Списък

Клас JList **(**javax.swing**)**

Списък с избор на един или много елементи. Съдържанието на списъка се представя чрез модел ListModel**. За създаване на прост динамичен списък се използва** DefaultListModel **клас за съхранение на елементите на списъка.**

public JList(ListModel dataModel)

Създава списък, изобразяващ елементите си в определен модел dataModel**.**

public void setSelectionMode(int selectionMode)

Определя вида на избора чрез selectionMode**:**

ListSelectionModel.SINGLE_SELECTION	**един елемент**
ListSelectionModel.SINGLE_INTERVAL_SELECTION	**интервал**
ListSelectionModel.MULTIPLE_INTERVAL_SELECTION	**много елементи**

public int[] getSelectedIndices()

Връща масив от избраните индекси.

Клас DefaultListModel **(**javax.swing**)**

Модел за елементите на динамичен списък.

public DefaultListModel()

Създава подразбиращ се модел.

public void addElement(Object obj)

Добавя елемент obj **към края на списъка.**

public Object elementAt(int index)

Връща елемента в избрания индекс index**.**

public void removeAllElements()

Премахва всички елементи от списъка и установява в 0 размера му.

Събития с избор на елементи от списък

1. **Клас** ListSelectionEvent **– показва промяна в текущия избор**

   public boolean getValueIsAdjusting()

   Връща true**, ако събитието е с многократна промяна.**

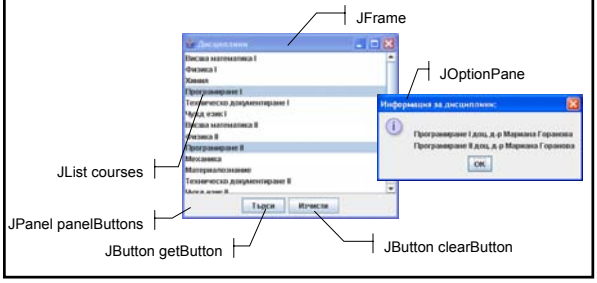
2. **Интерфейс** ListSelectionListener

```
public interface ListSelectionListener {
  // Извиква се при промяна на избора.
      void valueChanged(ListSelectionEvent e);
}
```

3. **Добавяне към списък на слушател** listener **за събитието** ListSelectionEvent

   public void addListSelectionListener(ListSelectionListener listener)

Пример: Приложение, което изобразява учебни дисциплини чрез списък. Бутонът Търси изобразява информация в диалогов прозорец за преподавателите на избраните дисциплини от списъка, а Изчисти – изчиства компонентите. Използва събитието ListSelectionEvent **за** JList**.**




```
import javax.swing.*;
import javax.swing.event.*;
import java.awt.event.*;
import java.awt.*;
public class ListDemo extends JFrame implements ActionListener,
                                                 ListSelectionListener {
  private JList courses;
  private DefaultListModel listModel;
  private int[] items;
  private String[] coursesYear1 = {"Висша математика I", "Физика I",
    "Химия", "Програмиране I", "Техническо документиране I",
    "Чужд език I", "Висша математика II", "Физика II",
    "Програмиране II", "Механика", "Материалознание",
    "Техническо документиране II", "Чужд език II","Физическа култура",
    "Учебна практика"};
  private String[] subjects = {"", "", "", "доц. д-р Мариана Горанова", "",
    "", "", "", "доц. д-р Мариана Горанова", "", "", "","","", ""};
  private JPanel panelButtons;
  private JButton getButton;
  private JButton clearButton;
```

```
public ListDemo() {
  super("Дисциплини");
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  Container container = this.getContentPane();
  listModel = new DefaultListModel();
  courses = new JList(listModel);
  courses.setSelectionMode
            (ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
  courses.addListSelectionListener(this);
  JScrollPane scroll = new JScrollPane(courses);
  container.add(scroll, BorderLayout.CENTER);
  panelButtons = new JPanel();
  getButton = new JButton("Търси");
  panelButtons.add(getButton);
  getButton.addActionListener(this);
  clearButton = new JButton("Изчисти");
  panelButtons.add(clearButton);
  clearButton.addActionListener(this);
  container.add(panelButtons, BorderLayout.PAGE_END);
  reset();
```

