 setBorder(BorderFactory.createCompoundBorder(
                 BorderFactory.createMatteBorder(
                         1,1,2,2,Color.black),
                 BorderFactory.createEmptyBorder(5,5,5,5)));
  }

 public void actionPerformed(ActionEvent e) {
      topTextArea.append(e.getActionCommand() + newline);
topTextArea.setCaretPosition(topTextArea.getDocument().
 getLength());
  }
private static void createAndShowGUI() {
      JFrame frame = new JFrame("MultiListener");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JComponent newContentPane = new MultiListener();
 newContentPane.setOpaque(true);
 frame.setContentPane(newContentPane);
      frame.pack();
      frame.setVisible(true);
   }
public static void main(String[] args) {
      //Schedule a job for the event-dispatching thread:
      //creating and showing this application's GUI.
      javax.swing.SwingUtilities.invokeLater(new Runnable() {
         public void run() {
            createAndShowGUI();
         }
      });
   }
}
class Eavesdropper implements ActionListener {
   JTextArea myTextArea;
   public Eavesdropper(JTextArea ta) {
      myTextArea = ta;
   }

   public void actionPerformed(ActionEvent e) {
      myTextArea.append(e.getActionCommand()
                  + MultiListener.newline);
    myTextArea.setCaretPosition(myTextArea.getDocument().
    getLength());
   }
}
```


Grafika użytkownika

Klasa JComponent

↓ Graphics2D

```
protected void paintComponent(Graphics g) {
 ...
   Insets insets = getInsets();
   int currentWidth = getWidth() - insets.left - insets.right;
   int currentHeight = getHeight() - insets.top -
    insets.bottom;
   ... ...
   Rysowanie: g.drawLine(),g.drawRect(), g.drawArc(),
              g.drawString(), g.fillRect(), g.setColor(), itp..
}
```

Tworzenie grafiki w Java2D (1)

```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import javax.swing.*;
public class ShapesDemo2D extends JApplet {
   final static int maxCharHeight = 15;
   final static int minFontSize = 6;

   final static Color bg = Color.white;
   final static Color fg = Color.black;
   final static Color red = Color.red;
   final static Color white = Color.white;

   final static BasicStroke stroke = new BasicStroke(2.0f);
   final static BasicStroke wideStroke = new
                BasicStroke(8.0f);

   final static float dash1[] = {10.0f};
   final static BasicStroke dashed = new BasicStroke(1.0f,
                                BasicStroke.CAP_BUTT,
                                BasicStroke.JOIN_MITER,
                                10.0f, dash1, 0.0f);
   Dimension totalSize;
   FontMetrics fontMetrics;
```

```
public void init() {
      setBackground(bg);
      setForeground(fg);
   }
 FontMetrics pickFont(Graphics2D g2, String longString,
                int xSpace) {
      boolean fontFits = false;
      Font font = g2.getFont();
      FontMetrics fontMetrics = g2.getFontMetrics();
      int size = font.getSize();
      String name = font.getName();
      int style = font.getStyle();
      while ( !fontFits ) {
         if ( (fontMetrics.getHeight() <= maxCharHeight)
            && (fontMetrics.stringWidth(longString) <=
                              xSpace) ) {
            fontFits = true;
         }
         else {
            if ( size <= minFontSize ) {
               fontFits = true;
            }
            else {
               g2.setFont(font = new Font(name, style,--size));
               fontMetrics = g2.getFontMetrics();
            }
         }
      }
      return fontMetrics;
   }
```


Tworzenie grafiki w Java2D (2)

```
public void paint(Graphics g) {
      Graphics2D g2 = (Graphics2D) g;
      g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                          RenderingHints.VALUE_ANTIALIAS_ON);
      Dimension d = getSize();
      int gridWidth = d.width / 6;
      int gridHeight = d.height / 2;

      fontMetrics = pickFont(g2, "Filled and Stroked GeneralPath",
                     gridWidth);

      Color fg3D = Color.lightGray;

      g2.setPaint(fg3D);
      g2.draw3DRect(0, 0, d.width - 1, d.height - 1, true);
      g2.draw3DRect(3, 3, d.width - 7, d.height - 7, false);
      g2.setPaint(fg);

      int x = 5;
      int y = 7;
      int rectWidth = gridWidth - 2*x;
      int stringY = gridHeight - 3 - fontMetrics.getDescent();
      int rectHeight = stringY - fontMetrics.getMaxAscent() - y - 2;

      g2.draw(new Line2D.Double(x, y+rectHeight-1, x +
                     rectWidth, y));
      g2.drawString("Line2D", x, stringY);
      x += gridWidth;
```

```
      g2.setStroke(stroke);
      g2.draw(new Rectangle2D.Double(x, y, rectWidth,
                     rectHeight));
      g2.drawString("Rectangle2D", x, stringY);
      x += gridWidth;

      // draw  RoundRectangle2D.Double
      g2.setStroke(dashed);
      g2.draw(new RoundRectangle2D.Double(x, y, rectWidth,
                           rectHeight, 10, 10));
      g2.drawString("RoundRectangle2D", x, stringY);
      x += gridWidth;

      // draw Arc2D.Double
      g2.setStroke(wideStroke);
      g2.draw(new Arc2D.Double(x, y, rectWidth, rectHeight,
                     90, 135, Arc2D.OPEN));
      g2.drawString("Arc2D", x, stringY);
      x += gridWidth;

