https://docs.oracle.com/javase/tutorial/uiswing/events/

ListBox

ListBox-urile afiseaza liste de optiuni. Ele se compun dintr-un element care se ocupa de vizualizare (derivat din clasa **JList**), respectiv dintr-un element care se ocupa cu gestionarea continutului (derivat din **ListModel**).

Interfata **ListModel** pune la dispozitie o metoda **addElement()** care permite adaugarea unei noi optiuni in lista.

Constructorul clasei **JList** primeste un obiect de tip **ListModel** pe care il va afisa pe ecran. Pentru a avea la dispozitie bare de derulare asociate listei, aceasta va trebui inclusa intr-un element de tip **JScrollPane** (un panou derulant).

Aditional listelor simple se pot defini si liste cu derulare (de tip ComboBox). Acestea afiseaza in mod obisnuit o singura optiune din lista iar pentru a accesa restul optinulor lista trebuie derulata de la un buton special. Listele derulante nu trebuie sa fie adaugate intr-un **JScrollPane**.

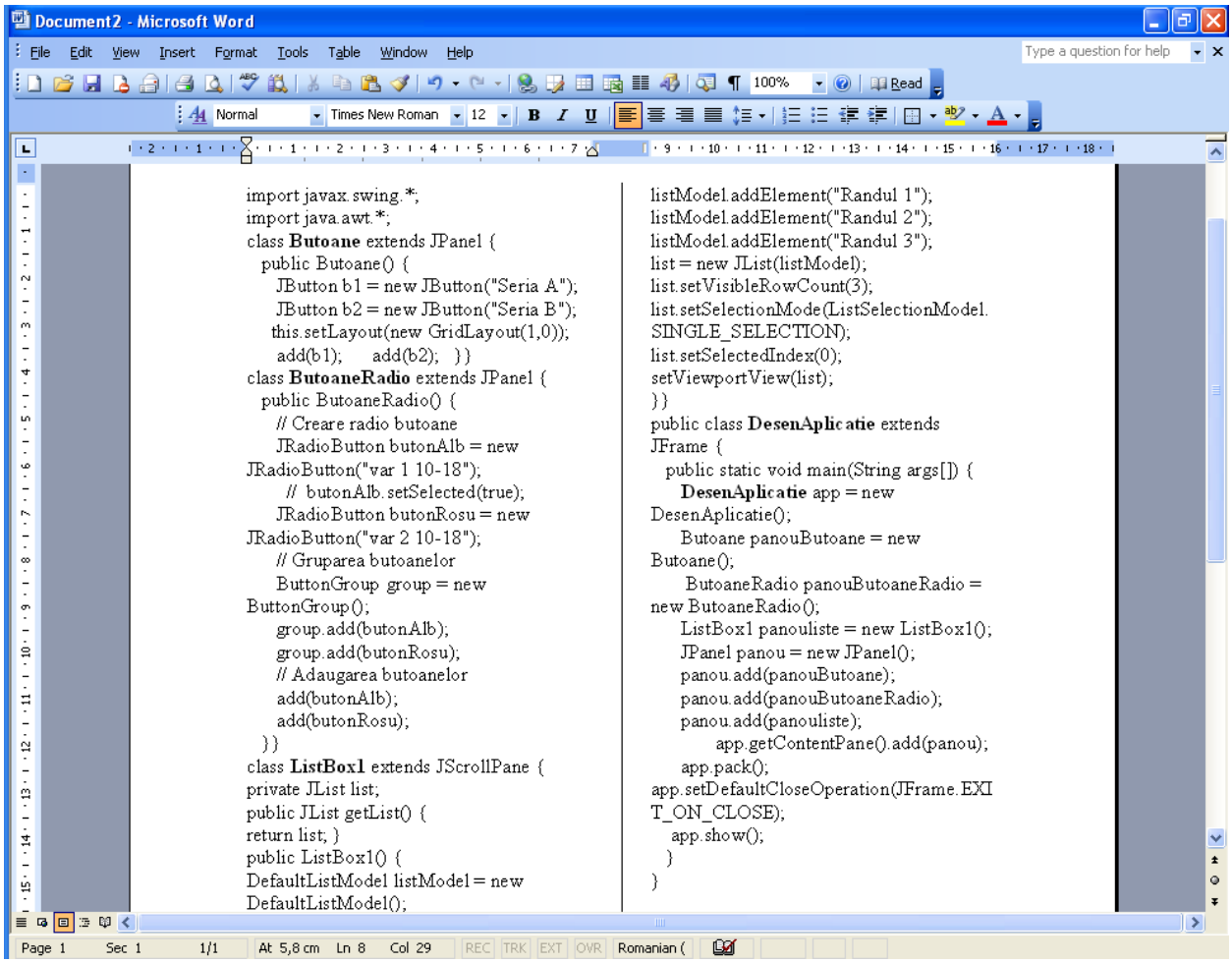
```
ListModel listModel = new DefaultListModel();
listModel.addElement("Linie1");
…
JList list = new JList(listModel);
list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

// pentru selectii multiple se utiliyeaya parametrul
MULTIPLE_INTERVAL_SELECTION
list.setSelectedIndex(0);

// primul element din lista este implicit selectat
JScrollPane listScrollPane = new JScrollPane(list);
--------
```

