

ECM251 – Linguagens de Programação I

Aula 11 – L1/1 e L2/1

Engenharia da Computação – 3ª série

JTable, JComboBox, JMenu e Diálogos
(L1/1 – L2/1)

2024

ECM251 – Linguagens de Programação I

Aula 11 – L1/1 e L2/1

Horário

Terça-feira: 2 x 2 aulas/semana

- L1/1 (07h40min-09h20min): *Prof. Calvetti*;
- L1/2 (09h30min-11h10min): *Prof. Calvetti*;
- L2/1 (07h40min-09h20min): *Prof. Igor Silveira*;
- L2/2 (11h20min-13h00min): *Prof. Calvetti*.

Tópico

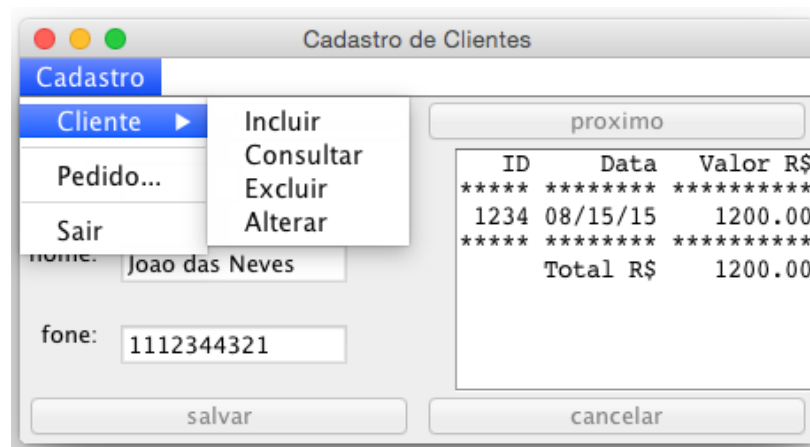
- Diálogo em Java

Diálogos

Definição



- Um diálogo em Java é uma janela de interface gráfica do usuário (GUI) que permite aos usuários interagirem com um programa Java por meio de entradas de texto, botões, caixas de seleção, menus suspensos e outros componentes visuais.



Diálogos

Exemplos



1. As classes apresentadas por este material, ***Dialogo1*** e ***Exemplo1***, são exemplos de diálogos que retornam valores entre eles, utilizando-se de instanciação:
 - A chamada do diálogo usa uma instanciação da classe ***Dialogo1***, classe que recebe outro valor;
2. As classes apresentadas por este material, ***Dialogo2*** e ***Exemplo2***, são exemplos de diálogos que retornam valores entre eles, utilizando-se de métodos:
 - A chamada do diálogo usa um método da classe ***Dialogo2***, classe que recebe outro valor.

Diálogos

Exemplos



- Nesses exemplos, o objetivo foi obter o retorno do valor de diálogo;
- Pode-se utilizar a mesma estrutura, para o caso se receber uma *senha*, por exemplo;
- O ***JDialog*** possui, no construtor, a classe que o chama, por isso utilizou-se do comando ***this*** e, também, uma ***flag*** que diz se o diálogo é **MODAL**, ou seja, só permite retornar à aplicação que o chamou após o seu término;
- Essa ***flag*** está atribuída inicialmente com ***true***.

Exemplos



```
1 // Exemplo de Diálogo por Instanciação
2 import javax.swing.*;
3 import java.awt.*;
4 import java.awt.event.*;
5 public class Dialogo1 extends JDialog implements ActionListener
6 {   JLabel lb;      // Rótulo
7     JTextField tf; // Campo Texto
8     JButton bt;    // Botão
9     int valor;     // Atributo de valor
10    public Dialogo1(JFrame fr)      // Método Construtor
11    {   super(fr, true);              // Invoca o construtor da superclasse
12        Container cp = getContentPane(); // Cria um Container de Tela
13        cp.setLayout(new FlowLayout()); // Determina a tela como um FlowLayout
14        lb = new JLabel("Valor:");    // Cria um Rótulo com o texto "Valor"
15        cp.add(lb);                  // Adiciona o Rótulo no Container
16        tf = new JTextField(10);      // Cria um Campo Texto de largura 10 pxls
17        cp.add(tf);                  // Adiciona o Campo Texto no Container
18        bt = new JButton ("OK");      // Cria um Botão com o texto "OK"
19        cp.add(bt);                  // Adiona o Botão no Container
20        bt.addActionListener(this);   // Adiciona uma ação ao Botão
21        setSize(200,100);             // Programa o tamanho inicial do FlowLayout em pxls
22        setLocation(200,200);         // Determina a localização inicial do FlowLayout em pxls
23        setTitle("Valores");          // Determina qual o título do FlowLayout
24        setVisible(true);             // Estabelece que o FlowLayout será visível inicialmente
25    }
```

