**Curso Profissional de Gestão e Programação de Sistemas Informáticos**

**Disciplina de Programação e Sistemas de Informação**

Módulo 11 – Programação Orientada a Objetos Avançada

Aluno: Carlos Tojal

Professora: Matilde Vieira

Caldas da Rainha, Janeiro de 2020

# Índice

[Índice 2](#__RefHeading___Toc12689_704678284)

[Agradecimentos 6](#__RefHeading___Toc12321_704678284)

[Índice de figuras 7](#__RefHeading___Toc13147_2739222848)

[Introdução 8](#__RefHeading___Toc12531_704678284)

[Desenvolvimento 9](#__RefHeading___Toc13149_2739222848)

[Exceções 9](#__RefHeading___Toc13151_2739222848)

[Streams 10](#__RefHeading___Toc13153_2739222848)

[ArrayList 11](#__RefHeading___Toc13155_2739222848)

[Interface Gráfica (GUI) 12](#__RefHeading___Toc13157_2739222848)

[Portfólio 13](#__RefHeading___Toc12533_704678284)

[Agenda 13](#__RefHeading___Toc12535_704678284)

[Contato.java 13](#__RefHeading___Toc12537_704678284)

[GestorAgenda.java 14](#__RefHeading___Toc12539_704678284)

[Menus.java 15](#__RefHeading___Toc12541_704678284)

[Principal.java 16](#__RefHeading___Toc12543_704678284)

[Container\_component 17](#__RefHeading___Toc12545_704678284)

[ScrollPaneDemo.java 17](#__RefHeading___Toc12547_704678284)

[UsaJButton.java 17](#__RefHeading___Toc12549_704678284)

[UsaJCheckBox.java 18](#__RefHeading___Toc12551_704678284)

[UsaJComboBox.java 18](#__RefHeading___Toc12553_704678284)

[UsaJFrame.java 19](#__RefHeading___Toc12555_704678284)

[UsaJLabel.java 19](#__RefHeading___Toc12557_704678284)

[UsaJOptionPane.java 20](#__RefHeading___Toc12559_704678284)

[UsaJOptionPane.java 20](#__RefHeading___Toc12561_704678284)

[UsaJRadioButton.java 21](#__RefHeading___Toc12563_704678284)

[UsaJTextField.java 21](#__RefHeading___Toc12565_704678284)

[UtilizaBorderLayout.java 22](#__RefHeading___Toc12567_704678284)

[UtilizaCardLayout.java 23](#__RefHeading___Toc12569_704678284)

[UtilizaFlowLayout.java 24](#__RefHeading___Toc12571_704678284)

[UtilizaGridLayout.java 24](#__RefHeading___Toc12573_704678284)

[Exemplos 25](#__RefHeading___Toc12575_704678284)

[DivZero.java 25](#__RefHeading___Toc12577_704678284)

[ExcepcaoEscalada.java 26](#__RefHeading___Toc12579_704678284)

[ExcepcaoEscalada2.java 26](#__RefHeading___Toc12581_704678284)

[ExcepcaoEscalada2a.java 27](#__RefHeading___Toc12583_704678284)

[Exemplo1.java 27](#__RefHeading___Toc12585_704678284)

[Exemplo2.java 28](#__RefHeading___Toc12587_704678284)

[Exemplo2a.java 29](#__RefHeading___Toc12589_704678284)

[Exemplo3.java 29](#__RefHeading___Toc12591_704678284)

[TesteCatch.java 30](#__RefHeading___Toc12593_704678284)

[TesteCatch1.java 30](#__RefHeading___Toc12595_704678284)

[ExemplosGUI 31](#__RefHeading___Toc12597_704678284)

[DisplayImage.java 31](#__RefHeading___Toc12599_704678284)

[ExtendedFrame.java 32](#__RefHeading___Toc12601_704678284)

[FT3 33](#__RefHeading___Toc12603_704678284)

[ConversorTemperatura.java 33](#__RefHeading___Toc12605_704678284)

[ConversorTemperaturaVersaoAvancada.java 34](#__RefHeading___Toc12607_704678284)

[UtilizaBufferedInputStream.java 36](#__RefHeading___Toc12609_704678284)

[UtilizaFileInputOutputStream1.java 37](#__RefHeading___Toc12611_704678284)

[UtilizaBufferedInputOutputStream2.java 38](#__RefHeading___Toc12613_704678284)

[UtilizaBufferedInputOutputStream3.java 38](#__RefHeading___Toc12615_704678284)

[FT4 39](#__RefHeading___Toc12617_704678284)

[UtilizaDataInputOutputStream.java 39](#__RefHeading___Toc12619_704678284)

[FT5 40](#__RefHeading___Toc12621_704678284)

[UtilizaArray.java 40](#__RefHeading___Toc12623_704678284)

[FT6 41](#__RefHeading___Toc12625_704678284)

[UtilizaSequenceInputOutputStream.java 41](#__RefHeading___Toc12627_704678284)

[FT7 42](#__RefHeading___Toc12629_704678284)

[UtilizaFileReaderWriter.java 42](#__RefHeading___Toc12631_704678284)

[FT8 43](#__RefHeading___Toc12633_704678284)

[UtilizaRandomAccessFile.java 43](#__RefHeading___Toc12635_704678284)

[FT9 44](#__RefHeading___Toc12637_704678284)

[UtilizaFile1.java 44](#__RefHeading___Toc12639_704678284)

[UtilizaFile2.java 44](#__RefHeading___Toc12641_704678284)

[SistemaBancario 46](#__RefHeading___Toc12643_704678284)

[Banco.java 46](#__RefHeading___Toc12645_704678284)

[SistemaBancario.java 48](#__RefHeading___Toc12647_704678284)

[Streams 51](#__RefHeading___Toc12649_704678284)

[LeituraCaracteres.java 51](#__RefHeading___Toc12651_704678284)

[Projeto adicional 52](#__RefHeading___Toc12653_704678284)

[Bank 52](#__RefHeading___Toc12655_704678284)

[AccountManagement.java 52](#__RefHeading___Toc12657_704678284)

[ClientManagement.java 56](#__RefHeading___Toc12659_704678284)

[Account.java 58](#__RefHeading___Toc12661_704678284)

[AccountMovement.java 60](#__RefHeading___Toc12663_704678284)

[Client.java 61](#__RefHeading___Toc12665_704678284)

[ClientOptionsWindow.java 63](#__RefHeading___Toc12667_704678284)

[ControlPanel.java 64](#__RefHeading___Toc12669_704678284)

[CreateClientAccountWindow.java 67](#__RefHeading___Toc12671_704678284)

[DepositWindow.java 69](#__RefHeading___Toc12673_704678284)

[ListAccountMovementsWindow.java 70](#__RefHeading___Toc12675_704678284)

[ListClientAccountsWindow.java 71](#__RefHeading___Toc12677_704678284)

[ListClientsWindow.java 73](#__RefHeading___Toc12679_704678284)

[LoginWindow.java 75](#__RefHeading___Toc12681_704678284)

[RegisterClientWindow.java 77](#__RefHeading___Toc12683_704678284)

[WithdrawWindow.java 80](#__RefHeading___Toc12685_704678284)

[Main.java 81](#__RefHeading___Toc12687_704678284)

[Conclusão 82](#__RefHeading___Toc13159_2739222848)

[Anexos 84](#__RefHeading___Toc13161_2739222848)

[Características da classe JFRAME 84](#__RefHeading___Toc13163_2739222848)

[Constructor Summary 84](#__RefHeading___Toc13165_2739222848)

[Method Summary 84](#__RefHeading___Toc13167_2739222848)

[Características da classe JPANEL 87](#__RefHeading___Toc13169_2739222848)

[Constructor Summary 87](#__RefHeading___Toc13171_2739222848)

[Method Summary 87](#__RefHeading___Toc13173_2739222848)

[Características da classe JBUTTON 88](#__RefHeading___Toc13175_2739222848)

[Constructor Summary 88](#__RefHeading___Toc13177_2739222848)

[Method Summary 88](#__RefHeading___Toc13179_2739222848)

[Características da classe JTEXTFIELD 89](#__RefHeading___Toc13181_2739222848)

[Constructor Summary 89](#__RefHeading___Toc13183_2739222848)

[Method Summary 89](#__RefHeading___Toc13185_2739222848)

[Características da classe JLABEL 92](#__RefHeading___Toc13187_2739222848)

[Constructor Summary 92](#__RefHeading___Toc13189_2739222848)

[Method Summary 92](#__RefHeading___Toc13191_2739222848)

[Características da classe JCOMBOBOX 95](#__RefHeading___Toc13193_2739222848)

[Constructor Summary 95](#__RefHeading___Toc13195_2739222848)

[Method Summary 96](#__RefHeading___Toc13197_2739222848)

[Características da classe JRADIOBUTTON 102](#__RefHeading___Toc13199_2739222848)

[Constructor Summary 102](#__RefHeading___Toc13201_2739222848)

[Method Summary 102](#__RefHeading___Toc13203_2739222848)

[Características da classe JSCROLLPANE 103](#__RefHeading___Toc13205_2739222848)

[Constructor Summary 103](#__RefHeading___Toc13207_2739222848)

[Method Summary 103](#__RefHeading___Toc13209_2739222848)

[Características da classe JCHECKBOX 106](#__RefHeading___Toc13211_2739222848)

[Constructor Summary 106](#__RefHeading___Toc13213_2739222848)

[Method Summary 107](#__RefHeading___Toc13215_2739222848)

[Características da classe JOPTIONPANE 107](#__RefHeading___Toc13217_2739222848)

[Constructor Summary 107](#__RefHeading___Toc13219_2739222848)

[Method Summary 108](#__RefHeading___Toc13221_2739222848)

[Características da classe BUFFEREDINPUTSTREAM 115](#__RefHeading___Toc13223_2739222848)

[Constructor Summary 115](#__RefHeading___Toc13225_2739222848)

[Method Summary 115](#__RefHeading___Toc13227_2739222848)

[Características da classe BufferedOutputStream 116](#__RefHeading___Toc13229_2739222848)

[Constructor Summary 116](#__RefHeading___Toc13231_2739222848)

[Method Summary 116](#__RefHeading___Toc13233_2739222848)

[Referências Bibliográficas 117](#__RefHeading___Toc13235_2739222848)

# Agradecimentos

Não podia deixar de agradecer a algumas pessoas pela conclusão deste módulo. Estas são:

* A professora Matilde Vieira, pelos conhecimentos transmitidos.
* A minha família, pelo tempo que dispensei com a realização de trabalhos.
* A mim próprio, pela vontade de aprender, motivação e ambição.

# Índice de figuras

Índice de figuras

[Imagem 1: Ocorrencia de exceção. 8](#Imagem!0|sequence)

[Imagem 2: Interface Gráfica criada com JAVA 11](#Imagem!1|sequence)

# Introdução

O presente portfólio foi realizado no âmbito do módulo 11 da disciplina de programação e sistemas de informação, intitulado “Programação Orientada a Objetos Avançada”.

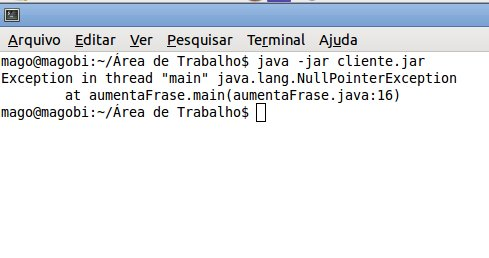
Neste módulo foram explorados os seguintes temas:

* Exceções
* *Streams*
* ARRAYLIST
* Interface Gráfica

# Desenvolvimento

## Exceções

As exceções ocorrem quando ocorre algo imprevisto na execução do programa. Podem ser causadas por culpa do utilizador, do programador, ou quando um recurso está inacessível (como por exemplo um ficheiro ou até mesmo conexão a um servidor).

  
Imagem 1: Ocorrencia de exceção.

Assim, as exceções devem ser tratadas, permitindo executar dadas ações quando estas exceções ocorrem. Para tratar exceções podemos contornar o problema, ou dar uma mensagem de alerta ao utilizador.

# Streams

Em JAVA, *streams* são fluxos de dados, tanto para leitura como escrita.

As *streams* são normalmente usadas para manipulação, leitura e escrita de ficheiros.

Como exemplos temos BUFFEREDINPUTSTREAM, BUFFEREDOUTPUTSTREAM, FILEINPUTSTREAM e FILEOUTPUTSTREAM.

Para a utilização de *streams* em JAVA, é indispensável também dominar o tema anterior: exceções. Isto porque a manipulação de ficheiros pode causar uma infinidade de exceções, como por exemplo causadas pela não permissão para criar e/ou editar ficheiros, tipo de dados lido e/ou escrito inválido, ficheiro não encontrado, etc.

# ArrayList

ARRAYLIST é uma classe JAVA para a criação e manipulação de listasdinâmicas.

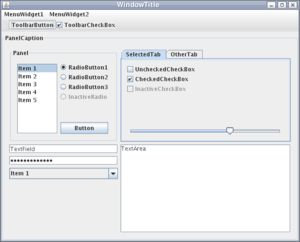
A diferença entre um vetor convencional e uma lista do tipo ARRAYLIST em JAVA é que o tamanho de um vetor não pode ser alterado depois da instanciação, enquanto que no caso de uma lista ARRAYLIST podem ser adicionados, removidos e editados elementos quando se quiser.

# Interface Gráfica (GUI)

Interface gráfica é um tipo de interface do utilizador que permite a interação com dispositivos digitais por meio de elementos gráficos, como ícones, imagens, botões, entre outros, em contraste com a interface de linha de comandos (CLI).

No desenvolvimento de aplicações JAVA, as interfaces gráficas são maioritariamente criadas com recurso ao pacote SWING, atualmente presente nas versões mais recentes do JAVA.

O pacote SWING contém bibliotecas que permitem a criação, edição e manipulação de eventos de elementos gráficos como botões, tabelas, rótulos, caixas de texto, entre outros.

  
Imagem 2: Interface Gráfica criada com JAVA

# Portfólio

## Agenda

### Contato.java

//

// Copyright (c) Carlos Tojal 2020

// Agenda

// Contato.java

//

public class Contato {

// Atributos

private String nome;

private String telefone;

private String email;

// Construtores

public Contato() {

}

public Contato(String nome, String telefone, String email) {

this.nome = nome;

this.telefone = telefone;

this.email = email;

}

// Getters e Setters

public String getNome() {

return nome;

}

public String getTelefone() {

return telefone;

}

public String getEmail() {

return email;

}

public void setNome(String nome) {

this.nome = nome;

}

public void setTelefone(String telefone) {

this.telefone = telefone;

}

public void setEmail(String email) {

this.email = email;

}

}

### GestorAgenda.java

import java.util.ArrayList;

import java.util.Iterator;

//

// Copyright (c) Carlos Tojal 2020

// Agenda

// GestorAgenda.java

//

public class GestorAgenda {

// Construtor

public GestorAgenda() {

}

// Métodos

public void adicionarContato(Contato contato, ArrayList<Contato> agenda) {

if(!agenda.contains(contato))

agenda.add(contato);

else

System.out.println("O contato já existe na agenda.");

}

public void listarContato(Contato contato) {

System.out.println("\n\n\*\* Contato \*\*");

System.out.println("Nome: " + contato.getNome());

System.out.println("Telefone: " + contato.getTelefone());

System.out.println("Email: " + contato.getEmail());

}

public void listarAgenda(ArrayList<Contato> agenda) {

for(int i = 0; i < agenda.size(); i++) {

listarContato(agenda.get(i));

}

}

}

### Menus.java

//

// Copyright (c) Carlos Tojal 2020

// Agenda

// Menus.java

//

import java.util.ArrayList;

import java.util.Scanner;

public class Menus {

// Construtor

public Menus() {

}

public int menu() {

Scanner scanner = new Scanner(System.in);

int opt = 0;

do {

System.out.println("\n\*\* Agenda \*\*\n");

System.out.println("1. Adicionar Contato");

System.out.println("2. Listar Contatos");

System.out.println("3. Remover Contato");

System.out.println("0. Sair\n");

System.out.print("Opção: ");

opt = scanner.nextInt();

} while(opt < 0 || opt > 3);

return opt;

}

public void menuAdicionarContato(ArrayList<Contato> agenda) {

Scanner scanner = new Scanner(System.in);

GestorAgenda gestorAgenda = new GestorAgenda();

Contato contato = new Contato();

System.out.println("\n\*\* Adicionar Contato \*\*\n");

System.out.print("Nome: ");

contato.setNome(scanner.nextLine());

System.out.print("Telefone: ");

contato.setTelefone(scanner.nextLine());

System.out.print("Email: ");

contato.setEmail(scanner.nextLine());

gestorAgenda.adicionarContato(contato, agenda);

}

public void menuRemoverContato(ArrayList<Contato> agenda) {

GestorAgenda gestorAgenda = new GestorAgenda();

String query;

boolean found = false;

gestorAgenda.listarAgenda(agenda);

System.out.println("\nTelefone ou email a remover: ");

query = new Scanner(System.in).nextLine();

for(int i = 0; i < agenda.size(); i++) {

if(String.valueOf(agenda.get(i).getTelefone()).equals(query) || agenda.get(i).getEmail().equals(query)) {

agenda.remove(agenda.get(i));

found = true;

}

}

if(found)

System.out.println("Contato removido com sucesso.");

else

System.out.println("Contato não encontrado.");

