Características das Linguagens de Programação I

Conteúdo: Projeto de aplicação - parte II

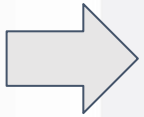
Prof. Dsc. Giomar Sequeiros
giomar@eng.uerj.br

Classes de negócio

Criando as classes de negócio (1)

- No pacote Negocio crie as classes Paciente, Psicologo, Anamnese e Sessao com seus respectivos métodos getters e setters. O código abaixo mostra a classe Paciente.

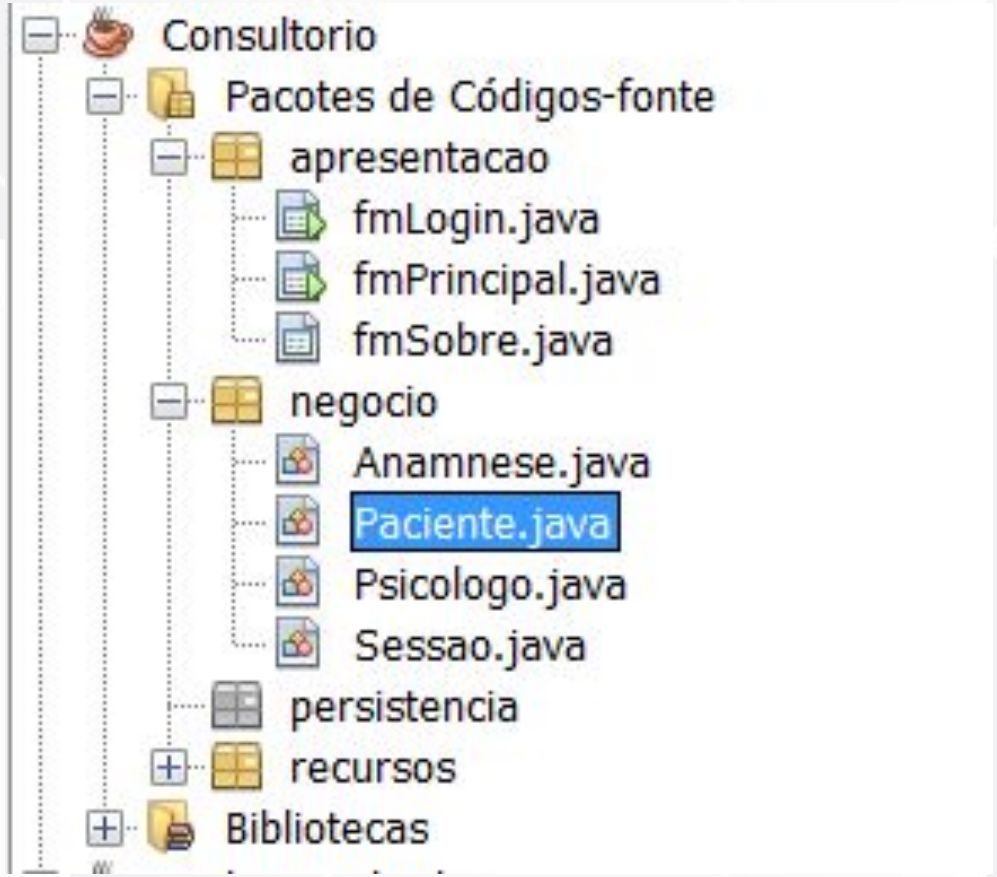
```
public class Paciente {
  //Atributos
  private int idPaciente;
  private String nome;
  private String cpf;
  private Calendar data_nascimento;
  private String sexo;
  private String endereco;
  private String telefone;
  private String foto;
  private String plano saude;
  private String observacoes;
  private Calendar data_cadastro;

  //Completar com construtores e Métodos getter e setters
}
```



```
package negocio;

import java.util.Calendar;

/**
 *
 * @author Giomar
 */
public class Paciente {
  //Atributos
  private int idPaciente;
  private String nome;
  private String cpf;
  private Calendar data_nascimento;
  private String sexo;
  private String endereco;
  private String telefone;
  private String foto;
  private String plano_saude;
  private String observacoes;
  private Calendar data_cadastro;

  //Métodos getter e setters
}
```

Criando as classes de negócio (2)

