Anotações da Aula

Tamer Cavalcante

Conteúdo

Capítulo 1

Utilizando JRadioButton

Um JRadioButton é um elemento gráfico com dois estados possíveis: selecionado ou não-selecionado. Este tipo de componente é utilizado para apresentar um conjunto de opções que são mutuamente exclusivas, ou seja, que não podem ocorrer ao mesmo tempo. Por exemplo, quando o usuário seleciona uma opção, caso já houvesse selecionado uma outra, esta seria automaticamente desselecionada. A Figura 1.1 apresenta um exemplo de utilização deste componente.

(a) Tela inicial. (b) Opção 1. (c) Opção 2.

Figura 1.1: Exemplo de utilização do JRadioButton.

O exemplo da Figura 1.1 funciona da seguinte forma: ao inicializar o sistema, na janela não apresenta nenhuma imagem, apenas 4 opções de redes sociais (Figura 1.1a). Em seguida, o usuário ao selecionar uma das opções, conforme podemos ver na Figura 1.1b, além de selecionar o Facebook, o sistema ainda apresenta uma imagem com a sua logo. Já na Figura 1.1c apresenta o comportamento da janela após o usuário selecionar uma outra opção. Neste caso, a imagem na tela é alterada e a opção do Facebook é desselecionada.

```
1 public class Tela implements ActionListener {
2   JLabel label;
3
4   public Tela () {
5     JFrame frame = new JFrame("Radio Button");
6     frame.setSize(240, 180);
7     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
8     frame.setVisible(true);
```

```
9
10      JRadioButton radio1 = new JRadioButton("Facebook");
11      radio1.setActionCommand("Facebook");
12      radio1.addActionListener(this);
13
14      JRadioButton radio2 = new JRadioButton("Instagram");
15      radio2.setActionCommand("Instagram");
16      radio2.addActionListener(this);
17
18      JRadioButton radio3 = new JRadioButton("Twitter");
19      radio3.setActionCommand("Twitter");
20      radio3.addActionListener(this);
21
22      JRadioButton radio4 = new JRadioButton("Youtube");
23      radio4.setActionCommand("Youtube");
24      radio4.addActionListener(this);
25
26      ButtonGroup group = new ButtonGroup();
27      group.add(radio1);
28      group.add(radio2);
29      group.add(radio3);
30      group.add(radio4);
31
32      JPanel panel = new JPanel(new GridLayout(4, 1));
33      panel.add(radio1);
34      panel.add(radio2);
35      panel.add(radio3);
36      panel.add(radio4);
37      frame.add(panel, BorderLayout.EAST);
38
39      label = new JLabel();
40      frame.add(label, BorderLayout.CENTER);
41    }
42
43    public void actionPerformed(ActionEvent e) {
44      String str = e.getActionCommand().toString();
45      label.setIcon(new ImageIcon("img/" +str.toLowerCase() +"
        .png"));
46    }
47
48    public static void main(String[] args) {
49      SwingUtilities.invokeLater(new Runnable() {
50        public void run() {
51          new Tela();
52        }
53      });
54    }
55  }
```

Código 1.1: Classe Tela.java

Agora vamos analisar o Código 1.1 que apresenta o fonte do exemplo da Figura 1.1. Após as configurações do JFrame, criamos os JRadioButtons da seguinte forma:

```
56 JRadioButton radio1 = new JRadioButton("Facebook");
57 radio1.setActionCommand("Facebook");
58 radio1.addActionListener(this);
```

Primeiro, instanciamos um objeto JRadioButton passando pelo seu construtor o texto que será apresentado pelo componente. Em seguida, alteramos o nome da ação que será executada pelo componente, isto é necessário pois precisamos diferenciar as ações entre os componentes. Por fim, adicionamos uma ação e repetimos este processo para todos os outros JRadioButtons.

O próximo passo na construção deste exemplo é agrupar todos os JRadioButton. O código abaixo apresenta como deve ser feito este agrupamento:

```
59 ButtonGroup group = new ButtonGroup();
60 group.add(radio1);
61 group.add(radio2);
62 group.add(radio3);
63 group.add(radio4);
```

Criamos um ButtonGroup e adicionamos todos os JRadioButton nele. Deste modo, só podemos selecionar uma opção entre as opções apresentadas. Recomendo a você comentar este trecho de código e ver o que acontece.

Para finalizar a implementação da janela, veremos como adicionar os componentes no JFrame. Como não especificamos o Layout dele, por padrão usamos o BorderLayout assunto visto no Capítulo 5. Assim, usaremos duas regiões: a CENTER e a EAST.

Na região CENTER, temos apenas a presença do JLabel para apresentar a imagem quando necessário. Adicionamos o JLabel da seguinte forma:

```
64 label = new JLabel();
65 frame.add(label, BorderLayout.CENTER);
```

Instanciamos o objeto, inicialmente vazio, e em seguida o adicionamos na região CENTER do JFrame.

