Características das Linguagens de Programação I

Conteúdo: Projeto de aplicação - parte I

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

Fundamentos

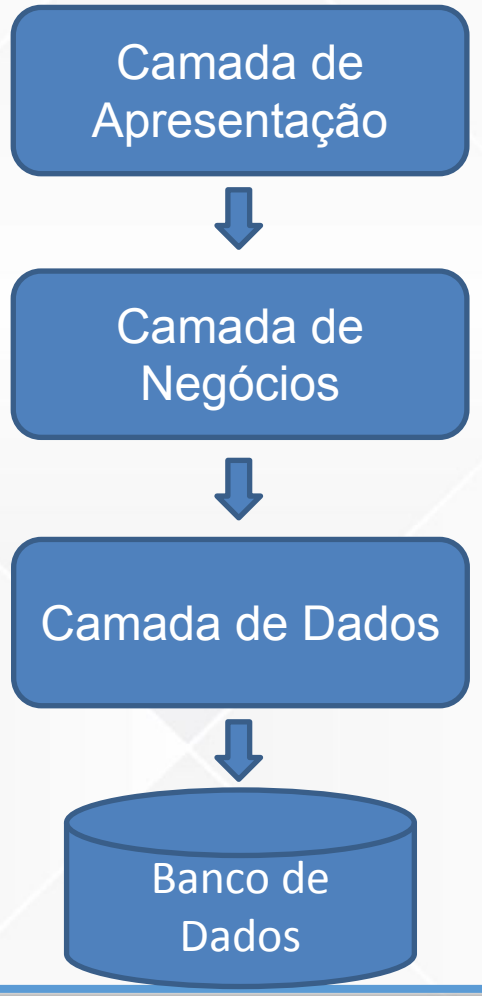
Modelo 3 camadas (3-tier)

- Modelo em três camadas, derivado do modelo **'n' camadas**. Estimula a **organização** da **arquitetura** do **sistema** em um conjunto de camadas coesas com fraco acoplamento entre elas.

- Cada camada possui um propósito bem definido

- Componentes:
  - Camada de Apresentação
  - Camada de Negócios
  - Camada de Dados

Modelo 3 camadas (3-tier)



Camada de apresentação

- É a chamada **GUI (Graphical User Interface)**, ou simplesmente interface.

- Esta camada **interage** diretamente com o **usuário**, é através dela que são feitas as requisições como consultas, por exemplo.

Camada de negócio

- Também chamada de **Lógica empresarial**, Regras de **negócio** ou **Funcionalidade**.

- É nela que ficam as funções e regras de todo o negócio. Inexiste uma interface para o usuário e seus dados são voláteis, ou seja, para que algum dado seja mantido deve ser utilizada a camada de dados.

Camada de dados

- A terceira camada é definida como o **repositório** das informações e as **classes** que a manipulam.

- Esta camada recebe as **requisições** da camada de negócios e seus **métodos** executam essas requisições em um **banco de dados**.

- Alterando o banco de dados alteraria apenas as classes da camada de dados, e o restante das camadas não seriam afetados por essa alteração.

Vantagens e Desvantagens

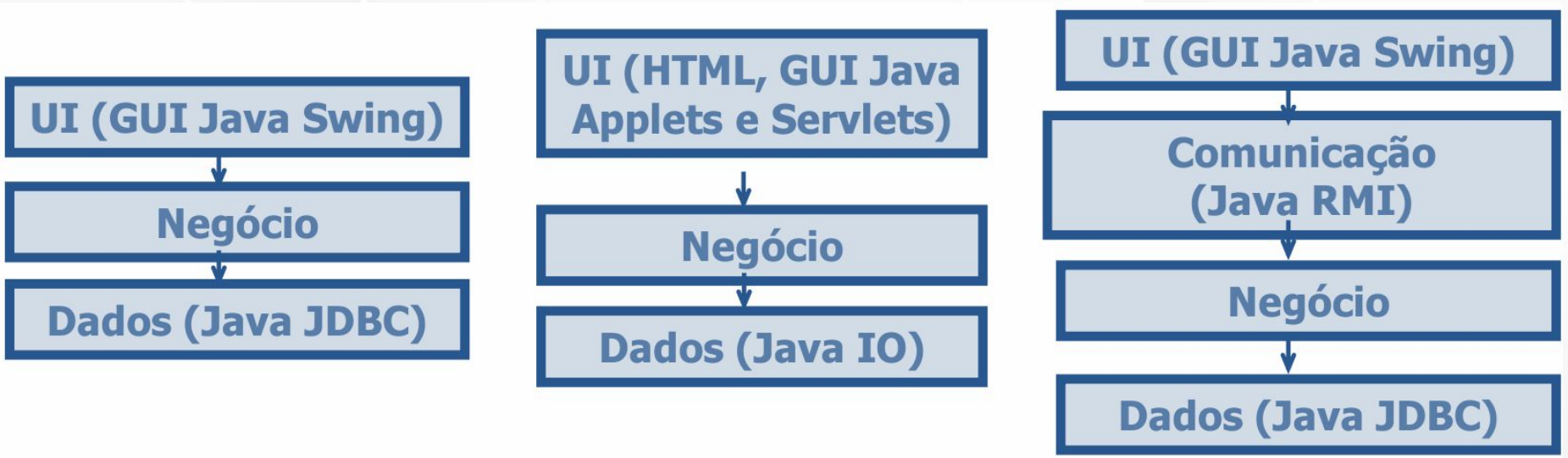
Vantagens:

- **Separação de código** relativo a interface com o usuário (UI), comunicação, negócio e dados.
- **Permite a mudança de implementação de uma camada sem afetar a outra**, desde que a interface entre as mesmas seja mantida.
- Possibilita que uma camada trabalhe com diferentes versões de outra camada.

Desvantagem:

- Aumento no número de classes existentes no sistema.

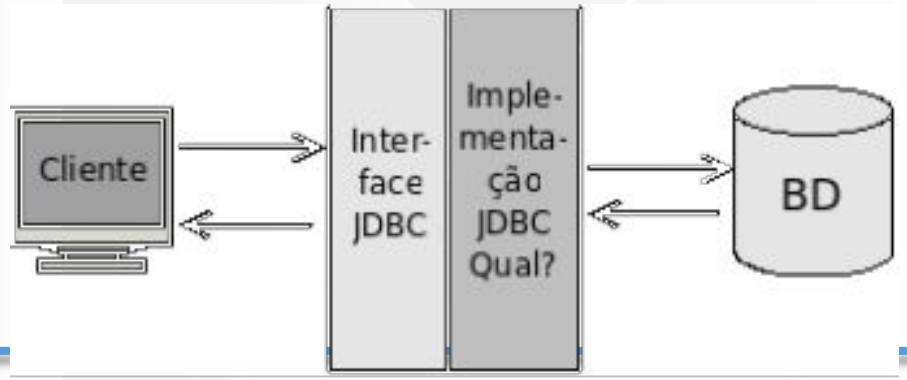
Exemplos de Modelo 3 camadas em Java



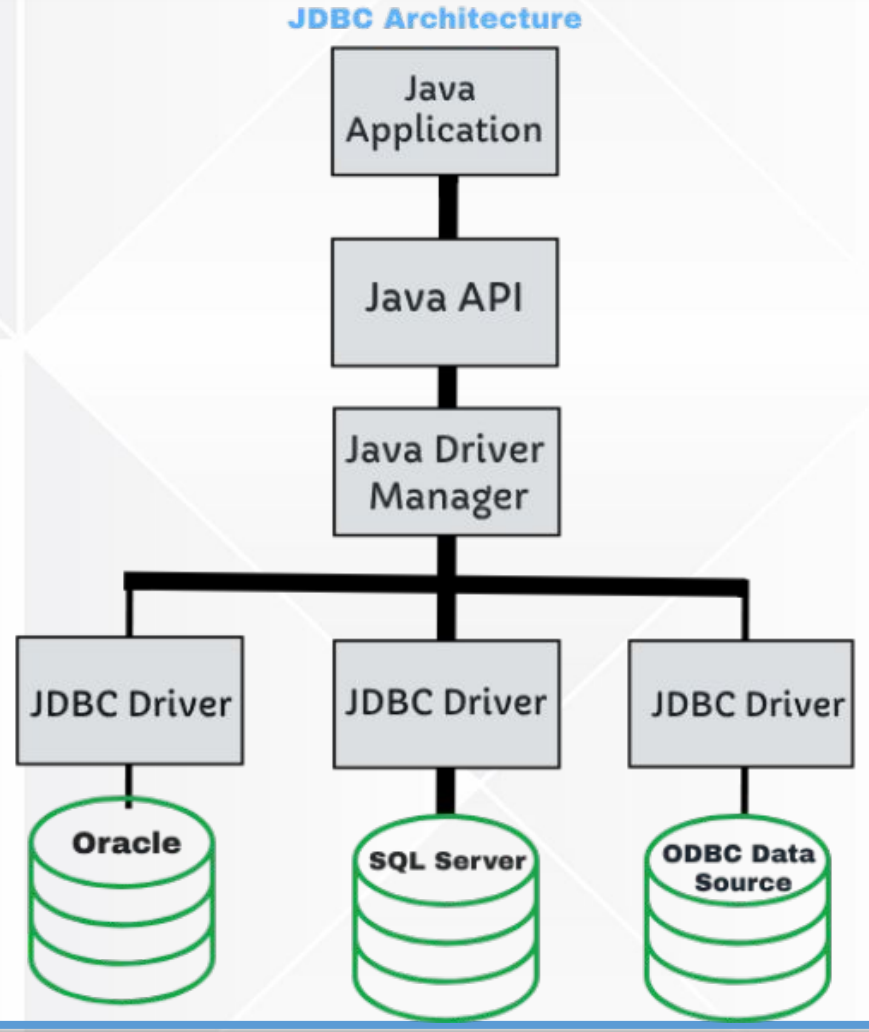
Persistência de dados

Persistência de dados: JDBC

- O processo de armazenamento de dados é também chamado de **persistência**.
- A biblioteca de persistência em banco de dados relacionais do Java é chamada **JDBC**, e também existem diversas ferramentas do tipo **ORM** (*Object Relational Mapping*) que facilitam bastante o uso do JDBC.
- Conectar-se a um banco de dados com Java é feito de maneira elegante. Para evitar que cada banco tenha a sua própria API e conjunto de classes e métodos, temos um único conjunto de interfaces muito bem definidas que devem ser implementadas. Esse conjunto de interfaces fica dentro do pacote java.sql e nos referiremos a ela como **JDBC**.



