Programação Orientada a Objeto

INSTITUTO FEDERAL
Triângulo Mineiro
Campus Paracatu

Trabalho Prático – **Estacionamento**

Parte 01

Professor: Edwar Saliba Júnior Valor: 15 pontos

Contextualização:

Precisa-se de um software para controle de um estacionamento. Este estacionamento é um pouco diferente dos estacionamentos tradicionais em sua forma de trabalhar. Neste estacionamento todos os clientes, seus respectivos veículos e até os pátios existentes para estacionar são controlados. O cliente paga sempre o valor de uma diária, mesmo que seu veículo fique apenas 5 minutos no estacionamento. No entanto, os clientes deste estacionamento têm duas vantagens: a primeira é que ele só pagará pelos dias que utilizar o estacionamento e a segunda é que ele tem sua vaga garantida o mês inteiro. Os pagamentos são realizados pelos clientes no último dia de cada mês. Para desenvolver este software a estrutura mínima, apresentada na Figura 1, deverá ser utilizada. Porém, o governo resolveu burocratizar a vida dos estacionamentos e lançou uma lei determinando que o estacionamento tem que registrar, além dos dados tradicionais coletados dos veículos, também o tipo do Veículo que está utilizando os serviços da empresa. As demais regras de funcionamento do estacionamento, segundo esta nova lei, podem permanecer da mesma forma que já estavam estabelecidas. Com o objetivo de atender esta nova lei e também melhorar o *software* já construído, você deverá implementar algumas melhorias e para isto seu *software* passará a ter as seguintes classes mostradas a seguir:

1) Estrutura de classes[1] (Figura 1):
   - Cliente,
   - Pátio,
   - Conta e
   - Veículo
     1. Moto,
     2. Carro,
     3. Caminhão e
     4. Ônibus.

2) Atributos das Classes:
   - Cliente
     1. código
     2. nome
     3. logradouro
     4. número
     5. bairro
     6. município
     7. estado
     8. cep
     9. telefone
   - Veículo
     1. marca
     2. modelo
     3. ano de fabricação
     4. ano do modelo
     5. chassi
     6. placa
   - Moto
     1. cilindradas
   - Carro
     1. quantidade de portas

1 Neste diagrama não estão representadas as classes de gerenciamento.

- Caminhão
    1. capacidade de carga
- Ônibus
    1. capacidade de passageiros
- Pátio
    1. nome
    2. logradouro
    3. número
    4. bairro
    5. município
    6. estado
    7. cep
    8. telefone
    9. capacidade de veículos
    10. valor da diária
- Conta
    1. Pessoa
    2. Veículo
    3. Pátio
    4. ano
    5. mês
    6. diárias
    7. paga

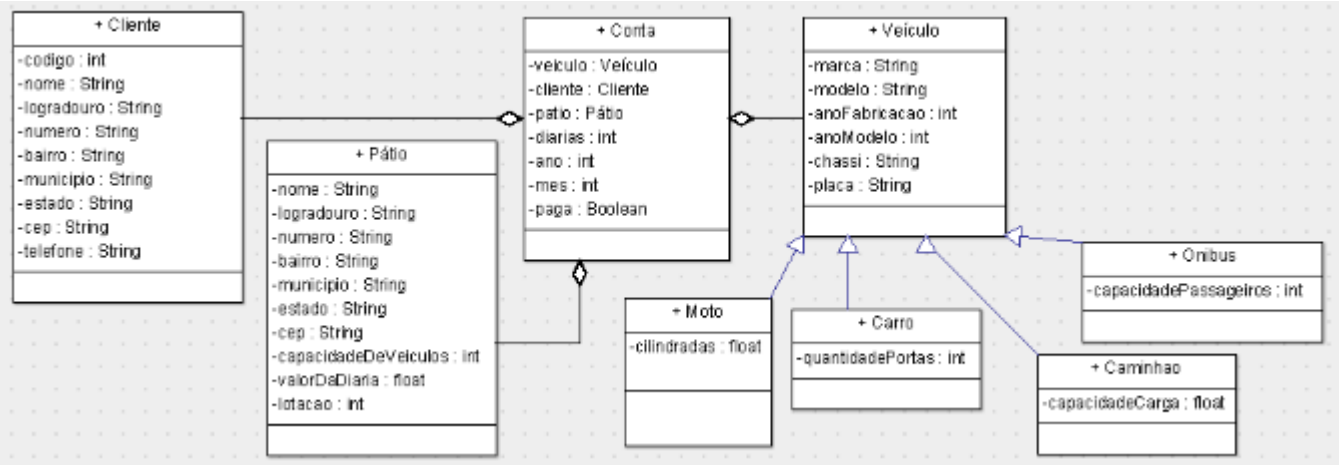

Figura 1: Diagrama de classes.

3) Caso seja necessário, os grupos poderão criar mais atributos nas classes. Os métodos deverão ser criados de acordo com a necessidade de cada *software*, por este motivo não foram especificados no diagrama UML (Figura 1);

4) Na classe Cliente deverá ser criado um atributo “dataCadastro”, que não consta no diagrama acima, do tipo LocalDate. Caso o grupo desconheça este conceito e o mesmo não seja apresentado em sala de aula, o grupo deverá pesquisar a respeito.

