SWING. Jlabel, Jbutton, JTextField

Swing komponentų privalumai lyginant su AWT

Parašyti išimtinai Java kodu ir nepriklauso nuo platformos

Žymės ir mygtukai be įprasto teksto gali turėti paveikslėlius

Daugeliui komponentų galima uždėti ar pakeisti rėmelius

Komponentai nebūtinai yra stačiakampės formos (pvz., apvalus mygtukas)

Lengvai keičiamas komponentų elgesys ar išvaizda, panaudojant esamus metodus ar juos perdengus

Galimos problemos

Reikia vengti mišraus AWT ir Swing komponentų vartojimo

Swing komponentus reikėtų dėti tik į Swing viršutinio lygio konteinerius (JFrame, JApplet)

Pagrindiniai Swing paketai

import javax.swing.*;

import javax.swing.event.*;

AWT paketai:

import java.awt.*;

import java.awt.event.*;

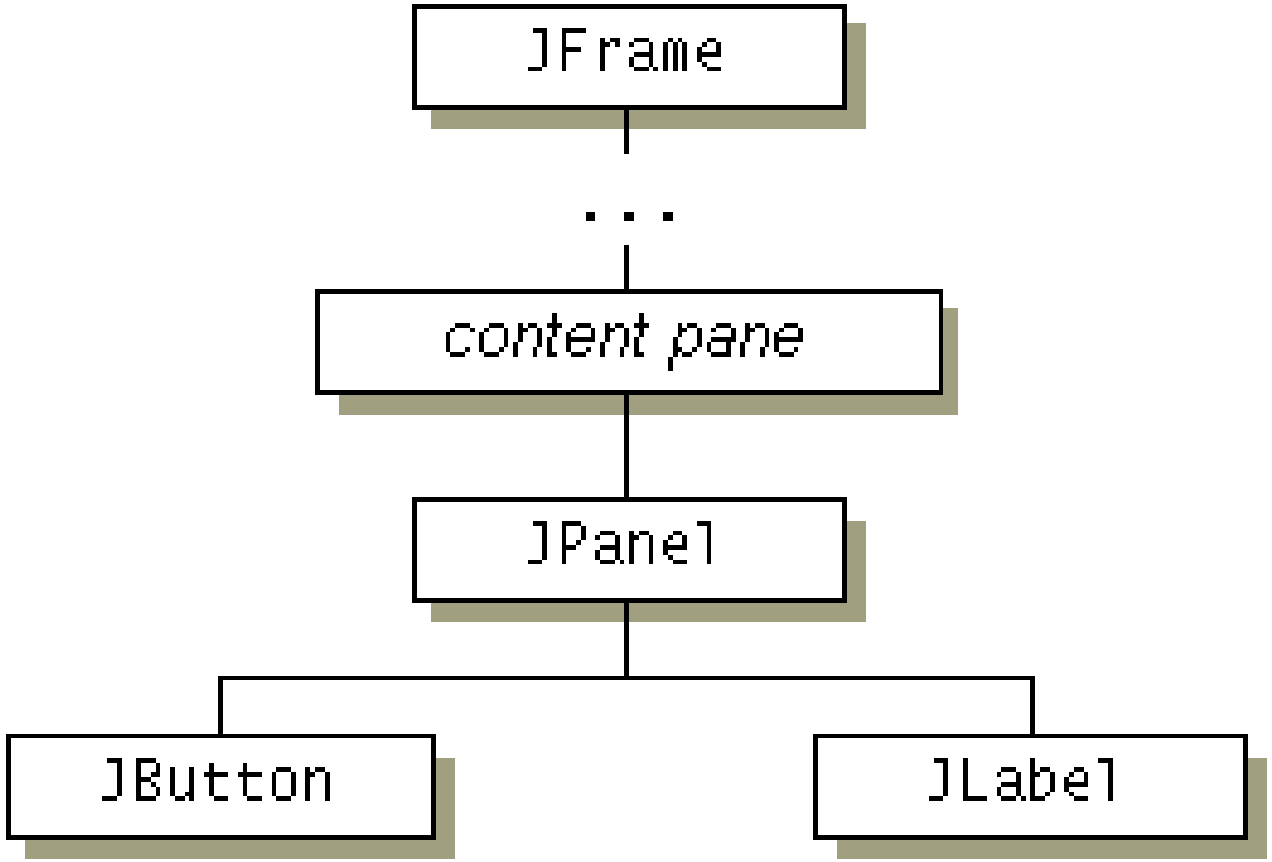
Swing’o komponentai ir jų turinio hierarchija