}

}

### Principal.java

//

// Copyright (c) Carlos Tojal 2020

// Agenda

// Principal.java

//

import java.util.ArrayList;

public class Principal {

static ArrayList<Contato> agenda = new ArrayList<Contato>();

static Menus menus = new Menus();

static GestorAgenda gestorAgenda = new GestorAgenda();

public static void main(String[] args) {

int opt;

do {

opt = menus.menu();

switch(opt) {

case 1:

menus.menuAdicionarContato(agenda);

break;

case 2:

gestorAgenda.listarAgenda(agenda);

break;

case 3:

menus.menuRemoverContato(agenda);

break;

}

}while(opt > 0);

}

}

## Container\_component

### ScrollPaneDemo.java

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

public class ScrollPaneDemo extends JFrame {

public ScrollPaneDemo() {

super("ScrollPane Demo");

ImageIcon img = new ImageIcon("Imagem.jpg");

JScrollPane png = new JScrollPane(new JLabel(img));

getContentPane().add(png);

setSize(300,250);

setVisible(true);

}

public static void main(String[] args) {

new ScrollPaneDemo();

}

}

### UsaJButton.java

import javax.swing.JFrame;

import javax.swing.JButton;

public class UsaJButton {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("JButton");

janela.setSize(350,150);

janela.setLocation(50,50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JButton botao = new JButton("OK");

janela.add(botao);

janela.setVisible(true);

}

}

### UsaJCheckBox.java

import javax.swing.JFrame;

import javax.swing.JCheckBox;

import java.awt.FlowLayout;

public class UsaJCheckBox {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("JCheckBox");

janela.setSize(350,150);

janela.setLocation(50,50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setLayout(new FlowLayout());

JCheckBox caixaVerificacao1 = new JCheckBox("Branco");

JCheckBox caixaVerificacao2 = new JCheckBox("Preto");

JCheckBox caixaVerificacao3 = new JCheckBox("Amarelo");

janela.add(caixaVerificacao1);

janela.add(caixaVerificacao2);

janela.add(caixaVerificacao3);

janela.setVisible(true);

}

}

### UsaJComboBox.java

//

// Copyright (c) Carlos Tojal 2020

// Container\_Component

// UsaJComboBox.java

//

import javax.swing.JFrame;

import javax.swing.JComboBox;

import javax.swing.JLabel;

import java.awt.FlowLayout;

public class UsaJComboBox {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("JComboBox");

janela.setSize(350, 150);

janela.setLocationRelativeTo(null);

// janela.setLocation(50, 50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setLayout(new FlowLayout());

JLabel rotulo = new JLabel("Escreva o seu nome: ");

JComboBox<String> caixaCombinacao = new JComboBox<String>();

janela.add(rotulo);

caixaCombinacao.addItem("Branco");

caixaCombinacao.addItem("Preto");

caixaCombinacao.addItem("Amarelo");

janela.add(caixaCombinacao);

janela.setVisible(true);

}

}

### UsaJFrame.java

//

// Copyright (c) Carlos Tojal 2020

// Container\_Component

// UsaJFrame.java

//

import javax.swing.JFrame;

public class UsaJFrame {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("JFrame");

janela.setSize(350, 150);

janela.setLocation(50, 50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setVisible(true);

}

}

### UsaJLabel.java

//

// Copyright (c) Carlos Tojal 2020

// Container\_Component

// UsaJLabel.java

//

import javax.swing.JFrame;

import javax.swing.JLabel;

public class UsaJLabel {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("JPanel");

janela.setSize(350, 150);

janela.setLocationRelativeTo(null);

// janela.setLocation(50, 50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JLabel rotulo = new JLabel("Escreva o seu nome: ");

janela.add(rotulo);

janela.setVisible(true);

}

}

### UsaJOptionPane.java

import javax.swing.JOptionPane;

public class UsaJOptionPane {

public static void main (String[] args) {

JOptionPane.showMessageDialog(null,"Obrigado por ter utilizado este programa");

JOptionPane.showInputDialog("Escreva o seu nome");

JOptionPane.showConfirmDialog(null,"Deseja guardar as alterações?");

}

}

### UsaJOptionPane.java

//

// Copyright (c) Carlos Tojal 2020

// Container\_Component

// UsaJPanel.java

//

import javax.swing.JFrame;

import javax.swing.JPanel;

public class UsaJPanel {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("JPanel");

janela.setSize(350, 150);

janela.setLocationRelativeTo(null);

// janela.setLocation(50, 50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel painel = new JPanel();

janela.add(painel);

janela.setVisible(true);

}

}

### UsaJRadioButton.java

import javax.swing.JFrame;

import javax.swing.JRadioButton;

import java.awt.FlowLayout;

public class UsaJRadioButton {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("JRadioButton");

janela.setSize(350,150);

janela.setLocation(50,50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setLayout(new FlowLayout());

JRadioButton botaoRadio1 = new JRadioButton("Branco");

JRadioButton botaoRadio2 = new JRadioButton("Preto");

JRadioButton botaoRadio3 = new JRadioButton("Amarelo");

janela.add(botaoRadio1);

janela.add(botaoRadio2);

janela.add(botaoRadio3);

janela.setVisible(true);

}

}

### UsaJTextField.java

//

// Copyright (c) Carlos Tojal 2020

// Container\_Component

// UsaJTextField.java

//

import javax.swing.JFrame;

import javax.swing.JTextField;

import javax.swing.JLabel;

import java.awt.FlowLayout;

public class UsaJTextField {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("JTextField");

janela.setSize(350, 150);

janela.setLocationRelativeTo(null);

// janela.setLocation(50, 50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setLayout(new FlowLayout());

JLabel rotulo = new JLabel("Escreva o seu nome: ");

JTextField caixaTexto = new JTextField(10);

janela.add(rotulo);

janela.add(caixaTexto);

janela.setVisible(true);

}

}

### UtilizaBorderLayout.java

import java.awt.BorderLayout;

import javax.swing.JFrame;

import javax.swing.JButton;

public class UtilizaBorderLayout {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("BorderLayout");

janela.setSize(350,150);

janela.setLocation(50,50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setLayout(new BorderLayout());

JButton botaoNorte = new JButton("Norte");

JButton botaoSul = new JButton("Sul");

JButton botaoOeste = new JButton("Oeste");

JButton botaoEste = new JButton("Este");

JButton botaoCentro = new JButton("Centro");

janela.add("North", botaoNorte);

janela.add("South", botaoSul);

janela.add("West", botaoOeste);

janela.add("East", botaoEste);

janela.add("Center", botaoCentro);

janela.setVisible(true);

}

}

### UtilizaCardLayout.java

import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.CardLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JLabel;

import javax.swing.JButton;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class UtilizaCardLayout implements ActionListener {

JFrame janela = new JFrame();

JPanel painelBotoes = new JPanel();

JButton botao1 = new JButton("Painel 1");

JButton botao2 = new JButton("Painel 2");

JPanel painelRotulo1 = new JPanel();

JPanel painelRotulo2 = new JPanel();

JLabel rotulo1 = new JLabel("Painel 1");

JLabel rotulo2 = new JLabel("Painel 2");

JPanel painelRotulos = new JPanel();

public static void main(String[] args) {

new UtilizaCardLayout();

}

private UtilizaCardLayout() {

janela.setTitle("Primeira aplicação gráfica - CardLayout");

janela.setSize(350, 150);

janela.setLocation(50, 50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setLayout(new BorderLayout());

painelBotoes.setLayout(new GridLayout(1, 2));

painelBotoes.add(botao1);

painelBotoes.add(botao2);

painelRotulo1.add(rotulo1);

painelRotulo2.add(rotulo2);

painelRotulos.setLayout(new CardLayout());

painelRotulos.add(painelRotulo1, "p1");

painelRotulos.add(painelRotulo2, "p2");

janela.add("North", painelBotoes);

janela.add("South", painelRotulos);

botao1.addActionListener(this);

botao2.addActionListener(this);

janela.setVisible(true);

}

public void actionPerformed(ActionEvent e) {

CardLayout cl = (CardLayout) painelRotulos.getLayout();

if (e.getSource() == botao1)

cl.show(painelRotulos, "p1");

if (e.getSource() == botao2)

cl.show(painelRotulos, "p2");

}

}

### UtilizaFlowLayout.java

import java.awt.FlowLayout;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JButton;

public class UtilizaFlowLayout {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("FlowLayout");

janela.setSize(350,150);

janela.setLocation(50,50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setLayout(new FlowLayout());

JLabel rotulo = new JLabel("Escreva o seu nome: ");

JTextField caixaTexto = new JTextField(10);

JButton botao = new JButton("OK");

janela.add(rotulo);

janela.add(caixaTexto);

janela.add(botao);

janela.setVisible(true);

}

}

### UtilizaGridLayout.java

import java.awt.GridLayout;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JComboBox;

import javax.swing.JButton;

public class UtilizaGridLayout {

public static void main(String[] args) {

JFrame janela = new JFrame();

janela.setTitle("GridLayout");

janela.setSize(350,150);

janela.setLocation(50,50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setLayout(new GridLayout(3,2));

JLabel rotulo1 = new JLabel("Seleccione um fruto: ");

JTextField caixaTexto = new JTextField(10);

JLabel rotulo2 = new JLabel ("Seleccione uma bebida: ");

JComboBox<String> caixaCombinacao = new JComboBox<String>();

caixaCombinacao.addItem("Água");

caixaCombinacao.addItem("Leite");

caixaCombinacao.addItem("Vinho");

JButton botao = new JButton("OK");

janela.add(rotulo1);

janela.add(caixaTexto);

janela.add(rotulo2);

janela.add(caixaCombinacao);

janela.add(botao);

janela.setVisible(true);

}

}

## Exemplos

### DivZero.java

//

// Copyright (c) Carlos Tojal 2019

// Exemplo1

// DivZero.java

//

public class DivZero {

public static void main(String[] args) {

int a = 1, b = 0;

System.out.println(a/b);

System.out.println("Divisão por zero");

System.out.println("Fim do programa");

}

}

/\*

public class DivZero {

public static void main(String[] args) {

int a = 1, b = 0;

try {

System.out.println(a/b);

} catch(Exception e) {

System.out.println("Divisão por zero");

}

System.out.println("Fim do programa");

}

}\*/

### ExcepcaoEscalada.java

//

// Copyright (c) Carlos Tojal 2019

// Exemplo1

// ExcepcaoEscalada.java

//

public class ExcepcaoEscalada {

public static void metodoComErro() {

System.out.println(3 / 0);

}

public static void main(String[] args) {

try {

metodoComErro();

} catch(RuntimeException rte) {

rte.printStackTrace();

}

System.out.println("Fim do Programa");

}

}

### ExcepcaoEscalada2.java

//

// Copyright (c) Carlos Tojal 2019

// Exemplo1

// ExcepcaoEscalada2.java

//

public class ExcepcaoEscalada2 {

public static void main(String[] args) throws java.io.IOException {

int n = 0;

n = System.in.read();

System.out.println("n=='a'" + (n=='a'));

System.out.println("n=" + n);

}

}

### ExcepcaoEscalada2a.java

//

// Copyright (c) Carlos Tojal 2019

// Exemplo1

// ExcepcaoEscalada2a.java

//

public class ExcepcaoEscalada2a {

public static void main(String[] args) {

int n = 0;

try {

n = System.in.read();

} catch(Exception e) {

e.printStackTrace();

} finally {

System.out.println("n=='a'" + (n=='a'));

System.out.println("n=" + n);

}

}

}

### Exemplo1.java

//

// Copyright (c) Carlos Tojal 2020

// Exemplo1

//

import java.util.Iterator;

import java.util.LinkedList;

import java.util.List;

// Iterator - Interface do pacote java.util.

// Permite percorrer as coleções do framework Collections que implementam a interface "Collection". Fornece os métodos next(), hasNext() e ermove().

public class Exemplo1 {

public static void main(String[] args) {

List<String> list = new LinkedList<>();

list.add("Welcome");

list.add("to");

list.add("our");

list.add("city");

System.out.println("The list is given as: " + list);

Iterator<String> itr = list.iterator();

while(itr.hasNext()) {

System.out.println(itr.next());

}

itr.remove();

System.out.println("After the remove() method is called: " + list);

}

}

### Exemplo2.java

//

// Copyright (c) Carlos Tojal 2020

// Exemplo2

//

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

// For-Each consiste num ciclo for adaptado a collections.

// Percorre todos os elementos de qualquer collection do framework "Collection".

public class Exemplo2 {

public static void main(String[] args) {

List<String> listaDeNomes = new ArrayList<String>();

listaDeNomes.add("Gustavo");

listaDeNomes.add("Maria");

listaDeNomes.add("José");

listaDeNomes.add("João");

listaDeNomes.add("Ana");

Iterator<String> iteratorDeNomes = listaDeNomes.iterator();

List<String> listaConvertidaDoIterator = new ArrayList<>();

while(iteratorDeNomes.hasNext()) {

listaConvertidaDoIterator.add(iteratorDeNomes.next());

}

listaConvertidaDoIterator.forEach(System.out::println);

}

}

### Exemplo2a.java

//

// Copyright (c) Carlos Tojal 2020

// Exemplo3

//

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

public class Exemplo2a {

public static void main(String[] args) {

List<String> listaDeNomes = new ArrayList<String>();

listaDeNomes.add("Gustavo");

listaDeNomes.add("Maria");

listaDeNomes.add("José");

listaDeNomes.add("João");

listaDeNomes.add("Ana");

Iterator<String> iteratorDeNomes = listaDeNomes.iterator();

List<String> listaConvertidaDoIterator = new ArrayList<>();

iteratorDeNomes.forEachRemaining(n -> listaConvertidaDoIterator.add(n));

iteratorDeNomes.forEachRemaining(listaConvertidaDoIterator::add);

listaConvertidaDoIterator.forEach(System.out::println);

}

}

### Exemplo3.java

//

// Copyright (c) Carlos Tojal 2020

// Exemplo3

//

import java.util.ArrayList;

public class Exemplo3 {

public static void main(String[] args) {

ArrayList<String> books = new ArrayList<String>();

books.add("C");

books.add("Java");

books.add("PHP");

for(String obj: books) {

System.out.println(obj);

books.add("C++");

}

}

}

### TesteCatch.java

//

// Copyright (c) Carlos Tojal 2019

// Exemplo1

// TesteCatch.java

//

public class TesteCatch {

public static void main(String[] args) {

int a;

try {

a = Integer.parseInt("123a");

} catch(NumberFormatException nfe) {

System.out.println("NumberFormatException");

} catch(ArithmeticException ae) {

System.out.println("ArithmeticExcetion");

} catch(ArrayIndexOutOfBoundsException aioobe) {

System.out.println("ArrayIndexOutOfBoundsException");

} catch(Exception e) {

System.out.println("Bloco Exception");

} finally {

System.out.println("Bloco Finally");

}

System.out.println("Fim do programa");

}

}

### TesteCatch1.java

//

// Copyright (c) Carlos Tojal 2019

// Exemplo1

// TesteCatch1.java

//

public class TesteCatch1 {

public static void main(String[] args) {

int[] a = new int[6];

try {

a[8] = 12;

} catch(NumberFormatException nfe) {

System.out.println("NumberFormatException");

} catch(ArithmeticException ae) {

System.out.println("ArithmeticExcetion");

} catch(ArrayIndexOutOfBoundsException aioobe) {

System.out.println("ArrayIndexOutOfBoundsException");

} catch(Exception e) {

System.out.println("Bloco Exception");

} finally {

System.out.println("Bloco Finally");

}

System.out.println("Fim do programa");

}

}

## ExemplosGUI

### DisplayImage.java

//

// Copyright (c) Carlos Tojal 2020

// ExemplosGUI

// DisplayImage.java

//

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JLabel;

public class DisplayImage {

static String IMG\_PATH = "Imagem.jpg";

public static void main(String[] args) throws IOException {

JFrame frame = new JFrame();

BufferedImage img = ImageIO.read(new File(IMG\_PATH));

ImageIcon icon = new ImageIcon(img);

JLabel label = new JLabel(icon);

frame.add(label);

frame.setExtendedState(frame.getExtendedState() | JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

}

}

### ExtendedFrame.java

//

// Copyright (c) Carlos Tojal 2020

// ExemplosGUI

// ExtendedFrame.java

//

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;;

public class ExtendedFrame {

public static void main(String[] args) {

new ExtendedFrame();

}

public ExtendedFrame() {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch(ClassNotFoundException | InstantiationException | IllegalAccessException | UnsupportedLookAndFeelException e) {

}

JFrame frame = new JFrame();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

});

}

}

## FT3

### ConversorTemperatura.java

//

// Copyright (c) Carlos Tojal 2020

// FT3

// ConversorTemperatura.java

//

import java.awt.GridLayout;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JRadioButton;

import javax.swing.ButtonGroup;

import javax.swing.JPanel;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.JOptionPane;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class ConversorTemperatura implements ActionListener {

JFrame janela = new JFrame();

JLabel rotulo1 = new JLabel("Tipo de conversão: ");

JRadioButton botaoRadio1 = new JRadioButton("ºC->ºF");

JRadioButton botaoRadio2 = new JRadioButton("ºF->ºC");

ButtonGroup grupo = new ButtonGroup();

JPanel painel = new JPanel();

JLabel rotulo2 = new JLabel("Valor a converter: ");

JTextField caixaTexto1 = new JTextField(5);

JButton botao1 = new JButton("Converter");

private ConversorTemperatura() {

janela.setTitle("Conversor Temperatura");

janela.setSize(300, 125);

janela.setLocation(50, 50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setLayout(new GridLayout(3, 2));

janela.add(rotulo1);

painel.setLayout(new GridLayout(1, 2));

painel.add(botaoRadio1);

painel.add(botaoRadio2);

janela.add(painel);

janela.add(rotulo2);

janela.add(caixaTexto1);

janela.add(botao1);

grupo.add(botaoRadio1);

grupo.add(botaoRadio2);

botao1.addActionListener(this);

janela.setVisible(true);

