Ciência da Computação

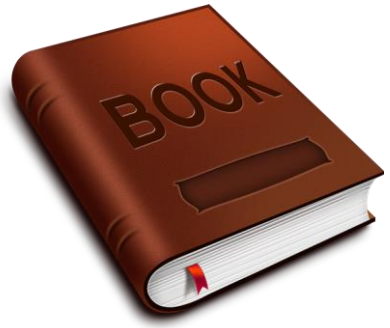
Programação Orientada a Objetos

Prof. Luciano Rodrigo Ferretto

🎯 **Desafio: Desenvolva um sistema de gestão de biblioteca em Java** que deve incluir as seguintes classes:

1. 📖 **Livro**: Representa os livros da biblioteca. Esta classe deve conter atributos como:
    - **Título**
    - **Autor**
    - **Ano de Publicação**
    - **Número de Páginas**
2. 🏛️ **Biblioteca**: Representa a própria biblioteca. Esta classe deve ter métodos para:
    - **Adicionar um livro** ao acervo ➕📚
    - **Remover um livro** do acervo ❌📚
    - **Buscar um livro** pelo título 🔍📖
    - **Listar todos os livros disponíveis** 📜📚
3. 🚀 **Main**: A classe principal do sistema. Aqui, você vai instanciar objetos das classes `Livro` e `Biblioteca` para testar as funcionalidades e garantir que tudo está funcionando direitinho!

Entidades – Classe Livro



- Representa um **objeto do mundo real** ou um **conceito relevante** para o domínio do sistema
- São os elementos centrais em torno dos quais as regras de negócio giram
- Características:
  - **Identidade própria:** (ex.: título)
  - **Estado:** (ex.: atributos como titulo, autor, anoPublicacao, e nPaginas fazem parte do estado de cada objeto Livro)
  - **Persistência:** As entidades geralmente são persistidas (armazenadas) em um banco de dados ou outra forma de armazenamento.

Arquitetura de Camadas – Classe Main

- View (Apresentação ou Interface do Usuário)
  - Responsável pela interação com o usuário.
  - Obrigações:
    - Exibir informações
    - Receber entrada de dados
    - Coordenar navegação
    - Tratamento de exceções na interface

Arquitetura de Camadas – Classe Biblioteca

- Service (Negócio)
  - Responsável pela lógica central do sistema.
  - Obrigações:
    - Gerenciar as regras de negócio
    - Orquestrar operações
    - Aplicar consistência e integridade dos dados
    - Tratar exceções de negócio

Melhorias View – Tratar entradas numéricas

- Aqui vamos tratar a exceção caso o usuário digite caracteres não numéricos onde a entrada dever ser numérica

```
private static int inputNumerico(String mensagem) {
    int valor = 0;
    boolean entradaValida = false;
    // Loop até o usuário fornecer uma entrada válida
    do {
        System.out.print(mensagem);
        String entrada = input.nextLine(); // lê como string
        try {
            valor = Integer.parseInt(entrada); // tenta converter para int
            entradaValida = true; // se for válido, sai do loop
        } catch (NumberFormatException e) {
            System.out.println("Erro: Por favor, insira um valor numérico válido.");
        }
    } while (!entradaValida);
    return valor;
}
```

Melhorias View – Ordenação por título

- Vamos melhorar a listagem de livros colocando em ordem alfabética por título:

```
private static void listar() {
    // List<Livro> livros = biblio.pesquisarTodos();
    var livros = biblio.pesquisarTodos();
    // Podemos ordenar a lista através do Comparator.comparing que retorna
    // uma função para comparação
    livros.sort(Comparator.comparing(Livro::getTitulo));

    System.out.println("======== LISTA DE LIVROS =========");
    for (Livro livro : livros) {
        System.out.println("Título: " + livro.getTitulo());
        System.out.println("Autor: " + livro.getAutor());
        System.out.println("Ano: " + livro.getAnoPublicacao());
        System.out.println("N. Páginas: " + livro.getnPaginas());
        System.out.println("==================================");
    }
}
```

