Laboratorul 7 Evenimente in Java

I. Evenimente in AWT

Vom incepe prin a prezenta doua tabele care sunt utile pana se retin asocierile intre containere, component si evenimente

Eveniment	Cine produce?	Interfete Listener	Metode de tratare
ComponentEvent	Toate componentele	ComponentListener	componentResized() componentMoved() componentShown() componentHidden()
FocusEvent	Toate componentele	FocusListener	focusGained() focusLost()
KeyEvent	Toate componentele	KeyListener	keyTyped() keyPressed() keyReleased()
MouseEvent	Toate componentele	MouseListener MouseMotionListener	mouseClicked() mousePressed() mouseReleased() mouseEntered() mouseExited() mouseDragged() mouseMoved()
ContainerEvent	Toti containerii	ContainerListener	componentAdded() componentRemoved()

Tabel 1. Evenimentele componentelor si containerelor AWT

Eveniment	Cine produce?	Interfete Listener	Metode de tratare
ActionEvent	TextField MenuItem List Button	ActionListener	actionPerformed()
ItemEvent	List CheckBox Choice CheckboxMenuItem	ItemListener	itemStateChanged()
AdjustmentEvent	ScrollPane Scrollbar	AdjustmentListener	adjustmentValueChanged()
TextEvent	TextArea TextField	TextListener	textValueChanged()
WindowEvent	Frame Dialog	WindowListener	windowOpened() windowClosing() windowClosed() windowIconified() windowDeiconified() windowActivated() windowDeactivated()

Tabel 2. Evenimentele specifice componentelor AWT

Preluarea informatiilor de la tastatura folosind java.awt.event.InputEvent

MouseEvent si Keyevent sunt evenimente de tip InputEvent. Daca utilizatorul misca mouse-ul sau apasa o tasta se genereaza evenimente de acest tip avand ca sursa componenta pe suprafata caruia a avut loc evenimentul.

Lista constantelor definite: SHIFT_MASK CTRL_MASK META_MASK ALT_MASK BUTTON1_MASK BUTTON2_MASK BUTTON#_MASK

Pentru a verifica aceste lucruri se va utiliza metoda getModifiers() a clasei InputEvent:

```
public void mousePressed( MouseEvent e){
   int mods = e.getModifiers();
   if( ( mods & InputEvent.SHIFT_MASK) != 0 ){
         // SHIFT apasat
         ...
   }
}
```

Suportul grafic

Desenarea componentelor se face automat si este un proces care se executa în urmatoarele situatii:

- la afisarea pentru prima data a unei componente
- ca raspuns al unei solicitari explicite a programului
- la operatii de minimizare, maximizare, redimensionare a suprafetei de afisare pe care este plasata o componenta

Metodele care controleaza procesul de desenare se gasesc în clasa Component :

void paint(Graphics g)

Deseneaza o componenta. Este o metoda supradefinita de fiecare componenta în parte pentru a furniza reprezentarea sa grafica specifica. Metoda este apelata de fiecare data când continutul componentei trebuie desenat (redesenat) - la afisarea pentru prima data a componentei, la operatii de redimensionare, etc. Nu se apeleaza explicit.

void update(Graphics g)

Actualizeaza starea grafica a unei componente. Actiunea acestei metode se realizeaza în trei pasi:
sterge componenta prin supradesenarea ei cu culoarea fundalului stabileste culoarea (foreground) a componentei apeleaza metoda paint pentru a redesena complet componenta. Nu se apeleaza explicit.

void repaint()

Executa explicit un apel al metodei update pentru a actualiza reprezentarea grafica a unei componente. Desenele care apar pe o suprafata de desenare se realizeaza în metoda paint a unei componente, în general apelata intern sau explicit cu metoda repaint, ori de câte ori componenta respectiva trebuie redesenata.

