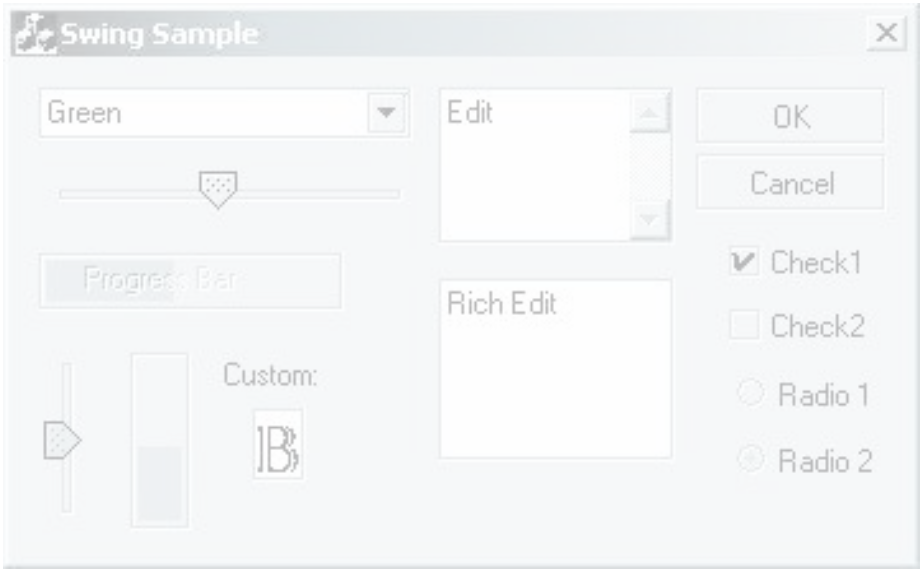


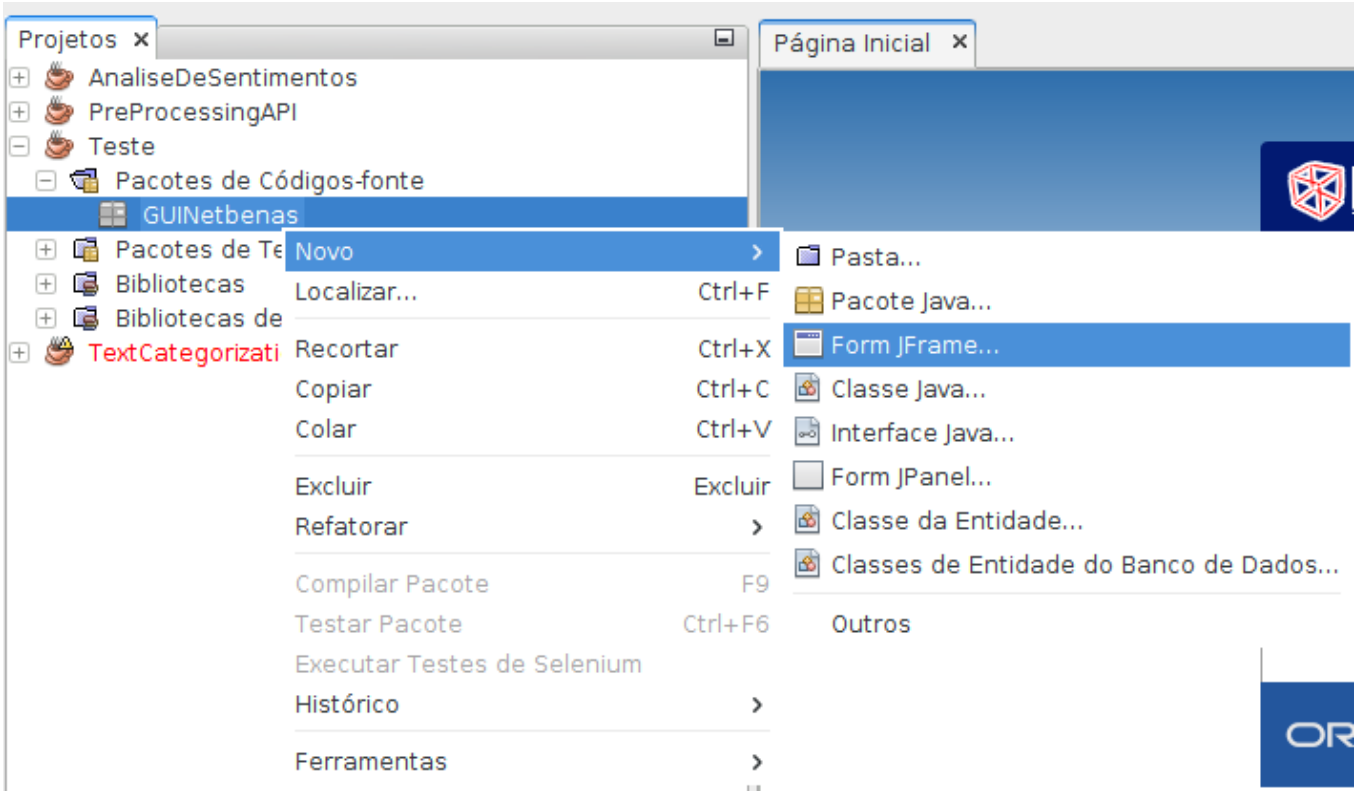
Aula 19

Componentes GUI no NetBeans

Rafael Geraldeli Rossi

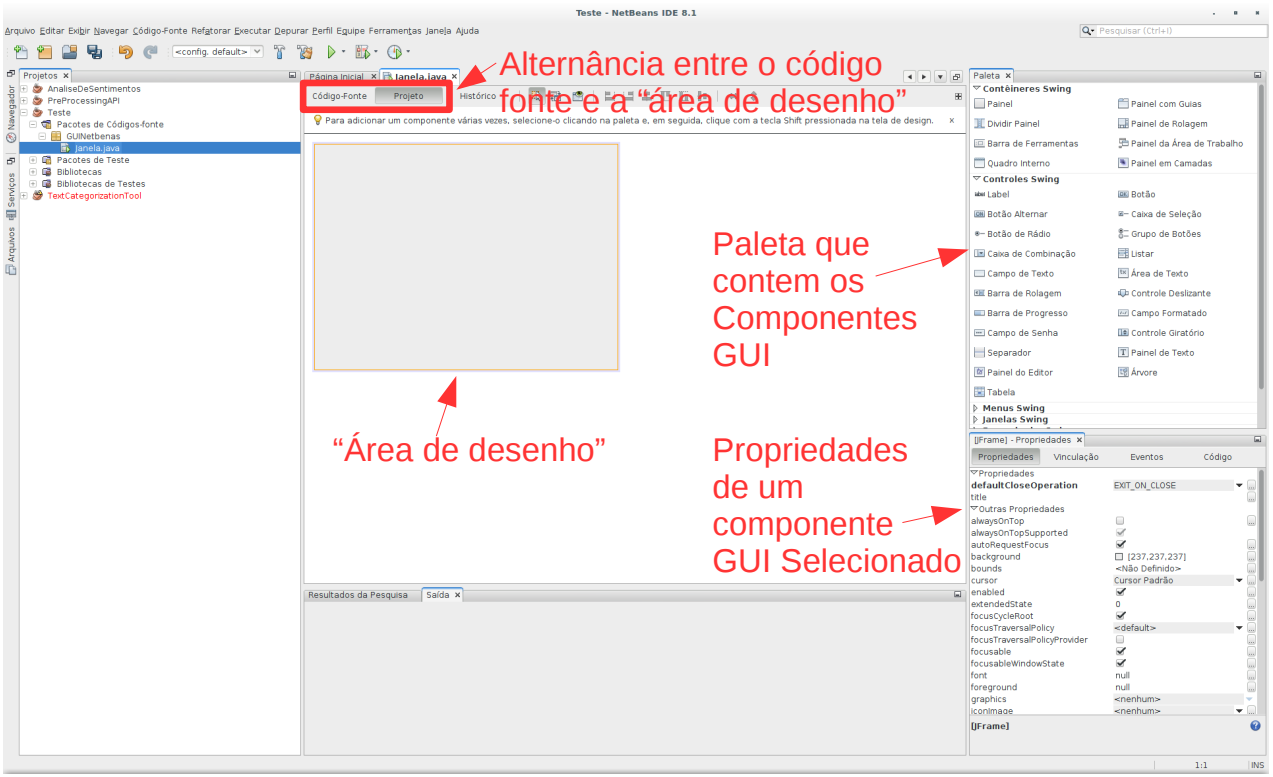
Criando uma Janela e Adicionando Componentes

- O NetBeans possibilita criar diretamente uma classe que já estende a classe `JFrame` para criar janelas