String[] culori = { "Alb", "Negru", "Rosu", "Verde", "Albastru" };

JComboBox listaCulori = new JComboBox(culori);

listaCulori.setSelectedIndex(1); // selecteaza elementul 2 (Negru)

listaCulori.setEditable(true);



Tema:

Sa se realizeze o interfata care contine:
-2 componente JButton
-2 component JCheckBox

-3 componente JRadioButton(care apartin aceluiasi ButtonGroup)
-1 componenta JList cu 3 elemente

JTable

Clasa JTable face parte din Swing si se extinde din Component, utilizeaza cate un model din mai multe interfete de tip „ascultător” , cum ar fi TableModelListener, TableColumnModelListener, ListSelectionListener ...

JTable este de obicei plasat într-un JScrollPane.

Fiecare JTable are trei modele

TableModel, TableColumnModel, si ListSelectionModel.

TableModel este folosit pentru a specifica modul în care datele tabelului se stochează. Datele JTable este gestionat intr-o matrice cu două dimensiuni sau un vector de vectori. TableModel, de asemenea, este utilizat pentru a specifica modul în care datele pot fi editate în tabel.

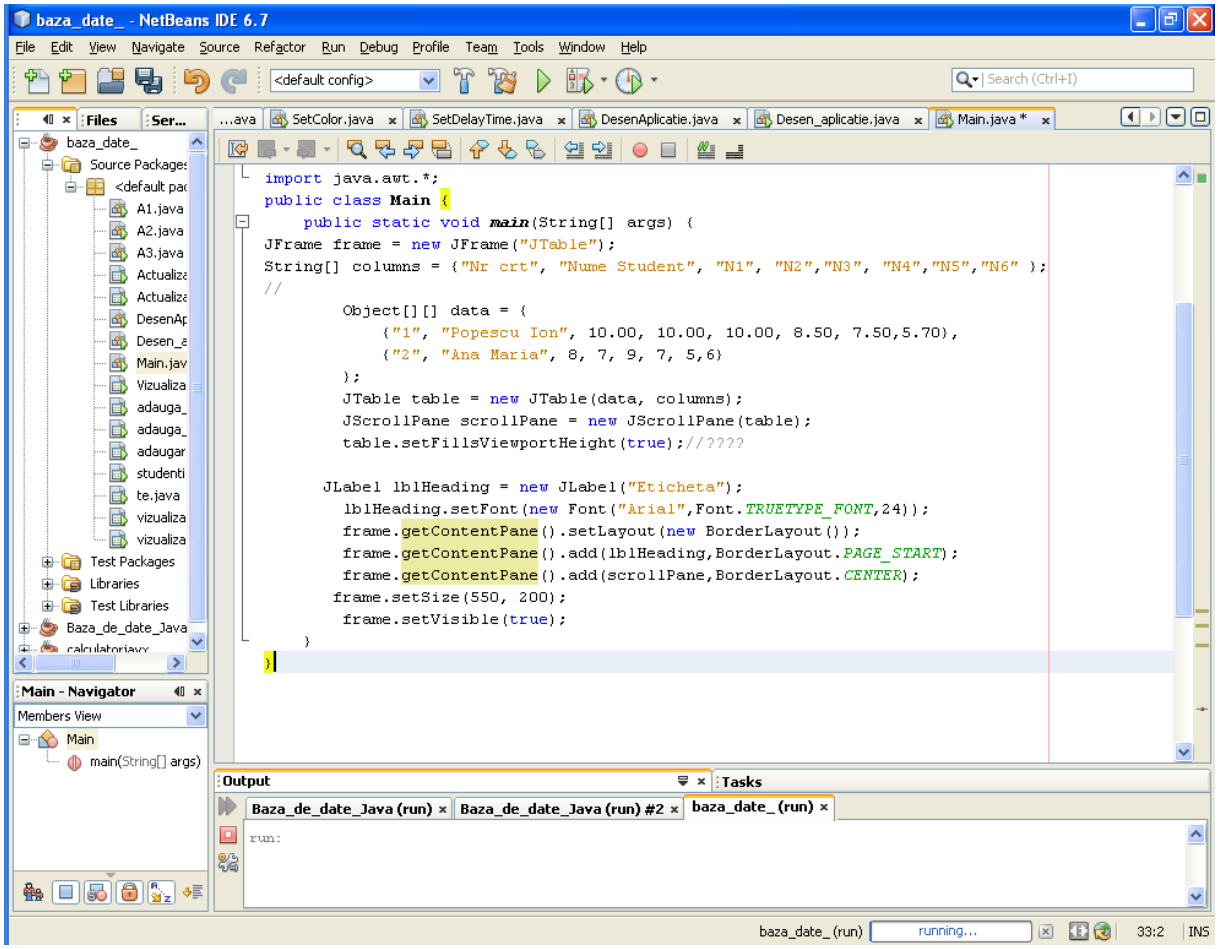
TableColumnModel este utilizat pentru a gestiona toate coloanele tabelului.

