**Resolução - Quarta Lista de Exercícios - Listeners**

**1 -**

import java.io.\*;

import java.util.Scanner;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

class EditorTexto extends JFrame{

// Botões

JButton btnAbrir = new JButton("Abrir");

JButton btnSalvar = new JButton("Salvar");

JButton btnSalvarComo = new JButton("Salvar Como");

JButton btnFechar = new JButton("Fechar");

// Área de Texto com Várias Linhas

JTextArea txtArea = new JTextArea(20, 40);

//String para Nome do Arquivo

String nomeArq = new String();

//StringBuffer para Texto do Arquivo

StringBuffer txtArq = new StringBuffer();

// Modificando o Construtor

EditorTexto(){

super("Editor de Texto");

//Painel contendo os botões

final JPanel painel = new JPanel(new GridLayout(4,1));

painel.add(btnAbrir);

painel.add(btnSalvar);

painel.add(btnSalvarComo);

painel.add(btnFechar);

//Painel contendo o painel anterior

final JPanel painel2 = new JPanel(new BorderLayout());

painel2.add(painel, BorderLayout.NORTH);

//Adicionando o JPanel na Interface Visual

add(painel2, BorderLayout.WEST);

//Adicionando o JTextArea na Interface Visual

add(txtArea);

//Arrumando design na inicialização

pack();

setLocationRelativeTo(null);

setVisible(true);

//Tratamento do Fechamento (Padrão)

addWindowListener(new fechaPrograma());

setDefaultCloseOperation(DO\_NOTHING\_ON\_CLOSE);

//Tratamento do Texto no JTextArea

txtArea.setFont(new Font(Font.MONOSPACED, Font.PLAIN, 20));

//Tratamento dos Botões

//Abrir: arquivo .txt

btnAbrir.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

nomeArq = JOptionPane.showInputDialog(null, "Digite o nome do arquivo de texto",

"Abrir", JOptionPane.QUESTION\_MESSAGE);

if(nomeArq != null && nomeArq.isEmpty() == false){

nomeArq += ".txt";

//Abrindo arquivo e colocando conteúdo no JTextArea

try{

FileInputStream in = new FileInputStream(nomeArq);

Scanner scn = new Scanner(in);

//Lendo arquivo e colocando em uma stringbuffer

while(scn.hasNextLine()){

txtArq.append(scn.nextLine());

txtArq.append("\n");

}

txtArea.setText(txtArq.toString());

txtArq.delete(0, txtArq.length());

in.close();

scn.close();

}catch(FileNotFoundException file){

JOptionPane.showMessageDialog(null, "Erro ao abrir arquivo. Tente novamente.",

"Erro de Operação", JOptionPane.ERROR\_MESSAGE);

}catch(IOException io){

JOptionPane.showMessageDialog(null, "Erro ao abrir arquivo. Tente novamente.",

"Erro de Operação", JOptionPane.ERROR\_MESSAGE);

}

}

}

});

//Salvar: arquivo atual .txt

btnSalvar.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

final int opcao = new JOptionPane().showConfirmDialog(null,

"Deseja realmente salvar o arquivo?", "Salvar",

JOptionPane.YES\_NO\_OPTION);

if(opcao == 0){

try{

//Salvando arquivo atual

FileOutputStream out = new FileOutputStream(nomeArq);

out.write(txtArea.getText().getBytes());

out.close();

//Criando um diálogo de sucesso na operação

JOptionPane.showMessageDialog(null, "Arquivo salvo com sucesso.",

"Salvar", JOptionPane.INFORMATION\_MESSAGE);

}catch(FileNotFoundException file){

JOptionPane.showMessageDialog(null, "Erro ao salvar arquivo. Tente novamente.",

"Erro de Operação", JOptionPane.ERROR\_MESSAGE);

}catch(IOException io){

JOptionPane.showMessageDialog(null, "Erro ao salvar arquivo. Tente novamente.",

"Erro de Operação", JOptionPane.ERROR\_MESSAGE);

}

}

}

});

