ELE078 - PROGRAMAÇÃO ORIENTADA A OBJETOS

Trabalho Pratico 1

I – Observações:

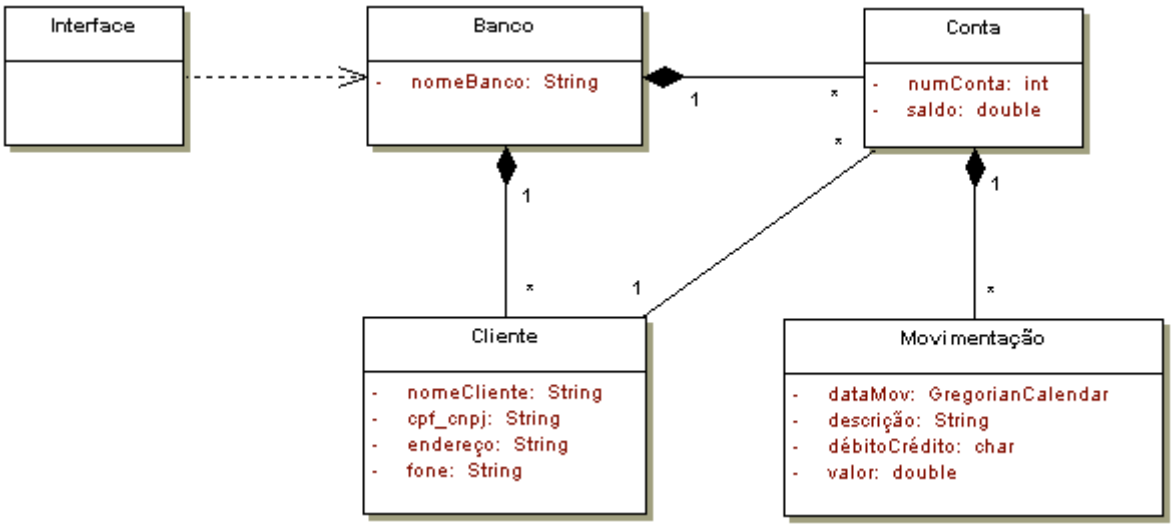
1) O trabalho pode ser feito em grupo de (no máximo) 5 pessoas. É permitido discutir os problemas e estratégias de solução com outros grupos, mas quando se tratar de escrever ou implementar computacionalmente as soluções, isto deve ser feito separadamente. O trabalho deve ser feito em Linguagem C++.

2) Forma de entrega: O trabalho deve ser entregue em formato digital por meio do Moodle. Utilizar a opção "Entrega do Trabalho Pratico 1". Anexe um único arquivo .zip contendo todos os arquivos do trabalho (códigos-fonte, executáveis, documentação, etc.). O nome do arquivo .zip a ser enviado deve ser formado pelos nomes de todos os integrantes do grupo. Assim, se o grupo é formado pelos alunos: Cristiano Leite Castro, Danilo Melges e André Freire, o nome do arquivo deverá ser: CristianoCL_DaniloM_AndreF.zip.

3) Trabalhos copiados receberão nota zero para todas as cópias. Trabalhos com erros de compilação não serão avaliados e receberão nota zero**.** O programa deve ser desenvolvido seguindo as boas normas de programação.

Sistema de Controle de Bancário

Implemente em C++ um conjunto de classes para um sistema de controle bancário. Um Banco necessita manter informações sobre seus clientes e contas. Por simplificação do trabalho, não existirá uma classe Agência e uma Conta pertencerá a um único Cliente (não existe conta conjunta). Para isso, foi feito o seguinte projeto (em UML):



- Classe Cliente:
  - Atributos: nomeCliente (string), cpf_cnpj (string), endereço (string) e fone (string). Todos com acesso privado;
  - Método construtor para inicializar todos os atributos através de parâmetros;
  - Métodos *get* e *set* para obter e modificar cada um dos valores dos atributos.

- Classe Movimentação:
  - Atributos: dataMov (tipo Data; pode-se usar algum tipo abstrato de dados para manipulação de datas já existente na biblioteca padrão), descrição (string), débitoCrédito (char) e valor (double). Todos com acesso privado;
  - Método construtor para inicializar os atributos descrição, débitoCrédito e valor através de parâmetros e dataMov com a data corrente do sistema;
  - Métodos *get* para obter cada um dos valores dos atributos.

