Exemplo de uma calculadora simples em Java (modo easy)

```
package exemplo2;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;


/*author NIVIAMARIADOMINGUES*/
public class Exemplo2 extends JFrame implements ActionListener{
 JLabel lblnum, lblnum1, lblresult;
 JTextField txtnum, txtnum1, txtresult;
 JButton btnsoma, btnsub, btnmult, btndiv, btnlimpar;

    public static void main(String[] args) {
        JFrame janela = new Exemplo2();
        janela.setUndecorated(true);
        janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
```

```
Exemplo2()
{
    setTitle("Calculadora");
    setBounds(300,50,350,90);
    getContentPane().setBackground(new Color(135,206,235));
    getContentPane().setLayout(new GridLayout(3,4));

    /*instanciando as JLabels*/
    lblnum = new JLabel("Num 1");
    lblnum.setForeground(Color.black);
    lblnum.setFont(new Font(" ",Font.BOLD, 14));
    lblnum1 = new JLabel("Num 2");
    lblnum1.setForeground(Color.black);
    lblnum1.setFont(new Font(" ",Font.BOLD, 14));
    lblresult = new JLabel("Total");
    lblresult.setForeground(Color.black);
    lblresult.setFont(new Font(" ",Font.BOLD, 14));

    /*instanciando os JButtons*/
    btnsoma = new JButton("+");          btnsoma.addActionListener(this);
    btnsub = new JButton("-");           btnsub.addActionListener(this);
    btnmult = new JButton("*");          btnmult.addActionListener(this);
    btndiv = new JButton("/");           btndiv.addActionListener(this);
    btnlimpar = new JButton("Limpar");   btnlimpar.addActionListener(this);

    /*instanciando as JTestField*/
    txtnum = new JTextField();
    txtnum1 = new JTextField();
    txtresult = new JTextField();
    txtresult.setEditable(false);
```

```
        /*adicioando os objetos na JFrame*/
        getContentPane().add(lblnum);    getContentPane().add(txtnum);
        getContentPane().add(btnsoma);    getContentPane().add(btnsub);
        getContentPane().add(lblnuml);   getContentPane().add(txtnuml);
        getContentPane().add(btnmult);    getContentPane().add(btndiv);
        getContentPane().add(lblresult);    getContentPane().add(txtresult);
        getContentPane().add(btnlimpar);

    }

    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource()==btnlimpar)
        {
            txtnum.setText("");
            txtnuml.setText("");
            txtresult.setText("");
            return;
        }
        double num=0, numl=0, result=0;
        try
        {
            num = Double.parseDouble(txtnum.getText());
            numl = Double.parseDouble(txtnuml.getText());
        }
        catch(NumberFormatException erro)
        {
           txtresult.setText("Só números");
           return;
        }
        if(e.getSource()==btnsoma) result = num+numl;
        if(e.getSource()==btnsub) result = num-numl;
        if(e.getSource()==btnmult) result = num*numl;
        if(e.getSource()==btndiv) result = num/numl;
        txtresult.setText(" " + result );
    }
}
```

Onde:

`implements ActionListener` é a implementação de uma interface, ou seja, essa interface é responsável por interpretar as ações do usuário com a janela. Existem várias interfaces diferentes e cada interface possui seus métodos que, obrigatoriamente devem ser inseridos na aplicação.

```
JLabel lblnum, lblnuml, lblresult;
JTextField txtnum, txtnuml, txtresult;
JButton btnsoma, btnsub, btnmult, btndiv, btnlimpar;
```

Criação de todos os objetos que serão utilizados na aplicação. Eles serão instanciados no construtor.

```
public static void main(String[] args) {
    JFrame janela = new Exemplo2();
    janela.setUndecorated(true);
    janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
    janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    janela.setVisible(true);
}
```

Parte executável, onde:

```
JFrame janela = new Exemplo2();
```

criação do objeto janela e sua instanciação.

```
janela.setUndecorated(true);
```

retira a decoração padrão da janela

```
janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
```

definição de um novo estilo para a janela, no caso, FRAME. Esse estilo corresponde ao padrão JVM. Essa constante gera uma janela no formato padrão com os botões de minimizar, maximizar e encerrar. Outras variações podem ser alcançadas pelas constantes: NONE, COLOR_CHOOSER_DIALOG, ERROR_DIALOG, FILE_CHOOSER_DIALOG, INFORMATION_DIALOG, PLAIN_DIALOG, QUESTION_DIALOG e WAENING_DIALOG.

```
janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Responsável por sair da aplicação quando a janela for fechada. A constante usada foi EXIT_ON_CLOSE. Outras constantes podem ser utilizadas: HIDE_ON_CLOSE (oculta a janela quando fechada), DO_NOTHING_ON_CLOSE (não faz nada, apenas desativa o botão de encerramento) e DISPOSE_ON_CLOSE (a janela desaparece).

