Java

Curs 10

Danciu Gabriel
Ianuarie 2019

Meniuri Swing

- Bara de meniuri – meniul principal al aplicației
- Meniul standard – poate conține meniuri sau submeniuri
- Meniul popup – activat de click dreapta
- Toolbar – cu acces rapid la funcționalități (de obicei în paralele cu meniul standard)
- Acțiunea – tratarea evenimentului de menu-click sau scurtături

Clasele Swing pentru meniuri

Class	Description
JMenuBar	An object that holds the top-level menu for the application.
JMenu	A standard menu. A menu consists of one or more **JMenuItems**.
JMenuItem	An object that populates menus.
JCheckBoxMenuItem	A check box menu item.
JRadioButtonMenuItem	A radio button menu item
JSeparator	The visual separator between menu items.
JPopupMenu	A menu that is typically activated by right-clicking the mouse.

JMenuBar, JMenu, și JMenuItem

- JMenuBar este un container pentru elemente de tip meniu
  - Moștenește Jcomponent
  - Adăugarea unui meniu la JMenuBar se face prin *JMenu add(JMenu menu)*
  - Ștergerea unui meniu se face prin void *remove(Component menu)*
  - Adăugarea unui JMenuBar la un Jframe se face prin *void setJMenuBar(JMenuBar mb)*
- JMenu reprezintă un meniu ce poate conține JMenuItems
  - Adăugarea unui item la un meniu se face prin *JMenuItem add(JMenuItem item)*
  - Se poate adăuga și un JSeparator prin *void addSeparator( )*
  - Ștergerea unui meniu se face prin *void remove(JMenuItem menu)*
- JMenuItem reprezintă un item al unui meniu și derivă din AbstractButton
  - Suportă mai mulți constructori printre care *JMenuItem(String name, Icon image)*
  - Activarea/Dezactivarea unui item se face cu *void setEnabled(boolean enable)*

Un prim exemplu

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class Demo implements ActionListener {
    JLabel jlab;
    Demo(); // implementarea pe slide-ul urmator
    public void actionPerformed(ActionEvent ae) {
// Get the action command from the menu selection.
        String comStr = ae.getActionCommand();
// If user chooses Exit, then exit the program.
        if (comStr.equals("Exit")) {
            System.exit(0);
        }
// Otherwise, display the selection.
        jlab.setText(comStr + " Selected");
    }
    public static void main(String args[]) {
// Create the frame on the event dispatching thread.
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new Demo();
            }
        });
    }
}
```

Un prim exemplu - continuare

```
Demo() {
// Create a new JFrame container.
    JFrame jfrm = new JFrame("Menu Demo");
// Specify FlowLayout for the layout manager.
    jfrm.setLayout(new FlowLayout());
// Give the frame an initial size.
    jfrm.setSize(220, 200);
// Terminate the program when the user closes the application.
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// Create a label that will display the menu selection.
    jlab = new JLabel();
// Create the menu bar.
    JMenuBar jmb = new JMenuBar();
// Create the File menu.
    JMenu jmFile = new JMenu("File");
    JMenuItem jmiOpen = new JMenuItem("Open");
    JMenuItem jmiClose = new JMenuItem("Close");
    JMenuItem jmiSave = new JMenuItem("Save");
    JMenuItem jmiExit = new JMenuItem("Exit");
    jmFile.add(jmiOpen);
    jmFile.add(jmiClose);
    jmFile.add(jmiSave);
    jmFile.addSeparator();
    jmFile.add(jmiExit);
    jmb.add(jmFile);
// Create the Options menu.
    JMenu jmOptions = new JMenu("Options");
// Create the Colors submenu.
    JMenu jmColors = new JMenu("Colors");
    JMenuItem jmiRed = new JMenuItem("Red");
    JMenuItem jmiGreen = new JMenuItem("Green");
    JMenuItem jmiBlue = new JMenuItem("Blue");
    jmColors.add(jmiRed);
    jmColors.add(jmiGreen);
    jmColors.add(jmiBlue);
    jmOptions.add(jmColors);
// Create the Priority submenu.
    JMenu jmPriority = new JMenu("Priority");
    JMenuItem jmiHigh = new JMenuItem("High");
    JMenuItem jmiLow = new JMenuItem("Low");
    jmPriority.add(jmiHigh);
    jmPriority.add(jmiLow);
    jmOptions.add(jmPriority);
// Create the Reset menu item.
    JMenuItem jmiReset = new JMenuItem("Reset");
    jmOptions.addSeparator();
    jmOptions.add(jmiReset);
// Finally, add the entire options menu to
// the menu bar
    jmb.add(jmOptions);
// Create the Help menu.
    JMenu jmHelp = new JMenu("Help");
    JMenuItem jmiAbout = new JMenuItem("About");
    jmHelp.add(jmiAbout);
    jmb.add(jmHelp);
// Add action listeners for the menu items.
    jmiOpen.addActionListener(this);
    jmiClose.addActionListener(this);
    jmiSave.addActionListener(this);
    jmiExit.addActionListener(this);
    jmiRed.addActionListener(this);
    jmiGreen.addActionListener(this);
    jmiBlue.addActionListener(this);
    jmiHigh.addActionListener(this);
    jmiLow.addActionListener(this);
    jmiReset.addActionListener(this);
    jmiAbout.addActionListener(this);
// Add the label to the content pane.
    jfrm.add(jlab);
// Add the menu bar to the frame.
    jfrm.setJMenuBar(jmb);
// Display the frame.
    jfrm.setVisible(true);
  }
```

Mnemonice și Acceleratorare

Mnemonic – o tastă ce permite selectarea unui item dintr-un meniu

Un accelerator – o combinație de taste ce permite selectarea unui item fără să fie nevoie de activarea meniului

Pentru mnemonic se apelează funcția *void setMnemonic(int mnem)*

unde *mnem* este un mnemomic (o constantă definită în ava.awt.event.KeyEvent). Exemplu: KeyEvent.VK_F sau KeyEvent.VK_Z.

Pentru accelerator se apelează funcția *void setAccelerator(KeyStroke ks)*

Unde *ks* este o combinație de taste

Exemplu: *static KeyStroke getKeyStroke(char ch)*

unde *ch* este un caracter accelerator

sau *static KeyStroke getKeyStroke(int ch, int modifier)* cu *modifier* o constantă de tip KeyEvent, definită în java.awt.event.InputEvent

Exemplu – modificări la primul exemplu

