TEHNICI AVANSATE DE PROGRAMARE

LUCRARE DE LABORATOR 7

Java Swing
JFrame, JApplet, JPanel, Borders, Tabbed Panes, Scrolling Panes, Split Panes, Etichete, Butoane

I. SCOPUL LUCRĂRII

Lucrarea de față are rolul de a prezenta şi familiariza studentul cu modul de construire a unei interfețe grafice utilizator folosind pachetul de clase **java.swing.** Se vor prezenta câteva componente vizuale utile, împreună cu modul de creare şi utilizare a acestora.

La sfârşitul acestei lucrări, studentul va avea posibilitatea să scrie mini-aplicații Java în care să utilizeze noțiunile învățate si să scrie interfete grafice Java in care sa integreze componentele invatate.

II. NOȚIUNI TEORETICE

Observație: Vezi noțiunile prezentate în cursurile 5 si 6.

Swing este un subset JFC (Java Foundation Classes) şi constă dintr-o serie de componente vizuale care extind (îmbunătățesc) componentele AWT, şi furnizează noi facilități precum tabele şi arbori. Structura de clase din Swing este asemănătoare cu cea din AWT, în sensul că toate componentele interfeței grafice sunt derivate dintr-un singur părinte numit JComponent (care este derivat din clasa AWT Container).

Pachetul de clase Swing reprezintă soluția furnizată de Sun pentru crearea unor interfețe utilizator grafice complet portabile pe orice platformă.

În Swing, toate numele claselor încep cu litera J, şi atunci când este posibil, numele este acelaşi cu cel al clasei AWT pe care o înlocuieşte.

La fel ca la AWT, punctul de plecare pentru un program bazat pe Swing, este clasa JFrame sau clasa JApplet.

1. JFrame

JFrame este o versiune extinsă a clasei Frame care adaugă suport pentru un comportament de desenare special. Adițional, JFrame permite componentelor Swing MenuBars să fie ataşate nu numai în partea de sus a ferestrei dar oriunde în fereastră.

Toate obiectele asociate unui JFrame sunt manipulate de o instanță a clasei JRootPane, care este singura componentă-fiu a unei instanțe JFrame. JRootPane este un container simplu pentru alte câteva componente. Iată care este ierarhia de obiecte dintr-o instanță JFrame:

JFrame
- JRootPane
  - glassPane
  - layeredPane

[menuBar]
contentPane

O aplicație JFrame

```
import java.awt.*;
import javax.swing.*;
class TestFrame extends JFrame
{
public TestFrame()
{
setTitle( "Test Application" );
setSize( 100, 100 );
setBackground( Color.gray );
Panel topPanel = new Panel();
topPanel.setLayout( new BorderLayout() );
getContentPane().add( topPanel );
Label labelHello = new Label( "Hello World!" );
topPanel.add( labelHello, BorderLayout.NORTH );
}
public static void main( String args[] )
{
TestFrame mainFrame = new TestFrame();
mainFrame.setVisible( true );
}
}
```

Este posibil să se mixeze componentele AWT cu cele Swing, într-o aplicație Java Swing, dar în general se recomandă utilizarea exclusivă a componentelor Swing (un posibil efect: componentele AWT sunt desenate mai rapid undeva în colțul din stânga sus al frame-ului înainte de a fi corect poziționate, rezultând un "flicker" neașteptat în timpul operațiilor de redesenare a ecranului).

Codul din exemplul anterior este foarte simplu şi arată ca şi cum s-ar fi utilizat componente AWT, cu o singură excepție:
getContentPane().add(topPanel);

O clasă JFrame prezintă o singură incompatibilitate în raport cu o clasă AWT. În AWT Frame se puteau adăuga componente direct la instanța frame (pentru că clasa AWT Frame creează automat o instanță Panel).
frame.add(component);

Pentru JFrame, trebuie să specificăm exact în care subcomponentă a lui JRootPane vom plasa componenta noastră. Cel mai adesea, componentele grafice se vor adăuga la contentPane. Trebuie utilizată următoarea sintaxă:
myJFrame.getContentPane().add(component);
Cea mai bună soluție este de se crea un Panel, de a se adăuga acesta la ContentPane, şi de a se adăuga apoi toate componentele la Panel-ul nou creat.

Similar, cînd se setează un "layout" pentru conținutul unui JFrame , de obicei se setează layout-ul pentru subcomponenta contentPane:
myJFrame.getContentPane().setLayout(new FlowLayout());

Variabile JFrame
protected JRootPane rootPane - această variabilă conține o instanță a JRootPane-ului asociat frame-ului.

Constructori JFrame
JFrame()
JFrame(String title)

- creează o nouă instanță JFrame care inițial este invizibilă.

Metode importante JFrame
public void setJMenuBar(JMenuBar menu);

O caracteristică unică a clasei JFrame este abilitatea de a determina cum se va efectua operația de închidere a ferestrei. JFrame implementează metode set/get pentru valoarea operației implicite de închidere.

public void setDefaultCloseOperation(int operation);
public int getDefaultCloseOperation();

"Layered pane" este un container invizibil plasat peste "root pane". Poate fi accesat pentru a afișa peste conținutul frame-ului obiecte dinamice (precum cursoarele)."Glass Pane" permite afișarea unor componente în fața instanței JFrame existente.

protected JRootPane createRootPane();
protected void setRootPane(JRootPane root);
public JRootPane getRootPane();
public Container getContentPane();
public void setLayeredPane(JLayeredPane layered);
public JLayeredPane getLayeredPane();
public void setGlassPane(Component glass);
public Component getGlassPane();

2. JWindow

JWindow este similară cu JFrame exceptând faltul că nu are no "title bar", nu este redimensionabilă, minimizabilă, maximizabilă, şi nu se poate inchide (fără a scrie cod pentru acest lucru). Se utilizează în general pentru a mesaje temporare ("splash screen").

