

POO

Programação Orientada a Objetos

Material 007

Professor Maromo





Agenda



- Interface Gráfica
- javax.swing X java.awt
- JavaFX
- Criar controles deslizantes, menus e janelas.
- Modificar aparência e comportamento de uma GUI
- Gerenciadores de layout

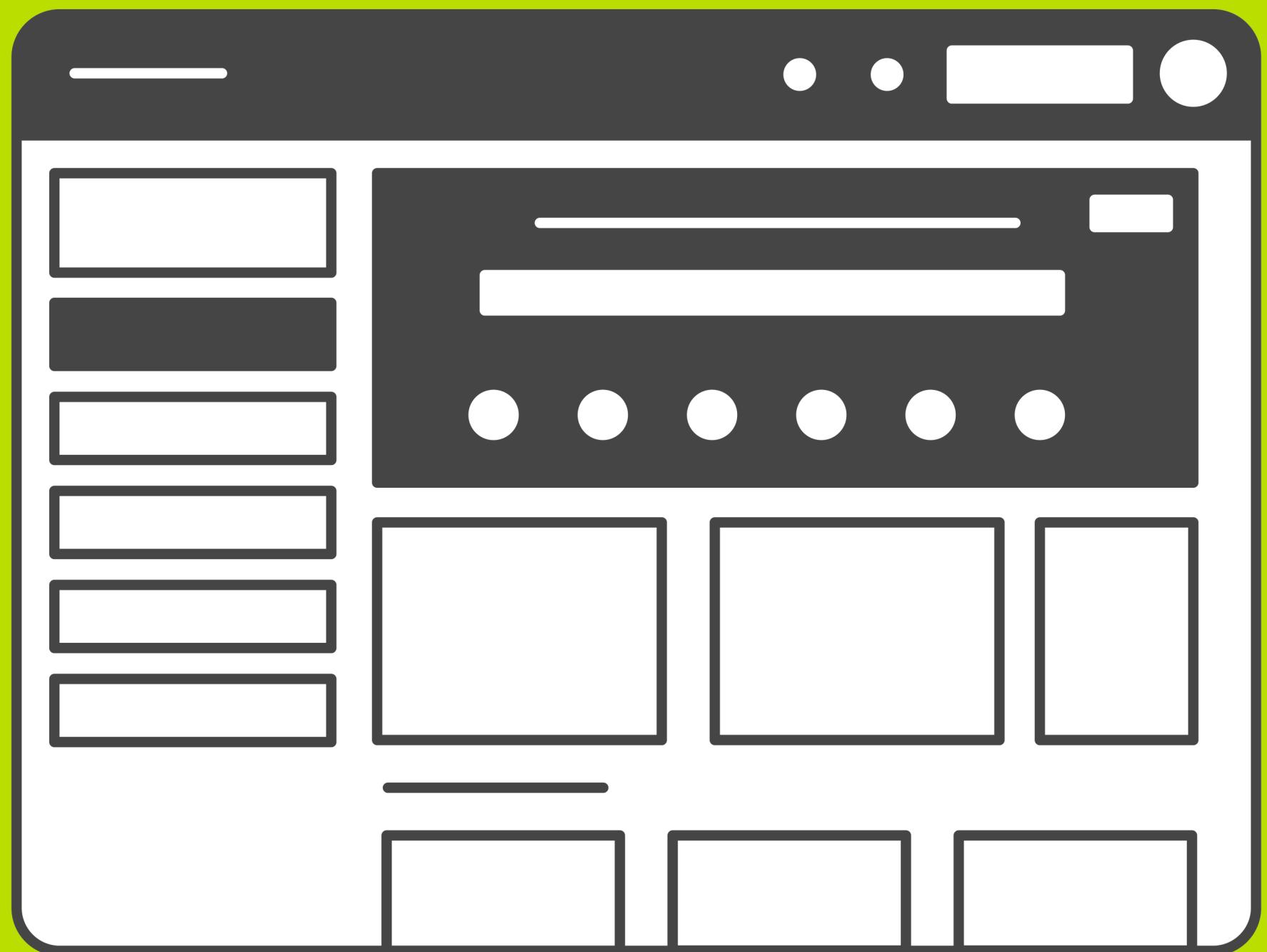
Introdução

- GUI é feita por meio de bibliotecas de classes.
- A AWT foi a primeira API para interfaces gráficas a surgir no Java e foi, mais tarde, superada pelo Swing (a partir do Java 1.2)
- A maneira como as classes dessa biblioteca funcionam garante a criação de elementos de interface com o usuário (Windows, Mac, Linux, Solaris, etc...).
- Atualmente além dessas podemos usar o JavaFX, estratégia da empresa para levar o Java ao desenvolvimento fácil de interfaces ricas com o usuário.