Exemplos



```
26  public void actionPerformed(ActionEvent e)    //Tratamento do evento da ação do botão
27  {  if (e.getSource() == bt)                  // Verifica se o evento pertence ao botão
28      {  valor = Integer.parseInt(tf.getText()); // Lê o valor do campo texto e atribui
29          dispose();                             // Fecha e Libera o FlowLayout
30      }
31  }
32  public int getValor()    // Método para acessar o atributo Valor
33  {  return  valor;
34  }
35  }
36
37
```


Exemplos



```
1 // Exemplo de Diálogo por Instanciação - Teste
2 import javax.swing.*;
3 import java.awt.*;
4 import java.awt.event.*;
5
6 public class Exemplo1 extends JFrame implements ActionListener
7 { // Botão
8     JButton b;
9     // Método Construtor
10    public Exemplo1()
11    { // Cria um Container de Tela
12        Container c = getContentPane();
13        // Determina a tela como um FlowLayout
14        c.setLayout(new FlowLayout());
15        // Cria um Botão com o texto "Clique"
16        b = new JButton("Clique");
17        // Adiona o Botão no Container
18        c.add(b);
19        // Adiciona uma ação ao Botão
20        b.addActionListener(this);
21        // Programa o tamanho inicial do FlowLayout em pxls
22        setSize(350,75);
23        // Determina a localização inicial do FlowLayout em pxls
24        setLocation(470,50);
```

Exemplos



```
25     // Determina qual o título do FlowLayout
26     setTitle("Exemplo de Diálogo (Instanciação)");
27     // Estabelece que o FlowLayout será visível inicialmente
28     setVisible(true);
29 }
30
31 // Método de tratamento do evento da ação do botão
32 public void actionPerformed(ActionEvent e)
33 { // Verifica se o evento pertence ao botão
34     if(e.getSource() == b)
35     { // Instancia um objeto de Dialogo
36         Dialogo1 a = new Dialogo1(this);
37         // Apresenta o atributo valor (de Dialogo)
38         JOptionPane.showMessageDialog(null, "Valor Digitado: " + a.valor);
39     }
40 }
41
42 // Método Principal
43 public static void main(String args[])
44 { // Instancia um objeto de Exemplo
45     Exemplo1 prog = new Exemplo1();
46     // Programa encerramento do Frame no Sair (X)
47     prog.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
48 }
49 }
```

Exemplos



```
1 // Exemplo de Diálogo por Método
2 import javax.swing.*;
3 import java.awt.*;
4 import java.awt.event.*;
5
6 public class Dialogo2 extends JDialog implements ActionListener
7 {
8     JLabel lb; // Rótulo
9     JTextField tf; // Campo Texto
10    JButton bt; // Botão
11    int valor; // Atributo de valor
12
13    private Dialogo2(JFrame fr) // Método Construtor
14    {
15        super(fr, true); // Invoca o método construtor da superclasse
16        Container cp = getContentPane(); // Cria um Container de Tela
17        cp.setLayout(new FlowLayout()); // Determina a tela como um FlowLayout
18        lb = new JLabel("Valor:"); // Cria um Rótulo com o texto "Valor"
19        cp.add(lb); // Adiciona o Rótulo no Container
20        tf = new JTextField(10); // Cria um Campo Texto de largura 10 pxls
21        cp.add(tf); // Adiciona o Campo Texto no Container
22        bt = new JButton ("OK"); // Cria um Botão com o texto "OK"
23        cp.add(bt); // Adiciona o Botão no Container
24        bt.addActionListener(this); // Adiciona uma ação ao Botão
25        setSize(200,100); // Programa o tamanho inicial do FlowLayout em pxls
26        setLocation(200,200); // Determina a localização inicial do FlowLayout em pxls
27        setTitle("Valores"); // Determina qual o título do FlowLayout
28        setVisible(true); // Estabelece que o FlowLayout será visível inicialmente
29    }
30 }
```