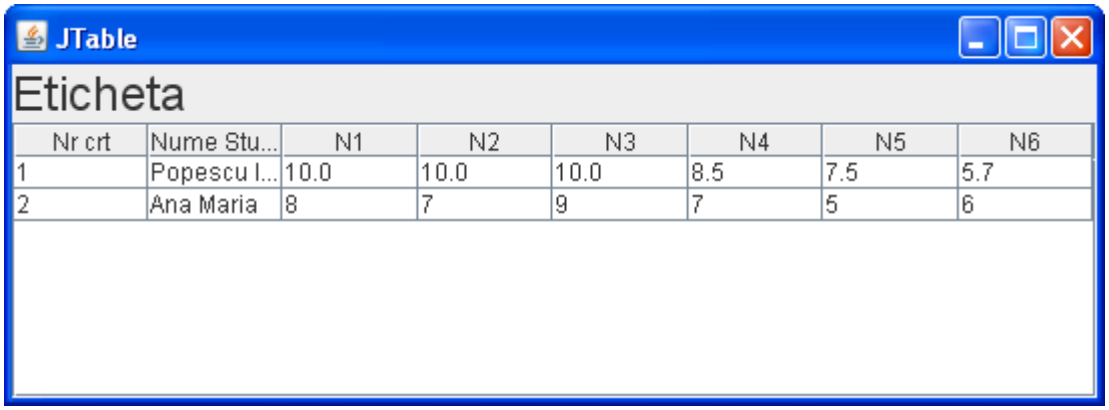
ListSelectionModel permite ca pentru tabel sa se definesca un alt mod de selectie, ca de exemplu intervalul.

Constructorii pentru obiectul JTable sunt:

Tabel 1

JTable Constructors	Description
JTable()	Create an empty table
JTable(int rows, int columns)	Create a table with rows and columns empty cell.
JTable(Object[][] data, Object[] heading)	Create a table with data specify in two-dimensional array *data* and column heading *heading*
JTable(TableModel dm)	Create a table with a given TableModel
JTable(TableModel dm, TableColumnModel cm)	Create a table with a given TableModel and TableColumnModel.
JTable(TableModel dm,TableColumnModel cm, ListSelectionModel sm)	Create a table with a given TableModel, TableColumnModel, and ListSelectionModel.
JTable(Vector data, Vector heading)	Create a table with data in vector of Vectors *data* and column headings *headin.*