}

public static void main(String[] args) {

new ConversorTemperatura();

}

public void actionPerformed(ActionEvent e) {

double resultado;

String mensagem = "";

if(e.getSource() == botao1) {

if(botaoRadio1.isSelected()) {

resultado = Double.parseDouble(caixaTexto1.getText()) \* 9 / 5 + 32;

mensagem = Double.toString(resultado) + "ºF";

} else if(botaoRadio2.isSelected()) {

resultado = Double.parseDouble(caixaTexto1.getText()) - 32 \* 5 / 9;

mensagem = Double.toString(resultado) + "ºC";

} else {

mensagem = "Não selecionou nenhuma opção!";

}

}

JOptionPane.showMessageDialog(null, mensagem);

}

}

### ConversorTemperaturaVersaoAvancada.java

//

// Copyright (c) Carlos Tojal 2020

// FT3

// ConversorTemperaturaVersaoAvancada.java

//

import java.awt.GridLayout;

import java.awt.FlowLayout;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JRadioButton;

import javax.swing.ButtonGroup;

import javax.swing.JPanel;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.JOptionPane;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.event.ItemListener;

import java.awt.event.ItemEvent;

public class ConversorTemperaturaVersaoAvancada implements ActionListener, ItemListener {

JFrame janela = new JFrame();

JLabel rotulo1 = new JLabel("Tipo de conversão: ");

JRadioButton botaoRadio1 = new JRadioButton("ºC->ºF");

JRadioButton botaoRadio2 = new JRadioButton("ºF->ºC");

ButtonGroup grupo = new ButtonGroup();

JPanel painel = new JPanel();

JLabel rotulo2 = new JLabel("Valor a converter: ");

JTextField caixaTexto1 = new JTextField(5);

JButton botao1 = new JButton("Converter");

private ConversorTemperaturaVersaoAvancada() {

janela.setTitle("Conversor Temperatura");

janela.setSize(300, 125);

janela.setLocation(50, 50);

janela.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

janela.setLayout(new GridLayout(3, 2));

janela.add(rotulo1);

painel.setLayout(new GridLayout(1, 2));

painel.add(botaoRadio1);

painel.add(botaoRadio2);

janela.add(painel);

janela.add(rotulo2);

janela.add(caixaTexto1);

janela.add(botao1);

grupo.add(botaoRadio1);

grupo.add(botaoRadio2);

botaoRadio1.addItemListener(this);

botaoRadio2.addItemListener(this);

botao1.addActionListener(this);

janela.setVisible(true);

}

public static void main(String[] args) {

new ConversorTemperaturaVersaoAvancada();

}

public void actionPerformed(ActionEvent e) {

double resultado;

String mensagem = "";

if(e.getSource() == botao1) {

if(botaoRadio1.isSelected()) {

resultado = Double.parseDouble(caixaTexto1.getText()) \* 9 / 5 + 32;

mensagem = Double.toString(resultado) + "ºF";

} else if(botaoRadio2.isSelected()) {

resultado = Double.parseDouble(caixaTexto1.getText()) - 32 \* 5 / 9;

mensagem = Double.toString(resultado) + "ºC";

} else {

mensagem = "Não selecionou nenhuma opção!";

}

}

JOptionPane.showMessageDialog(null, mensagem);

}

public void itemStateChanged(ItemEvent e) {

if(e.getSource() == botaoRadio1) {

if(e.getStateChange() == ItemEvent.SELECTED)

JOptionPane.showMessageDialog(null, "ºC->ºF");

}

if(e.getSource() == botaoRadio2) {

if(e.getStateChange() == ItemEvent.SELECTED)

JOptionPane.showMessageDialog(null, "ºF->ºC");

}

}

}

### UtilizaBufferedInputStream.java

//

// Copyright (c) Carlos Tojal 2020

// FT3

// UtilizaBufferedInputOutputStream1.java

//

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class UtilizaBufferedInputOutputStream1 {

static FileInputStream fis;

static FileOutputStream fos;

static BufferedInputStream bis;

static BufferedOutputStream bos;

static int conteudo;

public static void main(String[] args) {

try {

fis = new FileInputStream("Imagem1.jpg");

bis = new BufferedInputStream(fis);

fos = new FileOutputStream("Imagem3.jpg");

bos = new BufferedOutputStream(fos);

while((conteudo=bis.read()) != -1)

bos.write(conteudo);

fis.close();

bis.close();

fos.close();

bos.close();

} catch(FileNotFoundException fnfe) {

System.out.println("Ficheiro não encontrado.");

} catch(IOException ioe) {

System.out.println("Não foi possível ler o ficheiro.");

}

}

}

### UtilizaFileInputOutputStream1.java

//

// Copyright (c) Carlos Tojal 2020

// FT3

// UtilizaFileOutputStream1.java

//

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class UtilizaFileInputOutputStream1 {

static FileInputStream fis;

static FileOutputStream fos;

static int conteudo;

public static void main(String[] args) {

try {

fis = new FileInputStream("Inteiros1.txt");

fos = new FileOutputStream("Inteiros2.txt");

while((conteudo=fis.read()) != -1)

fos.write(conteudo);

fis.close();

fos.close();

} catch(FileNotFoundException fnfe) {

System.out.println("Ficheiro não encontrado.");

} catch(IOException ioe) {

System.out.println("Não foi possível ler o ficheiro.");

}

}

}

### UtilizaBufferedInputOutputStream2.java

//

// Copyright (c) Carlos Tojal 2020

// FT3

// UtilizaFileOutputStream2.java

//

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class UtilizaFileInputOutputStream2 {

static FileInputStream fis;

static FileOutputStream fos;

static int conteudo;

public static void main(String[] args) {

try {

fis = new FileInputStream("Caracteres1.txt");

fos = new FileOutputStream("Caracteres2.txt");

while((conteudo=fis.read()) != -1)

fos.write(conteudo);

fis.close();

fos.close();

} catch(FileNotFoundException fnfe) {

System.out.println("Ficheiro não encontrado.");

} catch(IOException ioe) {

System.out.println("Não foi possível ler o ficheiro.");

}

}

}

### UtilizaBufferedInputOutputStream3.java

//

// Copyright (c) Carlos Tojal 2020

// FT3

// UtilizaFileOutputStream3.java

//

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class UtilizaFileInputOutputStream3 {

static FileInputStream fis;

static FileOutputStream fos;

static int conteudo;

public static void main(String[] args) {

try {

fis = new FileInputStream("Imagem1.jpg");

fos = new FileOutputStream("Imagem2.jpg");

while((conteudo=fis.read()) != -1)

fos.write(conteudo);

fis.close();

fos.close();

} catch(FileNotFoundException fnfe) {

System.out.println("Ficheiro não encontrado.");

} catch(IOException ioe) {

System.out.println("Não foi possível ler o ficheiro.");

}

}

}

## FT4

### UtilizaDataInputOutputStream.java

//

// Copyright (c) Carlos Tojal 2020

// FT4

// UtilizaDataInputOutputStream1.java

//

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class UtilizaDataInputOutputStream1 {

static FileOutputStream fos;

static DataOutputStream dos;

public static void main(String[] args) {

float j = 0.5f;

try {

fos = new FileOutputStream("Floats.txt");

dos = new DataOutputStream(fos);

for(int i = 0; i < 3; i++) {

System.out.println(i + j);

dos.writeFloat(j + i);

}

} catch(FileNotFoundException fnfe) {

System.out.println("Ficheiro não encontrado.");

} catch(IOException ioe) {

System.out.println("Não foi possível ler o ficheiro.");

}

}

}

## FT5

### UtilizaArray.java

//

// Copyright (c) Carlos Tojal 2020

// FT5

// UtilizaArray.java

//

import java.util.ArrayList;

import java.lang.IndexOutOfBoundsException;

public class UtilizaArray {

public static void main(String[] args) {

ArrayList<Integer> arr = new ArrayList<Integer>();

arr.add(Integer.valueOf(2));

arr.add(4);

arr.add(6);

arr.add(8);

arr.add(10);

// System.out.println(arr.get(2));

try {

System.out.println(arr.get(5));

} catch(IndexOutOfBoundsException ioobe) {

System.out.println("O indice passado como parametro num dos metodos que esta a utilizar esta fora dos limites");

}

}

}

## FT6

### UtilizaSequenceInputOutputStream.java

//

// Copyright (c) Carlos Tojal 2020

// FT6

// UtilizaSequenceInputStream.java

//

import java.io.File;

import java.io.InputStream;

import java.io.FileInputStream;

import java.io.SequenceInputStream;

import java.io.FileWriter;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.FileNotFoundException;

import java.lang.Exception;

public class UtilizaSequenceInputStream {

public static void main(String[] args) {

File f;

InputStream is1;

InputStream is2;

SequenceInputStream sis;

FileWriter fw;

BufferedWriter bw;

double num = 0;

try {

is1 = new FileInputStream("Inteiros1.txt");

is2 = new FileInputStream("Inteiros2.txt");

sis = new SequenceInputStream(is1, is2);

num = sis.read();

sis.close();

} catch (IOException ioe) {

System.out.println("Erro na leitura.");

} catch (Exception e) {

System.out.println("Erro.");

}

try {

f = new File("Inteiros1e2.txt");

fw = new FileWriter(f);

bw = new BufferedWriter(fw);

bw.write(String.valueOf(num));

bw.close();

} catch (IOException ioe) {

System.out.println("Erro na escrita.");

} catch (Exception e) {

System.out.println("Erro.");

}

}

}

## FT7

### UtilizaFileReaderWriter.java

//

// Copyright (c) Carlos Tojal 2020

// FT7

// UtilizaFileReaderWriter.java

//

import java.io.Reader;

import java.io.FileReader;

import java.io.BufferedReader;

import java.io.Writer;

import java.io.FileWriter;

import java.io.BufferedWriter;

import java.io.IOException;

import java.lang.Exception;

public class UtilizaFileReaderWriter {

public static void main(String[] args) {

Writer w;

BufferedWriter bw;

Reader r;

BufferedReader br;

try {

w = new FileWriter("Strings.txt");

bw = new BufferedWriter(w);

for(int i = 0; i < 5; i++)

bw.write(String.valueOf(i+1) + " linha\n");

bw.close();

} catch(IOException ioe) {

System.out.println("Erro na escrita do ficheiro.");

} catch(Exception e) {

System.out.println("Erro.");

}

try {

String line;

r = new FileReader("Strings.txt");

br = new BufferedReader(r);

while((line = br.readLine()) != null)

System.out.println(line);

} catch(IOException ioe) {

System.out.println("Erro na leitura do ficheiro.");

} catch(Exception e) {

System.out.println("Erro.");

}

}

}

## FT8

### UtilizaRandomAccessFile.java

//

// Copyright (c) Carlos Tojal 2020

// FT8

// UtilizaRandomAccessFile.java

//

import java.io.File;

import java.io.RandomAccessFile;

import java.io.IOException;

import java.lang.Exception;

public class UtilizaRandomAccessFile {

public static void main(String[] args) {

int [] arr = new int[] {1, 2, 3, 4, 5, 6, 7, 8};

try {

File f = new File("numeros.txt");

RandomAccessFile raf = new RandomAccessFile(f, "rw");

for (int i = 0; i < 8; i++)

raf.write(arr[i]);

System.out.println("Posicao no ficheiro: " + raf.getFilePointer() + " bytes");

raf.seek(16);

raf.seek(20);

System.out.println(raf.read());

raf.close();

} catch(IOException ioe) {

System.out.println("Erro na escrita do ficheiro.");

} catch(Exception e) {

System.out.println("Erro.");

}

}

}

## FT9

### UtilizaFile1.java

//

// Copyright (c) Carlos Tojal 2020

// FT9

// UtilizaFile1.java

//

import java.io.File;

import java.io.IOException;

import java.lang.Exception;

public class UtilizaFile1 {

public static void main(String[] args) {

File file = new File(".." + File.separatorChar + "FT9" + File.separatorChar + "UtilizaFile1.java");

if(file.exists()) {

System.out.println("Nome: " + file.getName());

System.out.println("Diretorio: " + file.getParent());

System.out.println("Diretorio completo: " + file.getAbsoluteFile().getParent());

} else {

System.out.println("O ficheiro especificado nao existe");

}

}

}

### UtilizaFile2.java

//

// Copyright (c) Carlos Tojal 2020

// FT9

// UtilizaFile2.java

//

import java.io.File;

import java.io.IOException;

import java.lang.Exception;

public class UtilizaFile2 {

public static void main(String[] args) {

try {

File f1 = new File("Inteiros.txt");

File f2, f3, f4, f5;

if (f1.exists()) {

f2 = new File("NumsInteiros.txt");

f1.renameTo(f2);

f1.delete();

}

f3 = new File("LingProg");

f3.mkdir();

if (f3.exists()) {

f4 = new File("LingProg" + File.separatorChar + "Modulo11.txt");

f5 = new File("LingProg" + File.separatorChar + "Modulo12.txt");

f4.createNewFile();

f5.createNewFile();

}

String[] list = f3.list();

for(String filename: list)

System.out.println(filename);

} catch(IOException ioe) {

System.out.println("Erro no acesso aos diretorios e/ou ficheiros.");

} catch(Exception e) {

System.out.println("Erro.");

}

}

}

## SistemaBancario

### Banco.java

//

// Copyright (c) Carlos Tojal 2020

// SistemaBancario

// Banco.java

//

public class Banco

{

private int senha;

private double saldo,levantamento,cpmf,limite;

private String nome,sobrenome, genero;

public Banco(String nom, String sobre)

{

this.nome = nome;

this.sobrenome= sobre;

}

public Banco(int sen)

{

this.senha = sen;

}

public Banco(double sald, double lev, double cp, double lm)

{

if (sald > 0.00 && lev > 0.00)

this.saldo = sald;

this.levantamento = lev;

this.cpmf = cp;

this.limite = lm;

}

public void setNome(String n)

{

this.nome = n;

}

public void setSobre(String s)

{

this.sobrenome = s;

}

public void setGenero(String g) {

this.genero = g;

}

public void setSaldo(double sa)

{

this.saldo = sa;

}

public void setLevantamento(double lev)

{

this.levantamento = lev;

}

public void setCpmf(double cm)

{

this.cpmf = cm;

}

public void setLimite(double lt)

{

this.limite = lt;

}

public void setSenha(int sh)

{

this.senha = sh;

}

//exibe o valor armazenado no set

public String getNome()

{

return nome;

}

public String getSobre()

{

return sobrenome;

}

public String getGenero() {

return genero;

}

public double getSald()

{

return saldo;

}

public double getLevantamento()

{

return levantamento;

}

public double getCpmf()

{

return cpmf;

}

public double getLimite()

{

return limite;

}

public int getSenha()

{

return senha;

}

}

### SistemaBancario.java

import javax.swing.JOptionPane;

import java.lang.Exception;

import java.lang.NumberFormatException;

//

// Copyright (c) Carlos Tojal 2020

// SistemaBancario

// SistemaBancario.java

//

public class SistemaBancario

{

public static void main(String args[])

{

Banco objtlimite = new Banco(0.00, 1.00, 5.00, 0.50);

Banco objtnome = new Banco("a", "b");

Banco objtsobre = new Banco("c", "d");

Banco objtsaldo = new Banco(0.00, 1.00, 5.00, 0.50);

Banco objtlevantamento = new Banco(0.00, 1.00, 5.00, 0.50);

Banco objtcpmf = new Banco(0.00, 1.00, 5.00, 0.50);

Banco obtjsenha = new Banco("456", "8521");

String name = JOptionPane.showInputDialog("Digite o seu nome");

objtnome.setNome(name);

String sobre = JOptionPane.showInputDialog("Digite o seu sobrenome");

objtsobre.setSobre(sobre);

String genero;

do {

genero = JOptionPane.showInputDialog("Digite o seu genero (masculino/feminino)");

objtsobre.setGenero(genero);

if(!genero.equals("masculino") && !genero.equals("feminino"))

JOptionPane.showMessageDialog(null, "Genero indisponível.", "Genero indisponível", JOptionPane.ERROR\_MESSAGE);

} while(!genero.equals("masculino") && !genero.equals("feminino"));

String senha = JOptionPane.showInputDialog("Digite sua senha para acesso");

int se = Integer.parseInt(senha);

obtjsenha.setSenha(se);

String textoAviso = "Dados registados com sucesso! ";

if(genero.equals("masculino"))

textoAviso += "Sr. ";

else

textoAviso += "Sra. ";

textoAviso += name;

String aviso = String.format(textoAviso);

String aviso2 = String.format("ID: %s\nSenha: %s ", name, senha);

JOptionPane.showMessageDialog(null, aviso);

JOptionPane.showMessageDialog(null, aviso2);

//valida dados

String id = JOptionPane.showInputDialog("Digite sua id");

String ss = JOptionPane.showInputDialog("Digite sua senha");

int sh = Integer.parseInt(ss);

if (id != name && se != sh){

JOptionPane.showMessageDialog(null, "Dados Inválidos");

}

else

{

String deposito;

boolean numeroInvalido = false;

double dep = 0.00;

do {

deposito = JOptionPane.showInputDialog("Faca um deposito na conta");

try {

dep = Double.parseDouble(deposito);

} catch (NumberFormatException nfe) {

JOptionPane.showMessageDialog(null, "Valor invalido.", "Erro", JOptionPane.ERROR\_MESSAGE);

numeroInvalido = true;

