Trabalho prático Swing e JDBC.
Valor: 100% da quarta nota.
Data da entrega: Ver moodle
Data das defesas: Ver moodle
Equipes: Máximo 4 pessoas.

Um banco resolveu aprimorar os seus produtos financeiros, e por conta disso vai alterar o sistema que gerencia as contas de seus clientes. O novo sistema precisa ser em Java com interface SWING e seguindo o paradigma orientado a objetos. O sistema terá os seguintes requisitos:

1. Uma tela para manter (incluir, atualizar, excluir e listar) os clientes do banco (Nome, sobre nome, RG, CPF, Endereço)
    a. Nesta tela deve ser possível listar todos os clientes (Use AbstractTableModel)
    b. Deve ser possível atualizar os dados de um cliente.
    c. Deve ser possível excluir um cliente. Quando isso acontecer uma tela deve ser mostrada para confirmar o procedimento. Deve-se avisar que todas as contas do cliente serão apagadas. Todas as contas vinculadas a este cliente devem ser apagada.
    d. Deve ser possível listar os clientes por nome (ou parte do nome), por sobrenome (ou parte), por RG ou por CPF.
    e. Deve ser possível ordenar a lista por nome, sobrenome(ordem alfabética) ou salário (do maior para o menor). Implemente a interface Comparable.
2. Uma tela para vincular o cliente à uma conta. Para simplificar, um cliente pode estar vinculado à apenas um tipo de conta.
    a. Nesta tela deve ser possível selecionar um cliente.
    b. Os tipos de conta vinculados ao cliente podem ser selecionadas de uma combobox. Os tipos de conta serão: Conta Corrente e Conta Investimento
    c. Uma vez selecionado o cliente e o tipo de conta a tela deve apresentar os campos para criação da conta
        i. Se o tipo de conta selecionada for Conta Corrente, os campos serão:
            1. Depósito Inicial (valor em R$)
            2. Limite (valor em R$)
            3. O número da conta deve ser gerado automaticamente pelo sistema. Pode ser inteiro e gerado de forma sequencial.
        ii. Se o tipo da conta selecionada for Conta Investimento, os campos serão:
            1. Montante Mínimo
            2. Depósito Mínimo
            3. Depósito Inicial
3. Uma tela para manipular a conta de um cliente.
    a. Nesta tela deve ser possível selecionar a conta de um cliente pelo CPF do cliente.
    b. Uma vez selecionada a conta o sistema deve mostrar três operações possíveis:
        i. Saque: Permite efetuar um saque da conta do valor digitado
        ii. Depósito: Permite efetuar um depósito na conta do valor digitado
        iii. Ver Saldo: Mostra o saldo da conta para o cliente
        iv. Remunera: Aplica remuneração à conta: invoca método remunera().
4. Deve-se utilizar herança para definir as contas: Uma classe Conta deve ser criada, as classes ContaCorrente e ContaInvestimento devem herdar de conta. A classe conta deve implementar a seguinte interface:

```
public interface ContaI {
    public boolean deposita(double valor);
    public boolean saca(double valor);
    public Cliente getDono();
    public int getNumero();
    public double getSaldo();
    public void remunera();
}
```

5. As seguintes regras devem ser implementadas nas respectivas classes conta:
    a. Conta:
        i. **`public boolean deposita(double valor):`** o valor depositado deve ser positivo. Caso contrário o método retorna false
        ii. **`public boolean saca(double valor)`**: o valor sacado deve ser positivo. Caso contrário o método retorna false. Mostrar mensagem na tela informando usuário.
    b. ContaCorrente:
        i. **`public boolean saca(double valor)`**: Antes de efetuar o saque deve-se verificar se o valor sacado não ultrapassa o limite da conta. Ou seja, a conta poderá ficar negativa até o limite estipulado na sua criação. Mostrar mensagem na tela informando o usuário.
        ii. **`public void remunera()`**: Aplicar remuneração de 1% ao saldo da conta.
    c. ContaInvestimento:
        i. **`public boolean deposita(double valor):`** Recebe como parâmetro o valor a ser depositado. Se o valor a ser depositado for maior ou igual ao depositoMinimo então, o depósito deve ser efetuado. Para isso chame o método deposita da classe pai (Conta) e retorne true. Caso contrário, deve-se retornar false. Mostrar mensagem na tela informando usuário.

ii. **`public boolean saca(double valor):`** Recebe como parâmetro o valor a ser sacado. Se o novo valor do saldo (considerando o saque) for maior ou igual ao montanteMinimo, o saque deve ser efetuado. Para isso invoque o método saque da classe pai (Conta) e retorne true. Caso contrário, deve-se retornar false. Mostrar mensagem na tela informando usuário.

iii. **`public void remunera()`**: Aplicar remuneração de 2% ao saldo da conta.

O programa acima deve ser feito utilizando as seguintes tecnologias:

- Utilizar herança para definir as classes Conta, ContaCorrente, ContaInvestimento. Utilize classes abstratas quando necessário.
- Utilizar polimorfismo para manipular a conta de um cliente.
- Java Swing
- JDBC para persistência em banco de dados. Utilize o Design Pattern "Data Access Object" para persistência.

Itens para serem entregues:

1. Diagrama de classes
2. Diagrama de E-R
3. Projeto na IDE Eclipse ou Netbeans com código fonte
4. Script para criação do banco de dados
5. Arquivo .jar executável

Avaliação:

- Material entregue
- Qualidade do software (bugs encontrados na defesa)
- Defesa do código e a nota será individual, considerando a defesa