Persistência de dados: JDBC - arquitetura



Persistência de dados: JDBC

- Entre as diversas interfaces deste pacote, existe a interface **Connection** que define métodos para executar uma **query** (como um insert e select), comitar transação e fechar a conexão, entre outros. Caso queiramos trabalhar com o **MySQL**, precisamos de classes concretas que implementem essas interfaces do pacote j**ava.sql**.

- Esse conjunto de classes concretas é quem fará a ponte entre o código cliente que usa a API JDBC e o banco de dados. São essas classes que sabem se comunicar através do protocolo proprietário do banco de dados. Esse conjunto de classes recebe o nome de **driver**. Todos os principais bancos de dados do mercado possuem **drivers JDBC** para que você possa utilizá-los com Java.

Persistência de dados: JDBC - componentes

- **Driver JDBC**: É um intermediário entre a aplicação Java e o banco de dados. O driver converte as chamadas JDBC em comandos específicos do banco de dados.
- **Connection**: Estabelece a conexão com o banco de dados.
- **Statement**: Permite executar comandos SQL no banco de dados.
- **ResultSet**: Contém os resultados de uma consulta SQL.
- **SQLException**: Exceções que ocorrem ao interagir com o banco de dados.

Persistência de dados: JDBC - passos

Passos básicos para usar JDBC:

1. Carregar o Driver JDBC.
2. Estabelecer uma Conexão com o Banco de Dados.
3. Criar um Statement ou PreparedStatement.
4. Executar o SQL (consultas, inserções, atualizações, etc.).
5. Processar o Resultado (caso haja).
6. Fechar as Conexões.

Persistência de dados: DAO

- O padrão **Data Access Object** (DAO) é um padrão introduzido no ambiente JEE para simplificar e desacoplar a interação das aplicações Java com a API JDBC.

- O objeto **DAO** é responsável por operar o mecanismo de persistência em nome da aplicação tipicamente executando os quatro tipos de operações - **Criar**, **Recuperar** , **Alterar**, **Apagar** - conhecidas pela sigla **CRUD** - do inglês Create, Retrieve, Update , Delete.

Exemplo DAO (1)

- Uma aplicação prática das interfaces está na criação do CRUD (acrónimo de **C**reate, **R**ead, **U**pdate e **D**elete na língua Inglesa)

```
import java.util.List;

public interface BasicoDAO {

    public void salvar(Object bean);
    public void atualizar(Object bean);
    public void deletar(int id);
    public Object getById(int id);
    public List<Object>  getAll();
}
```

Exemplo DAO (2)

```
import java.util.List;
public class FuncionarioDAO implements BasicoDAO {
    @Override
    public void salvar(Object bean) {      }
    @Override
   public void atualizar(Object bean) {     }
    @Override
    public void deletar(int id) {     }
    @Override
    public Object getById(int id) {     }
    @Override
    public List<Object> getAll() {      }
      //Método a parte criado e implementado pelo próprio programador
    public void calcularSalario(){
    }
}
```

Sistema Consultório Psicologia

Objetivo

- Desenvolver um **sistema** de **gerenciamento** de um **consultório** de **psicologia** para uma clínica fictícia.

  - Criação de um banco de dados que armazene prontuários de pacientes e o histórico das sessões realizadas.
  - Desenvolvimento de uma interface de usuário desktop para gerenciar os dados, permitindo a inserção, alteração e remoção de registros.
  - Permitir recuperar informações de pacientes de acordo a critérios de busca.

Requisitos (1)

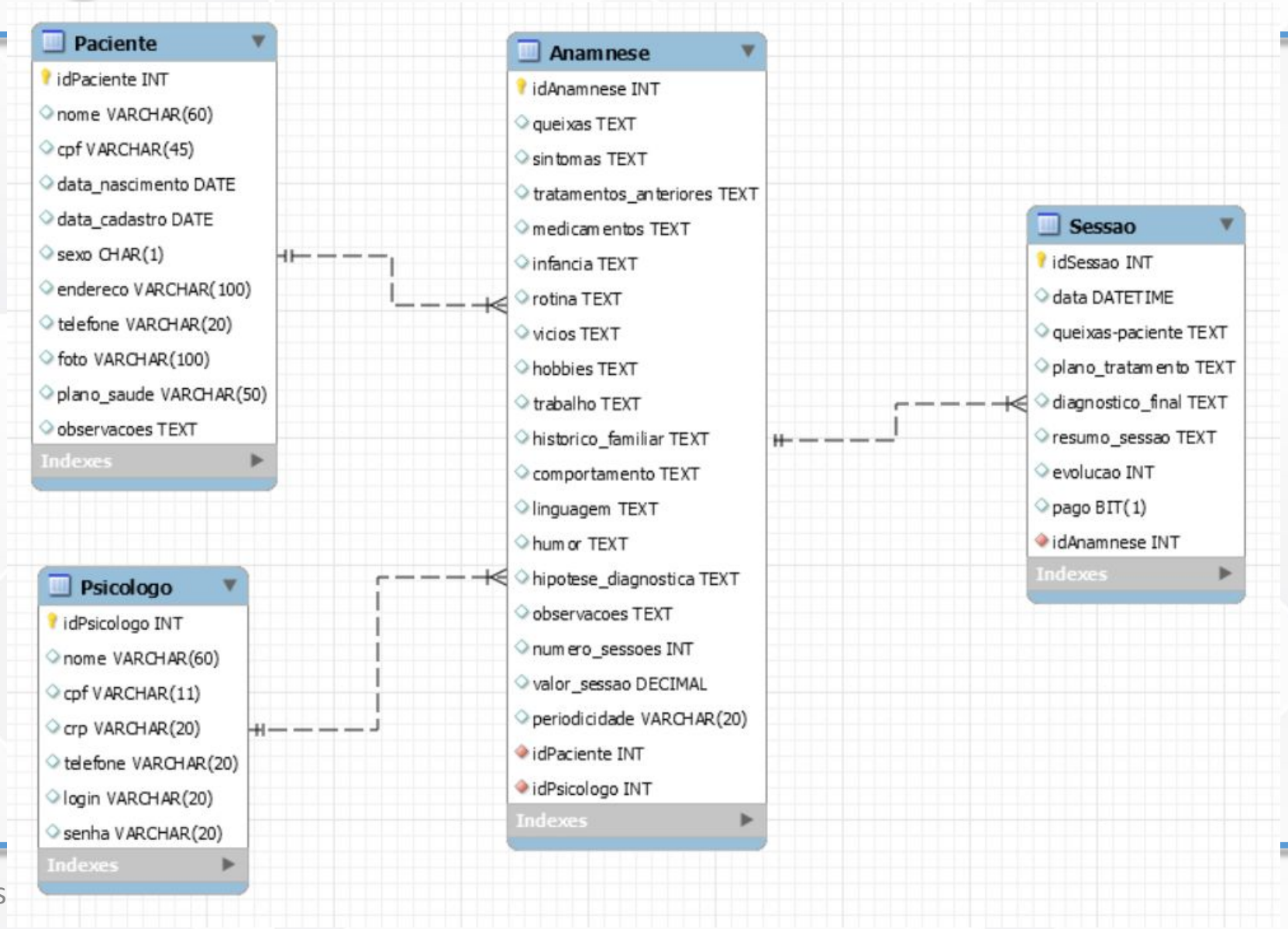
- Criação de um banco de dados que armazene prontuários de pacientes e o histórico das sessões realizadas.
- Os dados a serem mantidos são:
  - Pacientes: cpf, nome, data de nascimento, sexo, endereço, telefone, fotografia e plano de saúde.
  - Psicólogos: nome, cpf, crp, telefone, usuário e senha.
  - Anamnese: representa uma ficha resultado de uma entrevista inicial com o paciente, inicialmente deverá ter dados como queixas, sintomas, tratamentos anteriores, medicamentos, infância, rotina, vícios, hobbies, trabalho e histórico familiar. O especialista poderá adicionar dados relacionados ao comportamento, linguagem, hipótese diagnóstica, observações, número de seções, valor da seção e periodicidade das sessões
  - Sessão: data, queixas, plano tratamento, diagnóstico, resumo, evolução, estado de pagamento.

Requisitos (2)

- Desenvolvimento de uma interface de usuário desktop para gerenciar os dados, permitindo a inserção, alteração e remoção de registros.
  - Login de usuário, apenas usuários cadastrados (psicólogos) poderão acessar o sistema
  - Cadastro de pacientes
  - Cadastro de psicólogos
  - Registro de Anamnese e sessões

- Permitir recuperar informações de pacientes de acordo a critérios de busca.

