SSI2 - IFRS Porto Alegre
Linguagem de Programação II
Trabalho Final do Componente
Curricular Desenvolvimento de uma
Aplicação em Java

Eduardo Luis Dovigi Reis

Descrição do Diagrama de Classes (v1.3)

Diagrama de Classes: descrição das classes, atributos e métodos

Classes de modelo

Conta (Classe)

Uma conta possui os atributos agência, numeroConta, clientes, operacoes, saldo, tipoConta. Todos obrigatórios. Como regra, quando uma nova conta é aberta, é obrigatório que ocorra um depósito de no mínimo 50 reais.

buscarNumero(): retorna o número da conta.

buscarTipoConta(): retorna o tipo da conta.

consultarSaldo(): retorna o saldo disponível na conta no momento da consulta.

gerarExtrato(): retorna uma lista de operações realizadas na conta.

realizarTransferencia(ag: String, conta: String, valor: BigDecimal): envia uma quantia em dinheiro para a conta informada, caso o valor contido na conta de origem não seja inferior ao valor de transferência. Se tudo der certo retorna uma Operacao contendo todas as informações, senão retorna uma Operacao com o status false, ou seja, não concluída.

Obs: Caso a conta que deseja fazer a transferência for do tipo Salario, então a operação não deve ser concluída, pois a conta Salário não permite realizar operações de transferência.

pagarBoleto(linhaDigitavel: String): a operação de pagamento de boleto recebe como parâmetro a linha digitável do boleto, neste String, virão as informações de agenciaBeneficiário, contaBeneficiário,

dataVencimento e valor, separados por “.”. Será feito o split deste String e cada campo do Boleto será preenchido com as informações. No fim, será retornado um Boleto com todas as informações.

Obs: Caso a conta que deseja fazer o pagamento seja do tipo Salario, então a operação não deve ser concluída, pois a conta Salário não permite realizar operações de pagamento.

toString: retorna a representação de cada campo da classe como em um único String.

TipoConta(Enum)

Os tipos de conta possuem os seguintes valores: CORRENTE, POUPANCA, SALARIO. Este enum complementa a classe Conta.

Operacao(Classe)

Uma operação possui os atributos descricao, tipoOperacao, taxa, dataHora, valor, status, conta, numeroContaDestino. Todos obrigatórios.

buscarDescricao(): retorna a descrição da operação.

buscarDataHora(): retorna a data e hora da operação.

buscarStatus(): retorna o status da operação.

definirStatus(status: Boolean): recebe um status como parâmetro que será definido como o status da operação.

buscarTipoOperacao(): retorna o tipo da operação.

definirConta(conta: Conta): recebe uma conta como parâmetro que será definida como a conta de origem da operação.

buscarConta(): retorna a conta de origem da operação.

buscarValor(): retorna o valor da operação.

definirValor(valor: BigDecimal): recebe um valor como parâmetro que será definido com o valor da operação.

definirTipoOperacao(tO: TipoOperacao): recebe um tipo de operação por parâmetro que será definido como o tipo da operação.

toString(): retorna a representação de cada campo da classe como em um único String.

Boleto (Classe)

Um boleto possui os atributos linhaDigitavel, agenciaBeneficiario, contaBeneficiario, dataVencimento e herda todos os atributos de Operacao.

toString(): retorna a representação de cada campo da classe como em um único String.

TipoOperacao (Enum)

Os tipos de operação aceitos são: TRANSFERENCIA, DEPOSITO e PAGAMENTO. O enum TipoOperacao complementa a classe Operacao.

Agencia (Classe)

Uma agência possui os atributos numeroAgencia e endereço. Esta classe complementa a classe Conta.

Endereco (Classe)

Um endereço é composto por logradouro, bairro, cidade, uf, cep e país.

UF (Enum)

O Enum UF é composto pelas siglas de todas as unidades federativas do Brasil: RO, AC, AM, RR, PA, AP, TO, MA, PI, CE, RN, PB, PE, AL, SE, BA, MG, ES, RJ, SP, PR, SC, RS, MS, MT, GO. O Enum UF complementa a classe Endereco.