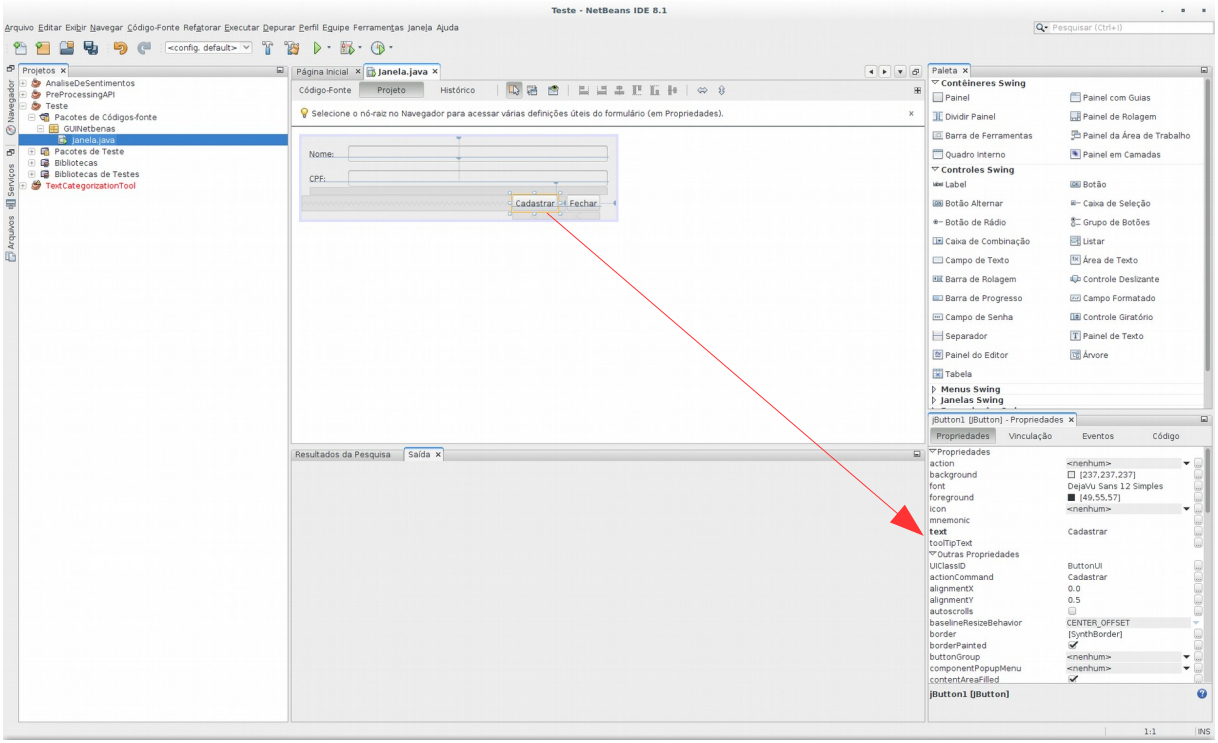
Criando uma Janela e Adicionando Componentes

- Para adicionar um componente em uma janela basta selecionar o componente na Paleta e depois "desenhar" (clicar e arrastar) o componente na área de desenho



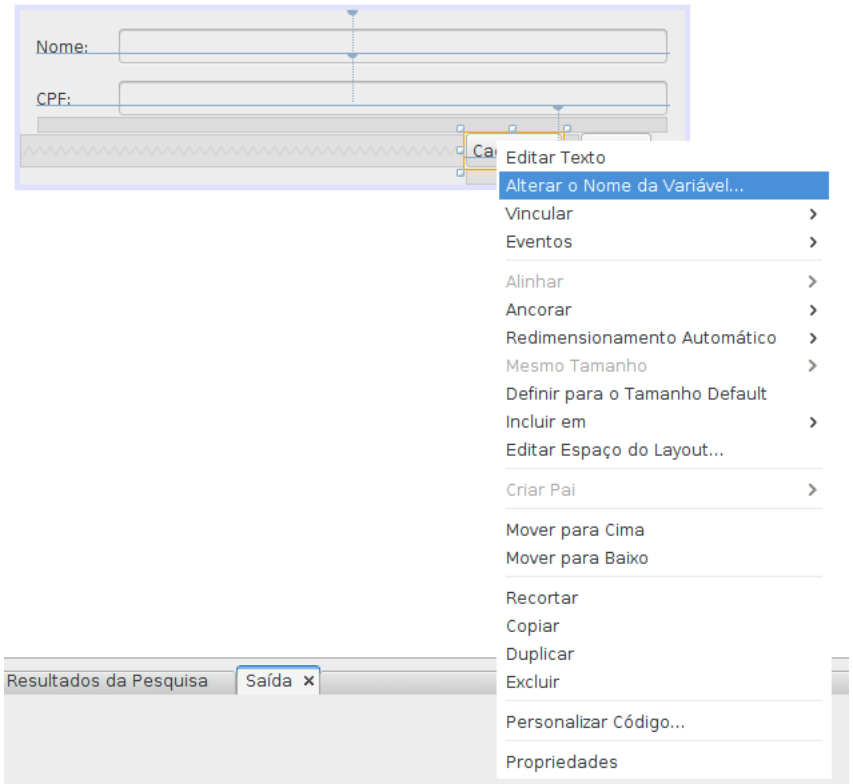
Criando uma Janela e Adicionando Componentes

- Para definir as propriedades dos objetos, como conteúdo, fonte, alinhamento do texto, habilitação, dentre outros, basta selecionar o objeto e editar suas propriedades na caixa de propriedades



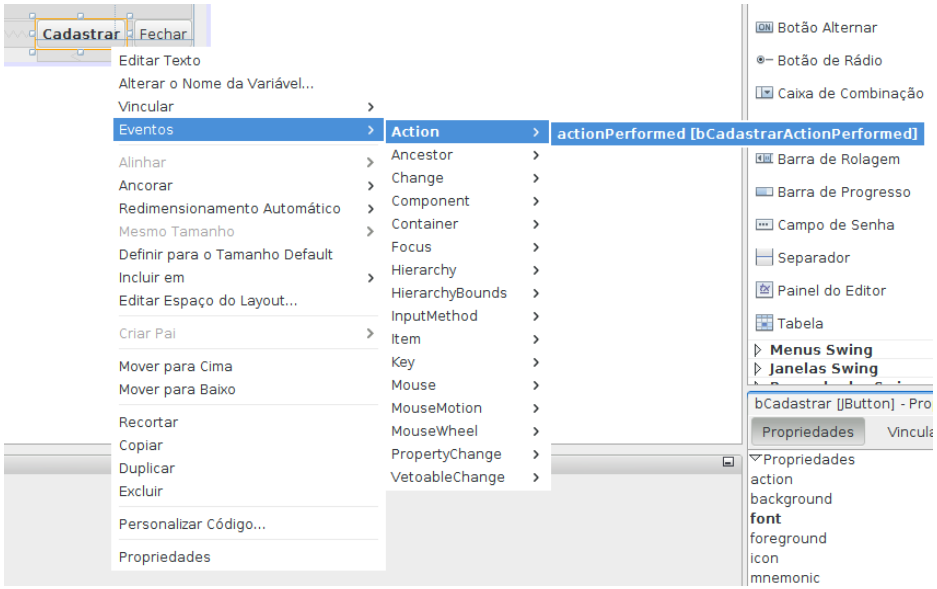
Criando uma Janela e Adicionando Componentes