```
JMenu jmFile = new JMenu("File");
jmFile.setMnemonic(KeyEvent.VK_F); //Alt + F
JMenuItem jmiOpen = new JMenuItem("Open",KeyEvent.VK_O);
jmiOpen.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,InputEvent.CTRL
_DOWN_MASK)); //Ctrl + O

JMenuItem jmiClose = new JMenuItem("Close",KeyEvent.VK_C);
jmiClose.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C,InputEvent.CTR
L_DOWN_MASK));

JMenuItem jmiSave = new JMenuItem("Save",KeyEvent.VK_S);
jmiSave.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,InputEvent.CTRL
_DOWN_MASK));

JMenuItem jmiExit = new JMenuItem("Exit",KeyEvent.VK_E);
jmiExit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_E,InputEvent.CTRL
_DOWN_MASK));
```

Adăugarea imaginilor și tooltip-urilor

- ImageIcon icon = new ImageIcon(“1.png”);
- JMenuItem jmiAbout = new JMenuItem("About", icon);

- jmiAbout.setToolTipText("Info about the MenuDemo program.");

JRadioButtonMenuItem și JCheckBoxMenuItem

- // Create the Options menu.
- JMenu jmOptions = new JMenu("Options");
- // Create the Colors submenu.
- JMenu jmColors = new JMenu("Colors");
- // Use check boxes for colors. This allows
- // the user to select more than one color.
- JCheckBoxMenuItem jmiRed = new JCheckBoxMenuItem("Red");
- JCheckBoxMenuItem jmiGreen = new JCheckBoxMenuItem("Green");
- JCheckBoxMenuItem jmiBlue = new JCheckBoxMenuItem("Blue");
- jmColors.add(jmiRed);
- jmColors.add(jmiGreen);
- jmColors.add(jmiBlue);
- jmOptions.add(jmColors);
- // Create the Priority submenu.
- JMenu jmPriority = new JMenu("Priority");
- JRadioButtonMenuItem jmiHigh =new JRadioButtonMenuItem("High", true);
- JRadioButtonMenuItem jmiLow =new JRadioButtonMenuItem("Low");
- jmPriority.add(jmiHigh);
- jmPriority.add(jmiLow);
- jmOptions.add(jmPriority);
- // Create button group for the radio button menu items.
- ButtonGroup bg = new ButtonGroup();
- bg.add(jmiHigh);
- bg.add(jmiLow);
- // Create the Reset menu item.
- JMenuItem jmiReset = new JMenuItem("Reset");
- jmOptions.addSeparator();
- jmOptions.add(jmiReset);
- // Finally, add the entire options menu to the menu bar
- jmb.add(jmOptions);

Meniu Popup

- Activarea unui meniu de tip popup se va face în trei pași
  - Inregistarea unui listener la evenimente de tip mouse
  - In event handler, se va scrie cod pentru trigger de tip popup
  - Când este declanșat acel trigger, se afișează popup-ul prin *show()*

Exemplu - Popup

