INSTITUTO FEDERAL
SÃO PAULO
Câmpus Araraquara

Introdução à Orientação a Objetos – ARQINOO
Prof. Fernando Vieira Duarte - E-mail: fernandoduarte@ifsp.edu.br

PROJETO FINAL

Data de Entrega: até 19/12/2023

Valor: 4,0 pontos.

Observações:

- O projeto final pode ser desenvolvido em duplas ou individualmente.
- Para o desenvolvimento do projeto, use conceitos de orientação a objetos e boas práticas de programação estudadas em aula.
- O tema do projeto envolve algum tópico não estudado em aula, o que exigirá a realização de uma pesquisa, que é requisito essencial para que determinadas habilidades em programação sejam aprimoradas.
- Considere que uma determinada operação somente será realizada se todos os dados necessários forem informados.
- Efetue todas as validações e atualizações necessárias, de acordo com os requisitos apresentados.
- Projetos cujos códigos sejam idênticos entre si serão anulados.

O objetivo do Projeto Prático é aplicar conceitos e princípios de orientação a objetos que foram estudados durante a disciplina. O resultado do projeto deve ser uma aplicação que satisfaça tanto os requisitos funcionais para o tema apresentado quanto atributos de qualidade tais como legibilidade de código, manutenibilidade e reusabilidade.

O Projeto Prático corresponde a 40% da nota final da disciplina (vale 4,0 pontos). A nota final do Projeto Prático levará em consideração o resultado dos testes realizados pelo professor para verificar se o comportamento da aplicação está conforme o que foi especificado, e também a análise de código após a entrega dos trabalhos. A seguir, são apresentados os requisitos do tema.

Tema: Sistema de Locação de Carros

O sistema de locação de carros deve gerenciar basicamente locações e devoluções de carros de uma locadora de carros. Portanto, considere os seguintes requisitos específicos:

1. A aplicação deve permitir inserir em uma estrutura de dados todos os clientes da locadora de carros. Cada cliente possui código (número sequencial gerado automaticamente pela aplicação), nome, lista de locações (locações serão detalhados a seguir) e pode ser uma pessoa física ou pessoa jurídica. Pessoas físicas possuem

especificamente o CPF, e pessoas jurídicas possuem especificamente CNPJ e razão social. A estrutura de dados utilizada não deve permitir a inserção de dois ou mais clientes iguais, isto é, clientes com o mesmo CPF ou CNPJ.

2. A aplicação deve permitir inserir em uma estrutura de dados todos os carros pertencentes a locadora de carros. Cada carro possui código (número sequencial gerado automaticamente pela aplicação), marca, modelo, ano, placa, quantidade de portas, ar-condicionado (indicar se possui ou não), valor da diária de locação e estado (disponível ou locado – inicialmente, todo carro cadastrado estará disponível).

3. A aplicação deve permitir realizar locações e mantê-las em uma estrutura de dados. Toda locação possui um número (número sequencial gerado automaticamente pela aplicação), data de realização (data atual capturada do sistema operacional), número de diárias, data máxima prevista para devolução (calculada pela aplicação de acordo com o número diárias informado pelo usuário), data de devolução (inicialmente com valor nulo), cliente e o carro escolhido. Após a realização da locação, todas as atualizações necessárias devem ser realizadas por uma classe específica de gerenciamento.

4. A aplicação deve permitir realizar devoluções. Realizar uma devolução significa calcular e exibir para cliente o valor total da locação (com base no valor da diária e número de diárias efetivadas), e atualizar o estado da aplicação de tal forma que se possa determinar qual carro foi devolvido e a locação finalizada. A responsabilidade pela realização de uma devolução deve ser atribuída a uma classe específica de gerenciamento.

5. Crie uma classe que permita gerenciar uma coleção de clientes e uma classe que permita gerenciar uma coleção de carros. Ambas as classes devem ser codificadas da maneira mais flexível possível. Além disso, observe que, para realizar locações e devoluções, operações de consulta de ambas as classes deverão ser solicitadas (outras classes que serão seus clientes). Pense em uma estratégia que garanta que exista uma única coleção de usuários e uma única coleção de carros e que elas sejam acessíveis em qualquer parte do código. Dica: Use um padrão de projeto Singleton.

6. Para cada funcionalidade (inserção de clientes, inserção de carros, realização de locações e devoluções), crie uma classe principal e que permita a interação com o usuário (funcionário da locadora). As funcionalidades devem acessadas a partir de um menu.

7. Utilize pacotes para agrupar todas as classes criadas de acordo com as suas responsabilidades. Para gerar e manipular datas, utilize a classe LocalDate da API Java (java.time).