- Andamento do projeto



Criando o DAO

Adicionando JDBC - opção maven

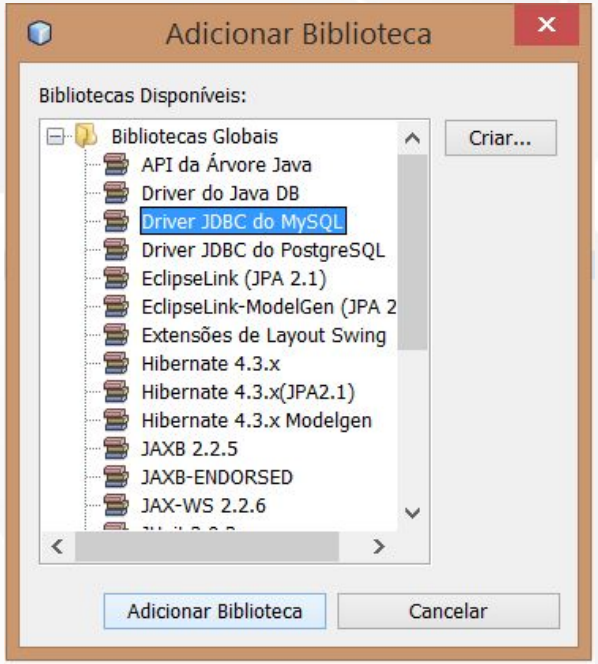
- Se estiver usando um sistema de build como Maven, adicione a dependência apropriada.no arquivo **pom.xml** de seu projeto

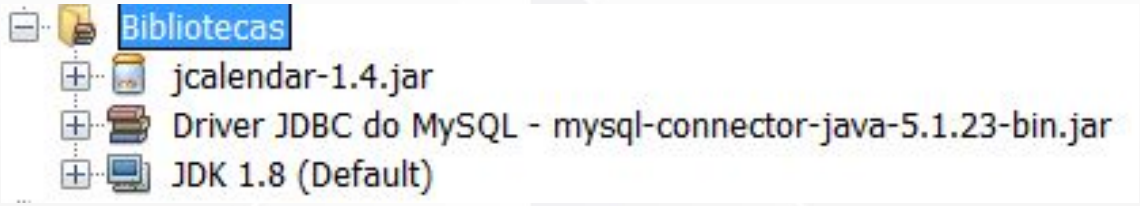
```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.34</version>
</dependency>
```

Os driver do MySQL podem ser obtidas no repositório Mavem:
https://mvnrepository.com/artifact/mysql/mysql-connector-java

Adicionando JDBC - opção manual

- Para que seja possível a comunicação entre Java e MySQL é necessário o driver de MySQL. No explorador de projetos, clicar no botão direito sobre a pasta biblioteca





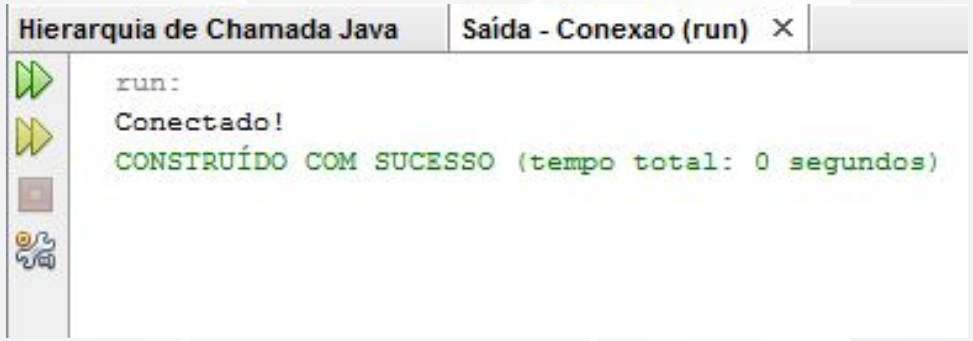
- Caso não exista a biblioteca, fazer download aqui: (http://dev.mysql.com/downloads/connector/odbc/). Após baixar o JAR, clique no botão direito na pasta bibliotecas para adicioná-lo

Adicionando JDBC

- Para testar a conexão pode-se criar uma classe de teste e no método main escreva o código abaixo.

- **Obs**. Substituir dbprontuarios por **dbconsultorio** e conferir o usuário e senha do MySql.

```
public static void main(String[] args) throws SQLException {
    Connection conexao = DriverManager.getConnection("jdbc:mysql://localhost/dbprontuarios","root","");
    System.out.println("Conectado!");
    conexao.close();
}
```



Criando uma conexão com o banco de dados

- Primeiramente criaremos uma fábrica de conexões. No pacote **persistencia** crie a classe **ConFactory** e escreva o código abaixo:

```
package persistencia;
/**
 *
 * @author Giomar
 */

import java.sql.Connection; // conexão SQL para Java
import java.sql.DriverManager; // driver de conexão SQL para Java
import java.sql.SQLException; // classe para tratamento de exceções

public class ConFactory {
    public Connection getConnection() {
        try {
            return DriverManager.getConnection("jdbc:mysql://localhost/dbconsultorio","root","");
        }
        catch(SQLException excecao) {
            throw new RuntimeException(excecao);
        }
    }
}
```