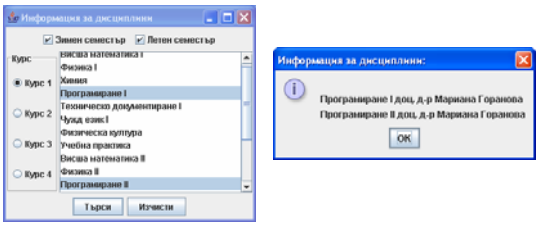
```
    this.setSize(350, 300);
    this.setVisible(true);
  }
  // Установява началното състояние на компонентите
  private void reset() {
    items = null;
    listModel.removeAllElements();
    for (int i = 0; i < coursesYear1.length; i++)
      listModel.addElement(coursesYear1[i]);
  }
  // Извиква се автоматично при промяна на избор в списъка
  public void valueChanged(ListSelectionEvent e) {
    if (e.getValueIsAdjusting() == false)
      items=courses.getSelectedIndices();
  }
```

```
  // Извиква се автоматично при натискане на бутон Търси или Изчисти
  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == getButton) {                  // Източник – бутон Търси
      String details = "";
      if (items != null && items.length != 0)
        for(int i = 0; i < items.length; i++)
          details += "\n" + listModel.elementAt(items[i])+" "+ subjects[items[i]];
      JOptionPane.showMessageDialog(null, details,
          "Информация за дисциплини:",
          JOptionPane.INFORMATION_MESSAGE);
    }
    else if (e.getSource() == clearButton)             // Източник – бутон Изчисти
      reset();
  }
  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        new ListDemo();
      }
    });
  }
}
```

