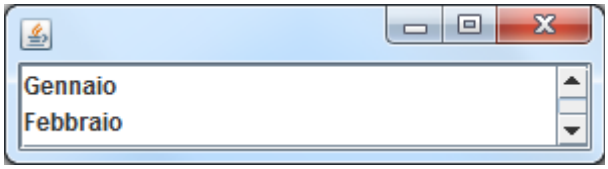
JList - ADT e Generics

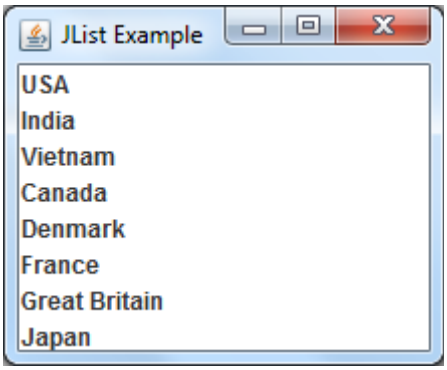
// con uso del costruttore JList(Vector<? extends E> *listData*)

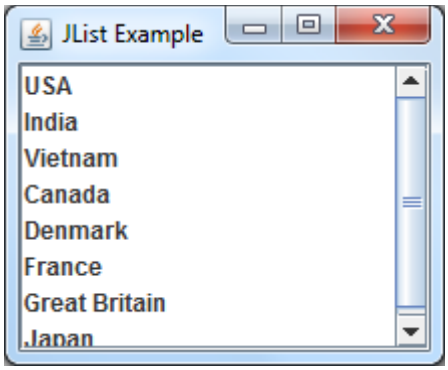
```
import java.util.*;
import javax.swing.*;

public class Liste_GUI {

  public Liste_GUI(){
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    Vector <String> months = new Vector <String>();
    JList <String> list = new JList <String> (months);

    months.add("Gennaio"); // anche months.addElement("Gennaio");
    months.add("Febbraio");
    months.add("Dicembre");

    frame.add(new JScrollPane(list));

    frame.setSize(300, 80); // per visualizzare barre a scorrimento
    frame.setVisible(true);
  }

  public static void main(String[] a) {
          new Liste_GUI();
  }
}
```



// con uso dell'implementazione di default: DefaultListModel

```
import javax.swing.DefaultListModel;
import javax.swing.JFrame;
import javax.swing.JList;
//import javax.swing.JScrollPane;
import javax.swing.SwingUtilities;

public class JListExample extends JFrame {
    private JList<String> countryList;          // dalla ver. 1.7

    public JListExample() {
        //create the model and add elements
        DefaultListModel<String> listModel = new DefaultListModel<>();
        listModel.addElement("USA");
        listModel.addElement("India");
        listModel.addElement("Vietnam");
        listModel.addElement("Canada");
        listModel.addElement("Denmark");
        listModel.addElement("France");
        listModel.addElement("Great Britain");
        listModel.addElement("Japan");

         countryList = new JList<>(listModel);   //create the list

        add(countryList);
        // add(new JscrollPane(countryList)); se si vuole aggiungere uno ScrollPane
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("JList Example");
        this.setSize(220,180);
        this.setLocationRelativeTo(null);
        this.setVisible(true);
    }
```





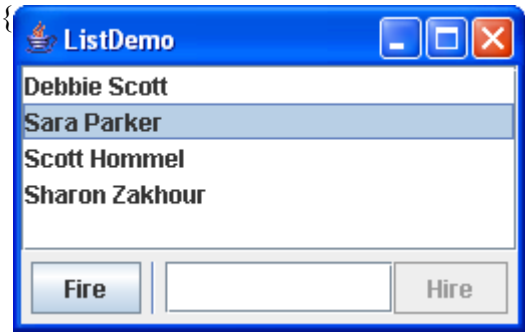
```
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {  // creazione della GUI come thread
            @Override
            public void run() {
                new JListExample();
            }
        });
    }
}
```