Viršutinio lygio konteineriai (JFrame, JApplet, JDialog)

Tarpiniai konteineriai (JPanel, JScrollPane, JTabbedPane)

Atominiai (atomic) komponentai, priimantys informaciją ar išvedantys ją vartotojui

Swing’o komponentai ir jų turinio hierarchija

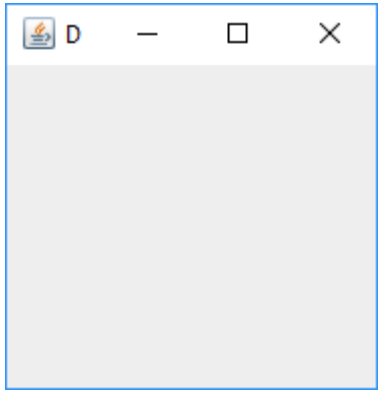


http://java.sun.com/docs/books/tutorial/uiswing/components/toplevel.html

Minimalistinė programa

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

        JFrame frame = new JFrame("Demo");        //Jframe objekto sukūrimas
        //Nustatymas, kad uždarius Frame užsidarytų ir programa
        frame.addWindowListener( new WindowAdapter() {
                public void windowClosing(WindowEvent evt) {System.exit(0);}
        });

        frame.setSize(200,200);                   //Ekrano dydžio nustatymas
        frame.setLocationRelativeTo(null);        //Nustatymas rodyti ekrano centre
        frame.setVisible(true);
    }
}
```

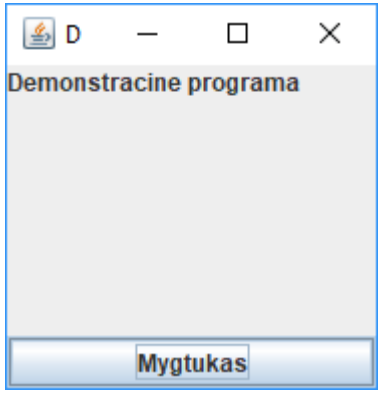
Minimalistinė programa



Elementų pridėjimas į formas

```
//Antraštės pirdėjimas
JLabel label = new JLabel("Demonstracine programa");


//Mygtuko pirdėjimas
JButton button = new JButton("Mygtukas");
//Elementų pridėjimas į Jframe panelę