Un obiect Component (sau o instanţă a unei clase derivate din Component) este desenat sau redesenat prin apelul metodei paint() a instanţei respective (apelul este făcut direct sau prin apelul unor metode intermediare de către AWT). Singurul argument al metodei paint() este un obiect de tip Graphics. Acest obiect oferă suport pentru imagini şi pentru operaţii de desenare primitive (linii, dreptunghiuri, text).

AWT invocă redesenarea componentei în diverse situaţii (redimensionare, acoperire/descoperire de către alte ferestre) dar şi programatorul poate provoca redesenarea unei componente prin apelul metodei repaint().

Exemplu:

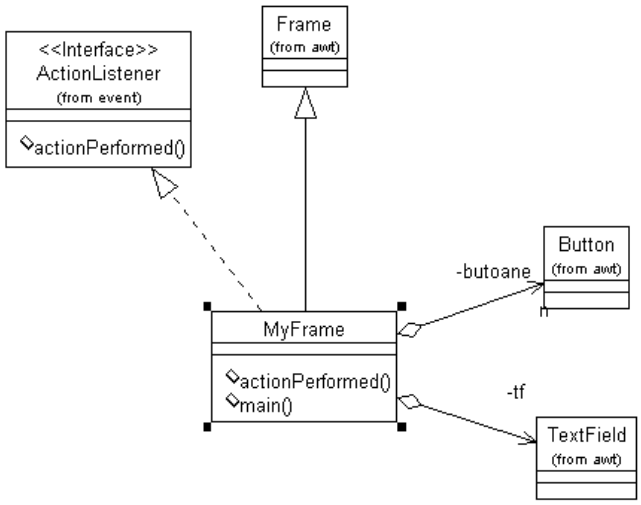
```
import java.awt.*;
class Fereastra extends Frame {
        public Fereastra(String titlu) { super(titlu); setSize (400, 400);
        }

        public void paint(Graphics g) {
                super.paint(g);//metoda paint a clasei de baza - Frame g.setFont(new Font("Arial", Font.BOLD,
                11)); g.setColor(Color.red); g.drawString("Aplicatie", 10, 10);
        }
}

public class TestPaint {
        public static void main(String args[]) {
                Fereastra f = new Fereastra("Test Paint");
                f.show();
        }
}
```

clasa Canvas

In Java a fost definit un tip special de componenta numita Canvas, al carui scop este de a fi extins pentru a implementa componente cu o anumita înfatisare. Clasa Canvas este o clasa generica din care se deriveaza subclase pentru crearea suprafetelor de desenare. Constructorul Canvas() creeaza o plansa, adica o componenta pe care se poate desena. Plansele nu pot contine alte componente grafice, ele fiind utilizate doar ca suprafete de desenat sau ca fundal pentru animatie.
Constructor Canvas ()
Metode addNotify () paint(Graphics)
Metoda paint() a clasei Canvas() picteaza plansa în culoarea implicita
a fundalului. Pentru a redesena plansa cu un alt continut, se recomanda supradefinirea acestei metode implicite.

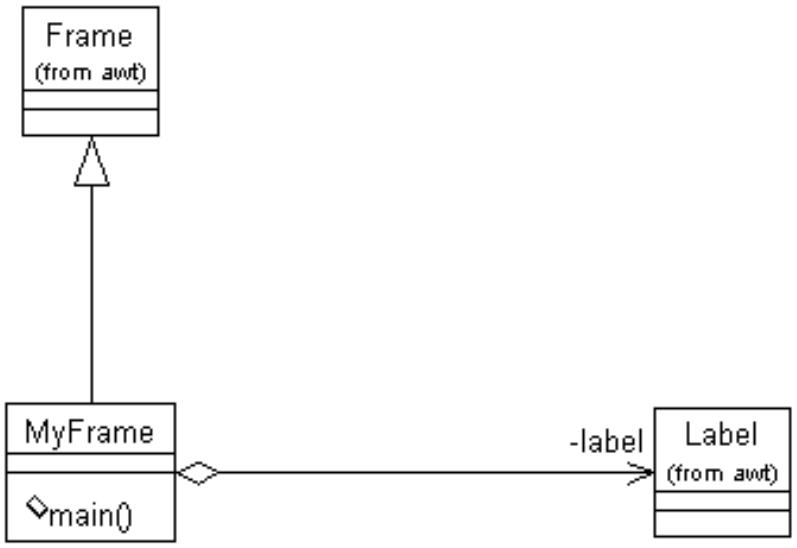
```
class Desen extends Canvas {
public void paint(Graphics g) {
//...desenare continut  }
}
```

