Quarto Exercício-Programa

Luciano Antonio Digiampietri
Norton Trevisan Roman

Prazo máximo para a entrega: 27/06/2020

1 Sistema de Gerenciamento de um Banco

Neste trabalho, você deverá desenvolver um sistema simplificado para o gerenciamento de um banco. De uma maneira resumida (que será detalhada ao longo deste enunciado), um banco é composto por gerentes, os quais gerenciam um conjunto de clientes (que podem ou não ser clientes especiais). Há apenas duas características dos clientes a serem gerenciadas: o valor em suas contas correntes e o valor de suas dívidas/empréstimos com o banco. Um cliente pode estar associado a mais de um gerente. Cada banco pode ter até 10 gerentes e cada gerente pode gerenciar até 20 clientes.

Detalhamento:

Nesta simplificação existem as seguintes classes principais, descritas a seguir: *Pessoa*, *Gerente*, *Cliente*, *ClienteEspecial* e *Banco*, e duas interfaces: *InterfaceCliente* e *InterfaceGerente*.

Classe *Pessoa*: é uma classe abstrata (não pode ser instanciada) que possui: dois atributos (*nome* do tipo String e *cpf* do tipo inteiro); um construtor que recebe o nome e o CPF da pessoa como parâmetros; e um método abstrato cuja assinatura é abstract String retornaTipo() e deve retornar o tipo da Pessoa, como será visto adiante ("C" para Cliente, "CE" para ClienteEspecial e "G" para Gerente).

A classe *Pessoa* é herdada pelas classes *Gerente* e *Cliente*.

Classe *Gerente*: possui, além dos atributos herdados da classe *Pessoa*, três atributos adicionais: uma constante chamada *tipo* cujo valor é "G", um arranjo de objetos do tipo Cliente, chamado *clientes* e um inteiro chamado *numClientes* que indica quantos clientes há dentro do arranjo *clientes*.

A classe já possui um construtor bem com os métodos String retornaTipo() e void imprimirClientes() implementados. O método imprimirClientes exibe algumas informações dos clientes gerenciados pelo gerente atual (ver código fornecido para mais informações).

Adicionalmente, a classe *Gerente* implementa a interface *InterfaceGerente*, que possui dois métodos (que deverão ser implementados/completados por você, em *Gerente*):

- public boolean adicionarCliente(Cliente cliente): método para adicionar um cliente no arranjo de clientes do gerente atual. Caso o número de clientes seja igual a 20, não deve adicionar e deve retornar *false*. Caso contrário, há duas situações: 1ª: o cliente já consta

no arranjo de clientes (verifique isso usando o número do CPF), neste caso o cliente não deve ser reinserido e o método deve retornar *false*; 2ª: o cliente passado como parâmetro não consta no arranjo de clientes, neste caso o cliente deve ser adicionado na posição *numClientes*, este atributo deve ser incrementado em 1 e o método deve retornar *true*.

- public void cobrarTodosEmprestimos(): método para cobrar os empréstimos de todos os clientes do gerente atual. Para cada um dos clientes presentes no arranjo clientes do gerente atual, este método deve: não fazer nada para o cliente, caso seu *valorDaDivida* seja igual a zero; caso contrário, há duas situações: 1ª: se o *valorContaCorrente* do cliente for maior ou igual ao *valorDaDivida*, deve fazer o cliente pagar a dívida, isto é, o *valorContaCorrente* será atualizado, descontando-se o valor da dívida e o *valorDaDivida* será zerado. 2ª: se o *valorContaCorrente* do cliente for menor do que o *valorDaDivida*, deve fazer o cliente pagar parte da dívida, isto é, o *valorDaDivida* será atualizado, tendo seu valor diminuído pelo *valorContaCorrente* e o *valorContaCorrente* será zerado.

Classe *Cliente*: possui, além dos atributos herdados da classe *Pessoa*, quatro atributos adicionais: uma constante chamada *tipo* cujo valor é "C", uma constante do tipo inteiro chamada *dividaMaxima*, que corresponde ao valor máximo de dívida/empréstimo que um cliente pode ter (valor igual a 30000); e outros dois atributos privados do tipo inteiro *valorContaCorrente* que armazena o valor que o cliente possui em sua conta corrente e *valorDaDivida* que armazena o valor que o cliente está devendo ao banco.

A classe já possui um construtor bem com o método String retornaTipo(), e getters e setters para seus atributos privados implementados.

Adicionalmente, a classe *Cliente* implementa a interface *InterfaceCliente*, que possui quatro métodos (que deverão ser implementados/completados por você em *Cliente*):

- public boolean obterEmprestimo(int valor): método para o cliente atual obter um empréstimo de acordo com o valor passado por parâmetro. Caso o valor do parâmetro seja menor ou igual a zero, o método deve retornar *false*. Caso contrário há duas situações: 1ª: se o valor do parâmetro mais o valor do atributo *valorDaDivida* seja maior do que o valor da constante *dividaMaxima*, o método deve retornar *false*; 2ª: caso contrário, o atributo *valorDaDivida* deve ser incrementado em *valor*, o atributo *valorContaCorrente* deve ser incrementado em *valor* e o método deve retornar *true*.