}

}while(numeroInvalido);

if (dep < 20.00)

{

JOptionPane.showMessageDialog(null, "De acordo com o contrato valor invalido para deposito");

}

else {

JOptionPane.showMessageDialog(null, "Obrigado por realizar um deposito");

dep = dep;

objtsaldo.setSaldo(dep);

double limit;

limit = dep \* (2);

objtlimite.setLimite(limit);

String avisolimite = String.format("Seu limite para emprestimos e: E%.2f", objtlimite.getLimite() );

JOptionPane.showMessageDialog(null, avisolimite);

String levantamento;

numeroInvalido = false;

double lev = 0.00;

do {

levantamento = JOptionPane.showInputDialog("Realize um levantamento ");

try {

lev = Double.parseDouble(levantamento);

} catch (NumberFormatException nfe) {

JOptionPane.showMessageDialog(null, "Valor invalido.", "Erro", JOptionPane.ERROR\_MESSAGE);

numeroInvalido = true;

}

}while(numeroInvalido);

if (lev > dep) {

JOptionPane.showMessageDialog(null, "Saldo Insuficiente");

}

else {

lev = lev;

objtlevantamento.setLevantamento(lev);

double r;

r = dep - lev;

double cpm = (lev \* 2) / 100;

objtcpmf.setCpmf(cpm);

double set = r - cpm;

objtsaldo.setSaldo(set);

limit = set \* (2);

objtlimite.setLimite(limit);

int cont = 785236;

String exibi = String.format("Numero da Conta %s\nSaldo E%.2f\nLevantamento realizado E%.2f\nLimite para emprestimo E%.2f\nValor de CPMF E%.2f\n\nSistema Desenvolvido por Carlos Tojal Version 1.1", cont, objtsaldo.getSald(), objtlevantamento.getLevantamento(), objtlimite.getLimite(), objtcpmf.getCpmf());

JOptionPane.showMessageDialog(null, exibi);

}

}

}

}

}

## Streams

### LeituraCaracteres.java

//

// Copyright (c) Carlos Tojal 2020

// Streams

// LeituraCaracteres.java

//

import java.io.FileReader;

import java.io.FileWriter;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

public class LeituraCaracteres {

public static void main(String[] args) {

try {

FileWriter fw = new FileWriter("Strings.txt");

BufferedWriter bw = new BufferedWriter(fw);

for(int i = 1; i < 10; i++) {

bw.write(i + "a linha");

bw.newLine();

}

bw.close();

/\* Neste caso, o FileReader não pode ser utilizado sozinho porque

nenhum dos seus métodos read devolve Strings \*/

FileReader fr = new FileReader("Strings.txt");

BufferedReader br = new BufferedReader(fr);

while(br.ready())

System.out.print((char)br.read());

br.close();

} catch(IOException e) {

System.out.println("Erro");

}

}

}

## Projeto adicional

## Bank

### AccountManagement.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// AccountManagement.java

//

package management;

import java.util.ArrayList;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.lang.Exception;

import structures.Client;

import structures.Account;

import structures.AccountMovement;

public class AccountManagement {

// Constructor

public AccountManagement() {

}

// Methods

public void loadAccountMovements(Account account) {

ArrayList<AccountMovement> movements = new ArrayList<AccountMovement>();

String raw;

AccountMovement movement = new AccountMovement();

File f;

FileReader fr;

BufferedReader br;

// Loads client accounts

try {

f = new File("account\_movements.csv");

fr = new FileReader(f);

br = new BufferedReader(fr);

while ((raw = br.readLine()) != null) {

if (account.getId().equals(raw.split(";")[0])) {

String account\_id = raw.split(";")[0];

double value = Double.parseDouble(raw.split(";")[1]);

byte type = (byte) Integer.parseInt(raw.split(";")[2]);

account.getAccountMovements().add(new AccountMovement(account\_id, value, type));

account.setAccountMovements(account.getAccountMovements());

if(type == 1)

account.deposit(value);

else

account.withdraw(value);

}

}

fr.close();

br.close();

} catch(FileNotFoundException e) {

System.out.println("[ERROR] File \"accounts.csv\" was not found.");

// e.printStackTrace();

} catch(IOException e) {

System.out.println("[ERROR] Couldn't load file \"accounts.csv\".");

// e.printStackTrace();

} catch(Exception e) {

System.out.println("[ERROR] An error has occurred.");

// e.printStackTrace();

}

}

public void registerAccountMovement(AccountMovement accountMovement) {

File f;

FileWriter fw;

BufferedWriter bw;

try {

fw = new FileWriter("account\_movements.csv", true);

bw = new BufferedWriter(fw);

bw.newLine();

bw.write(accountMovement.getAccount\_id() + ";" + accountMovement.getValue() + ";" + accountMovement.getType());

bw.close();

} catch(FileNotFoundException e) {

System.out.println("[ERROR] File \"account\_movements.csv\" was not found.");

// e.printStackTrace();

} catch(IOException e) {

System.out.println("[ERROR] Couldn't load file \"account\_movements.csv\".");

// e.printStackTrace();

} catch(Exception e) {

System.out.println("[ERROR] An error has occurred.");

// e.printStackTrace();

}

}

public void loadAccounts(ArrayList<Client> clients) {

ArrayList<Account> accounts = new ArrayList<Account>();

String raw;

Account account = new Account();

File f;

FileReader fr;

BufferedReader br;

// Loads client accounts

try {

f = new File("accounts.csv");

fr = new FileReader(f);

br = new BufferedReader(fr);

while ((raw = br.readLine()) != null) {

for (int i = 0; i < clients.size(); i++) {

if (clients.get(i).getId().equals(raw.split(";")[0])) {

String client\_id = raw.split(";")[0];

String id = raw.split(";")[1];

clients.get(i).getAccounts().add(new Account(client\_id, id, 0.0));

// load last added account movements

loadAccountMovements(clients.get(i).getAccounts().get(clients.get(i).getAccounts().size() - 1));

clients.get(i).setAccounts(clients.get(i).getAccounts());

}

}

}

fr.close();

br.close();

} catch(FileNotFoundException e) {

System.out.println("[ERROR] File \"accounts.csv\" was not found.");

// e.printStackTrace();

} catch(IOException e) {

System.out.println("[ERROR] Couldn't load file \"accounts.csv\".");

// e.printStackTrace();

} catch(Exception e) {

System.out.println("[ERROR] An error has occurred.");

// e.printStackTrace();

}

}

public void registerAccount(Account account) {

File f;

FileWriter fw;

BufferedWriter bw;

try {

fw = new FileWriter("accounts.csv", true);

bw = new BufferedWriter(fw);

bw.newLine();

bw.write(account.getClient\_id() + ";" + account.getId());

bw.close();

} catch(FileNotFoundException e) {

System.out.println("[ERROR] File \"accounts.csv\" was not found.");

// e.printStackTrace();

} catch(IOException e) {

System.out.println("[ERROR] Couldn't load file \"accounts.csv\".");

// e.printStackTrace();

} catch(Exception e) {

System.out.println("[ERROR] An error has occurred.");

// e.printStackTrace();

}

}

}

### ClientManagement.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// ClientManagement.java

//

package management;

import java.util.ArrayList;

import java.util.Random;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.PrintWriter;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.lang.Exception;

import structures.Client;

import structures.Account;

import management.AccountManagement;

public class ClientManagement {

// Constructor

public ClientManagement() {

}

// Methods

public String generateId() {

int leftLimit = 48; // number 0

int rightLimit = 57; // letter 9

int targetStringLength = 8;

Random random = new Random();

StringBuilder buffer = new StringBuilder(targetStringLength);

for (int i = 0; i < targetStringLength; i++) {

int randomLimitedInt = leftLimit + (int)

(random.nextFloat() \* (rightLimit - leftLimit + 1));

buffer.append((char) randomLimitedInt);

}

String generatedString = buffer.toString();

return generatedString;

}

public void updateClients(ArrayList<Client> clients) {

AccountManagement accountManagement = new AccountManagement();

try {

File file = new File("clients.csv");

PrintWriter writer = new PrintWriter(file);

writer.print("");

writer.close();

for (int i = 0; i < clients.size(); i++)

registerClient(clients.get(i));

} catch(IOException e) {

System.out.println("[ERROR] Couldn't load file \"clients.csv\".");

// e.printStackTrace();

} catch(Exception e) {

System.out.println("[ERROR] An error has occurred.");

// e.printStackTrace();

}

}

public ArrayList<Client> loadClients() {

AccountManagement accountManagement = new AccountManagement();

ArrayList<Client> clients = new ArrayList<Client>();

ArrayList<Account> accounts = new ArrayList<Account>();

String raw;

Client client = new Client();

File f;

FileReader fr;

BufferedReader br;

// Loads clients

try {

f = new File("clients.csv");

fr = new FileReader(f);

br = new BufferedReader(fr);

while ((raw = br.readLine()) != null) {

client = new Client();

client.setId(raw.split(";")[0]);

client.setName(raw.split(";")[1]);

client.setUsername(raw.split(";")[2]);

client.setPin(Integer.parseInt(raw.split(";")[3]));

client.setAccounts(new ArrayList<Account>());

client.setAccess\_level(Byte.parseByte(raw.split(";")[4]));

clients.add(client);

}

fr.close();

br.close();

// Loads client accounts

accountManagement.loadAccounts(clients);

} catch(FileNotFoundException e) {

System.out.println("[ERROR] File \"clients.csv\" was not found.");

// e.printStackTrace();

} catch(IOException e) {

System.out.println("[ERROR] Couldn't load file \"clients.csv\".");

// e.printStackTrace();

} catch(Exception e) {

System.out.println("[ERROR] An error has occurred.");

// e.printStackTrace();

}

return clients;

}

public void registerClient(Client client) {

File f;

FileWriter fw;

BufferedWriter bw;

try {

fw = new FileWriter("clients.csv", true);

bw = new BufferedWriter(fw);

bw.newLine();

bw.write(client.getId() + ";" + client.getName() + ";" + client.getUsername() + ";" + client.getPin() + ";" + client.getAccess\_level());

bw.close();

} catch(FileNotFoundException e) {

System.out.println("[ERROR] File \"clients.csv\" was not found.");

// e.printStackTrace();

} catch(IOException e) {

System.out.println("[ERROR] Couldn't load file \"clients.csv\".");

// e.printStackTrace();

} catch(Exception e) {

System.out.println("[ERROR] An error has occurred.");

// e.printStackTrace();

}

}

}

### Account.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// Account.java

//

package structures;

import java.util.ArrayList;

import structures.AccountMovement;

public class Account {

// Attributes

private String client\_id;

private String id;

private double balance;

private ArrayList<AccountMovement> accountMovements;

// Constructors

public Account() {

this.client\_id = "ClientID";

this.id = "AccountID";

this.balance = 0.00;

this.accountMovements = new ArrayList<AccountMovement>();

}

public Account(String client\_id, String id, double balance) {

this.client\_id = client\_id;

this.id = id;

this.balance = balance;

this.accountMovements = new ArrayList<AccountMovement>();

}

public Account(String client\_id, String id, double balance, ArrayList<AccountMovement> accountMovements) {

this.client\_id = client\_id;

this.id = id;

this.balance = balance;

this.accountMovements = accountMovements;

}

// Getters and setters

public String getClient\_id() {

return client\_id;

}

public void setClient\_id(String client\_id) {

this.client\_id = client\_id;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public double getBalance() {

return balance;

}

public ArrayList<AccountMovement> getAccountMovements() {

return accountMovements;

}

public void setAccountMovements(ArrayList<AccountMovement> accountMovements) {

this.accountMovements = accountMovements;

}

// Methods

public void deposit(double value) {

if(value > 0)

this.balance += value;

}

public void withdraw(double value) {

if(value > 0) {

if((this.balance - value) >= 0)

this.balance -= value;

}

}

}

### AccountMovement.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// AccountMovement.java

//

package structures;

public class AccountMovement {

// Attributes

private String account\_id;

private double value;

private byte type; // 1 - deposit; 2 - withdraw

// Constructors

public AccountMovement() {

this.account\_id = "AccountID";

this.value = 0.00;

this.type = 1;

}

public AccountMovement(String account\_id, double value, byte type) {

this.account\_id = account\_id;

this.value = value;

this.type = type;

}

// Getters and setters

public String getAccount\_id() {

return account\_id;

}

public void setAccount\_id(String account\_id) {

this.account\_id = account\_id;

}

public double getValue() {

return value;

}

public void setValue(double value) {

this.value = value;

}

public byte getType() {

return type;

}

public void setType(byte type) {

this.type = type;

}

}

### Client.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// Client.java

//

package structures;

import java.util.ArrayList;

public class Client {

// Attributes

private String id;

private String name;

private String username;

private int pin;

private ArrayList<Account> accounts;

private byte access\_level; // 1 - client; 2 - bank employee

// Constructors

public Client() {

this.id = "1234";

this.name = "ClientName";

this.accounts = new ArrayList<Account>();

}

public Client(String id, String name, ArrayList<Account> accounts) {

this.id = id;

this.name = name;

this.accounts = accounts;

}

// Getters and setters

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public int getPin() {

return pin;

}

public void setPin(int pin) {

this.pin = pin;

}

public ArrayList<Account> getAccounts() {

return accounts;

}

public void setAccounts(ArrayList<Account> accounts) {

this.accounts = accounts;

}

public byte getAccess\_level() {

return access\_level;

}

public void setAccess\_level(byte access\_level) {

this.access\_level = access\_level;

}

}

### ClientOptionsWindow.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// ClientOptionsWindow.java

//

package ui;

import java.util.ArrayList;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.GridLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JOptionPane;

import javax.swing.JButton;

import java.lang.Exception;

import structures.Client;

import ui.ListClientAccountsWindow;

import ui.CreateClientAccountWindow;

public class ClientOptionsWindow {

// Attributes

private Client client;

private JFrame frame;

private JPanel panel;

private JButton list\_accounts;

private JButton create\_account;

// Constructor

public ClientOptionsWindow(Client client) {

frame = new JFrame("Client Options - Bank");

panel = new JPanel(new GridLayout(1, 2));

list\_accounts = new JButton("List Accounts");

create\_account = new JButton("Create Account");

this.client = client;

panel.add(list\_accounts);

panel.add(create\_account);

frame.add(panel);

frame.setSize(400, 200);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frame.setVisible(true);

list\_accounts.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new ListClientAccountsWindow(client, (byte) 1);

frame.dispose();

}

});

create\_account.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new CreateClientAccountWindow(client);

frame.setVisible(false);

}

});

}

}

### ControlPanel.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// ControlPanel.java

//

package ui;

import java.util.ArrayList;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.awt.GridLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JLabel;

import javax.swing.JButton;

import javax.swing.JOptionPane;

import structures.Client;

import management.ClientManagement;

public class ControlPanel {

// Attributes

private Client client;

private ArrayList<Client> clients;

private ClientManagement clientManagement;

private JFrame frame;

private JPanel panel;

private JLabel title;

private JButton register\_client;

private JButton list\_clients;

private JButton view\_data;

private JButton deposit;

private JButton withdraw;

// Constructor

public ControlPanel(Client client) {

this.client = client;

clientManagement = new ClientManagement();

clients = clientManagement.loadClients();

frame = new JFrame("Bank");

panel = new JPanel(new GridLayout(5, 2));

title = new JLabel("Welcome, " + client.getName());

register\_client = new JButton("Register client");

list\_clients = new JButton("List clients");

view\_data = new JButton("View data");

deposit = new JButton("Deposit");

withdraw = new JButton("Withdraw");

panel.add(title);

if(client.getAccess\_level() == 2) {

panel.add(register\_client);

panel.add(list\_clients);

}

panel.add(view\_data);

panel.add(deposit);

panel.add(withdraw);

frame.add(panel);

frame.setSize(600, 450);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

register\_client.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new RegisterClientWindow();

}

});

list\_clients.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new ListClientsWindow();

}

});

view\_data.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e) {

String data = "";

data += "Client ID: " + client.getId() + "\n";

data += "Name: " + client.getName() + "\n";

data += "Username: " + client.getUsername() + "\n";

data += "PIN: " + client.getPin() + "\n";

data += "Access level: " + client.getAccess\_level() + "\n";

data += "Number of accounts: " + client.getAccounts().size();

JOptionPane.showMessageDialog(null, data, "Client data", JOptionPane.PLAIN\_MESSAGE);

}

});

deposit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new ListClientAccountsWindow(client, (byte) 2);

}

});

withdraw.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new ListClientAccountsWindow(client, (byte) 3);

}

});

}

}

### CreateClientAccountWindow.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// CreateClientAccountWindow.java

//

package ui;

import java.util.ArrayList;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

import java.awt.GridLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.JOptionPane;

import java.lang.NumberFormatException;

import java.lang.Exception;

import structures.Client;

import structures.Account;

import management.ClientManagement;

import management.AccountManagement;

public class CreateClientAccountWindow {

// Attributes

private Client client;

private Account account;

private ClientManagement clientManagement;

private AccountManagement accountManagement;

private JFrame frame;

private JPanel panel;

private JLabel id\_label;

private JTextField id;

private JLabel name\_label;

private JTextField name;

private JButton confirm;