Tema 1. Sa se scrie o aplicatie care afiseaza o fereastra avand o caseta text si trei butoane avand etichetele 1,2,3. La apasarea butoanelor in caseta text sa apara eticheta butonului apasat si care are diagrama de clasa de mai jos (20 min):



```
import java.awt.Frame;
import java.awt.event.ActionListener;
import java.awt.Button;
import java.awt.TextField;
import java.awt.event.ActionEvent;
public class MyFrame extends Frame implements ActionListener
{
      private Button butoane[];
      private TextField tf;

      public MyFrame()
      {      setLayout(…);
            …
            for( int i = 0; i< butoane.length; i++ )
            {
               …
            }
            // Tratarea evenimentelor cu fereastra
            addWindowListener( new java.awt.event.WindowAdapter()
            {
             …
            }
            });
      }

      public void actionPerformed(ActionEvent arg0)
      {    …
      }

      public static void main(String[] args)
      {    …
      }
}
```

Tema 2. Sa se scrie o aplicatie care se ruleaza intr-o fereastra si la apasarea mouseului in zona ferestrei sa apara o eticheta care sa contina coordonatele apasarii mouseului sic area are Diagrama de clasa de mai jos (20 min)



```
import java.awt.Frame;
import java.awt.Label;
public class MyFrame extends Frame
{
   private Label label;
   public MyFrame()
   {
      setLayout( … );
      ...
      // Inregistrarea ferestrei ca si receptor pentru evenimentele cu mouse-ul
      addMouseListener( new java.awt.event.MouseAdapter(){
         public void mouseClicked( java.awt.event.MouseEvent e )
         {
          ….
         }
      });
      // Inregistrarea ferestrei ca si receptor pentru evenimnetele cu fereastra
      addWindowListener( new java.awt.event.WindowAdapter(){
         public void windowClosing( java.awt.event.WindowEvent e ){
          …
         }
      });
   }
   public static void main(String[] args)
   {
    …
   }
}
```

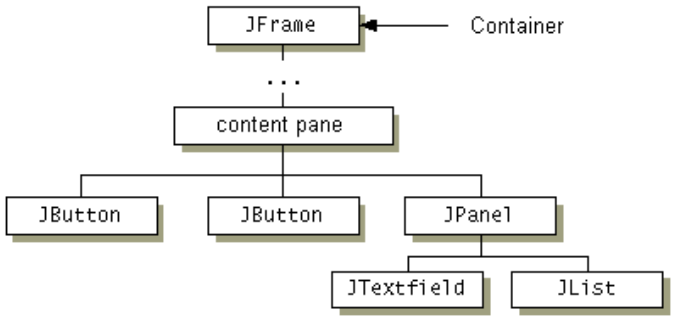
Tema 3. Sa se afiseze un cerc intr-un canvas (10 min)

II. Utilizarea componentelor Swing

Pachetul swing contine doua tipuri de componente:

- containere (*JFrame, JApplet, JWindow, JDialog*)
- componente "lightweight" (*JButton, JPanel, JList etc.*)

Containerele reprezinta cadrul in care aceste componente pot exista. Imaginea urmatoare prezinta ierarhia componentelor unei aplicatii cu o fereastra, doua butoane, o caseta text si o lista.



