MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO | 2º ANO 

EIC0013 | ALGORITMOS E ESTRUTURAS DE DADOS | 2018-2019 – 1º SEMESTRE

CI1 Parte prática. Duração: 2h00m

Pretende-se implementar um sistema de gestão de contas de uma instituição bancária. As contas bancárias (classe **Account**) são identificadas por nome e código de identificação do titular (*holder e codHolder*) e pelo número de conta bancária internacional (cod*IBAN*). Existem três tipos de contas: contas de depósito (**CheckingAccount**), contas poupança (**SavingsAccount**), e contas a prazo (**TimeAccount**). A conta a prazo (**TimeAccount**) é um tipo de conta poupança (**SavingsAccount**) em que apenas o juro pode ser levantado.

Todas as contas têm um saldo (*balance*). As contas poupança têm adicionalmente uma taxa de juro (*rate*) e as contas de depósito têm um limite inferior para movimentos (*limit*).

A classe ***BankOfficer*** identifica um funcionário do banco e inclui um identificador (*id*), o nome do bancário (*name*) e as contas bancárias que tem de gerir (*myAccounts*).

A classe ***Bank*** identifica uma filial bancária e inclui informação sobre os bancários que aí trabalham (*bankOfficers*).

As classes **Bank, BankOfficer, Account, CheckingAccount, SavingsAccount,** e **TimeAccount** estão parcialmente definidas a seguir.

Nota: NÃO PODE acrescentar membros-dado nas classes **Account, SavingsAccount, CheckingAccount** e **TimeAccount**, para além dos que já se encontram declarados**.**

class Account {

string holder;

//…

public:

Account(string hold, string codeH, string codeA, double bal);

// …

};

class CheckingAccount: public Account { double limit;

public:

CheckingAccount(string hold, string codeH, string codeA, double bal, double lim); //…

};

class SavingsAccount: public Account { double rate;

public:

SavingsAccount(string hold, string codeH, string codeA, double bal, double pct); //…

};

class TimeAccount: public SavingsAccount { double limit;

public:

TimeAccount(string hold, string codeH, string codeA, double bal, double pct); //…

};

class BankOfficer {

unsigned int id;

string name;

vector<Account \*> myAccounts; public:

BankOfficer();

// …

};

class Bank {

vector<Account \*> accounts; vector<BankOfficer> bankOfficers; public:

Bank();

// …

};

ANA PAULA ROCHA, LUÍS PAULO REIS, LILIANA FERREIRA, PEDRO NOGUEIRA 29/OUTUBRO/2018 **|** PÁG. 1 / 3

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO | 2º ANO EIC0013 | ALGORITMOS E ESTRUTURAS DE DADOS | 2018-2019– 1º SEMESTRE 

**a)** [**2.5 valores**] Implemente a função template seguinte:

*unsigned int numberDifferent (const vector<T> & v1)*

Esta função retorna o número de elementos diferentes existentes no vetor v1, passado por argumento. Considere que v1 é um vetor não ordenado.

**b)** [**2 valores**] Implemente, nas classes que considerar necessário, o membro-função que efetua o cálculo do valor das comissões de depósito, em cêntimos, para uma conta:

*double getCharge () const*

Esta função retorna o valor da comissão de depósito de uma conta bancária, sabendo que: ● Para as contas de depósito (**CheckingAccount**) este valor depende do limite (*limit*) definido na criação da conta:

0.50€, se *limit* <= 500€; 0.25€, se 500 < *limit* <= 1000€ ; 0€, se *limit* > 1000€

● Para todas as contas poupança (**SavingsAccount**) depende da taxa de juro:

0.80€, se a percentagem da taxa de juro (*rate\*100*) > 2.0; 0€ caso contrário

**c)** [**3 valores**] Implemente nas classes que considerar necessário, o membro-função *double getWithdraw () const*

Esta função retorna o valor disponível para levantamento na conta respetiva. Relembre que na conta a prazo (**TimeAccount**) apenas o juro (*balance\*rate*) pode ser levantado.

Implemente ainda na classe **Bank** o membro-função:

*double getWithdraw(string cod1) const*

Esta função retorna o montante disponível para levantamento em todas as contas do cliente de código *cod1*.

**d)** [**2.5 valores**] Implemente na classe **Bank** o membro-função:

*vector<Account \*> removeBankOfficer(string name)*

Esta função remove do vetor bankOfficers o bancário de nome name, retornando as contas geridas por esse bancário (myAccounts). Se não existe nenhum bancário de nome name, a função retorna um vetor vazio.

ANA PAULA ROCHA, LUÍS PAULO REIS, LILIANA FERREIRA, PEDRO NOGUEIRA 29/OUTUBRO/2018 **|** PÁG. 2 / 3

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO | 2º ANO EIC0013 | ALGORITMOS E ESTRUTURAS DE DADOS | 2018-2019– 1º SEMESTRE 

**e)** [**2.5 valores**] Implemente na classe **BankOfficer** o construtor:

*BankOfficer(string name)*

Este construtor recebe como argumento o nome (*name*) do bancário. Atribui a cada objeto bancário criado um novo identificador (*id*), sequencial e incremental. O primeiro bancário a ser criado terá *id* igual a 1, o segundo *id* igual a 2 e assim sucessivamente. O vetor das contas geridas pelo bancário (*myAccounts*) está vazio.

**f)** [**2.5 valores**] Implemente na classe **BankoOfficer** o operador >. Um bancário *b1* é maior que outro bancário b2, se o número de contas que titula (*myAccounts*) é superior. Se o número de contas que os dois bancários titulam for igual, *b1* é maior que *b2* se o seu nome for superior alfabeticamente.

**g)** [**2.5 valores**] Implemente na classe **Bank** o membro-função:

*const BankOfficer &addAccountToBankoOfficer(Account \*ac, string name)*

Esta função adiciona a conta *ac* ao vetor de contas bancárias que o bancário de nome *name* é responsável por gerir (*myAccounts*) e retorna uma referência para esse bancário. Se não existir um bancário de nome *name*, a função deve lançar a exceção *NoBankOfficerException*. Esta classe exceção deve incluir o membro-função *getName()* que retorna o nome do bancário não existente.

**h)** [**2.5 valores**] Implemente na classe **Bank** o membro-função

*void sortAccounts()*

Esta função ordena os elementos do vector *accounts*. As contas devem ficar ordenadas por ordem crescente de saldo (*balance*) e, para o mesmo saldo, por ordem crescente de código (*codeIBAN*).

ANA PAULA ROCHA, LUÍS PAULO REIS, LILIANA FERREIRA, PEDRO NOGUEIRA 29/OUTUBRO/2018 **|** PÁG. 3 / 3