TEHNOLOGII JAVA
PENTRU DEZVOLTAREA APLICAȚIILOR
LUCRARE DE LABORATOR 12

Java Swing
Tabbed Panes, Scrolling Panes, Split Panes

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 **javax.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 programe Java în care să utilizeze noțiunile învățate.

II. NOȚIUNI TEORETICE

1. 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" ) );
…
}
```

2. 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 );
}
}
```

3. 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 );
}
}
```

4. Split panes

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

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

Granița de diviziune poate fi ajustată de către utilizator cu mouse-ul, dar poate fi setată şi prin apelul metodei **setDividerLocation()**. Atunci când granița 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 graniței. 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 granița 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);
}
}
```

III. MODUL DE LUCRU

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**. Fişierul trebuie salvat la următoarea locaţie: **c:\JBulider7\jdk1.3.1\bin**
3. Compilarea aplicaţiei Java se va face din linia de comandă. Se poate proceda astfel. Se deschide o fereastră MS-Dos: Start ->Run, se tipăreşte **command** în căsuţa de text şi se apasă butonul OK. Printr-o schimbare de directoare şi subdirectoare se trece la locaţia: **c:\JBulider7\jdk1.3.1\bin.** Sau, se lansează WindowsCommander. Se trece la locaţia **c:\JBulider7\jdk1.3.1\bin.** Se deschide o fereastră MS-Dos: Commander ->Run Dos.
4. Pentru compilare, se tipăreşte la prompter **javac nume_fişier_sursă.java** şi se apasă Enter. De ex., dacă fişierul se numeşte Test.java, se va scrie **javac Test.java**. În cazul în care programul conţine erori acestea vor fi semnalate şi afişate în fereastra MS-Dos, după care va apare iar prompter-ul. Dacă programul nu conţine erori şi compilarea se face cu succes, atunci va apare numai prompter-ul.
5. Pentru rularea unei aplicaţii Java stand-alone, se lansează interpretorul Java. Se tipăreşte la prompter următoarea comandă **java nume_clasă_care_conţine_main** şi se apasă Enter. De ex., dacă clasa din program care conţine metoda *main( )* se numeşte Test, se va scrie **java Test**.
6. Dacă programul Java este un applet, se editează fişierul .**html**. Se salvează în acelaşi director cu fişierul **.class** rezultat în urma compilării cu succes a fişierului sursă java. Apoi pentru rulare se poate utiliza **appletviewer nume.html.**
   Alternativ, după compilarea aplicaţiei Java, fişierul **.class** împreună cu fişierul **.html** pot fi mutate în orice alt director (nu trebuie neapărat să fie în **c:\JBulider7\jdk1.3.1\bin**). Se încarcă fişierul **.html** într-un browser Web (ex., Internet Explorer).

IV. TEMĂ

1. Se vor parcurge toate exemplele prezentate în platforma de laborator testându-se practic.
2. Scrieţi o aplicaţie Java în care să construiţi o interfaţă utilizator folosind noile noţiuni prezentate în lucrarea de laborator.