



PROGRAMAÇÃO ORIENTADA A OBJETOS [POO]

Material 07 – POO_07

Prof. Mestre Marcos Roberto de Moraes [Maromo]

Interface gráfica com o usuário (GUI – AWT e SWING)

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.

Exemplos

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

Abordagem (AWT / Swing – coexistência)

Exemplo

- Vamos criar um exemplo chamado ControleDeCaixa, com as seguintes classes:
 - ▣ Caixa (Classe Java)
 - ▣ Principal (Classe Java)
 - ▣ TelaMovimento (Classe Java)

Classe: TelaMovimento

- Primeiramente vamos criar o design, conforme o Diagrama da Classe Abaixo:



```

1 package com.faculdade;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class TelaMovimento extends JFrame {
7     protected Dimension dFrame, dLabel, dTextField, dButton, dTextArea;
8     protected Button cmdEntrada, cmdRetirada, cmdConsulta, cmdSair;
9     protected TextField txtValor, txtSaldo;
10    protected Label lblValor, lblSaldo;
11    protected TextArea txtMsg;
12
13    public TelaMovimento() {
14        //Dimensão dos componentes
15        dFrame = new Dimension( width: 350, height: 400);
16        dLabel = new Dimension( width: 40, height: 20);
17        dTextField = new Dimension( width: 150, height: 20);
18        dButton = new Dimension( width: 95, height: 20);
19        dTextArea = new Dimension( width: 300, height: 140);
20
21        //Definição das propriedades da Janela Principal
22        setTitle("Controle de Caixa");
23        setResizable(false);
24        setSize(dFrame);
25        setLocation( x: 200, y: 200);
26        setLayout(null);
27    }
28 }

```

**Classe TelaMovimento
Designer**

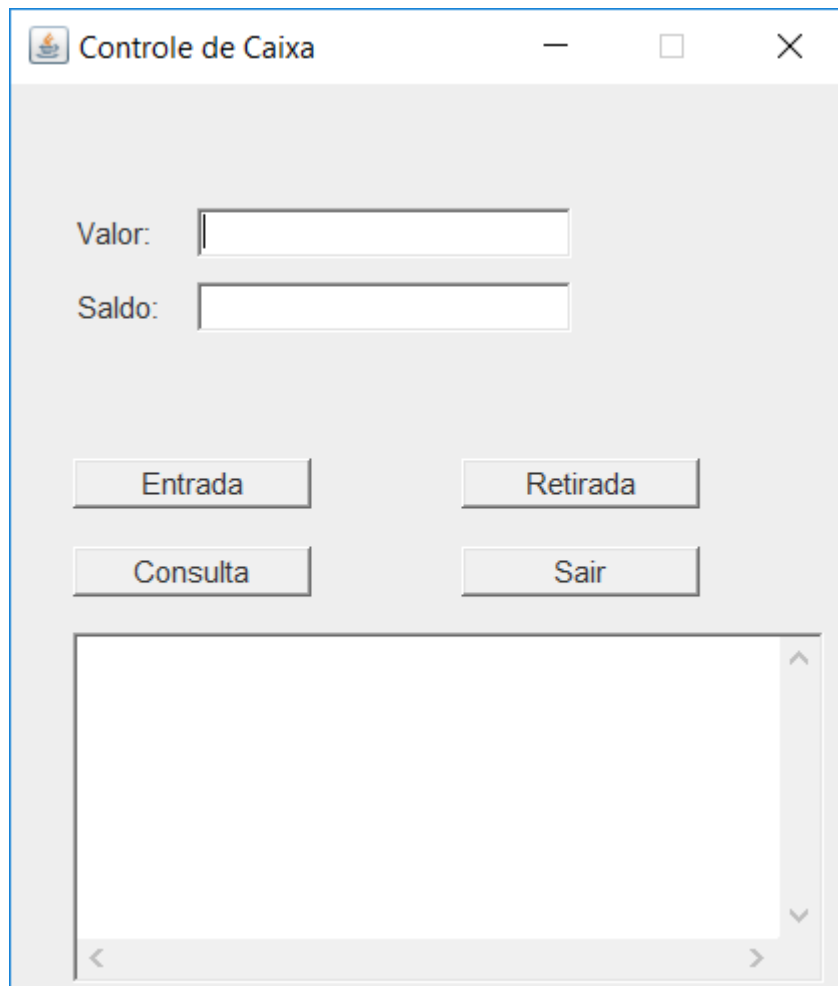
```
26      setLayout (null);
27
28      //Definir atributos dos outros componentes da Janela
29      lblValor = new Label ( text: "Valor: ");
30      lblValor.setSize (dLabel);
31      lblValor.setLocation ( x: 25, y: 50);
32      add (lblValor);
33      lblSaldo = new Label ( text: "Saldo: ");
34      lblSaldo.setSize (dLabel);
35      lblSaldo.setLocation ( x: 25, y: 80);
36      add (lblSaldo);
37      txtValor = new TextField (null);
38      txtValor.setSize (dTextField);
39      txtValor.setLocation ( x: 75, y: 50);
40      add (txtValor);
41      txtSaldo = new TextField (null);
42      txtSaldo.setSize (dTextField);
43      txtSaldo.setLocation ( x: 75, y: 80);
44      add (txtSaldo);
45      cmdEntrada = new Button ( label: "Entrada");
46      cmdEntrada.setSize (dButton);
47      cmdEntrada.setLocation ( x: 25, y: 150);
48      add (cmdEntrada);
49      cmdConsulta = new Button ( label: "Consulta");
50      cmdConsulta.setSize (dButton);
51      cmdConsulta.setLocation ( x: 25, y: 185);
52      add (cmdConsulta);
```

