Aula 01

- Definição de estudo de caso (CRUD para Cidade)
- Implementação das classes de modelo
- Implementação de classes DAO com persistência em banco de dados

1 - Menu Aquivo -> Novo Projeto -> Sem classe principal

2 - Pacote código fonte -> nova classe -> Cidade -> Criar Pacote modelo

implements Serializable (implementa o padrão Javabeans)-> **possibilita a persistência e restauração do estado do objeto das classes**

private Integer codigo; String nome, uf; botão direito mouse refatorar encapsular os campos - > tirar javadoc

Inserir construtor vazio -> **padrão javabeans**

Criar campo chave para a classe (codigo) onde será comparado duas cidade por ex. -> para fazer botão direito -> inserir código - > equals() e hascode() -> marcar código e gerar.

Aula 02

1 – Criar um BD no mysql, nome : cadastro, coleção: latin1_swedish_ci.

2 – Criar uma tabela com nome cidade (codigo, nome e uf).

3 – Adicionar a biblioteca do drive JDBC do mysql no projeto.

4 - No pacote modelo criar uma classe com o nome **Conexao** -> private static final String **banco** = "jdbc:mysql://localhost:3306/cadastro"; private static final String **driver** = "com.mysql.jdbc.Driver";

private static final String **usuario** = "root"; private static final String **senha** = "";

private static Connection con = null; public **Conexao**() { }

public static Connection **getConexao**() { if (con == null) { try { Class.forName(driver);

con = DriverManager.getConnection(banco, usuario, senha); } catch (ClassNotFoundException ex) {

System.out.println("Não encontrou o driver: " + ex.getMessage()); } catch (SQLException ex) {

System.out.println("Erro na conexão: " + ex.getMessage()); } } return con; }

Passa os comandos SQL para o BD executar -> public static PreparedStatement **getPreparedStatement**(String sql) { if (con == null) { con = getConexao(); } try { return con.prepareStatement(sql); } catch (SQLException ex) { System.out.println("Erro de SQL: " + ex.getMessage()); } return null; }

5 – Alterar DAOCidade -> public List<Cidade> **getLista**() { String sql = "select * from cidade";

List<Cidade> lista = new ArrayList<>(); try { PreparedStatement pst = Conexao.getPreparedStatement(sql); ResultSet rs = pst.executeQuery(); while (rs.next()) {

Cidade obj = new Cidade(); obj.setCodigo(rs.getInt("codigo")); obj.setNome(rs.getString("nome")); obj.setUf(rs.getString("uf")); lista.add(obj); } } catch (SQLException e) { JOptionPane.showMessageDialog(null, "Erro de SQL: " + e.getMessage()); } return lista; }**// testar**

Alterar depois do incluir e alterar - > public boolean **salvar**(Cidade obj) { if (obj.getCodigo() == null) { return incluir(obj); } else { return alterar(obj); } }

public boolean **incluir**(Cidade obj) { String sql = "insert into cidade (nome,uf) values(?,?)"; try { PreparedStatement pst = Conexao.getPreparedStatement(sql); pst.setString(1, obj.getNome()); pst.setString(2, obj.getUf()); if (pst.executeUpdate() > 0) **// se consiguiu alterar uma linha** { JOptionPane.showMessageDialog(null, "Cidade incluida com sucesso"); return true; } else { JOptionPane.showMessageDialog(null, "Cidade não incluida com sucesso"); return false; } } catch (SQLException e) { JOptionPane.showMessageDialog(null, "Erro de SQL: " + e.getMessage()); return false; } }

Copiar o incluir e alterar -> public boolean **alterar**(Cidade obj) { String sql = "**update cidade set nome = ?, uf = ? where codigo = ?";** try { PreparedStatement pst = Conexao.getPreparedStatement(sql); pst.setString(1, obj.getNome()); pst.setString(2, obj.getUf()); **pst.setInt(3, obj.getCodigo());** if (pst.executeUpdate() > 0) { JOptionPane.showMessageDialog(null, "**Cidade alterada com sucesso**"); return true; } else { JOptionPane.showMessageDialog(null, "**Cidade não alterada com sucesso**"); return false; }

} catch (SQLException e) { JOptionPane.showMessageDialog(null, "Erro de SQL: " + e.getMessage()); return false; } }

Copiar alterar e alterar -> public boolean **remover**(Cidade obj) { String sql = "**delete from cidade where codigo = ?";** try { PreparedStatement pst = Conexao.getPreparedStatement(sql); **pst.setInt(1, obj.getCodigo());** if (pst.executeUpdate() > 0) { JOptionPane.showMessageDialog(null, "Cidade **excluida com sucesso**"); return true; } else { JOptionPane.showMessageDialog(null, "Cidade **não excluida com sucesso**"); return false; } } catch (SQLException e) { JOptionPane.showMessageDialog(null, "Erro de SQL: " + e.getMessage()); return false; } }

return null; }

Aula 03

- Implementação da Interface Gráfica (menu)

1 - Criar o menu da aplicação -> Botão direito pacote de código fonte - > novo -> outro -> Form GUI Swing -> Form JFrame -> pacote visual -> nome da classe FormPrincipal. Propriedade dor form (janela) title = Sistema de Cadastro

Paleta -> arrastar Barra de Menu (mostrar código fonte gerado)

Mudar nome da variável no navegador para um nome mais sugestivo -> JMenuBar para barraMenu -> Jmenu1 para menuCadastro e Jmenu2 para menuAjuda

Mudar o nome File no menu para Cadastro e Edit para Ajuda - > botão direito mouse -> editar texto

Menu Cadastro - > botão direito mouse -> Adicionar da Paleta -> Item de menu ->editar texto -> Funcionario -> novamente para cidade.