3. JApplet

JApplet are structură similară cu JFrame. Permite adăugarea de componente MenuBars şi toolbars. Exemplu:

```
import java.awt.*;
import javax.swing.*;
public class TestApplet
extends JApplet
{
public TestApplet()
{
}
public void init()
{
setSize( 100, 100 );
setBackground( Color.gray );
Panel topPanel = new Panel();
topPanel.setLayout( new BorderLayout() );
getContentPane().add( topPanel );
Label labelHello = new Label( "Hello World!" );
```

```
topPanel.add( labelHello, BorderLayout.NORTH );
}
}
```

Constructori JApplet
JApplet() -creează o nouă instanță JApplet.

Metode importante JApplet

public void setJMenuBar(JMenuBar menu);
public JMenuBar getJMenuBar();
public void setContentPane(Container contentPane);
public Container getContentPane();
public void setLayeredPane(JLayeredPane layered);
public JLayeredPane getLayeredPane();
public void setGlassPane(Component glass);
public Component getGlassPane();
protected void setRootPane(JRootPane root);
public JRootPane getRootPane();
protected JRootPane createRootPane();

4. JPanel

Echivalentul Swing al clasei AWT Panel este JPanel. JPanel suportă toate tipurile de "layout manager" din AWT, plus cele noi din Swing.

```
import java.awt.*;
import javax.swing.*;
class TestPanel extends JFrame
{
public TestPanel()
{
setSize( 200, 200 );
setBackground( Color.gray );
JPanel topPanel = new JPanel();
topPanel.setLayout( new GridLayout( 3, 2 ) );
getContentPane().add( topPanel );
topPanel.setBackground( Color.lightGray );
topPanel.add( new Button( "One" ) );
topPanel.add( new Button( "Two" ) );
topPanel.add( new Button( "Three" ) );
topPanel.add( new Button( "Four" ) );
topPanel.add( new Button( "Five" ) );
}
public static void main( String args[] )
{
TestPanel mainFrame = new TestPanel();
mainFrame.setVisible( true );
}
}
```

Clasa TestPanel este derivată din JFrame; creează o instanță JPanel căreia i se aplică "GridLayout manager". Butoanele sunt adăugate instanței JPanel, ci nu ferestrei principale.

O instanță JPanel este implicit "double buffered", ceea ce reduce efectul de "flicker" în

timpul operațiilor de redesenare a ecranului, pentru programele de animație. Dacă utilizăm "double-buffering" pentru o componentă, toți fii acesteia vor utiliza de asemenea "double-buffering" (chiar daca nu este activat). JRootPane este componenta din vârful ierarhiei oricărei ferestre Swing, deci activând "double-buffering" pentru JRootPane, toate subcomponentele sale vor fi desenate utilizându-se tehnica "double-buffering".

Constructori JPanel
JPanel(LayoutManager layout, boolean isDoubleBuffered)
- creează o instanță JPanel cu un "layout" specifcat, iar capacitățile de "double buffering" sunt controlate de variabila de tip boolean.
JPanel(LayoutManager layout)
- creează o instanță JPanel cu un "layout" specifcat.
JPanel(boolean isDoubleBuffered)
- creează o instanță JPanel cu un "layout" implicit de tipul FlowLayout, iar capacitățile de "double buffering" sunt controlate de variabila de tip boolean.
JPanel()
- creează o instanță JPanel cu un "layout" implicit de tipul FlowLayout, iar capacitățile de "double buffering" sunt activate.

5. Borders

Pachetul **border** furnizează următoarele clase care por fi aplicate oricărei componente Swing:
BevelBorder – o margine 3D cu o înfățișare "ridicată" sau nu (respectiv **"raised"** sau **"lowered"**).
CompoundBorder - o combinație de 2 alte tipuri de margini: o margine interioară şi o margine exterioară.
EmptyBorder - o margine transparentă utilizată pentru a defini un spațiu vid în jurul unei componente.
EtchedBorder – o margine cu o linie gravată.
LineBorder - o margine cu o grosime şi culoare specificate.
MatteBorder - o margine constând fie dintr-o culoare, fie dintr-o imagine repetată ("tiled").
SoftBevelBorder –o margine 3D cu o înfățișare "ridicată" sau nu, şi cu capetele rotunjite.
TitledBorder – o margine care permite existența unui titlu într-o anume poziție şi locație.

Pentru a seta marginea unei componente Swing se apelează metoda *setBorder()* a JComponentei. Există de asemenea şi o clasă numită *BorderFactory* (în pachetul javax.swing), care conține un grup de metode statice utilizate pentru contruirea rapidă de margini. De exemplu:

myComponent.setBorder(BorderFactory.createEtchedBorder());

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
class BorderTest extends JFrame
{
public BorderTest() {
setTitle("Border Test");
setSize(450, 450);
JPanel content = (JPanel) getContentPane();
content.setLayout(new GridLayout(6,2));
JPanel p = new JPanel();
p.setBorder(new BevelBorder (BevelBorder.RAISED));
```