Cadeia de conexão, modificar usuário e senha conforme a configuração do MySQL

DAO (Data Access Object)

- No projeto criaremos interfaces por cada objeto a ser persistido depois uma classe que implemente essas interfaces. No pacote persistência, crie uma interface chamada de **IPacienteDAO** conforme a Figura abaixo:

```
package persistencia;

import java.util.List;
import negocio.Paciente;

/**
 *
 * @author Giomar
 */
public interface IPacienteDAO {
    public void adiciona(Paciente paciente);
    public void altera(Paciente paciente);
    public void remove(int id);
    public List<Paciente> listarTodos();
    public Paciente getByID(int id);
}
```

Exercício
De forma similar, crie as interfaces
- **IPsicologoDAO,**
- **IAnamneseDAO**
- **ISessaoDAO**

PacienteDAO (1)

- Agora criaremos as classes que implementem essas interfaces. No pacote **persistencia** crie a classe **PacienteDAO**, conforme o código mostrado:

```
package persistencia;

/**
 *
 * @author Giomar
 */
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Date;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import negocio.Paciente;

public class PacienteDAO implements IPacienteDAO{

    // a conexão com o banco de dados
    private Connection connection;

    //construtor
    public PacienteDAO() {
        this.connection = new ConFactory().getConnection();
    }

    //métodos
```

PacienteDAO (2)

- Sobrescreva o método adiciona:

```
//métodos
 @Override
public void adiciona(Paciente paciente) {
    //criamos a cadeia sql para inserção de dados na tabela paciente
    String sql = "insert into paciente " +
         "(nome, cpf, data_nascimento, sexo, endereco, telefone, foto, plano_saude, observacoes, data_cadastro)" +
         " values (?,?,?,?,?,?,?,?,?,now())";

 try {
     // preparar o comando para inserção
     PreparedStatement stmt = connection.prepareStatement(sql);

     //setando os atributos do objeto na cadeia sql
     stmt.setString(1,paciente.getNome());
     stmt.setString(2,paciente.getCpf());
     stmt.setDate(3, new Date(paciente.getData_nascimento().getTimeInMillis()));
     stmt.setString(4,paciente.getSexo());
     stmt.setString(5,paciente.getEndereco());
     stmt.setString(6,paciente.getTelefone());
     stmt.setString(7,paciente.getFoto());
     stmt.setString(8,paciente.getPlano_saude());
     stmt.setString(9,paciente.getObservacoes());

     // executa
     stmt.execute();
     stmt.close();
 } catch (SQLException e) {
     throw new RuntimeException(e);
 }
}
```

Comando SQL que será executado no do MySQL

PacienteDAO (3)