Exemplos



```
28
29 public void actionPerformed(ActionEvent e) // Tratamento do evento da ação do botão
30 { if (e.getSource() == bt) // Verifica se o evento pertence ao botão
31 { valor = Integer.parseInt(tf.getText()); // Lê o campo texto e o atribui a valor
32 dispose(); // Fecha e Libera o FlowLayout
33 }
34 }
35
36 static int getValor(JFrame fr) // Método para acessar o atributo Valor
37 { Dialogo2 dl = new Dialogo2(fr);
38 return dl.valor;
39 }
40 }
41
42
```

Exemplos



```
1 // Exemplo de Diálogo por Método - Teste
2 import javax.swing.*;
3 import java.awt.*;
4 import java.awt.event.*;
5
6 public class Exemplo2 extends JFrame implements ActionListener
7 { // Botão
8     JButton b;
9     // Método Construtor
10    public Exemplo2()
11    { // Cria um Container de Tela
12        Container c = getContentPane();
13        // Determina a tela como um FlowLayout
14        c.setLayout(new FlowLayout());
15        // Cria um Botão com o texto "Clique"
16        b = new JButton("Clique");
17        // Adiona o Botão no Container
18        c.add(b);
19        // Adiciona uma ação ao Botão
20        b.addActionListener(this);
21        // Programa o tamanho inicial do FlowLayout em pxls
22        setSize(350,75);
23        // Determina a localização inicial do FlowLayout em pxls
24        setLocation(470,50);
```

Exemplos



```
25     // Determina qual o título do FlowLayout
26     setTitle("Exemplo de Diálogo (Método)");
27     // Estabelece que o FlowLayout será visível inicialmente
28     setVisible(true);
29 }
30
31 // Método de tratamento do evento da ação do botão
32 public void actionPerformed(ActionEvent e)
33 { // Verifica se o evento pertence ao botão
34     if(e.getSource() == b)
35     { // Apresenta o atributo valor (de Dialogo)
36         JOptionPane.showMessageDialog(null,"Valor Digitado:"+Dialogo2.getValor(this));
37     }
38 }
39
40 // Método Principal
41 public static void main(String args[])
42 { // Instancia um objeto de Exemplo
43     Exemplo2 prog = new Exemplo2();
44     // Programa encerramento do Frame no Sair (X)
45     prog.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46 }
47 }
```

Exercícios



1. Baseado no contexto de validação de um **Login** e uma **Senha**, armazenados a título de teste em variáveis atribuídas através de *hardcode* no próprio programa, desenvolver um programa único que se utilize de **Diálogo** para tal aplicação, através de Instanciação; e
2. Idem ao exercício 1, porém, desenvolver um programa único que se utilize de **Diálogo** para tal aplicação, através de Método.

ECM251 – Linguagens de Programação I

JTable, JComboBox, JMenu e Diálogos

Tópico

- *JMenu, JMenuItems e JMenuBar*

Definição



- A classe **JMenu** do Java deve ser utilizada para se instanciar objetos do tipo **menu**, que criam menus em tela, por exemplo:

```
// Cria o menu "Arquivo"
JMenu x = new JMenu("Arquivo");

// Ajusta o mnemônico para 'A'
x.setMnemonic('A'); // O menu é acionado Alt+A

// Coloca uma linha de separação no menu
x.addSeparator();

// Adiciona elementos de sub-menus ou itens
x.add();
```

Definição



- A classe **JMenuItem** do Java deve ser utilizada para se instanciar objetos do tipo **itens de menu**, que criam itens de menus em tela, por exemplo:

```
// Cria o item "cor" no menu
JMenuItem y = new JMenuItem("cor");

// Ajusta o mnemônico para 'c'
y.setMnemonic('c');
```

Definição



- A classe **JMenuBar** do Java deve ser utilizada para se instanciar objetos do tipo **barra de menu**, que criam barras para menus em tela, por exemplo:

```
// Cria uma barra para menu
JMenuBar barra = new JMenuBar();

// Coloca a barra no Frame
setJMenuBar(barra);

// Adiciona o menu 'x' à barra
Barra.add(x);
```

Definição



- Para saber se um elemento do menu foi escolhido, deve ser utilizado o **ActionListener** para elementos comuns ou o **ItemListener** para **check-box** e **radio buttons**.