Classe
TelaMovimento
Designer


```
52 add(cmdConsulta);
53 cmdRetirada = new Button( label: "Retirada");
54 cmdRetirada.setSize(dButton);
55 cmdRetirada.setLocation( x: 180, y: 150);
56 add(cmdRetirada);
57 cmdSair = new Button( label: "Sair");
58 cmdSair.setSize(dButton);
59 cmdSair.setLocation( x: 180, y: 185);
60 add(cmdSair);
61 txtMsg = new TextArea(null);
62 txtMsg.setSize(dTextArea);
63 txtMsg.setLocation( x: 25, y: 220);
64 add(txtMsg);
65 }
```

**Classe TelaMovimento
Designer**

Resultado da Aparência



A screenshot of a Windows application window titled "Controle de Caixa". The window has a standard Windows title bar with a minimize button, a maximize button (disabled), and a close button. The main content area has a light gray background. It contains two input fields: "Valor:" and "Saldo:", each followed by a text box. Below these are four buttons arranged in a 2x2 grid: "Entrada", "Retirada", "Consulta", and "Sair". At the bottom is a large, empty rectangular area with a vertical scrollbar on the right and horizontal scrollbars at the bottom, suggesting a list or table view.

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

Interfaces Listeners

WindowListener possui 7 métodos

- ▣ `windowActivated()`

- Quando o formulário é ativado.

- ▣ `windowDeactivated()`

- Quando é desativado

- ▣ `windowIconified()`

- Minimizado

- ▣ `windowDeiconified()`

- Restaurado

- ▣ `windowOpened()`

- Aberto

- ▣ `windowClosed()`

- Fechado

- ▣ `windowClosing()`

- Fechando.

Interfaces Listeners

- A interface `KeyListener` possui 3 métodos, que são:
 - ▣ `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:
 - ▣ `void mouseClicked(MouseEvent e)` - Quando o botão do mouse é clicado (pressionado e liberado) em um componente.
 - ▣ `void mouseEntered(MouseEvent e)` - Quando o mouse entra na área de um componente.
 - ▣ `void mouseExited(MouseEvent e)` - Quando o mouse sai da área de um componente.
 - ▣ `void mousePressed(MouseEvent e)` - Quando o botão do botão é pressionado sobre um componente.
 - ▣ `void 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

```
4  import java.awt.*;
5  import java.awt.event.WindowEvent;
6  import java.awt.event.WindowListener;
7
8  public class TelaMovimento extends JFrame implements WindowListener {
9      protected Dimension dFrame, dLabel, dTextField, dButton, dTextArea;
10     protected Button cmdEntrada, cmdRetirada, cmdConsulta, cmdSair;
11     protected TextField txtValor, txtSaldo;
```