```
JPopupMenu jpu; //membru al clasei
        // Codul de mai jos se adauga constructorului.
        jpu = new JPopupMenu();
// Create the popup menu items.
        JMenuItem jmiCut = new JMenuItem("Cut");
        JMenuItem jmiCopy = new JMenuItem("Copy");
        JMenuItem jmiPaste = new JMenuItem("Paste");
        jmiCut.addActionListener(this);
        jmiCopy.addActionListener(this);
        jmiPaste.addActionListener(this);
// Add the menu items to the popup menu.
        jpu.add(jmiCut);
        jpu.add(jmiCopy);
        jpu.add(jmiPaste);
        // Add a listener for the popup trigger.
        jfrm.addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent me) {
                if (me.isPopupTrigger()) {
                    jpu.show(me.getComponent(), me.getX(), me.getY());
                }
            }
            public void mouseReleased(MouseEvent me) {
                if (me.isPopupTrigger()) {
                    jpu.show(me.getComponent(), me.getX(), me.getY());
                }
            }
        });
        jfrm.setVisible(true);
```

ToolBar

```
 // Create a Debug toolbar.
        JToolBar jtb = new JToolBar("Debug");
// Load the images.
        ImageIcon set = new ImageIcon("set.png");
        ImageIcon clear = new ImageIcon("delete.png");
        ImageIcon resume = new ImageIcon("resume.png");
// Create the toolbar buttons.
        JButton jbtnSet = new JButton(set);
        jbtnSet.setActionCommand("Set Breakpoint");
        jbtnSet.setToolTipText("Set Breakpoint");
        JButton jbtnClear = new JButton(clear);
        jbtnClear.setActionCommand("Clear Breakpoint");
        jbtnClear.setToolTipText("Clear Breakpoint");
        JButton jbtnResume = new JButton(resume);
        jbtnResume.setActionCommand("Resume");
        jbtnResume.setToolTipText("Resume");
        jbtnSet.addActionListener(this);
        jbtnClear.addActionListener(this);
        jbtnResume.addActionListener(this);
// Add the buttons to the toolbar.
        jtb.add(jbtnSet);
        jtb.add(jbtnClear);
        jtb.add(jbtnResume);
// Add the toolbar to the north position of
// the content pane.
        jfrm.add(jtb, BorderLayout.NORTH);
```

JDialog

- Extinde java.awt.Dialog
- Pot fi modale sau non-modale
- Constructori:
  - JDialog()
  - JDialog(Frame o)
  - JDialog(Frame o, String s)
  - JDialog(Window o)
  - JDialog(Window o, String t)
  - JDialog(Dialog o)
  - JDialog(Dialog o, String s)

Exemplu

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

class Demo extends JFrame implements ActionListener {

    // frame
    static JFrame f;
    // main class
    public static void main(String[] args) {
        // create a new frame
        f = new JFrame("frame");

        // create a object
        Demo s = new Demo();

        // create a panel
        JPanel p = new JPanel();
        JButton b = new JButton("click");
        // add actionlistener to button
        b.addActionListener(s);
        // add button to panel
        p.add(b);
        f.add(p);

        // set the size of frame
        f.setSize(400, 400);
        f.show();
    }
    public void actionPerformed(ActionEvent e) {
        String s = e.getActionCommand();
        if (s.equals("click")) {
            // create a dialog Box
            JDialog d = new JDialog(f, "dialog Box");

            // create a label
            JLabel l = new JLabel("this is a dialog box");
            d.add(l);
            // setsize of dialog
            d.setSize(100, 100);

            // set visibility of dialog
            d.setVisible(true);
        }
    }
}
```

Exemplu input – partea 1

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

public class Demo extends JFrame {

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                Demo form = new Demo();
                form.setVisible(true);
            }
        });
    }
    public Demo() {

        // Create Form Frame
        super("Dialog Demo");
        setSize(500, 300);
        setLocation(500, 280);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);

        // Label Result
        final JLabel lblResult = new JLabel("Result", JLabel.CENTER);
        lblResult.setBounds(40, 60, 400, 20);
        getContentPane().add(lblResult);