Definição



- A classe **JRadioButtonMenuItem** do Java deve ser utilizada para se instanciar objetos do tipo **botões radio optionbox**, que criam botões de opção tipo rádio em tela, por exemplo:

```
// Cria botões de opção
```

```
JRadioButton radio1 = new JRadioButton("Plain", true);
```

```
JRadioButton radio2 = new JRadioRubtton("Bold", false);
```

Definição



- A classe **JCheckBoxMenuItem** do Java deve ser utilizada para se instanciar objetos do tipo **caixa de checagem checkbox**, que criam caixas de checagem em tela, por exemplo:

```
// Cria caixas de seleção
```

```
JCheckBox caixa1 = new JCheckBox("Itálico");
```

Definição



- A classe **ButtonGroup** do Java deve ser utilizada para se instanciar objetos do tipo **grupo de botões**, que criam grupos de botões mutuamente exclusivos em tela, por exemplo:

```
// Cria um grupo 'b' de botões mutuamente exclusivos
ButtonGroup b = new ButtonGroup();

// Adiciona o botão 'radio1' ao grupo
b.add(radio1);

// Adiciona o botão 'radio2' ao grupo
b.add(radio2);
```

ECM251 – Linguagens de Programação I

JMenu, JMenuitem e JMenuBar

Exemplos



1. Os arquivos de códigos fonte em Java (*.java*), que contêm as classes **CheckBoxFrame** e **CheckBoxTest**, se encontram digitados no material[©] anexo a esta aula;
2. Os arquivos de códigos fonte em Java (*.java*), que contêm as classes **RadioButtonFrame** e **RadioButtonTest**, se encontram digitados no material[©] anexo a esta aula;
3. Os arquivos de códigos fonte em Java (*.java*), que contêm as classes **MenuFrame** e **MenuTest**, se encontram digitados no material[©] anexo a esta aula.

© Copyright 1992-2005 by Deitel & Associates, Inc. and Pearson Education, Inc. All Rights Reserved.

Exercício



Crie uma aplicação para atender a uma loja de roupas, capaz de configurar, via menu, um pedido, **em texto**, na tela, contendo:

1. Tipo: Camisa, camiseta, calça, vestido, saia etc.;
2. Gênero: Masculino, Feminino e Unissex;
3. Tamanho: PP, P, M, G, GG, XG;
4. Cor: Vermelha, Preta, Amarela etc.;
5. Pagamento: Dinheiro, Cartão de Crédito ou Cartão de Débito.

Exemplo de pedido impresso em tela: Camiseta masculina, Tamanho PP, Preta e pagamento em Dinheiro.

ECM251 – Linguagens de Programação I

JTable, JComboBox, JMenu e Diálogos

Tópico

- *JTable e JComboBox*

Definição



- Um componente de interface, ou *widget*, definido pela classe **JTable** do Java serve para apresentar uma tabela em uma interface gráfica, ou **JFrame**;
- Um de seus construtores recebe dois parâmetros:
 1. Um vetor com o título das colunas; e
 2. Uma matriz com as linhas e colunas da tabela.

Definição



- Um componente de interface, ou *widget*, definido pela classe **JComboBox** do Java serve para apresentar uma lista de elementos na tela;
- Cada vez que se clica em um elemento, ele dispara um evento do tipo **ItemEvent**;
- As classes que implementam a interface **ItemListener** e tem o método **public void itemStateChanged(ItemEvent)** podem realizar ações quando um item da lista é selecionado (clique);
- Um dos construtores do **JComboBox** recebe um vetor de itens como parâmetro.

