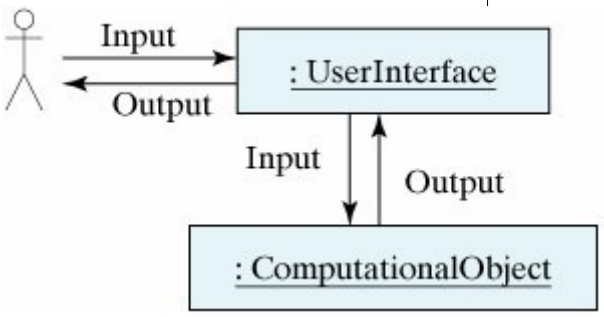
Programação Orientada a Objetos

Alexandre César Muniz de Oliveira

Entrada e Saída

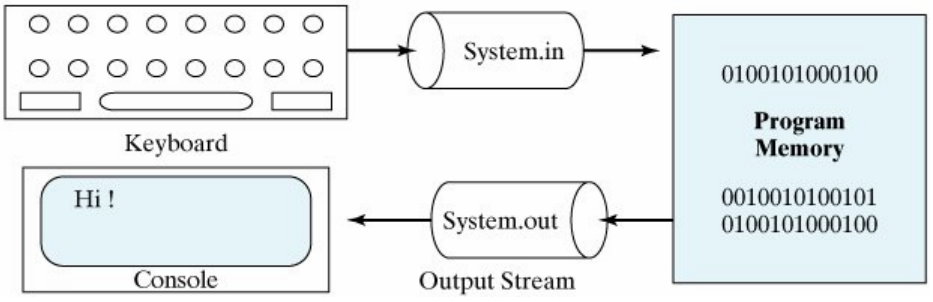
Parte IV

Entrada e Saída



- *Módulo de Interface com o usuário*
  - *Dividir trabalho*
  - *Separar processamento de E/S*

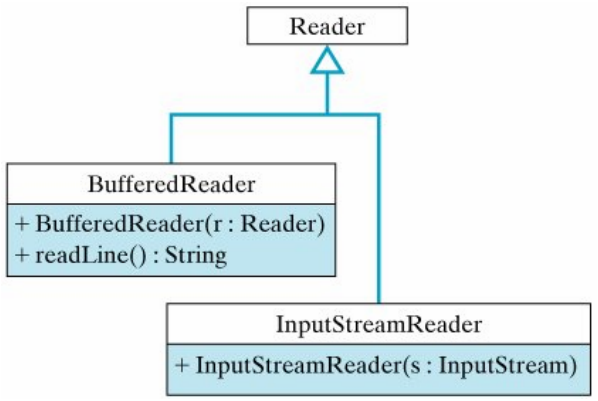
Entrada e Saída



- Em Java, E/S é tratada por objetos ***streams***.

Entrada e Saída

BufferedReader x = new BufferedReader (new InputStreamReader (System.in));



- http://java.sun.com/j2se/1.4.2/docs/api/java/io/package-tree.html

class java.lang.Object

* class java.io.File (implements java.lang.Comparable, java.io.Serializable)

(...)

*** class java.io.Reader**

o class java.io.BufferedReader

o class java.io.CharArrayReader

o class java.io.FilterReader

o class java.io.InputStreamReader

o class java.io.PipedReader

o class java.io.StringReader

- class java.io.StreamTokenizer (...)