- Para alterar o identificador do objeto (nome do objeto) basta clicar com o botão direito em cima do objeto de depois na opção `Alterar nome da variável...`



Criando uma Janela e Adicionando Componentes

- Para adicionar um *listener* para tratar um determinado tipo de evento, basta clicar com o botão direito do mouse em cima do componente que se deseja adicionar o *listener*, depois clicar em `Eventos`, clicar no tipo de evento (interface do *listener*) e depois no evento que se deseja tratar (método da interface do *listener*)



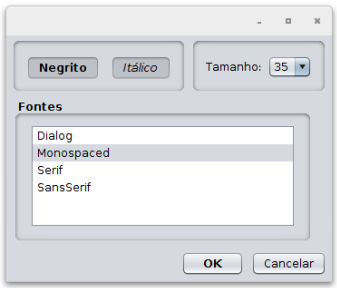
Geração de Código Automática

- Ao adicionar componente, e os *listeners* dos eventos, o NetBeans faz a geração dos códigos automaticamente

```
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    tNome = new javax.swing.JTextField();
    jLabel2 = new javax.swing.JLabel();
    tCPF = new javax.swing.JTextField();
    bCadastrar = new javax.swing.JButton();
    bFechar = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel1.setText("Nome:");

    jLabel2.setText("CPF:");

    bCadastrar.setFont(new java.awt.Font("DejaVu Sans", 1, 12)); // NOI18N
    bCadastrar.setText("Cadastrar");
    bCadastrar.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            bCadastrarActionPerformed(evt);
        }
    });

    bFechar.setText("Fechar");
    bFechar.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            bFecharActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
```

Comunicação Entre Janelas

- Para realizar a troca de "informações" entre janelas, podemos fazer uso de variáveis estáticas ou ainda passar variáveis de tipo por referência para as janelas e "armazenar" as informações nessas variáveis
- **Ex**: criando uma janela particular para o usuário alterar a fonte utilizando variáveis tipo por referência



Comunicação Entre Janelas

- Passando um objeto da janela "principal" como referência para a janela "secundária"

```
private void bFonteActionPerformed(java.awt.event.ActionEvent evt) {
    new JanelaFonte(tTexto);
}
```

Comunicação Entre Janelas

- Recebendo e armazenando a tipo por referência na janela secundária

```
public class JanelaFonte extends javax.swing.JFrame {

    JTextPane tTexto;

    public JanelaFonte(JTextPane tTexto) {
        this.tTexto = tTexto;
        initComponents();
        this.setVisible(true);
    }
```

Comunicação Entre Janelas

- Quando o usuário clica no botão OK, altera-se a fonte da referência

```
private void OKActionPerformed(java.awt.event.ActionEvent evt) {
    int tipo = 0;
    if(bNegrito.isSelected()){
        tipo += Font.BOLD;
    }
    if(bItalico.isSelected()){
        tipo += Font.ITALIC;
    }
    int size = Integer.parseInt(cSize.getSelectedItem().toString());
    String familia = lFontes.getSelectedValue();
    Font font = new Font(familia, tipo, size);
    tTexto.setFont(font);
    this.dispose();
}
```