Exemplo

Botões, listas, menus, componentes de textos, containers, caixas de diálogo (abrir, salvar), etc...



Classe: TelaMovimento

Vamos criar um exemplo chamado **ControleDeCaixa**, com as seguintes classes:

- **Caixa (Classe Java)**
- **GerenciarCaixa (Classe Java)**
- **TelaMovimento (Classe Java)**



Abordagem

Primeiramente vamos criar o design, conforme o Diagrama da Classe ao lado:



© TelaMovimento		
(f) ♀	txtMsg	TextArea
(f) ♀	cmdEntrada	Button
(f) ♀	cmdSair	Button
(f) ♀	dLabel	Dimension
(f) ♀	lblValor	Label
(f) ♀	dFrame	Dimension
(f) ♀	dTextArea	Dimension
(f) ♀	lblSaldo	Label
(f) ♀	cmdRetirada	Button
(f) ♀	dButton	Dimension
(f) ♀	txtValor	TextField
(f) ♀	cmdConsulta	Button
(f) ♀	txtSaldo	TextField

Classe TelaMovimento Designer

```
import javax.swing.*;  
import java.awt.*;  
  
public class TelaMovimento extends JFrame {  
  
    protected Dimension dLabel, dButton, dTextArea, dTextField, dFrame;  
    protected Button cmdEntrada, cmdRetirada, cmdConsulta, cmdSair;  
    protected TextField txtValor, txtSaldo;  
    protected Label lblValor, lblSaldo;  
    protected TextArea txtMsg;  
  
    public TelaMovimento() {  
        //Dimensão dos componentes  
        dFrame = new Dimension(350, 400);  
        dLabel = new Dimension(40, 20);  
        dTextField = new Dimension(150,20);  
        dButton = new Dimension(95, 20);  
        dTextArea = new Dimension(300, 140);  
        //Definindo propriedades da janela  
        setTitle("Controle de Caixa");  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        setResizable(false);  
        setSize(dFrame);  
        setLayout(null);  
    }  
}
```

```
lblValor = new Label("Valor");
lblValor.setSize(dLabel);
lblValor.setLocation(25, 50);
add(lblValor);
lblSaldo = new Label("Saldo");
lblSaldo.setSize(dLabel);
lblSaldo.setLocation(25, 80);
add(lblSaldo);
txtValor = new TextField(null);
txtValor.setSize(dTextField);
txtValor.setLocation(75,50);
add(txtValor);
txtSaldo = new TextField(null);
txtSaldo.setSize(dTextField);
txtSaldo.setLocation(75,80);
add(txtSaldo);
```

```
cmdEntrada = new Button("Depósito");
cmdEntrada.setSize(dButton);
cmdEntrada.setLocation(25, 150);
add(cmdEntrada);
cmdConsulta = new Button("Saldo");
cmdConsulta.setSize(dButton);
cmdConsulta.setLocation(25, 185);
add(cmdConsulta);
cmdRetirada = new Button("Saque");
cmdRetirada.setSize(dButton);
cmdRetirada.setLocation(225, 150);
add(cmdRetirada);
cmdSair = new Button("Sair");
cmdSair.setSize(dButton);
cmdSair.setLocation(225, 185);
add(cmdSair);
txtMsg = new TextArea(null);
txtMsg.setSize(dTextArea);
txtMsg.setLocation(25,220);
add(txtMsg);}
```

Resultado da Aparência



Controle de Caixa

Valor:

Saldo:

[Entrada](#) [Retirada](#)

[Consulta](#) [Sair](#)

Componentes Utilizados

- **JFrame (javax.swing.JFrame)**

- Um objeto do tipo JFrame contém os elementos básicos para manipular uma janela: abrir, fechar, mover e redimensionar.