Entrada e Saída

- Lendo números (Class BufferedReader)

```
String inputString = new String();
System.out.println("Quanto você correu? ");
inputString = input.readLine();
double miles = Double.parseDouble(inputString);
System.out.println("Quanto tempo? ");
inputString = input.readLine();
double minutes = Double.parseDouble(inStringg);
System.out.println("Seu pique foi de " +
   miles/minutes + "km/h");
```

Entrada e Saída

```
import java.io.*;
public class KeyboardReader
   {
   private BufferedReader reader;
   public KeyboardReader() {
      reader = new BufferedReader (new
   InputStreamReader(System.in));
   }
   public String getKeyboardInput()
   {
         return readKeyboard();
   } ...
```

Entrada e Saída

```
public int getKeyboardInteger()
  {
        return Integer.parseInt(readKeyboard());
  }
  public double getKeyboardDouble()
  {
        return Double.parseDouble(readKeyboard());
  }
  public void prompt(String s)
  {
        System.out.print(s);
  }
  public void display(String s)
  {
        System.out.print(s);
  } ...
```

Entrada e Saída

```
...
 private String readKeyboard()
   {
        String line = "";
        try
        {
          line = reader.readLine(); }
          catch (IOException e)  {  e.printStackTrace();    }
         return line;
      }
  }
```

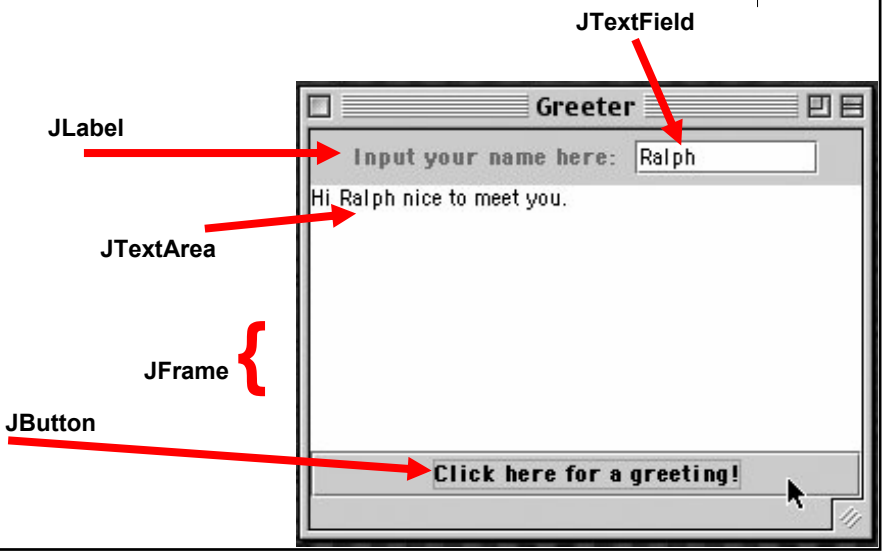
Interface gráfica

- Componentes de Interface Gráfica (GUI)
  - Um componente possui atributos e métodos
  - Definem um comportamento conhecido
  - Podem ser conectados formando componentes de alto-nível, formando a interface com o usuário
- Pacotes separados, mas interrelacionados
  - java.awt e
  - javax.swing
  - Componente java.awt.Button foi substituído pelo javax.swing.JButton

Interface gráfica

- Exemplos de componentes
  - **JLabel** é simplesmente uma string de texto mostrado na GUI
  - **JTextField** é um elemento de entrada que armazena uma única linha de texto.
  - **JTextArea** é um componente de saída que mostra múltiplas linhas de texto
  - **JButton** é um elemento de controle rotulado que permite a interação do usuário com o programa
  - **JFrame** é um container de alto-nível que pode conter outros componentes