Melhorias Service – Pesquisa avançada

- Ajustar Pesquisa para trazer lista de livros conforme parte do título pesquisado:

```
public List<Livro> pesquisarPorTitulo(String titulo) {
    List<Livro> livrosEncontrados = new ArrayList<>();
    for (Livro livro : acervo) {
        // Verifica se o título do livro contém a string recebida (ignora case sensitivity
        if (livro.getTitulo().toLowerCase().contains(titulo.toLowerCase())) {
            livrosEncontrados.add(livro);
        }
    }
    return livrosEncontrados;
}
```

Melhorias Service – Ajuste no método “removerPorTitulo()”

- Após a melhoria anterior, precisamos ajustar o método “removerPorTitulo”

```
public void removerPorTitulo(String titulo) {
    for (Livro livro : acervo) {
        if (livro.getTitulo().equalsIgnoreCase(titulo)) {
            acervo.remove(livro);
            break;
        }
    }
}
```

Melhorias Service – Regra de negócio

- Realizar regras de negócio
  - Não permitir cadastrar livro sem título
  - Não permitir cadastrar dois livros com o mesmo nome

```
// Precisamos declarar a exceção "Exception" através de "trhows Exception"
//   pq ela é uma Checked Exceptions, desta forma é precisso assinar o método
//   para que aqueles que forem consumi-lo realizem os determinados tratamentos
public void adicionar(Livro livro) throws Exception {
    if (livro.getTitulo() == null || livro.getTitulo().isEmpty())
        // Na linha abaixo é lançado uma exceção do tipo "Exception" (o mais genérico)
        throw new Exception("Não é permitido cadastrar um livro sem título");
    for (Livro livroFor : acervo) {
        // Na linha abaixo é lançado uma exceção do tipo "Exception" (o mais genérico)
        if (livroFor.getTitulo().equalsIgnoreCase(livro.getTitulo()))
            throw new Exception("Já existe um livro cadastrado com este título");
    }
    acervo.add(livro);
}
```

Melhorias View – Tratamento de exceções das regras de negócio

- Agora precisamos ajustar nossa camada View para tratar os possíveis lançamentos de exceções da camada Service

```
try {
    // Invoca o método "adicionar" na tentativa de criar o novo livro
    biblio.adicionar(novoLivro);
    // Caso não ocorra nenhuma exceção, então o sistema segue mostrando
    //   mensagem abaixo
    System.out.println("Livro adicionado com Sucesso!");
} catch (Exception e) {
    // Caso ocorra qualquer exceção, o usuário recebe a mensagem
    //   abaixo contendo a mensagem da exceção lançada.
    System.out.println("Erro ao adicionar o livro: " + e.getMessage());
}
```

Agora, é com vocês ...

- **<u>Regras de Negócios – Camada Service (Biblioteca)</u>**

- Nenhum livro deverá ser cadastrado com o campo “autor” nulo ou em branco.
- O ano de publicação permitido será somente entre 1400 e o ano atual (*LocalDate.now().getYear()*)
- Não deve ser permitido cadastrar livros com o número de páginas negativo ou zerado

Agora, é com vocês ...

- **<u>Apresentação – Camada View (Main)</u>**

- Limpe a tela sempre que o menu principal ser carregado, e sempre quando o usuário selecionar uma das opções

```java
public static void clear() {
    // Limpa o console usando sequências de escape ANSI
    System.out.print("\033[H\033[2J");
    System.out.flush(); // Garante que o buffer de saída seja limpo
}
```

Agora, é com vocês ...

- **<u>Apresentação – Camada View (Main)</u>**

- Limpe a tela sempre que o menu principal ser carregado, e sempre quando o usuário selecionar uma das opções

```
public static void clear() {
    // Limpa o console usando sequências de escape ANSI
    System.out.print("\033[H\033[2J");
    System.out.flush(); // Garante que o buffer de saída seja limpo
}
```

Agora, é com vocês ...

- **<u>Apresentação – Camada View (Main)</u>**

- Implemente o métodos para Pesquisar livros por título e para Remover livro por título.
- Lembre-se de sempre tratar os possíveis erros de digitação por parte do usuário, por exemplo, digitar letras em entradas numéricas.