```
janela.setVisible(true);
```

torna a janela visível

```
setTitle("Calculadora");
setBounds(300,50,350,90);
```

Definição do título da janela e do local onde irá aparecer a janela e seu tamanho.

```
getContentPane().setBackground(new Color(135,206,235));
```

Definição da cor de fundo da janela em modo RGB.

```
getContentPane().setLayout(new GridLayout(3,4));
```

O método setLayout é responsável por controlar a maneira em que os componentes serão inseridos na janela. A parte de dentro da janela recebe o nome de contêiner e se inserimos um componente sem ter um gerenciador, o componente irá ocupar todo o container.

Existem vários gerenciadores de layout, aqui estamos usando o GridLayout. Ele divide o contêiner em um conjunto de células espalhadas numa grade retangular,

de maneira que todas as células possuam a mesma dimensão. Sua dimensão é definida no momento da sua criação (quantidade de linhas x quantidade de colunas).

```
lblnum = new JLabel("Num 1");
lblnum.setForeground(Color.black);
lblnum.setFont(new Font(" ",Font.BOLD, 14));
```

Aqui, foi instanciado e definido o texto da JLabel, o método setForeground definiu a cor da letra e o método setFont o tipo da fonte, negrito e tamanho 14.

```
/*instanciando os JButtons*/
btnsoma = new JButton("+");          btnsoma.addActionListener(this);
btnsub = new JButton("-");           btnsub.addActionListener(this);
btnmult = new JButton("*");          btnmult.addActionListener(this);
btndiv = new JButton("/");           btndiv.addActionListener(this);
btnlimpar = new JButton("Limpar");   btnlimpar.addActionListener(this);
```

Aqui foi instanciado os JButtons e definido o seu texto e principalmente faz o registro do botão que irá receber a ação com o método addActionListener.

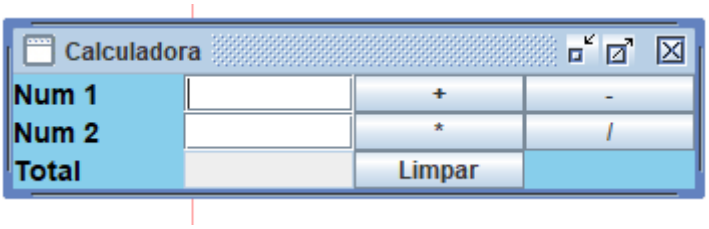
```
public void actionPerformed(ActionEvent e)
{
```

Aqui, temos o método que recebe todas as ações disparadas na janela, ou seja, todos os eventos gerados a partir da interação com o usuário com a aplicação. Esse método é obrigatório quando JFrame implementa a interface ActionListener. Dentro dos parênteses temos a declaração do objeto e da classe ActionEvent que recebe todos os eventos da janela e está classe é responsável por identificar o objeto que gerou o evento.

```
if (e.getSource()==btnlimpar)
{
```

A declaração acima verifica se o objeto que gerou o evento é o btnlimpar. Desta forma é reconhecido o objeto capturando a fonte do evento por meio do objeto do ActionEvent, no caso, o objeto e.

Tela da aplicação em execução



Observações: Sempre crie o layout ante de implementar os eventos.

Tratamento de exceções

Uso do try...catch

A estrutura try... catch

Esta estrutura tem como função desviar a execução de um programa caso ocorram certos tipos de erro, predefinidos durante o processamento das linhas, e evitar que o programador precise fazer testes de verificação e avaliação antes da realização de certas operações. Quando um erro ocorre, ele gera uma execução que pode ser tratada pelo programa.

Existem muitas exceções que podem ser geradas pelas mais diversas classes e enumerá-las seria um processo cansativo.

A estrutura do try..catch possui a seguinte sintaxe:

```
try
{
  instruções
}
catch (nome da exceção)
{
  tratamento do erro
}
```

Toda vez que o try for utilizado, obrigatoriamente após seu encerramento temos que ter pelo menos um catch.

O try tenta executar as instruções que foram definidas dentro dele, se isso não for possível a execução será desviada para o catch.