Interface gráfica



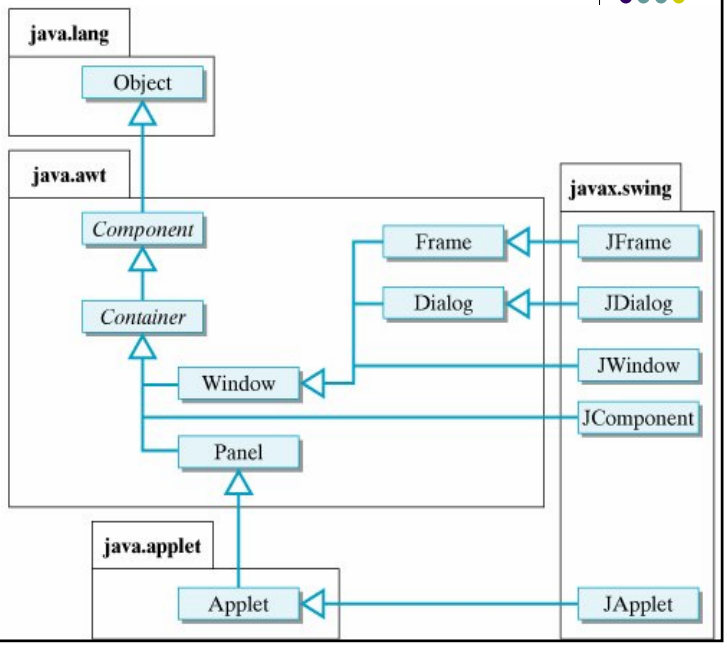
Interface gráfica

- Classes Swing
  - Superiores às suas similares do AWT.
  - Projeto orientado a objetos sofisticado
  - Arquitetura *model-view-controller* (MVC)
  - Exemplo:
    - AWT Button = “string”
    - Swing JButton = Image.bmp

Interface gráfica

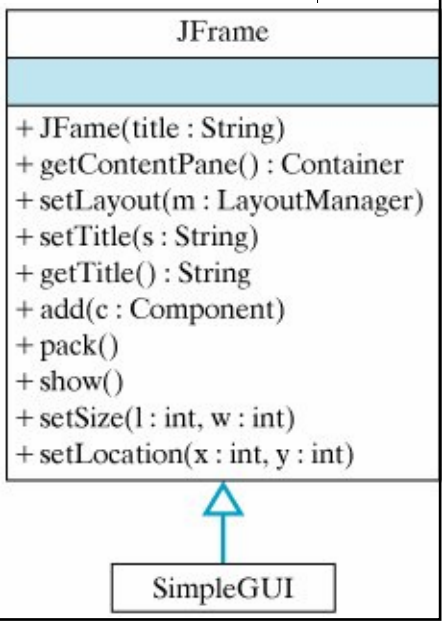
- Classes Swing
  - Componentes escritos em Java
  - Portabilidade
  - Aparência independe de S.O.
  - Diferentes *look-and-feel*
  - Utilização de componentes é feito por herança

Interface gráfica



Interface gráfica

- Subclasse de JFrame especializa sua funcionalidade para uma aplicação específica

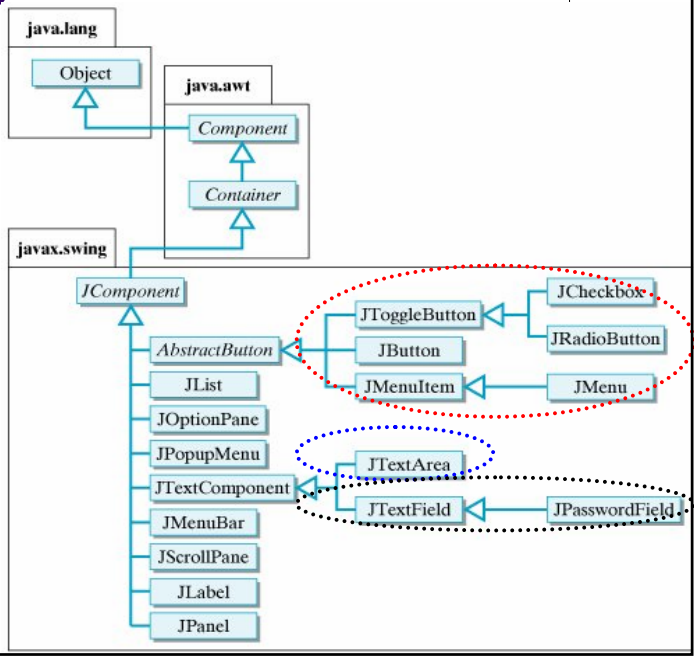


Interface gráfica

```
import javax.swing.*;
public class SimpleGUI extends JFrame
{   public SimpleGUI(String title)
   {  setSize(200,150);
      setLocation(100, 150);
      setTitle(title);
      setVisible(true);                  // Displays the JFrame
   }
public static void main(String args[])
{  new SimpleGUI("My GUI");  }
}
```