Modelo lógico do banco de dados



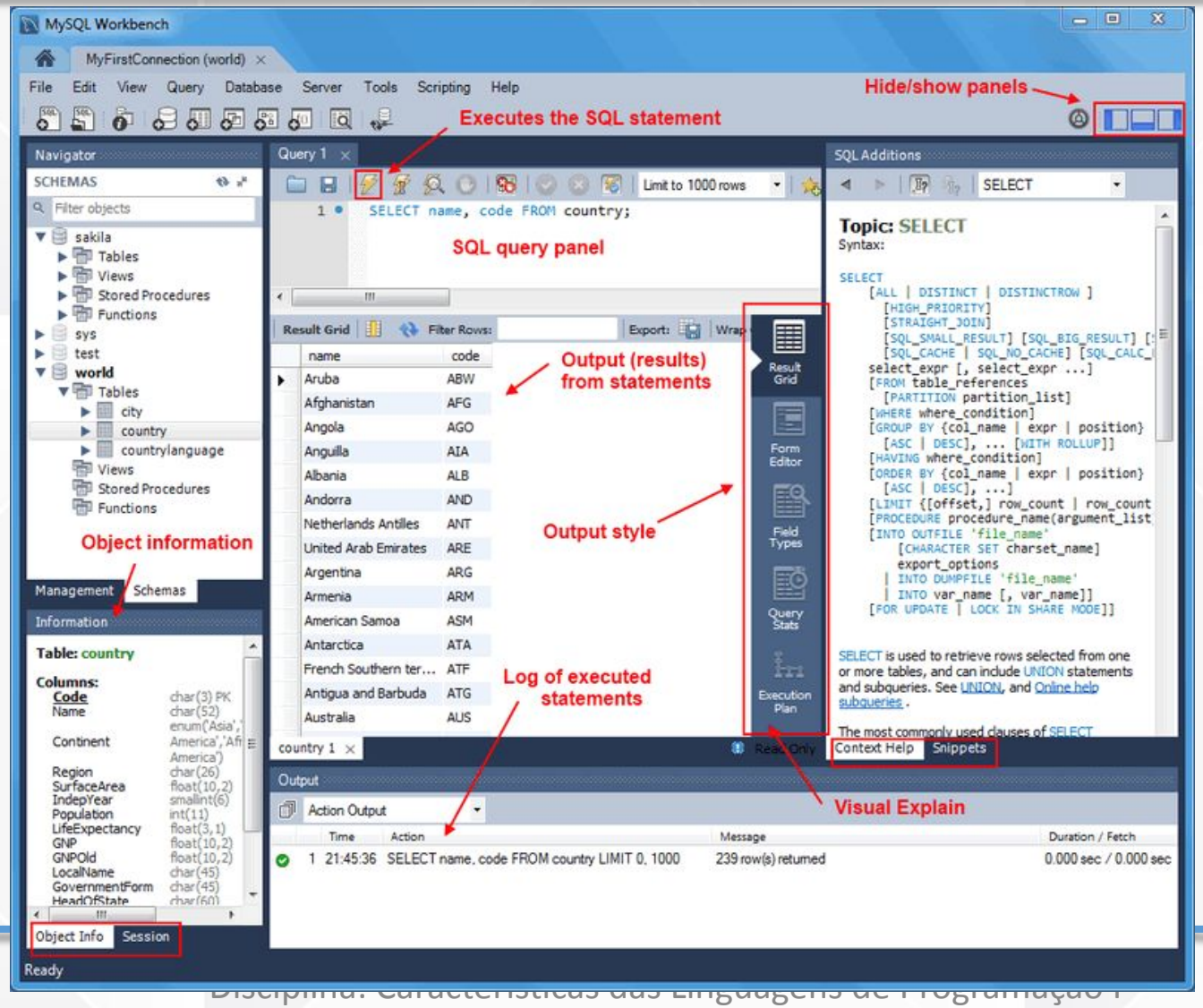
Criação do banco de dados

- O **MySQL Workbench** é uma ferramenta visual integrada, desenvolvida pela Oracle, usada para o projeto, desenvolvimento e administração de bancos de dados MySQL. Ele oferece uma interface gráfica amigável que facilita tarefas como:
    - Modelagem de banco de dados: criação e edição de diagramas ER (entidade-relacionamento).
    - Execução de consultas SQL: permite criar, testar e executar comandos SQL.
    - Administração do servidor: gestão de usuários, controle de permissões, backup/restauração e monitoramento do desempenho do servidor.
    - Migração de dados: ajuda a migrar esquemas e dados de outros bancos de dados (como PostgreSQL ou Microsoft SQL Server) para o MySQL.

Instalação MySQL Workbench

- Para instalar o **MySQL Workbench** siga as instruções disponibilizadas nos link abaixo:
  - Link de descarga: https://dev.mysql.com/downloads/installer/
  - Tutorial online MySQL: https://www.w3schools.com/mysql/

Interface MySQL Workbench



Criação do banco de dados

- Abra o MySQL Workbench e crie um banco de dados chamado de **DBConsultorio.** A seguir crie a tabela **Paciente** usando o script a seguir:

```
create database DBConsultorio;

use DBConsultorio;
```

Criação tabelas

- A seguir crie a tabela **Paciente** usando o script a seguir:

```
create table if not exists paciente (
  idPaciente int unsigned not null auto_increment primary key,
  nome varchar(60),
  cpf varchar(45),
  data nascimento DATE,
  sexo CHAR(1),
  endereco varchar(100),
  telefone varchar(20),
  foto varchar(100),
  plano saude varchar(50),
  observacoes text,
  data cadastro date
)ENGINE=MyISAM  DEFAULT CHARSET=utf8;
```

Criação tabelas

- Adicione a tabela **Psicologo** usando o script a seguir:

```
create table if not exists psicologo (
  idPsicologo int unsigned not null auto_increment primary key,
  nome varchar(60),
  cpf varchar(14),
  crp varchar(20),
  telefone varchar(20),
  foto varchar(100),
  login varchar(20) unique,
  senha varchar(20)
)ENGINE=MyISAM  DEFAULT CHARSET=utf8;
```

Criação tabelas

- Adicione a tabela **Anamnese** usando o script a seguir:

```
create table if not exists anamnese (
  idAnamnese int unsigned not null auto_increment primary key,
  queixas text,
  sintomas text,
  tratamentos_anteriores text,
  medicamentos text,
  infancia text,
  rotina text,
  vicios text,
  hobbies text,
  trabalho text,
  historico_familiar text,
  comportamento text,
  linguagem text,
  humor text,
  hipotese diagnostica text,
  observacoes text,
  numero sessoes int,
  valor sessao decimal,
  periodicidade varchar(20),
  idPaciente int not null,
  idPsicologo int not null,
  foreign key (idpaciente)references paciente(idpaciente),
  foreign key (idPsicologo)references psicologo(idPsicologo)
)ENGINE=MyISAM  DEFAULT CHARSET=utf8;
```

Criação tabelas

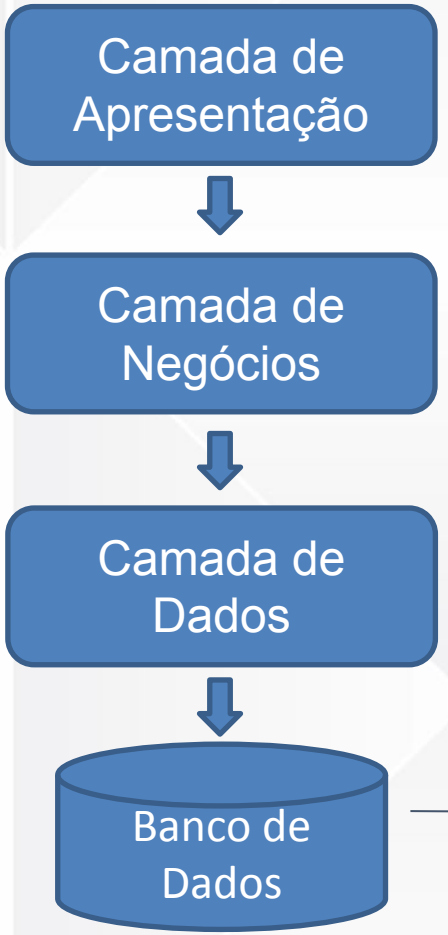
- Adicione a tabela **Sessao** usando o script a seguir:

```
create table if not exists sessao (
  idSessao int unsigned not null auto_increment primary key,
  data datetime,
  queixas_paciente text,
  plano tratamento text,
  diagnostico_final text,
  resumo_sessao text,
  evolucao int,
  pago bit(1),
  idAnamnese int,
  foreign key (idAnamnese)references anamnese(idAnamnese)
)ENGINE=MyISAM  DEFAULT CHARSET=utf8
```

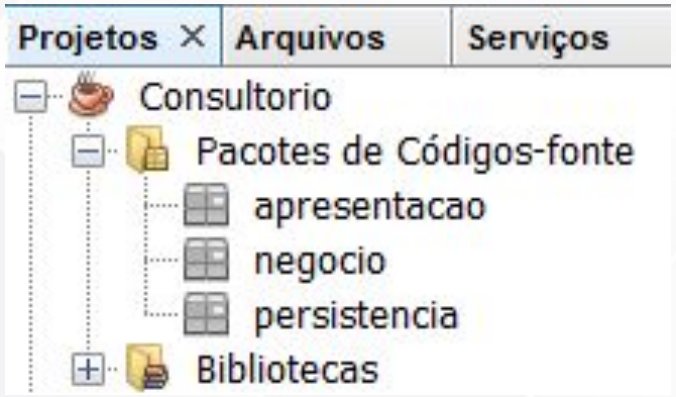
Interface principal

Estrutura do projeto

- Criar um novo projeto no netbeans vazio chamado **Consultorio**
- Criar 3 pacotes
  - persistencia
  - negocio
  - apresentacao