      // draw Ellipse2D.Double
      g2.setStroke(stroke);
      g2.draw(new Ellipse2D.Double(x, y, rectWidth,
                     rectHeight));
      g2.drawString("Ellipse2D", x, stringY);
      x += gridWidth;
```

Tworzenie grafiki w Java2D (3)

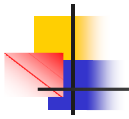
```
// draw GeneralPath (polygon)
  int x1Points[] = {x, x+rectWidth, x, x+rectWidth};
  int y1Points[] = {y, y+rectHeight, y+rectHeight, y};
  GeneralPath polygon = new GeneralPath(GeneralPath.WIND_EVEN_ODD,
                        x1Points.length);
  polygon.moveTo(x1Points[0], y1Points[0]);
  for ( int index = 1; index < x1Points.length; index++ ) {
        polygon.lineTo(x1Points[index], y1Points[index]);
  };
  polygon.closePath();
  g2.draw(polygon);
  g2.drawString("GeneralPath", x, stringY);

  // NEW ROW
  x = 5;
  y += gridHeight;
  stringY += gridHeight;

  // draw GeneralPath (polyline)
  int x2Points[] = {x, x+rectWidth, x, x+rectWidth};
  int y2Points[] = {y, y+rectHeight, y+rectHeight, y};
  GeneralPath polyline = new GeneralPath(GeneralPath.WIND_EVEN_ODD,
                         x2Points.length);
  polyline.moveTo (x2Points[0], y2Points[0]);
  for ( int index = 1; index < x2Points.length; index++ ) {
        polyline.lineTo(x2Points[index], y2Points[index]);
  };
  g2.draw(polyline);
  g2.drawString("GeneralPath (open)", x, stringY);
  x += gridWidth;
```

```
  // fill Rectangle2D.Double (red)
  g2.setPaint(red);
  g2.fill(new Rectangle2D.Double(x, y, rectWidth,
          rectHeight));
  g2.setPaint(fg);
  g2.drawString("Filled Rectangle2D", x, stringY);
  x += gridWidth;

  // fill RoundRectangle2D.Double
  GradientPaint redtowhite = new
      GradientPaint(x,y,red,x+rectWidth, y,white);
  g2.setPaint(redtowhite);
  g2.fill(new RoundRectangle2D.Double(x, y,
          rectWidth, rectHeight, 10, 10));
  g2.setPaint(fg);
  g2.drawString("Filled RoundRectangle2D", x,
                stringY);
  x += gridWidth;

  // fill Arc2D
  g2.setPaint(red);
  g2.fill(new Arc2D.Double(x, y, rectWidth,
          rectHeight, 90, 135, Arc2D.OPEN));
  g2.setPaint(fg);
  g2.drawString("Filled Arc2D", x, stringY);
  x += gridWidth;
```


Tworzenie grafiki w Java2D (4)

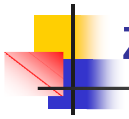
```
// fill Ellipse2D.Double
    redtowhite = new GradientPaint(x,y,red,x+rectWidth,
        y,white);
    g2.setPaint(redtowhite);
    g2.fill (new Ellipse2D.Double(x, y, rectWidth, rectHeight));
    g2.setPaint(fg);
    g2.drawString("Filled Ellipse2D", x, stringY);
    x += gridWidth;

    // fill and stroke GeneralPath
    int x3Points[] = {x, x+rectWidth, x, x+rectWidth};
    int y3Points[] = {y, y+rectHeight, y+rectHeight, y};
    GeneralPath filledPolygon = new
        GeneralPath(GeneralPath.WIND_EVEN_ODD,
                    x3Points.length);
    filledPolygon.moveTo(x3Points[0], y3Points[0]);
    for ( int index = 1; index < x3Points.length; index++ ) {
      filledPolygon.lineTo(x3Points[index], y3Points[index]);
    };
    filledPolygon.closePath();
    g2.setPaint(red);
    g2.fill(filledPolygon);
    g2.setPaint(fg);
    g2.draw(filledPolygon);
    g2.drawString("Filled and Stroked GeneralPath", x, stringY);
  }
```

```
  public static void main(String s[]) {
    JFrame f = new JFrame("ShapesDemo2D");
    f.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e)
        {System.exit(0);}
    });
    JApplet applet = new ShapesDemo2D();
    f.getContentPane().add("Center", applet);
    applet.init();
    f.pack();
    f.setSize(new Dimension(550,100));
    f.setVisible(true);
  }
}
```



Interakcja z użytkownikiem

```
public void mousePressed(MouseEvent e){
  last_x = rect.x - e.getX();
  last_y = rect.y - e.getY();
  if(rect.contains(e.getX(), e.getY()))
          updateLocation(e);
   ...

public void updateLocation(MouseEvent e){
  rect.setLocation(last_x + e.getX(), last_y + e.getY());
   ...
  repaint();
```

Programowanie sterowane zdarzeniami

- Event-driven programming
- Wykonywanie akcji jako odpowiedzi na zdarzenia (obsługa zdarzeń)
- Kolejność zdarzeń jest nieznana
- Obsługa zdarzeń powinna być jak najkrótsza:
  - Wielowątkowość
  - Asynchroniczne I/O
- Zastosowania:
  - GUI
  - Systemy wielozadaniowe
  - Serwery

Koniec