Classe: TelaMovimento

```
66         add(txtMsg);
67         addWindowListener( this );
68     }
69
70     @Override
71     public void windowOpened(WindowEvent e) {}
72
73
74
75     @Override
76     public void windowClosing(WindowEvent e) {}
77
78
79
80     @Override
81     public void windowClosed(WindowEvent e) {}
82
83
84
85     @Override
86     public void windowIconified(WindowEvent e) {}
87
88
89
90     @Override
91     public void windowDeiconified(WindowEvent e) {}
92
93
94
95     @Override
96     public void windowActivated(WindowEvent e) {}
97
98
99
100    @Override
101    public void windowDeactivated(WindowEvent e) {}
```


Classe: TelaMovimento

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

```
3  import javax.swing.*;
4  import java.awt.*;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import java.awt.event.WindowEvent;
8  import java.awt.event.WindowListener;
9
10 public class TelaMovimento extends JFrame implements WindowListener, ActionListener {
11     protected Dimension dFrame, dLabel, dTextField, dButton, dTextArea;
12     protected Button cmdEntrada, cmdRetirada, cmdConsulta, cmdSair;
13     protected TextField txtValor, txtSaldo;
```

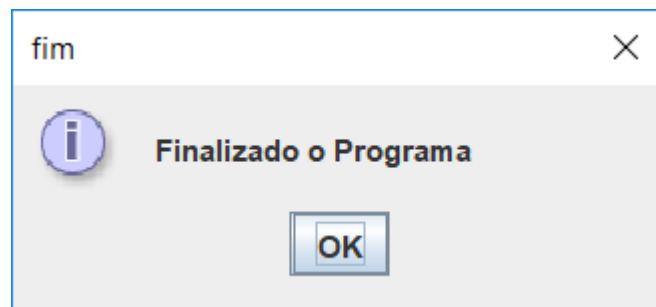
Classe: TelaMovimento – Botão Sair

```
62 cmdSair.setSize(dButton);
63 cmdSair.setLocation(x: 180, y: 185);
64 cmdSair.addActionListener(this);
65 add(cmdSair);
```

```
108
109 
110
111
112
113
114
115
```

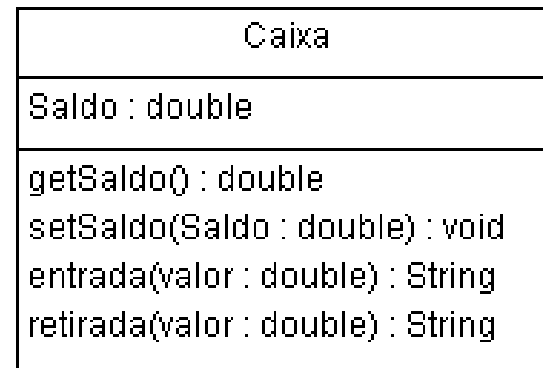
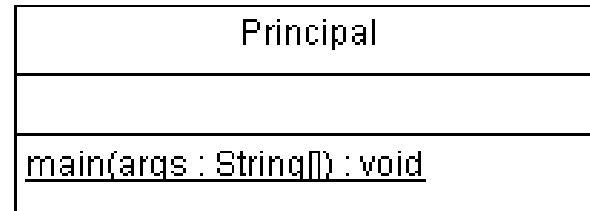
```
@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == cmdSair) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Finalizado o Programa",
            title: "fim", JOptionPane.INFORMATION_MESSAGE);
        System.exit(status: 0);
    }
}
```