DBConsultorio no MySQL



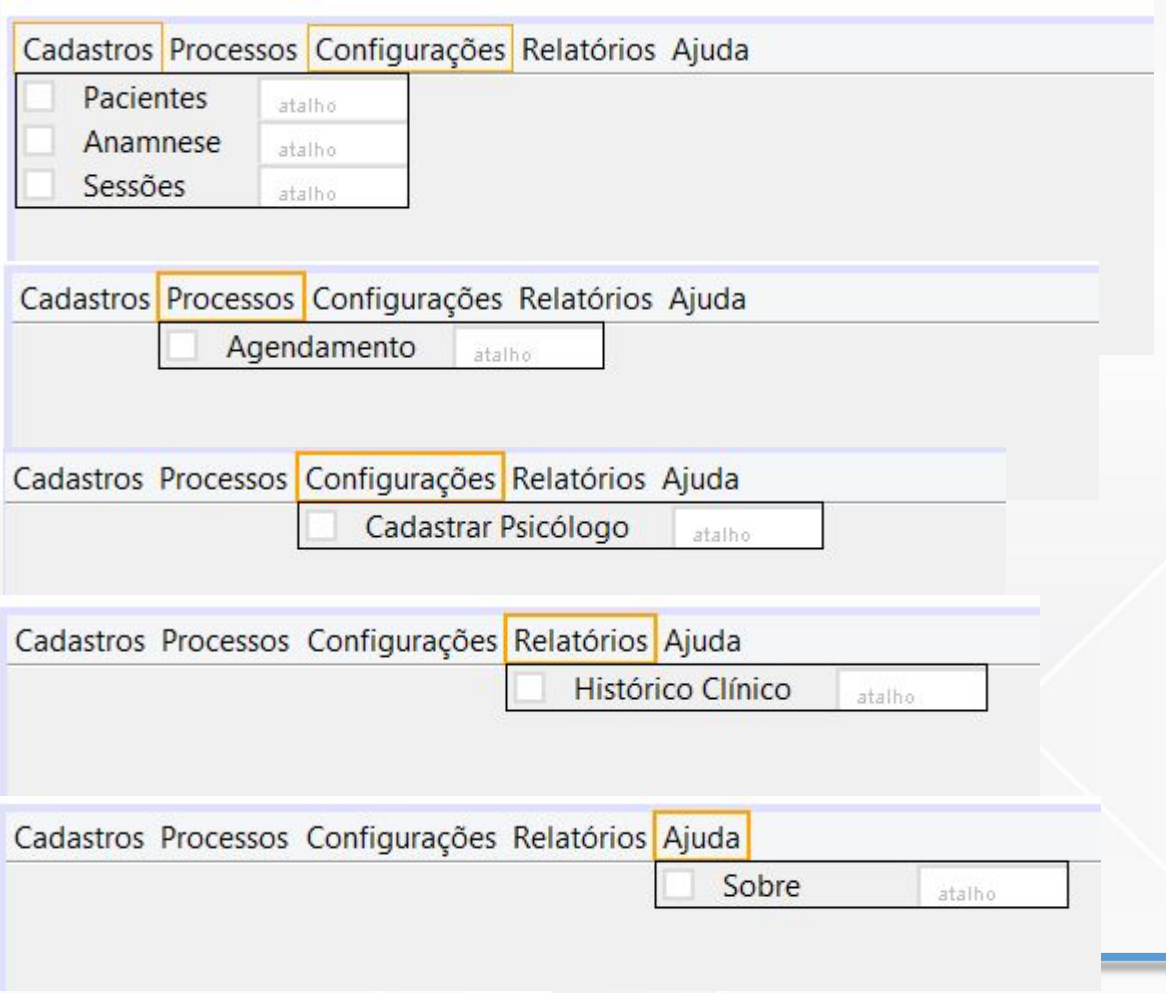
Criação do formulário principal (1)

- No pacote apresentação, criar um novo **JFrame,** com o nome **fmPrincipal**
- Na propriedade **Title** do formulário principal colocar um título como mostrado na figura.

- É importante que a propriedade **defaultCloseOperation** esteja em **EXIT_ON_CLOSE**

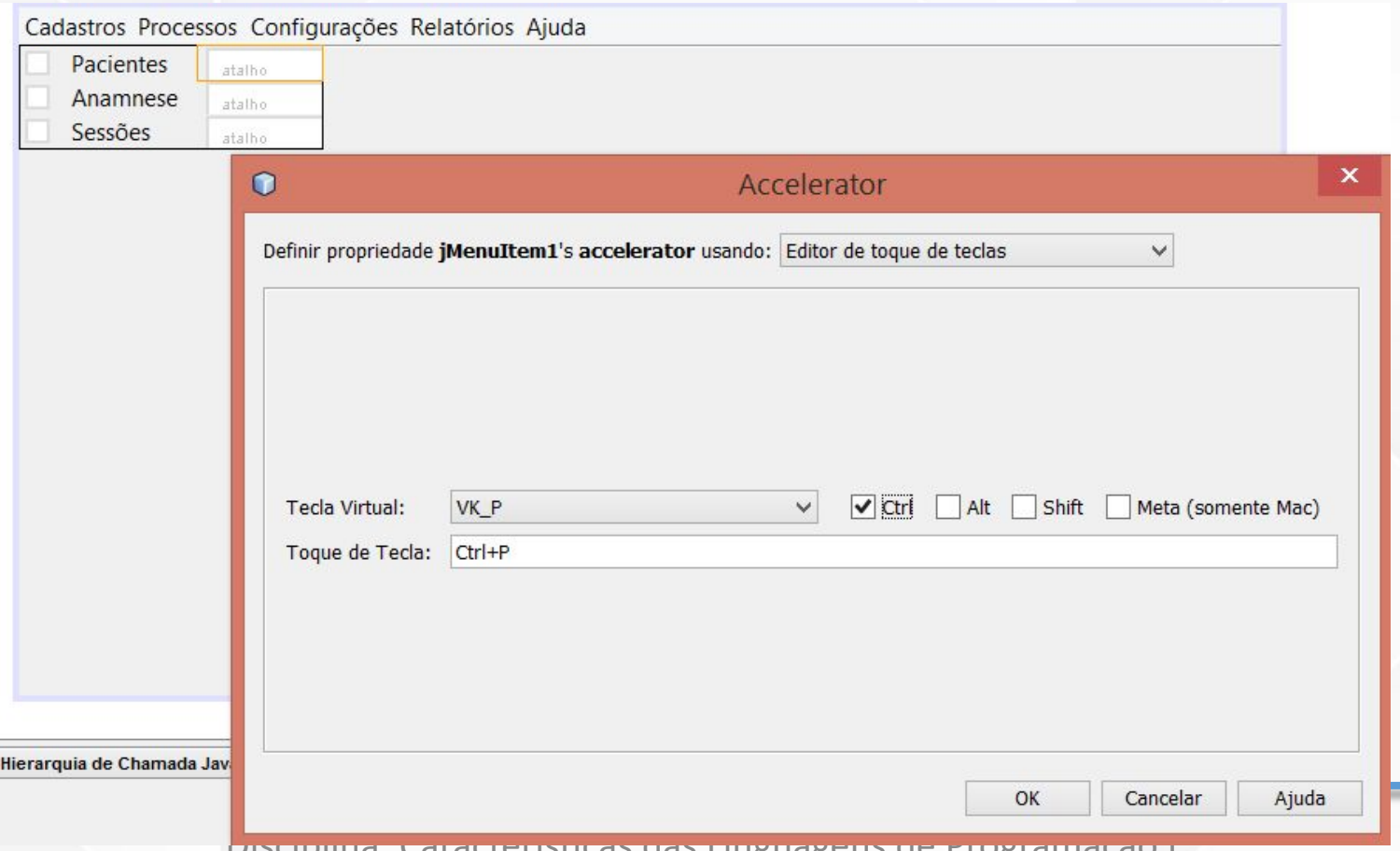
Criação do formulário principal (2)

- No formulário **fmPrincipal** adicionaremos Menus até obter a seguinte estrutura:
  - Cadastros
    - Pacientes
    - Anamnese
    - Sessões
  - Processos
    - Agendamento
  - Configurações
    - Cadastrar Psicólogo
  - Relatórios
    - Histórico Clínico
  - Ajuda
    - Sobre



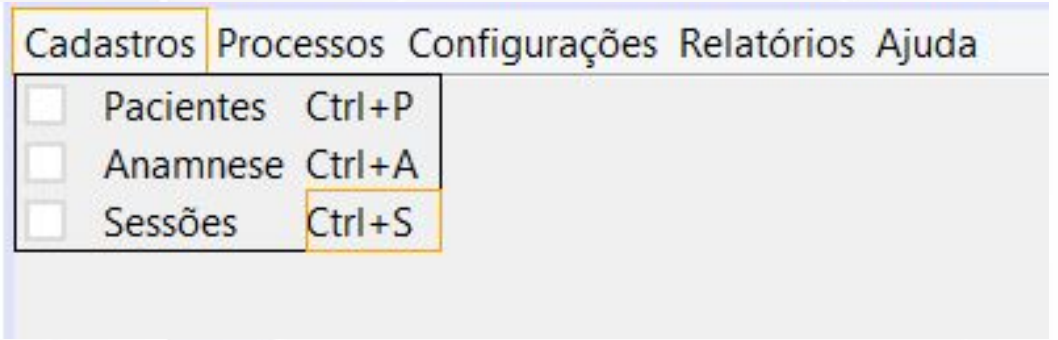
Criação do formulário principal (3)