ECM251 – Linguagens de Programação I

JTable e JComboBox

Exemplo



```
1 public class Cliente
2 { private String nome;
3   private String cpf;
4   private String genero;
5   private String estadoCivil;
6   public Cliente(String nome, String cpf, String genero, String estadoCivil)
7   { this.nome = nome;
8     this.cpf = cpf;
9     this.genero = genero;
10    this.estadoCivil = estadoCivil;
11  }
12  public String getNome()
13  { return nome;
14  }
15  public String getCpf()
16  { return cpf;
17  }
18  public String getGenero()
19  { return genero;
20  }
21  public String getEstadoCivil()
22  { return estadoCivil;
23  }
24 }
25
```

ECM251 – Linguagens de Programação I

JTable e JComboBox

Exemplo



```
1 import javax.swing.*;
2 import javax.swing.table.DefaultTableModel;
3 import java.util.ArrayList;
4 public class CadastroClientes
5 { private ArrayList<Cliente> meusClientes;
6   public CadastroClientes()
7   { meusClientes = new ArrayList<Cliente>();
8   }
9   public void adicionaCliente(Cliente c)
10  { meusClientes.add(c);
11  }
12  public void mostraTabelaDeClientes()
13  { // Cria um JFrame para exibir a tabela
14    JFrame frame = new JFrame("Exemplo JTable e JComboBox");
15    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16    // Define as colunas da tabela
17    String colunas[] = {"Nome", "CPF", "Genero", "Estado Civil"};
18    // Cria um modelo de tabela com as colunas definidas
19    DefaultTableModel modelo = new DefaultTableModel(colunas, 0);
20
21    // Adiciona os clientes do ArrayList ao modelo de tabela
22    for(Cliente cliente : meusClientes)
23    { Object dados[] = {cliente.getNome(), cliente.getCpf(), cliente.getGenero(), cliente.getEstadoCivil()};
24      modelo.addRow(dados);
25    }
```

ECM251 – Linguagens de Programação I

JTable e JComboBox

Exemplo



```
26 // Cria uma tabela com o modelo definido
27 JTable tabela = new JTable(modelo);
28 // Define um JComboBox com as opções para o menu suspenso
29 String opcoes[] = {"Solteiro", "Casado", "Divorciado", "Viúvo"};
30 JComboBox<String> comboBox = new JComboBox<String>(opcoes);
31 // Adiciona o JComboBox à tabela na coluna "Estado Civil"
32 tabela.getColumnModel().getColumn(3).setCellEditor(new DefaultCellEditor(comboBox));
33 // Cria um JScrollPane para exibir a tabela
34 JScrollPane scrollPane = new JScrollPane(tabela);
35 // Adiciona o JScrollPane ao JFrame
36 frame.add(scrollPane);
37 // Define o tamanho do JFrame e o torna visível
38 frame.setSize(500, 200);
39 frame.setVisible(true);
40 }
41 }
42 }
```

Exemplo

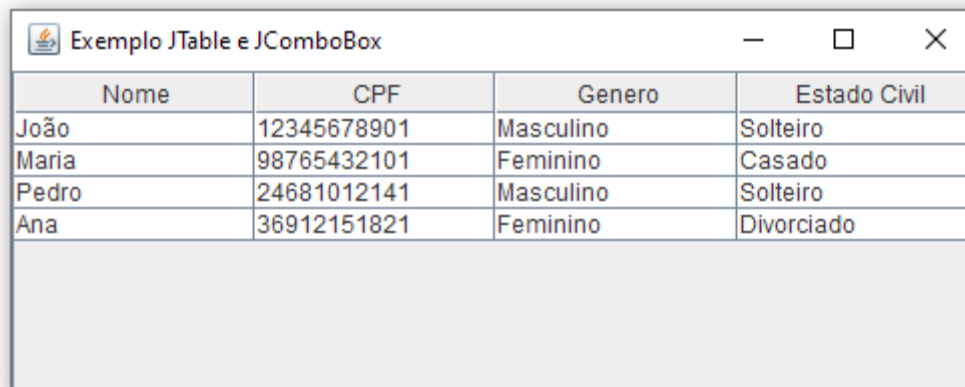


```
1 public class TesteCadastroClientes
2 {   public static void main(String args[])
3     {   CadastroClientes cc = new CadastroClientes();
4         cc.adicionaCliente(new Cliente("João", "12345678901", "Masculino", "Solteiro"));
5         cc.adicionaCliente(new Cliente("Maria", "98765432101", "Feminino", "Casado"));
6         cc.adicionaCliente(new Cliente("Pedro", "24681012141", "Masculino", "Solteiro"));
7         cc.adicionaCliente(new Cliente("Ana", "36912151821", "Feminino", "Divorciado"));
8         cc.mostraTabelaDeClientes();
9     }
10 }
11
```


Exemplo



- Quando executado o código fornecido neste exemplo, será apresentada a janela com a tabela de clientes a seguir:



Nome	CPF	Genero	Estado Civil
João	12345678901	Masculino	Solteiro
Maria	98765432101	Feminino	Casado
Pedro	24681012141	Masculino	Solteiro
Ana	36912151821	Feminino	Divorciado

ECM251 – Linguagens de Programação I

JTable, JComboBox, JMenu e Diálogos

Exercícios Extras



- Propostos pelo professor em aula, utilizando os conceitos abordados neste material...

