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

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Índice

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

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

POCH
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

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".

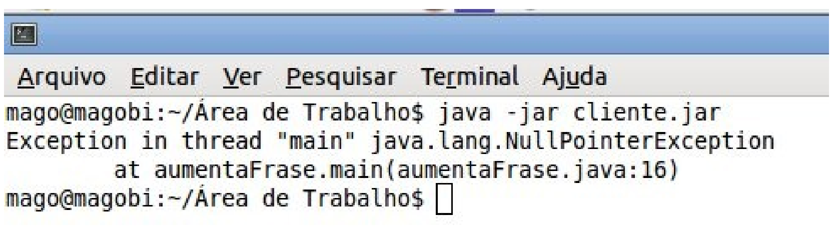
Neste módulo foram explorados os seguintes temas:

- 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).



```
mago@magobi:~/Área de Trabalho$ java -jar cliente.jar
Exception in thread "main" java.lang.NullPointerException
        at aumentaFrase.main(aumentaFrase.java:16)
mago@magobi:~/Área de Trabalho$
```

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 listas dinâmicas.

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.

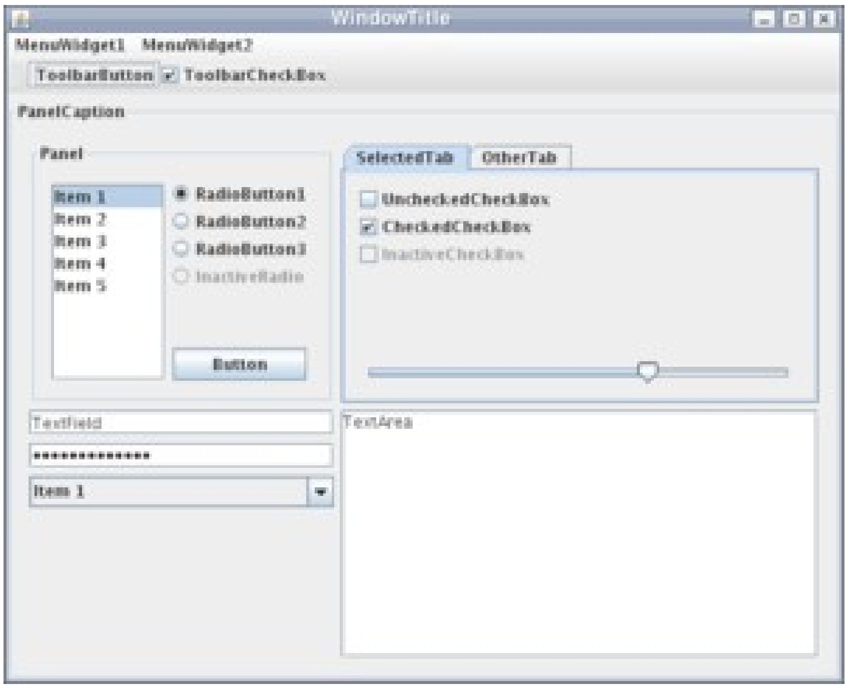


Imagem 2: Interface Gráfica criada com JAVA

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;
```

```java
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

```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);
}
}
```

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

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);
}
});
}
}
```

PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

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);
```

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

```
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;
```

PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

```
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)
```

PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

```
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.");
}
}
}
```

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

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.");
}
}
}
```

POCH
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

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;
```

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

```
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\".");
```

POCH
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

```
// 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
```

POCH
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

```
loadAccountMovements(clients.get(i).getAccounts().get(clients.get(i).getAccoun
ts().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.");
```

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

```
// 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;
```

PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

```
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() {
```

PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

```
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);
}
```

PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

```
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) {
```

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

```
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);
```

PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

```
}
}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);
```

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

```
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.");
```

PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

```
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();
}
}
```

POCH
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

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()` Constructs a new frame that is initially invisible.
`JFrame(GraphicsConfiguration gc)` Creates a `Frame` in the specified `GraphicsConfiguration` of a screen device and a blank title.
`JFrame(String title)` Creates a new, initially invisible `Frame` with the specified title.
`JFrame(String title, GraphicsConfiguration 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(Component comp, Object constraints, int index)` Adds the specified child `Component`.
`protected JRootPane`	`createRootPane()` Called by the constructor methods to create the default `rootPane`.
`protected void`	`frameInit()` Called by the constructors to init the `JFrame` properly.
`AccessibleContext`	`getAccessibleContext()` Gets the AccessibleContext associated with this JFrame.
`Container`	`getContentPane()` Returns the `contentPane` object for this frame.
`int`	`getDefaultCloseOperation()`