// Constructor

public CreateClientAccountWindow(Client client) {

frame = new JFrame("Create Client Account - Bank");

panel = new JPanel(new GridLayout(5, 1));

id\_label = new JLabel("Account ID: ");

id = new JTextField();

name\_label = new JLabel("Client Name: ");

name = new JTextField();

confirm = new JButton("Confirm");

this.client = client;

clientManagement = new ClientManagement();

accountManagement = new AccountManagement();

account = new Account(client.getId(), clientManagement.generateId(), 0.00);

id.setText(account.getId());

id.setEditable(false);

name.setText(client.getName());

name.setEditable(false);

panel.add(id\_label);

panel.add(id);

panel.add(name\_label);

panel.add(name);

panel.add(confirm);;

frame.add(panel);

frame.setSize(400, 500);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frame.setVisible(true);

confirm.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

accountManagement.registerAccount(account);

JOptionPane.showMessageDialog(null, "Account created successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

frame.dispose();

}

});

}

}

### DepositWindow.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// DepositWindow.java

//

package ui;

import javax.swing.JOptionPane;

import java.lang.NumberFormatException;

import java.lang.Exception;

import structures.Account;

import structures.AccountMovement;

import management.AccountManagement;

public class DepositWindow {

// Attributes

boolean success = true;

AccountMovement accountMovement;

AccountManagement accountManagement;

// Constructor

public DepositWindow(Account account) {

accountMovement = new AccountMovement();

accountMovement.setType((byte) 1);

accountManagement = new AccountManagement();

accountMovement.setAccount\_id(account.getId());

do {

try {

accountMovement.setValue(Double.parseDouble(JOptionPane.showInputDialog("Value to deposit: ")));

} catch (NumberFormatException e) {

success = false;

System.out.println("[ERROR] Invalid value input.");

JOptionPane.showMessageDialog(null, "Invalid value input.", "Error", JOptionPane.ERROR\_MESSAGE);

}

if(accountMovement.getValue() > 0) {

account.deposit(accountMovement.getValue());

} else {

success = false;

System.out.println("[ERROR] Only positive values are allowed.");

JOptionPane.showMessageDialog(null, "Only positive values are allowed.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}while(!success);

account.getAccountMovements().add(accountMovement);

accountManagement.registerAccountMovement(accountMovement);

account.setAccountMovements(account.getAccountMovements());

}

}

### ListAccountMovementsWindow.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// ListAccountMovementsWindow.java

//

package ui;

import java.util.ArrayList;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.GridLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JLabel;

import javax.swing.JList;

import javax.swing.JScrollPane;

import javax.swing.ListSelectionModel;

import javax.swing.JOptionPane;

import java.lang.Exception;

import structures.Account;

import structures.Client;

import management.AccountManagement;

import management.ClientManagement;

public class ListAccountMovementsWindow {

// Attributes

private Client client;

private ArrayList<Client> clients;

private ArrayList<String> options;

private ClientManagement clientManagement;

private JFrame frame;

private JPanel panel;

private JLabel label;

private JList<String> list;

// Constructor

public ListAccountMovementsWindow(Account account) {

frame = new JFrame("List Account Movements - Bank");

panel = new JPanel(new GridLayout(1, 1));

label = new JLabel("No movements in this account.");

if(account.getAccountMovements().size() > 0) {

options = new ArrayList<String>();

for (int i = 0; i < account.getAccountMovements().size(); i++) {

String output = "";

output += account.getAccountMovements().get(i).getType() == 1 ? "DEPOSIT - " : "WITHDRAW - ";

output += account.getAccountMovements().get(i).getValue();

options.add(output);

}

list = new JList<String>((String[]) options.toArray(new String[0]));

JScrollPane scrollPane = new JScrollPane(list);

panel.add(scrollPane);

} else {

panel.add(label);

}

frame.add(panel);

frame.setSize(400, 500);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frame.setVisible(true);

}

}

### ListClientAccountsWindow.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// ListClientAccountsWindow.java

//

package ui;

import java.util.ArrayList;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.GridLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JList;

import javax.swing.JScrollPane;

import javax.swing.ListSelectionModel;

import javax.swing.JOptionPane;

import java.lang.Exception;

import structures.Client;

import management.ClientManagement;

public class ListClientAccountsWindow {

// Attributes

private Client client;

private ArrayList<Client> clients;

private ArrayList<String> options;

private ClientManagement clientManagement;

private JFrame frame;

private JPanel panel;

private JList<String> list;

// Constructor

public ListClientAccountsWindow(Client client, byte action) {

frame = new JFrame("List Client Accounts - Bank");

panel = new JPanel(new GridLayout(1, 1));

this.client = client;

clientManagement = new ClientManagement();

clients = clientManagement.loadClients();

options = new ArrayList<String>();

for(int i = 0; i < client.getAccounts().size(); i++)

options.add(client.getAccounts().get(i).getId() + " - " + client.getAccounts().get(i).getBalance());

list = new JList<String>((String[])options.toArray(new String[0]));

JScrollPane scrollPane = new JScrollPane(list);

// 1 - list movements; 2 - deposit; 3 - withdraw

list.addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent evt) {

JList list = (JList) evt.getSource();

if(evt.getClickCount() == 2) {

int index = list.locationToIndex(evt.getPoint());

if(action == 1)

new ListAccountMovementsWindow(client.getAccounts().get(index));

else if(action == 2)

new DepositWindow(client.getAccounts().get(index));

else

new WithdrawWindow(client.getAccounts().get(index));

frame.dispose();

}

}

});

panel.add(scrollPane);

frame.add(panel);

frame.setSize(400, 500);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frame.setVisible(true);

}

}

### ListClientsWindow.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// ListClientsWindow.java

//

package ui;

import java.util.ArrayList;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.GridLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JList;

import javax.swing.JScrollPane;

import javax.swing.ListSelectionModel;

import javax.swing.JOptionPane;

import java.lang.Exception;

import structures.Client;

import management.ClientManagement;

public class ListClientsWindow {

// Attributes

private ArrayList<Client> clients;

private ArrayList<String> options;

private ClientManagement clientManagement;

private JFrame frame;

private JPanel panel;

private JList<String> list;

// Constructor

public ListClientsWindow() {

frame = new JFrame("List Clients - Bank");

panel = new JPanel(new GridLayout(1, 1));

clientManagement = new ClientManagement();

clients = clientManagement.loadClients();

options = new ArrayList<String>();

for(int i = 0; i < clients.size(); i++)

options.add(clients.get(i).getId() + " (" + clients.get(i).getUsername() + ")");

list = new JList<String>((String[])options.toArray(new String[0]));

JScrollPane scrollPane = new JScrollPane(list);

list.addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent evt) {

JList list = (JList) evt.getSource();

if(evt.getClickCount() == 2) {

int index = list.locationToIndex(evt.getPoint());

//JOptionPane.showMessageDialog(null, options.get(index), "Message", JOptionPane.ERROR\_MESSAGE);

new ClientOptionsWindow(clients.get(index));

frame.dispose();

}

}

});

panel.add(scrollPane);

frame.add(panel);

frame.setSize(400, 500);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frame.setVisible(true);

}

}

### LoginWindow.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// LoginWindow.java

//

package ui;

import java.util.ArrayList;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

import java.awt.GridLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JPasswordField;

import javax.swing.JButton;

import javax.swing.JOptionPane;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

import java.lang.NumberFormatException;

import java.lang.Exception;

import structures.Client;

import management.ClientManagement;

public class LoginWindow {

// Attributes

private Client client;

private ArrayList<Client> clients;

private ClientManagement clientManagement;

private JFrame frame;

private JPanel panel;

private JLabel username\_label;

private JTextField username;

private JLabel pin\_label;

private JTextField pin;

private JButton login;

// Constructor

public LoginWindow() {

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

System.out.println("[MESSAGE] Set system look and feel.");

} catch(UnsupportedLookAndFeelException e) {

System.out.println("[ERROR] System look and feel not supported.");

} catch(ClassNotFoundException e) {

System.out.println("[ERROR] System look and feel not found.");

} catch(InstantiationException e) {

System.out.println("[ERROR] Error setting system look and feel.");

} catch(IllegalAccessException e) {

System.out.println("[ERROR] Couldn't access system look and feel.");

}

clientManagement = new ClientManagement();

clients = clientManagement.loadClients();

frame = new JFrame("Login - Bank");

panel = new JPanel(new GridLayout(5, 1));

username\_label = new JLabel("Username: ");

username = new JTextField();

pin\_label = new JLabel("PIN: ");

pin = new JPasswordField();

login = new JButton("Login");

panel.add(username\_label);

panel.add(username);

panel.add(pin\_label);

panel.add(pin);

panel.add(login);

frame.add(panel);

frame.setSize(400, 300);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

login.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

boolean inputSuccess = true;

client = new Client();

client.setUsername(username.getText());

try {

client.setPin(Integer.parseInt(pin.getText()));

} catch(NumberFormatException nfe) {

inputSuccess = false;

System.out.println("[ERROR] Invalid PIN input.");

JOptionPane.showMessageDialog(null, "Invalid PIN input.", "Error", JOptionPane.ERROR\_MESSAGE);

}

if(inputSuccess) {

if(login(client, clients)) {

System.out.println("[MESSAGE] Login successful.");

frame.setVisible(false);

new ControlPanel(client);

} else {

System.out.println("[MESSAGE] Invalid username or password.");

JOptionPane.showMessageDialog(null, "Invalid username or password.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

});

}

// Getters

public Client getClient() {

return client;

}

// Methods

public boolean login(Client client, ArrayList<Client> clients) {

boolean success = false;

for(int i = 0; i < clients.size(); i++) {

if(clients.get(i).getUsername().equals(client.getUsername()) && clients.get(i).getPin() == client.getPin()) {

this.client = clients.get(i);

success = true;

return success;

}

}

return success;

}

}

### RegisterClientWindow.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// RegisterClientWindow.java

//

package ui;

import java.util.ArrayList;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

import java.awt.GridLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JPasswordField;

import javax.swing.JButton;

import javax.swing.JOptionPane;

import java.lang.NumberFormatException;

import java.lang.Exception;

import structures.Client;

import management.ClientManagement;

public class RegisterClientWindow {

// Attributes

private Client client;

private ArrayList<Client> clients;

private ClientManagement clientManagement;

private JFrame frame;

private JPanel panel;

private JLabel name\_label;

private JTextField name;

private JLabel username\_label;

private JTextField username;

private JLabel pin\_label;

private JTextField pin;

private JButton register;

// Constructor

public RegisterClientWindow() {

frame = new JFrame("Register Client - Bank");

panel = new JPanel(new GridLayout(7, 1));

name\_label = new JLabel("Name: ");

name = new JTextField();

username\_label = new JLabel("Username: ");

username = new JTextField();

pin\_label = new JLabel("PIN: ");

pin = new JPasswordField();

register = new JButton("Register");

panel.add(name\_label);

panel.add(name);

panel.add(username\_label);

panel.add(username);

panel.add(pin\_label);

panel.add(pin);

panel.add(register);

frame.add(panel);

frame.setSize(400, 500);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frame.setVisible(true);

register.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

boolean inputSuccess = true;

client = new Client();

clientManagement = new ClientManagement();

clients = clientManagement.loadClients();

client.setId(clientManagement.generateId());

client.setName(name.getText());

client.setUsername(username.getText());

client.setAccess\_level((byte) 1);

try {

client.setPin(Integer.parseInt(pin.getText()));

} catch(NumberFormatException nfe) {

inputSuccess = false;

System.out.println("[ERROR] Invalid PIN input.");

JOptionPane.showMessageDialog(null, "Invalid PIN input.", "Error", JOptionPane.ERROR\_MESSAGE);

}

if(inputSuccess) {

clientManagement.registerClient(client);

JOptionPane.showMessageDialog(null, "Client registered successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

frame.dispose();

}

}

});

}

// Getters

public Client getClient() {

return client;

}

}

### WithdrawWindow.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// WithdrawWindow.java

//

package ui;

import javax.swing.JOptionPane;

import java.lang.NumberFormatException;

import java.lang.Exception;

import structures.Account;

import structures.AccountMovement;

import management.AccountManagement;

public class WithdrawWindow {

// Attributes

boolean success = true;

AccountMovement accountMovement;

AccountManagement accountManagement;

// Constructor

public WithdrawWindow(Account account) {

accountMovement = new AccountMovement();

accountMovement.setType((byte )2);

accountManagement = new AccountManagement();

accountMovement.setAccount\_id(account.getId());

do {

try {

accountMovement.setValue(Double.parseDouble(JOptionPane.showInputDialog("Value to withdraw: ")));

} catch (NumberFormatException e) {

success = false;

System.out.println("[ERROR] Invalid value input.");

JOptionPane.showMessageDialog(null, "Invalid value input.", "Error", JOptionPane.ERROR\_MESSAGE);

}

if(accountMovement.getValue() > 0) {

account.withdraw(accountMovement.getValue());

} else {

success = false;

System.out.println("[ERROR] Only positive values are allowed.");

JOptionPane.showMessageDialog(null, "Only positive values are allowed.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}while(!success);

account.getAccountMovements().add(accountMovement);

accountManagement.registerAccountMovement(accountMovement);

account.setAccountMovements(account.getAccountMovements());

}

}

### Main.java

//

// Copyright (c) Carlos Tojal 2020

// Bank

// Main.java

//

import ui.LoginWindow;

public class Main {

public static void main(String[] args) {

new LoginWindow();

}

}

# Conclusão

Concluindo, com este módulo tive a oportunidade de adquirir conhecimentos avançados acerca da programação orientada a objetos.

Considero que obtive os conhecimentos presentes nos objetivos do módulo.

# Anexos

## Características da classe JFRAME

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [JFrame](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "JFrame())()  Constructs a new frame that is initially invisible. |
| [JFrame](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "JFrame(java.awt.GraphicsConfiguration))([GraphicsConfiguration](https://docs.oracle.com/javase/7/docs/api/java/awt/GraphicsConfiguration.html) gc)  Creates a Frame in the specified GraphicsConfiguration of a screen device and a blank title. |
| [JFrame](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "JFrame(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title)  Creates a new, initially invisible Frame with the specified title. |
| [JFrame](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "JFrame(java.lang.String, java.awt.GraphicsConfiguration))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, [GraphicsConfiguration](https://docs.oracle.com/javase/7/docs/api/java/awt/GraphicsConfiguration.html) gc)  Creates a JFrame with the specified title and the specified GraphicsConfiguration of a screen device. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| protected void | [addImpl](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "addImpl(java.awt.Component, java.lang.Object, int))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) comp, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) constraints, int index)  Adds the specified child Component. |
| protected [JRootPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRootPane.html) | [createRootPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "createRootPane())()  Called by the constructor methods to create the default rootPane. |
| protected void | [frameInit](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "frameInit())()  Called by the constructors to init the JFrame properly. |
| [AccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/accessibility/AccessibleContext.html) | [getAccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "getAccessibleContext())()  Gets the AccessibleContext associated with this JFrame. |
| [Container](https://docs.oracle.com/javase/7/docs/api/java/awt/Container.html) | [getContentPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "getContentPane())()  Returns the contentPane object for this frame. |
| int | [getDefaultCloseOperation](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "getDefaultCloseOperation())()  Returns the operation that occurs when the user initiates a "close" on this frame. |
| [Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) | [getGlassPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "getGlassPane())()  Returns the glassPane object for this frame. |
| [Graphics](https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html) | [getGraphics](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "getGraphics())()  Creates a graphics context for this component. |
| [JMenuBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JMenuBar.html) | [getJMenuBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "getJMenuBar())()  Returns the menubar set on this frame. |
| [JLayeredPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLayeredPane.html) | [getLayeredPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "getLayeredPane())()  Returns the layeredPane object for this frame. |
| [JRootPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRootPane.html) | [getRootPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "getRootPane())()  Returns the rootPane object for this frame. |
| [TransferHandler](https://docs.oracle.com/javase/7/docs/api/javax/swing/TransferHandler.html) | [getTransferHandler](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "getTransferHandler())()  Gets the transferHandler property. |
| static boolean | [isDefaultLookAndFeelDecorated](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "isDefaultLookAndFeelDecorated())()  Returns true if newly created JFrames should have their Window decorations provided by the current look and feel. |
| protected boolean | [isRootPaneCheckingEnabled](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "isRootPaneCheckingEnabled())()  Returns whether calls to add and setLayout are forwarded to the contentPane. |
| protected [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [paramString](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "paramString())()  Returns a string representation of this JFrame. |
| protected void | [processWindowEvent](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "processWindowEvent(java.awt.event.WindowEvent))([WindowEvent](https://docs.oracle.com/javase/7/docs/api/java/awt/event/WindowEvent.html) e)  Processes window events occurring on this component. |
| void | [remove](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "remove(java.awt.Component))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) comp)  Removes the specified component from the container. |
| void | [repaint](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "repaint(long, int, int, int, int))(long time, int x, int y, int width, int height)  Repaints the specified rectangle of this component within time milliseconds. |
| void | [setContentPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setContentPane(java.awt.Container))([Container](https://docs.oracle.com/javase/7/docs/api/java/awt/Container.html) contentPane)  Sets the contentPane property. |
| void | [setDefaultCloseOperation](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setDefaultCloseOperation(int))(int operation)  Sets the operation that will happen by default when the user initiates a "close" on this frame. |
| static void | [setDefaultLookAndFeelDecorated](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setDefaultLookAndFeelDecorated(boolean))(boolean defaultLookAndFeelDecorated)  Provides a hint as to whether or not newly created JFrames should have their Window decorations (such as borders, widgets to close the window, title...) provided by the current look and feel. |
| void | [setGlassPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setGlassPane(java.awt.Component))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) glassPane)  Sets the glassPane property. |
| void | [setIconImage](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setIconImage(java.awt.Image))([Image](https://docs.oracle.com/javase/7/docs/api/java/awt/Image.html) image)  Sets the image to be displayed as the icon for this window. |
| void | [setJMenuBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setJMenuBar(javax.swing.JMenuBar))([JMenuBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JMenuBar.html) menubar)  Sets the menubar for this frame. |
| void | [setLayeredPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setLayeredPane(javax.swing.JLayeredPane))([JLayeredPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLayeredPane.html) layeredPane)  Sets the layeredPane property. |
| void | [setLayout](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setLayout(java.awt.LayoutManager))([LayoutManager](https://docs.oracle.com/javase/7/docs/api/java/awt/LayoutManager.html) manager)  Sets the LayoutManager. |
| protected void | [setRootPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setRootPane(javax.swing.JRootPane))([JRootPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRootPane.html) root)  Sets the rootPane property. |
| protected void | [setRootPaneCheckingEnabled](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setRootPaneCheckingEnabled(boolean))(boolean enabled)  Sets whether calls to add and setLayout are forwarded to the contentPane. |
| void | [setTransferHandler](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "setTransferHandler(javax.swing.TransferHandler))([TransferHandler](https://docs.oracle.com/javase/7/docs/api/javax/swing/TransferHandler.html) newHandler)  Sets the transferHandler property, which is a mechanism to support transfer of data into this component. |
| void | [update](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html" \l "update(java.awt.Graphics))([Graphics](https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html) g)  Just calls paint(g). |