- Classe Conta:
  - Atributos: numConta (int), saldo (double), cliente (Cliente), movimentações (lista de Movimentação) e próximoNumConta (int). O atributo próximoNumConta deve ser **static** e será usado para armazenar o próximo número de conta. A lista de movimentações deve ser implementada usando alguma classe que manipula containers, tais como Listas ou Vetores Dinâmicos (vector). Todos com acesso privado;
  - Método construtor para gerar o número da conta, inicializar o saldo com valor zero e o cliente com um valor passado como parâmetro;
  - Métodos *get* para obter os valores dos atributos;
  - Método para debitar um valor da conta. Parâmetros: o valor a ser debitado e a descrição da movimentação. Deve adicionar um objeto de movimentação à lista de movimentações da conta com as seguintes informações: dataMov obtida do sistema, descrição passada como parâmetro, débitoCrédito com valor "D" e valor passado como parâmetro. O atributo saldo deve ser decrementado pelo valor do débito. Se o saldo se tornar negativo, a operação não deve ser efetuada;
  - Método para creditar um valor à conta. Parâmetros: o valor a ser creditado e a descrição da movimentação. Deve adicionar um objeto de movimentação à lista de movimentações da conta com as seguintes informações: dataMov obtida do sistema, descrição passada como parâmetro, débitoCrédito com valor "C" e valor passado como parâmetro. O atributo saldo deve ser incrementado pelo valor do crédito;
  - Método para obter um extrato da conta. Parâmetros: a data inicial e a data final. Deve retornar uma lista contendo todas as movimentações da conta no período entre a data inicial e a data final, inclusive;
  - Método para obter um extrato da conta. Parâmetros: a data inicial. Deve retornar uma lista contendo todas as movimentações da conta no período entre a data inicial e a data corrente, inclusive;
  - Método para obter um extrato da conta. Deve retornar uma lista contendo todas as movimentações da conta no mês corrente.

- Classe Banco:
  - Atributos: nomeBanco (String), clientes (lista de Clientes) e contas (lista de Contas). As listas devem ser implementadas usando alguma classe que manipula containers, tais como Listas ou Vetores Dinâmicos (Vector). Todos com acesso privado;
  - Método construtor para inicializar o atributo nomeBanco com um valor passado como parâmetro;
  - Método para inserir um novo cliente na lista de clientes. Parâmetro: o cliente;
  - Método para criar uma nova conta. Parâmetro: o cliente. A conta criada deve ser adicionada à lista de contas do Banco;
  - Método para excluir um cliente da lista de clientes. Parâmetro: o CPF ou CNPJ do cliente. O cliente não pode ser excluído se possuir alguma conta;

- Método para excluir uma conta da lista de contas. Parâmetro: o número da conta;
- Método para efetuar depósito em uma conta. Parâmetros: o número da conta e o valor a depositar. Gera uma movimentação de crédito com a descrição “Depósito”;
- Método para efetuar um saque em uma conta. Parâmetros: o número da conta e o valor a sacar. Gera uma movimentação de débito com a descrição “Saque”;
- Método para efetuar uma transferência entre contas. Parâmetros: o número da conta de origem, o número da conta de destino e o valor a transferir. Gera uma movimentação de débito na conta de origem com a descrição “Transferência para conta ‘número da conta de destino’” e uma movimentação de crédito na conta de destino com a descrição “Transferência da conta ‘número da conta de origem’”. Decrementa o saldo da conta de origem e incrementa o saldo da conta de destino;
- Método para cobrar tarifa. Debita R$15,00 do saldo de todas as contas do Banco. Gera uma movimentação de débito em cada conta com a descrição “Cobrança de tarifa”;
- Método para cobrar CPMF. Debita, de todas as contas, 0,38% do valor total das movimentações de débito nos últimos 7 dias. Gera uma movimentação de débito em cada conta com a descrição “Cobrança de CPMF”;
- Método para obter o saldo de uma conta. Parâmetro: o número da conta;
- Método para obter o extrato de uma conta. Parâmetro: o número da conta. Obtém o extrato para o mês corrente;
- Método para obter o extrato de uma conta. Parâmetros: o número da conta e a data inicial. Obtém o extrato a partir da data inicial até a data corrente;
- Método para obter o extrato de uma conta. Parâmetros: o número da conta, a data inicial e a data final. Obtém o extrato para o período entre a data inicial e a data final, inclusive;
- Método para obter a lista de clientes;
- Método para obter a lista de contas;
- Método para gravar os dados em arquivo;
- Método para ler os dados armazenados em arquivo.