- Criar atalhos para cada item de Menu (Clicar 2 vezes no item atalho e escolher uma combinação de teclas)



Criação do formulário principal (4)

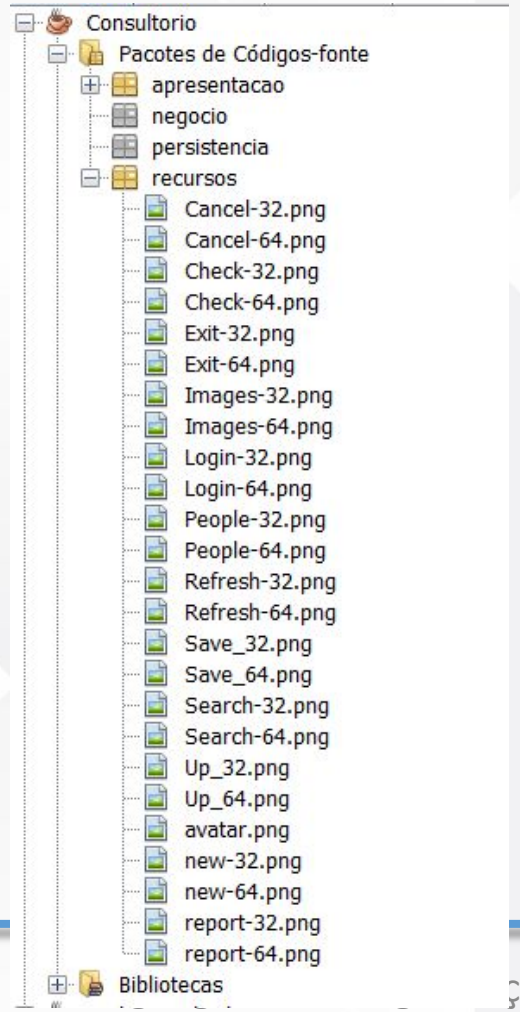
- Resultado dos atalhos



- Repetir o processo para cada item de menu

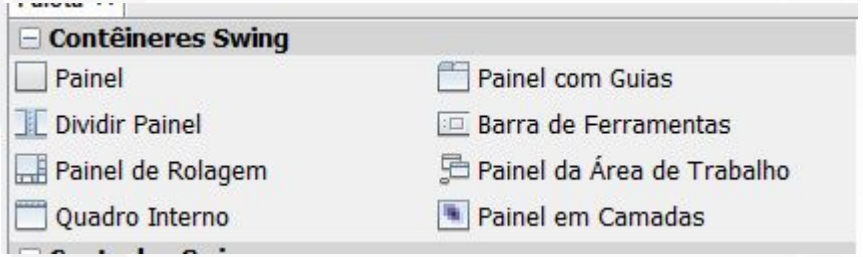
Criação do formulário principal (5)

- Vamos criar um novo pacote chamado recursos onde armazenaremos as Figuras e ícones utilizados pelo nosso programa.
- Arrastrar nossas imagens a esse pacote



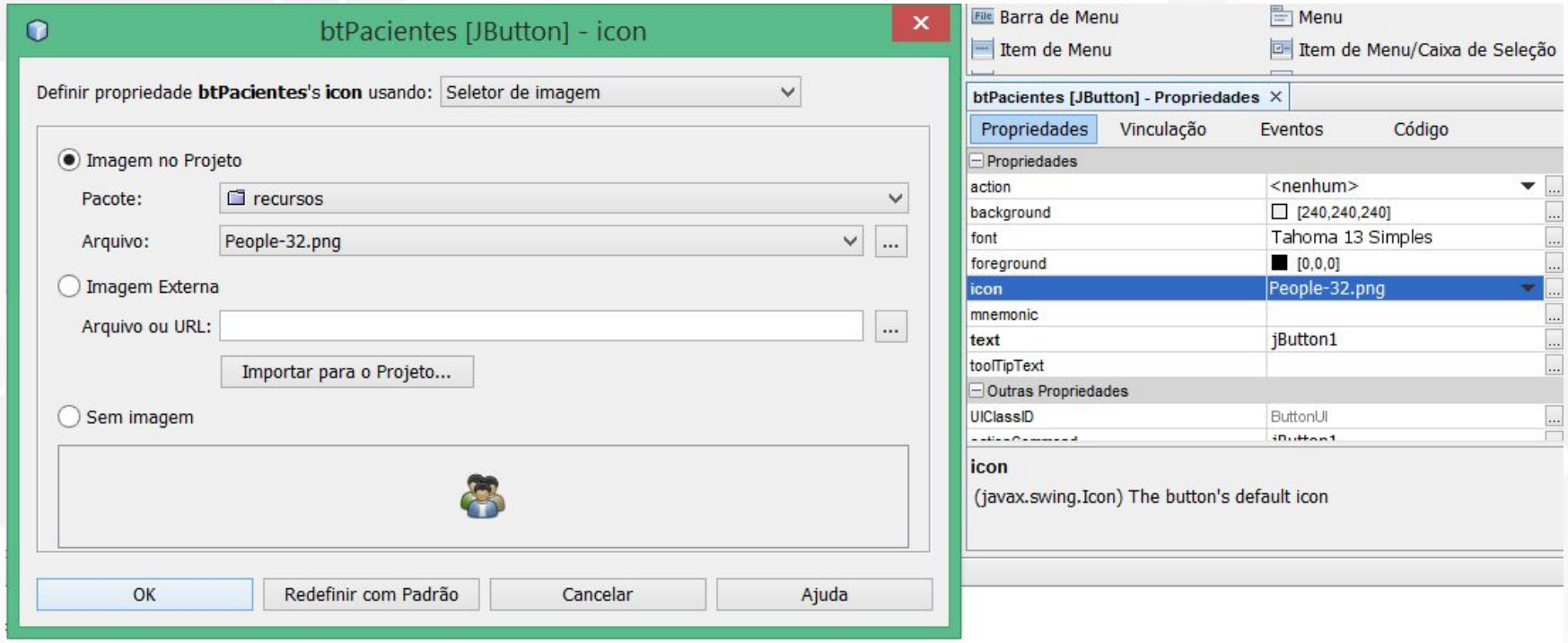
Criação do formulário principal (6)

- Agora vamos adicionar um **JToolBar** ao formulário Principal para permitir exibir ações ou controles usados com maior frequência.



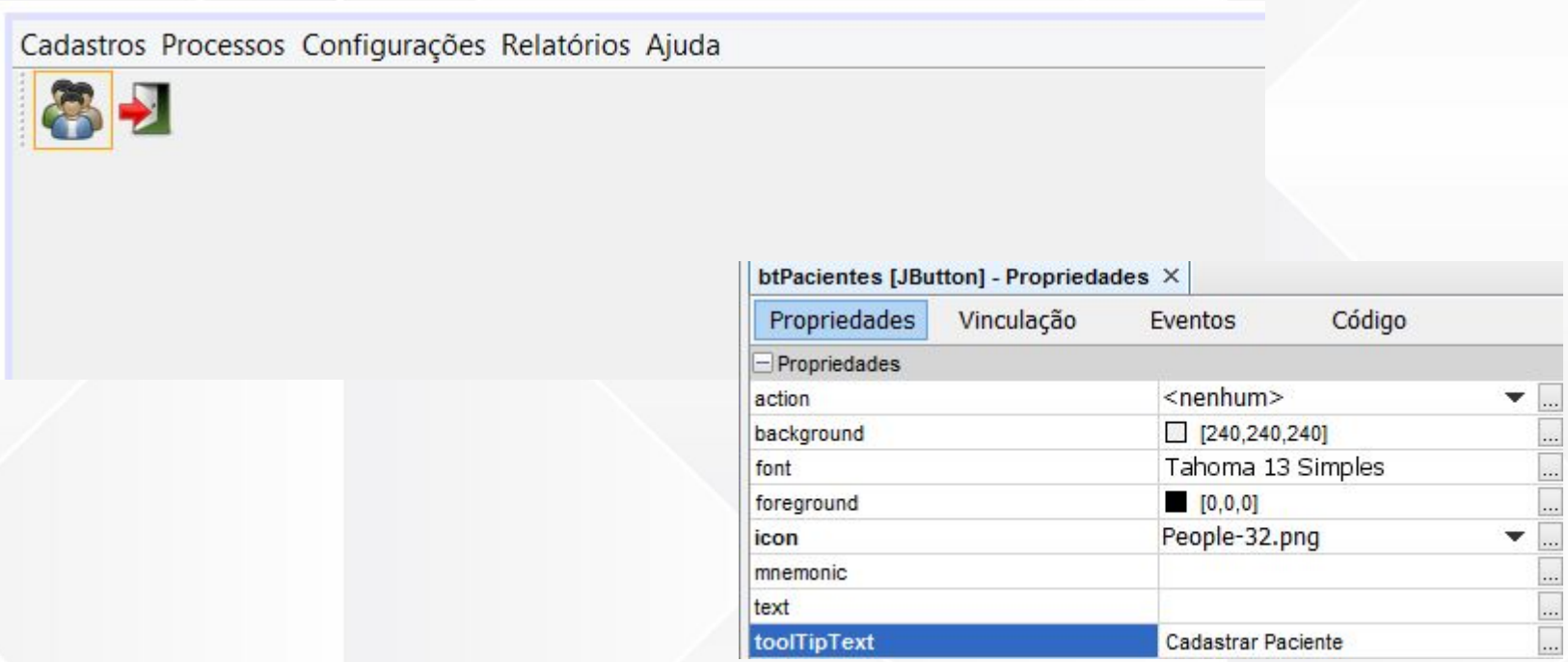
Criação do formulário principal (7)