//Salvar Como: novo arquivo .txt

btnSalvarComo.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

//Backup do NomeAtual

String nomeatualArq = nomeArq;

nomeArq = JOptionPane.showInputDialog(null, "Digite o novo nome do arquivo de texto", "Salvar Como",

JOptionPane.QUESTION\_MESSAGE);

if(nomeArq != null && nomeArq.isEmpty() == false){

nomeArq += ".txt";

try{

//Salvando arquivo digitado

FileOutputStream out = new FileOutputStream(nomeArq);

out.write(txtArea.getText().getBytes());

out.close();

// Criando um diálogo de sucesso na operação

JOptionPane.showMessageDialog(null, "Arquivo salvo com sucesso.", "Salvar Como",

JOptionPane.INFORMATION\_MESSAGE);

}catch(FileNotFoundException file){

JOptionPane.showMessageDialog(null, "Erro ao salvar arquivo. Tente novamente.",

"Erro de Operação", JOptionPane.ERROR\_MESSAGE);

}catch(IOException io){

JOptionPane.showMessageDialog(null, "Erro ao salvar arquivo. Tente novamente.",

"Erro de Operação", JOptionPane.ERROR\_MESSAGE);

}

}

else{

nomeArq = nomeatualArq;

}

}

});

//Fechar: fechar o arquivo .txt aberto

btnFechar.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

final int opcao = new JOptionPane().showConfirmDialog(null, "Deseja realmente fechar o arquivo atual?",

"Fechar", JOptionPane.YES\_NO\_OPTION);

if(opcao == 0){

//Zerando texto do JTextArea

txtArea.setText("");

}

}

});

}

//Listener para Tratamento do Fechamento

class fechaPrograma extends WindowAdapter{

//Método de Fechamento

public void windowClosing(final WindowEvent e){

final int opc = JOptionPane.showConfirmDialog(null,

"Deseja realmente fechar o programa?", "Fechar",

JOptionPane.YES\_NO\_OPTION);

if(opc == 0){

System.exit(0);

}

}

}

public static void main(final String[] args) {

new EditorTexto();

}

}

**2 -**

**Exercício 2 (GUI):**

import java.awt.event.\*;

import java.awt.\*;

import javax.swing.\*;

class Calculadora1 extends JFrame{

// Botões

JButton btnSoma = new JButton("Soma");

JButton btnSubtrai = new JButton("Subtrai");

JButton btnMultiplica = new JButton("Multiplica");

JButton btnDivide = new JButton("Divide");

// TextFields

JTextField txfTexto1 = new JTextField("0", 10);

JTextField txfTexto2 = new JTextField("0", 10);

// Label's

JLabel lbIgual = new JLabel(" = ");

JLabel lbResultado = new JLabel("0 ");

// Modificando o Construtor

Calculadora1() {

super("Calculadora");

setLayout(new FlowLayout());

// Criando os JPanel's

JPanel painelNum1 = new JPanel(new BorderLayout());

JPanel painelBotao = new JPanel(new GridLayout(4, 1));

JPanel painelResultado = new JPanel(new BorderLayout());

// Modificando Panel's

painelNum1.add(txfTexto1, BorderLayout.CENTER);

painelBotao.add(btnSoma);

painelBotao.add(btnSubtrai);

painelBotao.add(btnMultiplica);

painelBotao.add(btnDivide);

painelResultado.add(txfTexto2, BorderLayout.WEST);

painelResultado.add(lbIgual, BorderLayout.CENTER);

painelResultado.add(lbResultado, BorderLayout.EAST);

// Acrescentando na Interface

add(painelNum1);

add(painelBotao);

add(painelResultado);

// Automatização

pack();

setLocationRelativeTo(null); // Posição central

setVisible(true);

// Tratamento do Fechamento (Padrão)

addWindowListener(new WindowAdapter(){

public void windowClosing(final WindowEvent e) {

final int opc = JOptionPane.showConfirmDialog(null, "Deseja realmente fechar o programa?", "Fechar",