frame.add(label, BorderLayout.PAGE_START);
frame.add(button, BorderLayout.PAGE_END);
```

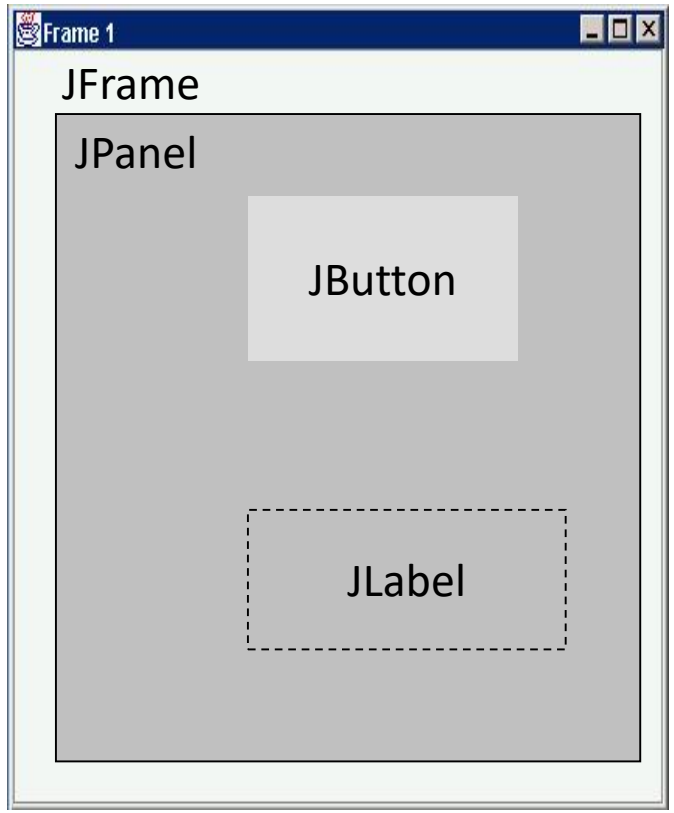
Programos pavyzdys

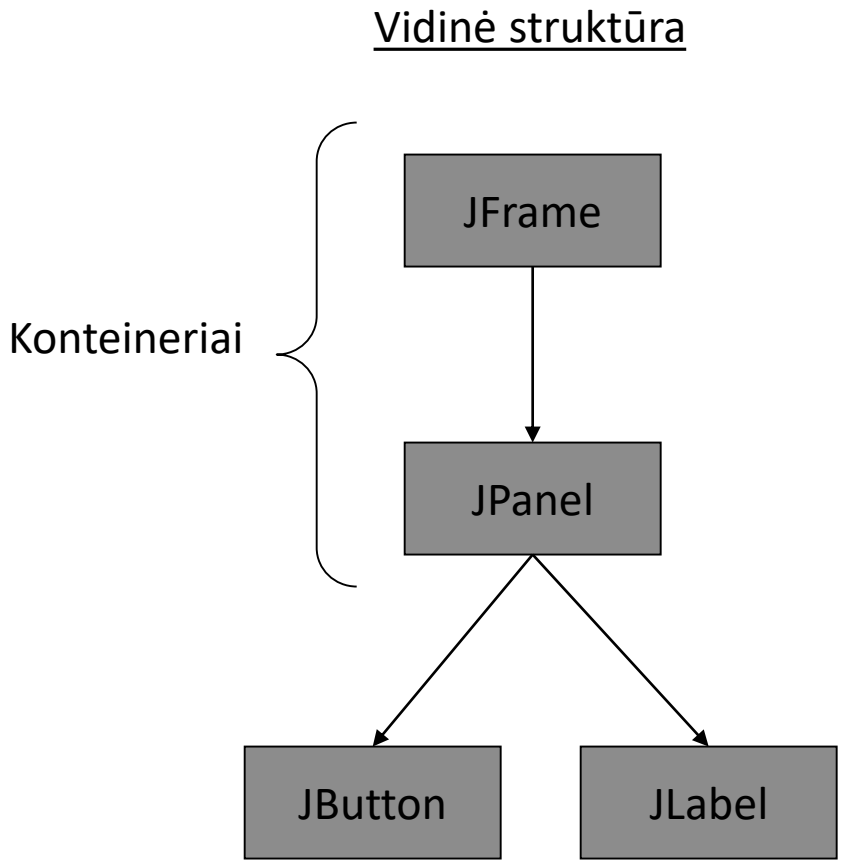


GUI komponentų kūrimo žingsniai

1. GUI elemento sukūrimas
   - Sukurti objektą : b = new JButton(“press me”);
2. Sukonfigūruoti elementą
   - Per atributus: b.text = “press me”; [avoided in java]
   - Metodų pagalba: b.setText(“press me”);
3. Pridėti jį prie konteinerio
   - panel.add(b);
4. Nurodyti įvykių klausimąsį
   - Events: Listeners

JAVA GUI aplikacijos struktūra





Jframe dydžio nustatymas

Dydžio nustatymas nurodant ilgį ir plotį:

```
frame.setSize(200,200);
```

Dydžio nustatymas nurodant X, Y koordinates, taip pat ilgį ir plotį:

```
frame.setBounds(10, 100, 200, 200);
```

Leisti dydį parinkti automatiškai pagal komponentus:

```
frame.pack();
```

Jframe ir kodo vykdymas

Viena aplikacija gali turėti neribotą skaičių Jframe.

Visi langai tiek bus vykdomos vienoje gijoje (jei jų neišskaidysime patys).

Vartotojo sąsajos komponentų išdėstymo būdai

Išdėstymo būdas gali būti nustatytas bet kuriam viršutinio lygio konteineriui

(pvz., *JFrame, JDialog, JApplet, JPanel ir pan.)*

Išdėstymui nustatyti naudojamas *Container* klasės metodas:

```
public void setLayout(LayoutManager isdestymoBudas)
```

Pagrindinės išdėstymo klasės:

BorderLayout

BoxLayout

FlowLayout

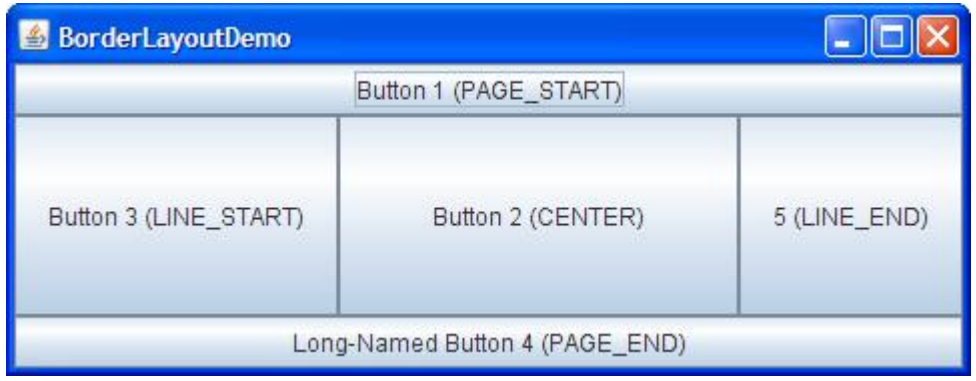
GridLayout

BorderLayout

Every content pane is initialized to use a BorderLayout. (As Using Top-Level Containers explains, the content pane is the main container in all frames, applets, and dialogs.)

A BorderLayout places components in up to five areas: top, bottom, left, right, and center.

All extra space is placed in the center area.



http://java.sun.com/docs/books/tutorial/uiswing/layout/border.html

BorderLayout pavyzdys

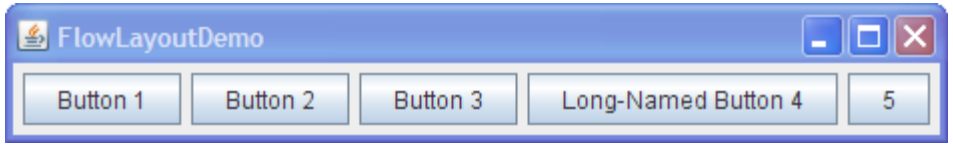
```
JPanel pane = new JPanel(new BorderLayout());
JButton button = new JButton("(PAGE_START)");
pane.add(button, BorderLayout.PAGE_START);

//Make the center component big, since that's the
//typical usage of BorderLayout.
button = new JButton("Button 2 (CENTER)");
button.setPreferredSize(new Dimension(200, 100));
pane.add(button, BorderLayout.CENTER);
```

```
button = new JButton("Button 3 (LINE_START)");
pane.add(button, BorderLayout.LINE_START);

button = new JButton("Button 4 (PAGE_END)");
pane.add(button, BorderLayout.PAGE_END);

button = new JButton("5 (LINE_END)");
pane.add(button, BorderLayout.LINE_END);
```

FlowLayout

FlowLayout is the default layout manager for every JPanel.

It simply lays out components in a single row, starting a new row if its container is not sufficiently wide.



http://java.sun.com/docs/books/tutorial/uiswing/layout/box.html

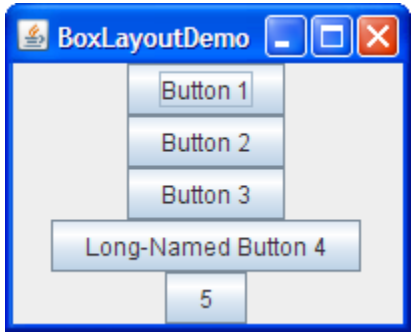
FlowLayout pavyzdys