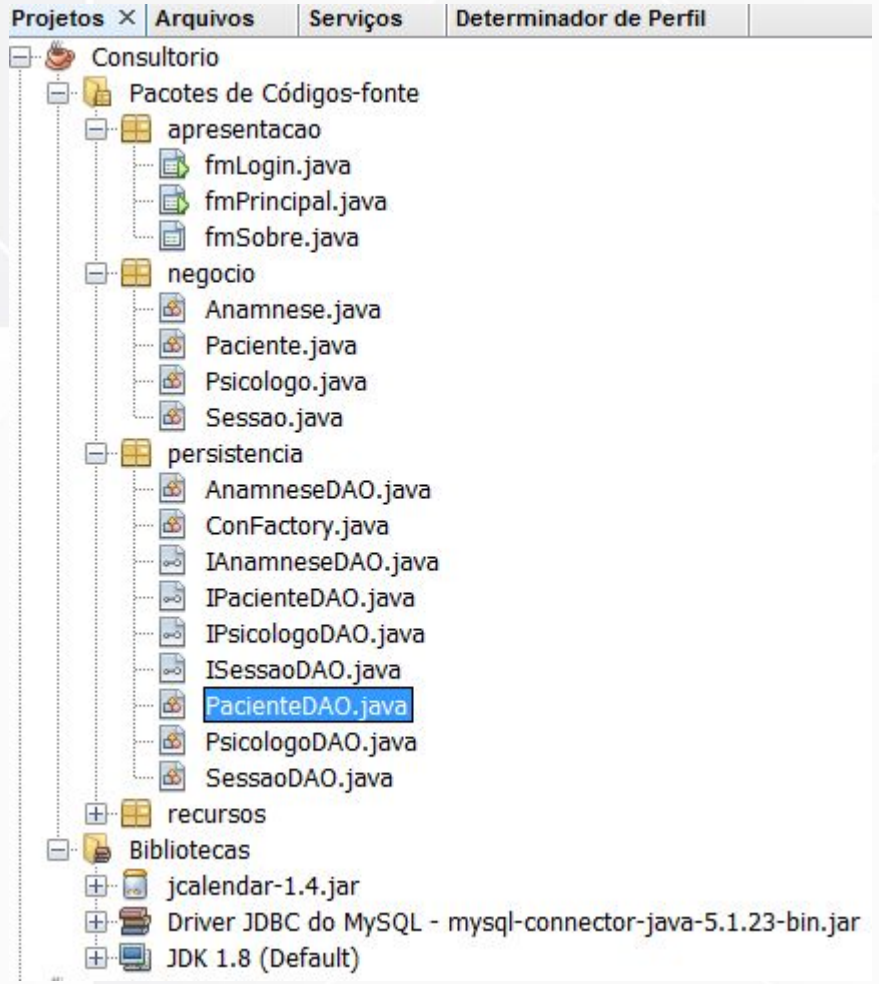
- Deixaremos os métodos restantes prontos para serem implementados no futuro:

```
@Override
public void altera(Paciente paciente) {
    throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public void remove(int id) {
    throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public List<Paciente> listarTodos() {
    throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public Paciente getByID(int id) {
    throw new UnsupportedOperationException("Not supported yet.");
}
```

Exercício:

- **De forma similar, crie as classes PsicologoDAO, AnamneseDAO e SessaoDAO**

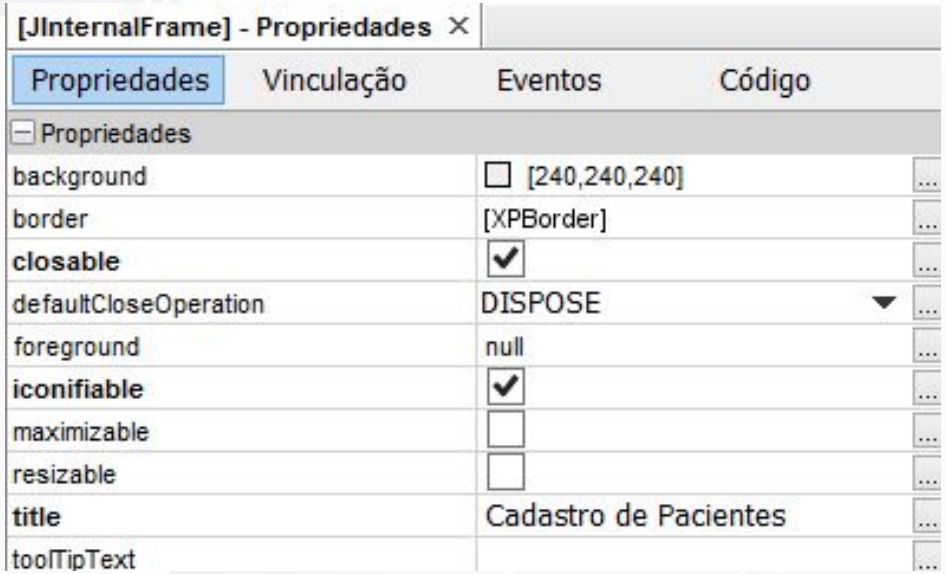
Andamento do projeto



Cadastro de paciente

Criação da GUI Paciente(1)

- No pacote apresentação, criar um novo **JInternalFrame,** com o nome fmPaciente.
- Modificar a propriedade **closable** e **iconifiable** do **JInternalFrame** fmPaciente para **true**. Finalmente acrescente um título adequado.



Criação da GUI Paciente(2)