JOptionPane.YES\_NO\_OPTION);

if (opc == 0) {

System.exit(0);

}

}

});

setDefaultCloseOperation(DO\_NOTHING\_ON\_CLOSE);

//Tratando os eventos

//Soma

btnSoma.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

//Recuperar TextField's

int num1 = Integer.parseInt(txfTexto1.getText());

int num2 = Integer.parseInt(txfTexto2.getText());

//Colocar resultado na label

lbResultado.setText(Integer.toString(num1+num2));

}

});

//Subtração

btnSubtrai.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// Recuperar TextField's

int num1 = Integer.parseInt(txfTexto1.getText());

int num2 = Integer.parseInt(txfTexto2.getText());

// Colocar resultado na label

lbResultado.setText(Integer.toString(num1 - num2));

}

});

//Multiplicação

btnMultiplica.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// Recuperar TextField's

int num1 = Integer.parseInt(txfTexto1.getText());

int num2 = Integer.parseInt(txfTexto2.getText());

// Colocar resultado na label

lbResultado.setText(Integer.toString(num1 \* num2));

}

});

//Divisão

btnDivide.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// Recuperar TextField's

int num1 = Integer.parseInt(txfTexto1.getText());

int num2 = Integer.parseInt(txfTexto2.getText());

// Colocar resultado na label

try{

lbResultado.setText(Integer.toString(num1 / num2));

}catch(ArithmeticException ae){

JOptionPane.showMessageDialog(null, "Erro de operação aritmética. Tente novamente.",

"Erro de Operação", JOptionPane.ERROR\_MESSAGE);

}

}

});

}

public static void main(String[] args) {

// Executando a função

new Calculadora1();

}

}

**Exercício 3 (GUI):**

import java.awt.event.\*;

import java.awt.\*;

import javax.swing.\*;

class Calculadora2 extends JFrame {

// Campos de Texto

JTextField txfNum1 = new JTextField("0", 10);

JTextField txfNum2 = new JTextField("0", 10);

// Label's

JLabel lbIgual = new JLabel(" = ");

JLabel lbResultado = new JLabel("0 ");

// JComboBox (Seletor)

JComboBox cbxOperacao = new JComboBox();

//Botão Calcular

Button btnCalcular = new Button("Calcular");

Calculadora2() {

super("Calculadora");

// Criando os Panel's

JPanel pnlSuperior = new JPanel(new FlowLayout());

JPanel pnlNum2 = new JPanel(new BorderLayout());

// Configurando o ComboBox

cbxOperacao.addItem("Soma");

cbxOperacao.addItem("Subtrai");

cbxOperacao.addItem("Multiplica");

cbxOperacao.addItem("Divide");

// Configurando os Panel's

pnlNum2.add(txfNum2, BorderLayout.WEST);

pnlNum2.add(lbIgual, BorderLayout.CENTER);

pnlNum2.add(lbResultado, BorderLayout.EAST);

pnlSuperior.add(txfNum1);

pnlSuperior.add(cbxOperacao);

pnlSuperior.add(pnlNum2);

// Adicionando na Interface Visual

add(pnlSuperior, BorderLayout.NORTH);

add(btnCalcular, BorderLayout.SOUTH);

// Configs Finais

pack();

setLocationRelativeTo(null); //Posição central

setVisible(true);

// Tratamento do Fechamento (Padrão)

addWindowListener(new WindowAdapter() {

public void windowClosing(final WindowEvent e) {

final int opc = JOptionPane.showConfirmDialog(null, "Deseja realmente fechar o programa?", "Fechar",

JOptionPane.YES\_NO\_OPTION);

if (opc == 0) {

System.exit(0);

}

}

});

setDefaultCloseOperation(DO\_NOTHING\_ON\_CLOSE);

//Tratando os eventos

//Botão Calcular

btnCalcular.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

//Recuperando valores para calculo

int num1 = Integer.parseInt(txfNum1.getText());

int num2 = Integer.parseInt(txfNum2.getText());