дисциплина

преподавател

Пример: Приложение, което изобразява учебни дисциплини чрез списък в зависимост от избран семестър и курс. Бутонът Търси изобразява информация в диалогов прозорец за преподавателите на избраните дисциплини от списъка, а Изчисти – изчиства компонентите.




```
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;
import java.awt.event.*;
import java.awt.*;
public class StudentForm extends JFrame implements ActionListener,
                                    ItemListener, ListSelectionListener {
  private JPanel panelSemester;
  private JCheckBox winter;
  private JCheckBox summer;
  private JPanel panelYear;
  private ButtonGroup yearGroup;
  private JRadioButton year1;
  private JRadioButton year2;
  private JRadioButton year3;
  private JRadioButton year4;
  private String command;
  private JPanel panelCourses;
  private JList courses;
  private DefaultListModel listModel;
  private int[] items;
```

```
  private String[] coursesYear1Winter = {"Висша математика I","Физика I",
    "Химия","Програмиране I", "Техническо документиране I",
    "Чужд език I", "Физическа култура", "Учебна практика"};
  private String[] subjectsYear1Winter = {"", "", "",
     "доц. д-р Мариана Горанова", "", "", "", ""};
  private String[] coursesYear1Summer = {"Висша математика II",
    "Физика II", "Програмиране II", "Механика", "Материалознание",
    "Техническо документиране II", "Чужд език II","Физическа култура",
    "Учебна практика"};
  private String[] subjectsYear1Summer = {"", "",
    "доц. д-р Мариана Горанова", "", "", "", "", "", ""};
  private String[] coursesYear2Winter = {"Висша математика III",
    "Теоретична електротехника I", "Машинни елементи и механизми",
    "Икономика", "Техническа безопастност", "Чужд език III",
    "Физическа култура"};
  private String[] subjectsYear2Winter = {"", "", "", "", "", "", ""};
  private String[] coursesYear2Summer = {"Теоретична електротехника II",
    "Електрически измервания", "Електроника",
    "Цифрова и микропроцесорна техника",
    "Електротехнически материали", "Физическа култура"};
  private String[]subjectsYear2Summer = {"", "", "", "", "", "", ""};
```

```
  private String[] coursesYear3Winter = {"Електрически машини I",
    "Електрически апарати I", "Техника на високите напрежения",
    "Телекомуникации", "Осветителна и инасталационна техника",
    "Енергийна техника"};
  private String[] subjectsYear3Winter = {"", "", "", "", "", ""};
  private String[] coursesYear3Summer = {"Електрически машини II",
    "Електрически апарати II", "Преобразувателна техника",
    "Проектиране на електрически машини", "Мениджмънт"};
  private String[] subjectsYear3Summer = {"", "", "", "", "", ""};
  private String[] coursesYear4Winter = {"Електрозадвижване",
    "Електроенергетика", "Комутационна техника",
    "Технологии в електротехниката и електрониката",
    "Екология и възобновяеми източници на енергия"};
  private String[] subjectsYear4Winter = {"", "", "", "", "", ""};
  private String[] coursesYear4Summer = {
    "Електрически машини с електронно управление",
    "Числено моделиране на процеси и полета",
    "Електрически микромашини",
    "Изпитване и андеждност на електрически машини",
    "Свободно избираема дисциплина",
    "Свободно избираема дисциплина"};
  private String[] subjectsYear4Summer = {"", "", "", "", "", ""};
```

```
private JPanel panelButtons;
private JButton getButton;
private JButton clearButton;
public StudentForm () {
  super("Информация за дисциплини");
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  Container container = this.getContentPane();
  panelSemester = new JPanel();
  winter = new JCheckBox("Зимен семестър");
  panelSemester.add(winter);
  winter.addItemListener(this);
  summer = new JCheckBox("Летен семестър");
  panelSemester.add(summer);
  summer.addItemListener(this);
  container.add(panelSemester, BorderLayout.PAGE_START);
```

```
  panelCourses = new JPanel(new BorderLayout());
  yearGroup = new ButtonGroup();
  year1 = new JRadioButton("Курс 1");
  yearGroup.add(year1);
  year1.addActionListener(this);
  year2 = new JRadioButton("Курс 2");
  yearGroup.add(year2);
  year2.addActionListener(this);
  year3 = new JRadioButton("Курс 3");
  yearGroup.add(year3);
  year3.addActionListener(this);
  year4 = new JRadioButton("Курс 4");
  yearGroup.add(year4);
  year4.addActionListener(this);
  panelYear = new JPanel(new GridLayout(0, 1));
  panelYear.add(year1);
  panelYear.add(year2);
  panelYear.add(year3);
  panelYear.add(year4);
  panelYear.setBorder(new TitledBorder(new EtchedBorder(), "Курс"));
  panelCourses.add(panelYear, BorderLayout.LINE_START);
```

```
  listModel = new DefaultListModel();
  courses = new JList(listModel);
  courses.setSelectionMode
                (ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
  courses.addListSelectionListener(this);
  JScrollPane scroll = new JScrollPane(courses);
  panelCourses.add(scroll, BorderLayout.CENTER);
  container.add(panelCourses, BorderLayout.CENTER);
  panelButtons = new JPanel();
  getButton = new JButton("Търси");
  panelButtons.add(getButton);
  getButton.addActionListener(this);
  clearButton = new JButton("Изчисти");
  panelButtons.add(clearButton);
  clearButton.addActionListener(this);
  container.add(panelButtons, BorderLayout.PAGE_END);
  reset();
  this.setSize(350, 300);
  this.setVisible(true);