	Returns the operation that occurs when the user initiates a "close" on this frame.
Component	getGlassPane() Returns the glassPane object for this frame.
Graphics	getGraphics() Creates a graphics context for this component.
JMenuBar	getJMenuBar() Returns the menubar set on this frame.
JLayeredPane	getLayeredPane() Returns the layeredPane object for this frame.
JRootPane	getRootPane() Returns the rootPane object for this frame.
TransferHandler	getTransferHandler() Gets the transferHandler property.
static boolean	isDefaultLookAndFeelDecorated() Returns true if newly created JFrames should have their Window decorations provided by the current look and feel.
protected boolean	isRootPaneCheckingEnabled() Returns whether calls to add and setLayout are forwarded to the contentPane.
protected String	paramString() Returns a string representation of this JFrame.
protected void	processWindowEvent(WindowEvent e) Processes window events occurring on this component.
void	remove(Component comp) Removes the specified component from the container.
void	repaint(long time, int x, int y, int width, int height) Repaints the specified rectangle of this component within time milliseconds.
void	setContentPane(Container contentPane) Sets the contentPane property.
void	setDefaultCloseOperation(int operation) Sets the operation that will happen by default when the user initiates a "close" on this frame.

static void	**setDefaultLookAndFeelDecorated**(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**(Component glassPane) Sets the glassPane property.
void	**setIconImage**(Image image) Sets the image to be displayed as the icon for this window.
void	**setJMenuBar**(JMenuBar menubar) Sets the menubar for this frame.
void	**setLayeredPane**(JLayeredPane layeredPane) Sets the layeredPane property.
void	**setLayout**(LayoutManager manager) Sets the LayoutManager.
protected void	**setRootPane**(JRootPane root) Sets the rootPane property.
protected void	**setRootPaneCheckingEnabled**(boolean enabled) Sets whether calls to add and setLayout are forwarded to the contentPane.
void	**setTransferHandler**(TransferHandler newHandler) Sets the transferHandler property, which is a mechanism to support transfer of data into this component.
void	**update**(Graphics g) Just calls paint(g).

Características da classe JPANEL

- Constructor Summary

Constructors

Constructor and Description
JPanel()   Creates a new `JPanel` with a double buffer and a flow layout.
JPanel(boolean isDoubleBuffered)   Creates a new `JPanel` with `FlowLayout` and the specified buffering strategy.
JPanel(LayoutManager layout)   Create a new buffered JPanel with the specified layout manager
JPanel(LayoutManager 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	**getAccessibleContext**() Gets the AccessibleContext associated with this JPanel.
PanelUI	**getUI**() Returns the look and feel (L&F) object that renders this component.
String	**getUIClassID**() Returns a string that specifies the name of the L&F class that renders this component.
protected String	**paramString**() Returns a string representation of this JPanel.
void	**setUI**(PanelUI ui) Sets the look and feel (L&F) object that renders this component.
void	**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()`   Creates a button with no set text or icon.
`JButton(Action a)`   Creates a button where properties are taken from the `Action` supplied.
`JButton(Icon icon)`   Creates a button with an icon.
`JButton(String text)`   Creates a button with text.
`JButton(String text, Icon icon)`   Creates a button with initial text and an icon.

- Method Summary

Methods

Modifier and Type	Method and Description
`AccessibleContext`	`getAccessibleContext()` Gets the `AccessibleContext` associated with this `JButton`.
`String`	`getUIClassID()` Returns a string that specifies the name of the L&F class that renders this component.
`boolean`	`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()` Gets the value of the `defaultCapable` property.
`protected String`	`paramString()` Returns a string representation of this `JButton`.
`void`	`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**(boolean defaultCapable) Sets the defaultCapable property, which determines whether this button can be made the default button for its root pane.
void	**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() Constructs a new TextField.
JTextField(Document doc, String text, int columns) Constructs a new JTextField that uses the given text storage model and the given number of columns.
JTextField(int columns) Constructs a new empty TextField with the specified number of columns.
JTextField(String text) Constructs a new TextField initialized with the specified text.
JTextField(String 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**(Action action, String propertyName) Updates the textfield's state in response to property changes in associated action.

