TEHNOLOGII JAVA
PENTRU DEZVOLTAREA APLICAȚIILOR
LUCRARE DE LABORATOR 13

Java Swing. Etichete şi 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 **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. 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 construcției obiectului JLabel.

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

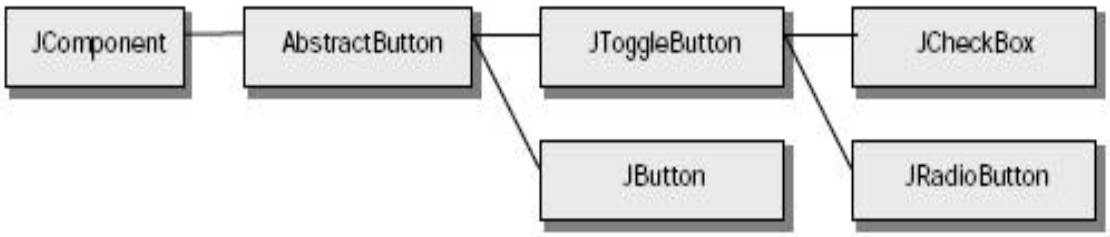
Se poate proceda şi altfel – imaginea să fie setată la orice moment.

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);

2. Butoane

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



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

Tratarea evenimentelor generate de butoane

Cel mai frecvent se utilizează interfața **ActionListener**. Se poate proceda în două moduri. Se ceează o clasă independentă de clasa ce conține instanța butonului, iar această clasă va implementa interfața **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 interfața **ActionListener** în clasa care deține instanța 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.

2.1 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 );
}
}
```

2.2 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 );
```

```
}
}
```

2.3 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 instanță **ButtonGroup**. Niciodată nu se utilizează izolat, ci numai într-un grup de stări mutual exclusive, permițând 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

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. 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.