```
p.add(new JLabel("RAISED BevelBorder"));
content.add(p);
p = new JPanel();
p.setBorder(new BevelBorder (BevelBorder.LOWERED));
p.add(new JLabel("LOWERED BevelBorder"));
content.add(p);
p = new JPanel();
p.setBorder(new LineBorder (Color.black, 5));
p.add(new JLabel("Black LineBorder, thickness = 5"));
content.add(p);
p = new JPanel();
p.setBorder(new EmptyBorder (10,10,10,10));
p.add(new JLabel("EmptyBorder with thickness of 10"));
content.add(p);
p = new JPanel();
p.setBorder(new EtchedBorder (EtchedBorder.RAISED));
p.add(new JLabel("RAISED EtchedBorder"));
content.add(p);
p = new JPanel();
p.setBorder(new EtchedBorder (EtchedBorder.LOWERED));
p.add(new JLabel("LOWERED EtchedBorder"));
content.add(p);
p = new JPanel();
p.setBorder(new SoftBevelBorder (SoftBevelBorder.RAISED));
p.add(new JLabel("RAISED SoftBevelBorder"));
content.add(p);
p = new JPanel();
p.setBorder(new SoftBevelBorder (SoftBevelBorder.LOWERED));
p.add(new JLabel("LOWERED SoftBevelBorder"));
content.add(p);
p = new JPanel();
p.setBorder(new MatteBorder (new ImageIcon("spiral.gif")));
p.add(new JLabel("MatteBorder"));
content.add(p);
p = new JPanel();
p.setBorder(new TitledBorder (
new MatteBorder (new ImageIcon("spiral.gif")),"Title String"));
p.add(new JLabel("TitledBorder using MatteBorder"));
content.add(p);
p = new JPanel();
p.setBorder(new TitledBorder (
new LineBorder (Color.black, 5),"Title String"));
p.add(new JLabel("TitledBorder using LineBorder"));
content.add(p);
p = new JPanel();
p.setBorder(new TitledBorder (
new EmptyBorder (10,10,10,10),"Title String"));
p.add(new JLabel("TitledBorder using EmptyBorder"));
content.add(p);
setVisible(true);
}
public static void main(String args[]) {
new BorderTest();
}
}
```

Crearea unei margini definite de utilizator

Se implementează interfața javax.swing.Border şi se definesc următoarele 3 metode:
void paintBorder(Component c, Graphics g)
Insets getBorderInsets(Component c)
boolean isBorderOpaque()

Să considerăm următorul exemplu. Programul construieşte o margine dreptunghiulară "ridicată" şi umbrită cu colțurile rotunjite. Variabilele instanță:
int m_w: valorile stanga şi dreapta
int m_h: valorile sus şi jos
Color m_topColor: culoarea non-shadow
Color m_bottomColor: culoarea shadow.

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
public class OvalBorder implements Border
{
protected int m_w=6;
protected int m_h=6;
protected Color m_topColor = Color.white;
protected Color m_bottomColor = Color.gray;
public OvalBorder() {
m_w=6;
m_h=6;
}
public OvalBorder(int w, int h) {
m_w=w;
m_h=h;
}
public OvalBorder(int w, int h, Color topColor,
Color bottomColor) {
m_w=w;
m_h=h;
m_topColor = topColor;
m_bottomColor = bottomColor;
}
public Insets getBorderInsets(Component c) {
return new Insets(m_h, m_w, m_h, m_w);
}
public boolean isBorderOpaque() { return true; }
public void paintBorder(Component c, Graphics g,
int x, int y, int w, int h) {
w--;
h--;
g.setColor(m_topColor);
g.drawLine(x, y+h-m_h, x, y+m_h);
g.drawArc(x, y, 2*m_w, 2*m_h, 180, -90);
g.drawLine(x+m_w, y, x+w-m_w, y);
g.drawArc(x+w-2*m_w, y, 2*m_w, 2*m_h, 90, -90);
g.setColor(m_bottomColor);
g.drawLine(x+w, y+m_h, x+w, y+h-m_h);
g.drawArc(x+w-2*m_w, y+h-2*m_h, 2*m_w, 2*m_h, 0, -90);
g.drawLine(x+m_w, y+h, x+w-m_w, y+h);
g.drawArc(x, y+h-2*m_h, 2*m_w, 2*m_h, -90, -90);
}
public static void main(String[] args) {
JFrame frame = new JFrame("Custom Border: OvalBorder");
JLabel label = new JLabel("OvalBorder");
```

```
((JPanel) frame.getContentPane()).setBorder(new CompoundBorder(
new EmptyBorder(10,10,10,10), new OvalBorder(10,10)));
frame.getContentPane().add(label);
frame.setBounds(0,0,300,150);
frame.setVisible(true);
}
}
```

6. Tipurile de Layout Manager specifice Java Swing

BoxLayout - aranjează componentele de-a lungul axelor X, Y ale panel-ului. Încearcă să utilizeze lățimea şi înălțimea preferate ale componentei.
OverlayLayout – aranjează componentele unele peste altele, efectuând o aliniere a punctului de bază al fiecărei componente într-o singură locație.
ScrollPaneLayout – specific pentru **"**scrolling panes"
ViewportLayout – specific panel-urilor cu "scrolling panes", aliniază de-a lungul axei X

Vom prezenta un exemplu pentru tipul **BoxLayout** (cel mai des utilizat). Acest layout manager organizează componentele de-a lungul axelor X, Y ale panel-ului care le deține. Alinierea componentelor se poate face la stânga, la dreapta sau centru (implicit).

```
import java.awt.*;
import javax.swing.*;
class TestFrame extends JFrame
{
public TestFrame()
{
setTitle( "BoxLayout Application" );
JPanel topPanel = new JPanel();
topPanel.setLayout( new BorderLayout() );
getContentPane().add( topPanel );
JPanel yAxisPanel = createYAxisPanel();
topPanel.add( yAxisPanel, BorderLayout.CENTER );
JPanel xAxisPanel = createXAxisPanel();
topPanel.add( xAxisPanel, BorderLayout.SOUTH );
}

public JPanel createYAxisPanel()
{
JPanel panel = new JPanel();
panel.setLayout( new BoxLayout( panel, BoxLayout.Y_AXIS ) );
panel.setBackground( Color.lightGray );
panel.add( new JButton( "Button 1" ) );
panel.add( new TextArea( "This is a text area" ) );
panel.add( new JCheckBox( "Checkbox 1" ) );
return panel;
}

public JPanel createXAxisPanel()
{
JPanel panel = new JPanel();
panel.setLayout( new BoxLayout( panel, BoxLayout.X_AXIS ) );
panel.setBackground( Color.gray );
panel.add( new JButton( "Button 1" ) );
panel.add( new TextArea( "This is a text area" ) );
panel.add( new JCheckBox( "Checkbox 1" ) );
return panel;
```

```
}

public static void main( String args[] )
{
TestFrame mainFrame = new TestFrame();
mainFrame.pack();
mainFrame.setVisible( true );
}
}
```