//Verificando cada campo do ComboBox

if(cbxOperacao.getSelectedIndex() == 0){

//Soma

lbResultado.setText(Integer.toString(num1+num2));

}

else if(cbxOperacao.getSelectedIndex() == 1){

//Subtração

lbResultado.setText(Integer.toString(num1 - num2));

}

else if (cbxOperacao.getSelectedIndex() == 2) {

// Multiplicação

lbResultado.setText(Integer.toString(num1 \* num2));

}

else if (cbxOperacao.getSelectedIndex() == 3) {

// Divisão

try{

lbResultado.setText(Integer.toString(num1 / num2));

}catch(ArithmeticException ar){

JOptionPane.showMessageDialog(null, "Erro de operação aritmética. Tente novamente.",

"Erro de Operação", JOptionPane.ERROR\_MESSAGE);

}

}

}

});

}

public static void main(String[] args) {

new Calculadora2();

}

}

**3 -**

**ActionListener:** O “ActionListener”, segundo a Oracle, é o manipulador de eventos mais fácil e comum de implementar. Ele está relacionado a manipulação do evento de uma operação executada pelo usuário, como apertar um botão, pressionar o Enter em um campo de texto, selecionar um item de menu, dentre outros. O resultado dessa ação torna-se uma mensagem “actionPerformed” (único método dessa interface) que é enviada a todos “ActionListeners” registrado em determinado componente. Por ser a mais usual, é fortemente recomendado seu uso pela Oracle, haja visto que torna possível a manipulação da ação do usuário em um componente tanto pelo mouse quanto pelo teclado. Um exemplo pode ser indicado no código abaixo em um evento de clique de um botão e uma interface visual que mostra quantas vezes ele foi clicado.

import java.awt.\*;

import java.awt.event.\*;

public class AL extends Frame implements WindowListener,ActionListener {

TextField text = new TextField(20);

Button b;

private int numClicks = 0;

public static void main(String[] args) {

AL myWindow = new AL("My first window");

myWindow.setSize(350,100);

myWindow.setVisible(true);

}

public AL(String title) {

super(title);

setLayout(new FlowLayout());

addWindowListener(this);

b = new Button("Click me");

add(b);

add(text);

b.addActionListener(this);

}

public void actionPerformed(ActionEvent e) {

numClicks++;

text.setText("Button Clicked " + numClicks + " times");

}

public void windowClosing(WindowEvent e) {

dispose();

System.exit(0);

}

public void windowOpened(WindowEvent e) {}

public void windowActivated(WindowEvent e) {}

public void windowIconified(WindowEvent e) {}

public void windowDeiconified(WindowEvent e) {}

public void windowDeactivated(WindowEvent e) {}

public void windowClosed(WindowEvent e) {}

}

**KeyListener:** O “KeyListener” refere-se ao tratamento de eventos do teclado. Ou seja, os “eventos do teclado” são disparados por um determinado componente a medida que o usuário pressiona ou solta as teclas do teclado. Segundo a Oracle, as notificações enviadas por esse tratamento de teclas são duas: a digitação do caractere Unicode; e o pressionar ou soltar de uma tecla do teclado. A utilização do “KeyListener” deve ser feita para uma manipulação inteiramente pelo teclado, como utilizar a tecla Enter para atalho em um botão após o preenchimento de um formulário (como dados de login e senha, por exemplo). Um exemplo pode ser visto no código abaixo na identificação (via console) de uma tecla pressionada.

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import javax.swing.\*;

class Teste extends JFrame {

public Teste() {

addKeyListener(new KeyListener() {

public void keyTyped(KeyEvent e) {

System.out.println("Tecla: " + e.getKeyChar());

}

@Override

public void keyReleased(KeyEvent e) {

}

@Override

public void keyPressed(KeyEvent e) {

}

});

setSize(300, 300);

setVisible(true);

}

public static void main(String[] args) {

new Teste();

}

}

**MouseListener:** O “MouseListener” refere-se ao tratamento dos eventos do mouse quando o usuário utiliza esse tipo de dispositivo de entrada para interagir com um componente. Esses eventos ocorrem em duas situações: quando o cursor entra ou sai da área de tela de um componente e quando o usuário pressiona ou solta um dos botões do mouse. Sendo assim, a importância de se utilizar o “MouseListener” se encontra em maior utilidade na programação de jogos e aplicações onde é necessário verificar as posições do mouse, bem como suas ações de clicar e soltar. Um exemplo de código testando as funcionalidades dessa interface pode ser visto abaixo.