- **TextField (java.awt.*)**

- Caixa de Texto

- **Label (java.awt.*)**

- Rótulo

- **Button (java.awt.*)**

- Botões de comando

- **TextArea (java.awt.*)**

- Caixa de Texto com linhas e barras de rolagens (horizontal e vertical)

Interfaces Listeners

- Para se controlar os eventos de um formulário deve se incluir na classe as interfaces gerenciadoras de eventos também chamadas de **listeners** da classe.
- Listener quer dizer ouvinte, e existem uma série de listeners que uma classe pode implementar, que recebem uma informação vinda dos objetos da tela cada vez que um evento é detectado por eles.

Interfaces Listeners

- Podem ocorrer eventos de vários tipos, dentre eles, os mais comuns são:
 - **Eventos de ação**
 - Implementados pela interface ActionListener
 - **Eventos de Janela**
 - Implementados pela interface WindowListener
 - **Eventos de Teclado**
 - Implementados pela interface KeyListener
 - **Eventos de Mouse**
 - Implementados pela interface MouseListener

Interfaces Listeners

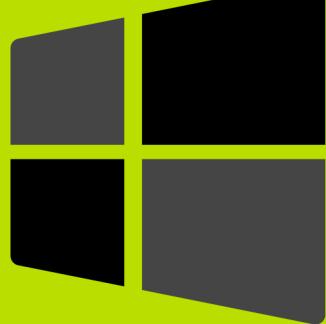
- A interface **ActionListener** possui 1 método que é:

Método	Descrição
actionPerformed()	Quando ocorrer um evento de ação é invocado esse método. Ex: clique do mouse sobre um determinado botão.

ACTION!

Interfaces Listeners

WindowListener possui 8 métodos



Método	Descrição
windowActivated	Chamado quando a janela é definida como a janela ativa.
windowClosed	Chamado quando uma janela foi fechada.
windowClosing	Chamado quando o usuário tenta fechar a janela.
windowDeactivated	Chamado quando uma janela não é mais a janela ativa.
windowDeiconified	Chamado quando uma janela muda de um estado minimizado para um estado normal.
windowIconified	Chamado quando uma janela muda de um estado normal para um estado minimizado.
windowOpened	Chamado pela primeira vez quando uma janela é tornada visível.
windowStateChanged	Chamado quando o estado da janela é alterado.

Interfaces Listeners

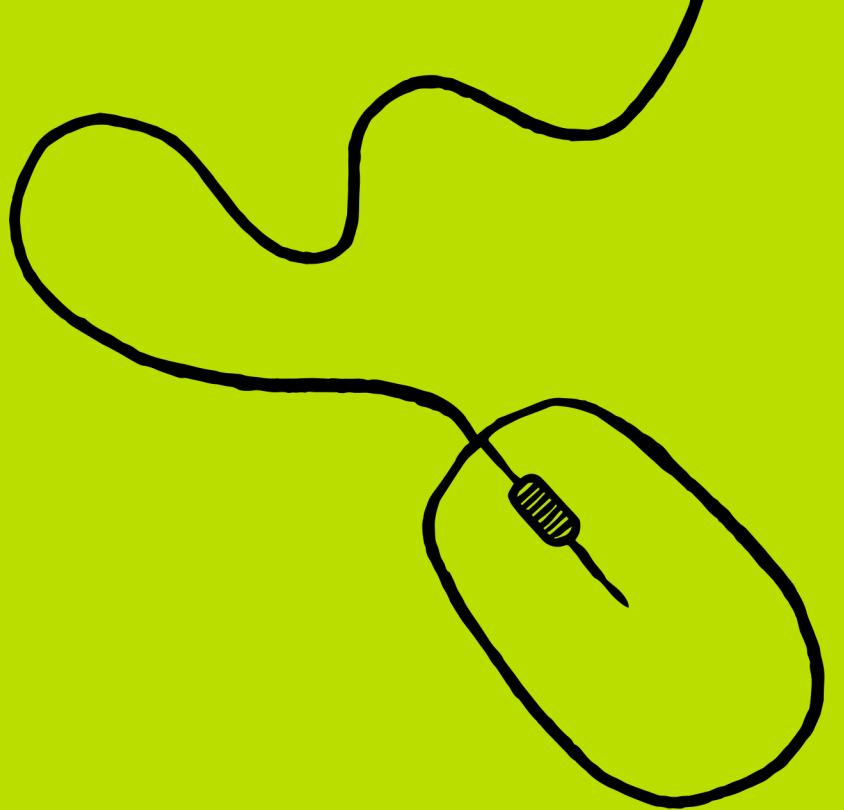
A interface KeyListener possui 3 métodos, que são:



Método	Descrição
keyTyped(KeyEvent)	Após o usuário digitar um caractere Unicode em um componente que detém o foco.
keyPressed(KeyEvent)	Após o usuário pressionar uma tecla no componente que detém o foco.
keyReleased(KeyEvent)	Após o usuário liberar uma tecla no componente que detém o foco.

