

PROGRAMOWANIE W JĘZYKU JAVA

Prowadzący: dr hab. inż. **Jan Prokop**, prof. PRz, e-mail: jprokop@prz.edu.pl,
Politechnika Rzeszowska, Wydział Elektrotechniki i Informatyki

LABORATORIUM 3

Temat: Java AWT, SWING - Obsługa zdarzeń

Java 8 API – <https://docs.oracle.com/javase/8/docs/api/>

Zadania

W oparciu o przykład 1.1 i pozostałe przykłady z punktów 1 – 6 zbudować aplikację biblioteki Swing o nazwie **ImageViewer** i wyglądzie przedstawionym na poniższym rysunku:



- Sposób obsługi zdarzeń od **komponentów** (w klasie głównej, klasie wewnętrznej, anonimowej klasie wewnętrznej, klasie zewnętrznej) wybiera prowadzący dla każdej grupy laboratoryjnej
- Sposób obsługi zdarzeń od **myszki** (kliknięcie, naciśnięcie klawisza myszki, zwolnienie klawisza myszki, itp. z modyfikatorami typu Alt, Ctrl, itp. lub bez modyfikatorów) oraz **klawiatury** (sterowanie klawiszami bez i z modyfikatorami typu Alt, Ctrl, itp.) podaje prowadzący
- Wybór **komponentów dodatkowych** i szczegóły funkcjonalności wybiera prowadzący dla każdej grupy laboratoryjnej, w szczególności w oparciu o klasy: JComboBox, JTabbedPane, JFileChooser, JMenu, JToolBar, JSplitPane, JTree i inne
- Środkowy przycisk nawigacji powinien uruchamiać **automatyczny** pokaz zdjęć (dodać komponent wyboru czasu) korzystając z klasy Timer
- W aplikacji dodać funkcję **logowania**, np. do całej aplikacji lub po oglądnięciu 5 obrazków bez logowania
- Liczbę i zawartość poszczególnych zakładek podaje prowadzący, np. należy zmodyfikować kod tak aby na wybranej zakładce po załadowaniu obrazka za każdym razem był **odtwarzany** inny plik dźwiękowy lub plik video
- Zbudować powyższą aplikację w architekturze **MVC**
- Budowa aplikacji może być realizowana za pomocą narzędzi podstawowych lub w dowolnie wybranym środowisku IDE, np. NetBeans, IntelliJ IDEA, Eclipse, JDeveloper
- Inne szczegóły obsługi zdarzeń i funkcjonalności podaje prowadzący dla każdej grupy laboratoryjnej

1. Zdarzenia od komponentów

1.1. Obsługa zdarzeń w klasie głównej

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ImageViewer extends JFrame implements ActionListener {

    int currentImage = 0;
    String images[] = {"image1.jpg", "image2.jpg", "image3.jpg"};
    JLabel title;
    JLabel display;
    JButton buttonNext;
    JButton buttonBack;
    public ImageViewer() {
        super("Java Slide Show");
        setSize(800, 600);
        title = new JLabel("", JLabel.CENTER);
        add(title, BorderLayout.NORTH);
        display = new JLabel();
        //display.setBounds(0, 0, getWidth(), getHeight());
        showCurrentImage(0);
        add(display, BorderLayout.CENTER);
        buttonNext = new JButton(">>");
        buttonNext.addActionListener(this);
        buttonBack = new JButton("<<");
        buttonBack.addActionListener(this);
        JPanel panel = new JPanel(new FlowLayout());
        panel.add(buttonBack);
        panel.add(buttonNext);
        add(panel, BorderLayout.SOUTH);
        setLocationRelativeTo(null);
        setResizable(false);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
        //showCurrentImage(0);
    }
    public void showCurrentImage(int i) {
        try {
            ImageIcon icon = new ImageIcon(images[i]);
            Image img = icon.getImage();
            Image newImg = img.getScaledInstance(display.getWidth(),
                                                display.getHeight(), Image.SCALE_SMOOTH);
            ImageIcon newImc = new ImageIcon(newImg);
            display.setIcon(newImc);
            title.setText("Image: " + (i + 1) + " / " + images.length);
        }
        catch (Exception e) {
        }
    }
    public void showNextImage() {
        currentImage += 1;
        if (currentImage >= images.length) {
            currentImage = 0;
        }
        showCurrentImage(currentImage);
    }
    public void showPreviousImage() {
        currentImage -= 1;
        if (currentImage < 0) {
            currentImage = images.length - 1;
        }
        showCurrentImage(currentImage);
    }
}
```

