Structuri de Date în Java (X)

- Buttons. Labels. Checkboxes. Combo boxes
- Spinner control: list, numeric, calendar
- Borders
- Menus
- PopupMenu. JScrollPane. JSplitPane
- JTabbedPane
- Scrollbar. Slider
- Dialogs

JButton

- Butonul generează un ActionEvent
- Programul înregistrează un ActionListener care
    - suprascrie actionPerformed ()

- Pentru orice componentă care generează o acțiune, Java permite definirea unui **action command** (de tip String)
- Pentru un buton, implicit, acest **action command** este chiar eticheta sa

```
JButton btn = ... ;
btn.setActionCommand (“agree”);
```

```
public void actionPerformed(ActionEvent e){
   if (e.getActionCommand( ).equals("Yes") {
                          // ...
                          }
                       }
```

Check boxes and radio buttons

UsingSwing 1 (a)

```
class UsingSwing1 {
    JFrame frame;
    JPanel sandwich;
    ButtonGroup group;    // folosit doar pentru activarea unică a radio buttons
    JPanel topping;
    JButton orderBtn;

    public UsingSwing1 () {
        frame = new JFrame ("DriveIn Menu");
        sandwich = new JPanel ();                // primul panel
        group = new ButtonGroup ();

        JRadioButton radioBtn;
        sandwich.add (radioBtn = new JRadioButton ("Chicken"));
        radioBtn.setActionCommand ("chicken");
        group.add (radioBtn);
        sandwich.add (radioBtn = new JRadioButton ("Beef"));
        radioBtn.setActionCommand ("beef");
        group.add (radioBtn);
        sandwich.add (radioBtn = new JRadioButton ("Colliflower"));
        radioBtn.setActionCommand ("colliflower");
        group.add (radioBtn);

        // ...
```

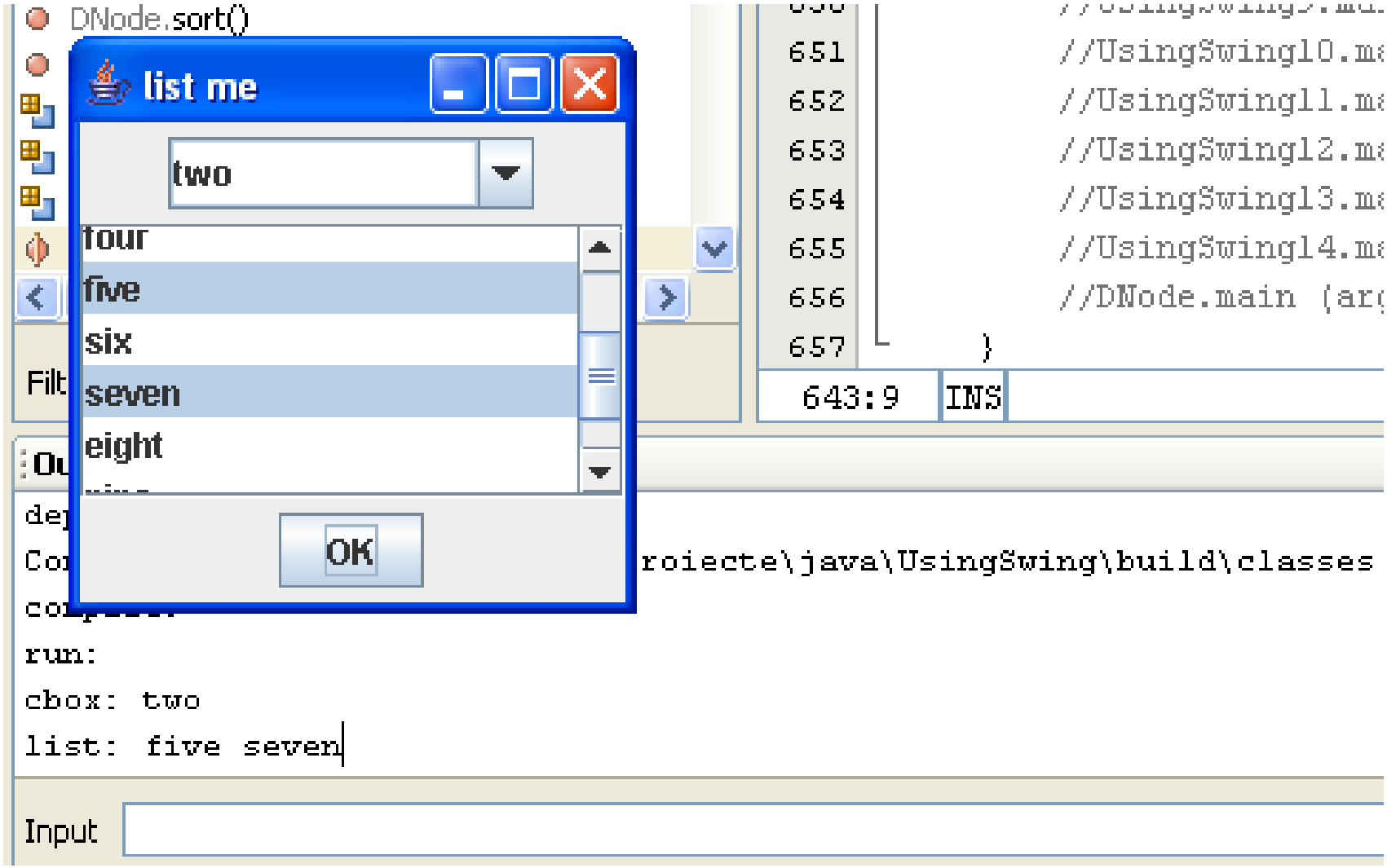
UsingSwing 1 (b)

```
    topping = new JPanel ();            // al doilea panel
    JCheckBox cb;
    topping.add (cb = new JCheckBox ("Ketchup"));
    cb.setActionCommand ("ketchup");
    topping.add (cb = new JCheckBox ("Mayonnaise"));
    cb.setActionCommand ("mayonnaise");
    topping.add (cb = new JCheckBox ("Jogurt"));
    cb.setActionCommand ("jogurt");

    JPanel order = new JPanel ();       // al treilea panel
    orderBtn = new JButton ("Place order");
    order.add (orderBtn);

    Container pane = frame.getContentPane ();
    pane.setLayout (new GridLayout (3, 1));
    pane.add (sandwich);
    pane.add (topping);
    pane.add (order);

    putListener ();

    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setSize (300, 150);
    frame.setVisible (true);
}
```

UsingSwing 1 (c)

```
void putListener () {
    orderBtn.addActionListener (new ActionListener () {
        public void actionPerformed (ActionEvent e) {
            String res = "sandwich with ";
            res += group.getSelection ().getActionCommand ();
            for (int i = 0 ; i < topping.getComponentCount () ; i ++) {
                JCheckBox cb = (JCheckBox)topping.getComponent (i);
                if (cb.isSelected ())
                    res += " and " + cb.getActionCommand ();
            }
            System.out.println (res);
        }
    });
}
```