baza_date_ - NetBeans IDE 6.7
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Search (Ctrl+I)
Files
Ser...
baza_date_
Source Packages
A1.java
A2.java
A3.java
Test Packages
Libraries
Test Libraries
Baza_de_date_Java
Main - Navigator
Members View
Main
main(String[] args)
SetColor.java
SetDelayTime.java
DesenAplicatie.java
Desen_aplicatie.java
Main.java *
import java.awt.*;
public class Main {
public static void main(String[] args) {
JFrame frame = new JFrame("JTable");
String[] columns = {"Nr crt", "Nume Student", "N1", "N2","N3", "N4","N5","N6" };
//
Object[][] data = {
{"1", "Popescu Ion", 10.00, 10.00, 10.00, 8.50, 7.50,5.70},
{"2", "Ana Maria", 8, 7, 9, 7, 5,6}
};
JTable table = new JTable(data, columns);
JScrollPane scrollPane = new JScrollPane(table);
table.setFillsViewportHeight(true);//????
JLabel lblHeading = new JLabel("Eticheta");
lblHeading.setFont(new Font("Arial",Font.TRUETYPE_FONT,24));
frame.getContentPane().setLayout(new BorderLayout());
frame.getContentPane().add(lblHeading,BorderLayout.PAGE_START);
frame.getContentPane().add(scrollPane,BorderLayout.CENTER);
frame.setSize(550, 200);
frame.setVisible(true);
}
}
Output
Tasks
Baza_de_date_Java (run)
Baza_de_date_Java (run) #2
baza_date_ (run)
run:
baza_date_ (run)
running...
33:2
INS


JTable
Eticheta
Nr crt
Nume Stu...
N1
N2
N3
N4
N5
N6
1
Popescu I...
10.0
10.0
10.0
8.5
7.5
5.7
2
Ana Maria
8
7
9
7
5
6

Crearea meniurilor

Modelul orientat obiect al meniurilor în Swing are următoarele caracteristici:

- bară principală reprezintă o instanţă a clasei *JMenuBar*, sau, pentru meniurile contextuale, *JPopUpMenu*;
- meniurile accesebile din bara principală sunt obţinute din clasa *JMenu*;
- elementele individuale ale unui menu (instanţă *JMenu*) sunt construite pe baza clasei *JMenuItem* care are următoarele subclase: *JCheckBoxMenuItem*, *JRadioButtonMenuItem* şi *JMenu*. Prin urmare un element individual dintr-un meniu poate fi, la rândul său, un al submenu (instanţă *JMenu*);

Definirea celui mai simplu meniu este următoarea:

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

public class MainMenuBar extends JMenuBar{
   JFrame frmParent;
   public MainMenuBar() {
      JMenu fileMenu = new JMenu("File");
      JMenu editMenu = new JMenu("Edit");
      JMenu quitMenu = new JMenu("Quit");

      // Meniul File: Open, Save, Close
      JMenuItem openItem = new JMenuItem("Open");
      JMenuItem saveItem = new JMenuItem("Save");
      JMenuItem closeItem = new JMenuItem("Close");
```

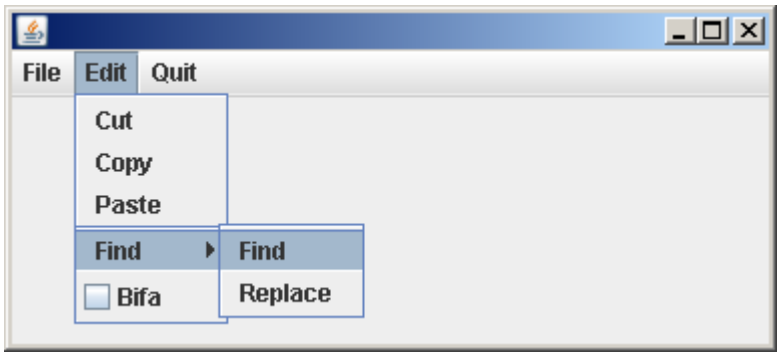
```
//Meniul Edit: Cut, Copy, Paste, Find (Find, Replace)
JMenuItem cutItem = new JMenuItem("Cut");
JMenuItem copyItem = new JMenuItem("Copy");
JMenuItem pasteItem = new JMenuItem("Paste");
JSeparator separator1 = new JSeparator();

JMenu submeniu = new JMenu("Find");

JMenuItem findItem = new JMenuItem("Find");
JMenuItem replaceItem = new JMenuItem("Replace");
//Meniul Quit
JMenuItem exitItem = new JMenuItem("Exit");
JMenuItem aboutItem = new JMenuItem("About");
fileMenu.add(openItem);
fileMenu.add(saveItem);
fileMenu.add(closeItem);
editMenu.add(cutItem);
editMenu.add(copyItem);
editMenu.add(pasteItem);
editMenu.add(separator1);
findMenu.add(findItem);
findMenu.add(replaceItem);
editMenu.add(findMenu);
editMenu.add(new JCheckBox("Alegeti optiunea"));
quitMenu.add(exitItem);
quitMenu.add(aboutItem);
 add(fileMenu);
```