## Características da classe JPANEL

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [JPanel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html" \l "JPanel())()  Creates a new JPanel with a double buffer and a flow layout. |
| [JPanel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html" \l "JPanel(boolean))(boolean isDoubleBuffered)  Creates a new JPanel with FlowLayout and the specified buffering strategy. |
| [JPanel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html" \l "JPanel(java.awt.LayoutManager))([LayoutManager](https://docs.oracle.com/javase/7/docs/api/java/awt/LayoutManager.html) layout)  Create a new buffered JPanel with the specified layout manager |
| [JPanel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html" \l "JPanel(java.awt.LayoutManager, boolean))([LayoutManager](https://docs.oracle.com/javase/7/docs/api/java/awt/LayoutManager.html) layout, boolean isDoubleBuffered)  Creates a new JPanel with the specified layout manager and buffering strategy. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| [AccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/accessibility/AccessibleContext.html) | [getAccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html" \l "getAccessibleContext())()  Gets the AccessibleContext associated with this JPanel. |
| [PanelUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/plaf/PanelUI.html) | [getUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html" \l "getUI())()  Returns the look and feel (L&F) object that renders this component. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getUIClassID](https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html" \l "getUIClassID())()  Returns a string that specifies the name of the L&F class that renders this component. |
| protected [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [paramString](https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html" \l "paramString())()  Returns a string representation of this JPanel. |
| void | [setUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html" \l "setUI(javax.swing.plaf.PanelUI))([PanelUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/plaf/PanelUI.html) ui)  Sets the look and feel (L&F) object that renders this component. |
| void | [updateUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html" \l "updateUI())()  Resets the UI property with a value from the current look and feel. |

## Características da classe JBUTTON

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [JButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "JButton())()  Creates a button with no set text or icon. |
| [JButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "JButton(javax.swing.Action))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) a)  Creates a button where properties are taken from the Action supplied. |
| [JButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "JButton(javax.swing.Icon))([Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Creates a button with an icon. |
| [JButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "JButton(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text)  Creates a button with text. |
| [JButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "JButton(java.lang.String, javax.swing.Icon))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Creates a button with initial text and an icon. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| [AccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/accessibility/AccessibleContext.html) | [getAccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "getAccessibleContext())()  Gets the AccessibleContext associated with this JButton. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getUIClassID](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "getUIClassID())()  Returns a string that specifies the name of the L&F class that renders this component. |
| boolean | [isDefaultButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "isDefaultButton())()  Gets the value of the defaultButton property, which if true means that this button is the current default button for its JRootPane. |
| boolean | [isDefaultCapable](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "isDefaultCapable())()  Gets the value of the defaultCapable property. |
| protected [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [paramString](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "paramString())()  Returns a string representation of this JButton. |
| void | [removeNotify](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "removeNotify())()  Overrides JComponent.removeNotify to check if this button is currently set as the default button on the RootPane, and if so, sets the RootPane's default button to null to ensure the RootPane doesn't hold onto an invalid button reference. |
| void | [setDefaultCapable](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "setDefaultCapable(boolean))(boolean defaultCapable)  Sets the defaultCapable property, which determines whether this button can be made the default button for its root pane. |
| void | [updateUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html" \l "updateUI())()  Resets the UI property to a value from the current look and feel. |

## Características da classe JTEXTFIELD

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [JTextField](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "JTextField())()  Constructs a new TextField. |
| [JTextField](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "JTextField(javax.swing.text.Document, java.lang.String, int))([Document](https://docs.oracle.com/javase/7/docs/api/javax/swing/text/Document.html) doc, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, int columns)  Constructs a new JTextField that uses the given text storage model and the given number of columns. |
| [JTextField](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "JTextField(int))(int columns)  Constructs a new empty TextField with the specified number of columns. |
| [JTextField](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "JTextField(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text)  Constructs a new TextField initialized with the specified text. |
| [JTextField](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "JTextField(java.lang.String, int))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, int columns)  Constructs a new TextField initialized with the specified text and columns. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| protected void | [actionPropertyChanged](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "actionPropertyChanged(javax.swing.Action, java.lang.String))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) action, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) propertyName)  Updates the textfield's state in response to property changes in associated action. |
| void | [addActionListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "addActionListener(java.awt.event.ActionListener))([ActionListener](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html) l)  Adds the specified action listener to receive action events from this textfield. |
| protected void | [configurePropertiesFromAction](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "configurePropertiesFromAction(javax.swing.Action))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) a)  Sets the properties on this textfield to match those in the specified Action. |
| protected [PropertyChangeListener](https://docs.oracle.com/javase/7/docs/api/java/beans/PropertyChangeListener.html) | [createActionPropertyChangeListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "createActionPropertyChangeListener(javax.swing.Action))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) a)  Creates and returns a PropertyChangeListener that is responsible for listening for changes from the specified Action and updating the appropriate properties. |
| protected [Document](https://docs.oracle.com/javase/7/docs/api/javax/swing/text/Document.html) | [createDefaultModel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "createDefaultModel())()  Creates the default implementation of the model to be used at construction if one isn't explicitly given. |
| protected void | [fireActionPerformed](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "fireActionPerformed())()  Notifies all listeners that have registered interest for notification on this event type. |
| [AccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/accessibility/AccessibleContext.html) | [getAccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getAccessibleContext())()  Gets the AccessibleContext associated with this JTextField. |
| [Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) | [getAction](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getAction())()  Returns the currently set Action for this ActionEvent source, or null if no Action is set. |
| [ActionListener](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html)[] | [getActionListeners](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getActionListeners())()  Returns an array of all the ActionListeners added to this JTextField with addActionListener(). |
| [Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html)[] | [getActions](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getActions())()  Fetches the command list for the editor. |
| int | [getColumns](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getColumns())()  Returns the number of columns in this TextField. |
| protected int | [getColumnWidth](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getColumnWidth())()  Returns the column width. |
| int | [getHorizontalAlignment](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getHorizontalAlignment())()  Returns the horizontal alignment of the text. |
| [BoundedRangeModel](https://docs.oracle.com/javase/7/docs/api/javax/swing/BoundedRangeModel.html) | [getHorizontalVisibility](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getHorizontalVisibility())()  Gets the visibility of the text field. |
| [Dimension](https://docs.oracle.com/javase/7/docs/api/java/awt/Dimension.html) | [getPreferredSize](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getPreferredSize())()  Returns the preferred size Dimensions needed for this TextField. |
| int | [getScrollOffset](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getScrollOffset())()  Gets the scroll offset, in pixels. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getUIClassID](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "getUIClassID())()  Gets the class ID for a UI. |
| boolean | [isValidateRoot](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "isValidateRoot())()  Calls to revalidate that come from within the textfield itself will be handled by validating the textfield, unless the textfield is contained within a JViewport, in which case this returns false. |
| protected [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [paramString](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "paramString())()  Returns a string representation of this JTextField. |
| void | [postActionEvent](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "postActionEvent())()  Processes action events occurring on this textfield by dispatching them to any registered ActionListener objects. |
| void | [removeActionListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "removeActionListener(java.awt.event.ActionListener))([ActionListener](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html) l)  Removes the specified action listener so that it no longer receives action events from this textfield. |
| void | [scrollRectToVisible](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "scrollRectToVisible(java.awt.Rectangle))([Rectangle](https://docs.oracle.com/javase/7/docs/api/java/awt/Rectangle.html) r)  Scrolls the field left or right. |
| void | [setAction](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "setAction(javax.swing.Action))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) a)  Sets the Action for the ActionEvent source. |
| void | [setActionCommand](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "setActionCommand(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) command)  Sets the command string used for action events. |
| void | [setColumns](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "setColumns(int))(int columns)  Sets the number of columns in this TextField, and then invalidate the layout. |
| void | [setDocument](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "setDocument(javax.swing.text.Document))([Document](https://docs.oracle.com/javase/7/docs/api/javax/swing/text/Document.html) doc)  Associates the editor with a text document. |
| void | [setFont](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "setFont(java.awt.Font))([Font](https://docs.oracle.com/javase/7/docs/api/java/awt/Font.html) f)  Sets the current font. |
| void | [setHorizontalAlignment](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "setHorizontalAlignment(int))(int alignment)  Sets the horizontal alignment of the text. |
| void | [setScrollOffset](https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html" \l "setScrollOffset(int))(int scrollOffset)  Sets the scroll offset, in pixels. |

## Características da classe JLABEL

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [JLabel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "JLabel())()  Creates a JLabel instance with no image and with an empty string for the title. |
| [JLabel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "JLabel(javax.swing.Icon))([Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) image)  Creates a JLabel instance with the specified image. |
| [JLabel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "JLabel(javax.swing.Icon, int))([Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) image, int horizontalAlignment)  Creates a JLabel instance with the specified image and horizontal alignment. |
| [JLabel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "JLabel(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text)  Creates a JLabel instance with the specified text. |
| [JLabel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "JLabel(java.lang.String, javax.swing.Icon, int))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, int horizontalAlignment)  Creates a JLabel instance with the specified text, image, and horizontal alignment. |
| [JLabel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "JLabel(java.lang.String, int))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, int horizontalAlignment)  Creates a JLabel instance with the specified text and horizontal alignment. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| protected int | [checkHorizontalKey](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "checkHorizontalKey(int, java.lang.String))(int key, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) message)  Verify that key is a legal value for the horizontalAlignment properties. |
| protected int | [checkVerticalKey](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "checkVerticalKey(int, java.lang.String))(int key, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) message)  Verify that key is a legal value for the verticalAlignment or verticalTextPosition properties. |
| [AccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/accessibility/AccessibleContext.html) | [getAccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getAccessibleContext())()  Get the AccessibleContext of this object |
| [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) | [getDisabledIcon](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getDisabledIcon())()  Returns the icon used by the label when it's disabled. |
| int | [getDisplayedMnemonic](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getDisplayedMnemonic())()  Return the keycode that indicates a mnemonic key. |
| int | [getDisplayedMnemonicIndex](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getDisplayedMnemonicIndex())()  Returns the character, as an index, that the look and feel should provide decoration for as representing the mnemonic character. |
| int | [getHorizontalAlignment](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getHorizontalAlignment())()  Returns the alignment of the label's contents along the X axis. |
| int | [getHorizontalTextPosition](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getHorizontalTextPosition())()  Returns the horizontal position of the label's text, relative to its image. |
| [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) | [getIcon](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getIcon())()  Returns the graphic image (glyph, icon) that the label displays. |
| int | [getIconTextGap](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getIconTextGap())()  Returns the amount of space between the text and the icon displayed in this label. |
| [Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) | [getLabelFor](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getLabelFor())()  Get the component this is labelling. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getText](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getText())()  Returns the text string that the label displays. |
| [LabelUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/plaf/LabelUI.html) | [getUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getUI())()  Returns the L&F object that renders this component. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getUIClassID](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getUIClassID())()  Returns a string that specifies the name of the l&f class that renders this component. |
| int | [getVerticalAlignment](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getVerticalAlignment())()  Returns the alignment of the label's contents along the Y axis. |
| int | [getVerticalTextPosition](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "getVerticalTextPosition())()  Returns the vertical position of the label's text, relative to its image. |
| boolean | [imageUpdate](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "imageUpdate(java.awt.Image, int, int, int, int, int))([Image](https://docs.oracle.com/javase/7/docs/api/java/awt/Image.html) img, int infoflags, int x, int y, int w, int h)  This is overridden to return false if the current Icon's Image is not equal to the passed in Image img. |
| protected [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [paramString](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "paramString())()  Returns a string representation of this JLabel. |
| void | [setDisabledIcon](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setDisabledIcon(javax.swing.Icon))([Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) disabledIcon)  Set the icon to be displayed if this JLabel is "disabled" (JLabel.setEnabled(false)). |
| void | [setDisplayedMnemonic](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setDisplayedMnemonic(char))(char aChar)  Specifies the displayedMnemonic as a char value. |
| void | [setDisplayedMnemonic](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setDisplayedMnemonic(int))(int key)  Specify a keycode that indicates a mnemonic key. |
| void | [setDisplayedMnemonicIndex](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setDisplayedMnemonicIndex(int))(int index)  Provides a hint to the look and feel as to which character in the text should be decorated to represent the mnemonic. |
| void | [setHorizontalAlignment](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setHorizontalAlignment(int))(int alignment)  Sets the alignment of the label's contents along the X axis. |
| void | [setHorizontalTextPosition](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setHorizontalTextPosition(int))(int textPosition)  Sets the horizontal position of the label's text, relative to its image. |
| void | [setIcon](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setIcon(javax.swing.Icon))([Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Defines the icon this component will display. |
| void | [setIconTextGap](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setIconTextGap(int))(int iconTextGap)  If both the icon and text properties are set, this property defines the space between them. |
| void | [setLabelFor](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setLabelFor(java.awt.Component))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) c)  Set the component this is labelling. |
| void | [setText](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setText(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text)  Defines the single line of text this component will display. |
| void | [setUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setUI(javax.swing.plaf.LabelUI))([LabelUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/plaf/LabelUI.html) ui)  Sets the L&F object that renders this component. |
| void | [setVerticalAlignment](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setVerticalAlignment(int))(int alignment)  Sets the alignment of the label's contents along the Y axis. |
| void | [setVerticalTextPosition](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "setVerticalTextPosition(int))(int textPosition)  Sets the vertical position of the label's text, relative to its image. |
| void | [updateUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html" \l "updateUI())()  Resets the UI property to a value from the current look and feel. |