}
```

```
// Установява в начално състояние компонентите
public void reset() {
  winter.setSelected(false);
  summer.setSelected(false);
  items = null;
  command = year1.getActionCommand();
  year1.setSelected(true);
  year2.setSelected(false);
  year3.setSelected(false);
  year4.setSelected(false);
  radioButtonReset();
}
// Добавя елемент към списъка с дисциплини
private void addCourses (String[] courses) {
  for (int i = 0; i < courses.length; i++)
    listModel.addElement(courses[i]);
}
```

```
// Установява радио бутоните
public void radioButtonReset() {
  listModel.removeAllElements();
  if (year1.isSelected()) {                    // Избран Курс 1
    if (winter.isSelected() && summer.isSelected()) {
      addCourses(coursesYear1Winter);
      addCourses(coursesYear1Summer);
    }
    else if (winter.isSelected())
      addCourses(coursesYear1Winter);
    else if (summer.isSelected())
      addCourses(coursesYear1Summer);
  }
  if (year2.isSelected()) {                    // Избран Курс 2
    if (winter.isSelected() && summer.isSelected()) {
      addCourses(coursesYear2Winter);
      addCourses(coursesYear2Summer);
    }
    else if (winter.isSelected())
      addCourses(coursesYear2Winter);
    else if (summer.isSelected())
      addCourses(coursesYear2Summer);
    }
```

```
  if (year3.isSelected()) {                    // Избран Курс 3
    if (winter.isSelected() && summer.isSelected()) {
      addCourses(coursesYear3Winter);
      addCourses(coursesYear3Summer);
    }
    else if (winter.isSelected())
      addCourses(coursesYear3Winter);
    else if (summer.isSelected())
      addCourses(coursesYear3Summer);
  }
  if (year4.isSelected()) {                    // Избран Курс 4
    if (winter.isSelected() && summer.isSelected()) {
      addCourses(coursesYear4Winter);
      addCourses(coursesYear4Summer);
    }
    else if (winter.isSelected())
      addCourses(coursesYear4Winter);
    else if (summer.isSelected())
      addCourses(coursesYear4Summer);
  }
}
```

```
// Извиква се при натискане на бутоните Търси, Изчисти и радио
// бутоните за избор на курс
public void actionPerformed(ActionEvent e) {
  if (e.getSource() == getButton) {
    String details = "";
    if (items != null && items.length != 0) {
      for (int i = 0; i < items.length; i++) {
        details += "\n" + listModel.elementAt(items[i]) + " " ;
        if (year1.isSelected()) {
          if (winter.isSelected() && summer.isSelected()) {
            if (items[i] < coursesYear1Winter.length)
              details += subjectsYear1Winter[items[i]];
            else
              details +=
                subjectsYear1Summer[items[i]-coursesYear1Winter.length];
          }
          else if (winter.isSelected())
            details += subjectsYear1Winter[items[i]];
          else if (summer.isSelected())
            details += subjectsYear1Summer[items[i]];
        }
```

```
        if (year2.isSelected()) {
          if (winter.isSelected() && summer.isSelected()) {
            if (items[i] < coursesYear2Winter.length)
              details += subjectsYear2Winter[items[i]];
            else details +=
                  subjectsYear2Summer[items[i]-coursesYear2Winter.length];
          }
          else if (winter.isSelected())
            details += subjectsYear2Winter[items[i]];
          else if (summer.isSelected())
            details += subjectsYear2Summer[items[i]];
        }
        if (year3.isSelected()) {
          if (winter.isSelected() && summer.isSelected()) {
            if (items[i] < coursesYear3Winter.length)
              details += subjectsYear3Winter[items[i]];
            else details +=
                  subjectsYear3Summer[items[i]-coursesYear3Winter.length];
          }
          else if (winter.isSelected())
            details += subjectsYear3Winter[items[i]];
          else if (summer.isSelected())
            details += subjectsYear3Summer[items[i]];
        }
```

```
        if (year4.isSelected()) {
          if (winter.isSelected() && summer.isSelected()) {
            if (items[i] < coursesYear4Winter.length)
              details += subjectsYear4Winter[items[i]];
            else details +=
                  subjectsYear4Summer[items[i]-coursesYear4Winter.length];
          }
          else if (winter.isSelected())
            details += subjectsYear4Winter[items[i]];
          else if (summer.isSelected())
            details += subjectsYear4Summer[items[i]];
        }
      }
    }
    JOptionPane.showMessageDialog(null, details,
      "Информация за дисциплини:",
      JOptionPane.INFORMATION_MESSAGE);
  }
  else if (e.getSource() == clearButton) reset();
  else if ((e.getSource() == year1) || (e.getSource() == year2) ||
    (e.getSource() == year3) || (e.getSource() == year4)) {
    command = e.getActionCommand();
    radioButtonReset();
  }
}
```

```
// Извиква се при промяна на състоянието на бутоните за избор на
// Зимен и Летен семестър
public void itemStateChanged(ItemEvent e) {
  radioButtonReset();
}
// Извиква се при промяна на избора в списъка с дисциплини
public void valueChanged(ListSelectionEvent e) {
  if (e.getValueIsAdjusting() == false)
    items = courses.getSelectedIndices();
}

public static void main(String[] args) {
  SwingUtilities.invokeLater(new Runnable() {
    public void run() {
      new StudentForm();
    }
  });
}
}
```

