

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

<http://ava.ufrpe.br/>

<https://sigs.ufrpe.br/sigaa/ava/index.jsf>

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.

Fique atento!

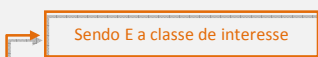
Prezado aluno, esta é a lista de exercícios é relativa ao assunto de "Tipos e estruturas dinâmicas de dados", viabilizando relacionamentos de cardinalidade múltipla dinâmica.

Saiba Mais!

Na API de Java há diversas collections que fornecem estruturas de dados predefinidas, nas quais disponibilizam métodos que criam, armazenam, organizam e recuperam dados. São contêineres que armazenam dados na memória RAM, por isso é necessário persistir esses dados em um contêiner que utiliza memória ROM.

Collections podem se dá em termos de interfaces e classes que realizam essas interfaces, todas presentes em `java.util`. A mais popular entre elas é o `ArrayList`, mas diferentemente de um array convencional seu tamanho pode ser modificado automaticamente durante a execução de um programa, ou seja, podendo alocar ou liberar memória conforme a conveniência da situação.

Desde a atualização JDK v. 1.5 passou a permitir a criação de lists de classes genéricas.


`ArrayList<E> itens = new ArrayList<E>();`

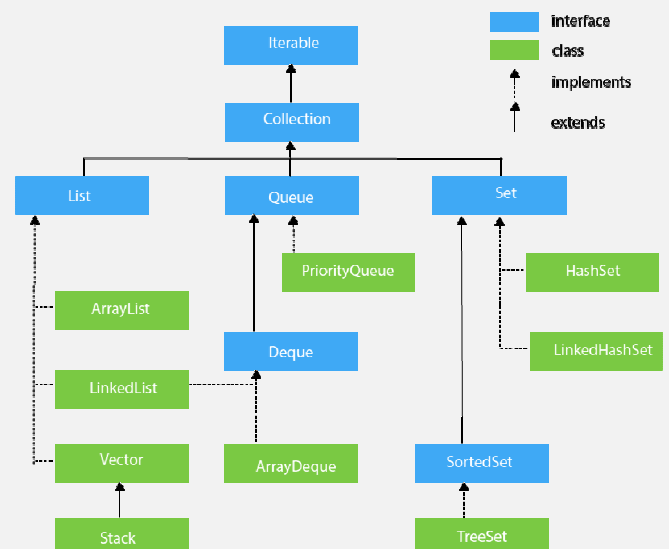
Logo, só podem lidar com objetos e, portanto, **não** é possível definir tipos primitivos: `ArrayList<int>`, `ArrayList<double>`, `ArrayList<float>` ou `ArrayList<char>`.

Uma alternativa é a utilização de classes Wrapper que empacotam os tipos primitivos, por exemplo: `Integer` para `int`, `Double` para `double`, `Long` para `long`, `Boolean` para `boolean`, `Char` para `char`, etc. Então: `ArrayList<Integer>`, `ArrayList<Double>`, `ArrayList<Float>` ou `ArrayList<Char>`.

E assim, como para tipos primitivos, tentar atribuir um tipo `Double` a um `ArrayList<Integer>` acarretará em um erro de sintaxe. E se realizada uma atribuição em tempo de execução, acarretará em uma Exceção (*tema de uma próxima aula!*):

**Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The method add(Integer) in the type ArrayList<Integer> is not applicable for the arguments (Double)**

Investigue mais sobre `ArrayList` em <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>



Você Sabia?

O mais comum é se pensar em varredura de arranjo utilizando contadores:

```
for (int cont=0; cont<itens.size(); cont++){  
    System.out.println(itens.get(cont).getAtr());  
}
```

objeto de interesse

atributo de interesse

Entretanto, em Java podemos utilizar o for de uma maneira especial, chamada de **enhanced for**, ou popularmente **foreach**. Lembrando que **foreach** não existe no Java como **comando**, mas como um caso especial do **for** :

```
for (E itemCurrent: itens) {    System.out.println (itemCurrent); }
```

Objeto de interesse. A cada laço assumirá uma instância armazenada no ArrayList.