Resultado do Clique no Botão Sair



Exemplo

ControleCaixa



Classe Caixa

```
1 package com.faculdade;
2
3 public class Caixa {
4     private double saldo;
5
6     public double getSaldo() {
7         return saldo;
8     }
9
10    //Método para o depósito
11    public String sacar(double valor) {
12        String msg = "Erro ao executar a operação";
13        try {
14            if((valor==0) || (valor<0)) {
15                throw new IllegalArgumentException("Valor inválido");
16            }
17            if(valor > getSaldo()) {
18                msg = "Saldo insuficiente para saque";
19            } else {
20                this.saldo -= valor;
21                msg = "Saque efetuado com sucesso";
22            }
23        } catch (IllegalArgumentException e) {
24            throw e;
25        } finally {
26            return msg;
27        }
28    }
29 }
```

Classe Caixa

```
29
30 public String depositar(double valor){
31     String msg = "Erro ao executar a operação";
32     try {
33         if((valor==0) || (valor<0)){
34             throw new IllegalArgumentException("Valor inválido");
35         }
36         this.saldo+= valor;
37         msg = "Depósito efetuado com sucesso";
38     }catch (IllegalArgumentException e){
39         throw e;
40     }finally {
41         return msg;
42     }
43
44 }
45 }
```

Classe TelaMovimento

```
10 public class TelaMovimento extends JFrame implements WindowListener, ActionListen
11     protected Dimension dFrame, dLabel, dTextField, dButton, dTextArea;
12     protected Button cmdEntrada, cmdRetirada, cmdConsulta, cmdSair;
13     protected TextField txtValor, txtSaldo;
14     protected Label lblValor, lblSaldo;
15     protected TextArea txtMsg;
16     public Caixa caixa;
17     public TelaMovimento(){
18         caixa = new Caixa();
19         //Dimensão dos componentes
20         dFrame = new Dimension( width: 350, height: 400);
```

Classe TelaMovimento

```
113 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142

public void actionPerformed(ActionEvent e) {
    if(e.getSource()==cmdSair){
        JOptionPane.showMessageDialog( parentComponent: null, message: "Finalizado o Programa",
            title: "fim", JOptionPane.INFORMATION_MESSAGE);
        System.exit( status: 0);
    }
    if(e.getSource()==cmdEntrada){
        try {
            txtMsg.append(caixa.depositar(Double.parseDouble(txtValor.getText())));
            txtMsg.append("\n");
            txtValor.setText(null);
        } catch (Exception ex){
            JOptionPane.showMessageDialog( parentComponent: null, ex.getMessage(),
                title: "Aconteceu um erro", JOptionPane.ERROR_MESSAGE);
        }
    }
    if(e.getSource()==cmdRetirada){
        try {
            txtMsg.append(caixa.sacar(Double.parseDouble(txtValor.getText())));
            txtMsg.append("\n");
            txtValor.setText(null);
        } catch (Exception ex){
            JOptionPane.showMessageDialog( parentComponent: null, ex.getMessage(),
                title: "Aconteceu um erro", JOptionPane.ERROR_MESSAGE);
        }
    }
    if(e.getSource()==cmdConsulta){
        txtSaldo.setText(Double.toString(caixa.getSaldo()));
    }
}
```

Classe Principal

```
1 package com.faculdade;
2
3 public class Principal {
4
5     public static void main(String[] args) {
6         // write your code here
7         TelaMovimento tela = new TelaMovimento();
8         tela.setVisible(true);
9     }
10 }
11
```

Resultado

Controle de Caixa

Valor:

Saldo:

Depósito efetuado com sucesso
Saque efetuado com sucesso

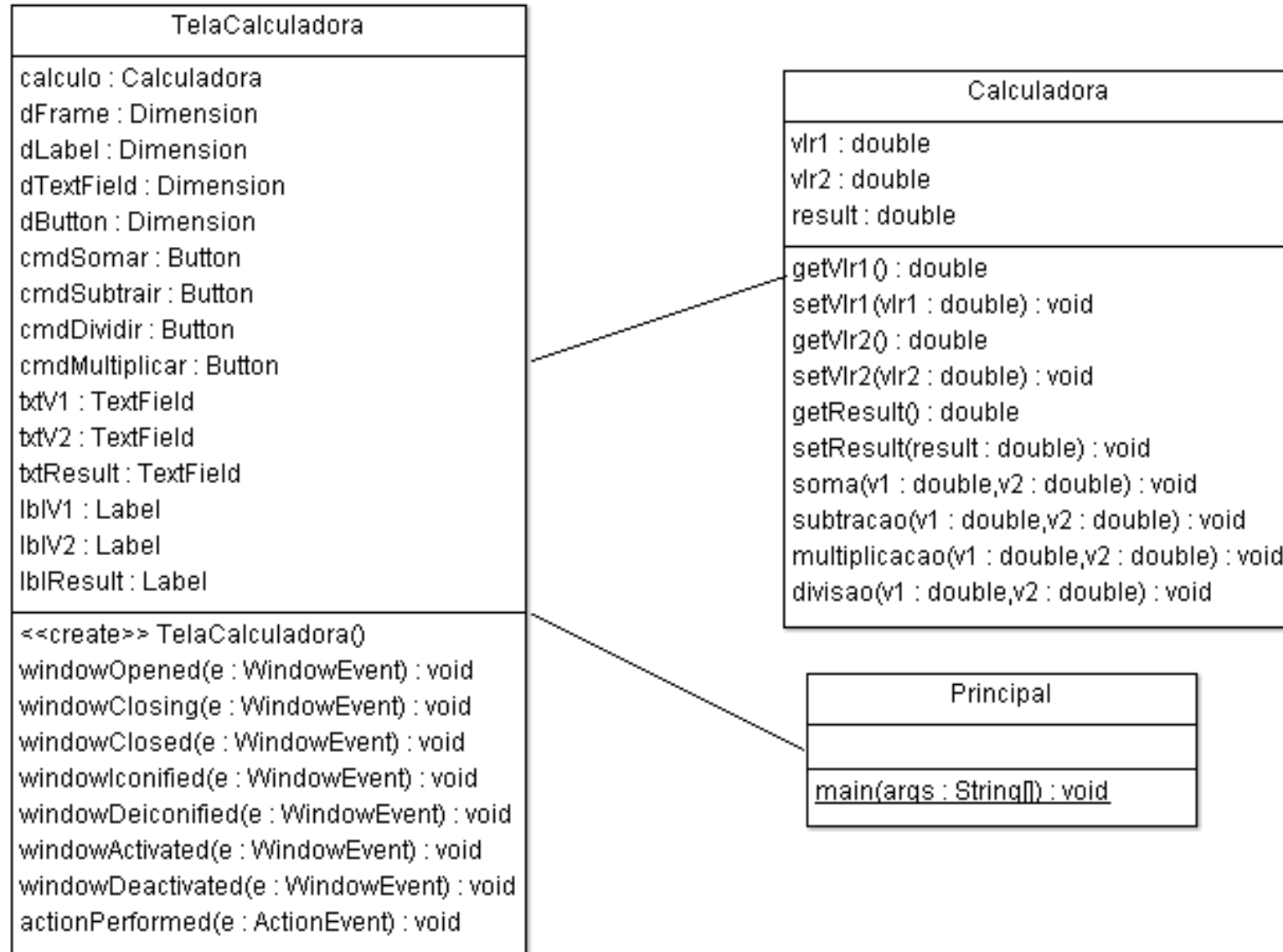
Exercício: Calculadora

□ Sua missão:

Objetivo: uso de GUI

- ▣ Criar um programa JAVA utilizando-se dos operadores básicos para criar uma calculadora com as 04 operações básicas.

Exercício: Calculadora



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.