```
}
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == buttonNext) {
        showNextImage();
    }
    else if (e.getSource() == buttonBack) {
        showPreviousImage();
    }
}
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new ImageViewer();
        }
    });
}
```

1.2. Obsługa zdarzeń w klasie wewnętrznej

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class JComboBoxDemo extends JFrame {
    JComboBox combo;
    JLabel label;
    public JComboBoxDemo(String title) {
        super(title);
        String[] data = {"Blue", "Green", "Red", "White", "Yellow"};
        DefaultComboBoxModel comboModel = new DefaultComboBoxModel(data);
        combo = new JComboBox(comboModel);
        combo.setPreferredSize(new Dimension(150, 25));
        combo.addActionListener(new MyListener());
        JPanel p = new JPanel();
        p.add(combo);
        label = new JLabel("Choose item !", JLabel.CENTER);
        Container content = this.getContentPane();
        content.add(p, BorderLayout.PAGE_START);
        content.add(label, BorderLayout.CENTER);
    }
    public static void main(String[] args) {
        JComboBoxDemo test = new JComboBoxDemo("JComboBoxDemo");
        test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        test.setBounds(10, 10, 250, 200);
        test.setVisible(true);
    }
    class MyListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            JComboBox cb = (JComboBox)e.getSource();
            String itemText = (String)cb.getSelectedItem();
            label.setText(itemText);
        }
    }
}
```

1.3. Obsługa zdarzeń w anonimowej klasie wewnętrznej

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Login extends JFrame {
    JButton blogin = new JButton("Login");
    JPanel panel = new JPanel(new GridLayout(2, 2));
```

```

JLabel user = new JLabel("User");
JTextField txuser = new JTextField(10);
JLabel password = new JLabel("Password");
JPasswordField pass = new JPasswordField(10);
Login() {
    super("Autentification");
    this.setSize(300, 150);
    panel.add(user);
    panel.add(txuser);
    panel.add(password);
    panel.add(pass);
    add(panel, BorderLayout.CENTER);
    add(blogin, BorderLayout.SOUTH);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setVisible(true);
    actionLogin();
}
public void actionLogin() {
    blogin.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
            String puname = txuser.getText();
            String ppaswd = pass.getText();
            if (puname.equals("admin") && ppaswd.equals("admin")) {
                NewWindow regFace = new NewWindow();
                regFace.setVisible(true);
                dispose();
            } else {
                JOptionPane.showMessageDialog(null, "Wrong Password / Username");
                txuser.setText("");
                pass.setText("");
                txuser.requestFocus();
            }
        }
    });
}
public static void main(String[] args) {
    Login login = new Login();
}
}
class NewWindow extends JFrame {
    public static void main(String[] args) {
        NewWindow newWindow = new NewWindow();
    }
    NewWindow() {
        super("Welcome");
        setSize(300, 200);
        setLocation(500, 280);
        JPanel panel = new JPanel();
        JLabel welcome = new JLabel("Welcome to a New Frame");
        panel.add(welcome);
        add(panel);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }
}

```

1.4. Obsługa zdarzeń w klasie zewnętrznej

- **Klasa główna**

```

import java.awt.event.*;
import javax.swing.*;

```

```
public class MainClass extends JFrame{
    JTextField tf;
    MainClass(){
        tf=new JTextField();
        tf.setBounds(60,50,150,30);
        JButton b=new JButton("Click me");
        b.setBounds(60,100,150,30);
        Outer o=new Outer(this);
        b.addActionListener(o);
        add(b);
        add(tf);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public static void main(String args[]){
        new MainClass();
    }
}
```

- **Klasa zewnętrzna**

