Programação Orientada a Objeto

Trabalho Prático – **Estacionamento** – Parte 01

Programação de Computadores II

Professor: Edwar Saliba Júnior Valor: 25 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 que deverá ser utilizada é a seguinte:

1) Estrutura de classes[1] (Figura 1):
   - Cliente,
   - Pátio,
   - Veículo e
   - Conta.

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
   - Pátio
     1. nome
     2. logradouro
     3. número
     4. bairro
     5. município
     6. estado

1 Neste diagrama não estão representadas as classes de gerenciamento.

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

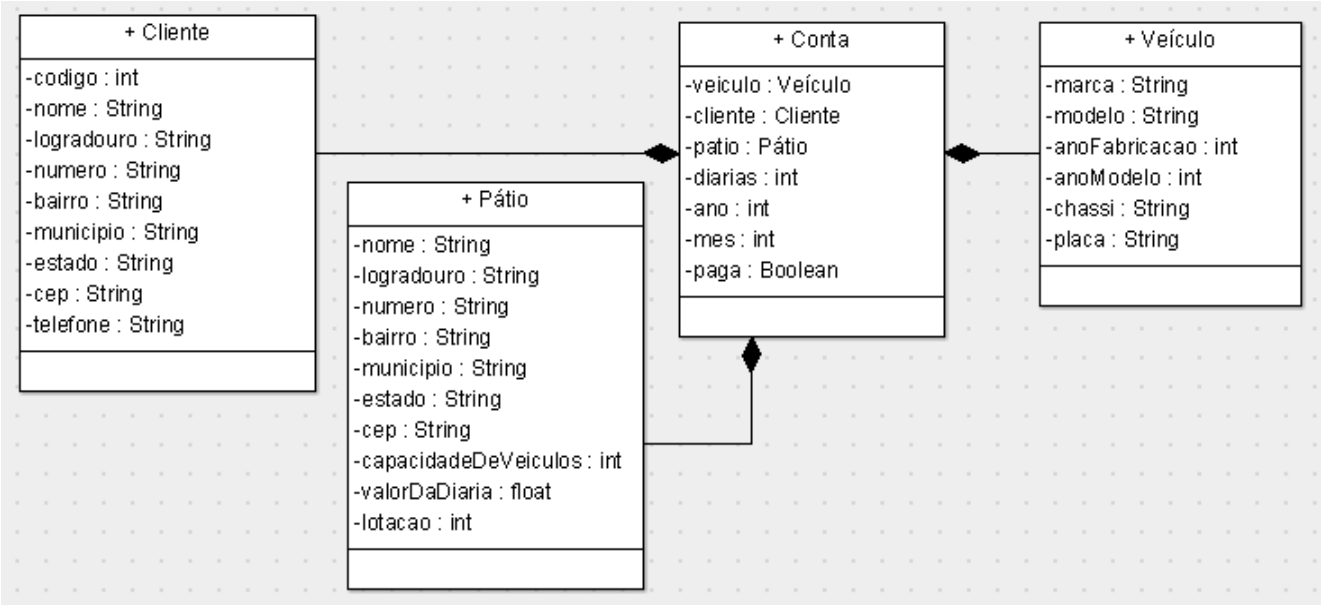

Figura 1: Diagrama UML

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

5) Neste *software*, o usuário deverá ser capaz de cadastrar até 100 clientes, 100 veículos, 5 pátios e até 100 contas por pátio, para tanto, faça uso de vetores.

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-los;
   - 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*):
   - Inclusão de Diária,
     - Deverá incrementar o atributo "diárias" de um Cliente/Veículo/Pátio.
   - Exclusão de Diária,
     - Deverá 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.
   - Voltar ao *menu* principal
     - Deverá possibilitar ao usuário a volta ao *menu* principal.

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.
- **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: **CEFET – PC2 – Campus N – Curso – Turma**
- Onde:
  - no lugar da letra "N", após a palavra "Campus", deverá ser colocado o número do mesmo;
  - no lugar da palavra "Curso" deverá ser colocado o nome do curso em que você está cursando esta disciplina;

    - no lugar da palavra “Turma” deverá ser colocado o nome da turma de laboratório (“G1” ou “G2”).
- No corpo do *e-mail* deverá conter, única e exclusivamente, o nome 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 <u>individual</u> a respeito do código-fonte apresentado.**