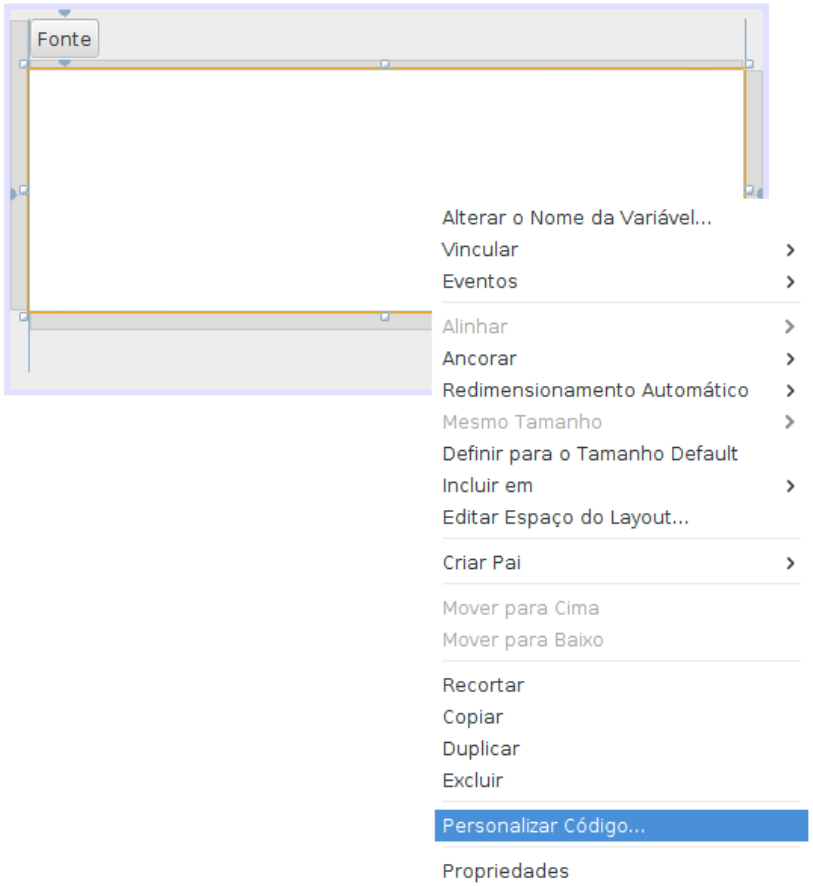
Comunicação Entre Janelas

- Para fazer o mesmo exemplo só que agora com variáveis estáticas é preciso fazer com que
  - As variáveis que irão receber a "comunicação" da outra janela tem que ser estáticas
  - Implementar métodos `gets` para essas variáveis

- Para tal é necessário alterar o código gerado pelo NetBeans

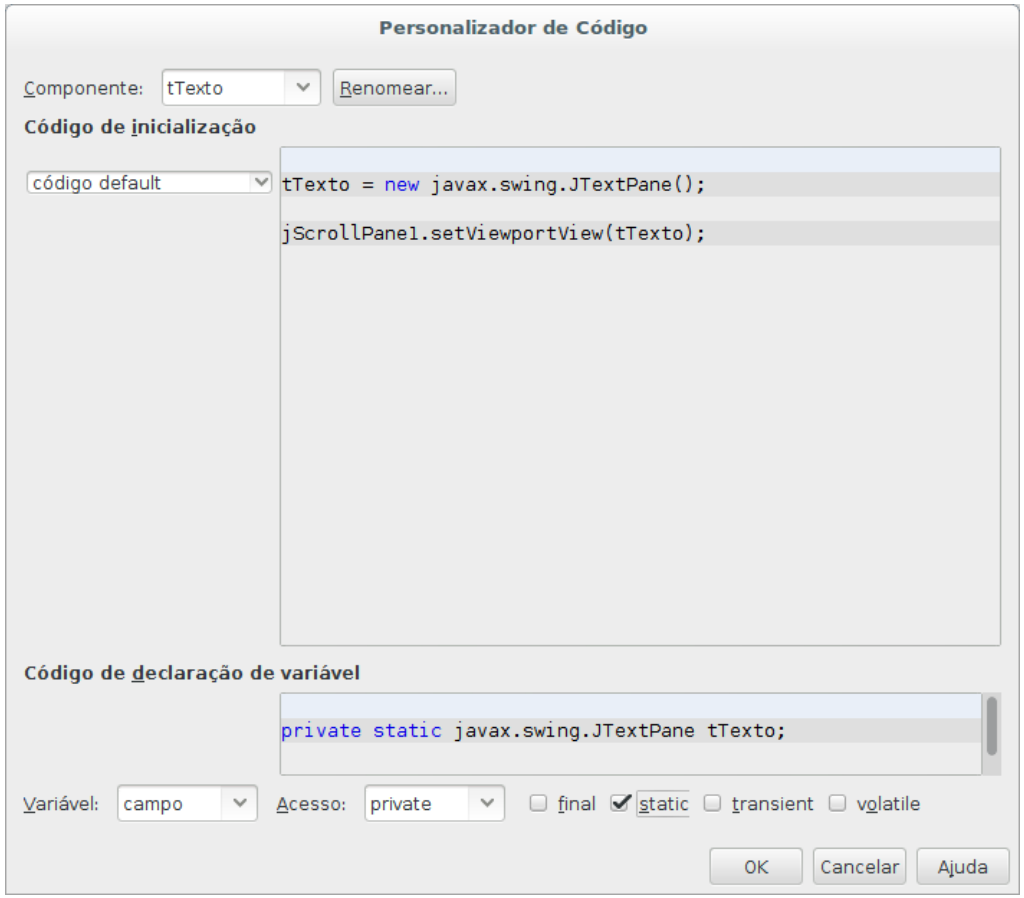
Comunicação Entre Janelas

- Alterando a declaração de variáveis gerada pelo NetBeans



Comunicação Entre Janelas

- Alterando a declaração de variáveis gerada pelo NetBeans



Comunicação Entre Janelas

- Criando métodos estáticos para retornar os objetos da "comunicação"

```
static public JTextPane getTTexto(){
    return tTexto;
}

// Variables declaration - do not modify
private javax.swing.JButton bFonte;
private javax.swing.JButton jButton1;
private javax.swing.JScrollPane jScrollPane1;
private static javax.swing.JTextPane tTexto;
```