`void`	**`addActionListener`**`(ActionListener l)` Adds the specified action listener to receive action events from this textfield.
`protected void`	**`configurePropertiesFromAction`**`(Action a)` Sets the properties on this textfield to match those in the specified `Action`.
`protected PropertyChangeListener`	**`createActionPropertyChangeListener`**`(Action a)` Creates and returns a `PropertyChangeListener` that is responsible for listening for changes from the specified `Action` and updating the appropriate properties.
`protected Document`	**`createDefaultModel`**`()` Creates the default implementation of the model to be used at construction if one isn't explicitly given.
`protected void`	**`fireActionPerformed`**`()` Notifies all listeners that have registered interest for notification on this event type.
`AccessibleContext`	**`getAccessibleContext`**`()` Gets the `AccessibleContext` associated with this `JTextField`.
`Action`	**`getAction`**`()` Returns the currently set `Action` for this `ActionEvent` source, or `null` if no `Action` is set.
`ActionListener[]`	**`getActionListeners`**`()` Returns an array of all the `ActionListener`s added to this JTextField with addActionListener().
`Action[]`	**`getActions`**`()` Fetches the command list for the editor.
`int`	**`getColumns`**`()` Returns the number of columns in this `TextField`.

`protected int`	**`getColumnWidth`**`()` Returns the column width.
`int`	**`getHorizontalAlignment`**`()` Returns the horizontal alignment of the text.
`BoundedRangeModel`	**`getHorizontalVisibility`**`()` Gets the visibility of the text field.
`Dimension`	**`getPreferredSize`**`()` Returns the preferred size `Dimensions` needed for this `TextField`.
`int`	**`getScrollOffset`**`()` Gets the scroll offset, in pixels.
`String`	**`getUIClassID`**`()` Gets the class ID for a UI.
`boolean`	**`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`	**`paramString`**`()` Returns a string representation of this `JTextField`.
`void`	**`postActionEvent`**`()` Processes action events occurring on this textfield by dispatching them to any registered `ActionListener` objects.
`void`	**`removeActionListener`**`(ActionListener l)` Removes the specified action listener so that it no longer receives action events from this textfield.
`void`	**`scrollRectToVisible`**`(Rectangle r)` Scrolls the field left or right.
`void`	**`setAction`**`(Action a)` Sets the `Action` for the `ActionEvent` source.
`void`	**`setActionCommand`**`(String command)` Sets the command string used for action events.

void	**setColumns**(int columns) Sets the number of columns in this TextField, and then invalidate the layout.
void	**setDocument**(Document doc) Associates the editor with a text document.
void	**setFont**(Font f) Sets the current font.
void	**setHorizontalAlignment**(int alignment) Sets the horizontal alignment of the text.
void	**setScrollOffset**(int scrollOffset) Sets the scroll offset, in pixels.

Características da classe JLABEL

- Constructor Summary

Constructors

Constructor and Description
JLabel() Creates a JLabel instance with no image and with an empty string for the title.
JLabel(Icon image) Creates a JLabel instance with the specified image.
JLabel(Icon image, int horizontalAlignment) Creates a JLabel instance with the specified image and horizontal alignment.
JLabel(String text) Creates a JLabel instance with the specified text.
JLabel(String text, Icon icon, int horizontalAlignment) Creates a JLabel instance with the specified text, image, and horizontal alignment.
JLabel(String 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**(int key, String message)

	Verify that key is a legal value for the horizontalAlignment properties.
protected int	**checkVerticalKey**(int key, String message) Verify that key is a legal value for the verticalAlignment or verticalTextPosition properties.
AccessibleContext	**getAccessibleContext**() Get the AccessibleContext of this object
Icon	**getDisabledIcon**() Returns the icon used by the label when it's disabled.
int	**getDisplayedMnemonic**() Return the keycode that indicates a mnemonic key.
int	**getDisplayedMnemonicIndex**() Returns the character, as an index, that the look and feel should provide decoration for as representing the mnemonic character.
int	**getHorizontalAlignment**() Returns the alignment of the label's contents along the X axis.
int	**getHorizontalTextPosition**() Returns the horizontal position of the label's text, relative to its image.
Icon	**getIcon**() Returns the graphic image (glyph, icon) that the label displays.
int	**getIconTextGap**() Returns the amount of space between the text and the icon displayed in this label.
Component	**getLabelFor**() Get the component this is labelling.
String	**getText**() Returns the text string that the label displays.
LabelUI	**getUI**() Returns the L&F object that renders this component.
String	**getUIClassID**() Returns a string that specifies the name of the l&f class that renders this component.
int	**getVerticalAlignment**()

