UAST
Unidade Acadêmica de Serra Talhada - PE
UFRPE
Desde 2006

Bacharelado em Sistemas de Informação

MPOO

Site: https://sites.google.com/site/profricodemery/mpoo

Site: http://ava.ufrpe.br/

Disciplina: Modelagem e Programação Orientada a Objetos (MPOO)

Profº: Richarlyson D'Emery

LISTA DE EXERCÍCIOS V

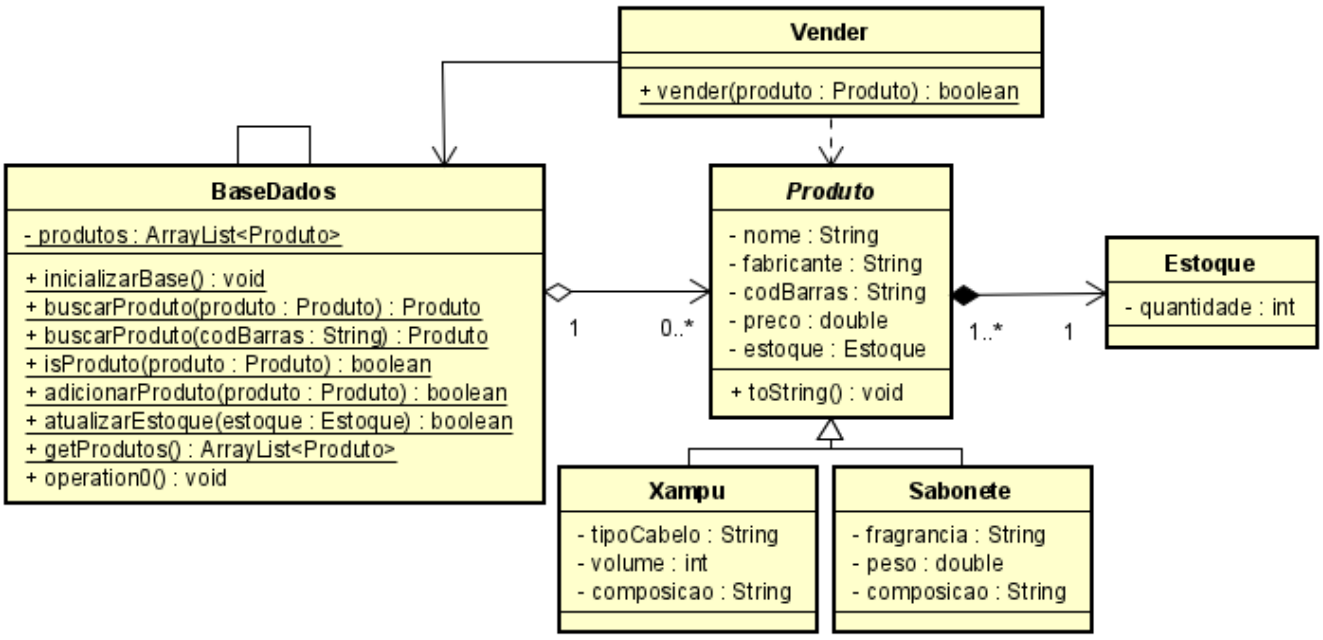
Leia atentamente as instruções gerais:

- No Eclipse crie um novo **projeto chamado br.edu.mpoo.listaV.SeuNomeSobrenome**, o qual deverá ter **pastas de pacotes** para cada questão: questao1, questao2, e assim sucessivamente, contendo todas as respostas da lista.
- A lista envolve questões práticas, então deverão ser entregues no AVA os códigos-fonte (projeto completo). Em caso de diagramas, você poderá salvar o arquivo também na pasta correspondente do projeto.
- A entrega da lista compõe sua frequência e avaliação na disciplina.

Mão na Massa!

Você, aluno de MPOO, está experienciando situações-problemas do universo de desenvolvimento de software e começará a ser desafiado a solucionar problemas a partir de conhecimentos de Programação e Orientação a Objetos.