Меню

1. Меню лента

Клас JMenuBar **(**javax.swing**)**

Реализира лента с менюта. Свързва се с JFrame **обект чрез извикване на метода** setJMenuBar **на фрейма.**

public void setJMenuBar(JMenuBar menubar)

public JMenuBar()

Създава лента с менюта.

public JMenu add(JMenu c)

Добавя меню c **в края на лентата с менюта.**

2. Меню

Клас JMenu **(**javax.swing**)**

Реализира меню – падащ прозорец, съдържащ JMenuItems **елементи, които се изобразяват при избор на елемент от** JMenuBar**. Може да съдържа разделител** JSeparators**.**

public JMenu(String s)

Създава меню с даден текст s**.**

public JMenuItem add(JMenuItem menuItem)

Добавя елемент menuItem **в края на менюто.**

3. Меню елемент

Клас JMenuItem **(**javax.swing**)**

Реализира елемент в меню.

public JMenuItem(String s)

Създава елемент в меню с даден текст s**.**

Събития

1. **Събития с действие**
- **клас** ActionEvent
- **интерфейс** ActionListener
  - **метод** actionPerformed
- **регистриране на компонентата за събитието** ActionEvent **чрез метода** addActionListener

4. Меню елемент с избор

Клас JCheckBoxMenuItem **(**javax.swing**)**

Реализира елемент в меню с възможност за избор.

public JCheckBoxMenuItem(String text)

Създава неселектиран меню елемент с избор с даден текст text**.**

Събития

1. **Събития с действие**
- **клас** ActionEvent
- **интерфейс** ActionListener
  - **метод** actionPerformed
- **регистриране на компонентата за събитието** ActionEvent **чрез метода** addActionListener

5. Радио бутон меню елемент

Клас JRadioButtonMenuItem **(**javax.swing**)**

Реализира радио бутон меню елемент. За да се управлява избора в група от радио бутони, се използва ButtonGroup **обект.**

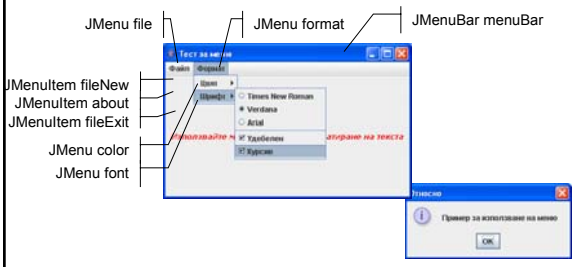
public JRadioButtonMenuItem(String text)

Създава радио бутон меню елемент с текст text**.**

Събития

1. **Събития с избор на елемент**
- **клас** ItemEvent
- **интерфейс** ItemListener
  - **метод** itemStateChanged
- **регистриране на компонентата за събитието** ItemEvent **чрез метода** addItemListener