	Returns the alignment of the label's contents along the Y axis.
`int`	**`getVerticalTextPosition`**`()` Returns the vertical position of the label's text, relative to its image.
`boolean`	**`imageUpdate`**`(Image 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`	**`paramString`**`()` Returns a string representation of this JLabel.
`void`	**`setDisabledIcon`**`(Icon disabledIcon)` Set the icon to be displayed if this JLabel is "disabled" (JLabel.setEnabled(false)).
`void`	**`setDisplayedMnemonic`**`(char aChar)` Specifies the displayedMnemonic as a char value.
`void`	**`setDisplayedMnemonic`**`(int key)` Specify a keycode that indicates a mnemonic key.
`void`	**`setDisplayedMnemonicIndex`**`(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`**`(int alignment)` Sets the alignment of the label's contents along the X axis.
`void`	**`setHorizontalTextPosition`**`(int textPosition)` Sets the horizontal position of the label's text, relative to its image.
`void`	**`setIcon`**`(Icon icon)` Defines the icon this component will display.
`void`	**`setIconTextGap`**`(int iconTextGap)` If both the icon and text properties are set, this property defines the space between them.
`void`	**`setLabelFor`**`(Component c)` Set the component this is labelling.
`void`	**`setText`**`(String text)` Defines the single line of text this component will

	display.
`void`	`setUI(LabelUI ui)` Sets the L&F object that renders this component.
`void`	`setVerticalAlignment(int alignment)` Sets the alignment of the label's contents along the Y axis.
`void`	`setVerticalTextPosition(int textPosition)` Sets the vertical position of the label's text, relative to its image.
`void`	`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()` Creates a `JComboBox` with a default data model.
`JComboBox(ComboBoxModel<E> aModel)` Creates a `JComboBox` that takes its items from an existing `ComboBoxModel`.
`JComboBox(E[] items)` Creates a `JComboBox` that contains the elements in the specified array.
`JComboBox(Vector<E> items)` Creates a `JComboBox` that contains the elements in the specified Vector.

- Method Summary

Methods

Modifier and Type	Method and Description
`void`	`actionPerformed(ActionEvent e)` This method is public as an implementation side effect.
`protected void`	`actionPropertyChanged(Action action, String propertyName)` Updates the combobox's state in response to property changes in associated action.
`void`	`addActionListener(ActionListener l)` Adds an `ActionListener`.
`void`	`addItem(E item)` Adds an item to the item list.
`void`	`addItemListener(ItemListener aListener)` Adds an `ItemListener`.
`void`	`addPopupMenuListener(PopupMenuListener l)` Adds a `PopupMenu` listener which will listen to notification messages from the popup portion of the combo box.
`void`	`configureEditor(ComboBoxEditor anEditor, Object anItem)` Initializes the editor with the specified item.
`protected void`	`configurePropertiesFromAction(Action a)`

	Sets the properties on this combobox to match those in the specified `Action`.
`void`	**`contentsChanged`**`(ListDataEvent e)` This method is public as an implementation side effect.
`protected PropertyChangeListener`	**`createActionPropertyChangeListener`**`(Action 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`	**`createDefaultKeySelectionManager`**`()` Returns an instance of the default key-selection manager.
`protected void`	**`fireActionEvent`**`()` Notifies all listeners that have registered interest for notification on this event type.
`protected void`	**`fireItemStateChanged`**`(ItemEvent e)` Notifies all listeners that have registered interest for notification on this event type.
`void`	**`firePopupMenuCanceled`**`()` Notifies `PopupMenuListeners` that the popup portion of the combo box has been canceled.
`void`	**`firePopupMenuWillBecomeInvisible`**`()` Notifies `PopupMenuListeners` that the popup portion of the combo box has become invisible.
`void`	**`firePopupMenuWillBecomeVisible`**`()` Notifies `PopupMenuListeners` that the popup portion of the combo box will become visible.
`AccessibleContext`	**`getAccessibleContext`**`()` Gets the AccessibleContext associated with this JComboBox.
`Action`	**`getAction`**`()` Returns the currently set `Action` for this `ActionEvent` source, or `null` if no `Action` is set.
`String`	**`getActionCommand`**`()` Returns the action command that is included in the