Lists and combo boxes



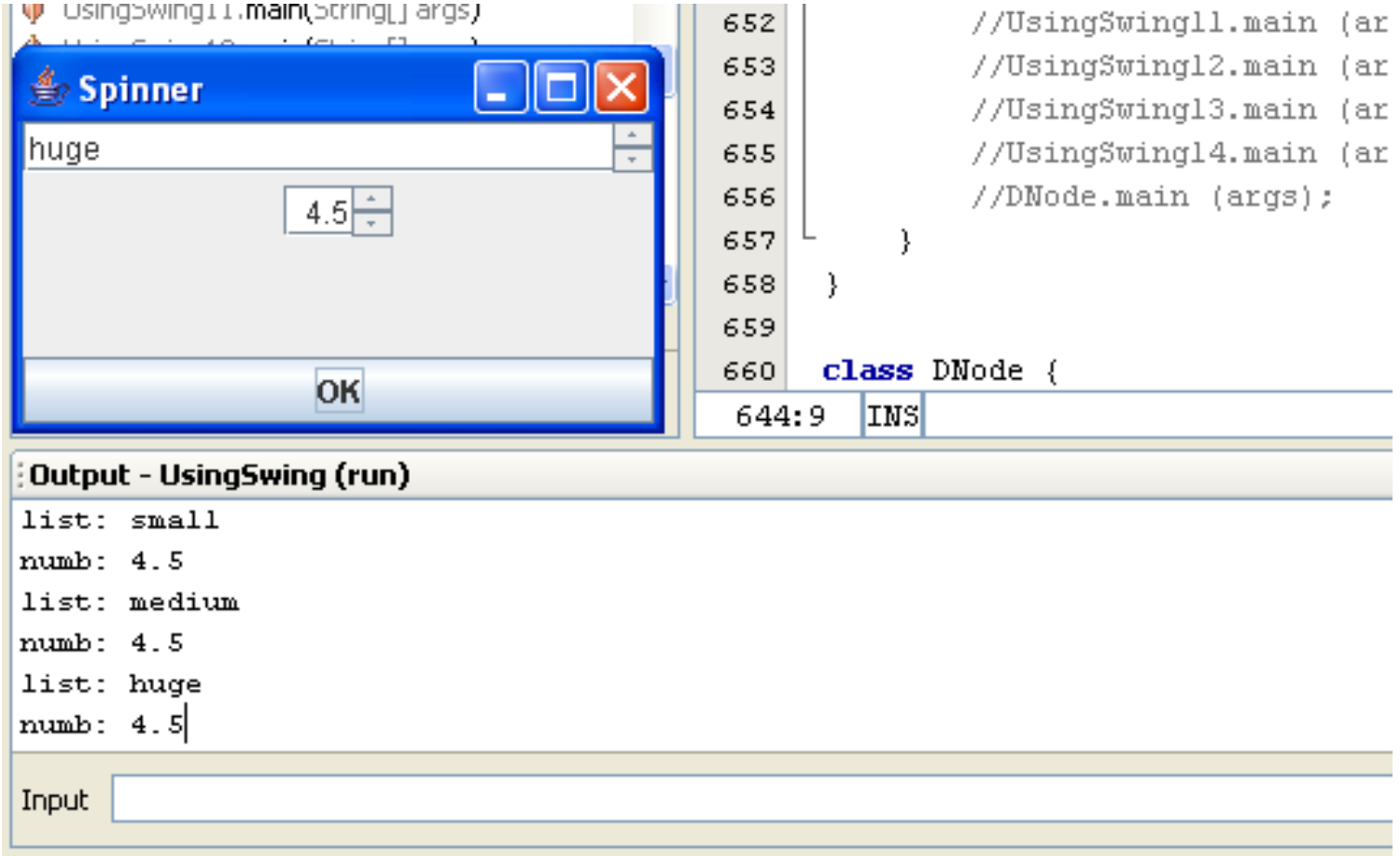
UsingSwing 2 (a)

```
class UsingSwing2 {
    final String[] items = { "one", "two", "three", "four",
            "five", "six", "seven", "eight", "nine", "ten"};
    JFrame frame;
    JComboBox cbox;
    JList list;
    JButton btn;

    public UsingSwing2 () {
        frame = new JFrame ("list me");
        cbox = new JComboBox (items);
        cbox.setEditable (true);
        list = new JList (items);
        btn = new JButton ("OK");

        Container pane = frame.getContentPane ();
        JPanel cpane = new JPanel ();
        cpane.add (cbox);
        pane.add (cpane, BorderLayout.NORTH);

        pane.add (new JScrollPane (list), BorderLayout.CENTER);

        cpane = new JPanel ();
        cpane.add (btn);
        pane.add (cpane, BorderLayout.SOUTH);
```

UsingSwing 2 (b)

```
        putListener ();

        frame.setSize (200, 200);
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.setVisible (true);
    }

    void putListener () {
        btn.addActionListener (new ActionListener () {
           public void actionPerformed (ActionEvent e) {
               String res = "cbox: " + cbox.getSelectedItem ();
               res += "\nlist:";
               Object[] sel = list.getSelectedValues ();

               for (int i = 0 ; i < sel.length ; i ++)
                   res += " " + sel[i];

               System.out.println (res);
           }
        });
    }
}
```

Spinner Component



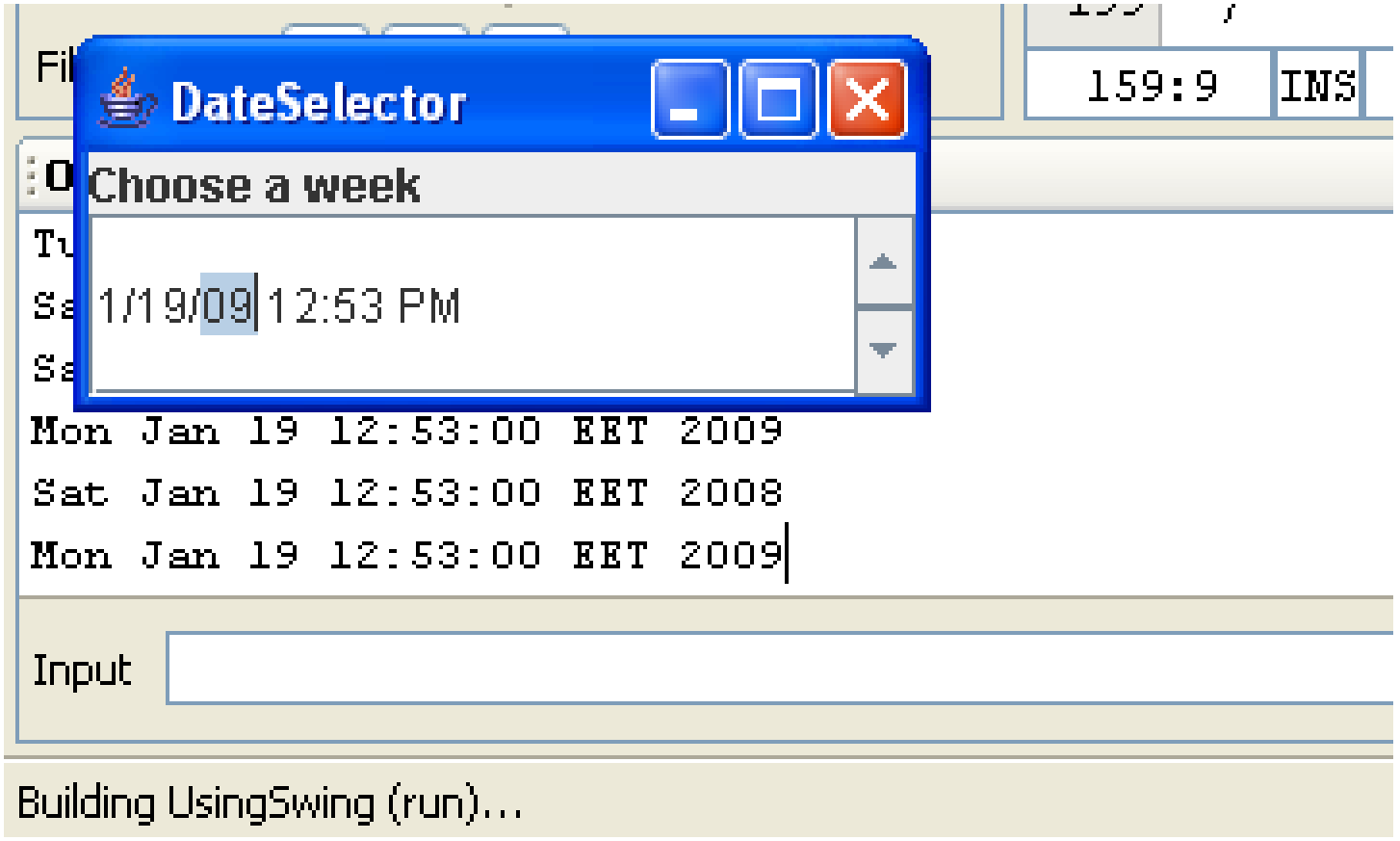
UsingSwing 3 (a)

```
class UsingSwing3 {
    JFrame frame;
    JSpinner listSpin;
    SpinnerListModel listModel;
    JSpinner numbSpin;
    SpinnerNumberModel numbModel;
    JButton btn;
    public UsingSwing3 () {
        frame = new JFrame ("Spinner");

        String[] opts = new String[] { "small", "medium", "large", "huge" };
        listModel = new SpinnerListModel (opts);
        listSpin = new JSpinner (listModel);

        double initial = 5.0, min = 0.0, max = 10.0, increment = 0.1;
        numbModel = new SpinnerNumberModel (initial, min, max, increment);
        numbSpin = new JSpinner (numbModel);

        btn = new JButton ("OK");
```