Пример: Приложение, което изобразява меню Файл и Формат за избор на цвят и шрифт при изобразяване на текст.




```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class MenuDemo extends JFrame implements ActionListener,
                                                 ItemListener {
  private JMenuBar menuBar;
  private JMenu file;
  private JMenuItem newFile;
  private JMenuItem about;
  private JMenuItem exitFile;
  private JMenu format;
  private JMenu color;
  private ButtonGroup group1;
  private JRadioButtonMenuItem black;
  private JRadioButtonMenuItem red;
  private JRadioButtonMenuItem green;
  private JMenu font;
  private ButtonGroup group2;
  private JRadioButtonMenuItem timesNewRoman;
  private JRadioButtonMenuItem verdana;
  private JRadioButtonMenuItem arial;
```

```
  private JCheckBoxMenuItem bold;
  private JCheckBoxMenuItem italic;
  private int style;
  private String fontName;
  private int size;
  private JLabel display;
  public MenuDemo() {
    super("Тест за меню");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    Container container = this.getContentPane();
    menuBar = new JMenuBar();
    this.setJMenuBar(menuBar);
    file = new JMenu("Файл");
    menuBar.add(file);
    newFile = new JMenuItem("Нов");
    file.add(newFile);
    newFile.addActionListener(this);
    about = new JMenuItem("Относно");
    file.add(about);
    about.addActionListener(this);
```

```
exitFile = new JMenuItem("Изход");
file.add(exitFile);
exitFile.addActionListener(this);
format = new JMenu("Формат");
menuBar.add(format);
color = new JMenu("Цвят");
format.add(color);
group1 = new ButtonGroup();
black = new JRadioButtonMenuItem("Черен");
group1.add(black);
color.add(black);
black.addActionListener(this);
red = new JRadioButtonMenuItem("Червен");
group1.add(red);
color.add(red);
red.addActionListener(this);
green = new JRadioButtonMenuItem("Зелен");
group1.add(green);
color.add(green);
green.addActionListener(this);
```

```
font = new JMenu("Шрифт");
format.add(font);
group2 = new ButtonGroup();
timesNewRoman = new JRadioButtonMenuItem("Times New Roman");
group2.add(timesNewRoman);
font.add(timesNewRoman);
timesNewRoman.addActionListener(this);
verdana = new JRadioButtonMenuItem("Verdana");
group2.add(verdana);
font.add(verdana);
verdana.addActionListener(this);
arial = new JRadioButtonMenuItem("Arial");
group2.add(arial);
font.add(arial);
arial.addActionListener(this);
font.addSeparator();
bold = new JCheckBoxMenuItem("Удебелен");
font.add(bold);
bold.addItemListener(this);
```

```
italic = new JCheckBoxMenuItem("Курсив");
font.add(italic);
italic.addItemListener(this);
display = new JLabel();
display.setHorizontalAlignment(JLabel.CENTER);
container.add(display, BorderLayout.CENTER);
fontName = "Times New Roman";
style = Font.PLAIN;
size = 12;
this.pack();
this.setSize(350, 250);
this.setVisible(true);
}
private void clearColor() {
black.setSelected(false);
red.setSelected(false);
green.setSelected(false);
}
private void clearFont() {
timesNewRoman.setSelected(false);
verdana.setSelected(false);
arial.setSelected(false);
}
```

```
public void actionPerformed (ActionEvent e) {
JMenuItem source = (JMenuItem)(e.getSource());
if (source == newFile) {
display.setText
("Използвайте меню Формат за форматиране на текста");
display.setForeground(Color.BLACK);
style = Font.PLAIN;
fontName = "Times New Roman";
display.setFont(new Font(fontName, style, size));
}
else if (source == exitFile)
System.exit(0);
else if (source == about)
JOptionPane.showMessageDialog(null,
"Пример за използване на меню", "Относно",
JOptionPane.INFORMATION_MESSAGE);
else if (source == black) {
clearColor();
display.setForeground(Color.BLACK);
black.setSelected(true);
}
```

```
else if (source == red) {
clearColor();
display.setForeground(Color.RED);
red.setSelected(true);
}
else if (source == green) {
clearColor();
display.setForeground(Color.GREEN);
green.setSelected(true);
}
else if (source == timesNewRoman) {
clearFont();
fontName = "Times New Roman";
display.setFont(new Font(fontName, style, size));
timesNewRoman.setSelected(true);
}
else if (source == verdana) {
clearFont();
fontName = "Verdana";
display.setFont(new Font(fontName, style, size));
verdana.setSelected(true);
}
```

```
else if (source == arial) {
clearFont();
fontName = "Arial";
display.setFont(new Font(fontName, style, size));
arial.setSelected(true);
}
}
public void itemStateChanged(ItemEvent e) {
JMenuItem source = (JMenuItem)(e.getSource());
if (source == bold) {
if (bold.isSelected())
style += Font.BOLD;
else
style -= Font.BOLD;
}
else if (source == italic) {
if (italic.isSelected())
style += Font.ITALIC;
else
style -= Font.ITALIC;
}
display.setFont(new Font(fontName, style, size));
}
```

```
  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        new MenuDemo();
      }
    });
  }
}
```