- Adicionar 2 botões no JToolBar com nomes de variáveis “btPaciente” e “btSair” respectivamente.
- Adicionar ícones aos botões como mostrado abaixo:



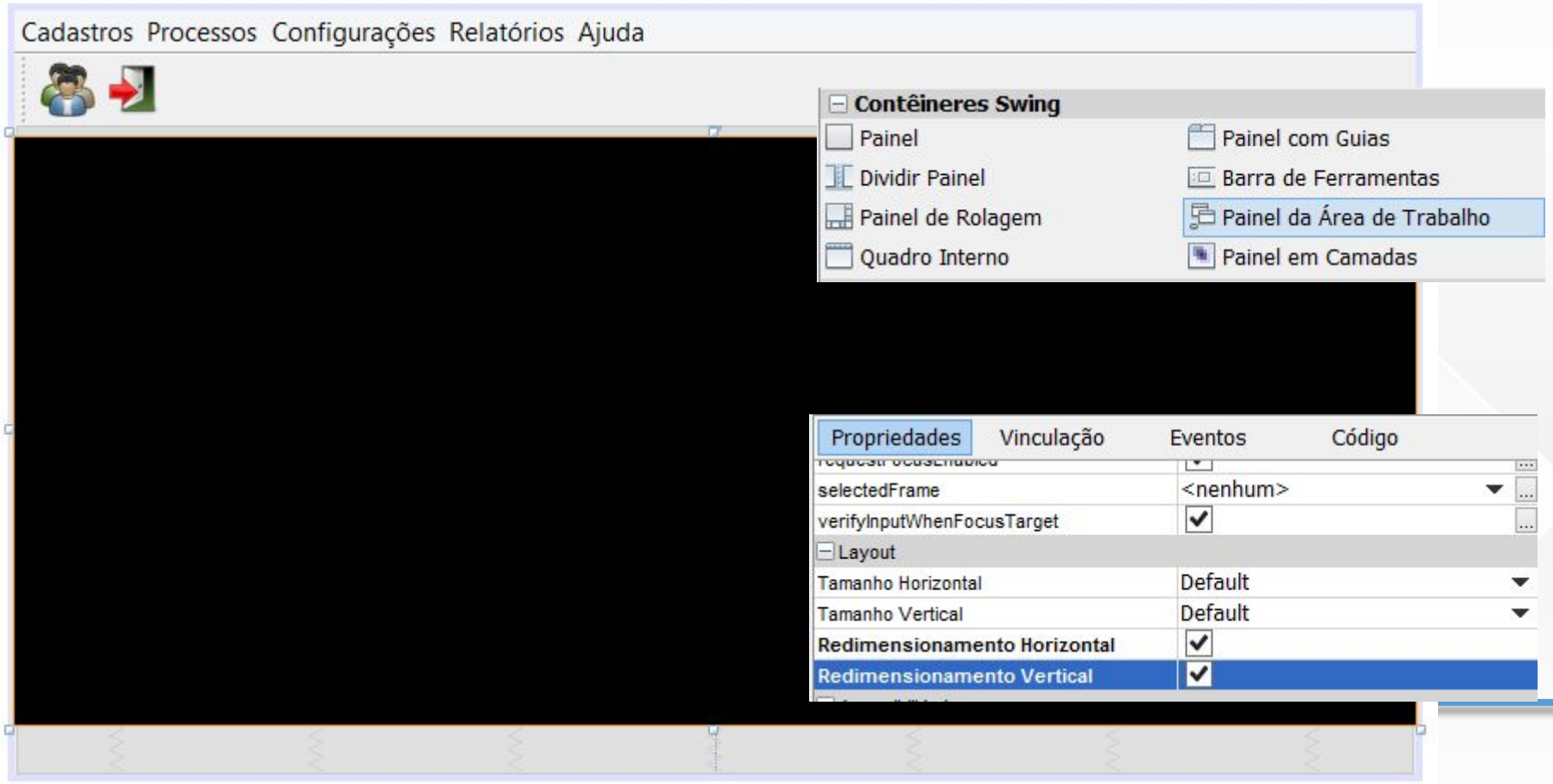
Criação do formulário principal (8)

- Remover o texto da propriedade Text e adicionar um valor à propriedade **toolTipText**



Criação do formulário principal (9)

- Adicionar o componente **JDesktopPane** para permitir criar um container de Formulários. Ou seja servirá como um container para criar uma interface com múltiplos documentos ou uma área de trabalho virtual.
- Marcar a propriedade Redimensionamento Horizontal e Vertical



Criação do formulário principal (10)

- Adicionar o componente **JPanel** na parte inferior do formulário principal para simular uma barra de estado (StatusBar)

- Adicionar 2 componentes JLabel sobre a barra de estatus criada, colocar o nome de variável para “laUsuario” e “laData” a modificar a propriedade Text como mostrado na Figura abaixo.

Criação do formulário principal (11)

- Executando o programa

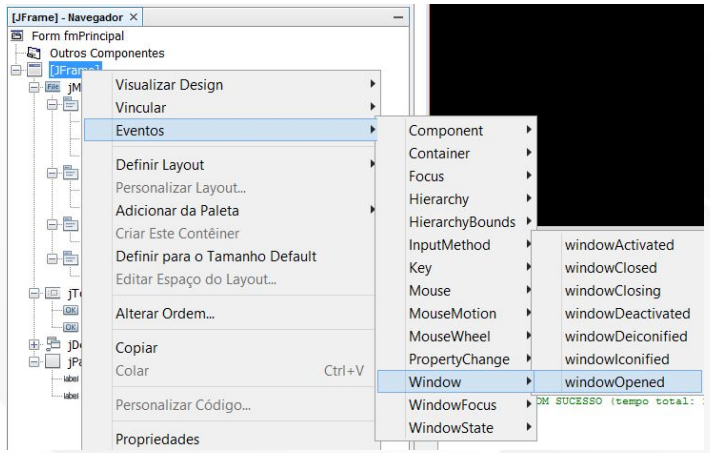
Criação do formulário principal (12)

- Adicionando eventos
  - No evento **ActionPerformed** do Botão btSair adicionar o código abaixo, para permitir sair da aplicação após confirmação do usuário

```
private void btSairActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int valor = JOptionPane.showConfirmDialog(null, "Tem certeza que deseja sair?","Sistema de Prontuários Médicos",1);
    if(valor==0){
        System.exit(0); // Finalizar o programa
    }
}
```

Criação do formulário principal (13)

- Adicionando eventos
  - No evento **WindowsOpened** do formulário principal adicionar o código abaixo para mostrar a hora atual do sistema.




```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    // TODO add your handling code here:
    DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    //obter data atual com Date()
    Date date = new Date();
    laData.setText(dateFormat.format(date));
}
```

Importar as seguintes classes:

```
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
```

- Executar o programa, o que acontece?

Criação do formulário principal (14)

- Para melhorar o código anterior criaremos o método mostrarData().

```
private String  mostrarData(){
     DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    //obter data atual com Date()
    Date date = new Date();
    return dateFormat.format(date);
}
```

Importar as seguintes classes:

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

- Modificaremos o código do evento **WindowsOpenend** para criar um **Timer** que gere eventos a cada segundo:

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    // TODO add your handling code here:
    laData.setText( mostrarData());

    ActionListener updater = new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent event) {
            laData.setText( mostrarData());
        }
    };
    Timer timer = new Timer(1000, updater);//Chamamos ao método updater a cada 1000 milisegundos
    timer.start();
}
```

- Executar novamente o programa.

Criação do formulário principal (15)

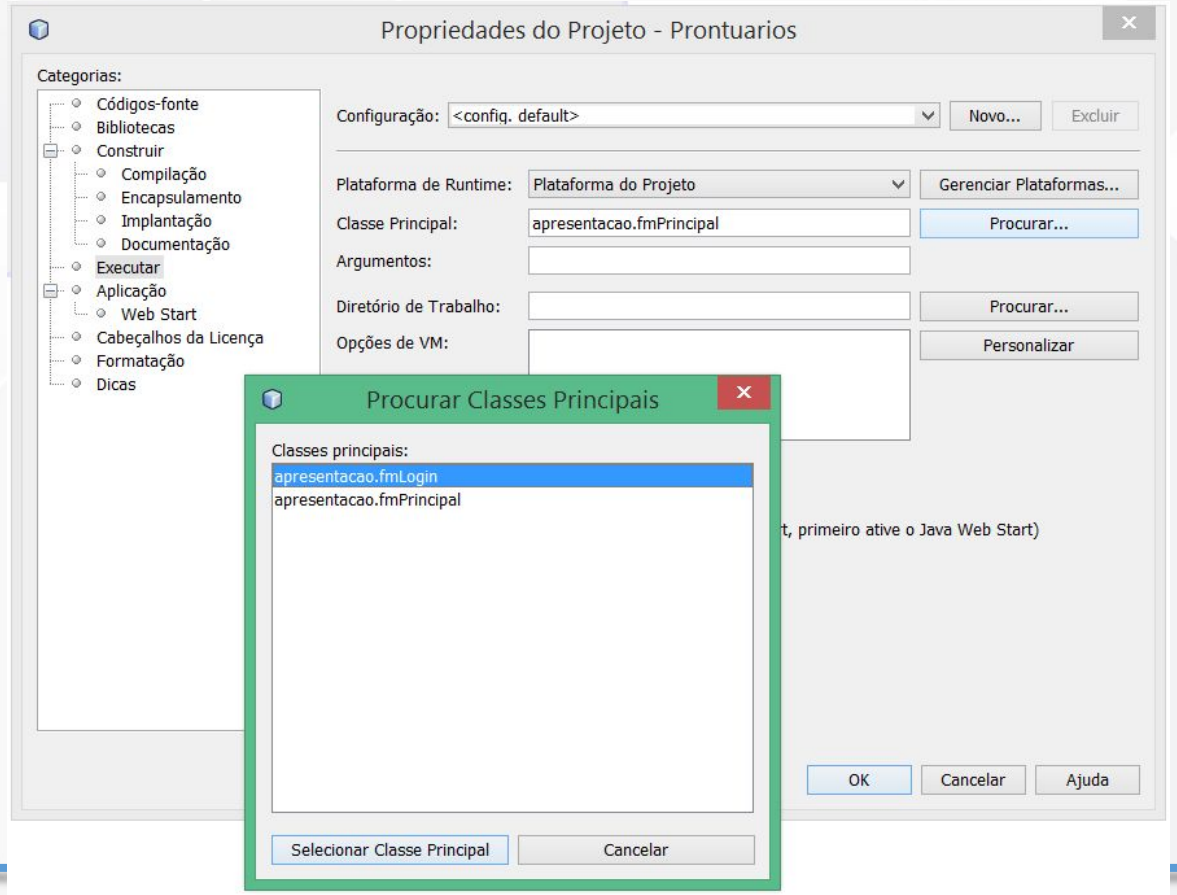
- Para que o formulário principal inicie sempre maximizado modificaremos o código do evento **WindowsOpenend** adicionando a linha abaixo.