Bibliografia Básica



- MILETTO, Evandro M.; BERTAGNOLLI, Silvia de Castro. Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, javascript e PHP (Tekne). Porto Alegre: Bookman, 2014. E-book. Referência Minha Biblioteca:
<https://integrada.minhabiblioteca.com.br/#/books/9788582601969>
- WINDER, Russel; GRAHAM, Roberts. Desenvolvendo Software em Java, 3ª edição. Rio de Janeiro: LTC, 2009. E-book. Referência Minha Biblioteca:
<https://integrada.minhabiblioteca.com.br/#/books/978-85-216-1994-9>
- DEITEL, Paul; DEITEL, Harvey. Java: how to program early objects. Hoboken, N. J: Pearson, c2018. 1234 p. ISBN 9780134743356.

Continua...

Bibliografia Básica (continuação)



- HORSTMANN, Cay S; CORNELL, Gary. Core Java. SCHAFRANSKI, Carlos (Trad.), FURMANKIEWICZ, Edson (Trad.). 8. ed. São Paulo: Pearson, 2010. v. 1. 383 p. ISBN 9788576053576.
- LIANG, Y. Daniel. Introduction to Java: programming and data structures comprehensive version. 11. ed. New York: Pearson, c2015. 1210 p. ISBN 9780134670942.
- TURINI, Rodrigo. Desbravando Java e orientação a objetos: um guia para o iniciante da linguagem. São Paulo: Casa do Código, [2017]. 222 p. (Caelum).

Bibliografia Complementar



- HORSTMANN, Cay. Conceitos de Computação com Java. Porto Alegre: Bookman, 2009. E-book. Referência Minha Biblioteca:
<https://integrada.minhabiblioteca.com.br/#/books/9788577804078>
- MACHADO, Rodrigo P.; FRANCO, Márcia H. I.; BERTAGNOLLI, Silvia de Castro. Desenvolvimento de software III: programação de sistemas web orientada a objetos em java (Tekne). Porto Alegre: Bookman, 2016. E-book. Referência Minha Biblioteca:
<https://integrada.minhabiblioteca.com.br/#/books/9788582603710>
- BARRY, Paul. Use a cabeça! Python. Rio de Janeiro: Alta Books, 2012. 458 p.
ISBN 9788576087434.

Continua...

ECM251 – Linguagens de Programação I

Aula 11 – L1/1 e L2/1

Bibliografia Complementar (continuação)



- LECHETA, Ricardo R. Web Services RESTful: aprenda a criar Web Services RESTful em Java na nuvem do Google. São Paulo: Novatec, c2015. 431 p.
ISBN 9788575224540.
- SILVA, Maurício Samy. JQuery: a biblioteca do programador. 3. ed. rev. e ampl. São Paulo: Novatec, 2014. 544 p.
ISBN 9788575223871.
- SUMMERFIELD, Mark. Programação em Python 3: uma introdução completa à linguagem Python. Rio de Janeiro: Alta Books, 2012. 506 p.
ISBN 9788576083849.

Continua...

ECM251 – Linguagens de Programação I

Aula 11 – L1/1 e L2/1

Bibliografia Complementar (continuação)



- YING, Bai. Practical database programming with Java. New Jersey: John Wiley & Sons, c2011. 918 p.
- ZAKAS, Nicholas C. The principles of object-oriented JavaScript. San Francisco, CA: No Starch Press, c2014. 97 p. ISBN 9781593275402.

ECM251 – Linguagens de Programação I

Aula 11 – L1/1 e L2/1

FIM

ECM251 – Linguagens de Programação I

Aula 11 – L1/2 e L2/2

Engenharia da Computação – 3ª série

JTable, JComboBox, JMenu e Diálogos
(L1/2 – L2/2)

2024

ECM251 – Linguagens de Programação I

Aula 11 – L1/2 e L2/2

Horário

Terça-feira: 2 x 2 aulas/semana

- L1/1 (07h40min-09h20min): *Prof. Calvetti*;
- L1/2 (09h30min-11h10min): *Prof. Calvetti*;
- L2/1 (07h40min-09h20min): *Prof. Igor Silveira*;
- L2/2 (11h20min-13h00min): *Prof. Calvetti*.