```
import java.awt.event.*;

public class Outer implements ActionListener {
    MainClass obj;
    Outer(MainClass obj) {
        this.obj = obj;
    }
    public void actionPerformed(ActionEvent e) {
        obj.tf.setText("Java");
    }
}
```

2. Zdarzenia od myszki

2.1. Obsługa zdarzeń od myszki w klasie głównej

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MouseEventDemo extends JFrame implements MouseListener {

    JLabel display;

    public MouseEventDemo () {
        super("Mouse Events");
        display = new JLabel("", JLabel.CENTER);
        add(display, BorderLayout.PAGE_START);
        addMouseListener(this);
        setSize(300, 200);
        setVisible(true);
    }

    public static void main(String[] args) {
        MouseEventDemo demo = new MouseEventDemo();
        demo.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void mouseClicked(MouseEvent e) {
        display.setText("Clicked at location: "+location(e));
    }

    private String location(MouseEvent e) {
        return("(" + e.getX() + ", " + e.getY() + ")");
    }
}
```

2.2. Obsługa zdarzeń od myszki z identyfikacją przycisków myszki

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MouseEventDemo extends JFrame {
    public MouseEventDemo() {
        setSize(300, 300);
        setTitle("MouseEventDemo");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        final JTextArea textArea = new JTextArea();
        textArea.setText("Kliknij w obszar okna aplikacji");
        textArea.addMouseListener(new MouseListener() {
            @Override
            public void mousePressed(MouseEvent e) {
                if ((e.getModifiers() & InputEvent.BUTTON1_MASK) ==
                    InputEvent.BUTTON1_MASK) {
                    textArea.setText("Naciśnięty lewy przycisk w pozycji " +
                                     e.getX() + ", " + e.getY());
                } else {
                    textArea.setText("Naciśnięty prawy przycisk w pozycji " +
                                     e.getX() + ", " + e.getY());
                }
            }
            @Override
            public void mouseReleased(MouseEvent e) {
                textArea.setText("Puściłeś przycisk myszy");
            }
            @Override
            public void mouseClicked(MouseEvent e) {
                textArea.setText("Kliknąłeś w pozycji " + e.getX() + ", " +
                                 e.getY());
            }
            @Override
            public void mouseExited(MouseEvent e) {
                textArea.setText("Kursor poza obszarem aplikacji");
            }
            @Override
            public void mouseEntered(MouseEvent e) {
                textArea.setText("Kursor w obszarze aplikacji");
            }
        });
        add(textArea, BorderLayout.CENTER);
    }
    public static void main(String[] args) {
        new MouseEventDemo().setVisible(true);
    }
}
```

2.3. Obsługa zdarzeń od myszki z zastosowaniem klasy adaptacyjnej, zastosować do obsługi myszy klasę adaptacyjną `MouseAdapter` według szablonu

```
public class MyClass extends MouseAdapter {
    ...
    someObject.addMouseListener(this);
    ...
    public void mouseClicked(MouseEvent e) {

        // kod zdarzenia

    }
}
```

3. Zdarzenia od klawiatury

3.1. Obsługa zdarzeń od klawiatury w klasie głównej

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class KeyEventApplication extends JFrame implements KeyListener {
    String str;
    KeyEventApplication() {
        addKeyListener(this);
    }
    public static void main(String[] args) {
        KeyEventApplication frame = new KeyEventApplication();
        frame.setTitle("KeyEventApplication");
        frame.setSize(300, 150);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
    public void keyPressed(KeyEvent e) {
        System.out.println("Key Pressed: " + e);
    }
    public void keyTyped(KeyEvent e) {
        str = e.getKeyChar() + "";
        System.out.println("Key Typed: " + str);
    }
    public void keyReleased(KeyEvent e) {
        System.out.println("Key Released: " + e);
    }
}
```

3.2. Obsługa zdarzeń od klawiatury

```
import java.awt.event.*;
import javax.swing.*;

public class KeyListenerExample implements KeyListener {

    JFrame frame;

    public KeyListenerExample() {
        frame = new JFrame();
        frame.addKeyListener(this);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 300);
        frame.setVisible(true);
    }

    @Override
    public void keyTyped(KeyEvent e) {
        System.out.println("Key Typed: " + e);
    }

    @Override
    public void keyPressed(KeyEvent e) {
        System.out.println("Key Pressed: " + e);
    }

    @Override
    public void keyReleased(KeyEvent e) {
        System.out.println("Key Released: " + e);
    }

    public static void main(String[] args) {
        KeyListenerExample keyListenerExample = new KeyListenerExample();
    }
}
```

```

    }
}