```
JPanel pane = new JPanel(new FlowLayout());
pane.add(new JButton("Button 1"));
pane.add(new JButton("Button 2"));
pane.add(new JButton("Button 3"));
pane.add(new JButton("Long-Named Button 4"));
pane.add(new JButton("5"));
```

BoxLayout

The BoxLayout class puts components in a single row or column.

It respects the components' requested maximum sizes and also lets you align components.



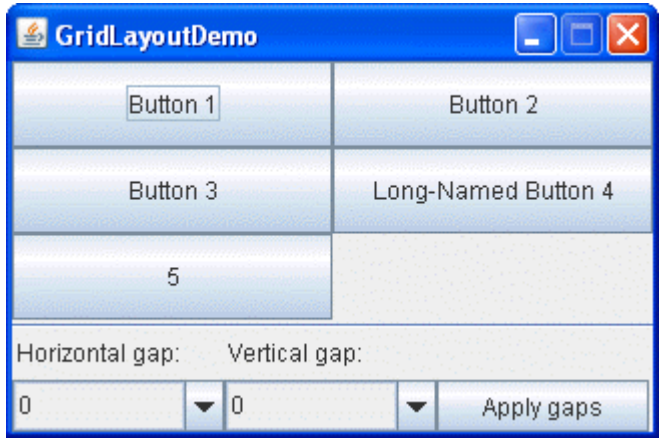
http://java.sun.com/docs/books/tutorial/uiswing/layout/flow.html

BoxLayout pavyzdys

```
JPanel pane = new JPanel();
pane.setLayout(new BoxLayout(pane, BoxLayout.X_AXIS));

pane.add(new JButton("Button 1") );
pane.add(Box.createRigidArea(new Dimension(5,0)));

pane.add(new JButton("Button 2"));
pane.add(Box.createHorizontalGlue());
pane.add(new JButton("Button 3"));
```

GridLayout

GridLayout simply makes a bunch of components equal in size and displays them in the requested number of rows and columns.



http://java.sun.com/docs/books/tutorial/uiswing/layout/grid.html

GridLayout pavyzdys

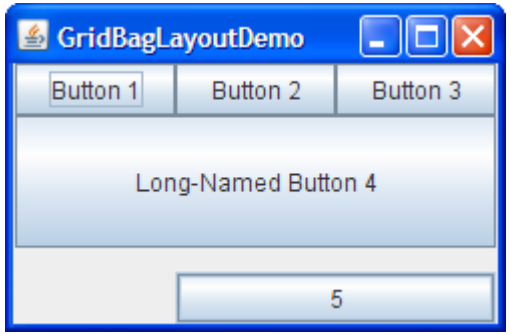
```
JPanel pane = new JPanel(new GridLayout(3,2));
pane.add(new JButton("Button 1"));
pane.add(new JButton("Button 2"));
pane.add(new JButton("Button 3"));
pane.add(new JButton("Long-Named Button 4"));
pane.add(new JButton("5"));
```

GridBagLayout

GridBagLayout is a sophisticated, flexible layout manager.

It aligns components by placing them within a grid of cells, allowing components to span more than one cell.

The rows in the grid can have different heights, and grid columns can have different widths.



http://java.sun.com/docs/books/tutorial/uiswing/layout/gridbag.html

GridBagLayout pavyzdys

```
JPanel pane = new JPanel(new GridBagLayout());
pane.setLayout(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();
JButton button = new JButton("Button 1");
c.fill = GridBagConstraints.HORIZONTAL;
c.gridx = 0;
c.gridy = 0;
pane.add(button, c);

button = new JButton("Button 2");
c.fill = GridBagConstraints.HORIZONTAL;
c.weightx = 0.5;
c.gridx = 1;
c.gridy = 0;
pane.add(button, c);

button = new JButton("Button 3");
c.fill = GridBagConstraints.HORIZONTAL;
c.weightx = 0.5;
c.gridx = 2;
c.gridy = 0;
pane.add(button, c);
```

```
button = new JButton("Long-Named Button 4");
c.fill = GridBagConstraints.HORIZONTAL;
c.ipady = 40;      //make this component tall
c.weightx = 0.0;
c.gridwidth = 2;
c.gridx = 0;
c.gridy = 1;
pane.add(button, c);