UsingSwing 3 (b)

```
        Container pane = frame.getContentPane ();
        pane.add (listSpin, BorderLayout.NORTH);
        JPanel cpane = new JPanel ();
        cpane.add (numbSpin);
        pane.add (cpane, BorderLayout.CENTER);
        pane.add (btn, BorderLayout.SOUTH);

        putListener ();

        frame.pack ();
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.setVisible (true);
    }

    void putListener () {
        btn.addActionListener (new ActionListener () {
           public void actionPerformed (ActionEvent e) {
               String res = "list: " + (String)listModel.getValue ();
               res += "\nnumb: " + numbModel.getNumber ();
               System.out.println (res);
           }
        });

    }
}
```

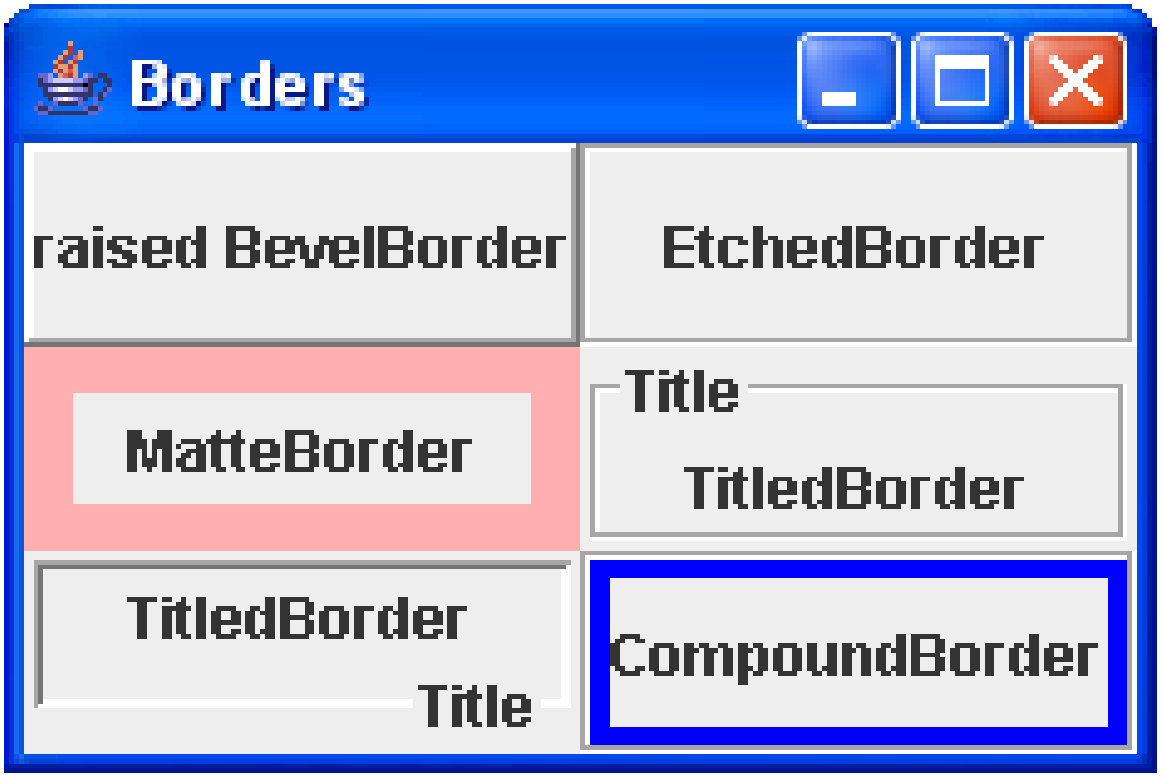
Calendar model



UsingSwing 4

```
class UsingSwing4 {
  public static void main (String[] args)
  {
    JFrame frame = new JFrame ("DateSelector");

    Calendar now = Calendar.getInstance ();
    Calendar earliest = (Calendar)now.clone ();
    earliest.add (Calendar.MONTH, -12);
    Calendar latest = (Calendar)now.clone ();
    latest.add (Calendar.MONTH, 12);
    SpinnerModel model = new SpinnerDateModel (
        now.getTime (), earliest.getTime (), latest.getTime (),
        Calendar.WEEK_OF_YEAR);
    final JSpinner spinner = new JSpinner (model);

    model.addChangeListener (new ChangeListener () {
        public void stateChanged (ChangeEvent e) {
            System.out.println (((SpinnerDateModel)e.getSource ()).getDate ());
        }
    } );

    frame.getContentPane ().add ("North", new JLabel ("Choose a week"));
    frame.getContentPane ().add ("Center", spinner);
    frame.pack ();
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setVisible (true);
  }
}
```

Borders


Borders
raised BevelBorder
EtchedBorder
MatteBorder
Title
TitledBorder
TitledBorder
Title
CompoundBorder

Implementările claselor Border

- BevelBorder (iluzie de adâncime)
- SoftBevelBorder (idem, dar mai fină)
- EmptyBorder (nu e vizibilă, ocupă doar spațiu)
- EtchedBorder (gravată)
- LineBorder (linie simplă)
- MatteBorder (colorată)
- TitledBorder (specifică un titlu)
- CompoundBorder (permite compunerea a două)

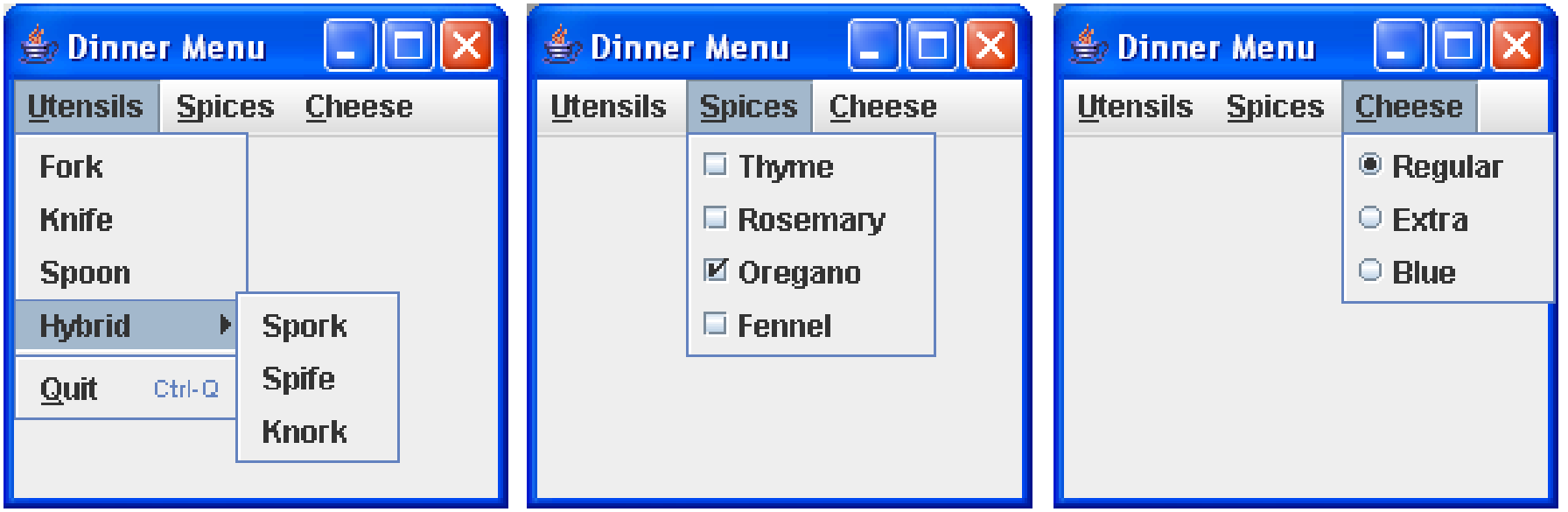
UsingSwing 5 (a)

```
class UsingSwing5 {
  public static void main (String[] args) {
    JFrame frame = new JFrame ("Borders");

    // Create labels with borders.
    int center = SwingConstants.CENTER;
    JLabel labelOne = new JLabel ("raised BevelBorder", center);
    labelOne.setBorder (
        BorderFactory.createBevelBorder (BevelBorder.RAISED));

    JLabel labelTwo = new JLabel ("EtchedBorder", center);
    labelTwo.setBorder (BorderFactory.createEtchedBorder ());

    JLabel labelThree = new JLabel ("MatteBorder", center);
    labelThree.setBorder (
        BorderFactory.createMatteBorder (10, 10, 10, 10, Color.pink));

    JLabel labelFour = new JLabel ("TitledBorder", center);
    Border etch = BorderFactory.createEtchedBorder ();
    labelFour.setBorder (
        BorderFactory.createTitledBorder (etch, "Title"));
```