```
this.setExtendedState(this.getExtendedState()|JFrame.MAXIMIZED_BOTH );
```

Interface Login

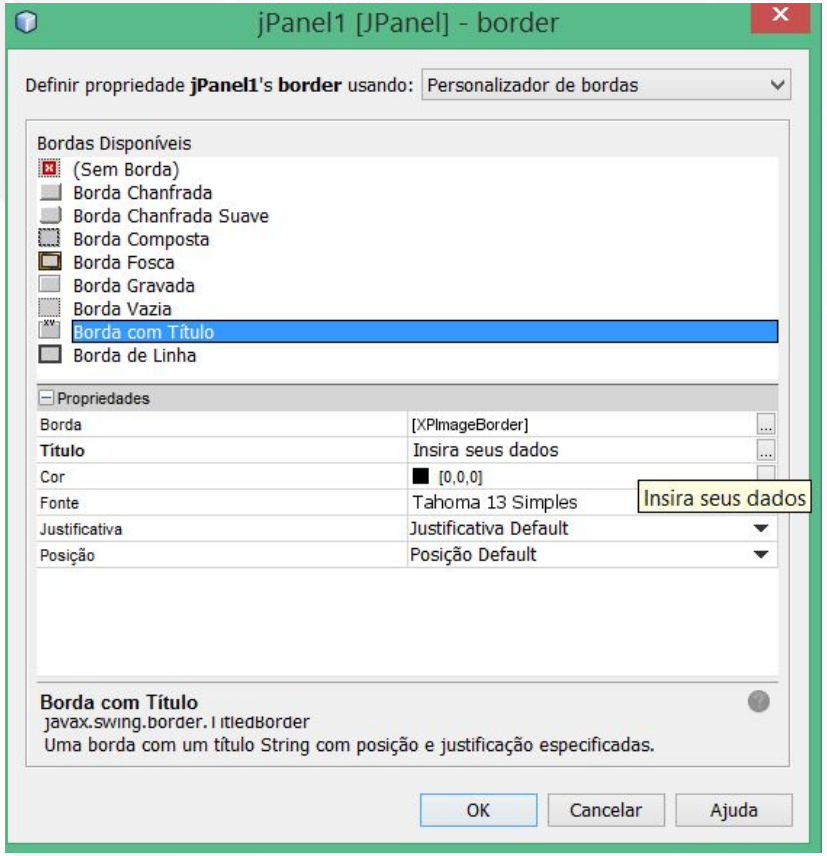
Criação do formulário Login (1)

- No pacote apresentação, criar um novo **JFrame,** com o nome **fmLogin** e fazemos que seja o formulário de início em nossa aplicação (ir a propriedades projeto)



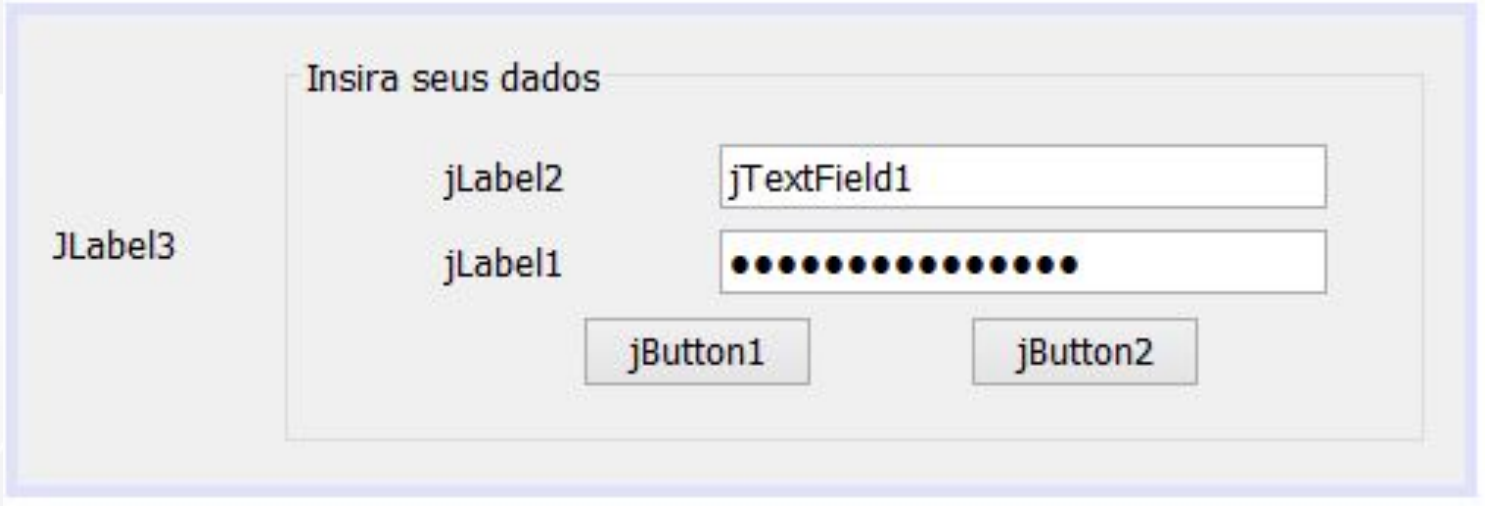
Criação do formulário Login (2)

- Arrastre 1 pane sobre o formulário fmLogin e na propriedade **border** escolha “Borda Com título” e coloque um título como a Figura abaixo



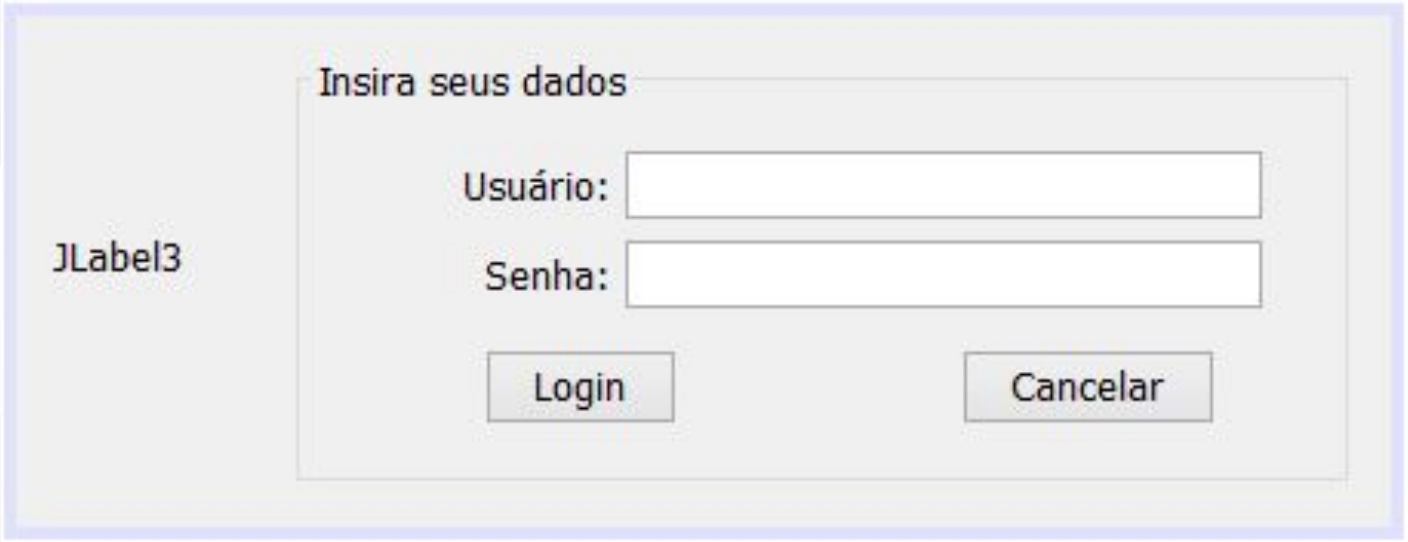
Criação do formulário Login (3)

- Arrastre componentes **JLabel**, **JTextField**, **JTextPasswordField** e **JButton** sobre o formulário fmLogin como a Figura abaixo



Criação do formulário Login (4)

- Modifique a propriedade **Text** dos componentes como na Figura e coloque os **nomes de variáveis** txtUsuario, txtSenha, btLogin, btCancelar



Criação do formulário Login (5)

- Modifique a propriedade **icon** dos componentes como JLabel3, btLogin e btCancelar como na Figura.
- Modifique a propriedade **Resizable** (para falso) do formulário fmLogin

Criação do formulário Login (6)

- No evento **ActioPerformed** do botão Login adicione o código abaixo

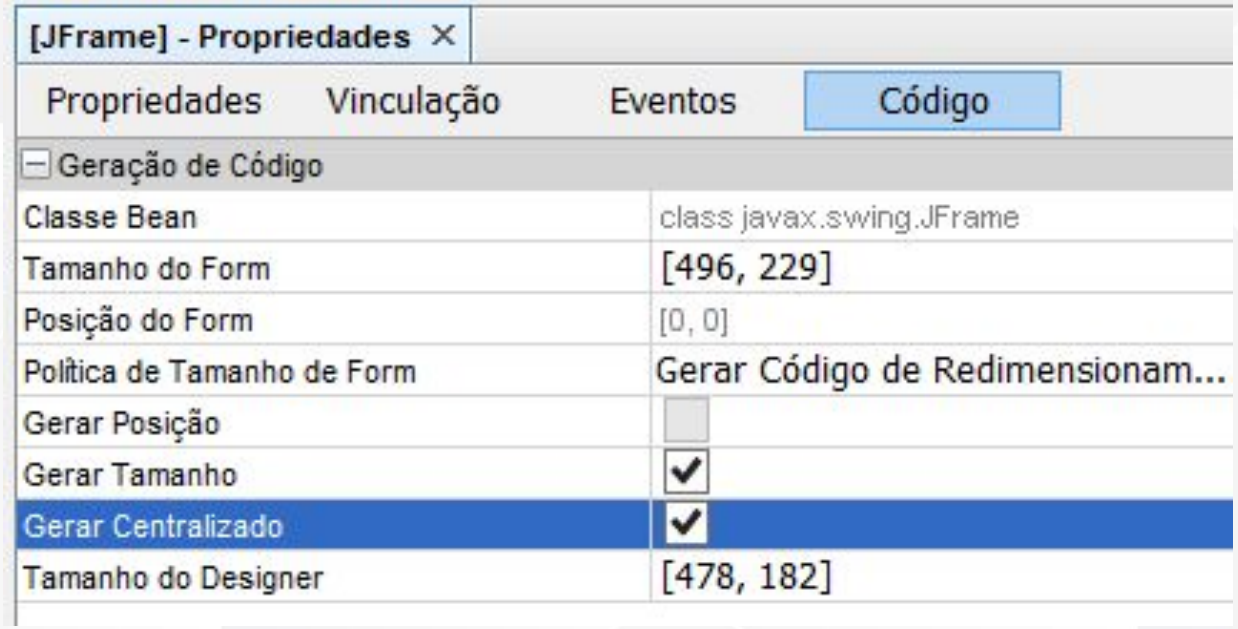
```
private void btLoginActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //Criamos um usuário e senha padrão (para teste)
    String usuarioDefault="user";
    String senhaDefault="123";