```

3.3. Obsługa zdarzeń od klawiatury w klasie wewnętrznej słuchacza

```

package jp;

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class FocusTest {
    JTextField tf1, tf2, tf3;
    public FocusTest() {
        JFrame f= new JFrame();
        Font font = new Font("Courier New", Font.BOLD, 30);
        tf1 = new JTextField();
        tf1.setFont(font);
        tf1.setBounds(50,50, 200,50);
        tf2 = new JTextField();
        tf2.setFont(font);
        tf2.setBounds(50,150, 200,50);
        tf3 = new JTextField();
        tf3.setFont(font);
        tf3.setBounds(50,250, 200,50);
        tf3.setEditable(false);
        MyKeyListener listener = new MyKeyListener();
        tf1.addKeyListener(listener);
        tf2.addKeyListener(listener);
        f.add(tf1);
        f.add(tf2);
        f.add(tf3);
        f.setTitle("FocusTest");
        f.setSize(350, 500);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        new FocusTest();
    }
    class MyKeyListener implements KeyListener {

        @Override
        public void keyPressed(KeyEvent e) {
            if ((e.getSource() == tf1) & (e.getKeyCode() == KeyEvent.VK_ENTER)) {
                tf3.setText(tf1.getText());
                tf1.setText("");
                tf2.requestFocus();
            } else if ((e.getSource() == tf2) & (e.getKeyCode() ==
                                                                    KeyEvent.VK_ENTER)) {
                tf3.setText(tf2.getText());
                tf2.setText("");
                tf1.requestFocus();
            }
        }
        @Override
        public void keyReleased(KeyEvent e) {
        }
        @Override
        public void keyTyped(KeyEvent e) {
        }
    }
}

```


3.4. Obsługa zdarzeń od klawiatury w zewnętrznej klasie słuchacza

```
import java.awt.event.*;
import javax.swing.*;
public class KeyListenerDemo {

    public static void main(String[] a) {
        JFrame frame = new JFrame("Popup JComboBox");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JTextField textField = new JTextField();
        textField.addKeyListener(new MyKeyListener());

        frame.add(textField);
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}

class MyKeyListener implements KeyListener {
    public void keyPressed(KeyEvent e) {
        System.out.println("Key Pressed");
    }
    public void keyTyped(KeyEvent e) {
        System.out.println("Key Typed: " + e.getKeyChar());
    }
    public void keyReleased(KeyEvent e) {
        System.out.println("Key Released");
    }
}
```

4. Obsługa wielu zdarzeń różnego typu

4.1. Implementacja dwóch interfejsów

```
import java.awt.*;
import java.awt.event.*;

public class MyEvents extends Frame implements MouseListener, ActionListener {
    MyEvents() {
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        setLayout(new FlowLayout());
        Button b=new Button("My New Button");
        b.addActionListener(this);
        b.addMouseListener(this);
        add(b);
    }
    public static void main(String args[]) {
        System.out.println("Starting MyEvents...");
        MyEvents mainFrame = new MyEvents();
        mainFrame.setSize(400, 400);
        mainFrame.setTitle("MyEvents");
        mainFrame.setVisible(true);
    }
    public void mouseEntered(MouseEvent e) {
        e.getComponent().setBackground(Color.red);
    }
    public void mouseExited(MouseEvent e) {
        e.getComponent().setBackground(SystemColor.control);
    }
    public void mousePressed(MouseEvent e) { System.out.println("Mouse pressed");
```

```
    }  
    public void mouseReleased(MouseEvent e) {  
        System.out.println("Mouse released");  
    }  
    public void mouseClicked(MouseEvent e) { System.out.println("Mouse clicked");  
    }  
    public void actionPerformed(ActionEvent e) {  
        System.out.println("Action performed on " + e.getActionCommand());  
    }  
}
```