Pentru a utiliza mai uşor **BoxLayout manager**, Swing furnizează şi o clasă numită **Box** care creează un container ce are aplicat BoxLayout manger. Se utlizează un cod similar cu:

```
{
…
// Creeaza un container nou
Box boxPanel = new Box(BoxLayout.Y_AXIS );
// Adauga componente
boxPanel.add( new JButton( "Button 1" ) );
panel.add( new TextArea( "This is a text area" ) );
panel.add( new JCheckBox( "Checkbox 1" ) );
…
}
```

7. Tabbed Panes

O componentă de tipul ”tabbed pane” permite gruparea mai multor pagini de informație într-un singur punct de referință. Se comportă ca orice altă componentă Swing: putem să îi adăugăm un panou (panel), să îi adăugăm componente de obicei sub formă de pagini. Fiecărei pagini îi putem asocia alte componente Java UI.

Crearea unui “tabbed pane”

```
import javax.swing.*;
{
. . .
tabbedPanel = new JTabbedPane();
topPanel.add( tabbedPanel, BorderLayout.CENTER );
. . .
}
```

Adăugarea şi inserarea de pagini

O pagină constă de obicei dintr-o instanță JPanel conținând componente fiu.

```
import javax.swing.*;
{
. . .
// Creeaza pagina (un “panel”)
pagePanel = new JPanel();
pagePanel.setLayout( new BorderLayout() );
pagePanel.add( new JLabel( "Sample Label" ),BorderLayout.NORTH );
pagePanel.add( new JTextArea( "" ),BorderLayout.CENTER );
pagePanel.add( new JButton( "Button 1" ),BorderLayout.SOUTH );

// Adauga pagina la “tabbed pane”
tabbedPanel.addTab( "Page 1", pagePanel );
. . .
```

```
}
```

Se va utiliza cod similar pentru crearea fiecărei pagini. Dar o astfel de secvență de cod adaugă paginile secvențial. Pentru a insera pagini oriunde într-o ierarhie de pagini se utilizează:

```
// Insereaza pagina in “tabbed pane”
tabbedPanel.insertTab( "Inserted Page",
                       new ImageIcon( "image.gif" ),
                       pagePanel,"My tooltip text",iLocation );
```

Variabila **iLocation** reprezintă index-ul (poziția) paginii. Se observă de asemenea cum se ataşează o imagine.

Ştergerea paginilor

```
tabbedPanel.removeTabAt( iLocation );
```

unde **iLocation** este index-ul paginii ce se va înlătura. Pentru a se şterge toate paginile, trebuie să se țină evidența numărului de pagini rămase, altfel Java VM va genera o excepție.

```
while( tabbedPanel.getTabCount() > 0 )
      tabbedPanel.removeTabAt( 0 );
```

Metoda **getTabCount()** returnează numărul total de pagini din panel.

Selectarea paginilor

Există 2 mecanisme pentru selectarea unei pagini. Cel mai simplu, utilizatorul va selecta cu un click pagina dorită, iar instanța **JTabbedPane** va muta automat pagina selectată în față. Dar se poate scrie şi cod pentru aceasta. Se apelează metoda **setSelectedIndex()** cu indexul paginii care se doreşte să apară în față.

```
tabbedPanel.setSelectedIndex( iLocation );
```

O a doua metodă utilizează instanța panel-ului care a fost referențiat atunci când pagina a fost adăugată.

```
tabbedPanel.setSelectedComponent( pagePanel );
```

Iată un exemplu complet:

```
import java.awt.*;
import javax.swing.*;
class TestTab extends JFrame
{
private JTabbedPane tabbedPane;
private JPanel panel1;
private JPanel panel2;
private JPanel panel3;
public TestTab()
{
setTitle( "Tabbed Pane Application" );
setSize( 300, 200 );
setBackground( Color.gray );
JPanel topPanel = new JPanel();
topPanel.setLayout( new BorderLayout() );
getContentPane().add( topPanel );
// Creeaza paginile
createPage1();
createPage2();
createPage3();
```

```
// Creeaza un “tabbed pane”
tabbedPane = new JTabbedPane();
tabbedPane.addTab( "Page 1", panel1 );
tabbedPane.addTab( "Page 2", panel2 );
tabbedPane.addTab( "Page 3", panel3 );
topPanel.add( tabbedPane, BorderLayout.CENTER );
}
public void createPage1()
{
panel1 = new JPanel();
panel1.setLayout( null );
JLabel label1 = new JLabel( "Username:" );
label1.setBounds( 10, 15, 150, 20 );
panel1.add( label1 );
JTextField field = new JTextField();
field.setBounds( 10, 35, 150, 20 );
panel1.add( field );
JLabel label2 = new JLabel( "Password:" );
label2.setBounds( 10, 60, 150, 20 );
panel1.add( label2 );
JPasswordField fieldPass = new JPasswordField();
fieldPass.setBounds( 10, 80, 150, 20 );
panel1.add( fieldPass );
}
public void createPage2()
{
panel2 = new JPanel();
panel2.setLayout( new BorderLayout() );
panel2.add( new JButton( "North" ), BorderLayout.NORTH );
panel2.add( new JButton( "South" ), BorderLayout.SOUTH );
panel2.add( new JButton( "East" ), BorderLayout.EAST );
panel2.add( new JButton( "West" ), BorderLayout.WEST );
panel2.add( new JButton( "Center" ), BorderLayout.CENTER );
}
public void createPage3()
{
panel3 = new JPanel();
panel3.setLayout( new GridLayout( 3, 2 ) );
panel3.add( new JLabel( "Field 1:" ) );
panel3.add( new TextArea() );
panel3.add( new JLabel( "Field 2:" ) );
panel3.add( new TextArea() );
panel3.add( new JLabel( "Field 3:" ) );
panel3.add( new TextArea() );
}

public static void main( String args[] )
{
TestTab mainFrame = new TestTab();
mainFrame.setVisible( true );
}
}
```