1) Observe o diagrama de classes abaixo e as regras de negócio definidas:



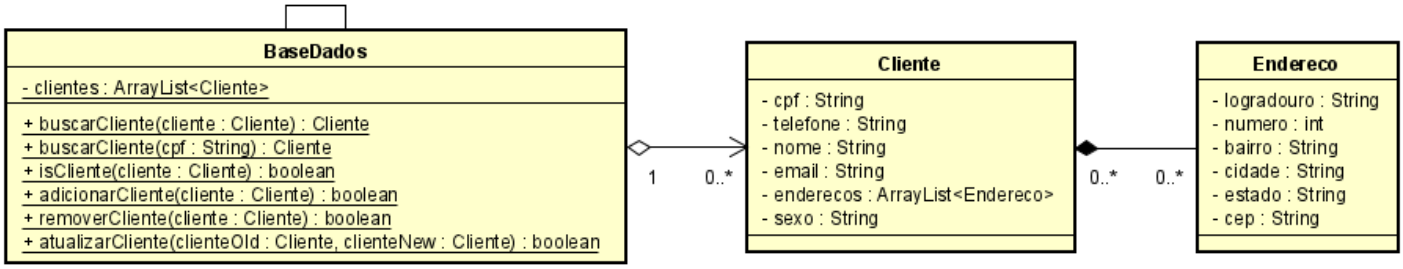
São regras de negócios:

- RN01 – um produto é identificado pelo codBarras;
- RN02 – um produto só poderá ser cadastrado uma única vez;
- RN03 – a quantidade de produto poder ser vista pelo atributo quantidade;
- RN04 – atualizarEstoque atualiza a quantidade de produtos disponíveis;
- RN05 – um produto quando vendido tem seu estoque atualizado;

A partir dessas características, implemente o sistema em Java. Demonstre em uma aplicação:

- A criação de diversos produtos;
- A atualização de estoque dos produtos criados,
- A venda de alguns produtos;

2) Um contratante solicitou a empresa MPOOSoftware LTDA um sistema de cadastro de clientes. O Scrum Master de MPOOSoftware LTDA solicitou a um de seus programadores (de codinome *mustela putórius furo* – "O Furão") que resolvesse essa demanda. Para isso apresentou o diagrama de classes abaixo produzido pela equipe de modelagem.



Antes de responder, analise as seguintes regras de negócios:

- RN01 – um cliente é identificado pelo seu cpf;
- RN02 – um cliente só poderá ser cadastrado uma única vez;
- RN03 – um cliente poderá ter diversos endereços, desde que contenham todos os dados válidos;
- RN04 – um cliente quando removido do sistema tem seus endereços removidos;
- RN05 – um cliente não poderá ter um mesmo endereço repetido;
- RN06 – diversos clientes podem ter um mesmo endereço;