UsingSwing 5 (b)

```
    JLabel labelFive = new JLabel ("TitledBorder", center);
    Border low = BorderFactory.createLoweredBevelBorder ();
    labelFive.setBorder (
        BorderFactory.createTitledBorder (low, "Title",
        TitledBorder.RIGHT, TitledBorder.BOTTOM));

    JLabel labelSix = new JLabel ("CompoundBorder", center);
    Border one = BorderFactory.createEtchedBorder ();
    Border two =
        BorderFactory.createMatteBorder (4, 4, 4, 4, Color.blue);
    labelSix.setBorder (BorderFactory.createCompoundBorder (one, two));

    // add components to the content pane
    Container c = frame.getContentPane ();
    c.setLayout (new GridLayout(3, 2));
    c.add (labelOne);
    c.add (labelTwo);
    c.add (labelThree);
    c.add (labelFour);
    c.add (labelFive);
    c.add (labelSix);

    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.pack ();
    frame.setVisible (true);
  }
}
```

Meniuri


Dinner Menu
Utensils
Spices
Cheese
Fork
Knife
Spoon
Hybrid
Quit
Ctrl-Q
Spork
Spife
Knork
Dinner Menu
Utensils
Spices
Cheese
Thyme
Rosemary
Oregano
Fennel
Dinner Menu
Utensils
Spices
Cheese
Regular
Extra
Blue

UsingSwing 6 (a)

```
class UsingSwing6
{
  public static void main (String[] args) {
    JFrame frame = new JFrame("Dinner Menu");

    // utensils menu
    JMenu utensils = new JMenu ("Utensils");
    utensils.setMnemonic (KeyEvent.VK_U);
    utensils.add (new JMenuItem ("Fork"));
    utensils.add (new JMenuItem ("Knife"));
    utensils.add (new JMenuItem ("Spoon"));

    // hybrid menu item
    JMenu hybrid = new JMenu ("Hybrid");
    hybrid.add (new JMenuItem ("Spork"));
    hybrid.add (new JMenuItem ("Spife"));
    hybrid.add (new JMenuItem ("Knork"));
    utensils.add (hybrid);
    utensils.addSeparator ();
```

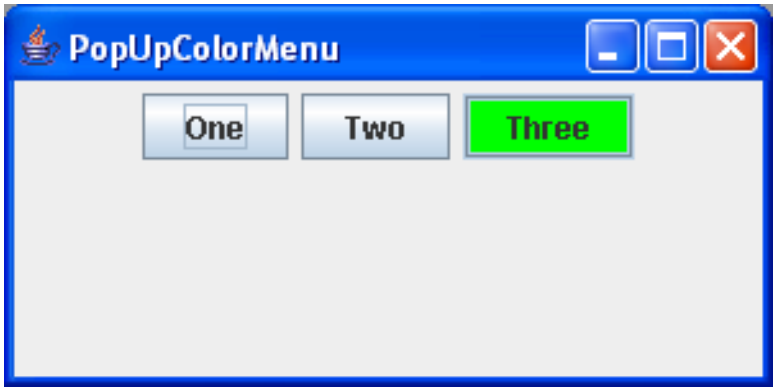
UsingSwing 6 (b)

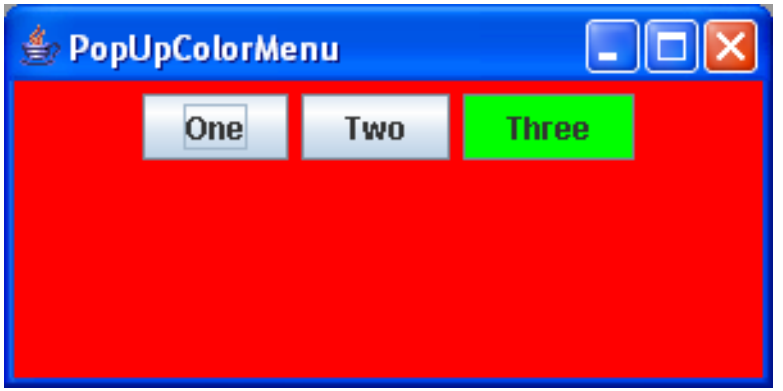
```
// quit shortcut
JMenuItem quitItem = new JMenuItem ("Quit");
quitItem.setMnemonic (KeyEvent.VK_Q);
quitItem.setAccelerator (
    KeyStroke.getKeyStroke (KeyEvent.VK_Q, Event.CTRL_MASK));
quitItem.addActionListener (new ActionListener () {
  public void actionPerformed (ActionEvent e) { System.exit(0); }
});
utensils.add (quitItem);

// spices menu
JMenu spices = new JMenu ("Spices");
spices.setMnemonic (KeyEvent.VK_S);
spices.add (new JCheckBoxMenuItem ("Thyme"));
spices.add (new JCheckBoxMenuItem ("Rosemary"));
spices.add (new JCheckBoxMenuItem ("Oregano", true));
spices.add (new JCheckBoxMenuItem ("Fennel"));
```

UsingSwing 6 (c)

```
    // cheese menu
    JMenu cheese = new JMenu ("Cheese");
    cheese.setMnemonic (KeyEvent.VK_C);
    ButtonGroup group = new ButtonGroup ();
    JRadioButtonMenuItem rbmi;
    rbmi = new JRadioButtonMenuItem ("Regular", true);
    group.add (rbmi);
    cheese.add (rbmi);
    rbmi = new JRadioButtonMenuItem ("Extra");
    group.add (rbmi);
    cheese.add (rbmi);
    rbmi = new JRadioButtonMenuItem ("Blue");
    group.add (rbmi);
    cheese.add (rbmi);

    // create menu bar
    JMenuBar menuBar = new JMenuBar ();
    menuBar.add (utensils);
    menuBar.add (spices);
    menuBar.add (cheese);
    frame.setJMenuBar (menuBar);

    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setSize (200, 200);
    frame.setVisible (true);
  }
}
```

PopMenu component





UsingSwing 7 (a)

```
class UsingSwing7 implements ActionListener
{
  Component selectedComponent;

  public UsingSwing7 () {
    JFrame frame = new JFrame ("PopUpColorMenu");

    final JPopupMenu colorMenu = new JPopupMenu ("Color");
    colorMenu.add (makeMenuItem ("Red"));
    colorMenu.add (makeMenuItem ("Green"));
    colorMenu.add (makeMenuItem ("Blue"));

    MouseListener mouseListener = new MouseAdapter () {
      public void mousePressed (MouseEvent e) { checkPopup (e); }
      public void mouseClicked (MouseEvent e) { checkPopup (e); }
      public void mouseReleased (MouseEvent e) { checkPopup (e); }
      private void checkPopup (MouseEvent e) {
        if (e.isPopupTrigger ()) {
          selectedComponent = e.getComponent ();
          colorMenu.show (e.getComponent (), e.getX (), e.getY ());
        }
      }
    };

    Container content = frame.getContentPane ();
    content.setLayout (new FlowLayout ());
```