8. Scrolling panes

În următorul exemplu vom crea un “scrolling pane”, şi îi vom adăuga o instanță JLabel care arată o imagine foarte mare. Pentru că imaginea este prea mare ca să fie afişată întreagă, barele de navigare “scroll bars” vor apare automat.

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

class TestScroll extends JFrame
{
private JScrollPane scrollPane;
public TestScroll()
{
setTitle( "Tabbed Pane Application" );
setSize( 300, 200 );
setBackground( Color.gray );
JPanel topPanel = new JPanel();
topPanel.setLayout( new BorderLayout() );
getContentPane().add( topPanel );
Icon image = new ImageIcon( "main.gif" );
JLabel label = new JLabel( image );
// Creeaza un “scroll pane”
scrollPane = new JScrollPane();
scrollPane.getViewport().add( label );
topPanel.add( scrollPane, BorderLayout.CENTER );
}
public static void main( String args[] )
{
TestScroll mainFrame = new TestScroll();
mainFrame.setVisible( true );
}
}
```

9. Split panes

Clasa **JSplitPane** este utilizată pentru a divide două componente, care prin intervenţia utilizatorului pot fi redimensionate interactiv. Divizarea se poate face în direcţia stânga-dreapta utilizând setarea **JSplitPane.HORIZONTAL_SPLIT**, sau în direcţia sus-jos utilizând **JSplitPane.VERTICAL_SPLIT**.

JSplitPane va divide numai două componente. Dacă este nevoie de o interfaţă mai complexă, se poate imbrica o instanţă JSplitPane într-o altă instanţă JSplitPane. Astfel, se va putea intermixa şi divizarea orizontală cu cea verticală.

Graniţa de diviziune poate fi ajustată de către utilizator cu mouse-ul, dar poate fi setată şi prin apelul metodei **setDividerLocation()**. Atunci când graniţa de diviziune este mutată cu mouse-ul de către utilizator, se vor utiliza setările dimensiunilor minime şi maxime ale componentelor, pentru a determina limitele deplasării graniţei. Astfel, dacă dimensiunea minimă a două componente este mai mare decât dimensiunea containerului “split pane”, codul JSplitPane nu va permite redimensionarea frame-urilor separate de graniţa de diviziune.

Exemplu:

```
import java.awt.*;
import javax.swing.*;
class TestSplit extends JFrame
{
private JSplitPane splitPaneV;
private JSplitPane splitPaneH;
private JPanel panel1;
private JPanel panel2;
private JPanel panel3;

public TestSplit()
```

```
{
setTitle( "Split Pane Application" );
setBackground( Color.gray );
JPanel topPanel = new JPanel();
topPanel.setLayout( new BorderLayout() );
getContentPane().add( topPanel );
createPanel1();
createPanel2();
createPanel3();
splitPaneV = new JSplitPane( JSplitPane.VERTICAL_SPLIT );
topPanel.add( splitPaneV, BorderLayout.CENTER );
splitPaneH = new JSplitPane( JSplitPane.HORIZONTAL_SPLIT );
splitPaneH.setLeftComponent( panel1 );
splitPaneH.setRightComponent( panel2 );
splitPaneV.setLeftComponent( splitPaneH );
splitPaneV.setRightComponent( panel3 );
}
public void createPanel1()
{
panel1 = new JPanel();
panel1.setLayout( new BorderLayout() );
panel1.add( new JButton( "North" ), BorderLayout.NORTH );
panel1.add( new JButton( "South" ), BorderLayout.SOUTH );
panel1.add( new JButton( "East" ), BorderLayout.EAST );
panel1.add( new JButton( "West" ), BorderLayout.WEST );
panel1.add( new JButton( "Center" ), BorderLayout.CENTER );
}
public void createPanel2()
{
panel2 = new JPanel();
panel2.setLayout( new FlowLayout() );
panel2.add( new JButton( "Button 1" ) );
panel2.add( new JButton( "Button 2" ) );
panel2.add( new JButton( "Button 3" ) );
}
public void createPanel3()
{
panel3 = new JPanel();
panel3.setLayout( new BorderLayout() );
panel3.setPreferredSize( new Dimension( 400, 100 ) );
panel3.setMinimumSize( new Dimension( 100, 50 ) );
panel3.add( new JLabel( "Notes:" ), BorderLayout.NORTH );
panel3.add( new JTextArea(), BorderLayout.CENTER );
}
public static void main( String args[] )
{
TestSplit mainFrame = new TestSplit();
mainFrame.pack();
mainFrame.setVisible(true);
}
}
```

10. Etichete (Labels)

Etichetele sunt string-uri de text care se utilizează pentru a identifica alte componente. Pot avea propriul tip de font, propriile culori "foreground" (culoarea textului) şi "background" (culoarea de fundal), şi pot fi poziționate oriunde în containerul căruia îi aparțin.

Exemplu de utilizare a clasei **JLabel**:

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

class TestLabel extends JFrame
{
public TestLabel()
{
setTitle( "JLabel Application" );
setSize( 300, 200 );
setBackground( Color.gray );
JPanel topPanel = new JPanel();
topPanel.setLayout( new GridLayout( 2, 2 ) );
getContentPane().add( topPanel );
JLabel label1 = new JLabel();
label1.setText( "Label1" );
label1.setForeground( Color.yellow );
topPanel.add( label1 );
JLabel label2 = new JLabel( "Label2" );
label2.setFont( new Font( "Helvetica", Font.BOLD, 18 ) );
topPanel.add( label2 );


Icon image = new ImageIcon( "myimage.gif" );
JLabel label3 = new JLabel( "Enabled", image,SwingConstants.CENTER );
label3.setVerticalTextPosition( SwingConstants.TOP );
topPanel.add( label3 );
JLabel label4 = new JLabel( "Label4",SwingConstants.RIGHT );
topPanel.add( label4 );
}
public static void main( String args[] )
{
TestLabel mainFrame = new TestLabel();
mainFrame.setVisible( true );
}
}
```