Mudar nome de variável no navegador do JmenuItem para menuFuncionario e menuCidades.

Menu Ajuda - > botão direito mouse -> Adicionar da Paleta -> Item de menu ->editar texto -> Sobre e mudar nome de variável no navegador para menuSobre.

Clicar botão direito menu Ajuda-> Sobre e adicionar o evento Action e digitar: JoptionPane.showMessageDialog(null,”Sistema de Cadastro \n Direitos Reservados”);

2 - Execute o formPrincipal através do shif + f6 ou play do projeto definindo a classe principal. Limpar e construir entrar na pasta dist e executar com dois cliques.

3 - Para aplicação executar no centro da tela. Entrar no método main e comentar new FormPrincipal().... e adicionar FormPrincipal form = new FormPrincipal(); form.setLocationRelativeTo(null); form.setVisible(true);

Aula 04

- **Interface Gráfica (CRUD - Formulário)**

1 - Botão direito mouse no pacote visual,outros-> Form GUI Swings-> Form Dialog-> nome FormCidade.

Propriedade -> Title -> Cadastro de Cidades.

2 - Chamar Cadastro de Cidadade no subitem do menu Cadastro-> Cidades -> botão direito-> eventos-> action-> digite -> **FormCidade form = new FormCidade(this,true);** this-> chamada a partir deste form e true-> modal: sobrepõe o form principal e não deixa acessar-lo.

form.setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE); ->não deixa fechar o form pelo x, **form.setTitle("Manutenção de Cidades"); form.setLocationRelativeTo(null);** ->centro da tela **form.setResizable(false);** -> retira botão maximizar **form.setVisible(true);**

3 - Arrastar um painel para dentro do formCidade e alterar o nome da variável para painelNavegacao. Clico no painel-> propriedade-> border ...-> borda com titulo -> Navegação. Redimensiono o painel no topo e arrasto 5 botões de navegação (Primeiro, anterior, próximo, ultimo e Fechar) e altero os nomes de variáveis ex,: btnPrimeiro.
4 - Para distribuir os botões uniformemente, botão direito em cima painel-> definir Layout-> Layout de Grade. Fechar o form somente no botão fechar-> botão direito mouse em fechar-> evento-> action-> digite-> dispose();

5 - Arrastar um Painel com Guias para o FormCidade e ajusta-lo abaixo dos botões de navegação e alterar nome de variável para abas. Depois arrasto um painel para dentro do abas, depois arrasto outro, solto somente quando ficar pontilhado ao abas. Alterar nome de variável-> primeiro painel -> abaListagem e titulo para Listagem e o segundo -> abaDados e titulo-> Dados.

7 - Arrasto uma tabela para aba Listagem - > altero nome variável-> tblObjetos. Para ajustar seleciono abaListagem -> definir Layout-> Layout de borda.

8 - Arrasto um Painel para aba Dados - > altero nome variável-> painelAcoes. Em propriedade selecionar borda ... -> com titulo - > Ações. Ajusto o tamanho do painel na parte superior da aba e coloco 5 botões (Novo, Editar, Cancelar, Salvar e Excluir). Seleciono o painel -> layout de grade. Mudo nome das variáveis (btnNovo, btnEditar, btnCancelar, btnSalvar e btnExcluir).

9 - Na aba Dados arraste 3 label (Código:, Nome:, e UF:), 2 campo de texto e 1 caixa combinação respectivamente -> retirar texto - > alterar nome de variável (txtCodigo, txtNome e cbxUF), selecionar os componentes e alinhar. Adicionar as UF (MG, SP) em propriedade model e desabilitar o campo código - > enable em propriedade.

10 - Adicionar biblioteca ao projeto -> botão direito do mouse no nome do projeto -> propriedades -> bibliotecas -> adicionar bibliotecas –> bibliotecas globais - > vinculação de beans.

11 - Na aba Listagem vincular a tabela aos dados (vinculação de beans), na paleta de componentes -> Java Persistence -> Resultado da Consulta (arrasto para dentro do painel Listagem). Renomear variável no navegador -> list1 para listObjetos. Seleciono listObjeto -> propriedade -> Código -> Parâmetros do Tipo-> digito: <Cidades> (Estou dizendo que esta lista só poderá ter objetos do tipo Cidade). Depois inicializo a lista dentro de Código de Criação Personalizado ... digito-> org.jdesktop.observablecollections.ObservalbleCollections.observalbleList(new ArrayList <Cidade> ( )) , depois importar as bibliotecas no código (import Java.util.ArrayList; e modelo.Cidade;).