UsingSwing 7 (b)

```
  JButton button = new JButton ("One");
  button.addMouseListener (mouseListener);
  content.add (button);

  button = new JButton ("Two");
  button.addMouseListener (mouseListener);
  content.add (button);

  button = new JButton ("Three");
  button.addMouseListener (mouseListener);
  content.add (button);

  frame.getContentPane ().addMouseListener (mouseListener);

  frame.setSize (300, 150);
  frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
  frame.setVisible (true);
}
```

UsingSwing 7 (c)

```
  public void actionPerformed (ActionEvent e) {
    String color = e.getActionCommand ();
    if (color.equals ("Red"))
      selectedComponent.setBackground (Color.red);
    else if (color.equals ("Green"))
      selectedComponent.setBackground (Color.green);
    else if (color.equals ("Blue"))
      selectedComponent.setBackground (Color.blue);
  }

  private JMenuItem makeMenuItem (String label) {
    JMenuItem item = new JMenuItem (label);
    item.addActionListener (this);
    return item;
  }

  public static void main (String[] args) {
     new UsingSwing7 ();
  }
}
```

JScrollPane component

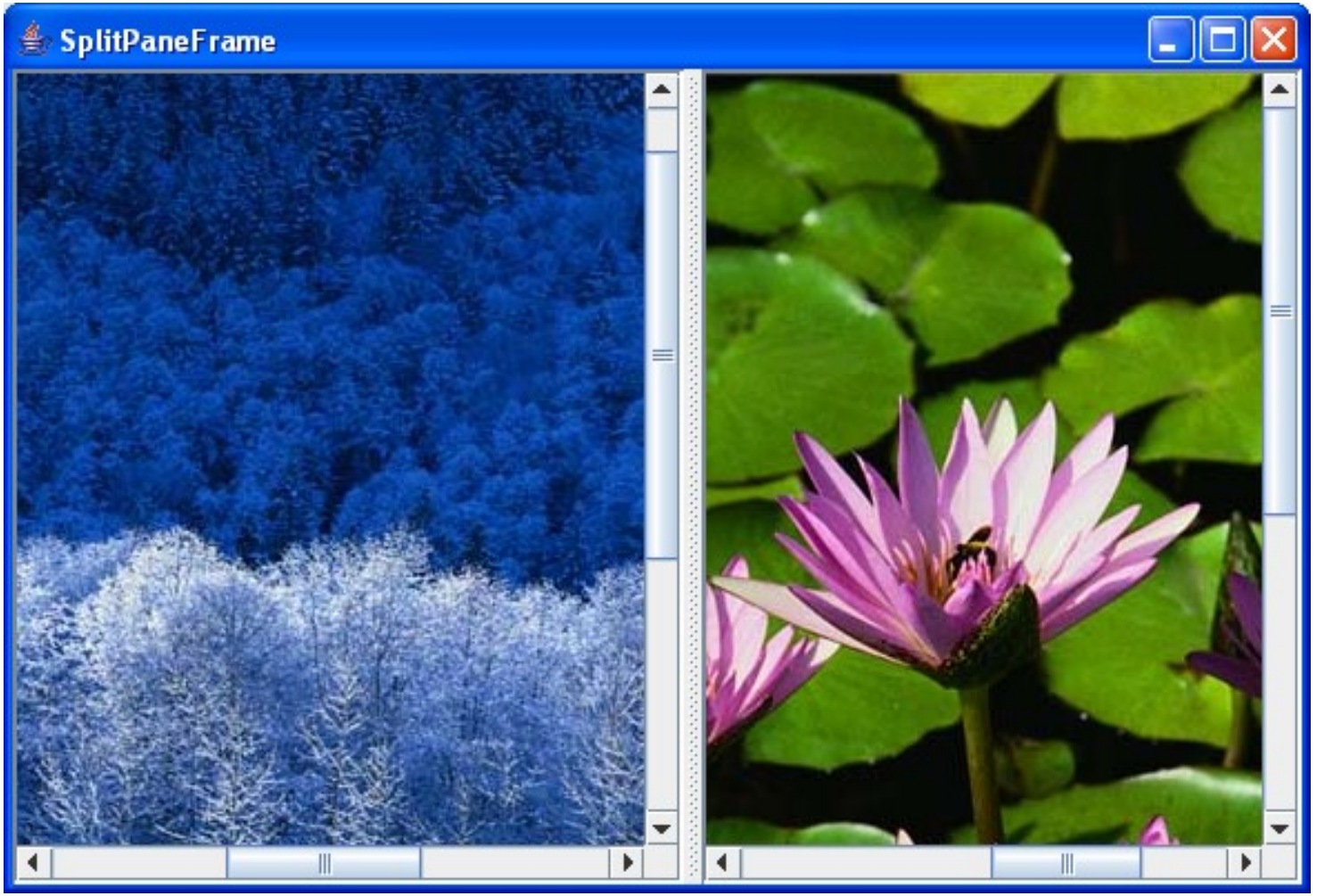
JScrollPane - Container special

- Derulează altă componentă
- Are un layout manager propriu, nemodificabil
  - Conține doar o singură componentă
- Policy-ul (când se afişează scroll bar-urile):
  - Doar la nepotrivire:
    HORIZONTAL_SCROLLBAR_AS_NEEDED
    VERTICAL_SCROLLBAR_AS_NEEDED
  - Mereu:
    HORIZONTAL_SCROLLBAR_ALLWAYS
    VERTICAL_SCROLLBAR_ALLWAYS
  - Niciodată:
    HORIZONTAL_SCROLLBAR_NEVER
    VERTICAL_SCROLLBAR_NEVER

UsingSwing 8

```
class UsingSwing8
{
  public static void main (String[] args) {
    String filename = "winter.jpg";
    if (args.length > 0)
      filename = args[0];

    JFrame frame = new JFrame ("ScrollPaneFrame");
    JLabel image = new JLabel (new ImageIcon(filename));
    frame.getContentPane ().add (new JScrollPane(image));

    frame.setSize (300, 300);
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setVisible (true);
  }
}
```