Să observăm că **Label1** îşi schimbă culorile utilizate (metodele *setForeground()* şi *setBackground()*), **Label2** efectuează o schimbare de font, iar **Label3** include o imagine grafică. Dimensiunea imaginii determină dimensiunea minimă a etichetei.

Alinierea textului

Interfața **SwingConstants** conține valori constante pentru toate clasele din biblioteca Swing. Cinci dintre aceste valori constante sunt aplicabile clasei **JLabel** pentru alinierea textului.

SwingConstants.LEFT - aliniere orizontală stânga
SwingConstants.CENTER - aliniere orizontală centru sau aliniere verticală
SwingConstants.RIGHT - aliniere orizontală dreapta
SwingConstants.TOP - aliniere verticală sus
SwingConstants.BOTTOM - aliniere verticală jos

```
label.setHorizontalAlignment( SwingConstants.RIGHT );
label.setVerticalAlignment( SwingConstants.BOTTOM );
```

Pentru etichetele care includ atât text cât şi imagine, textul poate fi aliniat independent de imagine:

```
label.setHorizontalTextAlignment( SwingConstants.LEFT );
label.setVerticalTextAlignment( SwingConstants.TOP );
```

Ataşarea unei imagini la o etichetă

Exemplul anterior arată cum se setează imaginea în timpul construcţiei obiectului JLabel.

Icon image = new ImageIcon("myimage.gif");
JLabel label3 = new JLabel("Label3", image,SwingConstants.CENTER);

Se poate proceda şi altfel – imaginea să fie setată la orice moment.

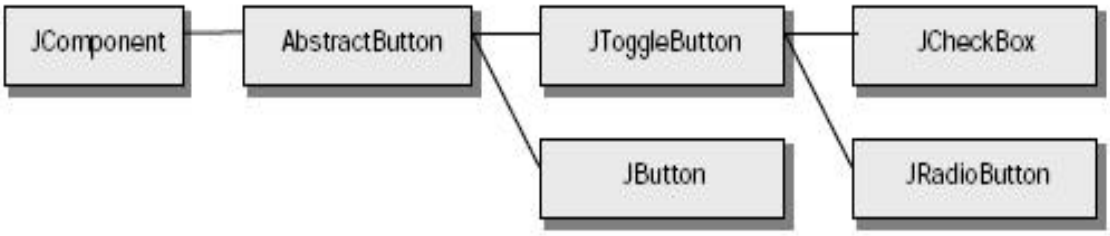
Icon image = new ImageIcon("myimage.gif");
label2.setIcon(image);

Imaginile nu sunt scalate pentru a se potrivi cu limitele dimensiunii etichetei, deci adăugarea unei imagini mari poate cauza efectul de "clipping". Pentru a se înlătura o imagine a unei etichete se utilizează:

label2.setIcon(null);

11. Butoane

Să ne amintim că toate clasele UI sunt derivate din **JComponent**.



Nu se va utiliza direct clasa **AbstractButton**, dar în implementarea diferitelor aplicaţii se vor utiliza clasele-fiu ale acesteia. Clasa **AbstractButton** manipulează aproape toate funcţionalităţile celorlalte clase Swing Button, de aceea o vom studia in detaliu.

Tratarea evenimentelor generate de butoane

Cel mai frecvent se utilizează interfaţa **ActionListener**. Se poate proceda în două moduri. Se ceează o clasă independentă de clasa ce conţine instanţa butonului, iar această clasă va implementa interfaţa **ActionListener (**şi va furniza cod pentru metoda **actionPerformed())**. Avantajul acestei abordări este că mai multe butoane din clase diferite ale unui proiect pot fi manipulate de o singură clasă. A doua abordare este de a implementa interfaţa **ActionListener** în clasa care deţine instanţa butonului. Exemplu:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class TestJBut extends JFrame implements ActionListener
{
private int iCounter = 0; // Keep track of button presses
private JButton button = null;
public TestJBut()
{
setTitle( "ActionListener Application" );
setBackground( Color.gray );
JPanel topPanel = new JPanel();
topPanel.setLayout( new FlowLayout() );
topPanel.setPreferredSize( new Dimension( 300, 200 ) );
```

