TEHNOLOGII JAVA
PENTRU DEZVOLTAREA APLICAȚIILOR
LUCRARE DE LABORATOR 16

JDialog. JOptionPane. JColorChooser. JFileChooser. JProgressBar. JScrollBar. JSlider

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. JDialog

Componenta **JDialog** operează asemănător cu **JFrame**. Să considerăm următorul exemplu. Vom crea o aplicație de tip "frame" care conține un singur buton, care acționat creează o nouă instanță de tipul JDialog.

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class Dialog1 extends JFrame implements ActionListener
{
private JPanel topPanel;
private JButton buttonDialog;
public Dialog1()
{
setTitle( "Dialog Test Frame" );
setSize( 310, 130 );
setBackground( Color.gray );
topPanel = new JPanel();
topPanel.setLayout( new BorderLayout() );
getContentPane().add( topPanel );
buttonDialog = new JButton( "Open Dialog" );
topPanel.add( buttonDialog, BorderLayout.CENTER );
buttonDialog.addActionListener( this );
}
```

```
public void actionPerformed( ActionEvent event )
{
System.out.println( event );
TestDialog testDialog = new TestDialog( this );
testDialog.setVisible( true );
}
public static void main( String args[] )
{
Dialog1 mainFrame = new Dialog1();
mainFrame.setVisible( true );
}
}
class TestDialog extends JDialog
{
private JFrame parentFrame;
private JScrollPane scrollPane1;
public TestDialog( JFrame parentFrame )
{
// Ne asiguram ca se apela parintele acestei ferestre
super( parentFrame );
// Salvam ferestra parinte in cazul in care va fi nevoie de ea mai
tarziu
this.parentFrame = parentFrame;
// Setam caracteristicile pentru aceasta instanta dialog
setTitle( "Test Dialog" );
setSize( 200, 200 );
setDefaultCloseOperation( DISPOSE_ON_CLOSE );
// Creeaza un panel pentru componente
JPanel topPanel = new JPanel();
topPanel.setLayout( new BorderLayout() );
getContentPane().add( topPanel );
// Populeaza panel-ul
CreateTopPane( topPanel );
}
private void CreateTopPane( JPanel topPanel )
{
JTextArea area = new JTextArea();
// Incarca un fisier in zona de editare
try {
FileReader fileStream = new FileReader( "Dialog1.java" );
area.read( fileStream, "Dialog1.java" );
}
catch( FileNotFoundException e )
{
System.out.println( "File not found" );
}
catch( IOException e )
{
System.out.println( "IOException occurred" );
}
// Creeaza barele de defilare pentru zona de editare
scrollPane1 = new JScrollPane();
```

```
scrollPane1.getViewport().add( area );
topPanel.add( scrollPane1, BorderLayout.CENTER );
}
}
```

Să observăm următoarea linie de cod:

```
setDefaultCloseOperation( DISPOSE_ON_CLOSE );
```

Această linie de cod controlează modul cum va reacționa fereastra atunci când utilizatorul dorește să o închidă. În cazul nostru, instanța ferestrei dialog va fi distrusă atunci când utilizatorul va închide fereastra. (Implicit, fereastra este doar ascunsă.)

În exemplul anterior, am observat că utilizatorul poate crea un număr nelimitat de ferestre dialog, în timp ce are control independent asupra ferestrei principale a aplicației. Acest tip de fereastră dialog se numește *nonmodală* deoarece fiecare instanță operează independent de celelalte. Clasa **JDialog** suportă și operația *modală*, adică numai o instanță a ferestrei dialog poate fi creată, și atâta timp cât aceasta este activă, fereastra sa părinte ignoră orice acțiune a utilizatorului. În acest caz, se utilizează metoda **setModal()**:

```
TestDialog.setModal( true );
```

2. JOptionPane

Vezi noțiunile teoretice prezentate la curs.

Clasa **JOptionPane** a fost creată pentru a simplifica procesul de prezentare rapidă ("popping up") a informației către utilizator. Următorul exemplu creează o mulțime de instanțe **JOptionPane** pentru tipuri de mesaje predefinite.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class Dialog2 extends JFrame implements ActionListener
{
private JPanel topPanel;
private JButton buttonError;
private JButton buttonWarning;
private JButton buttonInfo;
private JButton buttonQuestion;
private JButton buttonPlain;
public Dialog2()
{
setTitle( "Dialog Test Frame" );
setSize( 310, 130 );
setBackground( Color.gray );
topPanel = new JPanel();
topPanel.setLayout( new FlowLayout() );
getContentPane().add( topPanel );
buttonError = new JButton( "Error" );
topPanel.add( buttonError );
buttonWarning = new JButton( "Warning" );
topPanel.add( buttonWarning );
buttonInfo = new JButton( "Informational" );
```

```
topPanel.add( buttonInfo );
buttonQuestion = new JButton( "Question" );
topPanel.add( buttonQuestion );
buttonPlain = new JButton( "Plain" );
topPanel.add( buttonPlain );
buttonError.addActionListener( this );
buttonWarning.addActionListener( this );
buttonInfo.addActionListener( this );
buttonQuestion.addActionListener( this );
buttonPlain.addActionListener( this );
}
public void actionPerformed( ActionEvent event )
{
System.out.println( event );
if( event.getSource() == buttonError )
{
JOptionPane dialog = new JOptionPane();
dialog.showMessageDialog( this, "This is an error",
                          "Error", JOptionPane.ERROR_MESSAGE );
}
else if( event.getSource() == buttonWarning )
{
Object[] possibleValues = { "First", "Second", "Third" };
JOptionPane dialog = new JOptionPane();
Object selectedValue= dialog.showInputDialog( this,"This is a warning",
                     "Warning", JOptionPane.WARNING_MESSAGE,
                      null, possibleValues, possibleValues[0] );
}
else if( event.getSource() == buttonInfo )
{
JOptionPane dialog = new JOptionPane();
dialog.showConfirmDialog( this,"This is an informational message",
                          "Information", JOptionPane.CANCEL_OPTION,
                          JOptionPane.INFORMATION_MESSAGE, null );
}
else if( event.getSource() == buttonQuestion )
{
JOptionPane dialog = new JOptionPane();
dialog.showConfirmDialog( this, "Is this a question?","Question",

JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE, null );
}
else if( event.getSource() == buttonPlain )
{
JOptionPane dialog = new JOptionPane();
dialog.showConfirmDialog( this, "This is a plain message","Plain",
         JOptionPane.DEFAULT_OPTION,JOptionPane.PLAIN_MESSAGE, null );
}
}
public static void main( String args[] )
{
Dialog2 mainFrame = new Dialog2();
```

```
mainFrame.setVisible( true );
}
}
```

3. JColorChooser

Vezi noțiunile teoretice prezentate la curs.

Clasa **JColorChooser** se poate utiliza ca o fereastră dialog independentă, returnând culoarea selectată de utilizator, sau poate opera ca o componentă ce poate fi adăugată la o fereastră ("frame") care deja există. Să considerăm un exemplu pentru situația când clasa operează ca o fereastră dialog. În acest caz, creează 3 butoane standard: OK, Cancel, şi Reset.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class Dialog3 extends JFrame implements ActionListener
{
private JPanel topPanel;
private JButton buttonFile;
public Dialog3()
{
setTitle( "Color Chooser Dialog Example" );
setSize( 380, 120 );
setBackground( Color.gray );
topPanel = new JPanel();
topPanel.setLayout( new FlowLayout() );
getContentPane().add( topPanel );
buttonFile = new JButton( "Select Color" );
topPanel.add( buttonFile );
buttonFile.addActionListener( this );
}
public void actionPerformed( ActionEvent event )
{
System.out.println( event );
if( event.getSource() == buttonFile )
{
// Deschide un dialog "color chooser" si extrage culoarea
Color color=JColorChooser.showDialog(this,"Select Color",Color.white );
}
}