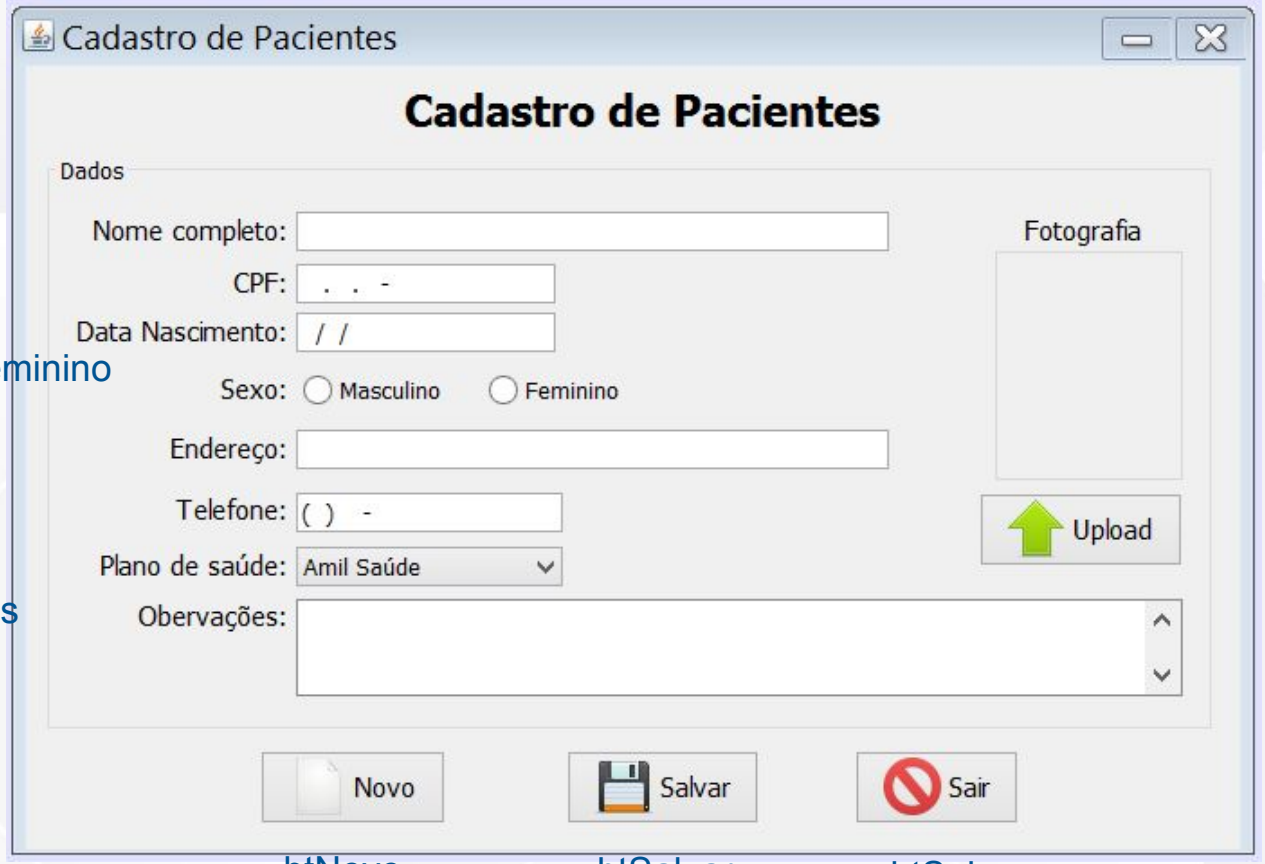
- A seguir vamos vincular nossa GUI principal com a GUI do paciente. Para isso adicione o evento **ActionPerformed** no item de menu “Paciente” e escrever o código abaixo.

```
fmPaciente paciente=new fmPaciente();
jDesktopPane1.add(paciente);//adicionar o formulário em nosso container
paciente.setVisible(true);
```

- Idem para o botão paciente do programa principal

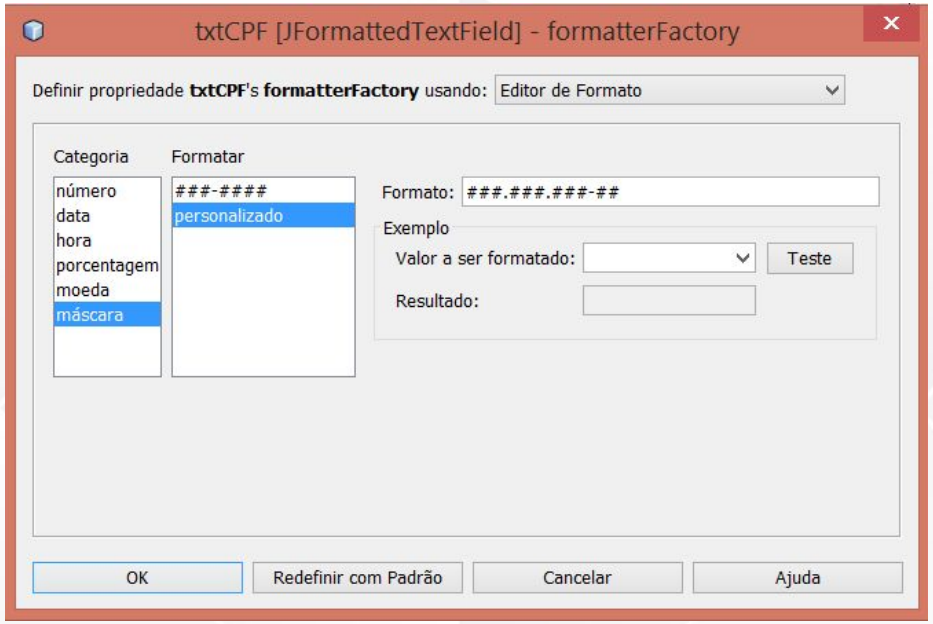
Criação da GUI Paciente(3)

