

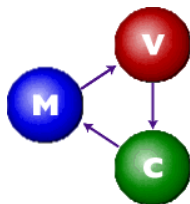
Programowanie i projektowanie obiektowe GUI

Paweł Daniluk

Wydział Fizyki

Jesień 2011

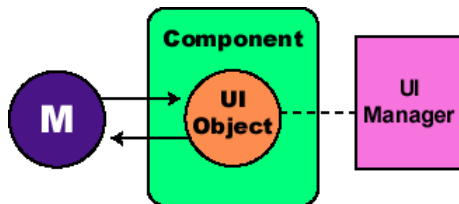




Model-View-Controller

- Model – Dane
- Widok – Graficzna reprezentacja
- Kontroler – Aktualizuje model na podstawie działań użytkownika

Swing



Swing

Jedna z bibliotek interfejsu graficznego w Javie. Uproszczona wersja modelu MVC.

Hello World

```
import javax.swing.*;

public class HelloWorldSwing {
    /**
     * Create the GUI and show it. For thread safety,
     * this method should be invoked from the
     * event-dispatching thread.
     */
    private static void createAndShowGUI() {
        //Create and set up the window.
        JFrame frame = new JFrame("HelloWorldSwing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Add the ubiquitous "Hello World" label.
        JLabel label = new JLabel("Hello World");
        frame.getContentPane().add(label);

        //Display the window.
        frame.pack();
        frame.setVisible(true);
    }
}
```

Hello World c.d.

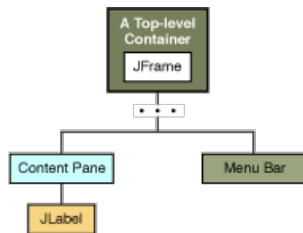
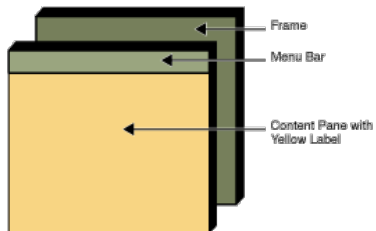
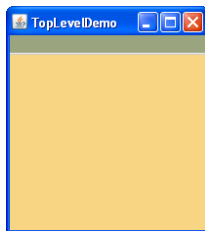
```
public static void main(String[] args) {
    //Schedule a job for the event-dispatching thread:
    //creating and showing this application's GUI.
    javax.swing.SwingUtilities.invokeLater(new Runnable() {

        public void run() {
            createAndShowGUI();
        }
    });
}
```

Hello World c.d.



Kontenery



Kontenery c.d.

```
private static void createAndShowGUI() {
    //Create and set up the window.
    JFrame frame = new JFrame("TopLevelDemo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

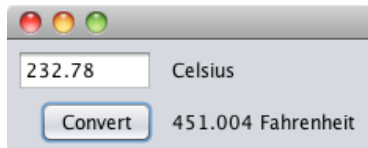
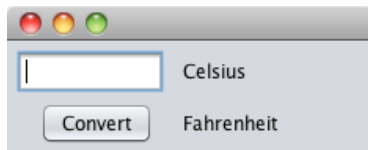
    //Create the menu bar. Make it have a green background.
    JMenuBar greenMenuBar = new JMenuBar();
    greenMenuBar.setOpaque(true);
    greenMenuBar.setBackground(new Color(154, 165, 127));
    greenMenuBar.setPreferredSize(new Dimension(200, 20));

    //Create a yellow label to put in the content pane.
    JLabel yellowLabel = new JLabel();
    yellowLabel.setOpaque(true);
    yellowLabel.setBackground(new Color(248, 213, 131));
    yellowLabel.setPreferredSize(new Dimension(200, 180));

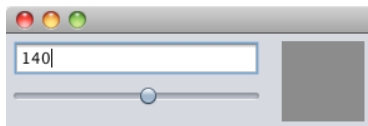
    //Set the menu bar and add the label to the content pane.
    frame.setJMenuBar(greenMenuBar);
    frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);

    //Display the window.
    frame.pack();
    frame.setVisible(true);
}
```