Interface gráfica

- Controle
- Saída
- Entrada



Interface gráfica

- Exemplo:
  - *JTextField* aceita entrada do usuário
  - *JTextArea* mostra a saída do programa
  - *JButton* permite uma ação de controle
  - *JLabel* aviso para o *JTextField*.

Interface gráfica

```
private JLabel prompt;
private JTextField inField;
private JTextArea display;
private JButton goButton;

prompt = new JLabel("Por favor, digite seu nome aqui: ");
inField = new JTextField(10);                    // tamanho de 10 chars
display = new JTextArea(10, 30);                 // 10 lin x 30 col
goButton = new JButton("Clique para saudação");
```

Interface gráfica

- Outros métodos

JButton
+ JButton(text : String) + addActionListener(al : ActionListener) + setEnabled(b : boolean)

JLabel
+ JLabel(text : String)

JTextField
+ JTextField(col : int) + JTextField(text : String) + getText() : String + addActionListener(al : ActionListener) + setEnabled(b : boolean)

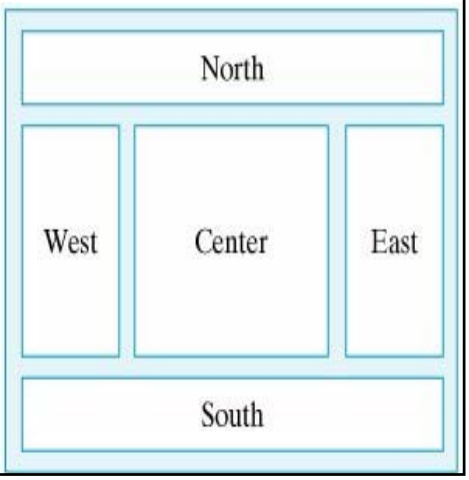
JTextArea
+ JTextArea(row : int, col : int) + append(text : String) + setText(text : String)

Interface gráfica

- Componentes adicionados a um *JFrame* através de *content panes*.

add(Component comp)
add(Component comp, int index)
*add(String **region**, Component comp)*

  - "North", "South",
  - "East", "West", or
  - "Center".



Interface gráfica

- Apenas um componente por área

Container contentPane = getContentPane();
contentPane.add("Center",display);

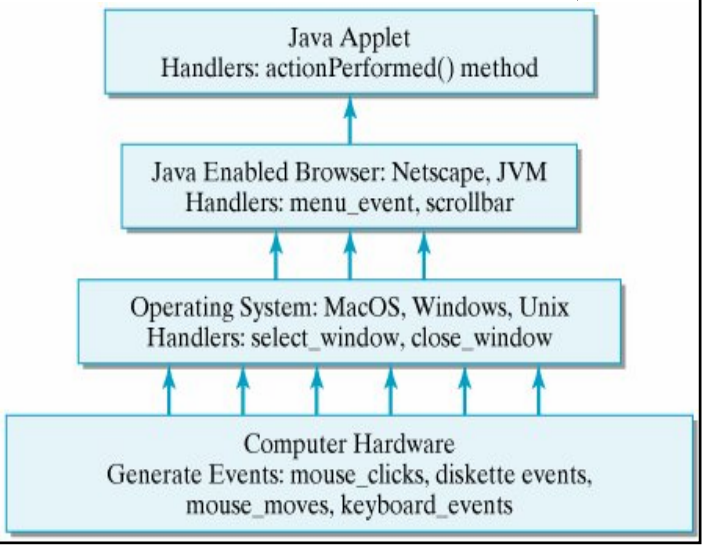
- *JPanels* podem ser usados para encapsular componentes relacionados

