class StyleHandler implements ItemListener {
    public void itemStateChanged(ItemEvent e) {
        style = 0;

        if (styleItems[0].isSelected())
            style += Font.BOLD;

        if (styleItems[1].isSelected())
            style += Font.ITALIC;

        display.setFont(new Font(
            display.getFont().getName(), style, 72));

        repaint();
    }
}
Snazzy and Fun

- Pop-up menus
- Internal frames
// Fig. 13.8: PopupTest.java
// Demonstrating JPopupMenu
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class PopupTest extends JFrame {
    private JRadioButtonMenuItem items[];
    private Color colorValues[] =
        { Color.blue, Color.yellow, Color.red }
;
    public PopupTest() {
        super( "Using JPopupMenu" );
        final JPopupMenu popupMenu = new JPopupMenu();
        ItemHandler handler = new ItemHandler();
        String colors[] = { "Blue", "Yellow", "Red" };
        ButtonGroup colorGroup = new ButtonGroup();
        items = new JRadioButtonMenuItem[3];

        // construct each menu item and add to popup menu; also
        // enable event handling for each menu item
        for ( int i = 0; i < items.length; i++ ) {
            items[ i ] = new JRadioButtonMenuItem( colors[ i ] );
            popupMenu.add( items[ i ] );
            colorGroup.add( items[ i ] );
            items[ i ].addActionListener( handler );
        }
    }
}
getContentPane().setBackground( Color.white );

    // define a MouseListener for the window that displays
    // a JPopupMenu when the popup trigger event occurs
    addMouseListener(
        new MouseAdapter() {
            public void mousePressed( MouseEvent e )
            {
                checkForTriggerEvent( e );
            }

            public void mouseReleased( MouseEvent e )
            {
                checkForTriggerEvent( e );
            }

            private void checkForTriggerEvent( MouseEvent e )
            {
                if ( e.isPopupTrigger() )
                {
                    popupMenu.show( e.getComponent(),
                        e.getX(), e.getY() );
                }
            }
        });

    setSize( 300, 200 );
    show();
public static void main( String args[] )
{
    PopupTest app = new PopupTest();

    app.addWindowListener(
        new WindowAdapter() {
            public void windowClosing( WindowEvent e )
            {
                System.exit( 0 );
            }
        }
    );

    private class ItemHandler implements ActionListener {
        public void actionPerformed( ActionEvent e )
        {
            // determine which menu item was selected
            for ( int i = 0; i < items.length; i++ )
                if ( e.getSource() == items[ i ] ) {
                    getContentPane().setBackground(
                        colorValues[ i ] );
                    repaint();
                    return;
                }
        }
    }
}
// Demonstrating JDesktopPane.
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class DesktopTest extends JFrame {
    public DesktopTest() {
        super( "Using a JDesktopPane" );

        JMenuBar bar = new JMenuBar();
        JMenu addMenu = new JMenu( "Add" );
        JMenuItem newFrame = new JMenuItem( "Internal Frame" );
        addMenu.add( newFrame );
        bar.add( addMenu );
        setJMenuBar( bar );

        final JDesktopPane theDesktop = new JDesktopPane();
        getContentPane().add( theDesktop );