Interfaces Listeners

- A interface MouseListener possui métodos, que são:



Método	Descrição
mouseClicked(MouseEvent e)	Quando o botão do mouse é clicado (pressionado e liberado) em um componente.
mouseEntered(MouseEvent e)	Quando o mouse entra na área de um componente.
mouseExited(MouseEvent e)	Quando o mouse sai da área de um componente.
mousePressed(MouseEvent e)	Quando o botão do mouse é pressionado sobre um componente.
mouseReleased(MouseEvent e)	Quando o botão do mouse é liberado sobre um componente.

Interfaces Listeners



- Já se sabe que a implementação de uma interface é feita através do comando **implements** inserido na abertura da classe.
- Os métodos das interfaces Listeners são **abstratos**, portanto, se implementarmos uma interface listener todos os seus métodos devem ser implementados na classe.

Classe: TelaMovimento

- Implementando o Listener na classe,
- Modifique ou acrescente as linhas destacadas.
Primeiro os WindowListeners

```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.WindowEvent;  
import java.awt.event.WindowListener;  
  
public class TelaMovimento extends JFrame implements WindowListener {
```

Classe: TelaMovimento

```
add(txtMsg);
addWindowListener(this);
}

@Override
public void windowOpened(WindowEvent e) {}

@Override
public void windowClosing(WindowEvent e) {}

@Override
public void windowClosed(WindowEvent e) {}

@Override
public void windowIconified(WindowEvent e) {}

@Override
public void windowDeiconified(WindowEvent e) {}

@Override
public void windowActivated(WindowEvent e) {}

@Override
public void windowDeactivated(WindowEvent e) {}
```

Classe: TelaMovimento

- Modifique ou acrescente as linhas destacadas. Agora os ActionListeners.

```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.WindowEvent;  
import java.awt.event.WindowListener;  
  
public class TelaMovimento extends JFrame implements WindowListener, ActionListener {
```

Classe: TelaMovimento – Botão Sair

```
cmdSair = new Button("Sair");
cmdSair.setSize(dButton);
cmdSair.setLocation(225, 185);
cmdSair.addActionListener(this);
add(cmdSair);
```