Interface gráfica

```
prompt = new JLabel("Por favor, digite seu nome aqui: ");
inField = new JTextField(10);
display = new JTextArea(10, 30);
goButton = new JButton("Clique para saudação");
JPanel inputPanel = new JPanel();

inputPanel.add(prompt);
inputPanel.add(inField);
inputPanel.add(goButton);

Container contentPane = getContentPane();
contentPane.add("South", inputPanel);
```

Eventos

- Programação orientada a eventos
- Eventos são gerados pelo hardware
- Componentes chamados *listeners*.
- Um "ouvinte" monitora eventos

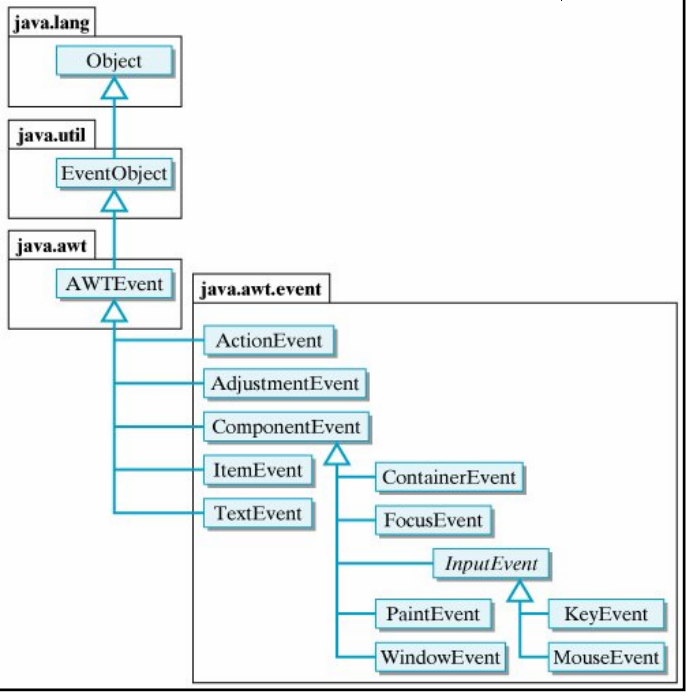


Eventos

- Programação orientada a eventos
  - Laço de evento
  - Ouvindo eventos
  - Tomada uma ação

Repeat forever or until the program is stopped
Listen for events
if event-A occurs, handle it with event-A-handler
if event-B occurs, handle it with event-B-handler

Eventos



Eventos

- Exemplos:
  - Clique do mouse
  - Acionamento de tecla
  - Foco do mouse
- Java cria um objeto *ActionEvent* contendo
  - instante
  - tipo de evento
- O usuário pode tratar (dar ações) os eventos que desejar
- Para tanto Java opera com o conceito de *Interface*

Interface para eventos

- Interface
  - Tipo de classe especial em Java
  - Contém apenas métodos e constantes (variáveis do tipo final)
  - Não podem conter variáveis de instância
- Uma classe define um comportamento
- Interface
  - Define apenas o “protocolo de comportamento” a ser implementado
  - Obriga a implementação de um protocolo ou conjunto de métodos

Interface para eventos

- Interface *ActionListener*

```
public abstract interface ActionListener extends
   EventListener
{   public abstract void actionPerformed(ActionEvent e); }
```

- A classe interessada em processar um evento deve implementar esta interface
- Qualquer classe implementando *actionPerformed()* pode funcionar como um ouvinte de eventos.