	event sent to action listeners.
ActionListener[]	getActionListeners() Returns an array of all the ActionListeners added to this JComboBox with addActionListener().
ComboBoxEditor	getEditor() Returns the editor used to paint and edit the selected item in the JComboBox field.
E	getItemAt(int index) Returns the list item at the specified index.
int	getItemCount() Returns the number of items in the list.
ItemListener[]	getItemListeners() Returns an array of all the ItemListeners added to this JComboBox with addItemListener().
JComboBox.KeySelectionManager	getKeySelectionManager() Returns the list's key-selection manager.
int	getMaximumRowCount() Returns the maximum number of items the combo box can display without a scrollbar
ComboBoxModel<E>	getModel() Returns the data model currently used by the JComboBox.
PopupMenuListener[]	getPopupMenuListeners() Returns an array of all the PopupMenuListeners added to this JComboBox with addPopupMenuListener().
E	getPrototypeDisplayValue() Returns the "prototypical display" value - an Object used for the calculation of the display height and width.
ListCellRenderer<? super E>	getRenderer() Returns the renderer used to display the selected item in the JComboBox field.
int	getSelectedIndex() Returns the first item in the list that matches the given item.
Object	getSelectedItem()

	Returns the current selected item.
`Object[]`	`getSelectedObjects()` Returns an array containing the selected item.
`ComboBoxUI`	`getUI()` Returns the L&F object that renders this component.
`String`	`getUIClassID()` Returns the name of the L&F class that renders this component.
`void`	`hidePopup()` Causes the combo box to close its popup window.
`void`	`insertItemAt(E item, int index)` Inserts an item into the item list at a given index.
`protected void`	`installAncestorListener()`
`void`	`intervalAdded(ListDataEvent e)` This method is public as an implementation side effect.
`void`	`intervalRemoved(ListDataEvent e)` This method is public as an implementation side effect.
`boolean`	`isEditable()` Returns true if the `JComboBox` is editable.
`boolean`	`isLightWeightPopupEnabled()` Gets the value of the `lightWeightPopupEnabled` property.
`boolean`	`isPopupVisible()` Determines the visibility of the popup.
`protected String`	`paramString()` Returns a string representation of this `JComboBox`.
`protected boolean`	`processKeyBinding(KeyStroke ks, KeyEvent e, int condition, boolean pressed)` Invoked to process the key bindings for `ks` as the result of the `KeyEvent` `e`.
`void`	`processKeyEvent(KeyEvent e)` Handles `KeyEvent`s, looking for the Tab key.
`void`	`removeActionListener(ActionListener l)`

	Removes an `ActionListener`.
`void`	**removeAllItems**`()` Removes all items from the item list.
`void`	**removeItem**`(Object anObject)` Removes an item from the item list.
`void`	**removeItemAt**`(int anIndex)` Removes the item at `anIndex` This method works only if the `JComboBox` uses a mutable data model.
`void`	**removeItemListener**`(ItemListener aListener)` Removes an `ItemListener`.
`void`	**removePopupMenuListener**`(PopupMenuListener l)` Removes a `PopupMenuListener`.
`protected void`	**selectedItemChanged**`()` This protected method is implementation specific.
`boolean`	**selectWithKeyChar**`(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**`(Action a)` Sets the `Action` for the `ActionEvent` source.
`void`	**setActionCommand**`(String aCommand)` Sets the action command that should be included in the event sent to action listeners.
`void`	**setEditable**`(boolean aFlag)` Determines whether the `JComboBox` field is editable.
`void`	**setEditor**`(ComboBoxEditor anEditor)` Sets the editor used to paint and edit the selected item in the `JComboBox` field.
`void`	**setEnabled**`(boolean b)` Enables the combo box so that items can be selected.
`void`	**setKeySelectionManager**`(JComboBox.KeySelectionManager aManager)` Sets the object that translates a keyboard character