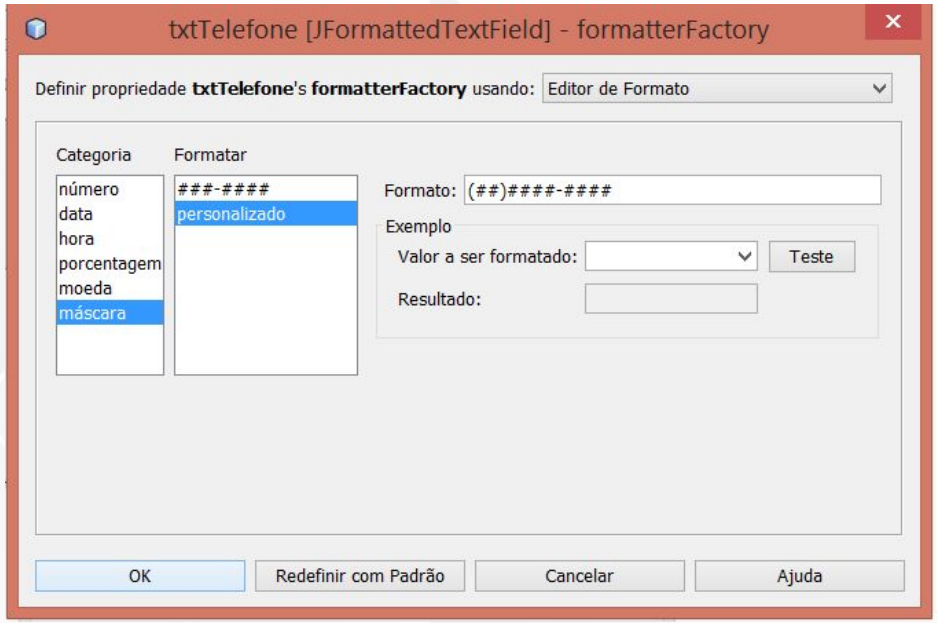
- Adicionar componentes JLabel e JTextField, JFormatedText, JComboBox, JRadioButton, JTextArea, Jpanel e Jbutton conforme a Figura abaixo. Modifique o nome de variável de cada componente.



Criação da GUI Paciente(4)

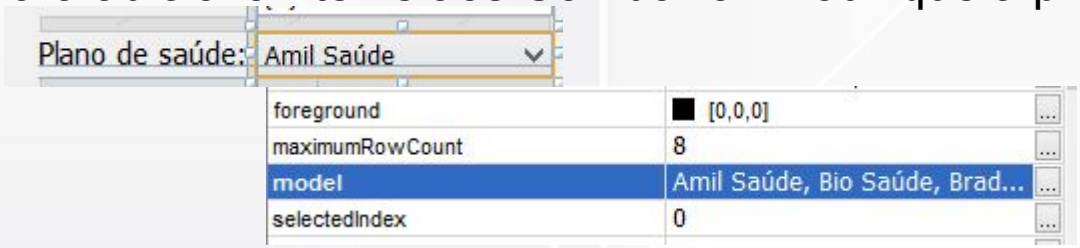
- Para formatar o cpf, data e telefone modifique a propriedade **Formatted Factory**, conforme a figura.