Interface para eventos

- Deve-se dar uma implementação para actionPerformed() dentro da classe GreeterGUI

```
…
 public void actionPerformed(ActionEvent e)
  {   if (e.getSource() == goButton)
      {   String name = inField.getText();
          display.append(greeter.greet(name) + "\n");
      }
  }
  ...
} // class
```

- Quando *goButton* é clicado o código em ActionPerformed() é executado

Interface para eventos

- Deve-se Informar que *GreeterGUI* será ActionListener para *goButton.*

```
public class GreeterGUI extends Frame implements
   ActionListener
{ ...
  public void buildGUI()
  {  ...
     goButton = new JButton(“Clique aqui para saudação!");
     goButton.addActionListener(this);
     ...
  }
  ...
```

Exemplo

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

public class GreeterGUI extends JFrame implements
   ActionListener
{ private JTextArea display;
  private JTextField inField;
  private JButton goButton;
  private Greeter greeter;
```

Exemplo

```
public GreeterGUI(String title)
 { greeter = new Greeter();
   buildGUI();
   setTitle(title);
   pack(); // java.awt Class Window: ajuste de janela
   setVisible(true);
 } // construtor
```

Exemplo

```
private void buildGUI()
 { Container contentPane = getContentPane();
   contentPane.setLayout(new BorderLayout());
   display = new JTextArea(10,30);
   inField = new JTextField(10);
   goButton = new JButton("Clique aqui para saudação!");
   goButton.addActionListener(this);
   JPanel inputPanel = new JPanel();
   inputPanel.add(new JLabel("Nome: "));
   inputPanel.add(inField);
   inputPanel.add(goButton);
   contentPane.add("Center", display);
   contentPane.add("South", inputPanel);   } // buildGUI()
```

Entrada e Saída

Exemplo

```
public void actionPerformed(ActionEvent e)
  { if (e.getSource() == goButton)
    { String name = inField.getText();
      display.append(greeter.greet(name) + "\n");
    }
} // actionPerformed()
} // GreeterGUI class
```

Exemplo

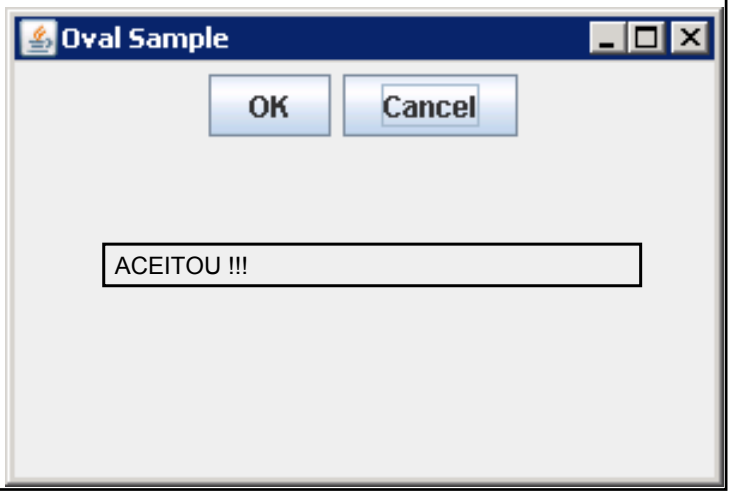
```
public class GreeterApplication
{  public static void main(String args[])
   {       new GreeterGUI("Greeter");    }
}