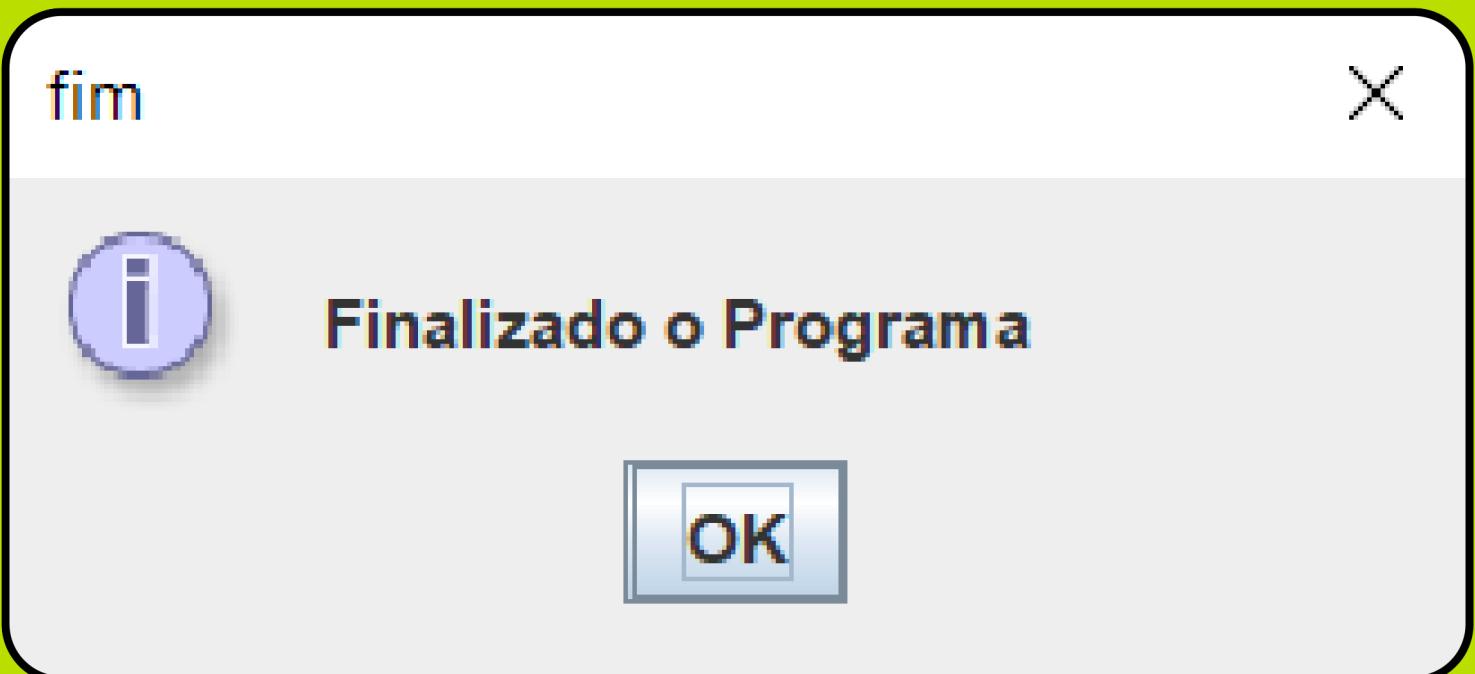
Construtor

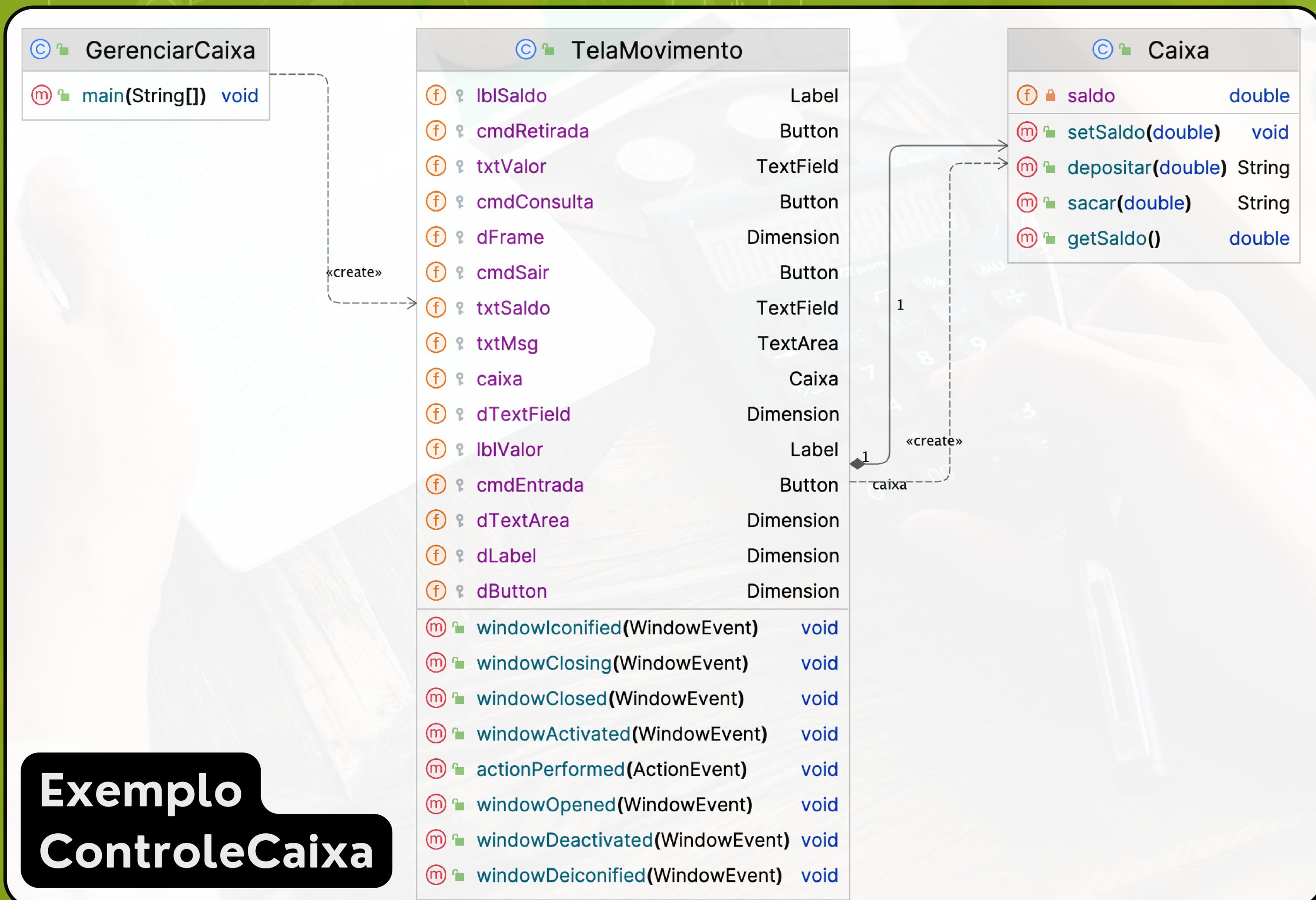
Método

```
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==cmdSair){
        JOptionPane.showMessageDialog(
            null,"Fim do programa",
            "Fim",JOptionPane.INFORMATION_MESSAGE);
        System.exit(0);
    }
}
```