Comunicação Entre Janelas

- Alterando o valor de um objeto estático de uma outra janela

```
private void OKActionPerformed(java.awt.event.ActionEvent evt) {
    int tipo = 0;
    if(bNegrito.isSelected()){
        tipo += Font.BOLD;
    }
    if(bItalico.isSelected()){
        tipo += Font.ITALIC;
    }
    int size = Integer.parseInt(cSize.getSelectedItem().toString());
    String familia = lFontes.getSelectedValue();
    Font font = new Font(familia, tipo, size);

    JanelaTexto.getTTexto().setFont(font);

    this.dispose();
}
```

Extra: *Look and Feels*

- Um *Look and Feel* determina como o componente será renderizado (cor, formato, ...)
- O pacote `javax.swing` possui um conjunto de classes do tipo `LookAndFeel` (esta última é abstrata)
- As classes `LookAndFeel` podem varia de acordo com o Sistema Operacional
- Entretanto, há classe `LookAndFeel` invariantes ao SO

Extra: *Look and Feels*

- Listando os `LookAndFeels` instalados

```
165             try {
166                 for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {
167                     System.out.println(info.getClassName());
168                 }
169             } catch (Exception e) {
170                 System.err.println("Erro no Look and Fell");
171             }
172
```

gui.FormExtrato > main >

Saída × | Resultados da Pesquisa ×

Console do Depurador × | Projeto Banco (run) × | Projeto Banco (run) #2 ×

```
run:
javax.swing.plaf.metal.MetalLookAndFeel
javax.swing.plaf.nimbus.NimbusLookAndFeel
com.sun.java.swing.plaf.motif.MotifLookAndFeel
com.sun.java.swing.plaf.gtk.GTKLookAndFeel
```

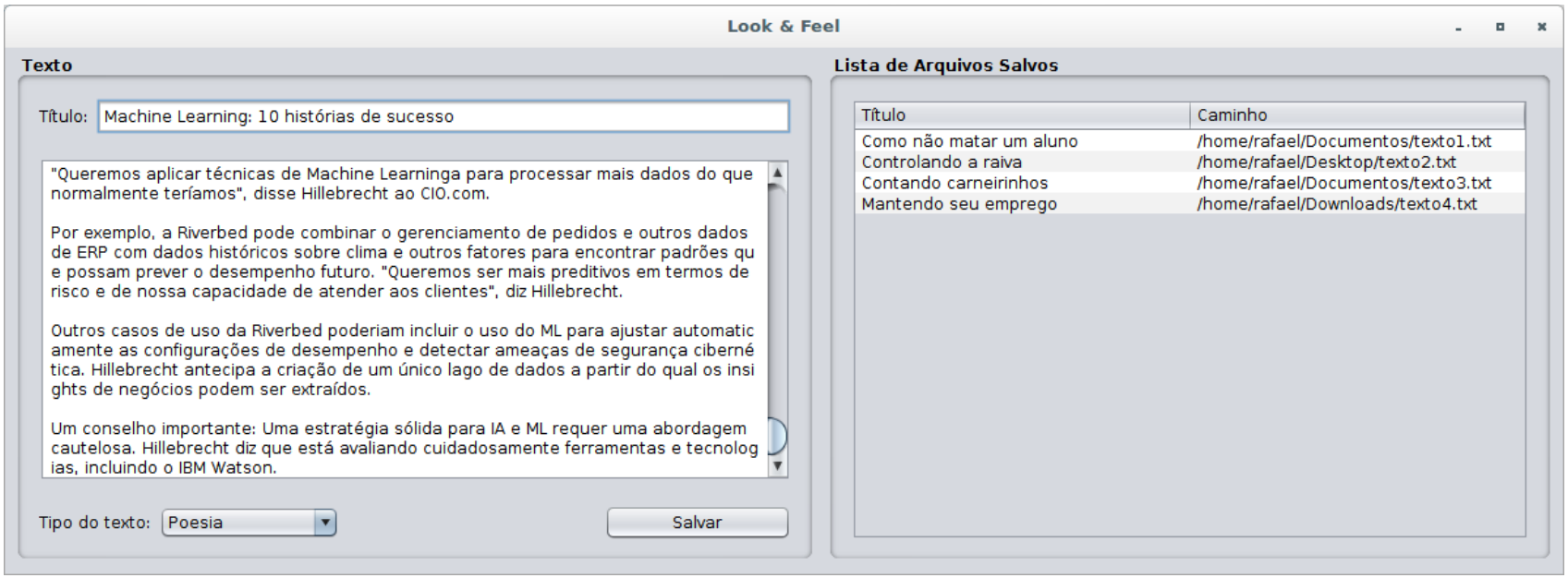
Extra: *Look and Feels*

- Para definir um *LookAndFeel*, basta informar o nome da classe dentro do método UIManager.setLookAndFeel(...)

```
try {
    UIManager.setLookAndFeel("com.sun.java.swing.plaf.gtk.GTKLookAndFeel");
} catch (Exception e) {
    System.err.println("Erro no Look and Fell");
}
```