- public boolean pagarEmprestimo(int valor): método para o cliente atual pagar parte de sua dívida de acordo com o valor passado por parâmetro. Caso o valor do parâmetro seja menor ou igual a zero, o método deve retornar *false*. Caso contrário, há duas situações: 1ª: se o valor do parâmetro for maior do que o *valorDaDivida* ou for maior do que *valorContaCorrente*, o método deve retornar *false*; 2ª: caso contrário, o atributo *valorDaDivida* deve ser decrementado em *valor*, o atributo *valorContaCorrente* deve ser decrementado em *valor* e o método deve retornar *true*.

- public boolean negativado(): método que retorna *true* caso *valorContaCorrente* seja menor do que *valorDaDivida*.

- public boolean realizarSaque(int valor): método para o cliente atual realizar um saque do valor passado por parâmetro. Caso o valor do parâmetro seja menor ou igual a zero, o método deve retornar *false*. Caso contrário há duas situações: 1ª: se o valor do parâmetro for maior do que o valor do atributo *valorContaCorrente*, o método deve retornar *false*; 2ª: caso contrário, o atributo *valorContaCorrente* deve ser decrementado em *valor* e o método deve retornar *true*.

A classe *Cliente* é herdada pela **classe *ClienteEspecial***. Neste EP, a diferença de um cliente especial para um cliente "normal" está no valor de dois atributos *static*: *dividaMaxima* (cujo valor para cliente especial é 50000 e *tipo*, cujo valor é "CE".

A classe já possui um construtor bem com o método String retornaTipo() implementados.

Cabe a você verificar se será necessário sobrescrever algum dos métodos herdados da classe *Cliente*.

Classe *Banco*: possui dois atributos: um arranjo de objetos do tipo Gerente, chamado *gerentes* e um inteiro chamado *numGerentes* que indica quantos gerentes o banco possui (estes gerentes estarão no arranjo *gerentes*).

A classe já possui um construtor bem com o método void imprimirGerentes() implementados. O método imprimirGerentes exibe algumas informações dos gerentes do banco (ver código fornecido para mais informações).

A classe *Banco* possui um método adicional (que deverá ser implementado/completado por você):

boolean adicionarGerente(Gerente gerente): método para adicionar um gerente no arranjo de gerentes do banco. Caso o número de gerentes seja igual a 10, não deve adicionar e deve retornar *false*. Caso contrário, há duas situações: 1ª: o gerente já consta no arranjo de gerentes (verifique isso usando o número do CPF), neste caso o gerente não deve ser reinserido e o método deve retornar *false*; 2ª: o gerente passado como parâmetro não consta no arranjo de gerentes: o gerente deve ser adicionado na posição *numGerentes*, este atributo deve ser incrementado em 1 e o método deve retornar *true*.

Classe *ExecutaBanco*: esta classe possui um método *main()* que, quando executado, cria um cenário de um banco com gerentes e clientes e testa diversos métodos envolvidos neste EP. Esta classe não fará parte da avaliação do EP e serve apenas para te auxiliar nos testes de seu EP (ela, não necessariamente, cobre todos os testes possíveis para este EP).

1.1 Material a Ser Entregue

Todos os arquivos .java envolvidos neste EP são fornecidos para você, cabendo a você completá-los.

Atenção!

1. Não modifique a assinatura dos métodos fornecidos!

2. Para avaliação, os diferentes métodos que devem ser implementados serão invocado diretamente. Em especial, qualquer código dentro do *main()* será ignorado. Então certifique-

se de que cada um dos métodos faz o que a especificação diz.

2 Entrega

A entrega será feita única e exclusivamente via eDisciplinas, até a data final marcada. Deverá ser postado no eDisciplinas um arquivo zip, tendo como nome seu número USP:

`número_usp.zip`

Dentro do zip deve constar todos os arquivos *.java* envolvidos neste EP. Não esqueça de preencher o cabeçalho constante do arquivo Banco.java, com seu nome, número USP, etc.

A responsabilidade de postagem é exclusivamente sua. Por isso, submeta e certifique-se de que o arquivo submetido é o correto (fazendo seu download, por exemplo). Problemas referentes ao uso do sistema devem ser resolvidos com antecedência.

3 Avaliação

Para avaliação, serão observados os seguintes quesitos:

1. Documentação: se há comentários explicando o que se faz nos passos mais importantes e para que serve o programa (tanto o método quanto o programa em que está inserido);
2. Apresentação visual: se o código está legível, indentado, etc;
3. Corretude: se o programa funciona.

Além disso, algumas observações pertinentes ao trabalho, que influem em sua nota, são:

- Este exercício-programa deve ser elaborado individualmente;
- Não será tolerado plágio, em hipótese alguma;
- Exercícios com erro de sintaxe (ou seja, erros de compilação), receberão nota ZERO.