```
        add(editMenu);
        add(quitMenu);
    }
}
```

Pentru a dispune un meniu într-un formular, avem nevoie de o instanţă *JFrame* pentru bara de meniu, stabilită prin metoda *setJMenuBar()*. De exemplu clasa principala este:

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class FormMain extends JFrame{
    public FormMain() {
        this.setJMenuBar(new MainMenuBar());
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(500, 500);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new FormMain();
    } }
```

va avea la runtime următorul rezultat:



Varianta 2

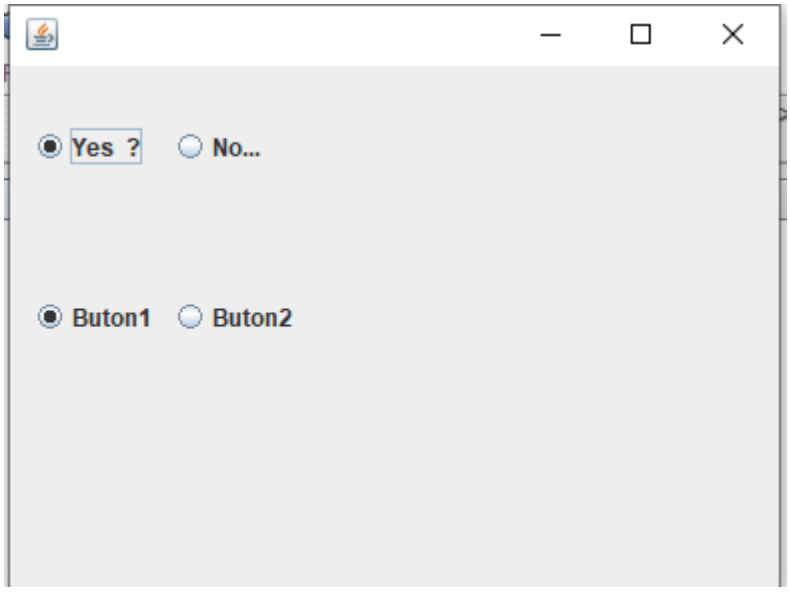
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class MainMenuBar_1 extends JMenuBar{
   JFrame frmParent;
   public MainMenuBar_1() {
      JMenu fileMenu = new JMenu("File");
      JMenu editMenu = new JMenu("Edit");
      JMenu quitMenu = new JMenu("Quit");
      // Meniul File: Open, Save, Close
      JMenuItem openItem = new JMenuItem("Open");
      JMenuItem saveItem = new JMenuItem("Save");
      JMenuItem closeItem = new JMenuItem("Close");
      //Meniul Edit: Cut, Copy, Paste, Find (Find, Replace)
      JMenuItem cutItem = new JMenuItem("Cut");
      JMenuItem copyItem = new JMenuItem("Copy");
      JMenuItem pasteItem = new JMenuItem("Paste");
      JSeparator separator1 = new JSeparator();
      JMenu findMenu = new JMenu("Find");
      JMenuItem findItem = new JMenuItem("Find");
      JMenuItem replaceItem = new JMenuItem("Replace");
      //Meniul Quit
      JMenuItem exitItem = new JMenuItem("Exit");
      JMenuItem aboutItem = new JMenuItem("About");
      fileMenu.add(openItem);
      fileMenu.add(saveItem);
      fileMenu.add(closeItem);
```

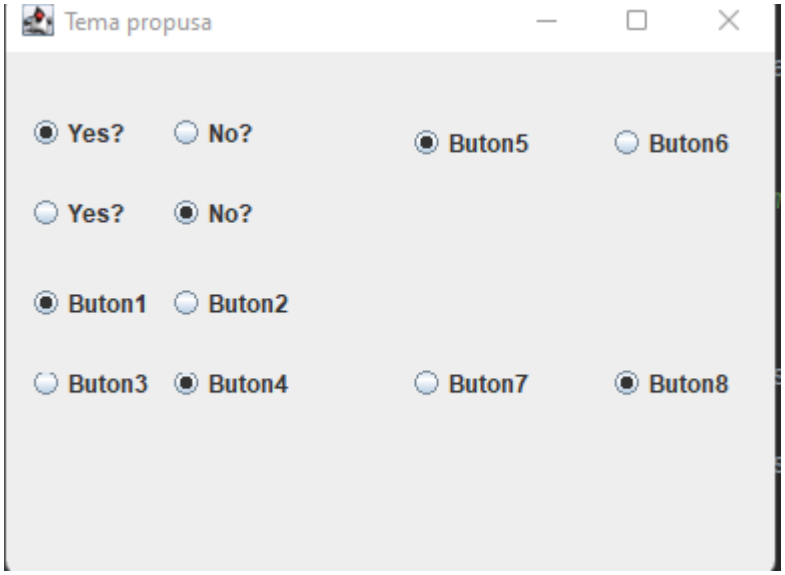
```
        editMenu.add(cutItem);
        editMenu.add(copyItem);
        editMenu.add(pasteItem);
        editMenu.add(separator1);
        findMenu.add(findItem);
        findMenu.add(replaceItem);
        editMenu.add(findMenu);
        editMenu.add(new JCheckBox("Bifa"));
        quitMenu.add(exitItem);
        quitMenu.add(aboutItem);
        add(fileMenu);
        add(editMenu);
        add(quitMenu);
    } }
public class Meniuri extends JFrame{
    public Meniuri() {
        this.setJMenuBar(new MainMenuBar_1());
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(500, 500);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new Meniuri();
    } }
```