JList (da Tutorial Sun**) aggiornando con uso Generics**

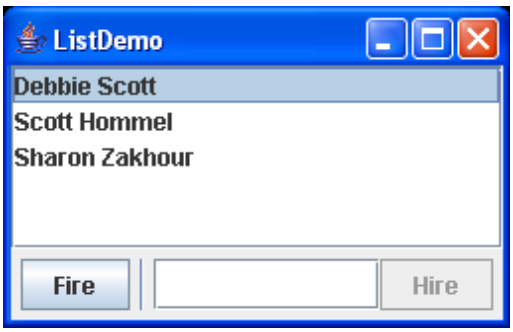
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class ListDemo extends JPanel implements ListSelectionListener {
    private JList <String> list; // con Generics
    DefaultListModel<String> listModel = new DefaultListModel<>();  // implementazione di default

    private static final String hireString = "Hire";
    private static final String fireString = "Fire";
    private JButton fireButton;
    private JTextField employeeName;

    public ListDemo() {
        super(new BorderLayout());

        listModel.addElement("Debbie Scott");
        listModel.addElement("Scott Hommel");
        listModel.addElement("Sharon Zakhour");

        //Create the list and put it in a scroll pane.
        list = new JList<>(listModel);
        list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        list.setSelectedIndex(0);
        list.addListSelectionListener(this);
        list.setVisibleRowCount(5);
        JScrollPane listScrollPane = new JScrollPane(list);

        JButton hireButton = new JButton(hireString);
        HireListener hireListener = new HireListener(hireButton);
        hireButton.setActionCommand(hireString);
        hireButton.addActionListener(hireListener);
        hireButton.setEnabled(false);

        fireButton = new JButton(fireString);
        fireButton.setActionCommand(fireString);
        fireButton.addActionListener(new FireListener());

        employeeName = new JTextField(10);
        employeeName.addActionListener(hireListener);
        employeeName.getDocument().addDocumentListener(hireListener);
        String name = listModel.getElementAt( list.getSelectedIndex()).toString();

        //Create a panel that uses BoxLayout.
        JPanel buttonPane = new JPanel();
        buttonPane.setLayout(new BoxLayout(buttonPane, BoxLayout.LINE_AXIS));

        buttonPane.add(fireButton);
        buttonPane.add(Box.createHorizontalStrut(5));
        buttonPane.add(new JSeparator(SwingConstants.VERTICAL));
        buttonPane.add(Box.createHorizontalStrut(5));
```

```
    buttonPane.add(employeeName);
    buttonPane.add(hireButton);
    buttonPane.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));

    add(listScrollPane, BorderLayout.CENTER);
    add(buttonPane, BorderLayout.PAGE_END);
}

class FireListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        //This method can be called only if there's a valid selection
        //so go ahead and remove whatever's selected.
        int index = list.getSelectedIndex();
        listModel.remove(index);

        int size = listModel.getSize();

        if (size == 0) {                                    //Nobody's left, disable firing.
            fireButton.setEnabled(false);

        } else {                                            //Select an index.
            if (index == listModel.getSize()) {
                                                            //removed item in last position
                index--;
            }

            list.setSelectedIndex(index);
            list.ensureIndexIsVisible(index);
        }
    }
}

//This listener is shared by the text field and the hire button.
class HireListener implements ActionListener, DocumentListener {
    private boolean alreadyEnabled = false;
    private JButton button;

    public HireListener(JButton button) {
        this.button = button;
    }

    //Required by ActionListener
    public void actionPerformed(ActionEvent e) {
        String name = employeeName.getText();

        //User didn't type in a unique name...
        if (name.equals("") || alreadyInList(name)) {
            Toolkit.getDefaultToolkit().beep();
            employeeName.requestFocusInWindow();
            employeeName.selectAll();
            return;
        }

        int index = list.getSelectedIndex(); //get selected index
        if (index == -1) { //no selection, so insert at beginning
            index = 0;
        } else {           //add after the selected item
            index++;
        }
        listModel.insertElementAt(employeeName.getText(), index);
        //If we just wanted to add to the end, we'd do this: listModel.addElement(employeeName.getText());
```