	into a list selection.
`void`	`setLightWeightPopupEnabled(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(int count)` Sets the maximum number of rows the `JComboBox` displays.
`void`	`setModel(ComboBoxModel<E> aModel)` Sets the data model that the `JComboBox` uses to obtain the list of items.
`void`	`setPopupVisible(boolean v)` Sets the visibility of the popup.
`void`	`setPrototypeDisplayValue(E prototypeDisplayValue)` Sets the prototype display value used to calculate the size of the display for the UI portion.
`void`	`setRenderer(ListCellRenderer<? super E> aRenderer)` Sets the renderer that paints the list items and the item selected from the list in the JComboBox field.
`void`	`setSelectedIndex(int anIndex)` Selects the item at index `anIndex`.
`void`	`setSelectedItem(Object anObject)` Sets the selected item in the combo box display area to the object in the argument.
`void`	`setUI(ComboBoxUI ui)` Sets the L&F object that renders this component.
`void`	`showPopup()` Causes the combo box to display its popup window.
`void`	`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()` Creates an initially unselected radio button with no set text.
`JRadioButton(Action a)` Creates a radiobutton where properties are taken from the Action supplied.
`JRadioButton(Icon icon)` Creates an initially unselected radio button with the specified image but no text.
`JRadioButton(Icon icon, boolean selected)` Creates a radio button with the specified image and selection state, but no text.
`JRadioButton(String text)` Creates an unselected radio button with the specified text.
`JRadioButton(String text, boolean selected)` Creates a radio button with the specified text and selection state.
`JRadioButton(String text, Icon icon)` Creates a radio button that has the specified text and image, and that is initially unselected.
`JRadioButton(String text, Icon 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`	`getAccessibleContext()` Gets the AccessibleContext associated with this JRadioButton.
`String`	`getUIClassID()` Returns the name of the L&F class that renders this component.
`protected String`	`paramString()` Returns a string representation of this JRadioButton.
`void`	`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()` Creates an empty (no viewport view) `JScrollPane` where both horizontal and vertical scrollbars appear when needed.
`JScrollPane(Component 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(Component 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(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`	`createHorizontalScrollBar()` Returns a `JScrollPane.ScrollBar` by default.
`JScrollBar`	`createVerticalScrollBar()` Returns a `JScrollPane.ScrollBar` by default.
`protected JViewport`	`createViewport()` Returns a new `JViewport` by default.
`AccessibleContext`	`getAccessibleContext()` Gets the AccessibleContext associated with this JScrollPane.
`JViewport`	`getColumnHeader()` Returns the column header.
`Component`	`getCorner(String key)` Returns the component at the specified corner.
`JScrollBar`	`getHorizontalScrollBar()` Returns the horizontal scroll bar that controls the

	viewport's horizontal view position.
`int`	`getHorizontalScrollBarPolicy()` Returns the horizontal scroll bar policy value.
`JViewport`	`getRowHeader()` Returns the row header.
`ScrollPaneUI`	`getUI()` Returns the look and feel (L&F) object that renders this component.
`String`	`getUIClassID()` Returns the suffix used to construct the name of the L&F class used to render this component.
`JScrollBar`	`getVerticalScrollBar()` Returns the vertical scroll bar that controls the viewports vertical view position.
`int`	`getVerticalScrollBarPolicy()` Returns the vertical scroll bar policy value.
`JViewport`	`getViewport()` Returns the current `JViewport`.
`Border`	`getViewportBorder()` Returns the `Border` object that surrounds the viewport.
`Rectangle`	`getViewportBorderBounds()` Returns the bounds of the viewport's border.
`boolean`	`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()` Indicates whether or not scrolling will take place in response to the mouse wheel.
`protected String`	`paramString()` Returns a string representation of this `JScrollPane`.
`void`	`setColumnHeader(JViewport 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`**`(Component view)` Creates a column-header viewport if necessary, sets its view, and then adds the column-header viewport to the scrollpane.
`void`	**`setComponentOrientation`**`(ComponentOrientation co)` Sets the orientation for the vertical and horizontal scrollbars as determined by the `ComponentOrientation` argument.
`void`	**`setCorner`**`(String key, Component corner)` Adds a child that will appear in one of the scroll panes corners, if there's room.
`void`	**`setHorizontalScrollBar`**`(JScrollBar horizontalScrollBar)` Adds the scrollbar that controls the viewport's horizontal view position to the scrollpane.
`void`	**`setHorizontalScrollBarPolicy`**`(int policy)` Determines when the horizontal scrollbar appears in the scrollpane.
`void`	**`setLayout`**`(LayoutManager layout)` Sets the layout manager for this `JScrollPane`.
`void`	**`setRowHeader`**`(JViewport 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`**`(Component view)` Creates a row-header viewport if necessary, sets its view and then adds the row-header viewport to the scrollpane.
`void`	**`setUI`**`(ScrollPaneUI ui)` Sets the `ScrollPaneUI` object that provides the look and feel (L&F) for this component.
`void`	**`setVerticalScrollBar`**`(JScrollBar verticalScrollBar)` Adds the scrollbar that controls the viewports vertical

	view position to the scrollpane.
void	`setVerticalScrollBarPolicy(int policy)` Determines when the vertical scrollbar appears in the scrollpane.
void	`setViewport(JViewport 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(Border viewportBorder)` Adds a border around the viewport.
void	`setViewportView(Component view)` Creates a viewport if necessary and then sets its view.
void	`setWheelScrollingEnabled(boolean handleWheel)` Enables/disables scrolling in response to movement of the mouse wheel.
void	`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()` Creates an initially unselected check box button with no text, no icon.
`JCheckBox(Action a)` Creates a check box where properties are taken from the Action supplied.
`JCheckBox(Icon icon)` Creates an initially unselected check box with an icon.
`JCheckBox(Icon icon, boolean selected)` Creates a check box with an icon and specifies whether or not it is initially selected.
`JCheckBox(String text)`