Resultado do Clique no Botão Sair





Classe Caixa

```
public class Caixa {  
    private double saldo;  
  
    public double getSaldo() {  
        return saldo;  
    }  
  
    public void setSaldo(double saldo) {  
        this.saldo = saldo;  
    }  
}
```

```
    public String depositar(double valor){  
        if(valor < 0){  
            return "Erro ao depositar, valor inválido";  
        }  
        saldo += valor;  
        return "Depósito efetuado com sucesso";  
    }  
  
    public String sacar(double valor){  
        if(valor > getSaldo()){  
            return "Sem saldo para a transação";  
        }  
        saldo -= valor;  
        return "Saque efetuado com sucesso";  
    }  
}
```

Classe TelaMovimento

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;

public class TelaMovimento extends JFrame implements WindowListener, ActionListener {

    protected Dimension dLabel, dButton, dTextArea, dTextField, dFrame;
    protected Button cmdEntrada, cmdRetirada, cmdConsulta, cmdSair;
    protected TextField txtValor, txtSaldo;
    protected Label lblValor, lblSaldo;
    protected TextArea txtMsg;
protected Caixa caixa = new Caixa();
```



Classe TelaMovimento

Botão Entrada

```
cmdEntrada = new Button("Depósito");
cmdEntrada.setSize(dButton);
cmdEntrada.setLocation(25, 150);
cmdEntrada.addActionListener(this);
```

```
public void actionPerformed(ActionEvent e) {
    //abaixo do getSource()==cmdSair
    if(e.getSource()==cmdEntrada){
        double valor = Double.parseDouble(txtValor.getText());
        String retorno = caixa.depositar(valor);
        txtMsg.append(retorno + "\n");
        txtValor.setText(null);
        txtValor.requestFocus();
    }
}
```

Classe TelaMovimento

Botão Retirada

```
cmdRetirada = new Button("Saque");
cmdRetirada.setSize(dButton);
cmdRetirada.setLocation(225, 150);
cmdRetirada.addActionListener(this);
```

```
public void actionPerformed(ActionEvent e) {
    //abaixo do getSource()==cmdEntrada
    if(e.getSource()==cmdRetirada){
        double valor = Double.parseDouble(txtValor.getText());
        String retorno = caixa.sacar(valor);
        txtMsg.append(retorno + "\n");
        txtValor.setText(null);
        txtValor.requestFocus();
    }
}
```

Classe TelaMovimento

Botão Consulta

```
cmdConsulta = new Button("Saldo");
cmdConsulta.setSize(dButton);
cmdConsulta.setLocation(25, 185);
cmdConsulta.addActionListener(this);

public void actionPerformed(ActionEvent e) {
    //abaixo do getSource() == cmdRetirada
    if(e.getSource() == cmdConsulta){
        txtSaldo.setText(Double.toString(caixa.getSaldo()));
        txtMsg.append("Consulta, saldo atual: " + txtSaldo.getText() + "\n");
        txtValor.setText(null);
        txtValor.requestFocus();
    }
}
```

Classe GerenciarCaixa

Método Main