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.Toolkit;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.event.MouseMotionListener;

import javax.swing.JFrame;

import javax.swing.JLabel;

class MouseTracker extends JFrame implements MouseListener, MouseMotionListener {

JLabel mousePosition = new JLabel();

MouseTracker(){

addMouseListener(this);

addMouseMotionListener(this);

setLayout(new FlowLayout());

add(mousePosition);

Dimension size = Toolkit.getDefaultToolkit().getScreenSize();

setSize(size);

setVisible(true);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

@Override

public void mouseClicked(MouseEvent e) {

mousePosition.setText("Mouse clicado na coordenada : [" + e.getX() + "," + e.getY() + "]");

}

@Override

public void mouseEntered(MouseEvent e) {

mousePosition.setText("Coordenada atual do mouse : [" + e.getX() + "," + e.getY() + "]");

}

@Override

public void mouseExited(MouseEvent e) {

mousePosition.setText("O Mouse está fora da janela de acesso");

}

@Override

public void mousePressed(MouseEvent e) {

mousePosition.setText("Mouse pressionado nas coordenadas : [" + e.getX() + "," + e.getY() + "]");

}

@Override

public void mouseReleased(MouseEvent e) {

mousePosition.setText("Coordenada atual do mouse : [" + e.getX() + "," + e.getY() + "]");

}

@Override

public void mouseDragged(MouseEvent e) {

mousePosition.setText("Mouse arrastado nas coordenadas : [" + e.getX() + "," + e.getY() + "]");

}

@Override

public void mouseMoved(MouseEvent e) {

mousePosition.setText("Mouse movido para as coordenadas : [" + e.getX() + "," + e.getY() + "]");

}

public static void main(String args[]) {

new MouseTracker();

}

}

Fonte principal: Documentação Oficial do Java 8 da Oracle (https://docs.oracle.com/javase/tutorial/uiswing/events/api.html).

**4 –** O “**Listener”** são interfaces que contém todos os métodos para tratamento de determinados erros. Ou seja, ao implementá-la em uma classe, é necessário definir **todos** os métodos dessa interface. Por exemplo: ao querer manipular o evento de clique do mouse, existe uma interface denominada MouseListener com o método de “mouseClicked”. Contudo, é necessário declarar todos os outros quatro métodos para que se possa utilizar a interface e o método de clique do mouse. O ponto positivo ao utilizar os “Listener’s” consiste em poder utilizar mais de um numa mesma classe de tratamento, haja visto que são interfaces. Já o ponto negativo consiste na declaração de todos os métodos daquela classe.

Já o “**Adapter”** se refere a classes que implementam os métodos de “Listener”, podendo o programador utilizar apenas o método desejado. Praticamente, para cada “Listener” existe uma classe “Adapter”. Utilizando o exemplo anterior do MouseListener: para tratar o evento de clique do mouse, basta **estender** a classe MouseAdapter e simplesmente utilizar o método “mouseClicked”, não precisando declarar todos os outros quatro métodos presentes em MouseListener. Sendo assim, a principal vantagem em se utilizar um “Adapter” consiste em declarar e manipular apenas o método de evento que lhe é desejado. Contudo, um ponto negativo está no conceito de herança: não se pode herdar de mais de uma classe “Adapter” numa mesma classe, sendo necessário a criação de várias classes para diferentes tratamentos de eventos.

Ou seja: deve-se utilizar “Listener” quando precisar editar ou utilizar-se de todos os métodos contido nela. Do contrário, por exemplo na utilização de apenas um método, deve-se utilizar “Adapter”.