**Trabalho Final de Programação Orientada a Objetos**

**Objetivo do trabalho:**

Neste trabalho, você deve desenvolver uma aplicação de banco chamada *Banco Malvader* em Java, com interface gráfica utilizando **Swing** e **salvamento de dados em arquivos**. O sistema deve implementar funcionalidades de gerenciamento de contas bancárias, tal como apresentado anteriormente, aplicando conceitos de Programação Orientada a Objetos (POO) e persistência de dados.

**Estrutura do Projeto e Pacotes**

Organize o projeto em pacotes para separar a lógica da aplicação da interface gráfica.

* **Pacotes Sugeridos**:
  + model: Contém as classes de modelo, como Conta, Cliente, Funcionario.
  + controller: Contém classes para gerenciamento de operações, como abertura de conta e geração de relatórios.
  + view: Contém todas as classes de interface gráfica (GUI).
  + utils: Contém classes auxiliares para persistência de dados em arquivos.

**3. Desenvolvimento da Interface Gráfica com Swing**

**Estrutura das Telas e Funcionalidades**

A aplicação deve conter as seguintes telas, de forma modular e organizada conforme o menu do sistema:

**Tela de Login**

* **Componentes**:
  + JTextField para o campo de usuário.
  + JPasswordField para a senha.
  + Dois JButton: um para acesso como Funcionário e outro para Cliente.
* **Funcionalidade**: Autentica o usuário e redireciona para o Menu Principal.

**Menu Principal**

* **Componentes**:
  + Três JButton para acessar as opções: Funcionário, Cliente e Sair.
* **Funcionalidade**: O botão "Funcionário" abre o Menu do Funcionário, e "Cliente" abre o Menu do Cliente.

**Menu Funcionário**

* **Componentes**:
  + Sete JButton para cada operação: Abertura de Conta, Encerramento de Conta, Consulta de Dados, Alteração de Dados, Cadastro de Funcionário, Geração de Relatórios e Sair.
* **Funcionalidade**: Cada botão executa uma função específica usando métodos da classe BancoController.

**Menu Cliente**

* **Componentes**:
  + Seis JButton para operações: Saldo, Depósito, Saque, Extrato, Consultar Limite e Sair.
* **Funcionalidade**: Cada botão chama uma função específica do sistema.

**Exemplo de Tela com Swing (Tela de Login):**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Estrutura de Classes e Funcionalidades (Modelo e Controle)**

**Classe BancoController (Controle de Operações)**

* **Atributos**:
  + Uma List<Conta> para armazenar as contas.
  + Uma List<Funcionario> para armazenar os funcionários.
* **Métodos**:
  + public void abrirConta(Conta conta): Registra uma nova conta.
  + public void encerrarConta(int numeroConta): Encerra uma conta.
  + public Conta consultarConta(int numeroConta): Consulta informações de uma conta específica.
  + public void salvarDados(): Salva todos os dados em arquivos.
  + public void carregarDados(): Carrega dados dos arquivos no início da aplicação.

**Classes de Modelo**

* **Conta** (Classe Abstrata):
  + Atributos: numero, saldo, tipoConta.
  + Métodos: depositar(), sacar(), consultarSaldo().
* **ContaCorrente** e **ContaPoupanca** (Subclasses de Conta):
  + Atributos específicos, como limite para ContaCorrente.
  + Métodos específicos, como verificarLimite() para ContaCorrente.
* **Cliente** e **Funcionario**:
  + Atributos como nome, cpf, telefone, endereço.
  + Métodos de getters e setters.

**5. Persistência de Dados com Arquivos**

Os dados do banco serão salvos em arquivos para garantir que as informações persistam entre as execuções do programa.

**Criando Métodos de Salvamento e Carregamento de Dados**

Crie uma classe DataManager em utils para lidar com operações de leitura e gravação de dados em arquivos.

* **Exemplo de Salvamento de Dados**:

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Uso no BancoController**:

* + No método salvarDados(), chame DataManager.salvarContas(contas, "contas.dat").
  + No método carregarDados(), chame DataManager.carregarContas("contas.dat").

Esses métodos devem ser chamados ao iniciar e encerrar a aplicação, para garantir que os dados estejam sempre atualizados.

**6. Exemplo de Implementação do Menu de Abertura de Conta**

No Menu Funcionário, implemente a funcionalidade de abertura de conta:

1. **Interface**: Ao selecionar "Abertura de Conta", exiba um formulário para o funcionário preencher os dados do cliente.
2. **Classe AberturaDeContaScreen**:
   * Inclua campos como JTextField para número da conta, nome, CPF, etc.
   * JComboBox para selecionar o tipo de conta (Corrente ou Poupança).
   * JButton para confirmar a criação da conta.
3. **Exemplo de Ação de Botão para Abertura de Conta**:

Texto

Descrição gerada automaticamente

**Entrega e Avaliação**

1. **Entrega do Código**: Organize o código em pacotes e classes claras, com comentários explicativos.
2. **Documentação**: Inclua uma breve documentação descrevendo as classes, o fluxo do sistema e as principais funcionalidades.
3. **Apresentação**: Apresente o funcionamento do sistema, explicando a interface e o processo de persistência dos dados.

**8. Critérios de Avaliação**

* **Funcionalidade**: Implementação correta das operações.
* **Interface Gráfica**: Interface amigável

**Tutorial Rápido de Interface com Swing**

Para criar uma interface gráfica em Java usando **Swing**, vamos explorar alguns conceitos básicos e componentes essenciais. Swing é uma biblioteca gráfica nativa do Java que facilita a criação de janelas, botões, campos de texto e outros elementos visuais.

**1. Criando Janelas com JFrame**

Para iniciar uma interface, criamos uma janela principal usando JFrame.

Texto

Descrição gerada automaticamente

**Organizando Componentes com JPanel e Layouts**

O JPanel é usado para organizar componentes como botões e campos de texto dentro do JFrame. Usamos diferentes layouts (como FlowLayout, GridLayout, BorderLayout) para posicionar os componentes na tela.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Adicionando Componentes Básicos**

Swing possui diversos componentes, como:

* **JLabel**: Exibe texto.
* **JTextField**: Campo para entrada de texto.
* **JPasswordField**: Campo para entrada de senha.
* **JButton**: Botão para executar ações.

**Exemplo**:

Tela de computador

Descrição gerada automaticamente

**Adicionando Ações aos Botões com ActionListener**

Para responder a cliques em botões, usamos o ActionListener, que permite associar uma ação ao clique do usuário.

Tela de computador

Descrição gerada automaticamente

**Usando Caixas de Diálogo com JOptionPane**

O JOptionPane permite criar caixas de diálogo simples para exibir mensagens ou capturar entradas.

* **Mostrar uma mensagem**:

Texto

Descrição gerada automaticamente

**Mostrar uma pergunta e capturar a resposta**:

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Exemplo Completo: Tela de Login com Swing**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Próximos Passos para o Projeto**

Para o projeto do Banco Malvader, vocês usarão esse conhecimento para criar menus e telas adicionais, conectando-as à lógica da aplicação. Experimente criar:

* **Menus com Botões** para as funcionalidades.
* **Formulários** de entrada para capturar dados dos usuários.
* **Caixas de Diálogo** para exibir mensagens e confirmações.

Pessoal, essas ferramentas vão permitir que vocês criem uma interface gráfica interativa para o sistema de banco, integrando funcionalidades conforme solicitado no projeto.

Vamos lá, vocês conseguem.