button = new JButton("5");
c.fill = GridBagConstraints.HORIZONTAL;
c.ipady = 0;       //reset to default
c.weighty = 1.0;   //request any extra vertical space
c.anchor = GridBagConstraints.PAGE_END; //bottom of space
c.insets = new Insets(10,0,0,0);  //top padding
c.gridx = 1;       //aligned with button 2
c.gridwidth = 2;   //2 columns wide
c.gridy = 2;       //third row
pane.add(button, c);
```

GUI kūrimo aplinkoms skirti layout‘ai

GUI kūrimo aplinkoms skirti layout‘ai:

- GroupLayout
- SpringLayout

JLabel komponentas

Labels

- Provide text instructions on a GUI
- Read-only text
- Programs rarely change a label's contents
- Class JLabel (subclass of JComponent)

Metodai

- Can declare label text in constructor
  - myLabel.setToolTipText("Text")
- Displays "Text" in a tool tip when mouse over label
  - myLabel.setText("Text")
  - myLabel.getText()

JLabel komponentas

`Icon`

- Object that implements interface **`Icon`**
  - Icon bug = new ImageIcon("bug1.gif");

- One class is **`ImageIcon`** (**`.gif`** and **`.jpeg`** images)
  - Assumed same directory as program

- Display an icon with **`JLabel`**'s **`setIcon`** method
  - **`myLabel.setIcon( myIcon );`**
  - **`myLabel.getIcon(); //returns current Icon`**

JButton

Methods of class **JButton**

- Constructors

```
JButton myButton = new JButton( "Label" );
JButton myButton = new JButton( "Label", myIcon );
```

- Sets image to display when mouse over button
  - **setRolloverIcon(myIcon)**

Class **ActionEvent getActionCommand**

- returns label of button that generated event

```
Icon bug1 = new ImageIcon( "bug1.gif" );

fancyButton = new JButton( "Fancy Button", bug1 );

fancyButton.setRolloverIcon( bug2 );
```

JTextField

Konstruktoriai:

JTextField field=**new JTextField(20);**

// įvedimo lauko ilgis 20 simbolių

JTextField field=**new JTextField("Labas");**

// įvedimo laukas įgauna tokią reikšmę, jo ilgis tampa lygus teksto ilgiui

JTextField field=**new JTextField("Labas",20);**

// įvedimo laukas įgauna tokią reikšmę, jo ilgis lygus 20 simbolių

Įvykių apdorojimas

Įvykis – tai veiksmas, kurį atlieka vartotojas, dirbdamas su grafine vartotojo sąsaja

Kiekvieną įvykį aprašo jam skirtas objektas (pvz., ActionEvent, MouseEvent)

Bet kuris Swing komponentas (šaltinis) gali fiksuoti su juo susijusius įvykius

Bet kuris Swing komponentas gali fiksuoti kelis skirtingo tipo įvykius

Bet kuris “įvykio klausytojas” (event listener) gali būti užregistruotas keliems šaltiniams

Kai kurių įvykių pavyzdžiai

ActionEvent – paspaudžiamas mygtukas, Enter klavišas ar pasirenkamas meniu punktas

WindowEvent – vartotojas uždaro programos langą

MouseEvent – vartotojas paspaudžia kurį nors pelės klavišą ant komponento

MouseMotionEvent – vartotojas užveda kursorių virš komponento

Įvykį apdorojanti klasė

Įvykį apdorojanti klasė privalo:

- implementuoti “įvykio klausytojo” interfeisą ir realizuoti visus jo metodus
- paveldėti “klausytojo” interfeisą realizuojančią klasę ir perdengti reikalingus metodus

Įvykio apdorojimo realizavimas

Įvykių apdorojimo interfeiso implementavimas

```
public class MyListener implements ActionListener {
  public void actionPerformed(ActionEvent evt) {
        // atitinkami veiksmai
  }
}
```

Klausytojo klasės paveldėjimas

```
class MyMouseListener extends MouseAdapter {
  public void mouseClicked(MouseEvent evt) {
        // atitinkami veiksmai
  }
}
```

Įvykio apdorojimo realizavimas

Naudojant *addXXXListener()* metodą, reikiamiems komponentams, užregistruojama įvykį apdorojanti klasė :

- `komponentas.addActionListener(new MyListener());`
- `komponentas.addMouseListener(new MyMouseListener());`

Anoniminės “įvykio klausytojo” klasės panaudojimas:

```
komponentas.addActionListener( new ActionListener() {
 public void actionPerformed(ActionEvent evt) {
    // atitinkami veiksmai
 }
});
```

Pastabos dėl įvykių apdorojimo

Visi *xxxListener* interfeisai, turintys daugiau negu vieną metodą, turi specialias juos realizuojančias adapterių klases

Įvykių klasės *xxxEvent* objektas turi metodus papildomai informacijai apie įvykį gauti

Visos *xxxEvent* klasės paveldi *ObjectEvent* klasę, kurios metodas `getSource()` gražina įvykio šaltinį