TEHNOLOGII JAVA
LUCRARE DE LABORATOR 11

Java Swing
JFrame, JApplet, JPanel, Borders

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 programe Java în care să utilizeze noțiunile învățate.

II. NOȚIUNI TEORETICE

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 afişa peste conținutul frame-ului obiecte dinamice (precum cursoarele)."Glass Pane" permite afişarea 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 colţurile rotunjite. Variabilele instanţă:
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);
}
}
```

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**

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.

3. 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.
4. 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**.
5. 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. 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 un applet Java care să conţină 3 componente de tip Panel, fiecare având un tip diferit de margine (border). Pe fiecare panel se vor plasa unul sau mai multe butoane. Se vor utiliza numai componente grafice swing, cu excepţia butoanelor care vor fi AWT.