    //recuperamos os dados inseridos pelo usuário
    String usuarioDigitado=txtUsuario.getText();
    String senhaDigitada= new String(txtSenha.getPassword());

    //comparamos com a função equals
    if(usuarioDefault.equals(usuarioDigitado) && senhaDefault.equals(senhaDigitada)){
        //caso os dados confiram criamos uma instancia do formulário principal
        fmPrincipal principal=new fmPrincipal();
        principal.setVisible(true);
        principal.setExtendedState(JFrame.MAXIMIZED_BOTH);//mostramos maximizado
        this.dispose();//liberamos o formulário de login
    }
    else
        JOptionPane.showMessageDialog(null, "Usuário ou senha incorretos");
}
```

Criação do formulário Login (7)

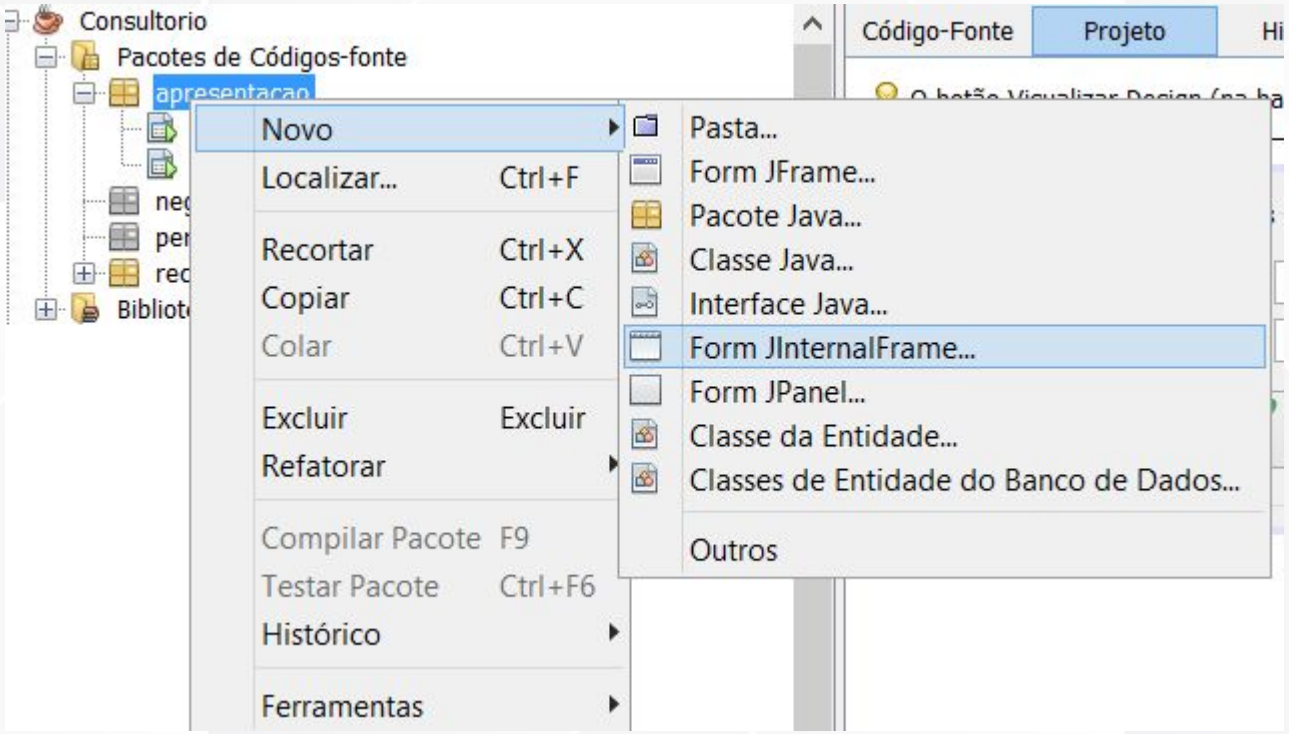
- Finalmente para centralizar o formulário marcar o campo Gerar Centralizado na seção Código da paleta de propriedades.



Interface Sobre

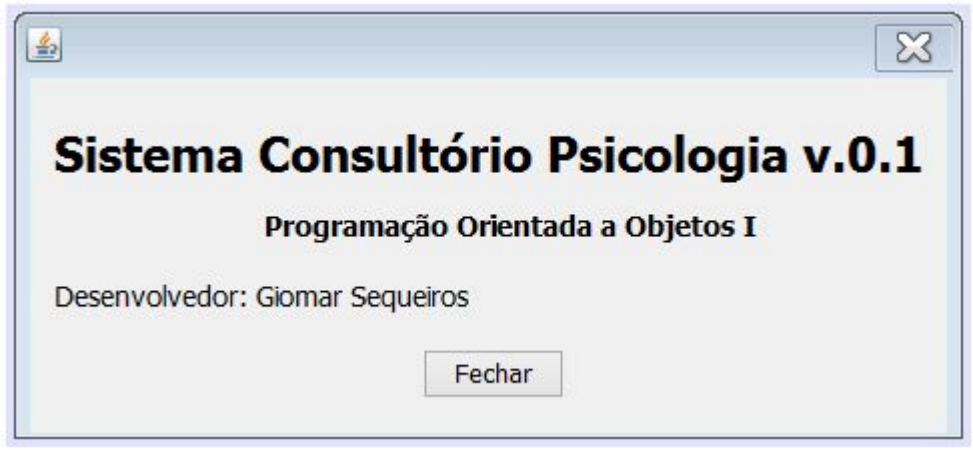
Criação do formulário Sobre (1)

- No pacote apresentação, criar um novo **JInternalFrame,** com o nome fmSobre



Criação do formulário Sobre (2)

- Adicionar controles **JLabel** como a Figura e botão com nome de variável btFechar.
- Modificar a propriedade closable do **JInternalFrame** para true



Criação do formulário Sobre (3)

- Adicionar o evento **ActionPerformed** no item de menu “Sobre” e escrever o código abaixo.

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    fmSobre sobre=new fmSobre();
    jDesktopPane1.add(sobre);//adicionar o formulário em nosso container
    sobre.setVisible(true);
}
```

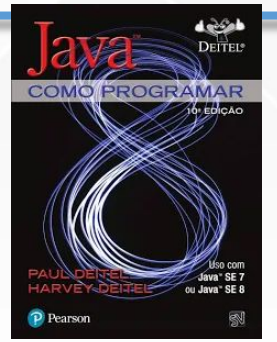
- No botão btFechar do formulário fmSobre, no evento **ActionPerformed** adicionar o código

```
private void btFecharActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.dispose();
}
```

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.