```
        //Reset the text field.
        employeeName.requestFocusInWindow();
        employeeName.setText("");

        //Select the new item and make it visible.
        list.setSelectedIndex(index);
        list.ensureIndexIsVisible(index);
    }
    //This method tests for string equality. You could certainly
    //get more sophisticated about the algorithm.  For example,
    //you might want to ignore white space and capitalization.
    protected boolean alreadyInList(String name) {
        return listModel.contains(name);
    }

    //Required by DocumentListener
    public void insertUpdate(DocumentEvent e) {
        enableButton();
    }

    //Required by DocumentListener.
    public void removeUpdate(DocumentEvent e) {
        handleEmptyTextField(e);
    }

    //Required by DocumentListener.
    public void changedUpdate(DocumentEvent e) {
        if (!handleEmptyTextField(e)) {
            enableButton();
        }
    }

    private void enableButton() {
        if (!alreadyEnabled) {
            button.setEnabled(true);
        }
    }

    private boolean handleEmptyTextField(DocumentEvent e) {
        if (e.getDocument().getLength() <= 0) {
            button.setEnabled(false);
            alreadyEnabled = false;
            return true;
        }
        return false;
    }
}

//This method is required by ListSelectionListener.
public void valueChanged(ListSelectionEvent e) {
    if (e.getValueIsAdjusting() == false) {
        if (list.getSelectedIndex() == -1) {
        //No selection, disable fire button.
            fireButton.setEnabled(false);

        } else {
        //Selection, enable the fire button.
            fireButton.setEnabled(true);
        }
    }
}
```

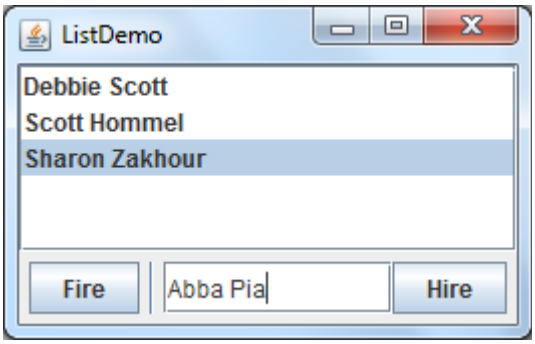


```
    /**
     * Create the GUI and show it.  For thread safety,
     * this method should be invoked from the
     * event-dispatching thread.
     */
    private static void createAndShowGUI() {
        //Create and set up the window.
        JFrame frame = new JFrame("ListDemo");
        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);

        //Create and set up the content pane.
        JComponent newContentPane = new ListDemo();
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);

        //Display the window.
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {

        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.

        javax.swing.SwingUtilities.invokeLater(new Runnable() {

            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

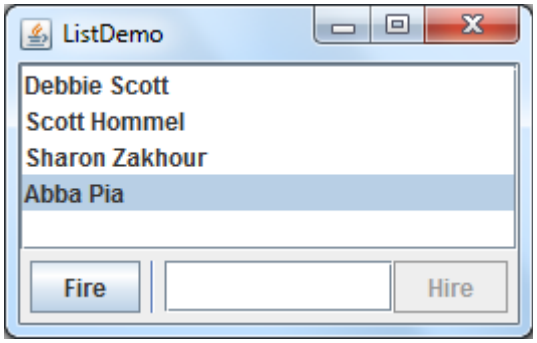


Inserimento: non automaticamente ordinato

prima…

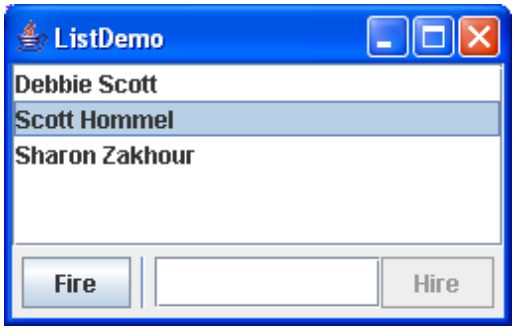


… e dopo



Cancellazione della voce selezionata:

prima…



dopo ….