Fique por Dentro!

- 1) Em Java podemos utilizar a estrutura de dados lista pela classe ArrayList, mas existem outras formas em Java de armazenar uma lista das quais devem ser apresentadas e em que situações são utilizadas. Qual a melhor do ponto de vista de custo computacional?
- 2) Analise o código da classe SaibaMais

```
import java.util.ArrayList;  
  
public class SaibaMais {  
    public static void main(String[] args) {  
        String saida1 = "Aprendendo";  
        String saida2 = "a";  
        String saida3 = "Programar";  
        String saida4 = "Java";  
  
        ArrayList<String> saidas = new ArrayList<>();  
        saidas.add(saida1);  
        saidas.add(saida2);  
        saidas.add(saida3);  
        saidas.add(saida4);  
        ///?  
    }  
}
```

Responda:

- 2.1) Qual a saída do programa quando substituímos `///?` por:

2.1.1) `System.out.println(saidas);`

2.1.2) `saidas.forEach(saida -> { System.out.print(saida + " "); });`

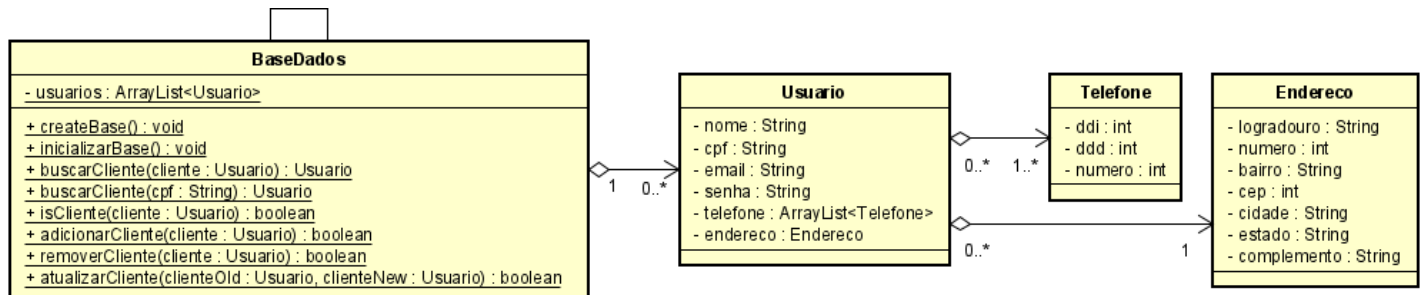
- 2.2) Pesquise qual conceito é pertencente ao Java 8 e está sendo utilizado no método nativo `forEach` da questão 2.1.2)

Desafio: 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.



3) Codifique em Java o seguinte diagrama de classes:



São regras de negócios:

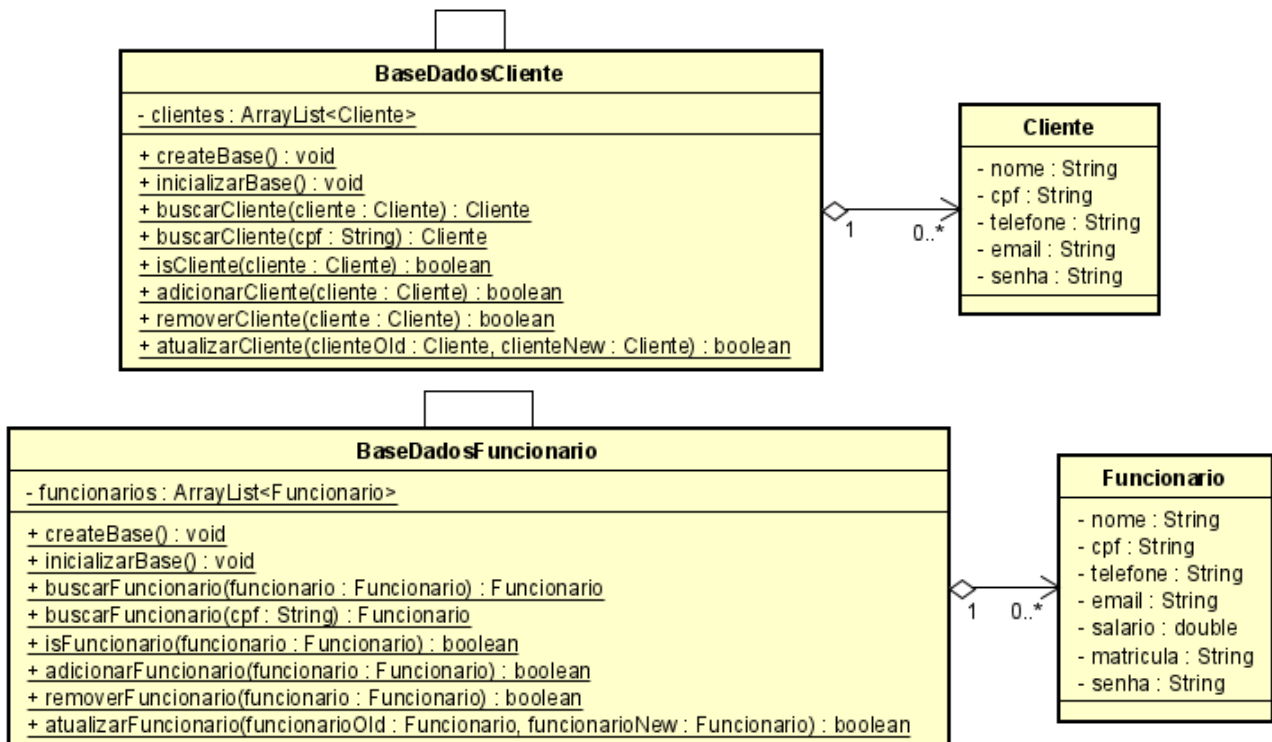
- RN01 – um usuário é identificado pelo seu cpf;
- RN02 – um usuário só é adicionado na base se tiver um cpf válido;
- RN03 – um usuário só poderá ser cadastrado uma única vez;
- RN04 – a codificação deve aproveitar comportamentos já definidos, evitando a duplicidade de programação;

3.1) Crie uma aplicação que contém os seguintes usuários:

- Usuário1: nome: “José Santos”, cpf: 941.860.760-30, telefone: +55 (81) 99999-0000, e-mail: josesantos@gmail.com, senha: jose123, domiciliado à rua Av. dos Cactos, n. 100, bairro Cactolândia, CEP: 56970-000, Serra Talhada-PE.
- Usuário2: nome: “Maria Silva”, cpf: 575.373.110-4830, telefone: +55 (81) 99999-0001, e-mail: mariasilva@gmail.com, senha: mariamaria, domiciliada à Rua Flores, n. 50, bairro Encruzilhada, CEP: 52041-430, Recife-PE.
- Usuário3: nome: “João Mamão”, cpf: 080.075.880-35, telefone1: +55 (87) 99999-0002, telefone2: +55 (87) 99999-0003 e-mail: jmamao@gmail.com, senha: mamãoDocE, domiciliado à Rua Mamão Doce, n. 100, bairro Saudade, CEP: 56870-000, Triunfo-PE.

3.2) Do ponto de vista de segurança de dados, o sistema apresenta falhas. Justifique os porquês dessas falhas e como poderiam ser solucionadas. Reflita essa solução no diagrama UML e na codificação Java.

4) Um contratante solicitou a empresa MPOOSoftware LTDA a atualização de um sistema de cadastro. 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 uma evolução dos diagramas de classes da empresa (vide questão 2) da Lista de Exercícios IV):



//continua....