        // Create Button Open Dialog
        JButton btnButton = new JButton("Submit");
        btnButton.setBounds(200, 100, 100, 30);
        btnButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                DialogPopup dialog = new DialogPopup();
                dialog.setModal(true);
                dialog.setVisible(true);

                lblResult.setText(dialog.sName);

            }
        });
        getContentPane().add(btnButton);
    }
}
```

Exemplu input – partea 2

```
class DialogPopup extends JDialog {

    public String sName;

    public DialogPopup() {
        setBounds(100, 100, 300, 200);
        setTitle("Input Dialog");
        setLocationRelativeTo(null);
        getContentPane().setLayout(null);

        // Create Input
        final JTextField name = new JTextField();
        name.setBounds(60, 40, 200, 20);
        getContentPane().add(name);

        // Button OK
        JButton btnOK = new JButton("OK");
        btnOK.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                sName = name.getText();
                dispose();
            }
        });
        btnOK.setBounds(100, 100, 100, 30);
        getContentPane().add(btnOK);

        // Button Cancel
        JButton btnCancel = new JButton("Cancel");
        btnCancel.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e) {
                sName = "";
                dispose();
            }
        });
        btnCancel.setBounds(200, 100, 100, 30);
        getContentPane().add(btnCancel);
    }
}
```

JFileChooser

```
package demo;
import javax.swing.*;
import javax.swing.filechooser.*;
import java.io.File;

public class Demo {
        public static void main(String[] args) {
                JFileChooser jfc = new
JFileChooser(FileSystemView.getFileSystemView().getHomeDirectory());
                int returnValue = jfc.showOpenDialog(null);
                if (returnValue == JFileChooser.APPROVE_OPTION) {
                        File selectedFile = jfc.getSelectedFile();
                        System.out.println(selectedFile.getAbsolutePath());
                }
        }
}
```

Open/Save Dialog

```
package demo;


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


public class Demo extends JFrame {
  private JTextField filename = new JTextField(), dir =
new JTextField();
  private JButton open = new JButton("Open"), save = new
JButton("Save");
  public Demo() {
    JPanel p = new JPanel();
    open.addActionListener(new OpenL());
    p.add(open);
    save.addActionListener(new SaveL());
    p.add(save);
    Container cp = getContentPane();
    cp.add(p, BorderLayout.SOUTH);
    dir.setEditable(false);
    filename.setEditable(false);
    p = new JPanel();
    p.setLayout(new GridLayout(2, 1));
    p.add(filename);
    p.add(dir);
    cp.add(p, BorderLayout.NORTH);
  }
```

```
class OpenL implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    JFileChooser c = new JFileChooser();
    // Demonstrate "Open" dialog:
    int rVal = c.showOpenDialog(Demo.this);
    if (rVal == JFileChooser.APPROVE_OPTION) {
      filename.setText(c.getSelectedFile().getName());
      dir.setText(c.getCurrentDirectory().toString());
    }
    if (rVal == JFileChooser.CANCEL_OPTION) {
      filename.setText("You pressed cancel");
      dir.setText("");
    }
  }
}


class SaveL implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    JFileChooser c = new JFileChooser();
    // Demonstrate "Save" dialog:
    int rVal = c.showSaveDialog(Demo.this);
    if (rVal == JFileChooser.APPROVE_OPTION) {
      filename.setText(c.getSelectedFile().getName());
      dir.setText(c.getCurrentDirectory().toString());
    }
    if (rVal == JFileChooser.CANCEL_OPTION) {
      filename.setText("You pressed cancel");
      dir.setText("");
    }
  }
}
```

```
  public static void main(String[] args) {
    run(new Demo(), 400, 200);
  }

  public static void run(JFrame frame, int width, int
height) {
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(width, height);
    frame.setVisible(true);
  }
}
```

Exerciții

- Se dă fișierul obiecte.csv cu următoarea structură:
  - Nume, culoare, cantitate
- Creați o aplicație ce permite:
  - Selectarea (Open) fișierului (se pot numi 1.csv, 2.csv etc). Fișierele trebuie să respecte structura de mai sus.
  - Încărcarea datelor pe o formă (1 JList)
  - La selectarea unui obiect acesta poate fi editat (3 JTextField)
  - Adăugarea unui obiect
  - Ștergerea unui obiect
  - Salvarea modificărilor în fișierul inițial sau în altul nou (Save și Save As)
  - Folosiți toate tipurile de meniuri din acest curs