4.2. Obsługa zdarzeń akcji

```
import java.awt.*;  
import javax.swing.*;  
  
public class AbstractActionDemo extends JFrame {  
    public AbstractActionDemo() {  
        Action open = new OpenFileAction();  
        JMenuBar menuBar = new JMenuBar();  
        JMenu menu = new JMenu("File");  
        JMenuItem item = new JMenuItem(open);  
        menuBar.add(menu);  
        menu.add(item);  
        setJMenuBar(menuBar);  
        JToolBar toolBar = new JToolBar();  
        toolBar.add(open);  
        getContentPane().add(toolBar, BorderLayout.PAGE_START);  
        JPanel panel = new JPanel();  
        JButton button = new JButton(open);  
        panel.add(button);  
        add(panel, BorderLayout.CENTER);  
    }  
}  
  
class OpenFileAction extends AbstractAction {  
    public OpenFileAction() {  
        super("Open", new ImageIcon("open.gif"));  
    }  
    public void actionPerformed(ActionEvent e) {  
        Component comp = (Component)e.getSource();  
        System.out.println("Open" + comp);  
    }  
}
```

5. Obsługa zdarzeń wybranych komponentów Swing

5.1. Okno wyboru pliku

```
import java.awt.*;  
import java.awt.event.*;  
import java.awt.image.*;  
import java.io.*;  
import javax.swing.*;  
  
public class OtwieraniePlikowGIF {  
    public static void main(String[] args) {  
        JFrame ramka = new RamkaPrzegladarki();  
        ramka.setTitle("OtwieraniePlikowGIF");  
        ramka.setSize(400, 500);  
        ramka.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        ramka.show();  
    }  
}  
  
class RamkaPrzegladarki extends JFrame {  
    private JLabel etykieta;  
    public RamkaPrzegladarki() {
```

```

JMenuBar pasekMenu = new JMenuBar();
setJMenuBar(pasekMenu);
JMenu menu = new JMenu("Plik");
pasekMenu.add(menu);
JMenuItem pozycjaOtworz = new JMenuItem("Otwórz");
menu.add(pozycjaOtworz);
pozycjaOtworz.addActionListener(new SluchaczOtwarciaPliku());
JMenuItem pozycjaZakoncz = new JMenuItem("Zakończ");
menu.add(pozycjaZakoncz);
pozycjaZakoncz.addActionListener(new
    ActionListener() {
        public void actionPerformed(ActionEvent zdarzenie) {
            System.exit(0);
        }
    });
etykieta = new JLabel();
Container zawartosc = getContentPane();
zawartosc.add(etykieta);
}
private class SluchaczOtwarciaPliku implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JFileChooser wybor = new JFileChooser();
        wybor.setCurrentDirectory(new File("."));
        wybor.setFileFilter(new
            javax.swing.filechooser.FileFilter() {
                public boolean accept(File p) {
                    return p.getName().toLowerCase().endsWith(".gif") ||
                                                                    p.isDirectory();
                }
                public String getDescription() {
                    return "Obraz GIF";
                }
            });
        int r = wybor.showOpenDialog(RamkaPrzegladarki.this);
        if(r == JFileChooser.APPROVE_OPTION) {
            String nazwa = wybor.getSelectedFile().getPath();
            etykieta.setIcon(new ImageIcon(nazwa));
        }
    }
}
}