## Características da classe JCOMBOBOX

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [JComboBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "JComboBox())()  Creates a JComboBox with a default data model. |
| [JComboBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "JComboBox(javax.swing.ComboBoxModel))([ComboBoxModel](https://docs.oracle.com/javase/7/docs/api/javax/swing/ComboBoxModel.html)<[E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html)> aModel)  Creates a JComboBox that takes its items from an existing ComboBoxModel. |
| [JComboBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "JComboBox(E[]))([E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html)[] items)  Creates a JComboBox that contains the elements in the specified array. |
| [JComboBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "JComboBox(java.util.Vector))([Vector](https://docs.oracle.com/javase/7/docs/api/java/util/Vector.html)<[E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html)> items)  Creates a JComboBox that contains the elements in the specified Vector. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| void | [actionPerformed](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "actionPerformed(java.awt.event.ActionEvent))([ActionEvent](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionEvent.html) e)  This method is public as an implementation side effect. |
| protected void | [actionPropertyChanged](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "actionPropertyChanged(javax.swing.Action, java.lang.String))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) action, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) propertyName)  Updates the combobox's state in response to property changes in associated action. |
| void | [addActionListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "addActionListener(java.awt.event.ActionListener))([ActionListener](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html) l)  Adds an ActionListener. |
| void | [addItem](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "addItem(E))([E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html) item)  Adds an item to the item list. |
| void | [addItemListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "addItemListener(java.awt.event.ItemListener))([ItemListener](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ItemListener.html) aListener)  Adds an ItemListener. |
| void | [addPopupMenuListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "addPopupMenuListener(javax.swing.event.PopupMenuListener))([PopupMenuListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/event/PopupMenuListener.html) l)  Adds a PopupMenu listener which will listen to notification messages from the popup portion of the combo box. |
| void | [configureEditor](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "configureEditor(javax.swing.ComboBoxEditor, java.lang.Object))([ComboBoxEditor](https://docs.oracle.com/javase/7/docs/api/javax/swing/ComboBoxEditor.html) anEditor, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) anItem)  Initializes the editor with the specified item. |
| protected void | [configurePropertiesFromAction](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "configurePropertiesFromAction(javax.swing.Action))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) a)  Sets the properties on this combobox to match those in the specified Action. |
| void | [contentsChanged](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "contentsChanged(javax.swing.event.ListDataEvent))([ListDataEvent](https://docs.oracle.com/javase/7/docs/api/javax/swing/event/ListDataEvent.html) e)  This method is public as an implementation side effect. |
| protected [PropertyChangeListener](https://docs.oracle.com/javase/7/docs/api/java/beans/PropertyChangeListener.html) | [createActionPropertyChangeListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "createActionPropertyChangeListener(javax.swing.Action))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) a)  Creates and returns a PropertyChangeListener that is responsible for listening for changes from the specified Action and updating the appropriate properties. |
| protected [JComboBox.KeySelectionManager](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.KeySelectionManager.html) | [createDefaultKeySelectionManager](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "createDefaultKeySelectionManager())()  Returns an instance of the default key-selection manager. |
| protected void | [fireActionEvent](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "fireActionEvent())()  Notifies all listeners that have registered interest for notification on this event type. |
| protected void | [fireItemStateChanged](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "fireItemStateChanged(java.awt.event.ItemEvent))([ItemEvent](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ItemEvent.html) e)  Notifies all listeners that have registered interest for notification on this event type. |
| void | [firePopupMenuCanceled](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "firePopupMenuCanceled())()  Notifies PopupMenuListeners that the popup portion of the combo box has been canceled. |
| void | [firePopupMenuWillBecomeInvisible](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "firePopupMenuWillBecomeInvisible())()  Notifies PopupMenuListeners that the popup portion of the combo box has become invisible. |
| void | [firePopupMenuWillBecomeVisible](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "firePopupMenuWillBecomeVisible())()  Notifies PopupMenuListeners that the popup portion of the combo box will become visible. |
| [AccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/accessibility/AccessibleContext.html) | [getAccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getAccessibleContext())()  Gets the AccessibleContext associated with this JComboBox. |
| [Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) | [getAction](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getAction())()  Returns the currently set Action for this ActionEvent source, or null if no Action is set. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getActionCommand](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getActionCommand())()  Returns the action command that is included in the event sent to action listeners. |
| [ActionListener](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html)[] | [getActionListeners](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getActionListeners())()  Returns an array of all the ActionListeners added to this JComboBox with addActionListener(). |
| [ComboBoxEditor](https://docs.oracle.com/javase/7/docs/api/javax/swing/ComboBoxEditor.html) | [getEditor](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getEditor())()  Returns the editor used to paint and edit the selected item in the JComboBox field. |
| [E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html) | [getItemAt](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getItemAt(int))(int index)  Returns the list item at the specified index. |
| int | [getItemCount](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getItemCount())()  Returns the number of items in the list. |
| [ItemListener](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ItemListener.html)[] | [getItemListeners](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getItemListeners())()  Returns an array of all the ItemListeners added to this JComboBox with addItemListener(). |
| [JComboBox.KeySelectionManager](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.KeySelectionManager.html) | [getKeySelectionManager](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getKeySelectionManager())()  Returns the list's key-selection manager. |
| int | [getMaximumRowCount](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getMaximumRowCount())()  Returns the maximum number of items the combo box can display without a scrollbar |
| [ComboBoxModel](https://docs.oracle.com/javase/7/docs/api/javax/swing/ComboBoxModel.html)<[E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html)> | [getModel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getModel())()  Returns the data model currently used by the JComboBox. |
| [PopupMenuListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/event/PopupMenuListener.html)[] | [getPopupMenuListeners](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getPopupMenuListeners())()  Returns an array of all the PopupMenuListeners added to this JComboBox with addPopupMenuListener(). |
| [E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html) | [getPrototypeDisplayValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getPrototypeDisplayValue())()  Returns the "prototypical display" value - an Object used for the calculation of the display height and width. |
| [ListCellRenderer](https://docs.oracle.com/javase/7/docs/api/javax/swing/ListCellRenderer.html)<? super [E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html)> | [getRenderer](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getRenderer())()  Returns the renderer used to display the selected item in the JComboBox field. |
| int | [getSelectedIndex](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getSelectedIndex())()  Returns the first item in the list that matches the given item. |
| [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) | [getSelectedItem](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getSelectedItem())()  Returns the current selected item. |
| [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] | [getSelectedObjects](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getSelectedObjects())()  Returns an array containing the selected item. |
| [ComboBoxUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/plaf/ComboBoxUI.html) | [getUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getUI())()  Returns the L&F object that renders this component. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getUIClassID](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "getUIClassID())()  Returns the name of the L&F class that renders this component. |
| void | [hidePopup](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "hidePopup())()  Causes the combo box to close its popup window. |
| void | [insertItemAt](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "insertItemAt(E, int))([E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html) item, int index)  Inserts an item into the item list at a given index. |
| protected void | [installAncestorListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "installAncestorListener())() |
| void | [intervalAdded](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "intervalAdded(javax.swing.event.ListDataEvent))([ListDataEvent](https://docs.oracle.com/javase/7/docs/api/javax/swing/event/ListDataEvent.html) e)  This method is public as an implementation side effect. |
| void | [intervalRemoved](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "intervalRemoved(javax.swing.event.ListDataEvent))([ListDataEvent](https://docs.oracle.com/javase/7/docs/api/javax/swing/event/ListDataEvent.html) e)  This method is public as an implementation side effect. |
| boolean | [isEditable](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "isEditable())()  Returns true if the JComboBox is editable. |
| boolean | [isLightWeightPopupEnabled](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "isLightWeightPopupEnabled())()  Gets the value of the lightWeightPopupEnabled property. |
| boolean | [isPopupVisible](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "isPopupVisible())()  Determines the visibility of the popup. |
| protected [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [paramString](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "paramString())()  Returns a string representation of this JComboBox. |
| protected boolean | [processKeyBinding](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "processKeyBinding(javax.swing.KeyStroke, java.awt.event.KeyEvent, int, boolean))([KeyStroke](https://docs.oracle.com/javase/7/docs/api/javax/swing/KeyStroke.html) ks, [KeyEvent](https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html) e, int condition, boolean pressed)  Invoked to process the key bindings for ks as the result of the KeyEvent e. |
| void | [processKeyEvent](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "processKeyEvent(java.awt.event.KeyEvent))([KeyEvent](https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html) e)  Handles KeyEvents, looking for the Tab key. |
| void | [removeActionListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "removeActionListener(java.awt.event.ActionListener))([ActionListener](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html) l)  Removes an ActionListener. |
| void | [removeAllItems](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "removeAllItems())()  Removes all items from the item list. |
| void | [removeItem](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "removeItem(java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) anObject)  Removes an item from the item list. |
| void | [removeItemAt](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "removeItemAt(int))(int anIndex)  Removes the item at anIndex This method works only if the JComboBox uses a mutable data model. |
| void | [removeItemListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "removeItemListener(java.awt.event.ItemListener))([ItemListener](https://docs.oracle.com/javase/7/docs/api/java/awt/event/ItemListener.html) aListener)  Removes an ItemListener. |
| void | [removePopupMenuListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "removePopupMenuListener(javax.swing.event.PopupMenuListener))([PopupMenuListener](https://docs.oracle.com/javase/7/docs/api/javax/swing/event/PopupMenuListener.html) l)  Removes a PopupMenuListener. |
| protected void | [selectedItemChanged](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "selectedItemChanged())()  This protected method is implementation specific. |
| boolean | [selectWithKeyChar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "selectWithKeyChar(char))(char keyChar)  Selects the list item that corresponds to the specified keyboard character and returns true, if there is an item corresponding to that character. |
| void | [setAction](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setAction(javax.swing.Action))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) a)  Sets the Action for the ActionEvent source. |
| void | [setActionCommand](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setActionCommand(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) aCommand)  Sets the action command that should be included in the event sent to action listeners. |
| void | [setEditable](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setEditable(boolean))(boolean aFlag)  Determines whether the JComboBox field is editable. |
| void | [setEditor](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setEditor(javax.swing.ComboBoxEditor))([ComboBoxEditor](https://docs.oracle.com/javase/7/docs/api/javax/swing/ComboBoxEditor.html) anEditor)  Sets the editor used to paint and edit the selected item in the JComboBox field. |
| void | [setEnabled](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setEnabled(boolean))(boolean b)  Enables the combo box so that items can be selected. |
| void | [setKeySelectionManager](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setKeySelectionManager(javax.swing.JComboBox.KeySelectionManager))([JComboBox.KeySelectionManager](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.KeySelectionManager.html) aManager)  Sets the object that translates a keyboard character into a list selection. |
| void | [setLightWeightPopupEnabled](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setLightWeightPopupEnabled(boolean))(boolean aFlag)  Sets the lightWeightPopupEnabled property, which provides a hint as to whether or not a lightweight Component should be used to contain the JComboBox, versus a heavyweight Component such as a Panel or a Window. |
| void | [setMaximumRowCount](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setMaximumRowCount(int))(int count)  Sets the maximum number of rows the JComboBox displays. |
| void | [setModel](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setModel(javax.swing.ComboBoxModel))([ComboBoxModel](https://docs.oracle.com/javase/7/docs/api/javax/swing/ComboBoxModel.html)<[E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html)> aModel)  Sets the data model that the JComboBox uses to obtain the list of items. |
| void | [setPopupVisible](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setPopupVisible(boolean))(boolean v)  Sets the visibility of the popup. |
| void | [setPrototypeDisplayValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setPrototypeDisplayValue(E))([E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html) prototypeDisplayValue)  Sets the prototype display value used to calculate the size of the display for the UI portion. |
| void | [setRenderer](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setRenderer(javax.swing.ListCellRenderer))([ListCellRenderer](https://docs.oracle.com/javase/7/docs/api/javax/swing/ListCellRenderer.html)<? super [E](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html)> aRenderer)  Sets the renderer that paints the list items and the item selected from the list in the JComboBox field. |
| void | [setSelectedIndex](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setSelectedIndex(int))(int anIndex)  Selects the item at index anIndex. |
| void | [setSelectedItem](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setSelectedItem(java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) anObject)  Sets the selected item in the combo box display area to the object in the argument. |
| void | [setUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "setUI(javax.swing.plaf.ComboBoxUI))([ComboBoxUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/plaf/ComboBoxUI.html) ui)  Sets the L&F object that renders this component. |
| void | [showPopup](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "showPopup())()  Causes the combo box to display its popup window. |
| void | [updateUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html" \l "updateUI())()  Resets the UI property to a value from the current look and feel. |

## Características da classe JRADIOBUTTON

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [JRadioButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "JRadioButton())()  Creates an initially unselected radio button with no set text. |
| [JRadioButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "JRadioButton(javax.swing.Action))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) a)  Creates a radiobutton where properties are taken from the Action supplied. |
| [JRadioButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "JRadioButton(javax.swing.Icon))([Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Creates an initially unselected radio button with the specified image but no text. |
| [JRadioButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "JRadioButton(javax.swing.Icon, boolean))([Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, boolean selected)  Creates a radio button with the specified image and selection state, but no text. |
| [JRadioButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "JRadioButton(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text)  Creates an unselected radio button with the specified text. |
| [JRadioButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "JRadioButton(java.lang.String, boolean))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, boolean selected)  Creates a radio button with the specified text and selection state. |
| [JRadioButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "JRadioButton(java.lang.String, javax.swing.Icon))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Creates a radio button that has the specified text and image, and that is initially unselected. |
| [JRadioButton](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "JRadioButton(java.lang.String, javax.swing.Icon, boolean))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, boolean selected)  Creates a radio button that has the specified text, image, and selection state. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| [AccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/accessibility/AccessibleContext.html) | [getAccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "getAccessibleContext())()  Gets the AccessibleContext associated with this JRadioButton. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getUIClassID](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "getUIClassID())()  Returns the name of the L&F class that renders this component. |
| protected [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [paramString](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "paramString())()  Returns a string representation of this JRadioButton. |
| void | [updateUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html" \l "updateUI())()  Resets the UI property to a value from the current look and feel. |

## Características da classe JSCROLLPANE

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [JScrollPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "JScrollPane())()  Creates an empty (no viewport view) JScrollPane where both horizontal and vertical scrollbars appear when needed. |
| [JScrollPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "JScrollPane(java.awt.Component))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) view)  Creates a JScrollPane that displays the contents of the specified component, where both horizontal and vertical scrollbars appear whenever the component's contents are larger than the view. |
| [JScrollPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "JScrollPane(java.awt.Component, int, int))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) view, int vsbPolicy, int hsbPolicy)  Creates a JScrollPane that displays the view component in a viewport whose view position can be controlled with a pair of scrollbars. |
| [JScrollPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "JScrollPane(int, int))(int vsbPolicy, int hsbPolicy)  Creates an empty (no viewport view) JScrollPane with specified scrollbar policies. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| [JScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollBar.html) | [createHorizontalScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "createHorizontalScrollBar())()  Returns a JScrollPane.ScrollBar by default. |
| [JScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollBar.html) | [createVerticalScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "createVerticalScrollBar())()  Returns a JScrollPane.ScrollBar by default. |
| protected [JViewport](https://docs.oracle.com/javase/7/docs/api/javax/swing/JViewport.html) | [createViewport](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "createViewport())()  Returns a new JViewport by default. |
| [AccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/accessibility/AccessibleContext.html) | [getAccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getAccessibleContext())()  Gets the AccessibleContext associated with this JScrollPane. |
| [JViewport](https://docs.oracle.com/javase/7/docs/api/javax/swing/JViewport.html) | [getColumnHeader](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getColumnHeader())()  Returns the column header. |
| [Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) | [getCorner](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getCorner(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) key)  Returns the component at the specified corner. |
| [JScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollBar.html) | [getHorizontalScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getHorizontalScrollBar())()  Returns the horizontal scroll bar that controls the viewport's horizontal view position. |
| int | [getHorizontalScrollBarPolicy](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getHorizontalScrollBarPolicy())()  Returns the horizontal scroll bar policy value. |
| [JViewport](https://docs.oracle.com/javase/7/docs/api/javax/swing/JViewport.html) | [getRowHeader](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getRowHeader())()  Returns the row header. |
| [ScrollPaneUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/plaf/ScrollPaneUI.html) | [getUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getUI())()  Returns the look and feel (L&F) object that renders this component. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getUIClassID](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getUIClassID())()  Returns the suffix used to construct the name of the L&F class used to render this component. |
| [JScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollBar.html) | [getVerticalScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getVerticalScrollBar())()  Returns the vertical scroll bar that controls the viewports vertical view position. |
| int | [getVerticalScrollBarPolicy](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getVerticalScrollBarPolicy())()  Returns the vertical scroll bar policy value. |
| [JViewport](https://docs.oracle.com/javase/7/docs/api/javax/swing/JViewport.html) | [getViewport](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getViewport())()  Returns the current JViewport. |
| [Border](https://docs.oracle.com/javase/7/docs/api/javax/swing/border/Border.html) | [getViewportBorder](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getViewportBorder())()  Returns the Border object that surrounds the viewport. |
| [Rectangle](https://docs.oracle.com/javase/7/docs/api/java/awt/Rectangle.html) | [getViewportBorderBounds](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "getViewportBorderBounds())()  Returns the bounds of the viewport's border. |
| boolean | [isValidateRoot](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "isValidateRoot())()  Overridden to return true so that any calls to revalidate on any descendants of this JScrollPane will cause the entire tree beginning with this JScrollPane to be validated. |
| boolean | [isWheelScrollingEnabled](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "isWheelScrollingEnabled())()  Indicates whether or not scrolling will take place in response to the mouse wheel. |
| protected [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [paramString](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "paramString())()  Returns a string representation of this JScrollPane. |
| void | [setColumnHeader](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setColumnHeader(javax.swing.JViewport))([JViewport](https://docs.oracle.com/javase/7/docs/api/javax/swing/JViewport.html) columnHeader)  Removes the old columnHeader, if it exists; if the new columnHeader isn't null, syncs the x coordinate of its viewPosition with the viewport (if there is one) and then adds it to the scroll pane. |
| void | [setColumnHeaderView](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setColumnHeaderView(java.awt.Component))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) view)  Creates a column-header viewport if necessary, sets its view, and then adds the column-header viewport to the scrollpane. |
| void | [setComponentOrientation](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setComponentOrientation(java.awt.ComponentOrientation))([ComponentOrientation](https://docs.oracle.com/javase/7/docs/api/java/awt/ComponentOrientation.html) co)  Sets the orientation for the vertical and horizontal scrollbars as determined by the ComponentOrientation argument. |
| void | [setCorner](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setCorner(java.lang.String, java.awt.Component))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) key, [Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) corner)  Adds a child that will appear in one of the scroll panes corners, if there's room. |
| void | [setHorizontalScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setHorizontalScrollBar(javax.swing.JScrollBar))([JScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollBar.html) horizontalScrollBar)  Adds the scrollbar that controls the viewport's horizontal view position to the scrollpane. |
| void | [setHorizontalScrollBarPolicy](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setHorizontalScrollBarPolicy(int))(int policy)  Determines when the horizontal scrollbar appears in the scrollpane. |
| void | [setLayout](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setLayout(java.awt.LayoutManager))([LayoutManager](https://docs.oracle.com/javase/7/docs/api/java/awt/LayoutManager.html) layout)  Sets the layout manager for this JScrollPane. |
| void | [setRowHeader](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setRowHeader(javax.swing.JViewport))([JViewport](https://docs.oracle.com/javase/7/docs/api/javax/swing/JViewport.html) rowHeader)  Removes the old rowHeader, if it exists; if the new rowHeader isn't null, syncs the y coordinate of its viewPosition with the viewport (if there is one) and then adds it to the scroll pane. |
| void | [setRowHeaderView](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setRowHeaderView(java.awt.Component))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) view)  Creates a row-header viewport if necessary, sets its view and then adds the row-header viewport to the scrollpane. |
| void | [setUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setUI(javax.swing.plaf.ScrollPaneUI))([ScrollPaneUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/plaf/ScrollPaneUI.html) ui)  Sets the ScrollPaneUI object that provides the look and feel (L&F) for this component. |
| void | [setVerticalScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setVerticalScrollBar(javax.swing.JScrollBar))([JScrollBar](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollBar.html) verticalScrollBar)  Adds the scrollbar that controls the viewports vertical view position to the scrollpane. |
| void | [setVerticalScrollBarPolicy](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setVerticalScrollBarPolicy(int))(int policy)  Determines when the vertical scrollbar appears in the scrollpane. |
| void | [setViewport](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setViewport(javax.swing.JViewport))([JViewport](https://docs.oracle.com/javase/7/docs/api/javax/swing/JViewport.html) viewport)  Removes the old viewport (if there is one); forces the viewPosition of the new viewport to be in the +x,+y quadrant; syncs up the row and column headers (if there are any) with the new viewport; and finally syncs the scrollbars and headers with the new viewport. |
| void | [setViewportBorder](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setViewportBorder(javax.swing.border.Border))([Border](https://docs.oracle.com/javase/7/docs/api/javax/swing/border/Border.html) viewportBorder)  Adds a border around the viewport. |
| void | [setViewportView](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setViewportView(java.awt.Component))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) view)  Creates a viewport if necessary and then sets its view. |
| void | [setWheelScrollingEnabled](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "setWheelScrollingEnabled(boolean))(boolean handleWheel)  Enables/disables scrolling in response to movement of the mouse wheel. |
| void | [updateUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html" \l "updateUI())()  Replaces the current ScrollPaneUI object with a version from the current default look and feel. |