São regras de negócios:

- RN01 – um cliente ou funcionário é identificado pelo seu cpf;
- RN02 – um cliente ou funcionário só é adicionado em uma base se tiver um cpf válido;
- **RN03 – um cliente ou funcionário só poderá ser cadastrado uma única vez;**
- RN04 – para entrar no sistema um cliente ou funcionário deverá informar seu login (email ou cpf) e senha;
- RN05 – a senha de um cliente ou funcionário deverá ser criada na primeira utilização do sistema;
- RN06 – a senha de um cliente ou funcionário deverá ter pelo menos 6 dígitos.
- **RN07 – a codificação deve aproveitar comportamentos já definidos, evitando a duplicidade de programação;**

Assumindo o fato de que:

- A base de clientes contém:
 - Cliente1: nome: “José Santos”, cpf: 941.860.760-30, telefone: (81) 99999-0000, e-mail: josesantos@gmail.com
 - Cliente2: nome: “Maria Silva”, cpf: 575.373.110-4830, telefone: (82) 99999-0001, e-mail: mariasilva@gmail.com
 - Cliente3: nome: “João Mamão”, cpf: 080.075.880-35, telefone: (87) 99999-0002, e-mail: jmamao@gmail.com
- A base de funcionários contém:
 - Funcionario1: nome: “Severino de Jesus”, cpf: 064.749.190-78, telefone: (81) 99999-1111, e-mail: sevjesus@gmail.com, salario: R\$ 1500,00, matricula: “func001”
 - Funcionario2: nome: “Maria Silva”, cpf: 575.373.110-4830, telefone: (82) 99999-0001, e-mail: mariasilva@gmail.com, salario: R\$ 1500,00, matricula: “func002”
 - Funcionario3: nome: “José Santos”, cpf: 941.860.760-30, telefone: (81) 99999-0000, e-mail: josesantos@gmail.com, salario: R\$ 3000,00, matricula: “func001”

Codifique em Java as seguintes GUI's do sistema:

- Para o usuário realizar o *login* no sistema deve informar seu login e senha (Fig. 1a). Caso algum campo não seja informado ao Entrar, devem-se exibir as informações para campo obrigatório (Fig. 1b ou Fig. 1c).

A Java Swing window titled "Login" with standard window controls. It contains two text input fields: "Login:" and "Senha:". Below the fields are two buttons: "Entrar" and "Sair".

Fig. 1a

A Java Swing window titled "Login" with standard window controls. The "Login:" field contains the text "ricodemery". The "Senha:" field is empty. Below the fields is a red error message: "(*) Informe o campo obrigatório". There are "Entrar" and "Sair" buttons.

Fig. 1b

A Java Swing window titled "Login" with standard window controls. Both the "Login:" and "Senha:" fields are empty. Each field has a red asterisk (*) to its right. Below the fields is a red error message: "(*) Informe o campo obrigatório". There are "Entrar" and "Sair" buttons.

Fig. 1c

- Quando se tratar de primeiro acesso, o sistema deve tratar a regra de negócio RN04 através da GUI para cadastrar senha (Fig. 2a). Observe a necessidade de informar campo obrigatório quando login (Fig. 2b) ou login e senha (Fig. 2c) não é (são) informado(s) após clicar no botão Cadastrar. O campo login é automaticamente carregado a partir da tela Login (Fig. 2a)

A Java Swing window titled "Login" with standard window controls. The "Login:" field contains the text "ricodemery". The "Senha:" field is empty. Below the fields is a button labeled "Cadastrar".

Fig.2a

A Java Swing window titled "Login" with standard window controls. The "Login:" field contains the text "ricodemery". The "Senha:" field is empty. Below the fields is a red error message: "(*) Informe o campo obrigatório". There is a "Cadastrar" button.

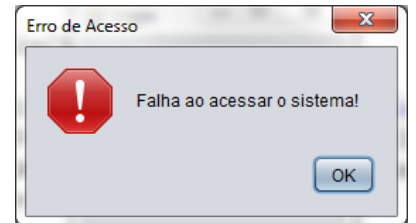
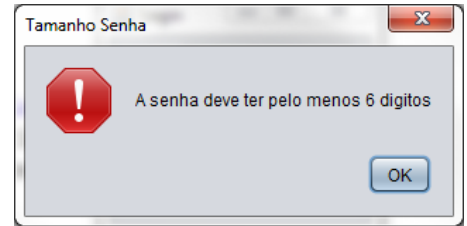
Fig. 2b

A Java Swing window titled "Login" with standard window controls. Both the "Login:" and "Senha:" fields are empty. Each field has a red asterisk (*) to its right. Below the fields is a red error message: "(*) Informe o campo obrigatório". There is a "Cadastrar" button.