```

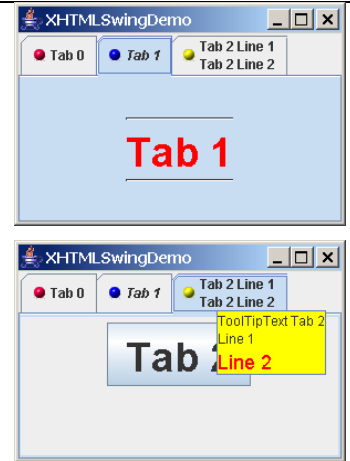
5.2. Panele z zakładkami

Kod przykładowej aplikacji	Zadania
<pre> import javax.swing.*; import javax.swing.event.*; public class JTabbedPaneDemo { public static void main(String[] args) { JFrame frame = new JTabbedPaneFrame(); frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); frame.setVisible(true); } } class JTabbedPaneFrame extends JFrame { ImageIcon icon1, icon2; public JTabbedPaneFrame() { setTitle("JTabbedPaneDemo"); setSize(250, 200); final JTabbedPane tp=new TabbedPane(JTabbedPane.TOP); tp.setTabLayoutPolicy(JTabbedPane.WRAP_TAB_LAYOUT); icon1 = new ImageIcon("blue.gif"); icon2 = new ImageIcon("red.gif"); JLabel label = new JLabel(new ImageIcon("Tab0.jpg")); </pre>	<p>1.3.1. Sprawdzić działanie pól i metod: JTabbedPane.LEFT RIGHT BOTTOM tp.setTabLayoutPolicy(JTabbedPane.SROLL_TAB_LAYOUT);</p> <p>1.3.2. Dodać do każdej zakładki inny panel z komponentami, którego kolor tła będzie można wybrać z okna dialogowego z przykładu 1</p> <p>1.3.3. Sformatować wygląd zakładek jak na rysunkach</p>

```

tp.addTab("Tab0", icon1, label, "ToolTipText Tab 0");
tp.addTab("Tab1", icon2, null, "ToolTipText Tab 1");
// tutaj dodać następne zakładki
getContentPane().add(tp);
tp.addChangeListener(new ChangeListener() {
    public void stateChanged(ChangeEvent e) {
        int n = tp.getSelectedIndex();
        ImageIcon foto = new
            ImageIcon(tp.getTitleAt(n) + ".jpg");
        tp.setComponentAt(n, new JLabel(foto));
        tp.setIconAt(n, icon1);
    }
});
}
}

```



5.3. Panele dzielone

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class JSplitPaneDemo {
    JSplitPane splitPane;
    public JSplitPaneDemo() {
        JButton leftButton = new JButton("Left");
        JButton rightButton = new JButton("Right");
        ActionListener leftButtonActionListener = new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                splitPane.resetToPreferredSizes();
            }
        };
        ActionListener rightButtonActionListener = new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                splitPane.setDividerLocation(30);
                splitPane.setContinuousLayout(true);
            }
        };
        leftButton.addActionListener(leftButtonActionListener);
        rightButton.addActionListener(rightButtonActionListener);
        JPanel p1 = new JPanel();
        p1.add(leftButton);
        JPanel p2 = new JPanel();
        p2.add(rightButton);
        splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
        splitPane.setOneTouchExpandable(true);
        splitPane.setLeftComponent(p1);
        splitPane.setRightComponent(p2);
    }
    public void actionJFrame() {
        JFrame frame = new JFrame("JSplitPaneDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().add(splitPane, BorderLayout.CENTER);
        frame.setSize(300, 150);
        frame.setVisible(true);
    }
    public static void main(String[] args){
        JSplitPaneDemo demo = new JSplitPaneDemo();
        demo.actionJFrame();
    }
}

```

5.4. Okna wewnętrzne

```

import java.awt.*;

```

```

import javax.swing.*;

public class JInternalFrameDemo extends JFrame {
    public JInternalFrameDemo() {
        super("JInternalFrameDemo");
        setSize(500, 400);
        JDesktopPane desktop = new JDesktopPane();
        JInternalFrame iframe1 = new JInternalFrame(
            "JInternalFrame 1", // title
            false, // resizable
            false, // closable
            false, // maximizable
            false); // iconifiable
        iframe1.setBounds(20, 20, 300, 200);
        iframe1.getContentPane().add(new JLabel(new ImageIcon("fig1.jpg")));
        iframe1.setVisible(true);
        desktop.add(iframe1);
        JInternalFrame iframe2 = new JInternalFrame("JInternalFrame 2", true,
                                                    true, true, true);