```
public class GerenciarCaixa {  
    public static void main(String[] args) {  
        TelaMovimento tela = new TelaMovimento();  
        tela.setVisible(true);  
    }  
}
```

Resultado

Controle de Caixa

Valor

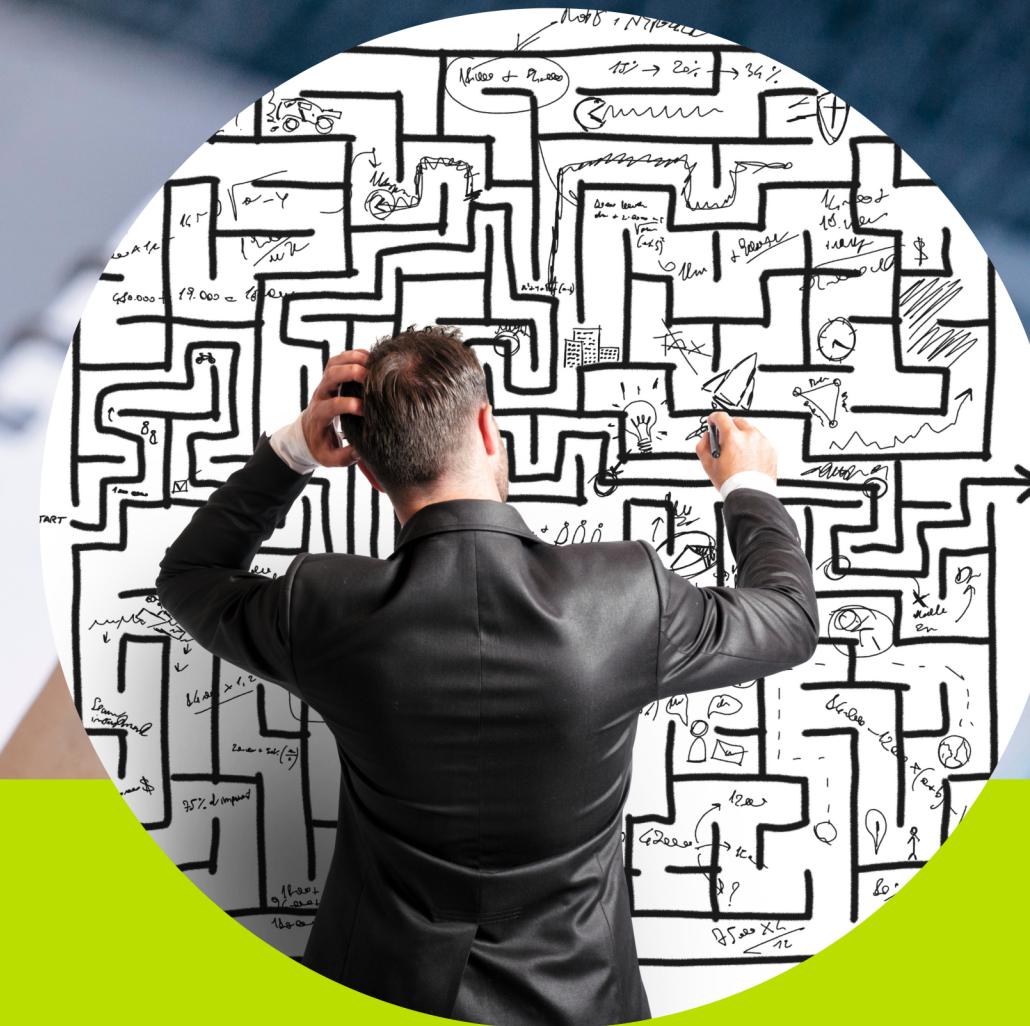
Saldo

Depósito efetuado com sucesso
Depósito efetuado com sucesso
Consulta, saldo atual: 300.0
Saque efetuado com sucesso
Consulta, saldo atual: 65.0





EXERCÍCIOS PARA PRÁTICA



Exercício: Calculadora

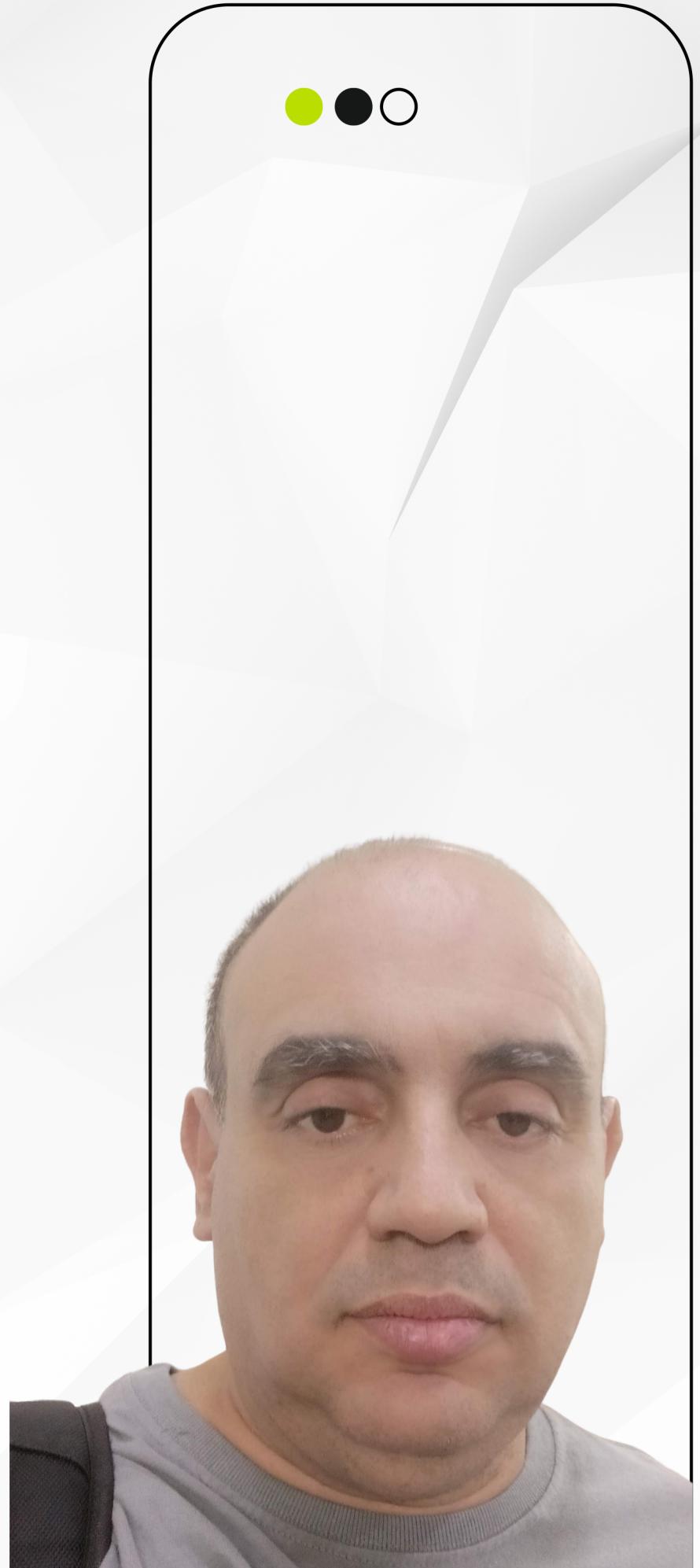
- Sua missão:
 - Criar um programa JAVA utilizando-se dos operadores básicos para criar uma calculadora com as 04 operações básicas.
- Nota:
 - Usando Interface Gráfica com Usuário

Exercício: Calculadora

Método	Descrição
soma	Esse método deve receber como parâmetros dois valores do tipo double. Deve calcular no campo result o resultado da soma.
subtracao	Esse método deve receber como parâmetros dois valores do tipo double. Deve calcular no campo result o resultado da subtração.
multiplicacao	Esse método deve receber como parâmetros dois valores do tipo double. Deve processar no campo result o resultado da multiplicação.
divisao	Esse método deve receber como parâmetros dois valores do tipo double. Deve calcular no campo result o resultado da divisão. Caso o divisor seja igual a ZERO deve ocorrer uma Exception informando que é impossível dividir por zero. Mostrar zero como resultado no campo result.

Aprendendo mais...

- Link com artigo apresentando exemplos práticos usando JavaFX.
- <http://code.makery.ch/library/javafx-8-tutorial-pt/part1/>



Obrigado

fim

Até a próxima aula



Referências Bibliográficas



- Mendes; **Java com Ênfase em Orientação a Objetos**, Novatec.
- Deitel; **Java, como programar** – 10º edição. Java SE 7 e 8
- Arnold, Gosling, Holmes; **A linguagem de programação Java** – 4º edição.
- Apostilas da Caelum.