Exercícios



1. Utilizando-se de **JMenu** e **JMenuItem**, desenvolver um aplicativo em Java capaz de atender às seguintes solicitações:
 - a. Criar um menu denominado **Vetor**. Ao ser selecionado, apresente os itens denominados **Dimensiona**, **Digita** e **Apresenta**.
 - No item **Dimensiona** será solicitado o tamanho de um vetor do tipo **double**;
 - No item **Digita** serão solicitados todos os elementos deste vetor;
 - No item **Apresenta** serão apresentados todos os elementos deste vetor.

Exercícios



Utilizando-se de **JMenu** e **JMenuItem**, desenvolver um aplicativo em Java capaz de atender às seguintes solicitações:

- b. Criar um menu denominado **PROBEST**. Ao ser selecionado, apresente os itens denominados **Média**, **Desvio Padrão**, **Variância**, **Mediana**, **Coef. Assimetria** e **Coef. Variação**.
- Ao ser selecionado cada um destes itens, deverá ser apresentado o seu respectivo cálculo estatístico, sobre o vetor digitado no menu **Vetor**;

Exercícios



Utilizando-se de **JMenu** e **JMenuItem**, desenvolver um aplicativo em Java capaz de atender às seguintes solicitações:

- c. Criar um menu denominado **CDI**. Ao ser selecionado, apresente os itens denominados **Função** e **Derivada**.
- Ao ser selecionado cada um destes itens, deverá ser apresentado o seu respectivo cálculo, sobre o vetor digitado no menu **Vetor**;

Exercícios



Utilizando-se de **JMenu** e **JMenuItem**, desenvolver um aplicativo em Java capaz de atender às seguintes solicitações:

- d. Criar um menu denominado **ALGESD**. Ao ser selecionado, apresente os itens denominados **Ordenação** e **Busca**.
- Caso seja selecionado o item **Ordenação**, deverão ser apresentados os itens **Trocas**, **Seleção**, **Inserção**, **Quick** e **Merge**.
- Ao ser selecionado cada um destes itens, deverá ser realizada a ordenação do vetor digitado no menu **Vetor**, através do respectivo algoritmo de ordenação escolhido.

Exercícios



Utilizando-se de **JMenu** e **JMenuItem**, desenvolver um aplicativo em Java capaz de atender às seguintes solicitações:

- Caso seja selecionado o item **Busca**, deverão ser apresentados os itens **Linear Iterativa**, **Linear Recursiva**, **Binária Iterativa** e **Binária Recursiva**;
- Ao ser selecionado cada um destes itens, deverá ser solicitado um valor **chave** para a busca e, a seguir, realizada a busca desse valor chave no vetor digitado no menu **Vetor**, através do respectivo algoritmo de busca;

Exercícios



Utilizando-se de ***JMenu*** e ***JMenuItem***, desenvolver um aplicativo em Java capaz de atender às seguintes solicitações:

- Lembrar que para realizar qualquer uma das buscas binárias oferecidas, o vetor deverá estar ordenado antecipadamente (solicitar a escolha do algoritmo para realizar a prévia ordenação).

ECM251 – Linguagens de Programação I

JTable, JComboBox, JMenu e Diálogos

Exercícios



2. Utilizar o tempo de aula restante para adiantar a codificação do Projeto I, semestral, solicitado na matéria, pelo professor.

Bibliografia (apoio)



- LOPES, ANITA. GARCIA, GUTO. Introdução à Programação: 500 algoritmos resolvidos. Rio de Janeiro: Elsevier, 2002.
- DEITEL, P. DEITEL, H. Java: como programar. 8 Ed. São Paulo: Prentice-Hall (Pearson), 2010;
- BARNES, David J.; KÖLLING, Michael. Programação orientada a objetos com Java: uma introdução prática usando o BlueJ . 4. ed. São Paulo: Pearson Prentice Hall, 2009.

ECM251 – Linguagens de Programação I

Aula 11 – L1/2 e L2/2

FIM