Obiectul *contentpane* din figura precedenta este un obiect *Container* obisnuit si este sub containerul swing.

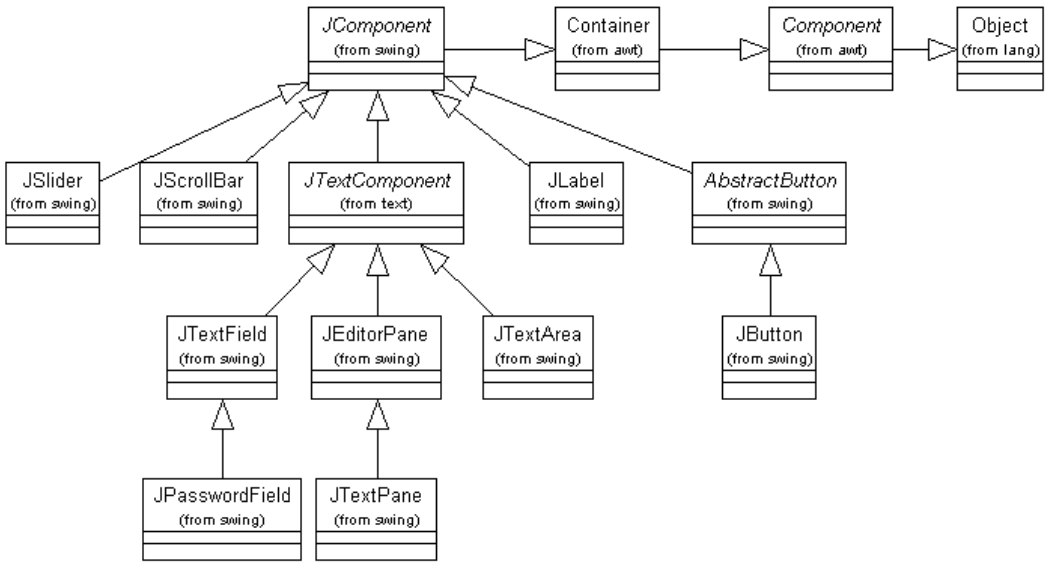
Codul pentru crearea unei suprafetei grafice:

```
JPanel panel = new JPanel();
panel.setLayout(new FlowLayout());
panel.add(textField);
panel.add(list);
```

```
Container contentPane = this.getContentPane();
contentPane.setLayout(new FlowLayout());
contentPane.add(button1);
contentPane.add(button2);
contentPane.add(panel);
```

Observatie: Nu se pot adauga componentele direct la containerul swing.

Figura urmatoare contine cateva componente impreuna cu ierarhia de clase in care se incadreaza:



JButton

Un buton *JButton* se instantieaza analog ca si un butom *java.awt.Button*. Evenimentul care se produce la apasarea acestuia este tot un *ActionEvent* si evenimentul se livreaza tuturor receptorilor inregistrati. Culoarea butonului este identic cu culoarea containerului, deci trebuie modificat la *SystemColor.control*.

```
setLayout( new FlowLayout() );
JButton simplebutton = new JButton("Simple");
add( simplebutton );
simplebutton.setBackground(SystemColor.control);
Icon icon = new ImageIcon("icon.gif");
JButton iconbutton = new JButton("Icon", icon );
iconbutton.setBackground(SystemColor.control);
add( iconbutton );
```

JTextComponent

JTextComponent este o clasa generala cu toate functiile unui editor de text simplu. Metodele cele mai utilizate:

- copy()
- cut()
- paste()
- getSelectedText()
- setSelectionStart()
- setSelectionEnd()selectAll()
- replaceSelection()
- getText()
- setText()

- setEditable()
- setCaretPosition()