5) Neste *software*, o usuário deverá ser capaz de cadastrar clientes, veículos, pátios e contas por pátio. Não deverá haver limite de cadastrados;

6) Crie um *menu* principal com os itens: Cliente, Veículo, Pátio, Conta e Sair. E para cada um destes itens, com exceção do item “Conta”, deverá existir as seguintes opções (*submenu*):
    - Cadastro,
    - Alteração
        - a operação de alteração deverá dar a possibilidade do usuário ver os valores já cadastrados para um determinado item do vetor. Item este, escolhido pelo usuário para que o mesmo possa alterá-lo,
        - atenção! Excluir um objeto e criar um novo para seu lugar não é alteração e será contado como erro no *software*;
    - Exclusão

        - antes de excluir um item do vetor, deverá ser emitida uma mensagem de confirmação da operação para o usuário. Caso o usuário confirme a operação, então o item será apagado. Caso contrário o item não será apagado;
    - Consulta
        - deverá possibilitar ao usuário do *software* a visualização de um item do vetor;
    - Relatório
        - deverá possibilitar ao usuário do *software* a visualização de todos os itens do vetor;
    - Voltar ao *menu* principal
        - Deverá possibilitar ao usuário a volta ao *menu* principal.

7) Para o item "Conta", deverá existir as seguintes opções (*submenu*):
    - Cadastro de Conta
    - Exclusão de Conta
        - uma conta só poderá ser excluída se não houver débitos;
    - Inclusão de Diária
        - deverá dar a possibilidade, ao usuário do *software*, de incrementar o atributo "diárias" de um Cliente/Veículo/Pátio;
    - Exclusão de Diária,
        - deverá dar a possibilidade, ao usuário do *software*, decrementar o atributo "diárias" de um Cliente/Veículo/Pátio;
    - Total a Pagar
        - deverá mostrar quanto um Cliente/Veículo/Pátio está devendo em um determinado mês/ano, conta não paga;
        - nesta tela deverá haver também, uma opção que permita realizar o pagamento das diárias (todas de uma única vez) ou não;
    - Relatório de Contas a Receber
        - deverá dar a opção de apresentar o relatório:
            - baseado no mês / ano escolhido pelo usuário ou
            - contendo todas as contas já recebidas;
    - Relatório de Conas Recebidas
        - deverá dar a opção de apresentar o relatório:
            - baseado no mês / ano escolhido pelo usuário ou
            - contendo todas as contas já recebidas;
    - Voltar ao *menu* principal
        - deverá possibilitar ao usuário a volta ao *menu* principal.

8) Caso seja necessário, os grupos poderão criar mais itens no *submenu* de Conta;

9) **Os grupos não poderão criar mais atributos nas classes sem consulta prévia ao professor.** Os métodos deverão ser criados de acordo com a necessidade de cada *software*, por este motivo não foram especificados no diagrama UML (Figura 1);

10) Devem ser criadas as classes: Moto, Carro, Caminhão e Ônibus. Todas herdeiras de Veículo.

11) O atributo veículo da classe conta e toda e qualquer outra variável/atributo que trate de armazenamento de veículos, deverão ser do tipo Veículo obrigatoriamente. Para que não haja problema na manipulação destas variáveis/atributos, você deverá usar o recurso de *Polimorfismo* para armazenamento e recuperação dos objetos nelas eventualmente guardados.

12) Para armazenamento dos objetos do tipo: Moto, Carro, Caminhão e Ônibus, deverá ser utilizado um único *ArrayList* chamado "veículos". Para tanto, faça uso do recurso conhecido como Polimorfismo.

13) A classe Veículo deverá ser uma classe abstrata. Caso o grupo desconheça este conceito e o mesmo não seja apresentado em sala de aula, o grupo deverá pesquisar a respeito.

14) Faça uso de classes do tipo "fichário" já apresentadas em sala de aula.

Como o seu *software* deverá funcionar

- A estrutura de *menus*, deverá proporcionar ao usuário:
  - a possibilidade de navegar entre o *menu* principal e seus *submenus* sem efetuar qualquer operação;
  - o usuário só poderá sair do programa através do *menu* principal, ou seja, acessando a opção “Sair”;
- Um *software* deve ter uma boa aparência e ser de fácil utilização, para agradar e facilitar a vida de quem o utilizará.

Regras para a entrega do trabalho

- Deverá ser apresentado e entregue, o projeto (compactado) do código-fonte. Antes de compactar o projeto, exclua a pasta “dist” se ela existir.
- **O código-fonte que será entregue e apresentado não deverá possuir nenhum tipo de comentário.**
- Deverá ser enviado para o *e-mail*: eddiesaliba2@yahoo.com (de acordo com as regras a seguir).
- **Não serão recebidos trabalhos após a data marcada para entrega.**
- **Para a apresentação no laboratório deverá ser levado pelo grupo, em *pendrive*, uma cópia do arquivo que foi enviado por *e-mail*. Caso o grupo possua alguma restrição ou dificuldade no cumprimento desta regra, então, deverá avisar ao professor com antecedência mínima de 24 horas da data de apresentação.**

Regras para envio do *e-mail* com o trabalho

- No assunto do *e-mail* deve constar apenas o título: **IFTM – POO – Paracatu – ADS3PA – Fase 02**
- No corpo do *e-mail* deverá conter, única e exclusivamente, o nome completo de todos os integrantes do grupo (**um em cada linha**).
- Só será aceito UM *e-mail* por grupo. Portanto, verifique se está tudo certo com seu *e-mail* e trabalho antes de enviá-lo.
- **O *e-mail* deverá ser enviado, no máximo, até UM dia antes da data marcada para apresentação.**

Obs.: O desrespeito a qualquer das regras acima implicará na perda de créditos para o grupo.

Critérios de Avaliação no Laboratório:

- Conformidade do *software* em relação ao solicitado;
- Legibilidade do código (**organização, endentação e etc.**);
- Usabilidade das interfaces de interação com o usuário;
- **Entendimento individual a respeito do código-fonte apresentado.**