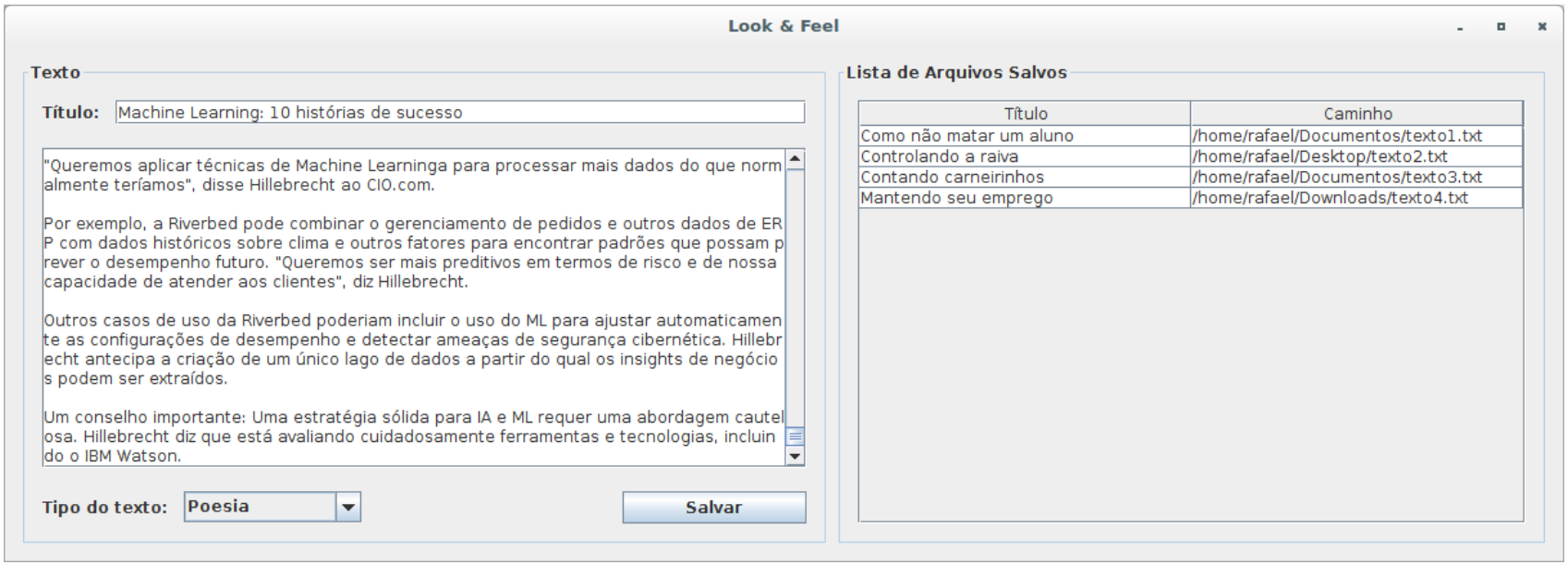
Extra: *Look and Feels*

Nimbus



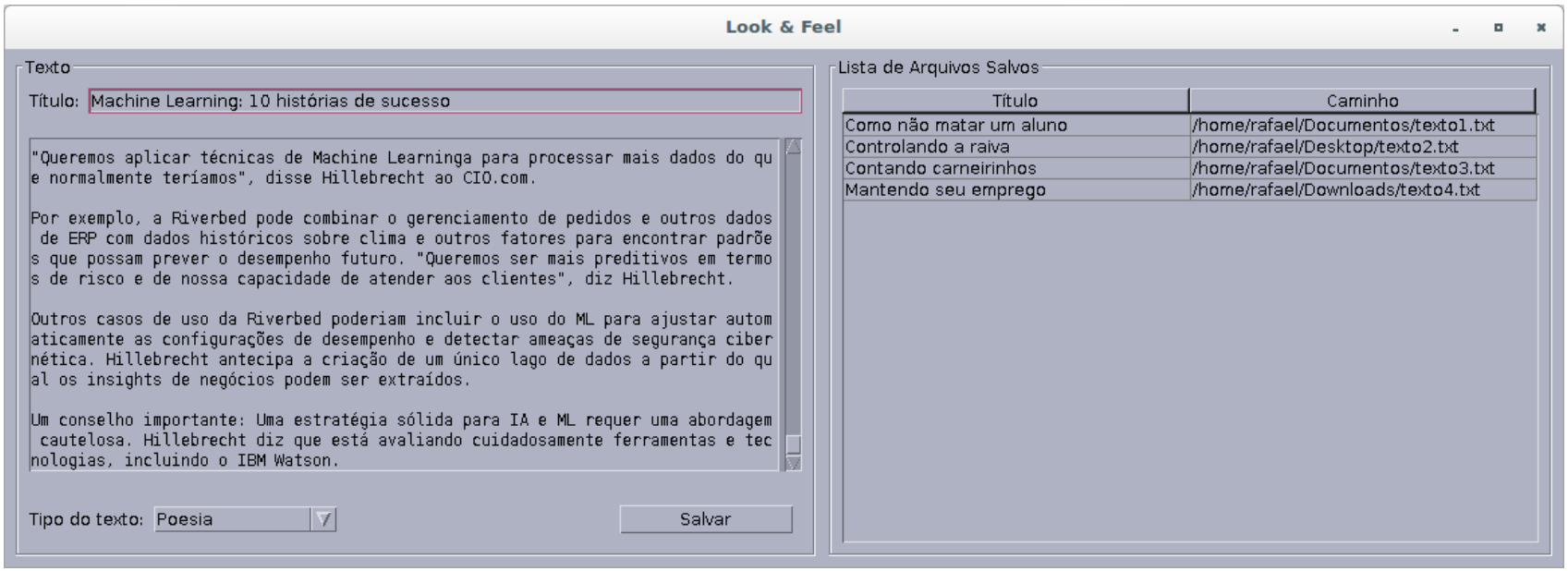
Extra: *Look and Feels*

Metal



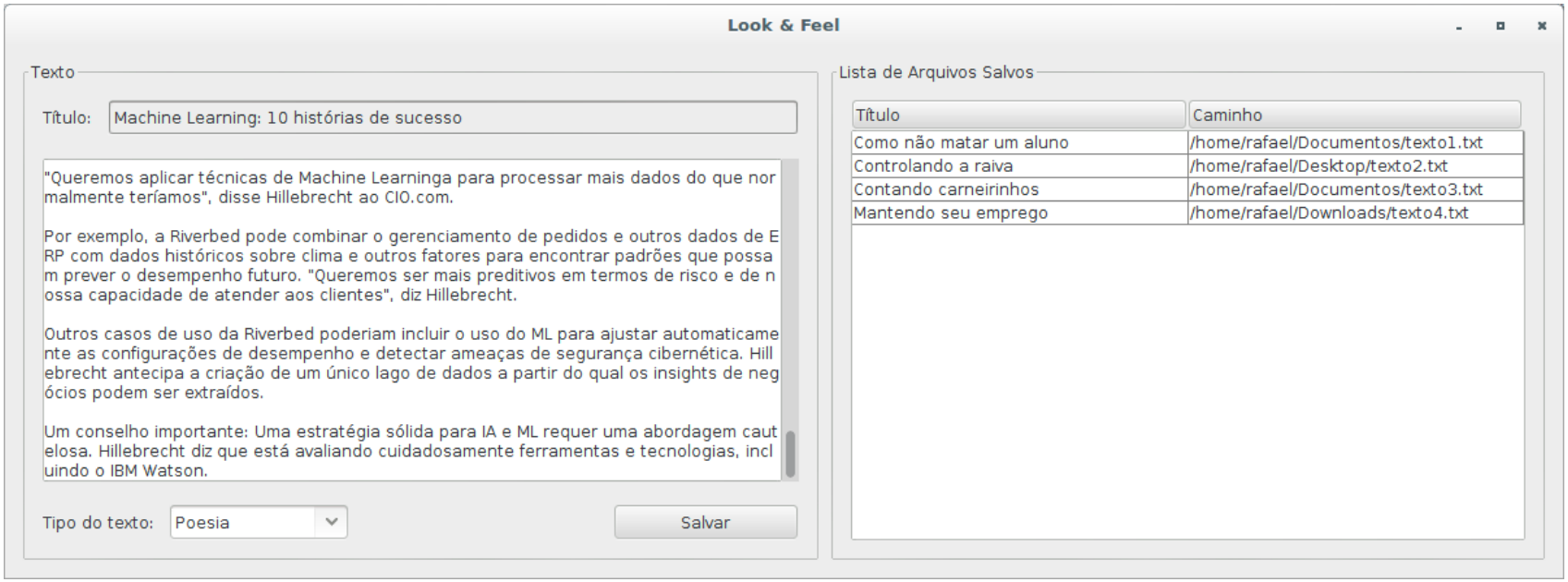
Extra: *Look and Feels*

Motif



Extra: *Look and Feels*

GTK



Material Complementar

- NetBeans GUI Builder - Introduction

  `https://www.youtube.com/watch?v=8CI718A8OUI`

- NetBeans GUI Builder - ActionListener

  `https://www.youtube.com/watch?v=iWyXqgZk4iw`

- Projetando uma GUI Swing no NetBeans IDE

  `https://netbeans.org/kb/docs/java/quickstart-gui_pt_BR.html`

Material Complementar

- Pluggable Look-and-feel com Swing

  https://www.devmedia.com.br/pluggable-look-and-feel-com-swing/6327

- Testando vários Look And Feels

  http://javafree.uol.com.br/artigo/871502/

Imagem do Dia

Programação Orientada a Objetos
http://lives.ufms.br/moodle/

Rafael Geraldeli Rossi
rafael.g.rossi@ufms.br

Slides baseados em [Deitel and Deitel, 2010]

Referências Bibliográficas I

Deitel, P. and Deitel, H. (2010).
Java: How to Program.
How to program series. Pearson Prentice Hall, 8th edition.