Fig. 2c

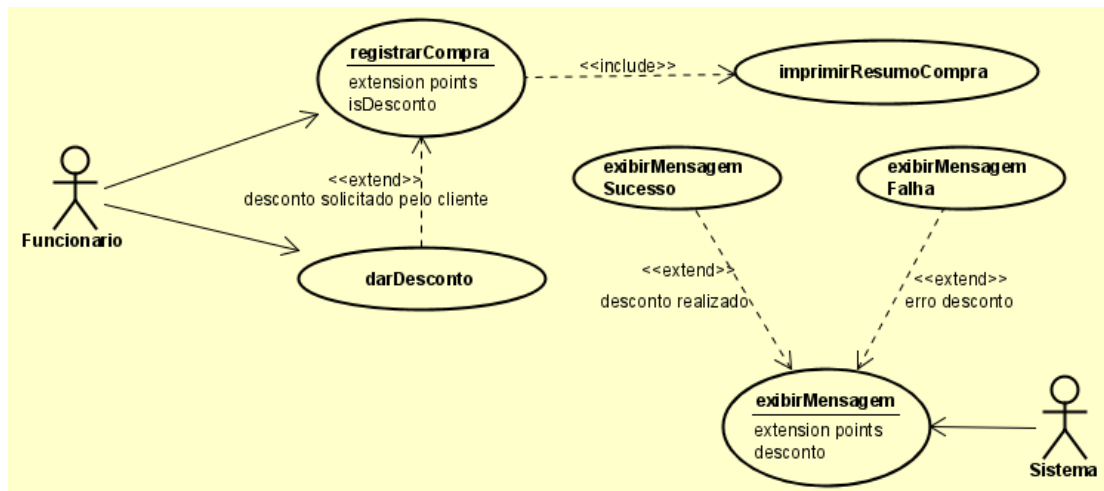
//continua....

- Ao tentar cadastrar uma senha de maneira que a quantidade de dígitos seja inferior a 6, deve-se exibir a mensagem de erro:
- Quando um cliente ou funcionário não cadastrado tentar realizar o login ou cadastrar uma senha no primeiro acesso, o sistema deve exibir as mensagens de erro:

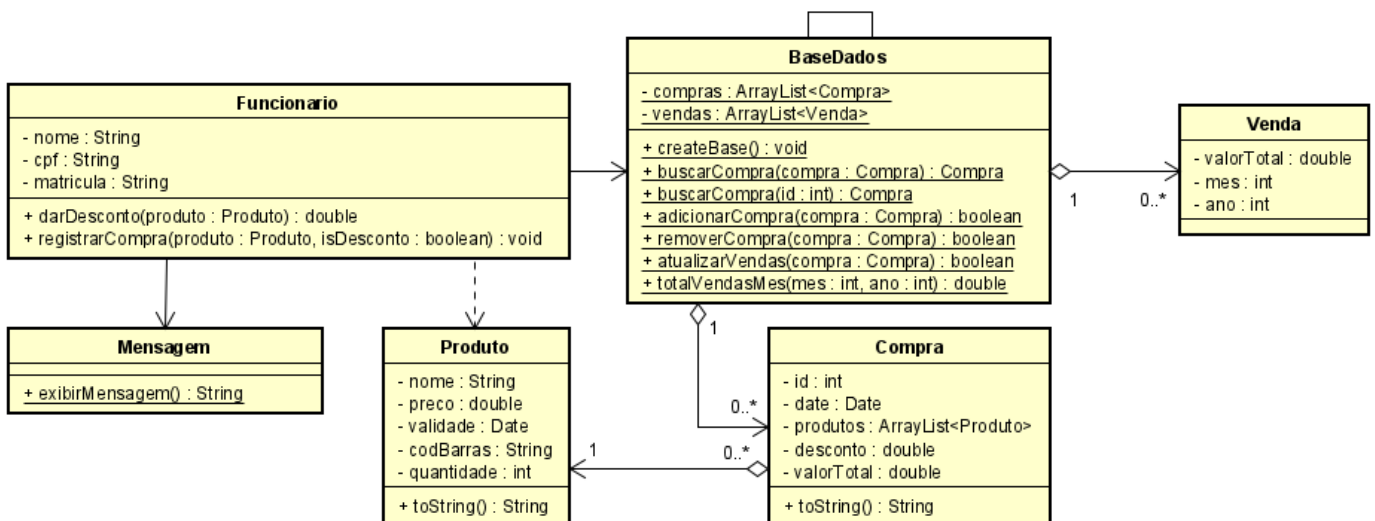


Desafio!

- 5) **(Desafio)** Uma Empresa solicitou a um de seus programadores (de codinome *mustela putórius furo* – “O Furão”) que resolvesse uma demanda de funcionalidades de seu sistema criada pelo setor de venda da empresa. Cada ator detém comportamento(s) específico(s) conforme sua função ilustrada no diagrama de use case abaixo:



Sabe-se que o sistema da empresa deve ser codificado em Java e também deve refletir o diagrama de classes abaixo:



Antes de responder, analise as situações contendo as seguintes regras de negócios:

- RN01 – uma compra possui id autoincrementável;
- RN02 – a data de uma compra utiliza a data do sistema de acordo com o momento em que é realizada;
- RN03 – toda compra é registrada na base;
- RN04 – toda nova compra deve atualizar o valor de vendas de um mês;
- RN05 – toda compra cancelada deve atualizar o valor de vendas de um mês;
- RN06 – um funcionário só pode executar os comportamentos definidos;
- RN07 – a codificação deve aproveitar comportamentos já definidos, evitando a duplicidade de programação;
- RN08 – uma compra tem identificação autoincrementável;
- RN09 – um desconto só é atribuído a uma compra se for solicitado verbalmente por um cliente no momento do registro de uma compra (registrado no atributo desconto de Compra);
- RN10 – um desconto possui valor fixo de 10%;
- RN11 – Um desconto só poderá ser executado por um funcionário válido; e
- RN12 – Em nenhuma hipótese deve-se alterar o valor de um produto.

5.1) Ilustre em uma aplicação: (i) a compra de um produto em que o cliente solicita verbalmente um desconto; (ii) a compra de um produto em que o cliente não solicita um desconto; (iii) uma compra que contém 3 produtos em que o cliente solicita verbalmente um desconto; (iv) uma compra que contém 3 produtos em que o cliente não solicita um desconto; e (v) a atualização das vendas para uma compra realizada e para outra cancelada. Em todos os casos deve-se exibir em console o resumo da compra e das vendas.

5.2) Qual a problemática envolvida na operação `darDesconto(produto:Produto):double` ? Reflita no diagrama de classes e na codificação uma solução que minimize essa problemática.

Saiba Mais

Como vimos, o tamanho de um `Collection` pode ser modificado automaticamente durante a execução de um programa, ou seja, podendo alocar ou liberar memória conforme a conveniência da situação. Mas vimos que trata-se de memória RAM e, portanto, leva ao seguinte questionamento: *“Minha aplicação Java pode consumir quanto da memória do meu computador?”* e *“Como configurar o espaço de memória da JVM?”*.

Para saber como responder a essas perguntas visite:

- <https://www.devmedia.com.br/configurando-o-espaco-de-memoria-das-areas-heap-e-perm-da-jvm/37161> e
- https://docs.oracle.com/cd/E23943_01/web.1111/e13814/jvm_tuning.htm#PERFM150
- https://wiki.eclipse.org/FAQ_How_do_I_increase_the_heap_size_available_to_Eclipse%3F