public static void main( String args[] )
{
Dialog3 mainFrame = new Dialog3();
mainFrame.setVisible( true );
}
}
```

4. JFileChooser

Vezi noțiunile teoretice prezentate la curs. Să considerăm următorul exemplu.

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class FDialog extends JFrame implements ActionListener
{
private final int ITEM_PLAIN = 0; // Item types
private final int ITEM_CHECK = 1;
private final int ITEM_RADIO = 2;
private JTextArea textA;
private JMenuBar menuBar;
private JMenu menuFile;
private JMenuItem menuFileNew;
private JMenuItem menuFileOpen;
private JMenuItem menuFileSave;
private JMenuItem menuFileExit;
private JFileChooser chooser;
public FDialog()
{
setTitle( "Complete Menu Application" );
setSize( 700, 700 );
// Creeaza zona de text
textA = new JTextArea();
textA.setEditable(false);
JScrollPane scrollPane = new JScrollPane();
scrollPane.getViewport().add( textA );
getContentPane().add( scrollPane,BorderLayout.CENTER );
// Creeaza bara de meniu
menuBar = new JMenuBar();
// Seteaza aceasta instanta ca fiind bara de meniu a aplicatiei
setJMenuBar( menuBar );
// Creeaza un meniu "File"
menuFile = new JMenu( "File" );
menuFile.setMnemonic( 'F' );
menuBar.add( menuFile );
// Construieste optiunile meniului "File"
menuFileNew=CreateMenuItem(menuFile,ITEM_PLAIN,"New",null, 'N', null );
menuFileOpen=CreateMenuItem(menuFile,ITEM_PLAIN,"Open",null,
                            'O',"Open a new file" );
menuFileSave = CreateMenuItem( menuFile, ITEM_PLAIN, "Save",null,
                              'S'," Save this file" );
menuFileExit = CreateMenuItem( menuFile, ITEM_PLAIN,"Exit", null,
                               'x',"Exit the program" );
//Creeaza un dialog "file chooser"
chooser=new JFileChooser();
chooser.setCurrentDirectory(new File("."));
}
```

```
public JMenuItem CreateMenuItem( JMenu menu, int iType,String sText,
ImageIcon image,
int acceleratorKey, String sToolTip )
{
// Creeaza optiunea meniului
JMenuItem menuItem;
switch( iType )
{
case ITEM_RADIO:
menuItem = new JRadioButtonMenuItem();
break;
case ITEM_CHECK:
menuItem = new JCheckBoxMenuItem();
break;
default:
menuItem = new JMenuItem();
break;
}
// Adauga textul optiunii
menuItem.setText( sText );
// Adauga imaginea (optional)
if( image != null )
menuItem.setIcon( image );
// Adauga acceleratorul (combinatia de taste)
if( acceleratorKey > 0 )
menuItem.setMnemonic( acceleratorKey );
// Adauga textul pentru "tool tip" (optional)
if( sToolTip != null )
menuItem.setToolTipText( sToolTip );
// Adauga un obiect ascultator de evenimente pentru aceasta optiune a
meniului
menuItem.addActionListener( this );
menu.add( menuItem );
return menuItem;
}

public void actionPerformed( ActionEvent event )
{
if (event.getSource()==menuFileNew)
  {
     textA.setEditable(true);
     textA.setText("");
  }
if (event.getSource()==menuFileOpen)
  {
    int returnVal = chooser.showOpenDialog(FDialog.this);
    if (returnVal == chooser.APPROVE_OPTION)
     {
       File file = chooser.getSelectedFile();
       try {
             FileReader in=new FileReader(file);
```

```
            textA.read(in,null);
            textA.setEditable(true);
            in.close();
           }
     catch(IOException ex) {ex.printStackTrace();}
    }
  }
if (event.getSource()==menuFileSave)
  {
    int returnVal = chooser.showSaveDialog(FDialog.this);
    if (returnVal == chooser.APPROVE_OPTION)
     {
       File file = chooser.getSelectedFile();
       try {
            FileWriter out=new FileWriter(file);
            textA.write(out);
            out.close();
           }
     catch(IOException ex) {ex.printStackTrace();}
    }
  }
if (event.getSource()==menuFileExit)
  {
    System.exit(0);
  }
}
public static void main( String args[] )
{
FDialog mainFrame = new FDialog();
mainFrame.setVisible( true );
}
}
```

5. JProgressBar

Vezi noțiunile teoretice prezentate la curs. Să considerăm următorul exemplu.