```
getContentPane().add( topPanel );
button = new JButton( "Press Me" );
topPanel.add( button );
button.addActionListener( this );
}
public void actionPerformed( ActionEvent event )
{
if( event.getSource() == button )
{
iCounter++;
button.setText( "Pressed " + iCounter + " times" );
System.out.println( "Click" );
pack();
}
}
public static void main( String args[] )
{
TestJBut mainFrame = new TestJBut();
mainFrame.pack();
mainFrame.setVisible( true );
}
}
```

Ataşarea unei imagini pentru un buton

Vom modifica exemplul anterior prin adăugarea la suprafața butonului a unei imagini GIF animate.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class JButImg extends JFrame implements ActionListener
{
private int iCounter = 0;
private JButton button = null;
public JButImg()
{
setTitle( "Animated Button Application" );
setBackground( Color.gray );
JPanel topPanel = new JPanel();
topPanel.setLayout( new FlowLayout() );
getContentPane().add( topPanel );
ImageIcon image = new ImageIcon( "computers.gif" );
button = new JButton( "Press Me", image );
button.setPreferredSize( new Dimension( 250, 90 ) );
ImageIcon image1 = new ImageIcon( "appleguy.gif" );
button.setPressedIcon(image1);
button.setMnemonic( 'P' );
topPanel.add( button );
button.addActionListener( this );
JButton but1=new JButton(" OK ");
but1.setEnabled( false );
topPanel.add(but1);
JButton but2=new JButton(" OK ");
but2.setEnabled( true );
topPanel.add(but2);
}
public void actionPerformed( ActionEvent event )
```

```
{
if( event.getSource() == button )
{
iCounter++;
button.setText( "Pressed " + iCounter + " times" );
pack();
}
}
public static void main( String args[] )
{
JButImg mainFrame = new JButImg ();
mainFrame.pack();
mainFrame.setVisible( true );
}
}
```

Dacă un buton conține o imagine, se pot asigna opțional imagini individuale pentru următoarele stări ale butonului: normal, selectat, apăsat, cursorul mouse-ului se află deasupra suprafeței butonului, dezactivat. Se utilizează următoarele metode (vezi documentație clasa AbstractButton): setIcon(),setDisabledIcon(), setDisabledSelectedIcon(), setPressedIcon(), setRolloverIcon(), setRolloverSelectedIcon(), setSelectedIcon().

Butoane activate şi dezactivate

De exemplu, pentru aplicațiile ce conțin formulare bazate pe ferestre de dialog, este util să se dezactiveze butonul OK până când utilizatorul completează toate câmpurile obligatorii. Pentru a activa şi dezactiva un buton se utilizează codul:
button.setEnabled(bState);
unde **bState** este **true** (pentru activare) sau **false** (pentru dezactivare). Dacă butonul este dezactivat, va fi redesenat într-o nuanță de gri şters.

Adăugarea unei combinații de taste

Se pot crea aplicații care să suporte operații fără mouse, numai prin utilizarea tastaturii. Swing aplică această capabilitate familiei sale de componente vizuale, permițând asignarea unei combinații de taste (“keyboard mnemonic”) numită şi accelerator, pentru orice clasă-fiu a clasei părinte JComponent, inclusiv pentru butoane şi “check boxes”. Se utilizează codul:
button.setMnemonic('R');
- asignează tasta *R* instanței button. Pentru efectul de apăsare a butonului, în loc de a se utiliza un click cu mouse-ul, se poate utiliza combinația de taste Alt+R.
Dacă litera se află în eticheta butonului, atunci va apare subliniată.

11.1 Butoane “Toggle”

Swing furnizează şi o clasă numită **JToggleButton**. Butoanele din această clasă au acelaşi aspect ca şi cele JButton, diferența constând în faptul că au 2 stări. Butoanele “Toggle” lucrează aşa cum lucrează tasta Caps Lock de pe tastatură, în timp ce butoanele JButton operează în aceeaşi manieră ca tastele ce reprezintă litere sau cifre. Clasa **JToggleButton** furnizează un mecanism “press-and-hold”, deci sunt ideale pentru interfețele utilizator care necesită operații modale.

Mai multe butoane **JToggleButton** pot fi grupate în aceeaşi manieră ca şi butoanele de tip radio, utilizându-se clasa **ButtonGroup**. Exemplu:

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

```
import javax.swing.*;
class ToggleBut extends JFrame
{
public ToggleBut ()
{
setTitle( "ToggleButton Application" );
setBackground( Color.gray );
JPanel topPanel = new JPanel();
topPanel.setLayout( new FlowLayout() );
getContentPane().add( topPanel );
JToggleButton button1 = new JToggleButton( "Button 1", true );
topPanel.add( button1 );
JToggleButton button2 = new JToggleButton( "Button 2", false );
topPanel.add( button2 );
JToggleButton button3 = new JToggleButton( "Button 3", false );
topPanel.add( button3 );
ButtonGroup buttonGroup = new ButtonGroup();
buttonGroup.add( button1 );
buttonGroup.add( button2 );
buttonGroup.add( button3 );
}
public static void main( String args[] )
{
ToggleBut mainFrame = new ToggleBut ();
mainFrame.pack();
mainFrame.setVisible( true );
}
}
```

11.2 Butoane “CheckBox”

Swing furnizează o clasă, numită **JCheckBox**, care extinde **JToggleButton** pentru a implementa un control standard de tip “check box”. Un “check box” are 2 stări care pot fi setate de către utilizator cu ajutorul mouse-ului sau al unui accelerator al tastaturii, sau programatic utilizând codul:
boolean bValue = checkbox.isSelected();
sau:
checkbox.setSelected(bValue); unde **bValue** este true sau false.

Cel mai eficient se utilizează în grup, pentru a arăta faptul că un obiect poate avea mai multe stări simultan, şi că utilizatorul poate modifica una din ele fără a le afecta pe celelalte. Se pot utiliza si izolat.

Observație:

BoxLayout manager simplifică afişarea componentelor **JCheckBox** în coloană (este valabil şi pentru componentele **JRadioButton**).

