MC322 – Programação Orientada a Objetos
Projeto Final – 2s2023

Prof. Dr. Bruno Cafeo (Professor) Fillipe dos Santos Silva Jesamin Zevallos Quispe Vinicius Leme Soares

Discord: `https://discord.gg/mHcUFRDGgQ`

1 Plataforma Digital Integrada para Turismo

O sistema proposto é uma plataforma digital focada no setor de turismo e viagens. Esta plataforma permite que os usuários naveguem por diversos pacotes de viagem, visualizem detalhes sobre destinos específicos, e efetuem reservas de acordo com suas preferências e necessidades. O software é projetado para organizar pacotes de viagem em várias categorias, como "Aventura", "Cultura"e "Relaxamento", oferecendo uma variedade de opções para atender a diferentes interesses e estilos de viagem.

Além disso, o sistema é altamente modular e segue os princípios da Programação Orientada a Objetos (POO), usando técnicas de modularização, herança, polimorfismo, e padrões de projeto. Isso garante que a plataforma seja flexível, extensível e mantenível. A plataforma também enfatiza a segurança e integridade dos dados, implementando métodos de persistência robustos, seja por meio de arquivos ou de um sistema de banco de dados.

Os administradores têm a capacidade de gerenciar o conteúdo do sistema, adicionando, editando ou removendo pacotes de viagem conforme necessário. Em contraste, os usuários finais, que podem ser turistas ou viajantes, podem navegar pelo sistema, visualizar informações detalhadas sobre pacotes e destinos, e fazer reservas.

O sistema também é projetado para lidar com situações excepcionais, como tentativas de reserva em datas indisponíveis, garantindo que os usuários recebam feedback adequado e evitando frustrações. Além disso, com a aplicação do padrão Observer, os usuários podem ser notificados sobre ofertas especiais, promoções e novidades, incentivando a interação e o engajamento contínuo com a plataforma.

Conceitos Básicos de POO e Modularização

- Crie classes para representar entidades do mundo das viagens, como `Destino`, `PacoteViagem` e `Usuário`.
- Utilize construtores para inicializar objetos com informações relevantes.
- Organize suas classes em pacotes lógicos, como `model`, `controller` e `view`.

Visibilidade e Relacionamentos

- Aplique os diferentes níveis de visibilidade (public, private, protected) para controlar o acesso a atributos e métodos nas classes.
- Estabeleça relacionamentos entre classes, como a associação entre `Usuário` e `PacoteViagem`.

Herança e Polimorfismo

- Implemente herança criando uma classe base como `Pacote` e estenda-a para diferentes tipos de pacotes de viagem.
- Utilize polimorfismo para tratar diferentes tipos de pacotes de viagem de maneira uniforme.

Interfaces e Classes Abstratas

- Crie interfaces como `Reservável` para permitir que pacotes de viagem sejam reservados.
- Utilize classes abstratas para compartilhar comportamentos entre diferentes tipos de pacotes.

Enum, Coleções e Classes Genéricas

- Utilize um enum para representar categorias de destinos, como `Praia`, `Montanha`, `Cidade`, etc.
- Use coleções, como Listas, para armazenar e gerenciar pacotes de viagem.
- Explore classes genéricas para criar estruturas de dados flexíveis, como uma Lista Genérica de Reservas.

Tratamento de Exceções e Metaclasses

- Implemente tratamento de exceções para lidar com situações como reservas inválidas ou destinos não disponíveis.
- Pesquise e apresente conceitos sobre metaclasses e reflexão em Java, mostrando como eles podem ser aplicados no contexto do sistema de reservas.

Persistência de Objetos e Padrões de Projeto

- Explore maneiras de persistir informações de destinos, pacotes e usuários, considerando o uso de arquivos ou um banco de dados simples.
- Implemente a funcionalidade de persistência para salvar informações sobre destinos, pacotes e reservas.
- Aplique padrões de projeto, como o padrão Repository, para separar a lógica de acesso aos dados.
- Aprofunde-se no uso de padrões de projeto como o padrão de Projeto de Mapeamento de Objeto-Relacional (ORM).

Projeto Orientado a Objetos (SOLID) e Padrões de Projeto

- Demonstre como os princípios SOLID podem ser aplicados em seu projeto, especialmente o Princípio da Responsabilidade Única e o Princípio Aberto/Fechado.
- Considere a utilização de padrões de projeto como o padrão Observer para notificar os usuários sobre ofertas e promoções.

Requisitos e Regras de Negócio

- Os usuários poderão visualizar informações detalhadas sobre destinos, incluindo descrições, imagens e atividades disponíveis.
- Os pacotes de viagem serão organizados em categorias, como "Aventura", "Cultura"e "Relaxamento", representadas por um enum.

- Cada pacote de viagem conterá informações como destino, preço, datas disponíveis e atividades inclusas.
- Os usuários poderão buscar pacotes por destino, categoria e preço máximo.
- A reserva de um pacote de viagem deverá garantir a disponibilidade das datas selecionadas.
- Será necessário lidar com exceções, como tentativas de reserva em datas esgotadas ou seleção de um destino indisponível.
- O sistema permitirá que os administradores adicionem, editem ou removam pacotes de viagem.
- Ao realizar uma reserva, o usuário receberá uma confirmação via e-mail.
- Os princípios SOLID serão aplicados para garantir que cada classe tenha uma única responsabilidade e que o sistema possa ser estendido sem modificações.
- Um padrão de projeto Observer será implementado para notificar os usuários sobre ofertas e promoções especiais.
- A persistência de dados será assegurada, permitindo que as informações sobre destinos, pacotes e reservas sejam armazenadas de forma segura.

2 Avaliação

- Entrega realizada dentro do prazo estipulado.
- Execução do código.
- Qualidade do código desenvolvido (saída dos dados na tela, tabulação, comentários).
- Desenvolvimento correto de tudo solicitado.
- **Entregar no diretório files do repositório o diagrama de classes UML**

3 Entrega

- Você não precisa criar outro repositório ou outro projeto, a ideia é avançar sobre o repositório que você já tem.
- A entrega do Laboratório é realizada exclusivamente via **Github**. Para a submissão no Github, gere um release (tag) com a identificação do laboratório no formato <projeto_final-RA>. Por exemplo, para o aluno com RA 123456, a tag será: projeto_final-123456.
- Evite criar *releases* enquanto não tiver certeza que seu código está funcionando como esperado.
- Utilize os horários de atendimentos para tirar eventuais dúvidas de submissão e também relacionadas ao desenvolvimento do projeto.
- Data da entrega: **05/11/2023** até às **23h59**.