Creates an initially unselected check box with text.
`JCheckBox(String text, boolean selected)`

Creates a check box with text and specifies whether or not it is initially selected.
`JCheckBox(String text, Icon icon)`

Creates an initially unselected check box with the specified text and icon.
`JCheckBox(String text, Icon 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`	`getAccessibleContext()` Gets the AccessibleContext associated with this JCheckBox.
`String`	`getUIClassID()` Returns a string that specifies the name of the L&F class that renders this component.
`boolean`	`isBorderPaintedFlat()` Gets the value of the `borderPaintedFlat` property.
`protected String`	`paramString()` Returns a string representation of this JCheckBox.
`void`	`setBorderPaintedFlat(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()` Resets the UI property to a value from the current look and feel.

Características da classe JOPTIONPANE

- Constructor Summary

Constructors

Constructor and Description
`JOptionPane()`

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Creates a `JOptionPane` with a test message.

`JOptionPane``(Object 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``(Object message, int messageType)`

Creates an instance of `JOptionPane` to display a message with the specified message type and the default options,

`JOptionPane``(Object message, int messageType, int optionType)`

Creates an instance of `JOptionPane` to display a message with the specified message type and options.

`JOptionPane``(Object message, int messageType, int optionType, Icon icon)`

Creates an instance of `JOptionPane` to display a message with the specified message type, options, and icon.

`JOptionPane``(Object message, int messageType, int optionType, Icon icon, Object[] options)`

Creates an instance of `JOptionPane` to display a message with the specified message type, icon, and options.

`JOptionPane``(Object message, int messageType, int optionType, Icon icon, Object[] options, Object 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`	**`createDialog`**`(Component parentComponent, String title)` Creates and returns a new `JDialog` wrapping `this` centered on the `parentComponent` in the `parentComponent`'s frame.
`JDialog`	**`createDialog`**`(String title)` Creates and returns a new parentless `JDialog` with the specified title.
`JInternalFrame`	**`createInternalFrame`**`(Component parentComp`

	onent, String title) Creates and returns an instance of JInternalFrame.
AccessibleContext	**getAccessibleContext**() Returns the AccessibleContext associated with this JOptionPane.
static JDesktopPane	**getDesktopPaneForComponent**(Component parentComponent) Returns the specified component's desktop pane.
static Frame	**getFrameForComponent**(Component parentComponent) Returns the specified component's Frame.
Icon	**getIcon**() Returns the icon this pane displays.
Object	**getInitialSelectionValue**() Returns the input value that is displayed as initially selected to the user.
Object	**getInitialValue**() Returns the initial value.
Object	**getInputValue**() Returns the value the user has input, if wantsInput is true.
int	**getMaxCharactersPerLineCount**() Returns the maximum number of characters to place on a line in a message.
Object	**getMessage**() Returns the message-object this pane displays.
int	**getMessageType**() Returns the message type.
Object[]	**getOptions**() Returns the choices the user can make.
int	**getOptionType**() Returns the type of options that are displayed.
static Frame	**getRootFrame**() Returns the Frame to use for the class methods in which a frame is not provided.
Object[]	**getSelectionValues**()

	Returns the input selection values.
OptionPaneUI	**getUI**() Returns the UI object which implements the L&F for this component.
String	**getUIClassID**() Returns the name of the UI class that implements the L&F for this component.
Object	**getValue**() Returns the value the user has selected.
boolean	**getWantsInput**() Returns the value of the `wantsInput` property.
protected String	**paramString**() Returns a string representation of this `JOptionPane`.
void	**selectInitialValue**() Requests that the initial value be selected, which will set focus to the initial value.
void	**setIcon**(Icon newIcon) Sets the icon to display.
void	**setInitialSelectionValue**(Object newValue) Sets the input value that is initially displayed as selected to the user.
void	**setInitialValue**(Object newInitialValue) Sets the initial value that is to be enabled -- the `Component` that has the focus when the pane is initially displayed.
void	**setInputValue**(Object newValue) Sets the input value that was selected or input by the user.
void	**setMessage**(Object newMessage) Sets the option pane's message-object.
void	**setMessageType**(int newType) Sets the option pane's message type.
void	**setOptions**(Object[] newOptions) Sets the options this pane displays.
void	**setOptionType**(int newType) Sets the options to display.
static void	**setRootFrame**(Frame newRootFrame)