```
setLayout(null);
```

Aplicatii de testat

```
import javax.swing.ButtonGroup;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
public class A1 extends JPanel {

   public A1()
        {
           JRadioButton r1_Yes = new JRadioButton("Yes  ?",true);
           JRadioButton r1_No = new JRadioButton("No  ?",false);


           ButtonGroup radioGroup1 = new ButtonGroup();
           ButtonGroup radioGroup2 = new ButtonGroup();
            setLayout(null);
           add(r1_Yes);
           add(r1_No);

           radioGroup1.add(r1_Yes);
           radioGroup1.add(r1_No);

           r1_Yes.setBounds(10,10,60,60);
           r1_No.setBounds(80,30,50,20);


           JRadioButton buton1 = new JRadioButton("Buton1",true);
           JRadioButton buton2 = new JRadioButton("Buton2",false);

           add(buton1);
           add(buton2);
           buton1.setBounds(10,90,70,70);
           buton2.setBounds(80,90,70,70);


        }

   public static void main(String[] args) {
      JFrame frame = new JFrame();
      frame.getContentPane().add(new A1());
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(400,300);
      frame.setVisible(true);
   }
   }
```


Yes ?
No...
Buton1
Buton2


Tema propusa
Yes?
No?
Buton5
Buton6
Yes?
No?
Buton1
Buton2
Buton3
Buton4
Buton7
Buton8

Rezolvare

```
import javax.swing.ButtonGroup;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JRadioButton;

public class Aplicatii extends JPanel {

  /**
   * @param args the command line arguments
   */
  public Aplicatii()
       {
          JRadioButton r1_Yes = new JRadioButton("Yes?",true);
          JRadioButton r1_No = new JRadioButton("No?",false);
          JRadioButton r2_Yes = new JRadioButton("Yes?",false);
          JRadioButton r2_No = new JRadioButton("No?",true);

          ButtonGroup radioGroup1 = new ButtonGroup();
          ButtonGroup radioGroup2 = new ButtonGroup();

          setLayout(null);

          add(r1_Yes);
          add(r1_No);
          add(r2_Yes);
          add(r2_No);

          radioGroup1.add(r1_Yes);
          radioGroup1.add(r1_No);
          radioGroup2.add(r2_Yes);
          radioGroup2.add(r2_No);

          r1_Yes.setBounds(10,10,60,60);
          r1_No.setBounds(80,30,50,20);

          r2_Yes.setBounds(10,50,60,60);
          r2_No.setBounds(80,70,50,20);


          JRadioButton buton1 = new JRadioButton("Buton1",true);
          JRadioButton buton2 = new JRadioButton("Buton2",false);
          JRadioButton buton3 = new JRadioButton("Buton3",false);
          JRadioButton buton4 = new JRadioButton("Buton4",true);