Dacă se doreşte, de exemplu, intermixarea unui grup de “check boxes” cu câmpuri de editare (text Fields), cea mai simplă abordare este de a crea pentru componentele JCheckBox un subpanel (utilizând JPanel) având manager-ul BoxLayout, şi de a îl adăuga apoi containerului principal în locația corectă. Exemplu:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
class TestCheckBox extends JFrame
{
```

```
public TestCheckBox ()
{
setTitle( "BoxLayout Application" );
setBackground( Color.gray );
JPanel topPanel = new JPanel();
topPanel.setLayout( new FlowLayout() );
getContentPane().add( topPanel );
JButton button1 = new JButton( "Button 1" );
button1.setMaximumSize( new Dimension( 100, 25 ) );
topPanel.add( button1 );
JPanel innerPanel = new JPanel();
innerPanel.setLayout(new BoxLayout(innerPanel,BoxLayout.Y_AXIS));
innerPanel.setPreferredSize( new Dimension( 150, 120 ) );
innerPanel.setBorder(new TitledBorder(new EtchedBorder(),
                        "Checkboxes"));
topPanel.add( innerPanel );
JCheckBox check1 = new JCheckBox( "Checkbox 1" );
check1.setSelected(true);
innerPanel.add( check1 );
JCheckBox check2 = new JCheckBox( "Checkbox 2" );
innerPanel.add( check2 );
JCheckBox check3 = new JCheckBox( "Checkbox 3" );
innerPanel.add( check3 );
JCheckBox check4 = new JCheckBox( "Checkbox 4" );
innerPanel.add( check4 );
JPanel textPanel = new JPanel( new BorderLayout() );
textPanel.setBorder(new TitledBorder(new EtchedBorder(),"TextArea"));
JTextArea area = new JTextArea( "", 10, 30 );
area.setPreferredSize( new Dimension( 170, 130 ) );
textPanel.add( area );
topPanel.add( textPanel );
}
public static void main( String args[] )
{
TestCheckBox mainFrame = new TestCheckBox ();
mainFrame.pack();
mainFrame.setVisible( true );
}
}
```

11.3 Butoane Radio

În Swing, butoanele Radio sunt implementate în clasa **JRadioButton**, şi sunt şiruri de butoane utilizate pentru a aplica stări mutual exclusive. În Swing, sunt întotdeauna asociate cu o instanţă **ButtonGroup**. Niciodată nu se utilizează izolat, ci numai într-un grup de stări mutual exclusive, permiţând utlizatorului selectarea la un moment dat numai a uneia dintre stări. Exemplu:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
class TestRadio extends JFrame implements ActionListener
{
JButton button1=null;
String string1=" 1 ";
String string2=" 2 ";
String string3=" 3 ";
```

```
public TestRadio ()
{
setTitle( "Radio Buttons Application" );
setBackground( Color.gray );
JPanel topPanel = new JPanel();
topPanel.setLayout( new FlowLayout() );
getContentPane().add( topPanel );

button1 = new JButton( "Button 1" );
button1.setMaximumSize( new Dimension( 100, 25 ) );
topPanel.add( button1 );
//creeaza un grup pentru butoanele radio
ButtonGroup rgroup=new ButtonGroup();
JRadioButton radio1 = new JRadioButton ( string1);
radio1.setActionCommand(string1);
rgroup.add( radio1 );
JRadioButton radio2 = new JRadioButton (string2 );
radio2.setActionCommand(string2);
rgroup.add( radio2 );
JRadioButton radio3 = new JRadioButton ( string3);
radio3.setActionCommand(string3);
rgroup.add( radio3 );
//inregistreaza un listener pentru butoanele radio
radio1.addActionListener(this);
radio2.addActionListener(this);
radio3.addActionListener(this);
//plaseaza butoanele radio orizontal pe un JPanel
JPanel radioPanel = new JPanel( );
radioPanel.setLayout(new BoxLayout(radioPanel,BoxLayout.X_AXIS));
radioPanel.setPreferredSize( new Dimension( 300, 50 ) );
radioPanel.setBorder(new TitledBorder(new EtchedBorder(),
                          "Radio buttons"));
radioPanel.add(radio1);
radioPanel.add(radio2);
radioPanel.add(radio3);
topPanel.add(radioPanel);
}
public void actionPerformed(ActionEvent e)
   {
      button1.setText(e.getActionCommand());
      pack();
   }
public static void main( String args[] )
{
TestRadio mainFrame = new TestRadio ();
mainFrame.pack();
mainFrame.setVisible( true );
}
}
```

III. MODUL DE LUCRU

- Clasic:

1. Se editează codul sursă al programului Java folosind un editor de text disponibil (de ex., se poate utiliza Notepad).
2. Se salvează fişierul cu extensia **.java**.

3. Compilarea aplicației Java se va face din linia de comandă:
   javac nume_fişier_sursă.java
   În cazul în care programul conține erori acestea vor fi semnalate şi afişate.
4. Pentru rularea aplicației Java, se lansează interpretorul Java:
   java nume_clasă_care_conține_main

- Se folosește utilitarul disponibil în laborator J2SDK Net Beans IDE.

IV. TEMĂ

1. Se vor parcurge toate exemplele prezentate în platforma de laborator testându-se practic şi explicând rezultatele obținute.
2. Scrieți o aplicație Java utilizând componente Swing. Aplicația va conține o etichetă, un buton ce are ataşată o imagine şi o combinație de taste, şi un grup de butoane radio. Atunci când se apasă butonul se va modifica imaginea ataşată acestuia. Când se selectează una din opțiunile radio, se va modifica textul etichetei de pe ecran la textul ataşat acelei opțiuni curent selectată.