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

	Sets the frame to use for class methods in which a frame is not provided.
`void`	**setSelectionValues**(Object[] newValues) Sets the input selection values for a pane that provides the user with a list of items to choose from.
`void`	**setUI**(OptionPaneUI ui) Sets the UI object which implements the L&F for this component.
`void`	**setValue**(Object newValue) Sets the value the user has chosen.
`void`	**setWantsInput**(boolean newValue) Sets the `wantsInput` property.
`static int`	**showConfirmDialog**(Component parentComponent, Object message) Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	**showConfirmDialog**(Component parentComponent, Object message, String title, int optionType) Brings up a dialog where the number of choices is determined by the `optionType` parameter.
`static int`	**showConfirmDialog**(Component parentComponent, Object message, String 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**(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon) Brings up a dialog with a specified icon, where the number of choices is determined by the `optionType` parameter.
`static String`	**showInputDialog**(Component parentComponent, Object message)

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

	Shows a question-message dialog requesting input from the user parented to `parentComponent`.
`static String`	**`showInputDialog`**`(Component parentComponent, Object message, Object initialSelectionValue)` Shows a question-message dialog requesting input from the user and parented to `parentComponent`.
`static String`	**`showInputDialog`**`(Component parentComponent, Object message, String 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`	**`showInputDialog`**`(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object 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`	**`showInputDialog`**`(Object message)` Shows a question-message dialog requesting input from the user.
`static String`	**`showInputDialog`**`(Object message, Object initialSelectionValue)` Shows a question-message dialog requesting input from the user, with the input value initialized to `initialSelectionValue`.
`static int`	**`showInternalConfirmDialog`**`(Component parentComponent, Object message)` Brings up an internal dialog panel with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	**`showInternalConfirmDialog`**`(Component parentComponent, Object message, String title, int optionType)`

POCH
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

	Brings up a internal dialog panel where the number of choices is determined by the `optionType` parameter.
`static int`	`**showInternalConfirmDialog**(Component parentComponent, Object message, String 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**(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon)` Brings up an internal dialog panel with a specified icon, where the number of choices is determined by the `optionType` parameter.
`static String`	`**showInternalInputDialog**(Component parentComponent, Object message)` Shows an internal question-message dialog requesting input from the user parented to `parentComponent`.
`static String`	`**showInternalInputDialog**(Component parentComponent, Object message, String 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`	`**showInternalInputDialog**(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object 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`**`(Component parentComponent, Object message)` Brings up an internal confirmation dialog panel.
`static void`	**`showInternalMessageDialog`**`(Component parentComponent, Object message, String 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`**`(Component parentComponent, Object message, String title, int messageType, Icon icon)` Brings up an internal dialog panel displaying a message, specifying all parameters.
`static int`	**`showInternalOptionDialog`**`(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon, Object[] options, Object 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`**`(Component parentComponent, Object message)` Brings up an information-message dialog titled "Message".
`static void`	**`showMessageDialog`**`(Component parentComponent, Object message, String title, int messageType)` Brings up a dialog that displays a message using a default icon determined by the `messageType` parameter.
`static void`	**`showMessageDialog`**`(Component parentComponent, Object message, String title, int messageType, Icon icon)`

	Brings up a dialog displaying a message, specifying all parameters.
static int	`showOptionDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon, Object[] options, Object 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()` Notification from the `UIManager` that the L&F has changed.

Características da classe BUFFEREDINPUTSTREAM

- Constructor Summary

Constructors

Constructor and Description
`BufferedInputStream(InputStream in)` Creates a `BufferedInputStream` and saves its argument, the input stream `in`, for later use.
`BufferedInputStream(InputStream 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()` 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()` Closes this input stream and releases any system resources associated with the stream.
`void`	`mark(int readlimit)` See the general contract of the `mark` method of `InputStream`.
`boolean`	`markSupported()` Tests if this input stream supports the `mark` and `reset` methods.
`int`	`read()` See the general contract of the `read` method of `InputStream`.
`int`	`read(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()` See the general contract of the `reset` method of `InputStream`.
`long`	`skip(long n)` See the general contract of the `skip` method of `InputStream`.

Características da classe BufferedOutputStream

- Constructor Summary

Constructors

Constructor and Description
`BufferedOutputStream(OutputStream out)` Creates a new buffered output stream to write data to the specified underlying output stream.
`BufferedOutputStream(OutputStream 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()` Flushes this buffered output stream.
`void`	`write(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(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)