ESPECIFICAÇÃO DE TRABALHO PRÁTICO

Análise e Desenvolvimento de Sistemas
INF008 – Programação Orientada a Objetos
Professor: Sandro Santos Andrade

1. Objetivo Geral

Desenvolver um sistema em Java para gerenciamento de uma locadora de veículos, utilizando práticas avançadas de programação orientada a objetos, como interfaces, polimorfismo e ligação dinâmica (*dynamic binding*). O sistema deverá possuir uma interface gráfica construída em JavaFX e deverá ser extensível por meio de plug-ins. O sistema deverá funcionar com o banco de dados MariaDB fornecido pelo professor.

2. Requisitos Funcionais

- Aluguel de veículo:
  - O sistema deve conter uma UI onde o usuário poderá realizar uma nova locação de veículo
  - Fluxo de locação:
    - O sistema deverá apresentar um combobox com os emails de todos os clientes, para que o usuário informe qual cliente está realizando a locação.
    - O banco de dados contém veículos de 6 tipos diferentes: *econômico*, *compact*, *SUV*, *luxo*, *VAN* e *elétrico*. Estas informações devem ser apresentadas em um combobox, para que o usuário escolha o tipo de carro que deseja locar.
    - Com base no valor atual do combobox de tipo de veículo, o sistema apresentará, em uma tabela, todos os veículos do tipo atualmente selecionado no combobox que estejam com *status* = "AVAILABLE". A tabela deverá apresentar as seguintes informações: marca (*make*) e modelo (*model*); ano (*year*); tipo de combustível (*fuel_type*), tipo de câmbio (*transmission*) e quilometragem (*mileage*).
    - O usuário deve selecionar apenas um item desta tabela para indicar o veículo que deseja locar.

        - O usuário deverá informar o dia de início (*start_date*) e de fim (*scheduled_end_date*) da locação.
        - O usuário deverá informar o local de retirada do veículo (*pickup_location*).
        - O usuário deverá informar o valor da diária (*base_rate*) e o valor do seguro (*insurance_fee*).
        - O sistema deverá apresentar o valor total da locação, solicitar a confirmação do usuário, inserir um novo registro na tabela *rentals* e alterar o *status* do veículo locado para "RENTED".
        - NOTA: o cálculo do valor total da locação deve ser implementado através de um método polimórfico, com cálculo conforme descrito a seguir:
            - Valor total = (valor diária * número de dias de locação) + qualquer número de taxas adicionais descritas no campo *additional_fees* da tabela *vehicle_types*. Utilize apenas valores com o sufixo "*_fee*". Ex: *extra_passenger_fee*, *concierge_fee*, etc.
        - Outras informações importantes:
            - Considerar que o tipo de locação (*rental_type*) será sempre "DAILY".
- Relatório 1
    - Relatório que apresenta um gráfico de pizza com informações sobre distribuição de veículos por tipo de combustível.
    - Utilize o JavaFX charts (https://docs.oracle.com/javafx/2/charts/jfxpub-charts.htm) para implementação.
    - A query com os dados a serem apresentados está disponível no arquivo *report1.sql*.
- Relatório 2
    - Relatório que apresenta uma tabela com dados gerais sobre as locações.
    - Utilize uma tabela JavaFX para apresentar os dados do relatório.
    - A query com os dados a serem apresentados está disponível no arquivo *report2.sql*.

3. Requisitos Não-Funcionais

- O sistema deve ser implementado em Java 25.0.1.
- A interface gráfica deve ser desenvolvida em JavaFX.
- O acesso ao banco de dados deve ser feito via JDBC ou algum framework de ORM, utilizando a base de dados MariaDB gerada pelo ambiente Docker em anexo.
- A arquitetura do sistema deve ser modular, favorecendo a reutilização de código e a extensibilidade via plug-ins.

- A implementação **DEVE** ser realizada com base no esqueleto de microkernel disponibilizada como parte desta especificação. A equipe está livre para realizar **quaisquer** mudanças neste microkernel, tais como extensões na API dos controllers ou criação de novos controllers.
- Para usuários do Windows, sugere-se o uso do WSL (https://learn.microsoft.com/pt-br/windows/wsl/install) para execução do container Docker do MariaDB.
- O projeto deve utilizar o Maven como *build tool*.
- Os seguintes plug-ins devem ser desenvolvidos:
  - 1 plug-in para cada tipo de veículo: *econômico*, *compact*, *SUV*, *luxo*, *VAN* e *elétrico*.
  - Plug-in que implementa o relatório 1.
  - Plug-in que implementa o relatório 2.
- Deve ser possível adicionar ou remover funcionalidades (plug-ins) sem recompilar o sistema principal.

Boas Práticas:

- Use nomes siginificativos para variáveis, métodos e classes.
- Divida a lógica em métodos pequenos e reutilizáveis.
- Siga as convenções do programação do próprio Java.
- Lembre-se que uma solução simples é sempre melhor.

4. Critérios de Avaliação

Critério	Pontos
Implementação correta dos requisitos	3,0
Uso adequado de interfaces e polimorfismo	2,0
Arquitetura modular com plug-ins	2,5
Qualidade da interface gráfica (UX/UI)	1,5
Clareza, organização e documentação do código	1,0

5. Entregáveis

- Código-fonte do sistema + arquivo README.md compactados em formato *.zip* ou *.tar.gz*.
  - **OBS:** o pacote *.zip* ou *.tar.gz* não deve conter arquivos *.class*, apenas arquivos *.java*. Limpe o seu projeto antes de criar o pacote.
- Arquivo README.md deve conter:
  - Instruções de compilação e execução.

        - **OBS:** a compilação e execução do projeto não deve requerer a instalação de nenhuma IDE.
    - Link para vídeo no YouTube ou plataforma similar onde os dois integrantes da equipe explicam as partes mais importantes do código desenvolvido. O vídeo deve ter duração máxima de 5 minutos. As informações presentes no vídeo irão influenciar todos os critérios de avaliação.

6. Prazo e Forma de Entrega

- O trabalho deve ser enviado até o dia **03/02/2026** às 23:59:59 (não extensível).
- O código-fonte e o arquivo README.md deve ser enviado por email para sandroandrade@ifba.edu.br, com o assunto "INF008 <nome-completo-da-dupla>", sem as aspas e sem os caracteres < e >. Ex: INF008 Eduardo da Silva Santos e Maria Joaquina.
- Trabalhos enviados com outro assunto ou para outro email não serão corrigidos.

7. Informações Importantes

- Todos os códigos-fonte entregues serão checados por plágio utilizando as ferramentas **MOSS** (Measure of Software Similarity - https://theory.stanford.edu/~aiken/moss/) e **JPlag** (https://github.com/jplag/JPlag). Desenvolva sua própria solução, é sua chance de aprender.
- Recursos de Inteligência Artificial (Chat-GPT, DeepSeek e similares) devem ser utilizados com moderação. Códigos-fonte centrais não explicados no vídeo serão desconsiderados. Equipes com códigos similares e explicações similares no vídeo terão suas notas zeradas.
- Todas as dúvidas sobre o trabalho deverão ser abertamente discutidas no grupo da disciplina no Telegram.

Bom trabalho!