Avaliação de OOP - Desafio de Programação

Chegou a hora de avaliarmos os seus conhecimentos em OOP na prática. Para isso, preparamos um desafio de programação que engloba todos os conceitos vistos durante esse módulo.

**ORIENTAÇÕES PARA REALIZAÇÃO DA AVALIAÇÃO**

* Leia atentamente o problema antes de começar a programar
* Utilize boas práticas de programação na realização do desafio;
* Verifique se todos os requisitos do problema foram contemplados;
* A entrega do projeto deve ser feita através da atividade do classroom no formato especificado pelo instrutor.

Boa sorte! 🍀

# Desafio - Sistema para Bibliotecas

Você foi contratado por uma empresa de desenvolvimento de software para criar uma aplicação para gerenciar uma biblioteca. A aplicação deve permitir a realização das seguintes tarefas:

* Adicionar um novo livro à biblioteca;
* Excluir um livro da biblioteca;
* Listar todos os livros da biblioteca;
* Realizar o empréstimo de um livro para um usuário;
* Registrar a devolução de um livro.

Para a realização deste desafio, você deverá utilizar os seguintes conceitos:

1. **SOLID:** Utilize os princípios SOLID para estruturar seu código de forma coesa e flexível, garantindo que ele possa ser facilmente mantido e evoluído.
2. **ENUM:** Utilize enumerações para representar estados e tipos de dados que sejam fixos e bem definidos, como por exemplo o status de empréstimo de um livro (emprestado, disponível, atrasado).
3. **INTERFACES:** Utilize interfaces para definir contratos que devem ser implementados pelas classes que interagem com a aplicação. Por exemplo, crie uma interface "LivroRepository" que define os métodos necessários para adicionar, excluir e listar livros.
4. **ABSTRAÇÃO:** Utilize a abstração para definir classes e métodos genéricos que possam ser reutilizados em diferentes partes da aplicação. Por exemplo, crie uma classe abstrata "ItemBiblioteca" que representa um item que pode ser emprestado pela biblioteca, e que contenha métodos como "emprestar" e "devolver".
5. **ENCAPSULAMENTO:** Utilize o encapsulamento para garantir que o acesso aos dados da aplicação seja restrito e controlado. Por exemplo, crie classes "Livro" e "Usuario" com atributos privados e métodos públicos para acessá-los.
6. **COMPOSIÇÃO:** Utilize a composição para criar objetos complexos a partir de outros objetos mais simples. Por exemplo, crie uma classe "Emprestimo" que contenha objetos "Livro" e "Usuario".