Resultado

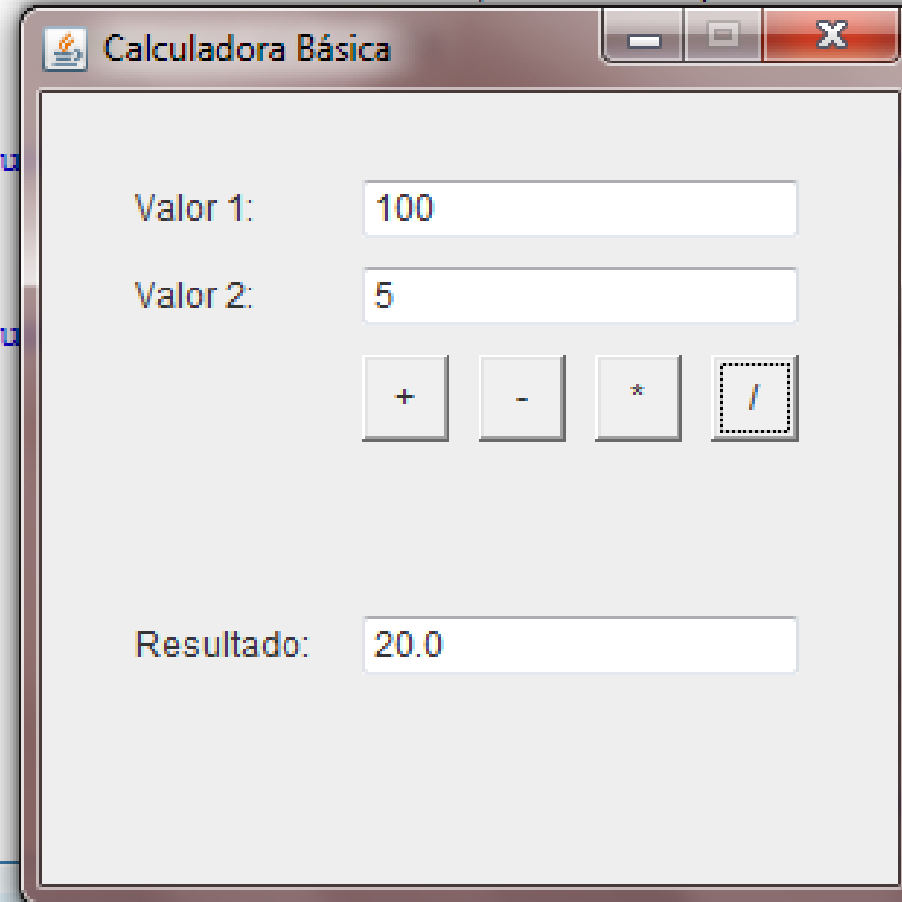
```
public void subtracao(double v1, double v2){
```

```
}
```

```
pu
```

```
}
```

```
pu
```



A screenshot of a Java Swing window titled "Calculadora Básica". The window has a standard Mac OS X-style title bar with a red close button, a yellow maximize button, and a green minimize button. Inside the window, there are two input fields labeled "Valor 1:" and "Valor 2:". The "Valor 1:" field contains the number "100" and the "Valor 2:" field contains the number "5". Below these fields are four buttons representing arithmetic operations: "+", "-", "*", and "/". The "/" button is highlighted with a dashed border. At the bottom of the window, there is a label "Resultado:" followed by a text field containing the value "20.0".

```
le v2){
```

```
{
```

```
on();
```

```
l, "Aconte
```

Aprendendo mais...

- ❑ Link com artigo apresentando exemplos práticos usando JavaFX.
- ❑ [http://code.makery.ch/library/javafx-8-tutorial/pt/part1 /](http://code.makery.ch/library/javafx-8-tutorial/pt/part1/)

Referências

□ Bibliográficas:

- ▣ Mendes – Java com Ênfase em Orientação a Objetos [Exercícios do Capítulo 1]
- ▣ Deitel – Java, como programar – 6º edição.
- ▣ Arnold, Gosling, Holmes – A linguagem de programação Java – 4º edição.
- ▣ Apostilas Caelum
- ▣ Material do Curso de Capacitação Java do CPS

□ Internet

- ▣ <http://java.sun.com>
- ▣ <http://www.guj.com.br>
- ▣ <http://www.portaljava.com>

FIM

Obrigado,
Maromo