Celsius to Fahrenheit



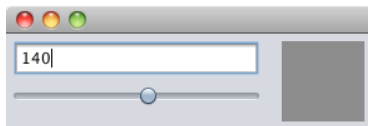
Modele i zdarzenia



Widok i kontroler

- 1 `javax.swing.JTextField numberTextField` – zmiana pola powoduje przesunięcie suwaka i zmianę koloru etykiety
- 2 `javax.swing.JSlider slider` – przesunięcie suwaka powoduje aktualizację pola i zmianę koloru etykiety
- 3 `javax.swing.JLabel colorLabel` – etykieta tylko zmienia kolor

Modele i zdarzenia



Widok i kontroler

- 1 `javax.swing.JTextField numberTextField` – zmiana pola powoduje przesunięcie suwaka i zmianę koloru etykiety
- 2 `javax.swing.JSlider slider` – przesunięcie suwaka powoduje aktualizację pola i zmianę koloru etykiety
- 3 `javax.swing.JLabel colorLabel` – etykieta tylko zmienia kolor

Model

`DefaultBoundedRangeModel model` – zmiana modelu powoduje aktualizację wszystkich komponentów, zmiana wartości w edytowalnym komponencie powoduje zmianę modelu.

Modele i zdarzenia c.d.

Model

```
DefaultBoundedRangeModel model = new DefaultBoundedRangeModel(0, 0, 0, 255);
```

Konstruktor okna (podklasy JFrame)

```
public EventExamples() {  
    initComponents();  
  
    slider.setModel(model);  
  
    model.addChangeListener(new ModelChangeListener());  
  
    numberTextField.getDocument().addDocumentListener(new TextFieldChangeListener()  
}
```

Modele i zdarzenia c.d.

Po aktualizacji modelu

```
private class ModelChangeListener implements ChangeListener {
    @Override
    public void stateChanged(ChangeEvent ce) {
        int val = model.getValue();

        if (!String.valueOf(val).equals(numberTextField.getText())) {
            numberTextField.setText(String.valueOf(model.getValue()));
        }

        colorLabel.setBackground(new Color(val, val, val));
        colorLabel.repaint();
    }
}
```

Modele i zdarzenia c.d.

Po aktualizacji pola tekstowego

```
private class TextFieldChangeListener implements DocumentListener {
    @Override
    public void insertUpdate(DocumentEvent de) {
        textFieldUpdated();
    }

    @Override
    public void removeUpdate(DocumentEvent de) {
        textFieldUpdated();
    }

    @Override
    public void changedUpdate(DocumentEvent de) {
        textFieldUpdated();
    }
}
```

Modele i zdarzenia c.d.

```
private void textFieldUpdated() {
    int val;

    try {
        val = new Integer(numberTextField.getText());
    } catch (NumberFormatException ex) {
        return;
    }

    if (val >= model.getMinimum() && val <= model.getMaximum()) {
        model.setValue(val);
    }
}
```

Skąd się biorą zdarzenia?

```
protected EventListenerList listenerList = new EventListenerList();

public void addChangeListener(ChangeListener l) {
    listenerList.add(ChangeListener.class, l);
}

public void removeChangeListener(ChangeListener l) {
    listenerList.remove(ChangeListener.class, l);
}

protected void fireStateChanged() {
    Object[] listeners = listenerList.getListenerList();
    for (int i = listeners.length - 2; i >= 0; i -= 2) {
        if (listeners[i] == ChangeListener.class) {
            if (changeEvent == null) {
                changeEvent = new ChangeEvent(this);
            }
            ((ChangeListener)listeners[i+1]).stateChanged(changeEvent);
        }
    }
}
```


Zadanie 1 – Kolory

Zadanie

Rozszerzyć przykład do trzech liczb określających kolor RGB.