12 - Vincular a tabela com os dados -> botão direito do mouse em cima da tabela em Listagem -> Vincular -> elements -> Código fonte de Vinculação -> listObjetos -> ok. Posso editar os label da Tabela -> botão direito do mouse -> Conteúdo da tabela -> Colunas -> mudar os títulos colocando acento -> e retirar opção Editável.

Aula 5

- Implementação da Interface Gráfica (Cidade)

1 – Vincular campos com a tabela: botão direito mouse campo txtCodigo -> vincular text -> seleciono tblObjetos -> ${selectedElement.codigo}. O mesmo para o campo nome. Para txtUF -> vincular -> selectedItem

2 - No código fonte da classe FormCidade instanciar a classe **DAOCidade dao = new DAOCidade();**

3 - Em FormCidade crie o método: public void atualizaTabela(){ listObjetos.clear(); listObjetos.addAll(dao.getLista()); int linha = listObjetos.size()-1; if(linha >=0) { tblObjetos.setRowSelectionInterval(linha, linha); tblObjetos.scrollRectToVisible(tblObjetos.getCellRect(linha, linha, true)); } }

4 - No contrutor de FormCidade chame o método: atualizaTabela();

5 – No action do botão Novo: listObjetos.add((Cidade) new Cidade());-> **cria mais uma linha na tabela** int linha = listObjetos.size() -1;

tblObjetos.setRowSelectionInterval(linha, linha); txtNome.requestFocus();

6 – No action do botão Salvar: int linhaSelecionada = tblObjetos.getSelectedRow(); Cidade obj = listObjetos.get(linhaSelecionada); dao.salvar(obj); atualizaTabela();

7 – Na classe FormCidade crie o método: private void trataEdicao(boolean editando) { btnCancelar.setEnabled(editando); btnSalvar.setEnabled(editando); btnEditar.setEnabled(!editando); int linha = listObjeto.size() - 1; if(linha<0){ btnExcluir.setEnabled(false); txtCodigo.setText(""); txtNome.setText(""); }else{ btnExcluir.setEnabled(!editando); } btnNovo.setEnabled(!editando); btnFechar.setEnabled(!editando); btnPrimeiro.setEnabled(!editando); btnProximo.setEnabled(!editando); btnAnterior.setEnabled(!editando); btnUltimo.setEnabled(!editando); txtNome.setEnable(editando); cbxUF.setEnable(editando); tblObjetos.setEnabled(editando); }

8 – No construtor de FormCidade adicione o método: trataEdicao(false);

9 – No action do btnNovo : adicione o método trataEdicao(true);

10 – No action do btnSalvar : adicione o método trataEdicao(false);

11 – Validar Campos: Na classe FormCidade crie o método: public boolean validaCampos(){ if(!(txtNome.getText().length()>0)){ JOptionPane.showMessageDialog(null, "Informe o nome da Cidade"); txtNome.requestFocus(); return false; } if(!(cbxUF.getSelectedIndex()>=0)){ JOptionPane.showMessageDialog(null, "Informe o UF da Cidade"); cbxUF.requestFocus(); return false; } return true; }

11.1 – No action do botão salvar -> if(validaCampo()){ todo código de salvar }

12 – No action do botão Editar: trataEdicao(true); txtNome.requestFocus();

13 – No action do botão Cancelar: trataEdicao(false); atualizaTabela();

14 – No action do botão Excluir: int opcao = JOptionPane.showOptionDialog(null, "Confirma a exclusão?", "Pergunta",JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, new String [] {"Sim","Não"},"Sim"); if(opcao==0){ ***int linhaSelecionada = tblObjetos.getSelectedRow(); Cidade obj = listObjetos.get(linhaSelecionada); dao.remover(obj); atualizaTabela();*** trataEdicao(false); }

15 – No action do botão Primeiro: tblObjetos.setRowSelectionInterval(0, 0); tblObjetos.scrollRectToVisible(tblObjetos.getCellRect(0, 0, true));

16 – No action do botão Anterior: int linha = tblObjetos.getSelectedRow(); if((linha-1)>=0){ linha --;

} tblObjetos.setRowSelectionInterval(linha, linha); tblObjetos.scrollRectToVisible(tblObjetos.getCellRect(linha, 0, true));

17 – No action do botão Próximo: int linha = tblObjetos.getSelectedRow();

if((linha+1)<=(tblObjetos. getRowCount()-1)){ linha ++; } tblObjetos.setRowSelectionInterval(linha, linha); tblObjetos.scrollRectToVisible(tblObjetos.getCellRect(linha, 0, true));

18 – No action do botão Ultimo: int linha = tblObjetos.getRowCount()-1; tblObjetos.setRowSelectionInterval(linha, linha); tblObjetos.scrollRectToVisible(tblObjetos.getCellRect(linha, 0, true));