import javax.swing.*;
public class GreeterApplet extends JApplet
{    public void init()
     {         new GreeterGUI("Greeter");     }
}
```

Exercício

- Faça *JTextField* se tornar ouvinte de eventos
  - Ao teclar <enter> o usuário causa o mesmo efeito de um clique de mouse
  - Designe *inField* como um *ActionListener*
  - Alterações em *buildGUI()* e *actionPerformed()*

Exercício

- Aceita ou cancela

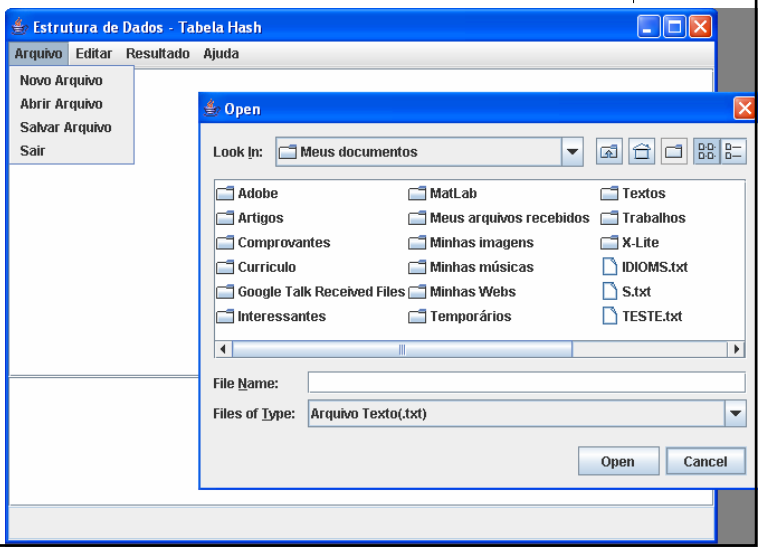


Exercício

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class PanelWithComponents {
  public static void main(String[] a) {
    JPanel panel = new JPanel();
    JButton okButton = new JButton("OK");
    panel.add(okButton);
    JButton cancelButton = new JButton("Cancel");
    panel.add(cancelButton);
    JFrame frame = new JFrame("Exemplo Aceita ou Cancela");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.add(panel);
    frame.setSize(300, 200);
    frame.setVisible(true);
  } }
```

Casos

- Menu



Casos

```
public class Interface extends javax.swing.JFrame {
    private JPopupMenu jPopup = new JPopupMenu();
    private JMenuItem cortar1 = new JMenuItem();
    private JMenuItem copiar1 = new JMenuItem();
    private JMenuItem colar1 = new JMenuItem();
    private JMenuItem selecionarTudo1 = new JMenuItem();
    private JScrollPane jScrollPane1= new JScrollPane();
    private JTextArea jTextArea1 = new JTextArea();
    private JTextField jTextField1 = new JTextField();
    private JMenuBar barraDeMenu = new JMenuBar();
    private JMenu arquivo = new JMenu();
    private JMenuItem novoArquivo = new JMenuItem();
    private JMenuItem abrirArquivo = new JMenuItem();
    private JMenuItem salvarArquivo = new JMenuItem();
    private JMenuItem sair = new JMenuItem();
...
```

Casos

```
private void Menus() {
        cortar1.setText("Cortar");
        jPopup.add(cortar1);
        copiar1.setText("Copiar");
        jPopup.add(copiar1);
        colar1.setText("Colar");
        jPopup.add(colar1);
        selecionarTudo1.setText("Selecionar Tudo");
        jPopup.add(selecionarTudo1);
        setTitle("Estrutura de Dados - Tabela Hash");
        inserir.setText("Inserir");
        inserir.setEnabled(false);
        menuResultado.add(inserir);
```

Casos

```
private class TrataEventos implements ActionListener{
  public void actionPerformed(ActionEvent actionEvent){
    if(actionEvent.getSource()== novoArquivo){
        jTextArea1.setText(null);
        textoResultado.setText(null);
        setarFalseSubMenus();
        jTextField1.setText("Pronto...");
    }else if(actionEvent.getSource()== abrirArquivo){
        jTextField1.setText("Abrindo Arquivo...");
        textoResultado.setText(null);
        setarFalseSubMenus();
        int resultado = escolher.showOpenDialog(null); ...
```

Casos

```
private void adicionaListenerEFiltro(){
    TrataEventos evento = new TrataEventos();
    novoArquivo.addActionListener(evento);
    abrirArquivo.addActionListener(evento);
    salvarArquivo.addActionListener(evento);
    sair.addActionListener(evento);
    cortar.addActionListener(evento);
    copiar.addActionListener(evento);
    colar.addActionListener(evento);
    selecionarTudo.addActionListener(evento); ...
```