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
class Progres extends JFrame implements ActionListener
{
private JProgressBar progress;
private JButton button;
private JLabel label1;
private JPanel topPanel;
public Progres()
{
```

```
setTitle( "Progress Bar Application" );
setSize( 310, 130 );
setBackground( Color.gray );
topPanel = new JPanel();
topPanel.setPreferredSize( new Dimension( 310, 130 ) );
getContentPane().add( topPanel );
// Creeaza o eticheta si o bara de progres
label1 = new JLabel( "Waiting to start tasks..." );
label1.setPreferredSize( new Dimension( 280, 24 ) );
topPanel.add( label1 );
progress = new JProgressBar();
progress.setPreferredSize( new Dimension( 300, 20 ) );
progress.setMinimum( 0 );
progress.setMaximum( 20 );
progress.setValue( 0 );
progress.setBounds( 20, 35, 260, 20 );
topPanel.add( progress );
button = new JButton( "Start" );
topPanel.add( button );
button.addActionListener( this );
}
public void actionPerformed( ActionEvent event )
{
if( event.getSource() == button )
{
// Nu permite mai multe apasari ale butonului
button.setEnabled( false );
for( int iCtr = 1; iCtr < 21; iCtr++ )
{
DoBogusTask( iCtr );
// Actualizeaza indicatorul si eticheta din bara de progres
label1.setText( "Performing task " + iCtr + " of 20" );
Rectangle labelRect = label1.getBounds();
labelRect.x = 0;
labelRect.y = 0;
label1.paintImmediately( labelRect );
progress.setValue( iCtr );
Rectangle progressRect = progress.getBounds();
progressRect.x = 0;
progressRect.y = 0;
progress.paintImmediately( progressRect );
}
}
}
public void DoBogusTask( int iCtr )
{
Random random = new Random( iCtr );
// Pierde un timp
for( int iValue = 0;iValue < random.nextFloat() * 10000; iValue++ )
{
System.out.println( "iValue=" + iValue );
}
```

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

6. JScrollBar

Să considerăm următorul exemplu.

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class Scrol extends JFrame implements AdjustmentListener
{
private JScrollBar scrollerR;
private JScrollBar scrollerG;
private JScrollBar scrollerB;
private JLabel fieldR;
private JLabel fieldG;
private JLabel fieldB;
private JLabel labelR;
private JLabel labelG;
private JLabel labelB;
private JPanel labelColor;
private JPanel topPanel;
public Scrol()
{
setTitle( "ScrollBar Application" );
setBackground( Color.gray );
topPanel = new JPanel();
topPanel.setPreferredSize( new Dimension( 300, 220 ) );
getContentPane().add( topPanel );
labelR = new JLabel( "Red" );
labelR.setPreferredSize( new Dimension( 300, 24 ) );
topPanel.add( labelR );
// Creeaza barele de defilare
scrollerR = new JScrollBar( SwingConstants.HORIZONTAL,0, 0, 0, 255 );
scrollerR.setPreferredSize( new Dimension( 200, 15 ) );
scrollerR.addAdjustmentListener( this );
topPanel.add( scrollerR );
fieldR = new JLabel( "0" );
fieldR.setPreferredSize( new Dimension( 50, 20 ) );
topPanel.add( fieldR );
labelG = new JLabel( "Green" );
labelG.setPreferredSize( new Dimension( 300, 24 ) );
```

```
topPanel.add( labelG );
scrollerG = new JScrollBar( SwingConstants.HORIZONTAL,0, 0, 0, 255 );
scrollerG.setPreferredSize( new Dimension( 200, 15 ) );
scrollerG.addAdjustmentListener( this );
topPanel.add( scrollerG );
fieldG = new JLabel( "0" );
fieldG.setPreferredSize( new Dimension( 50, 20 ) );
topPanel.add( fieldG );
labelB = new JLabel( "Blue" );
labelB.setPreferredSize( new Dimension( 300, 24 ) );
topPanel.add( labelB );
scrollerB = new JScrollBar( SwingConstants.HORIZONTAL,0, 0, 0, 255 );
scrollerB.setPreferredSize( new Dimension( 200, 15 ) );
scrollerB.addAdjustmentListener( this );
topPanel.add( scrollerB );
fieldB = new JLabel( "0" );
fieldB.setPreferredSize( new Dimension( 50, 20 ) );
topPanel.add( fieldB );
labelColor = new JPanel();
labelColor.setPreferredSize( new Dimension( 100, 40 ) );
labelColor.setBackground( new Color( 0, 0, 0 ) );
topPanel.add( labelColor );
}
public void adjustmentValueChanged( AdjustmentEvent event )
{
if( event.getSource() == scrollerR ||event.getSource() == scrollerG
                    ||event.getSource() == scrollerB )
{
// Obtine setarile curente ale culorilor
int iRed = scrollerR.getValue();
int iGreen = scrollerG.getValue();
int iBlue = scrollerB.getValue();
// Seteaza valorile etichetelor
fieldR.setText( "" + iRed );
fieldG.setText( "" + iGreen );
fieldB.setText( "" + iBlue );
// Actualizeaza culoarea
labelColor.setBackground(new Color( iRed, iGreen, iBlue ) );
labelColor.repaint();
}
}
public static void main( String args[] )
{
Scrol mainFrame = new Scrol();
mainFrame.setVisible( true );
mainFrame.pack();
}
}
```