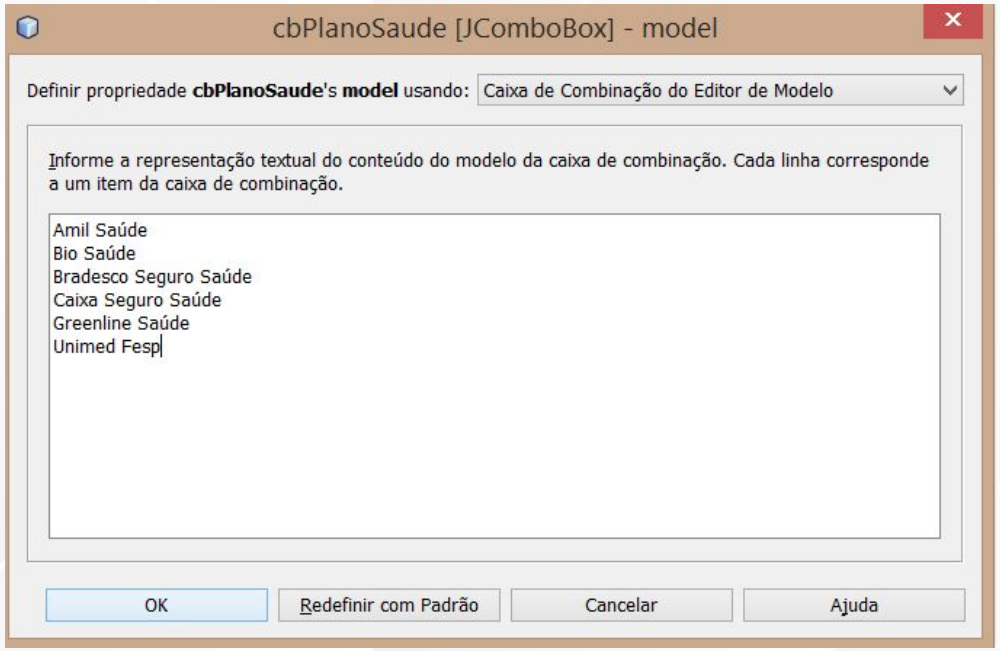




Criação da GUI Paciente(5)

- Para adicionar items aos ComboBox modifique a propriedade **model.**

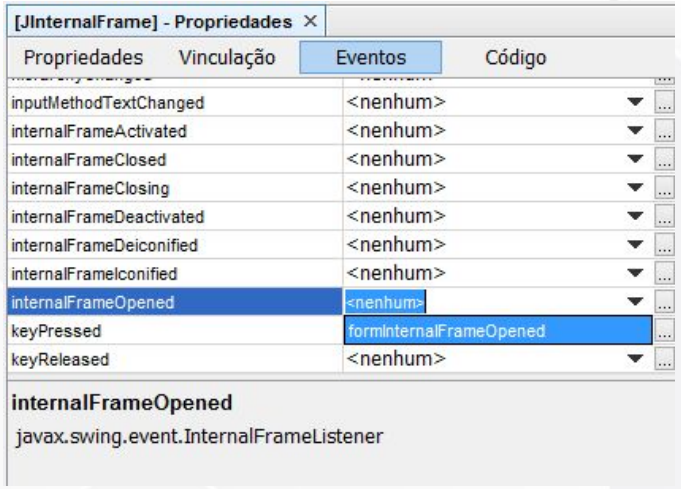




- Adicione os items: Amil Saúde, Bio Saúde, Bradesco Seguro Saúde, Caixa Seguro Saúde, Greenline Saúde, Unimed Fesp.

Criação da GUI Paciente(6)

- Crie o método **habilitar** que recebe um valor booleano. Este método permitirá habilitar e desabilitar componentes. Quando a GUI paciente é instanciada os componentes devem estar desabilitados (a exceção do botão Novo).
- Crie o método **limpar**(), este será chamado cada vez que clicamos no botão Novo, e após salvar um registro
- No formulário **fmPacientes**, adicionar o evento **internalFrameOpened**, que servirá para realizar uma ação uma vez o usuário abra o formulário:



- Depois escreva o código abaixo:

```
private void formInternalFrameOpened(javax.swing.event.InternalFrameEvent evt) {
    habilitar(false); //Desabilitar os componentes
    limpar(); //Limpar os componentes
}
```

Criação da GUI Paciente(7)

- No evento **ActionPerformed** do botão btNovo adicione o código abaixo:

```
private void btNovoActionPerformed(java.awt.event.ActionEvent evt) {
    habilitar(true); //Habilitar componentes
    limpar(); //Preparamos os componentes para inserção de dados
}
```

- Não esqueça de implementar a ação do botão **Sair**: this.dispose();

Salvar Paciente(1)

- Para fazer upload da foto do paciente escreva o código abaixo no evento **ActionPerformed** do botão **Upload**:

Foi adicionado o atributo **arquivoFoto** para armazenar temporalmente a **url** da foto.