        iframe2.setBounds(120, 120, 300, 200);
        JScrollPane scrPane = new JScrollPane();
        scrPane.getViewPort().setView(new JLabel(new ImageIcon("fig1.jpg")));
        iframe2.getContentPane().add(scrPane);
        iframe2.setVisible(true);
        desktop.add(iframe2);
        JMenuBar menuBar = new JMenuBar();
        JMenu file = new JMenu("File");
        JMenuItem item1 = new JMenuItem("New");
        JMenuItem item2 = new JMenuItem("Open");
        JMenuItem item3 = new JMenuItem("Close");
        JMenuItem item4 = new JMenuItem("Save");
        file.add(item1);
        file.add(item2);
        file.add(item3);
        file.add(item4);
        menuBar.add(file); iframe2.setJMenuBar(menuBar);
        getContentPane().add(desktop, BorderLayout.CENTER);
    }
    public static void main(String[] args) {
        JInternalFrameDemo demo = new JInternalFrameDemo();
        demo.setVisible(true);
    }
}

```

5.5. Drzewo klasy JTree

```

import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.tree.*;

public class TreeDemo extends JFrame {

    public TreeDemo() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Tree Demo");
        DefaultMutableTreeNode treeRoot = new DefaultMutableTreeNode("Root");
        DefaultMutableTreeNode item1 = new DefaultMutableTreeNode("Item 1");
        DefaultMutableTreeNode item2 = new DefaultMutableTreeNode("Item 2");
        treeRoot.add(item1);
        treeRoot.add(item2);
        final JTree tree = new JTree(treeRoot);
        JScrollPane scroll = new JScrollPane();
        tree.setShowsRootHandles(true);
        tree.setRootVisible(true);
        getContentPane().add(tree);
    }
}

```

```

        tree.getSelectionModel().addTreeSelectionListener(
            new TreeSelectionListener() {
                @Override
                public void valueChanged(TreeSelectionEvent e) {
                    DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) tree
                        .getLastSelectedPathComponent();
                    JOptionPane.showMessageDialog(null, selectedNode
                        .getUserObject().toString());
                }
            });
    }
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                TreeDemo frame = new TreeDemo();
                frame.setSize(400, 300);
                frame.setVisible(true);
            }
        });
    }
}

```

5.6. Okno wyboru koloru

Kod przykładowej aplikacji	Zadania
<pre> import java.awt.*; import java.awt.event.*; import javax.swing.*; public class JColorChooserTest extends JFrame implements ActionListener { public static void main(String[] args) { new JColorChooserTest(); } public JColorChooserTest() { super("Test JColorChooser"); Container content = getContentPane(); content.setBackground(Color.white); content.setLayout(new FlowLayout()); JButton colorButton = new JButton("Wybierz kolor tła"); colorButton.addActionListener(this); content.add(colorButton); setSize(300, 100); setVisible(true); } public void actionPerformed(ActionEvent e) { JColorChooser ch = new JColorChooser(); Color bgColor=ch.showDialog(this,"Wybór koloru tła", getBackground()); if (bgColor != null) getContentPane().setBackground(bgColor); } } </pre>	<p>1.1.1. Zmienić kod tak aby wybierać kolor tła przycisku</p> <p>1.1.2. Zmodyfikować aplikację tak aby w oknie głównym była etykieta z napisem oraz komponent wyboru koloru, z którego po wyborze zmienia się kolor tekstu etykiety. Zastosować interfejs <code>ChangeListener</code></p> <p>Podpowiedź:</p> <pre> JColorChooser ch = new JColorChooser(); ch.getSelectionModel l().addChangeListen er(this); public void stateChanged(Change Event e) { Color newColor = ch.getColor(); label.setForeground (newColor);} </pre>

5.7. Menu z obsługą zdarzeń

```

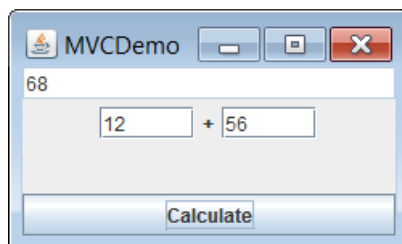
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class Menu extends JPanel implements ActionListener, MenuListener {
    JTextField fldStatus;
    public Menu(JFrame frm) {
        JMenuBar bar = new JMenuBar();
        JMenu menu = new JMenu("File");
        JMenuItem tmp;

```