          ButtonGroup radioGroup3 = new ButtonGroup();
          ButtonGroup radioGroup4 = new ButtonGroup();

          add(buton1);
          add(buton2);
          add(buton3);
          add(buton4);

          radioGroup3.add(buton1);
```

```
        radioGroup3.add(buton2);
        radioGroup4.add(buton3);
        radioGroup4.add(buton4);

        buton1.setBounds(10,90,70,70);
        buton2.setBounds(80,90,70,70);

        buton3.setBounds(10, 130, 70, 70);
        buton4.setBounds(80, 130, 70, 70);


        JRadioButton buton5 = new JRadioButton("Buton5",true);
        JRadioButton buton6 = new JRadioButton("Buton6",false);
        JRadioButton buton7 = new JRadioButton("Buton7",false);
        JRadioButton buton8 = new JRadioButton("Buton8",true);

        ButtonGroup radioGroup5 = new ButtonGroup();
        ButtonGroup radioGroup6 = new ButtonGroup();

        add(buton5);
        add(buton6);
        add(buton7);
        add(buton8);

        radioGroup5.add(buton5);
        radioGroup6.add(buton6);
        radioGroup5.add(buton7);
        radioGroup6.add(buton8);

        buton5.setBounds(200,10,70,70);
        buton6.setBounds(300,10,70,70);

        buton7.setBounds(200, 130, 70, 70);
        buton8.setBounds(300, 130, 70, 70);
    }

  public static void main(String[] args) {
    JFrame frame = new JFrame();
    frame.getContentPane().add(new Aplicatii());
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400,300);
    frame.setVisible(true);
  }

}
```

Evenimente și ascultători

Evenimentele sunt generate de fiecare dată când apăsăm o tastă sau un buton al mouse-ului şi sunt tratate corespunzător prin cod implementat de programator.

Evenimentele au două pachete: java.awt.event se referă la evenimete generale și javax.swing.event se referă la evenimetele specifice componentelor grafice Swing.

Lista evenimetelor de bază din Swing, a ascultătorilor, a metodelor de adăugare, ştergere, respectiv componentele pentru care se aplică:

Evenimente, ascultatori si metodele de adaugare-stergere	Componentele care suportă aceste evenimente
ActionEvent ActionListener addActionListener() removeActionListener()	JButton, JList, JTextField, JMenuItem si componentele derivate inclusiv JCheckBoxMenuItem, JMenu si JPopupMenu.
AdjustmentEvent AdjustmentListener addAdjustmentListener() removeAdjustmentListener()	JScrollbar si orice implementare a interfetei Adjustable.
ComponentEvent ComponentListener addComponentListener() removeComponentListener()	Component si derivatele: JButton, JCheckBox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileDialog, JFrame, JLabel, JList, JScollbar, JTextArea si JTextField.
ContainerEvent ContainerListener addContainerListener() removeContainerListener()	Container si derivatele: JPanel, JApplet, JScrollPane, Window, JDialog, JfileDialog si JFrame.
FocusEvent FocusListener addFocusListener() removeFocusListener()	Componente si derivate.
KeyEvent KeyListener addKeyListener() removeKeyListener()	Component si derivate.
MouseEvent(folosit si pentru click si pentru miscare) MouseListener addMouseListener() removeMouseListener()	Component si derivate
MouseEvent(folosit si pentru click si pentru miscare) MouseMotionListener	Component si derivate

addMouseMotionListener() removeMouseMotionListener()	
WindowEvent WindowListener addWindowListener() removeWindowListener()	JWindow si derivatele incluzand JDiaolog, JFileDialog si JFrame.
ItemEvent ItemListener addItemListener() removeItemListener()	Window si derivatele incluzand JDiaolog, JComboBox, JList si toate implementarile interfetei ItemSelectable.
TextEvent TextListener addTextListener() removeTextListener()	Orice derivare din JTextComponent, incluzand JTextArea si JTextField.

Lista evenimentelor şi a metodelor corespunzătoare ce trebuie suprascrise:

Interfața ascultator/adaptor	Metodele interfeței
ActionListener	actionPerformed(ActionEvent)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
ComponentListener ComponentAdapter	componentHidden(ComponentEvent) componentShown(ComponentEvent) componentMoved(ComponentEvent) componentResized(ComponentEvent)
ContainerListener ContainerAdapter	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
FocusListener FocusAdapeter	focusGained(FocusEvent) focusLost(FocusEvent)
KeyListener KeyAdapter	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
MouseListener MouseAdapter	mouseClicked(MouseEvent) mouseEntered(MouseEvent)

	mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)
MouseMotionListener MouseMotionAdapter	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
WindowListener WindowAdapter	windowOpened(WindowEvent) windowClosing(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent)
ItemListener	itemStateChanged(ItemEvent)

In gestionarea evenimentelor intervin obiecte de tip ***Listener*** si ***Event***. Pentru a trata evenimentele ce apar intr-o aplicatie trebuie respectati pasii:

- Implementarea unei clase derivata dintr-o clasa Listener:

```
public class MyClass implements xxxListener {
      public void eveniment(xxxEvent e)
{ ...//tratarea evenimentului... }
  }
```

- Inregistrarea unei instante a clasei precedente pe post de *Listener* pentru o componenta derivata din ***JComponent***:

```
oComponenta.addActionListener(MyClass);
```

- Tratarea evenimentul de apasare pe un buton se va efectua utiliza clasa ***ActionListener***:

```
class ListenerPtButon implements ActionListener {
   private int numClicks = 0;

   public void actionPerformed(ActionEvent e){

      ((JButton)e.getSource()).setText("Apasat: de x ori“);
   }
}

JButton butonulMeu = new JButton(“Apasa: 0”);
butonulMeu. addActionListener(new ListenerPtButon());
```

---

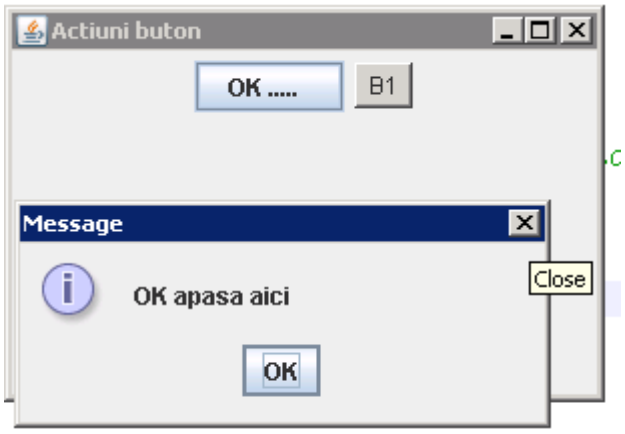
https://docs.oracle.com/javase/tutorial/uiswing/events/

JOptionPane contine trei metode, după cum urmează:

- **showMessageDialog ()** care este folosit pentru a afişa un mesaj simplu.
- **showInputDialog ()** care este folosita pentru a afişa un prompt. Această metodă returnează o valoare şir care este scris de utilizator.
- **showConfirmDialog ()** care solicită ghidul de confirmare (Da / Nu), prin afişarea mesajului. Această metodă returnează o valoare numerică fie 0 sau 1. Dacă faceţi clic pe butonul "Yes", apoi metoda returneaza 1 altfel 0

Exemplu 1

```
import java.awt.Button;
import javax.swing.*;
import java.awt.event.*;
class ButtonOKListener implements ActionListener {
   public void actionPerformed(ActionEvent e) {
      JOptionPane.showMessageDialog(null, "OK apasa aici");
   } }
 public class Main extends JFrame {
   JButton buton;
   JPanel panel;
   ButtonOKListener ActiuneButon;
                public Main() {
      panel = new JPanel();
      ActiuneButon = new ButtonOKListener();
       buton = new JButton("OK .....  ");
      Button b1 = new Button("B1");
       buton.addActionListener(ActiuneButon);
       panel.add(buton);          panel.add(b1);
      this.add(panel);
   }
   public static void main(String[] args) {
      Main GUI = new Main();
      GUI.setTitle("Actiuni buton");
      GUI.setSize(300, 200);
      GUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      GUI.setVisible(true);
   }
}
```



Aplic 1

```
import javax.swing.*;
import java.awt.event.*;
public class OptionPane {
 public static void main(String[] args){
                JFrame frame = new JFrame("Input Dialog Box Frame");
                JButton button = new JButton("Show Input Dialog Box");
                button.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent ae){
                String str = JOptionPane.showInputDialog(null, "Enter some text : ",
"diverse", 1);
                if(str != null)
                JOptionPane.showMessageDialog(null, "You entered the text : " + str,
"diverse", 2);
                                else
                JOptionPane.showMessageDialog(null, "You pressed cancel button",
"diverse", 3);
                                }
                });
                JPanel panel = new JPanel();
                panel.add(button);
                frame.add(panel);
                frame.setSize(400, 400);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setVisible(true);
        }
}
```