O código exibirá a foto selecionada no painel paFoto.

```
File arquivoFoto=null;
private void btUploadActionPerformed(java.awt.event.ActionEvent evt) {
    //abre uma janela para procurar a foto
    JFileChooser filechooser= new JFileChooser();
    filechooser.setDialogTitle("Escolha uma foto");
    filechooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
    //após selecionar o arquivo
    int returnval=filechooser.showOpenDialog(this);
    if(returnval==JFileChooser.APPROVE_OPTION)
    {
        arquivoFoto = filechooser.getSelectedFile();
        BufferedImage bi;
        try
        {   //mostrar a imagem no painel
            bi=ImageIO.read(arquivoFoto);
            JLabel imgLabel = new JLabel(new ImageIcon(bi));
            imgLabel.setSize(150,150);
            paFoto.add(imgLabel);
            paFoto.revalidate();
            paFoto.repaint();
        }
        catch(IOException e)
        {
            System.out.println("Houve um erro ao carregar a foto");
        }
        this.pack();
    }

}
```

Salvar Paciente(2)

- No evento **ActionPerformed** do botão **Salvar** escreva o código mostrado:

Recupera os dados inseridos pelo usuário e cria um objeto **paciente**.

Invoca o método adiciona para salvar os dados fornecidos pelo usuário

```
private void btSalvarActionPerformed(java.awt.event.ActionEvent evt) {
    int valor = JOptionPane.showConfirmDialog(null, "Tem certeza que deseja salvar?","Sistema Consultório Médicos", 1);
    if(valor==0){
        //recuperar os dados inseridos
        Paciente paciente = new Paciente();
        paciente.setNome(txtNome.getText());
        paciente.setCpf(txtCPF.getText());
        //tratando a data
        try {
            SimpleDateFormat data = new SimpleDateFormat("dd/MM/yyyy");
            Calendar cal = Calendar.getInstance();
            cal.setTime(data.parse(txtData.getText()));
            paciente.setData_nascimento(cal);
        } catch (ParseException e) {
            System.out.println(e);
        }
        if(rbMasculino.isSelected())
            paciente.setSexo("M");
        else if(rbFeminino.isSelected())
            paciente.setSexo("F");
        paciente.setEndereco(txtEndereco.getText());
        paciente.setTelefone(txtTelefone.getText());
        paciente.setPlano_saude(cbPlanoSaude.getSelectedItem().toString());

        //movemos a foto para um especifico D:/fotos/
        String novoNomeFoto=txtNome.getText().replaceAll(" ","") + txtNome.getText().hashCode()+".png";
        File urlFoto = new File("D:/fotos/"+ novoNomeFoto);
        try {
            copiarArquivo(arquivoFoto, urlFoto); //método copiar arquivo (origem, destino)
        } catch (IOException ex) {
            System.out.println("Nao foi possivel mover o arquivo, "+ex);
        }
        paciente.setFoto(urlFoto.getPath());
        paciente.setObservacoes(txtObservacoes.getText());
        // gravamos os dados
        IPacienteDAO dao = new PacienteDAO();
        dao.adiciona(paciente);
        JOptionPane.showMessageDialog(null, "Os dados foram gravados");
        //limpar formulário
        limpar();
        habilitar(false);
    }
}
```

Trata o campo data inserido.

Salvar Paciente(3)

- O método copiar arquivo:

```
private void copiarArquivo(File origem, File destino) throws IOException {
    InputStream in = new FileInputStream(origem);
    OutputStream out = new FileOutputStream(destino);   // Transferindo bytes de entrada para saída
    byte[] buf = new byte[1024];
    int len;
    while ((len = in.read(buf)) > 0) {
        out.write(buf, 0, len);
    }
    in.close();
    out.close();
}
```

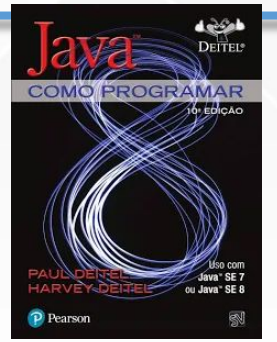
- O destino das fotos será no diretório "**D:/fotos**", antes de executar o programa, tenha certeza que existe esse diretório.

- Teste a funcionalidade!

Referências

BIBLIOGRAFIA BÁSICA:

- DEITEL, Harvery M.. Java : como programar. 10ª ed. São Paulo: Pearson - Prentice Hall, 2017.



- BORATTI, Isaías Camilo. Programação Orientada a Objetos em Java : Conceitos Fundamentais de Programação Orientada a Objetos. 1ª ed. Florianópolis: VisualBooks, 2007.

- SIERRA, Kathy; BATES, Bert. Use a Cabeça! Java. 2ª ed. Rio de Janeiro: Alta Books, 2007.