GEIA 25.1- INTRODUÇÃO A CIÊNCIA DE DADOS

Aluno:

SEU NOME COMPLETO (MATRÍCULA INSTITUCIONAL), ex: Fulano da Silva (1234567)

Monitor:

Arthur Veras

Fortaleza

2025

Desafio – Sistema de Biblioteca com Programação Orientada a Objetos e Estruturas de Dados

Descrição do Desafio:

Você será responsável por criar um sistema de gerenciamento de uma biblioteca utilizando **Programação Orientada a Objetos (POO)** e as **estruturas de dados** que você aprendeu, como listas encadeadas e filas. O sistema deverá ser capaz de realizar as seguintes operações:

1. **Cadastro de Livros:**
   Cada livro possui um título, autor, ano de publicação e um status que indica se está **disponível** ou **emprestado**.
2. **Empréstimo e Devolução de Livros:**
   O sistema deve ser capaz de emprestar um livro a um usuário ou registrar a devolução de um livro. Antes de realizar o empréstimo, é necessário verificar se o livro está **disponível**. Quando o livro é devolvido, o status deve ser alterado para **disponível** novamente.
3. **Gerenciamento de Usuários:**
   Cada usuário terá informações como nome, e-mail e uma lista de livros emprestados. Será necessário implementar funcionalidades para adicionar e remover livros dessa lista de acordo com os empréstimos e devoluções.
4. **Busca de Livros:**
   O sistema deve permitir que o usuário busque livros por **título** ou **autor**. A busca deve ser eficiente e retornar uma lista com todos os livros que correspondem ao critério de busca.
5. **Relatório de Livros Emprestados:**
   O sistema deve gerar um relatório com todos os livros que estão **emprestados**, juntamente com o nome do **usuário** que os pegou. Esse relatório pode ser exibido em formato de lista ou tabela.

Requisitos Técnicos:

1. **Utilização de Classes:**
    - **Livros:** Cada livro será representado por uma classe Livro que possui os atributos **título**, **autor**, **ano de publicação** e **status** (disponível ou emprestado).
    - **Usuários:** Cada usuário será representado por uma classe Usuario com atributos como **nome**, **email** e **lista de livros emprestados**.
    - **Biblioteca:** A classe Biblioteca gerenciará a coleção de livros e os usuários, fornecendo métodos para adicionar, buscar e emprestar livros.
2. **Estruturas de Dados:**
   Utilize **listas encadeadas** ou **filas** para organizar os livros e usuários. Essas estruturas são ideais para gerenciar a coleção de livros e os registros dos usuários de forma eficiente.S
3. **Polimorfismo e Herança:**
    - **Polimorfismo:** Aplique polimorfismo criando **diferentes tipos de livros**, como "livros físicos" e "livros digitais". Cada tipo de livro pode ter comportamentos específicos, mas todos devem compartilhar a interface comum de **livro**.
    - **Herança:** Crie diferentes tipos de **usuários** (como "usuário regular" e "usuário VIP"). O usuário VIP pode ter privilégios adicionais, como a capacidade de emprestar mais livros ou ter prazos de devolução mais longos. Ambos os tipos de usuários devem herdar de uma classe base Usuario.
4. **Encapsulamento:**
    - Aplique o **encapsulamento** nos atributos das classes Livro e Usuario. Os dados internos devem ser protegidos e acessados através de **métodos públicos**. Por exemplo, o status de um livro pode ser alterado apenas por métodos que realizem verificações de disponibilidade.
5. **Abstração:**
   Use a abstração para criar interfaces para as operações de **empréstimo** e **devolução**. Defina um método abstrato para as operações de empréstimo e devolução nas classes relevantes, garantindo que cada tipo de livro ou usuário implemente esses métodos de forma específica.

PS: Quaisquer interação(input ou output) deve ser feita **via terminal.**

Critérios de Avaliação:

O trabalho será avaliado com base nos seguintes critérios:

1. **Cumprimento dos Requisitos (30 pontos):**
   O sistema implementa corretamente todas as funcionalidades solicitadas, incluindo o cadastro de livros, empréstimos e devoluções, gerenciamento de usuários, busca de livros e relatório de livros emprestados.
2. **Organização e Clareza do Código (25 pontos):**
   O código deve ser bem estruturado, com **nomes significativos** para variáveis, métodos e classes. As **funções e métodos** devem ser curtos, claros e cumprir uma única responsabilidade.
   A **documentação do código** (comentários) deve ser suficiente para entender o funcionamento das principais partes do sistema.
3. **Aplicação Correta de Conceitos de POO (25 pontos):**
   - **Abstração**: A utilização de classes abstratas e a separação de responsabilidades entre classes devem ser bem aplicadas.
   - **Encapsulamento**: Os dados devem ser protegidos e acessados de maneira controlada.
   - **Herança**: A utilização de herança para criar diferentes tipos de usuários e livros deve ser eficaz.
   - **Polimorfismo**: O uso de polimorfismo para diferentes tipos de livros e usuários deve ser implementado de forma clara e eficiente.
4. **Uso Eficiente de Estruturas de Dados (20 pontos):**
   O uso de **qualquer estrutura de dado** deve ser bem justificado pelo aluno, e essas estruturas devem ser aplicadas de maneira eficiente para organizar os livros e usuários.

Importante:

- **Originalidade:** O uso de **códigos prontos** (de fontes externas), **compartilhamentos de códigos entre os alunos** ou a **aplicação de inteligência artificial**, como resposta automática, **resultará em nota zero**. O trabalho deve ser feito de forma independente, com o código escrito pelo aluno.
- **Não é permitido a importação de quaisquer bibliotecas, todo o código deve ser escrito pelo aluno.**
- **Entrega:** A entrega deve ser feita via **GitHub** do grupo de estudo.

Conclusão:

Este desafio visa proporcionar uma compreensão completa do uso de **Programação Orientada a Objetos** e **Estruturas de Dados** aplicadas a um sistema realista, além de testar suas habilidades em organização e desenvolvimento de código. A clareza, a implementação dos conceitos aprendidos e a eficiência serão os principais pontos de avaliação. Bom trabalho a todos!