```
        setBackground(Color.white);
        setLayout(new BorderLayout());
        setDoubleBuffered(true);
        menu.addMenuListener(this);
        tmp = new JMenuItem("New");
        tmp.addActionListener(this);
        tmp.setActionCommand("New");
        menu.add(tmp);
        tmp = new JMenuItem("Open");
        tmp.addActionListener(this);
        tmp.setActionCommand("Open");
        menu.add(tmp);
        tmp = new JMenuItem("Quit");
        tmp.addActionListener(this);
        tmp.setActionCommand("Quit");
        menu.add(tmp);
        bar.add(menu);
        frm.setJMenuBar(bar);
        fldStatus = new JTextField(10);
        fldStatus.setEditable(false);
        add(fldStatus, "South");
    }
    public void actionPerformed(ActionEvent e) {
        String cmd;
        cmd = e.getActionCommand();
        if (cmd.equals("New")) { fldStatus.setText("Action: New"); }
        if (cmd.equals("Open")) { fldStatus.setText("Action: Open"); }
        if (cmd.equals("Quit")) { System.exit(0); }
    }
    public void menuSelected(MenuEvent e) {
        fldStatus.setText("Menu Selected");
    }
    public void menuDeselected(MenuEvent e) {
        fldStatus.setText("Menu Deselected");
    }
    public void menuCanceled(MenuEvent e) {
        fldStatus.setText("Menu Cancelled");
    }
    public static void main(String s[]) {
        JFrame f = new JFrame("Menu");
        Menu panel = new Menu(f);
        f.setForeground(Color.black);
        f.setBackground(Color.lightGray);
        f.getContentPane().add(panel, "Center");
        f.setSize(300,200);
        f.setVisible(true);
        f.addWindowListener(new WindowCloser());
    }
}
class WindowCloser extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
        Window win = e.getWindow();
        win.setVisible(false);
        win.dispose();
        System.exit(0);
    }
}
```

6. Budowa aplikacji w architekturze MVC



- **MVCMain**

```
public class MVCMain {
    public static void main(String[] args) {
        View view = new View();
        Model model = new Model();
        Controller controller = new Controller(view, model);
        view.setVisible(true);
    }
}
```

- **Model**

```
public class Model {
    private int value;
    public void addNumbers(int a, int b){
        value = a + b;
    }
    public int getValue(){
        return value;
    }
}
```

- **Widok**

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class View extends JFrame {
    private JTextField a = new JTextField(5);
    private JLabel label = new JLabel("+");
    private JTextField b = new JTextField(5);
    private JButton button = new JButton("Calculate");
    private JTextField display = new JTextField();
    View() {
        setTitle("MVC");
        setSize(250, 150);
        display.setEditable(false);
        display.setBackground(Color.white);
        add(display, BorderLayout.NORTH);
        add(button, BorderLayout.SOUTH);
        JPanel panel = new JPanel();
        panel.add(a);
        panel.add(label);
        panel.add(b);
        add(panel, BorderLayout.CENTER);
    }
    public int getFirstNumber() {
        return Integer.parseInt(a.getText());
    }
    public int getSecondNumber() {
        return Integer.parseInt(b.getText());
    }
    public void setSolution(int solution) {
        display.setText(Integer.toString(solution));
    }
}
```



```
void addCalculateListener(ActionListener listener) {
    button.addActionListener(listener);
}
void displayErrorMessage(String error) {
    JOptionPane.showMessageDialog(this, error);
}
}
```

- **Sterownik**

```
import java.awt.event.*;
public class Controller {
    private View view;
    private Model model;
    public Controller(View view, Model model) {
        this.view = view;
        this.model = model;
        this.view.addCalculateListener(new CalculateListener());
    }
    class CalculateListener implements ActionListener{
        public void actionPerformed(ActionEvent e) {
            int a, b = 0;
            try{
                a = view.getFirstNumber();
                b = view.getSecondNumber();
                model.addNumbers(a, b);
                view.setSolution(model.getValue());
            }
            catch (NumberFormatException ex) {
                System.out.println(ex);
                view.displayErrorMessage("Enter 2 Integers");
            }
        }
    }
}
```

7. Zadania

Zadania do wykonania podaje prowadzący dla każdej grupy laboratoryjnej