Já na região EAST, teremos que organizar os componentes de forma que eles fiquem com apresentado na Figura 1.1. O trecho do código abaixo mostra como organizar estes componentes:

```
66 JPanel panel = new JPanel(new GridLayout(4, 1));
67 panel.add(radio1);
68 panel.add(radio2);
69 panel.add(radio3);
70 panel.add(radio4);
71 frame.add(panel, BorderLayout.EAST);
```

Observe que os JRadioButtons estão organizado de cima para baixo, ou seja, estão organizados em 4 linhas e 1 coluna. O Layout que possui esta característica

é o GridLayout. Então, criamos um JPanel configurado com GridLayout de 4 linhas e 1 coluna. Depois disto, adicionamos os JRadioButtons através da função add() do panel. E finalizamos a construção da janela adicionando o JPanel na região EAST do JFrame.

Ação que deve ser executada ao selecionar uma opção é esta:

```
72 public void actionPerformed(ActionEvent e) {
73   String str = e.getActionCommand().toString();
74   label.setIcon(new ImageIcon("img/" +str.toLowerCase() +".
       png"));
75 }
```

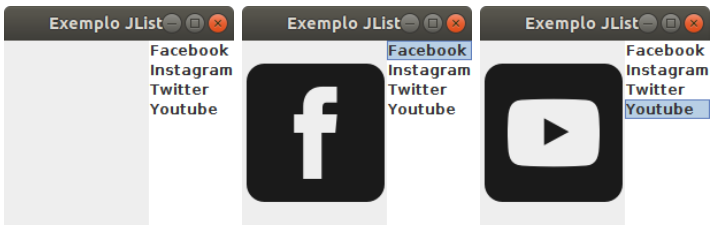
Lembrando que ao clicar em um JRadioButton, a função actionPerformed() será executada. Além disso, lembre-se que conseguimos saber qual ação foi selecionada através do método getActionCommand(). Assim, como definimos os nomes das ações com os mesmos nomes das imagens à serem exibidas, usamos esta informação para instanciar os ImageIcons que serão utilizados no JLabel. O código fonte apresentado nesta seção pode ser baixado através do link:

`https://www.dropbox.com/s/qd2eqg5c8q8v7gj/ExemploJRadioButton.zip`

Capítulo 2

Utilizando JList

O JList é um componente que apresenta ao usuário uma lista de itens para serem selecionados. Como elas podem possuir vários itens, geralmente são utilizadas em conjunto com o JScrollPane. Nesta seção analisaremos dois exemplos, o primeiro deles está apresentado na Figura 2.1. Neste exemplo, o usuário seleciona uma rede social e a sua logo é apresentada ao lado, bem semelhante aos exemplos que vimos com JComboBox e JRadioButton. O seu código fonte é apresentado no Código 2.1.



(a) Tela inicial. (b) Opção 1. (c) Opção 4.

Figura 2.1: Exemplo de utilização do JList.

Observe na linha 2 do Código 2.1 que para declarar um JList precisamos definir o tipo de informação que será armazenada nele, neste caso o JList receberá apenas Strings. Em seguida, definimos um Array de Strings com os valores que devem ser apresentados ao usuário e o utilizamos no construtor do JList (linhas 11 e 12). O próximo passo é definir como o usuário poderá selecionar os itens do JList, neste exemplo utilizamos o SINGLE_SELECTION.

```
1 public class ExemploList1 implements ListSelectionListener{
2   JList<String> list;
3   JLabel label;
4
5   public ExemploList1 () {
6     JFrame frame = new JFrame("Exemplo JList");
```

```
7       frame.setSize(200, 190);
8       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9       frame.setVisible(true);
10
11      String[] redesSociais = { "Facebook", "Instagram", "
        Twitter", "Youtube" };
12      list = new JList<String>(redesSociais);
13      list.setSelectionMode(ListSelectionModel.
        SINGLE_SELECTION);
14      list.addListSelectionListener(this);
15      frame.add(list, BorderLayout.EAST);
16
17      label = new JLabel();
18      frame.add(label, BorderLayout.CENTER);
19    }
20
21    @Override
22    public void valueChanged(ListSelectionEvent e) {
23      String str = list.getSelectedValue();
24      label.setIcon(new ImageIcon("img/" +str.toLowerCase() +"
        .png"));
25    }
26  }
```

Código 2.1: Classe ExemploList1.java

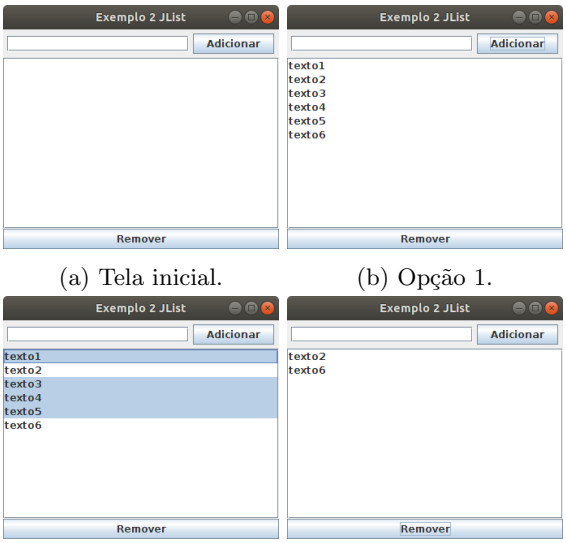
No SINGLE_SELECTION apenas um item pode ser selecionado por vez, assim quando o usuário seleciona um item, qualquer outro que esteja selecionado será desmarcado. As outras opções que podemos usar são:

- SINGLE_INTERVAL_SELECTION
- MULTIPLE_INTERVAL_SELECTION

No SINGLE_INTERVAL_SELECTION permite que o usuário selecione vários itens contíguos, enquanto o MULTIPLE_INTERVAL_SELECTION permite qualquer tipo de combinação de itens. Vale ressaltar que se não selecionarmos uma opção, o MULTIPLE_INTERVAL_SELECTION será escolhido por padrão.

Após a escolha de como selecionar os itens, adicionamos uma nova ação do JList que é a ListSelectionListener. Ou seja, ao selecionar um item, ou mais do que um dependendo do que você escolheu, a função valueChanged() será executada. Neste caso, apenas pegamos o valor do item selecionado pelo método getSelectedValue() do JList e usamos essa informação para carregar as imagens.

Nosso próximo exemplo, representado pela Figura 2.2, apresenta outras funcionalidades que o primeiro não possui. Ele apresenta um sistema na qual o usuário digita algum texto no JTextField e ao clicar no botão Adicionar, o conteúdo será copiado para o JList. Além disso, o usuário também poderá selecionar vários itens utilizando o MULTIPLE_INTERVAL_SELECTION e removê-los do JList.



(a) Tela inicial. (b) Opção 1.

(c) Opção 2. (d) Opção 2.

Figura 2.2: Exemplo 2 de utilização do JList.

Agora vamos analisar o Código 2.2. Utilizaremos DefaultListModel para conseguir adicionar e remover itens dinamicamente no JList. Para isso, definimos tanto o JList quanto o DefaultListModel do tipo String, pois neste exemplo manipularemos apenas texto. Caso seja necessário, você poderá manipular qualquer tipo de classe que você definir. Observe nas linhas 38 à 41, que instanciamos um objeto do tipo DefaultListModel e o passamos pelo construtor do JList. Em seguida, definimos um JScrollPane e utilizamos o JList na sua instanciação. E por fim, adicionamos o JScrollPane na região CENTER do JFrame.

```
27 public class ExemploList2 implements ActionListener{
28   JList<String> jlist;
29   DefaultListModel<String> listModel;
30   JTextField txt;
31
32   public ExemploList2 () {
33     JFrame frame = new JFrame("Exemplo 2 JList");
34     frame.setSize(340, 300);
35     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
36     frame.setVisible(true);
37
38     listModel = new DefaultListModel<String>();
39     jlist = new JList<String>(listModel);
40     JScrollPane scroll = new JScrollPane(jlist);
41     frame.add(scroll, BorderLayout.CENTER);
42
43     JPanel panel = new JPanel(new FlowLayout(FlowLayout.LEFT
```

```
    ));
44      txt = new JTextField(20);
45      panel.add(txt);
46      JButton button = new JButton("Adicionar");
47      button.addActionListener(this);
48      panel.add(button);
49      frame.add(panel, BorderLayout.NORTH);
50
51      button = new JButton("Remover");
52      button.addActionListener(this);
53      frame.add(button,BorderLayout.SOUTH);
54    }
55
56    @Override
57    public void actionPerformed(ActionEvent e) {
58      if(e.getActionCommand().equalsIgnoreCase("Adicionar")) {
59        if(!txt.getText().equalsIgnoreCase("")) {
60          listModel.addElement(txt.getText());
61          txt.setText("");
62        }
63      }else if(e.getActionCommand().equalsIgnoreCase("Remover"
      )) {
64        int indices[] = jlist.getSelectedIndices();
65        for(int i = indices.length-1; i>=0; i--)
66          listModel.remove(indices[i]);
67      }
68    }
69  }
```

Código 2.2: Classe ExemploList2.java

Da linha 43 à 53, temos a criação dos outros componentes e as suas inserções no JFrame, como esses assuntos já foram vistos nas outras seções, não será explicado aqui. Caso tenha alguma dúvida, verificar as seções anteriores.

Por fim, vamos analisar a função actionPerformed() que será executada ao clicar em um dos botões. Observe que dividimos em dois blocos: um para tratar quando o usuário clicar em Adicionar e outro para tratar o Remover. Em Adicionar, verificamos se o usuário digitou alguma coisa no JTextField. Caso o usuário digite algum texto, o seu conteúdo é adicionado no DefaultListModel através da função addElement(). Já para remover os elementos do JList, recuperamos os índices do itens selecionados através da função getSelectedIndices(). Esta função retorna um Array com as posições que foram selecionadas pelo usuário. Utilizamos esta informação para remover os itens através da função remove(). O código fonte dos exemplos apresentados nesta seção podem ser baixados através do link:

`https://www.dropbox.com/s/4oef8616op7tvt7/ExemploJList.zip`