Subclasele clasei *JTextComponent* sunt: *JTextField, JTextArea, JTextPane*.

```
setLayout( new BorderLayout());
JTextPane tp = new JTextPane();
MutableAttributeSet attr = new SimpleAttributeSet();
StyleConstants.setFontFamily(attr,"Serif");
StyleConstants.setFontSize(attr,18);
StyleConstants.setBold(attr,true);
tp.setCharacterAttributes( attr, false );
add( tp, BorderLayout.CENTER );
```

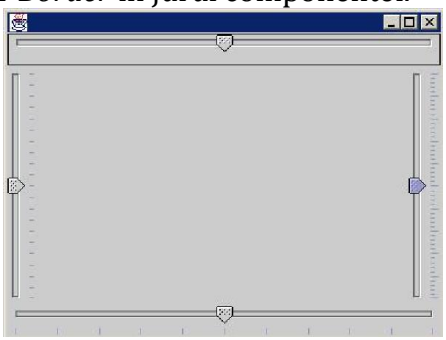
JScrollBar

Este varianta "lightweight" a componentei java.awt.Scrollbar.

```
setLayout( new BorderLayout() );
JScrollBar vertical = new JScrollBar(JScrollBar.VERTICAL,0,5,0,100);
add(vertical, BorderLayout.EAST );
JScrollBar horizontal = new JScrollBar(JScrollBar.HORIZONTAL,0,5,0,100);
add(horizontal, BorderLayout.SOUTH );
```

JSlider

JSlider este similar cu *JScrollBar*, dar permite afisarea marcajelor minore respectiv a celor majore precum si desenarea unui chenar *Border* in jurul componentei.



```
setLayout( new BorderLayout() );

JSlider s1 = new JSlider( JSlider.VERTICAL, 0, 100, 50 );
s1.setPaintTicks( true );
s1.setMajorTickSpacing(10);
s1.setMinorTickSpacing(2);
add( s1, BorderLayout.EAST);

JSlider s2 = new JSlider( JSlider.VERTICAL, 0, 100, 50 );
s2.setPaintTicks( true );
s2.setMinorTickSpacing(5);
add( s2, BorderLayout.WEST);

JSlider s3 = new JSlider( JSlider.HORIZONTAL, 0, 100, 50 );
s3.setPaintTicks( true );
s3.setMajorTickSpacing(10);
add( s3, BorderLayout.SOUTH);


JSlider s4 = new JSlider( JSlider.HORIZONTAL, 0, 100, 50 );
s4.setPaintTicks( true );
s4.setBorder( LineBorder.createBlackLineBorder());
add( s4, BorderLayout.NORTH);
```

JList

Componenta are doua implementari, una care respecta arhitectura MVC si una care nu, se comporta la fel ca si o lista AWT. Totusi si aceasta implementare mai simpla s-a imbunatatit. Putem adauga elementele prin contructor transmitandu-i un tablou de stringuri. JList nu suporta scroll, adica daca dorim aceasta facilitate atunci trebuie s-o punem intr-un ScrollPane.

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

public class SwingPanel extends JPanel
{
    String elements[] = {"Orange","Lemon","Strawberry","Raspberry","Apple","Banana"};

    public SwingPanel()
    {
        setLayout( new BorderLayout() );
        JList list = new JList(elements);
        ScrollPane pane = new ScrollPane();
        pane.add( list );
        add( pane, BorderLayout.CENTER );
    }
}
```

Tema 4. Sa se Creeze un editor de tip notepad (avem unde edita, bare de scrol, butoane control, etc) 50 min.

Tema pe acasa. Sa se proiecteze (clase creion hartie etc) si sa se implementeze intefata grafica a jocului cu bataie in spatiu de ora trecuta

Va contine o zona de desenare (unde se va afisa pentru test o nava piramidala desenata), lista de selectie tip de nava, layout-urile vor fi functie de imaginatia fiecaruia dar trebuie, butoane de comanda si control, zona grafica separata de afisare a pozitiei spatiale globale pentru toate navele, cutie pt preluarea datelor unde preiau numele jucatorului, afisare coordonate, etc)