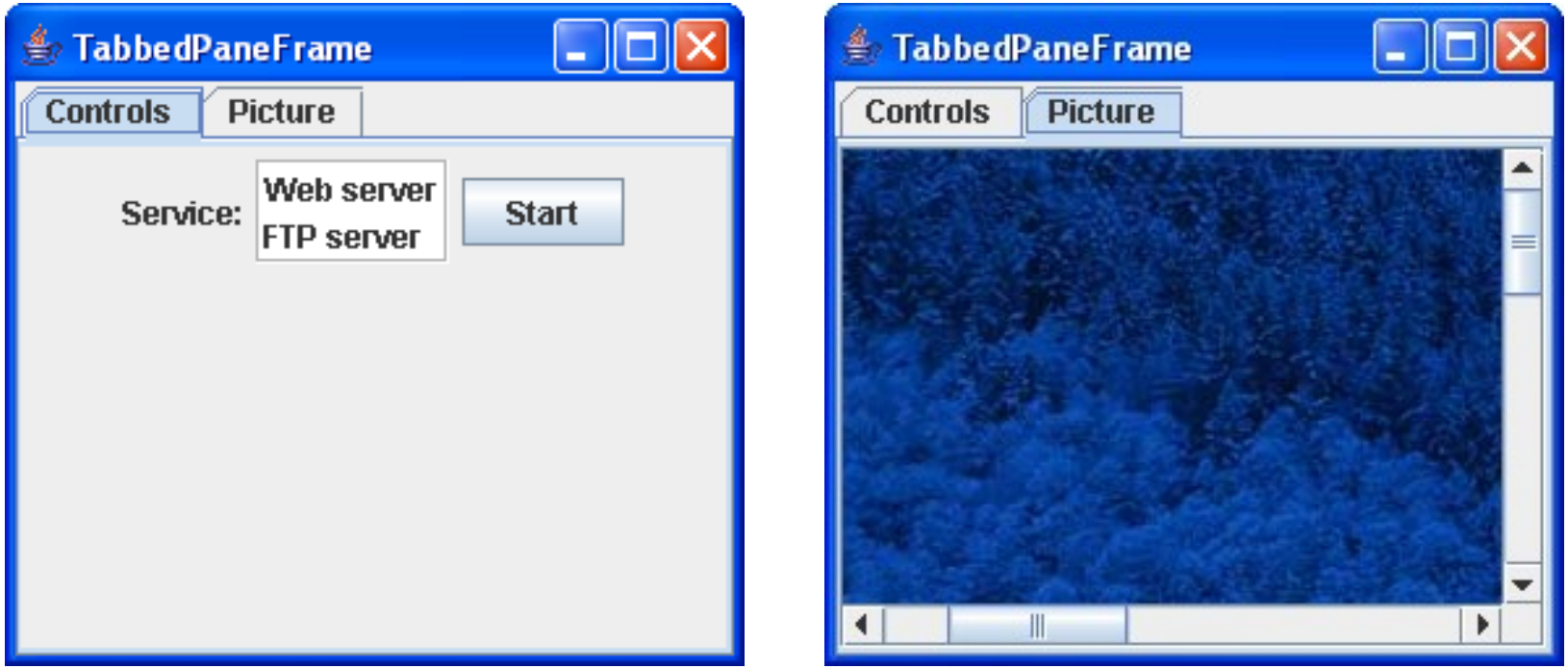
JSplitPane component



UsingSwing 9

```
class UsingSwing9 {
  public static void main (String[] args) {
    String fileOne = "Winter.jpg";
    String fileTwo = "Water lilies.jpg";
    if (args.length > 0) fileOne = args[0];
    if (args.length > 1) fileTwo = args[1];

    JFrame frame = new JFrame ("SplitPaneFrame");

    JLabel leftImage = new JLabel (new ImageIcon (fileOne));
    Component left = new JScrollPane (leftImage);
    JLabel rightImage = new JLabel (new ImageIcon (fileTwo));
    Component right = new JScrollPane (rightImage);

    JSplitPane split =
      new JSplitPane (JSplitPane.HORIZONTAL_SPLIT, left, right);
    split.setDividerLocation (100);
    frame.getContentPane ().add (split);

    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setSize (300, 200);
    frame.setVisible (true);
  }
}
```

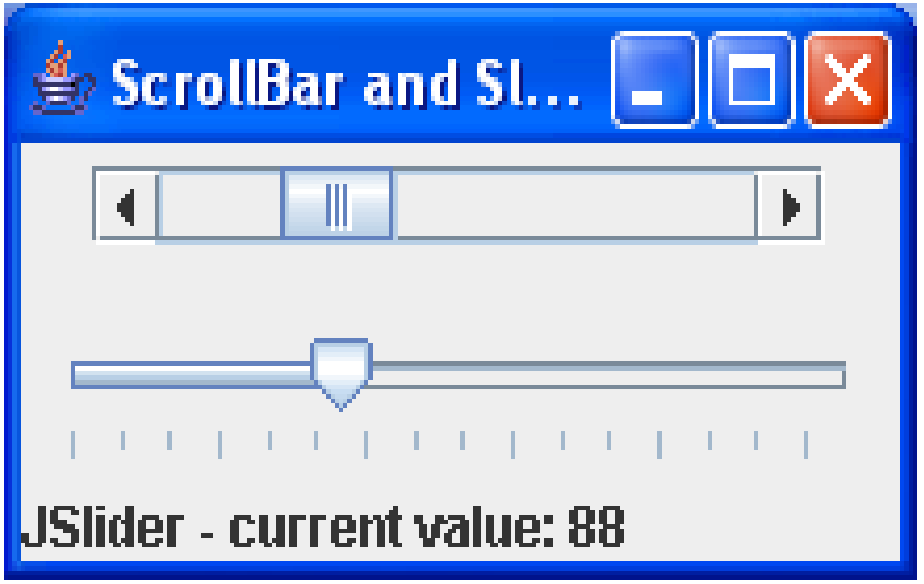
JTabbedPane component



UsingSwing 10

```
class UsingSwing10 {
  public static void main (String[] args)
  {
    JFrame frame = new JFrame ("TabbedPaneFrame");
    JTabbedPane tabby = new JTabbedPane ();

    // create controls pane
    JPanel controls = new JPanel ();
    controls.add (new JLabel ("Service:"));
    JList list = new JList (
        new String[] { "Web server", "FTP server" });
    list.setBorder (BorderFactory.createEtchedBorder ());
    controls.add (list);
    controls.add (new JButton ("Start"));

    // create image pane
    String filename = "Winter.jpg";
    JLabel image = new JLabel (new ImageIcon (filename));
    JComponent picture = new JScrollPane (image);
    tabby.addTab ("Controls", controls);
    tabby.addTab ("Picture", picture);

    frame.getContentPane ().add (tabby);

    frame.setSize (200, 200);
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setVisible (true);
  }
}
```

Scrollbar and slider



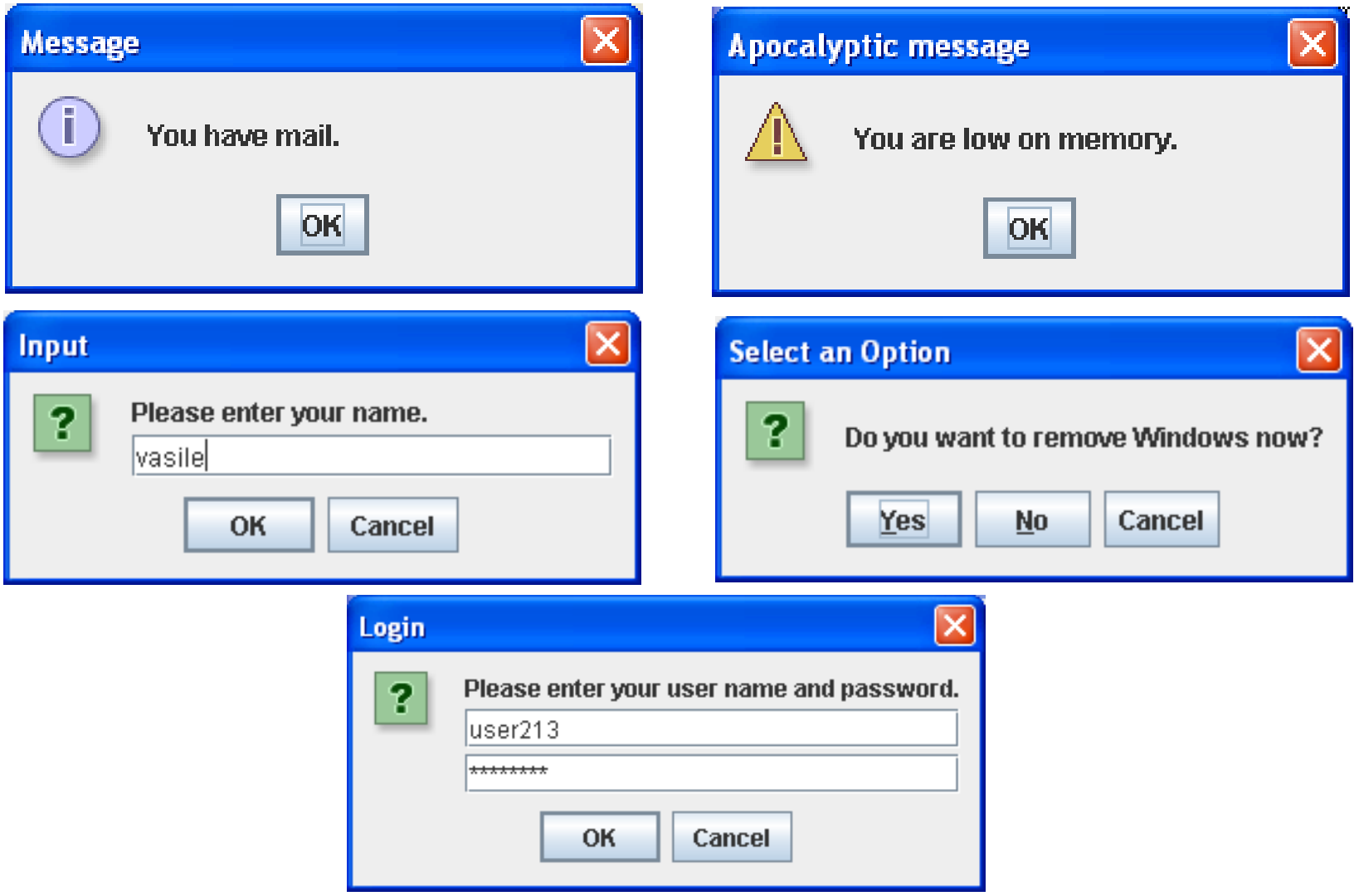
UsingSwing 11 (a)

```
class UsingSwing11 {
  public static void main (String[] args)
  {
    JFrame frame = new JFrame ("ScrollBar and Slider");
    Container content = frame.getContentPane ();

    JPanel main = new JPanel (new GridLayout (2, 1));
    JPanel scrollBarPanel = new JPanel ();
    final JScrollBar scrollBar =
        new JScrollBar (JScrollBar.HORIZONTAL, 0, 48, 0, 255);
    int height = scrollBar.getPreferredSize ().height;
    scrollBar.setPreferredSize (new Dimension (175, height));
    scrollBarPanel.add (scrollBar);
    main.add (scrollBarPanel);

    JPanel sliderPanel = new JPanel ();
    final JSlider slider =
        new JSlider (JSlider.HORIZONTAL, 0, 255, 128);
    slider.setMajorTickSpacing (48);
    slider.setMinorTickSpacing (16);
    slider.setPaintTicks (true);
    sliderPanel.add (slider);
    main.add (sliderPanel);

    content.add (main, BorderLayout.CENTER);
```