- Classe Interface:
  - Método para apresentar um menu (interface de caracteres) com opções para: cadastrar um novo cliente, criar uma nova conta, excluir um cliente, excluir uma conta, efetuar depósito, efetuar saque, efetuar transferência entre contas, cobrar tarifa, cobrar CPMF, obter saldo, obter extrato, listar clientes, listar contas e sair do programa;
  - Método para cadastrar um novo cliente. Deve permitir que o usuário entre com os dados do cliente a ser inserido;
  - Método para criar uma nova conta. Deve permitir que o usuário escolha o cliente para a conta;
  - Método para excluir um cliente. Deve permitir que o usuário escolha o cliente a ser excluído;
  - Método para excluir uma conta. Deve permitir que o usuário escolha a conta a ser excluída;
  - Método para efetuar um depósito. Deve permitir que o usuário escolha a conta e o entre com o valor a depositar;
  - Método para efetuar um saque. Deve permitir que o usuário escolha a conta e o entre com o valor a sacar;
  - Método para efetuar uma transferência. Deve permitir que o usuário escolha a conta de origem, a conta de destino e o entre com o valor a transferir;

- Método para cobrar tarifa. Na prática, este método seria chamado automaticamente uma vez por mês em uma data específica. No trabalho, ele poderá ser acionado a qualquer momento;
- Método para cobrar CPMF. Na prática, este método seria chamado automaticamente uma vez por semana em uma data específica. No trabalho, ele poderá ser acionado a qualquer momento;
- Método para obter saldo. Deve permitir que o usuário escolha a conta. O saldo deve ser exibido na tela;
- Método para obter extrato. Deve ter opções para que o usuário escolha entre extrato do mês, extrato a partir de uma data ou extrato em um período específico e, dependendo da escolha, entre com os dados necessários. Todas as informações do extrato (data da movimentação, descrição, débito/crédito e valor) devem ser listadas e, no final, o saldo atual deve ser exibido;
- Método para listar todos os clientes. Deve listar os dados de todos os clientes;
- Método para listar todas as contas. Deve listar os dados de todas as contas;
- Método *main* para ler os dados armazenados em arquivo, apresentar o menu principal e, antes de sair do programa, gravar os dados em arquivo.

Observe a separação entre as classes de negócio (Cliente, Conta e Movimentação) e a classe Interface. As classes de negócio não devem possuir nada que crie uma dependência da interface. Desta forma, se posteriormente você decidir mudar a interface, por exemplo, usar uma interface gráfica, as classes de negócio não precisam ser alteradas. A classe Banco atua como uma classe gerente, fazendo a interligação das classes de negócio com a classe de interface. Em um projeto mais profissional, provavelmente existiriam várias interfaces e classes gerente. A mesma idéia deveria ser usada para separar as classes de negócio e as classes de persistência. Por simplicidade, nesse trabalho o armazenamento dos dados será feito em arquivo e as próprias classes de negócio e de gerência sabem como fazer isso, o que significa que se decidirmos passar a usar um banco de dados teremos que alterar essas classes. Em um projeto mais profissional, deveríamos criar uma camada de persistência, separando as classes de negócio da estrutura de armazenamento.

Observações a serem seguidas:

- As classes, atributos e métodos descritos constituem o projeto básico do sistema. Pode-se adicionar outros métodos e atributos necessários para completar o sistema. Porém qualquer mudança “radical” no projeto deve ser discutida com o professor;
- Leitura de dados do usuário e impressão na tela devem ser feitos somente na classe Interface;