7. JSlider

Vezi noțiunile teoretice prezentate la curs.

Componenta permite utilizatorului să selecteze un număr fix de valori text sau numerice dintr-un domeniu de valori. Să considerăm următorul exemplu:

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
class Slider extends JFrame implements ChangeListener
{
private JSlider scrollerR;
private JSlider scrollerG;
private JSlider scrollerB;
private JLabel labelR;
private JLabel labelG;
private JLabel labelB;
private Label labelColor;
private JPanel topPanel;
public Slider()
{
setTitle( "Slider Application" );
setSize( 330, 280 );
setBackground( Color.gray );
topPanel = new JPanel();
topPanel.setLayout( null );
topPanel.setDoubleBuffered( false );
getContentPane().add( topPanel );
labelR = new JLabel( "Red" );
labelR.setBounds( 20, 15, 250, 20 );
topPanel.add( labelR );
labelG = new JLabel( "Green" );
labelG.setBounds( 20, 75, 250, 20 );
topPanel.add( labelG );
labelB = new JLabel( "Blue" );
labelB.setBounds( 20, 135, 250, 20 );
topPanel.add( labelB );
labelColor = new Label();
labelColor.setBounds( 100, 210, 100, 30 );
labelColor.setBackground( new Color( 0, 0, 0 ) );
topPanel.add( labelColor );
scrollerR = new JSlider( SwingConstants.HORIZONTAL,0, 255, 0 );
scrollerR.setBounds( 20, 35, 290, 40 );
scrollerR.setMajorTickSpacing( 40 );
scrollerR.setMinorTickSpacing( 10 );
scrollerR.setPaintTicks( true );
scrollerR.setPaintLabels( true );
scrollerR.addChangeListener( this );
topPanel.add( scrollerR );
```

```
scrollerG = new JSlider( SwingConstants.HORIZONTAL,0, 255, 0 );
scrollerG.setBounds( 20, 95, 290, 40 );
scrollerG.setMajorTickSpacing( 40 );
scrollerG.setMinorTickSpacing( 10 );
scrollerG.setPaintTicks( true );
scrollerG.setPaintLabels( true );
scrollerG.addChangeListener( this );
topPanel.add( scrollerG );
scrollerB = new JSlider( SwingConstants.HORIZONTAL,0, 255, 0 );
scrollerB.setBounds( 20, 155, 290, 40 );
scrollerB.setMajorTickSpacing( 40 );
scrollerB.setMinorTickSpacing( 10 );
scrollerB.setPaintTicks( true );
scrollerB.setPaintLabels( true );
scrollerB.addChangeListener( this );
topPanel.add( scrollerB );
}
public void stateChanged( ChangeEvent event )
{
if( event.getSource() == scrollerR ||
event.getSource() == scrollerG ||
event.getSource() == scrollerB )
{
// Obtine setarile curente ale culorilor
int iRed = scrollerR.getValue();
int iGreen = scrollerG.getValue();
int iBlue = scrollerB.getValue();
// Actualizeaza culoarea
labelColor.setBackground(
new Color( iRed, iGreen, iBlue ) );
}
}
public static void main( String args[] )
{
Slider mainFrame = new Slider();
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 locația **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 conține erori acestea vor fi semnalate şi afişate în fereastra MS-Dos, după care va apare iar prompter-ul. Dacă programul nu conține erori şi compilarea se face cu succes, atunci va apare numai prompter-ul.
5. Pentru rularea unei aplicații Java stand-alone, se lansează interpretorul Java. Se tipăreşte la prompter următoarea comandă **java nume_clasă_care_conține_main** şi se apasă Enter. De ex., dacă clasa din program care conține 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 aplicației 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.