UsingSwing 11 (b)

```
    final JLabel statusLabel =
        new JLabel ("Demo Program");
    content.add (statusLabel, BorderLayout.SOUTH);

    // add event handlers
    scrollBar.addAdjustmentListener (new AdjustmentListener () {
      public void adjustmentValueChanged (AdjustmentEvent e) {
        statusLabel.setText ("JScrollBar - current value: "
                             + scrollBar.getValue ());
      }
    });

    slider.addChangeListener (new ChangeListener () {
      public void stateChanged (ChangeEvent e) {
        statusLabel.setText ("JSlider - current value: "
                             + slider.getValue ());
      }
    });

    frame.pack ();
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setVisible (true);
  }
}
```

Dialogs


Message
You have mail.
OK
Apocalyptic message
You are low on memory.
OK
Input
Please enter your name.
vasile
OK
Cancel
Select an Option
Do you want to remove Windows now?
Yes
No
Cancel
Login
Please enter your user name and password.
user213
****
OK
Cancel

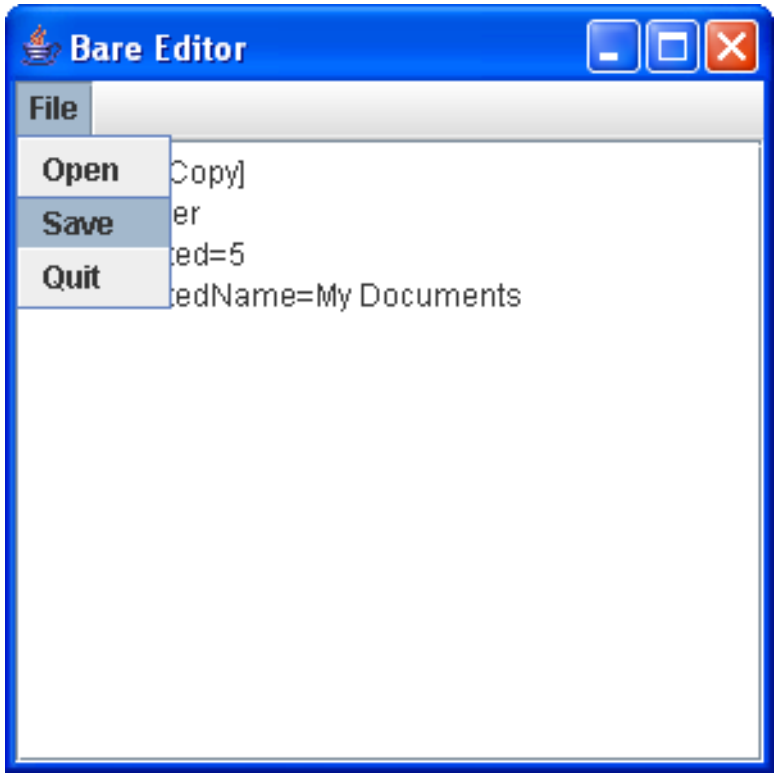
UsingSwing 12 (a)

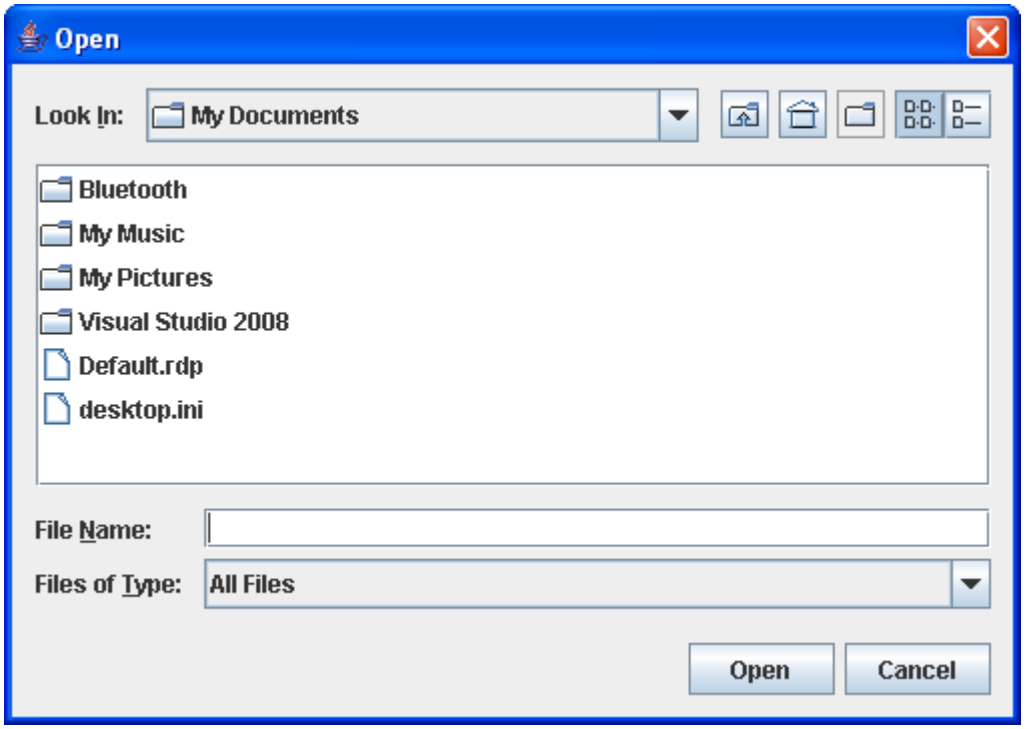
```
class UsingSwing12 {
  public static void main (String[] args) {
    JFrame frame = new JFrame ("Various Dialogs");
    frame.setSize (200, 200);
    frame.setVisible (true);

    JOptionPane.showMessageDialog (frame, "You have mail.");
    JOptionPane.showMessageDialog (frame, "You are low on memory.",
        "Apocalyptic message", JOptionPane.WARNING_MESSAGE);

    int result = JOptionPane.showConfirmDialog (
            null, "Do you want to remove Windows now?");
    switch (result) {
      case JOptionPane.YES_OPTION:
        System.out.println ("Yes"); break;
      case JOptionPane.NO_OPTION:
        System.out.println ("No"); break;
      case JOptionPane.CANCEL_OPTION:
        System.out.println ("Cancel"); break;
      case JOptionPane.CLOSED_OPTION:
        System.out.println ("Closed"); break;
    }
```