Cliente (Classe)

Um cliente possui os atributos nome, cpf, email, telefone, endereco, renda, contas e chaves.

buscarNome(): retorna o nome do cliente;

buscarCpf(): retorna o cpf do cliente;

definirTelefone(telefone: String): recebe um telefone como parâmetro que será definido como o telefone do cliente.

definirRenda(renda: BigDecimal): recebe uma renda como parâmetro que será definida como a renda do cliente.

definirConta(conta: Conta): recebe uma conta como parâmetro que será adicionada à lista de contas do cliente.

buscarContas(): retorna a lista de contas do cliente.

extrairNumeroConta(contas: List<Conta>): recebe uma lista de contas como parêmetro, esta lista será iterada e retornará apenas o número das contas do cliente.

cadastrarChave(c: String, tC: TipoChave): recebe como parâmetro o valor da chave desejada e o tipo de chave. O método retorna um objeto Pix.

Pix (Classe)

A classe pix possui os atributos chave e tipoChave. A classe Pix complementa a classe Cliente.

toString(): retorna a representação de cada campo da classe como em um único String.

TipoChave (Enum)

O tipo de chave possui os valores CPF, CELULAR e EMAIL. O enum TipoChave complementa a classe Pix.

Administrador (Classe)

Um administrador possui os atributos nome, codigoId e todos os atributos de Usuario.

buscarNome: retorna o nome do Administrador

toString(): retorna a representação de cada campo da classe como em um único String.

Usuario (Classe)

A classe Usuario possui os atributos login, senha. Ela possui como descendentes as classes Cliente e Adminsitrador.

buscarLogin(): retorna o login do Usuário.

buscarSenha(): retorna a senha do Usuário.

Classes de Serviço

LoginService (Classe)

A classe LoginService realiza a operação de login tanto de usuários clientes quanto administradores.

fazerLoginCliente(login: String, senha: String): Recebe como parâmetro o login e senha de um cliente, estas informações serão validadas e caso os valores informados sejam iguais aos de algum cliente no banco de dados, o acesso é permitido. Por fim, é retornado um objeto contendo as informações do cliente que está acessando o sistema.

fazerLoginAdministrador(login: String, senha: String): Recebe como parâmetro o login e senha de um administrador, estas informações serão validadas e caso os valores informados sejam iguais aos de algum administrador no banco de dados, o acesso é permitido. Por fim, é retornado um objeto contendo as informações do administrador que está acessando o sistema.

AdminService (Classe)

A classe AdminService contém os métodos que permitem com que um administrador consiga manipular informações de clientes.

listarClientes(): Lista todos os clientes cadastrados na base de dados.

buscarCliente(cpf: String): Busca um cliente no banco de dados pelo seu cpf de cadastro.

atualizarCliente(cpf: String, c: Cliente): Busca um cliente pelo cpf, caso encontre, atualiza uma ou mais informações no cliente encontrado e então o modifica com as informações passadas por parâmetro.

BancoDeDados (Classe)

A classe BancoDeDados possui os atributos enderecos, agencias, clientes, administradores e contas. Ela foi utilizada para simular um banco de dados real, ela guarda listas de algumas das classes modelo contidas no sistema.

definirEnderecos(): possui a criação manual de vários endereços.

buscarEnderecos(): retorna a lista de todos os endereços guardados no banco de dados.

definirAgencias(): possui a criação manual de várias agências.

buscarAgencias(): retorna a lista de todas as agências guardadas no banco de dados.

definirClientes(): possui a criação manual de vários clientes.

buscarClientes(): retorna a lista de todos os clientes guardados no banco de dados.

definirContas(): possui a criação manual de várias contas.

buscarContas(): retorna a lista de todas as contas guardadas no banco de dados.

buscarClientesComConta(): retorna a lista de todos os clientes, assim como no método anterior, porém possuindo as contas de cada cliente.

definirAdmnistradores(): possui a criação manual de vários administradores.

buscarAdministradores(): retorna a lista de todos os administradores no banco de dados.