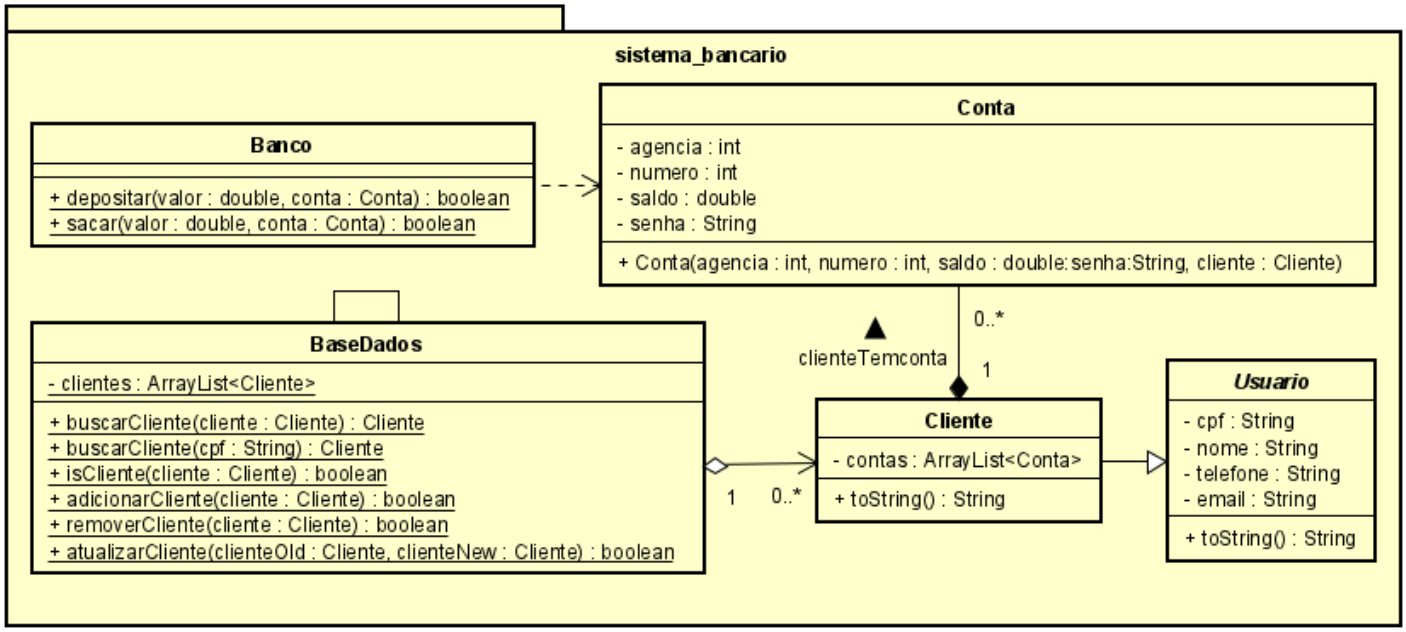
Ilustre em uma aplicação:

(i) o cadastro de dois clientes em que o primeiro cliente possui um endereço e o segundo dois endereços.
(ii) São endereços:
- Rua dos Cactos, n. 100, Cactolândia, Serra Talhada-PE, CEP 56970-000
- Av. Gregório Ferraz Nogueira, n. 20, José Tomé de Souza Ramos, Serra Talhada-PE, CEP 56909-535

Desafio

3) **(Desafio)** Após a entrega da primeira versão do sistema, o contratante solicitou a **inclusão de uma nova funcionalidade: "*o sistema deveria ter uma base de endereços independente de seus clientes*"**. O Scrum Master de MPOOSoftware LTDA solicitou para "O Furão" que aproveitasse a primeira versão do sistema de maneira a economizar tempo de programação. Entretanto "O Furão" não soube atender a essa demanda o que levou a empresa de desenvolvimento de software a promover um *hackathon* para a contratação de um novo funcionário. Mostre que você é um exímio programador e **apresente uma solução contendo um diagrama de classes e a codificação Java** para garantir essa vaga!

4) Implemente em Java a codificação para o sistema representado no diagrama de classes e abaixo:



Antes de responder, analise as seguintes descrições:

- Cliente e Conta representam um caso de composição;
- As operações no saldo de uma conta são dadas pelos comportamentos de sacar e depositar. O saque só deve ser realizado se o valor do saque for igual ou superior ao valor do saldo;
- Falhas de saque ou depósito devem exibir em console as mensagens, respectivamente: “Erro ao efetuar o saque” ou “Erro ao efetuar o depósito”;
- A consulta dos dados de um usuário deve utilizar o método toString();
- Na classe BaseDados, os clientes deverão ser manipulado pelos métodos definidos.
    - Um cliente só deve ser adicionado se não existir na base. Use como critério de existência o cpf do cliente.
    - A verificação da existência de um cliente é dada por `boolean isCliente (Cliente cliente) {}`. Utilize o cpf como critério de existência.
    - Observe as diversas maneiras para buscar um cliente: ou pelo cpf ou por uma instância válida de cliente.
- As contas do sistema bancário devem ser *auto-increment*.

Crie uma App, em que:

- A base possui um cliente com duas contas e outro cliente com apenas uma conta;
- Exibe os dados dos clientes;
- Realiza o saque de uma conta e deposita a quantia em outra conta; e
- Exibe em console os valores antes e depois de cada operação.