UsingSwing 12 (b)

```
    String name = JOptionPane.showInputDialog (
            null, "Please enter your name.");
    System.out.println (name);

    JTextField userField = new JTextField ();
    JPasswordField passField = new JPasswordField ();
    String message = "Please enter your user name and password.";
    result = JOptionPane.showOptionDialog (
            frame, new Object[] { message, userField, passField },
            "Login", JOptionPane.OK_CANCEL_OPTION,
            JOptionPane.QUESTION_MESSAGE,
            null, null, null);
    if (result == JOptionPane.OK_OPTION)
      System.out.println (userField.getText () +
          " " +  new String (passField.getPassword ()));

    System.exit(0);
  }
}
```

File selection dialog





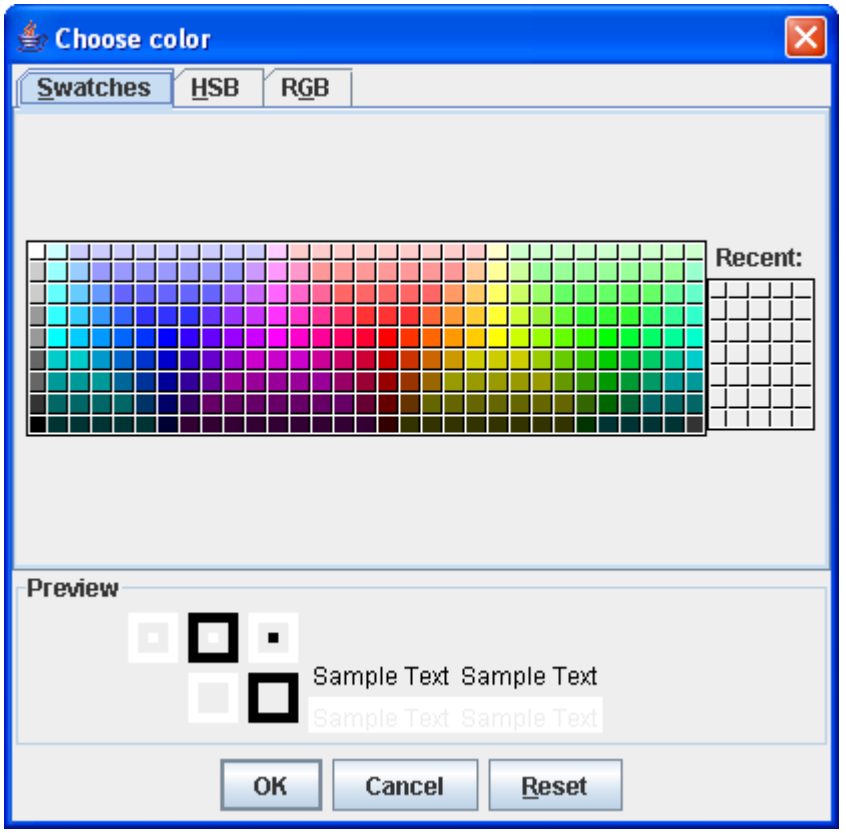
UsingSwing 13 (a)

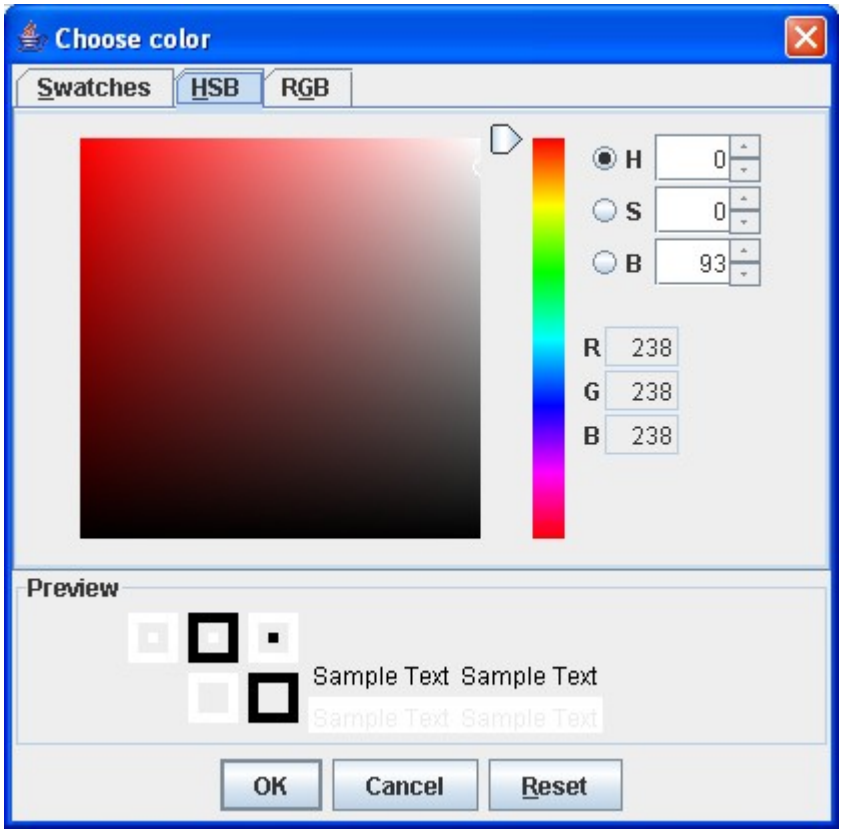
```
class UsingSwing13 extends JFrame implements ActionListener
{
  private JEditorPane textPane = new JEditorPane ();

  public UsingSwing13 () {
    super ("Bare Editor");
    Container content = getContentPane ();
    content.add (new JScrollPane (textPane), BorderLayout.CENTER);
    JMenu menu = new JMenu ("File");
    menu.add (makeMenuItem ("Open"));
    menu.add (makeMenuItem ("Save"));
    menu.add (makeMenuItem ("Quit"));
    JMenuBar menuBar = new JMenuBar ();
    menuBar.add (menu);
    setJMenuBar (menuBar);
    setSize (300, 300);
    setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
  }

  public void actionPerformed (ActionEvent e) {
    String command = e.getActionCommand ();
    if (command.equals ("Quit")) System.exit(0);
    else if (command.equals ("Open")) loadFile ();
    else if (command.equals ("Save")) saveFile ();
  }
```

UsingSwing 13 (b)

```
  private void loadFile () {
    JFileChooser chooser = new JFileChooser ();
    int result = chooser.showOpenDialog (this);
    if (result == JFileChooser.CANCEL_OPTION) return;
    try {
      File file = chooser.getSelectedFile ();
      java.net.URL url = file.toURL ();
      textPane.setPage (url);
    } catch (Exception e) {
      textPane.setText ("Could not load file: " + e);
    }
  }

  private void saveFile () {
    JFileChooser chooser = new JFileChooser ();
    chooser.showSaveDialog (this);
    // code to save file ...
  }

  private JMenuItem makeMenuItem (String name) {
    JMenuItem m = new JMenuItem (name);
    m.addActionListener (this);
    return m;
  }

  public static void main (String[] s) {
    new UsingSwing13 ().setVisible (true);
  }
}
```

Color selection dialog





UsingSwing 14

```
class UsingSwing14 {
  public static void main (String[] args) {
    final JFrame frame = new JFrame ("Color Chooser");
    final Container content = frame.getContentPane ();
    content.setLayout (new GridBagLayout ());
    JButton button = new JButton ("Change color...");
    content.add (button);

    button.addActionListener (new ActionListener () {
      public void actionPerformed (ActionEvent e) {
        Color c = JColorChooser.showDialog (
                frame, "Choose color", content.getBackground ());
        if (c != null) content.setBackground (c);
      }
    });

    frame.setSize (200, 200);
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setVisible (true);
  }
}
```