## Características da classe JCHECKBOX

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [JCheckBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "JCheckBox())()  Creates an initially unselected check box button with no text, no icon. |
| [JCheckBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "JCheckBox(javax.swing.Action))([Action](https://docs.oracle.com/javase/7/docs/api/javax/swing/Action.html) a)  Creates a check box where properties are taken from the Action supplied. |
| [JCheckBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "JCheckBox(javax.swing.Icon))([Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Creates an initially unselected check box with an icon. |
| [JCheckBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "JCheckBox(javax.swing.Icon, boolean))([Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, boolean selected)  Creates a check box with an icon and specifies whether or not it is initially selected. |
| [JCheckBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "JCheckBox(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text)  Creates an initially unselected check box with text. |
| [JCheckBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "JCheckBox(java.lang.String, boolean))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, boolean selected)  Creates a check box with text and specifies whether or not it is initially selected. |
| [JCheckBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "JCheckBox(java.lang.String, javax.swing.Icon))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Creates an initially unselected check box with the specified text and icon. |
| [JCheckBox](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "JCheckBox(java.lang.String, javax.swing.Icon, boolean))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) text, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, boolean selected)  Creates a check box with text and icon, and specifies whether or not it is initially selected. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| [AccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/accessibility/AccessibleContext.html) | [getAccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "getAccessibleContext())()  Gets the AccessibleContext associated with this JCheckBox. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getUIClassID](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "getUIClassID())()  Returns a string that specifies the name of the L&F class that renders this component. |
| boolean | [isBorderPaintedFlat](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "isBorderPaintedFlat())()  Gets the value of the borderPaintedFlat property. |
| protected [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [paramString](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "paramString())()  Returns a string representation of this JCheckBox. |
| void | [setBorderPaintedFlat](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "setBorderPaintedFlat(boolean))(boolean b)  Sets the borderPaintedFlat property, which gives a hint to the look and feel as to the appearance of the check box border. |
| void | [updateUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html" \l "updateUI())()  Resets the UI property to a value from the current look and feel. |

## Características da classe JOPTIONPANE

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [JOptionPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "JOptionPane())()  Creates a JOptionPane with a test message. |
| [JOptionPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "JOptionPane(java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message)  Creates a instance of JOptionPane to display a message using the plain-message message type and the default options delivered by the UI. |
| [JOptionPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "JOptionPane(java.lang.Object, int))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, int messageType)  Creates an instance of JOptionPane to display a message with the specified message type and the default options, |
| [JOptionPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "JOptionPane(java.lang.Object, int, int))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, int messageType, int optionType)  Creates an instance of JOptionPane to display a message with the specified message type and options. |
| [JOptionPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "JOptionPane(java.lang.Object, int, int, javax.swing.Icon))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, int messageType, int optionType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Creates an instance of JOptionPane to display a message with the specified message type, options, and icon. |
| [JOptionPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "JOptionPane(java.lang.Object, int, int, javax.swing.Icon, java.lang.Object[]))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, int messageType, int optionType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] options)  Creates an instance of JOptionPane to display a message with the specified message type, icon, and options. |
| [JOptionPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "JOptionPane(java.lang.Object, int, int, javax.swing.Icon, java.lang.Object[], java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, int messageType, int optionType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] options, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) initialValue)  Creates an instance of JOptionPane to display a message with the specified message type, icon, and options, with the initially-selected option specified. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| [JDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JDialog.html) | [createDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "createDialog(java.awt.Component, java.lang.String))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title)  Creates and returns a new JDialog wrapping this centered on the parentComponent in the parentComponent's frame. |
| [JDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JDialog.html) | [createDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "createDialog(java.lang.String))([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title)  Creates and returns a new parentless JDialog with the specified title. |
| [JInternalFrame](https://docs.oracle.com/javase/7/docs/api/javax/swing/JInternalFrame.html) | [createInternalFrame](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "createInternalFrame(java.awt.Component, java.lang.String))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title)  Creates and returns an instance of JInternalFrame. |
| [AccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/accessibility/AccessibleContext.html) | [getAccessibleContext](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getAccessibleContext())()  Returns the AccessibleContext associated with this JOptionPane. |
| static [JDesktopPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JDesktopPane.html) | [getDesktopPaneForComponent](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getDesktopPaneForComponent(java.awt.Component))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent)  Returns the specified component's desktop pane. |
| static [Frame](https://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html) | [getFrameForComponent](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getFrameForComponent(java.awt.Component))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent)  Returns the specified component's Frame. |
| [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) | [getIcon](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getIcon())()  Returns the icon this pane displays. |
| [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) | [getInitialSelectionValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getInitialSelectionValue())()  Returns the input value that is displayed as initially selected to the user. |
| [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) | [getInitialValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getInitialValue())()  Returns the initial value. |
| [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) | [getInputValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getInputValue())()  Returns the value the user has input, if wantsInput is true. |
| int | [getMaxCharactersPerLineCount](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getMaxCharactersPerLineCount())()  Returns the maximum number of characters to place on a line in a message. |
| [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) | [getMessage](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getMessage())()  Returns the message-object this pane displays. |
| int | [getMessageType](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getMessageType())()  Returns the message type. |
| [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] | [getOptions](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getOptions())()  Returns the choices the user can make. |
| int | [getOptionType](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getOptionType())()  Returns the type of options that are displayed. |
| static [Frame](https://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html) | [getRootFrame](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getRootFrame())()  Returns the Frame to use for the class methods in which a frame is not provided. |
| [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] | [getSelectionValues](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getSelectionValues())()  Returns the input selection values. |
| [OptionPaneUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/plaf/OptionPaneUI.html) | [getUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getUI())()  Returns the UI object which implements the L&F for this component. |
| [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [getUIClassID](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getUIClassID())()  Returns the name of the UI class that implements the L&F for this component. |
| [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) | [getValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getValue())()  Returns the value the user has selected. |
| boolean | [getWantsInput](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "getWantsInput())()  Returns the value of the wantsInput property. |
| protected [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [paramString](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "paramString())()  Returns a string representation of this JOptionPane. |
| void | [selectInitialValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "selectInitialValue())()  Requests that the initial value be selected, which will set focus to the initial value. |
| void | [setIcon](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setIcon(javax.swing.Icon))([Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) newIcon)  Sets the icon to display. |
| void | [setInitialSelectionValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setInitialSelectionValue(java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) newValue)  Sets the input value that is initially displayed as selected to the user. |
| void | [setInitialValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setInitialValue(java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) newInitialValue)  Sets the initial value that is to be enabled -- the Component that has the focus when the pane is initially displayed. |
| void | [setInputValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setInputValue(java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) newValue)  Sets the input value that was selected or input by the user. |
| void | [setMessage](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setMessage(java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) newMessage)  Sets the option pane's message-object. |
| void | [setMessageType](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setMessageType(int))(int newType)  Sets the option pane's message type. |
| void | [setOptions](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setOptions(java.lang.Object[]))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] newOptions)  Sets the options this pane displays. |
| void | [setOptionType](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setOptionType(int))(int newType)  Sets the options to display. |
| static void | [setRootFrame](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setRootFrame(java.awt.Frame))([Frame](https://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html) newRootFrame)  Sets the frame to use for class methods in which a frame is not provided. |
| void | [setSelectionValues](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setSelectionValues(java.lang.Object[]))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] newValues)  Sets the input selection values for a pane that provides the user with a list of items to choose from. |
| void | [setUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setUI(javax.swing.plaf.OptionPaneUI))([OptionPaneUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/plaf/OptionPaneUI.html) ui)  Sets the UI object which implements the L&F for this component. |
| void | [setValue](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setValue(java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) newValue)  Sets the value the user has chosen. |
| void | [setWantsInput](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "setWantsInput(boolean))(boolean newValue)  Sets the wantsInput property. |
| static int | [showConfirmDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showConfirmDialog(java.awt.Component, java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message)  Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**. |
| static int | [showConfirmDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showConfirmDialog(java.awt.Component, java.lang.Object, java.lang.String, int))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int optionType)  Brings up a dialog where the number of choices is determined by the optionType parameter. |
| static int | [showConfirmDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showConfirmDialog(java.awt.Component, java.lang.Object, java.lang.String, int, int))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int optionType, int messageType)  Brings up a dialog where the number of choices is determined by the optionType parameter, where the messageType parameter determines the icon to display. |
| static int | [showConfirmDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showConfirmDialog(java.awt.Component, java.lang.Object, java.lang.String, int, int, javax.swing.Icon))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int optionType, int messageType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Brings up a dialog with a specified icon, where the number of choices is determined by the optionType parameter. |
| static [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [showInputDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInputDialog(java.awt.Component, java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message)  Shows a question-message dialog requesting input from the user parented to parentComponent. |
| static [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [showInputDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInputDialog(java.awt.Component, java.lang.Object, java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) initialSelectionValue)  Shows a question-message dialog requesting input from the user and parented to parentComponent. |
| static [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [showInputDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInputDialog(java.awt.Component, java.lang.Object, java.lang.String, int))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int messageType)  Shows a dialog requesting input from the user parented to parentComponent with the dialog having the title title and message type messageType. |
| static [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) | [showInputDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInputDialog(java.awt.Component, java.lang.Object, java.lang.String, int, javax.swing.Icon, java.lang.Object[], java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int messageType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] selectionValues, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) initialSelectionValue)  Prompts the user for input in a blocking dialog where the initial selection, possible selections, and all other options can be specified. |
| static [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [showInputDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInputDialog(java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message)  Shows a question-message dialog requesting input from the user. |
| static [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [showInputDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInputDialog(java.lang.Object, java.lang.Object))([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) initialSelectionValue)  Shows a question-message dialog requesting input from the user, with the input value initialized to initialSelectionValue. |
| static int | [showInternalConfirmDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalConfirmDialog(java.awt.Component, java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message)  Brings up an internal dialog panel with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**. |
| static int | [showInternalConfirmDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalConfirmDialog(java.awt.Component, java.lang.Object, java.lang.String, int))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int optionType)  Brings up a internal dialog panel where the number of choices is determined by the optionType parameter. |
| static int | [showInternalConfirmDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalConfirmDialog(java.awt.Component, java.lang.Object, java.lang.String, int, int))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int optionType, int messageType)  Brings up an internal dialog panel where the number of choices is determined by the optionType parameter, where the messageType parameter determines the icon to display. |
| static int | [showInternalConfirmDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalConfirmDialog(java.awt.Component, java.lang.Object, java.lang.String, int, int, javax.swing.Icon))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int optionType, int messageType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Brings up an internal dialog panel with a specified icon, where the number of choices is determined by the optionType parameter. |
| static [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [showInternalInputDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalInputDialog(java.awt.Component, java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message)  Shows an internal question-message dialog requesting input from the user parented to parentComponent. |
| static [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [showInternalInputDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalInputDialog(java.awt.Component, java.lang.Object, java.lang.String, int))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int messageType)  Shows an internal dialog requesting input from the user parented to parentComponent with the dialog having the title title and message type messageType. |
| static [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) | [showInternalInputDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalInputDialog(java.awt.Component, java.lang.Object, java.lang.String, int, javax.swing.Icon, java.lang.Object[], java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int messageType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] selectionValues, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) initialSelectionValue)  Prompts the user for input in a blocking internal dialog where the initial selection, possible selections, and all other options can be specified. |
| static void | [showInternalMessageDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalMessageDialog(java.awt.Component, java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message)  Brings up an internal confirmation dialog panel. |
| static void | [showInternalMessageDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalMessageDialog(java.awt.Component, java.lang.Object, java.lang.String, int))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int messageType)  Brings up an internal dialog panel that displays a message using a default icon determined by the messageType parameter. |
| static void | [showInternalMessageDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalMessageDialog(java.awt.Component, java.lang.Object, java.lang.String, int, javax.swing.Icon))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int messageType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Brings up an internal dialog panel displaying a message, specifying all parameters. |
| static int | [showInternalOptionDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showInternalOptionDialog(java.awt.Component, java.lang.Object, java.lang.String, int, int, javax.swing.Icon, java.lang.Object[], java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int optionType, int messageType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] options, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) initialValue)  Brings up an internal dialog panel with a specified icon, where the initial choice is determined by the initialValue parameter and the number of choices is determined by the optionType parameter. |
| static void | [showMessageDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showMessageDialog(java.awt.Component, java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message)  Brings up an information-message dialog titled "Message". |
| static void | [showMessageDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showMessageDialog(java.awt.Component, java.lang.Object, java.lang.String, int))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int messageType)  Brings up a dialog that displays a message using a default icon determined by the messageType parameter. |
| static void | [showMessageDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showMessageDialog(java.awt.Component, java.lang.Object, java.lang.String, int, javax.swing.Icon))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int messageType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon)  Brings up a dialog displaying a message, specifying all parameters. |
| static int | [showOptionDialog](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "showOptionDialog(java.awt.Component, java.lang.Object, java.lang.String, int, int, javax.swing.Icon, java.lang.Object[], java.lang.Object))([Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) parentComponent, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) message, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) title, int optionType, int messageType, [Icon](https://docs.oracle.com/javase/7/docs/api/javax/swing/Icon.html) icon, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)[] options, [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) initialValue)  Brings up a dialog with a specified icon, where the initial choice is determined by the initialValue parameter and the number of choices is determined by the optionType parameter. |
| void | [updateUI](https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html" \l "updateUI())()  Notification from the UIManager that the L&F has changed. |

## Características da classe BUFFEREDINPUTSTREAM

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [BufferedInputStream](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html" \l "BufferedInputStream(java.io.InputStream))([InputStream](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html) in)  Creates a BufferedInputStream and saves its argument, the input stream in, for later use. |
| [BufferedInputStream](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html" \l "BufferedInputStream(java.io.InputStream, int))([InputStream](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html) in, int size)  Creates a BufferedInputStream with the specified buffer size, and saves its argument, the input stream in, for later use. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| int | [available](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html" \l "available())()  Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream. |
| void | [close](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html" \l "close())()  Closes this input stream and releases any system resources associated with the stream. |
| void | [mark](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html" \l "mark(int))(int readlimit)  See the general contract of the mark method of InputStream. |
| boolean | [markSupported](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html" \l "markSupported())()  Tests if this input stream supports the mark and reset methods. |
| int | [read](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html" \l "read())()  See the general contract of the read method of InputStream. |
| int | [read](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html" \l "read(byte[], int, int))(byte[] b, int off, int len)  Reads bytes from this byte-input stream into the specified byte array, starting at the given offset. |
| void | [reset](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html" \l "reset())()  See the general contract of the reset method of InputStream. |
| long | [skip](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html" \l "skip(long))(long n)  See the general contract of the skip method of InputStream. |

## Características da classe BufferedOutputStream

### Constructor Summary

Constructors

|  |
| --- |
| Constructor and Description |
| [BufferedOutputStream](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedOutputStream.html" \l "BufferedOutputStream(java.io.OutputStream))([OutputStream](https://docs.oracle.com/javase/7/docs/api/java/io/OutputStream.html) out)  Creates a new buffered output stream to write data to the specified underlying output stream. |
| [BufferedOutputStream](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedOutputStream.html" \l "BufferedOutputStream(java.io.OutputStream, int))([OutputStream](https://docs.oracle.com/javase/7/docs/api/java/io/OutputStream.html) out, int size)  Creates a new buffered output stream to write data to the specified underlying output stream with the specified buffer size. |

### Method Summary

Methods

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| void | [flush](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedOutputStream.html" \l "flush())()  Flushes this buffered output stream. |
| void | [write](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedOutputStream.html" \l "write(byte[], int, int))(byte[] b, int off, int len)  Writes len bytes from the specified byte array starting at offset off to this buffered output stream. |
| void | [write](https://docs.oracle.com/javase/7/docs/api/java/io/BufferedOutputStream.html" \l "write(int))(int b)  Writes the specified byte to this buffered output stream. |

# Referências Bibliográficas

* https://www.devmedia.com.br/tratando-excecoes-em-java/25514
* https://docs.oracle.com/javase/tutorial/essential/io/streams.html
* https://www.devmedia.com.br/explorando-a-classe-arraylist-no-java/24298
* https://pt.wikipedia.org/wiki/Interface\_gr%C3%A1fica\_do\_utilizador
* <https://pt.wikipedia.org/wiki/Swing_(Java>)