PO24CP - Programação Orientada a Objetos
Profa. Luciene de Oliveira Marin
lucienemarin@utfpr.edu.br

Trabalho #02 - Simulador de Corrida de Veículos

Data de entrega: 04/12/2018 (até 23h55), via moodle.
O trabalho poderá ser individual ou em grupo de até 3 alunos.

Um simulador de corrida de veículos irá controlar veículos do tipo bicicleta, motocicleta, carro popular e ferrari através de seu centro de comandos. Os veículos estarão competindo no estilo corrida.

Cada veículo criado possuirá uma identificação única (que deverá ser gerada automaticamente), uma quantidade de rodas e uma quantidade de combustível (para aqueles que possuem motor).

Os veículos motorizados devem ser abastecidos e consomem combustível à medida que se deslocam. Eles apenas se movimentam se há combustível suficiente para tal e se os pneus das rodas estiverem calibrados. Assume-se que para mover um traço, a motocicleta gasta 0,5 litro, o carro polular gasta um litro e a ferrari gasta 1,5 litros de combustível. O veículo não deve se movimentar se ele não possuir a quantidade de combustível suficiente e se seus pneus não estiverem calibrados.

Os veículos se movem sempre na horizontal da esquerda para direita de acordo com suas respectivas quantidades de traços (unidade de movimento):

- bicicleta: de um em um traço,
- motocicleta: de três em três traços,
- carro popular: de cinco em cinco traços,
- ferrari: de dez em dez traços.

Com base no detalhamento anterior, faça:

1. Descreva o diagrama UML das classes do simulador tomando como modelo o esboço apresentado na Figura 1 (gerar o arquivo pdf do diagrama)
2. Com base do diagrama UML elaborado acima, desenvolva um aplicativo Java com um menu interativo que permita ao usuário executar o simulador de corrida de veículos com no máximo 20 veículos:
   (a) Incluir veículo
      – Solicitar o tipo do veículo (B, M, C, F). Gerar um *id* automático para o veículo criado e assumir que os pneus estão vazios.
      – Para os veículos motorizados, considerar que estão sem combustível.
   (b) Remover um veículo (deve-se informar o identificador do veículo)

(c) Abastecer veículo (deve-se informar o identificador do veículo e a quantidade de combustível em litros)

(d) Movimentar veículo (deve-se informar o identificador do veículo)

(e) Movimentar veículos por tipo (deve-se informar o tipo de veículo e movimentar todos os veículos daquele tipo)

(f) Imprimir dados de veículos por tipo (deve-se informar o tipo de veículo e imprimir os dados de todos os veículos daquele tipo)

(g) Imprimir pista de corrida (imprime na ordem em que estão no array os veículos com seus respectivos traços percorridos, como mostra o exemplo abaixo:

- O identificador do veículo na pista de corrida será composto pela sua sigla (**B** para bicicleta, **M** para motocicleta, **C** para carro popular e **F** para ferrari) concatenado com o **id** do veículo

```
|F1
|B2
|C3
|M4
|B5

|----------F1
|-B2
|-----C3
|---M4
|-B5
```

(h) Esvaziar pneus de um veículo

(i) Calibrar pneus de um veículo

(j) Sair da aplicação

Avaliação:

O trabalho será avaliado em função da:

- Correção (o aplicativo cumpre com as exigências);
- Documentação (o aplicativo está devidamente comentado);
- Paradigma orientado a objetos (o aplicativo está seguindo os princípios da programação OO: -encapsulamento, -associação de classes, - herança, - polimorfismo?)
- Modularidade (o aplicativo está bem estruturado onde necessário, com métodos (funções) parametrizados);
- Robustez (o aplicativo não trava em tempo de execução).

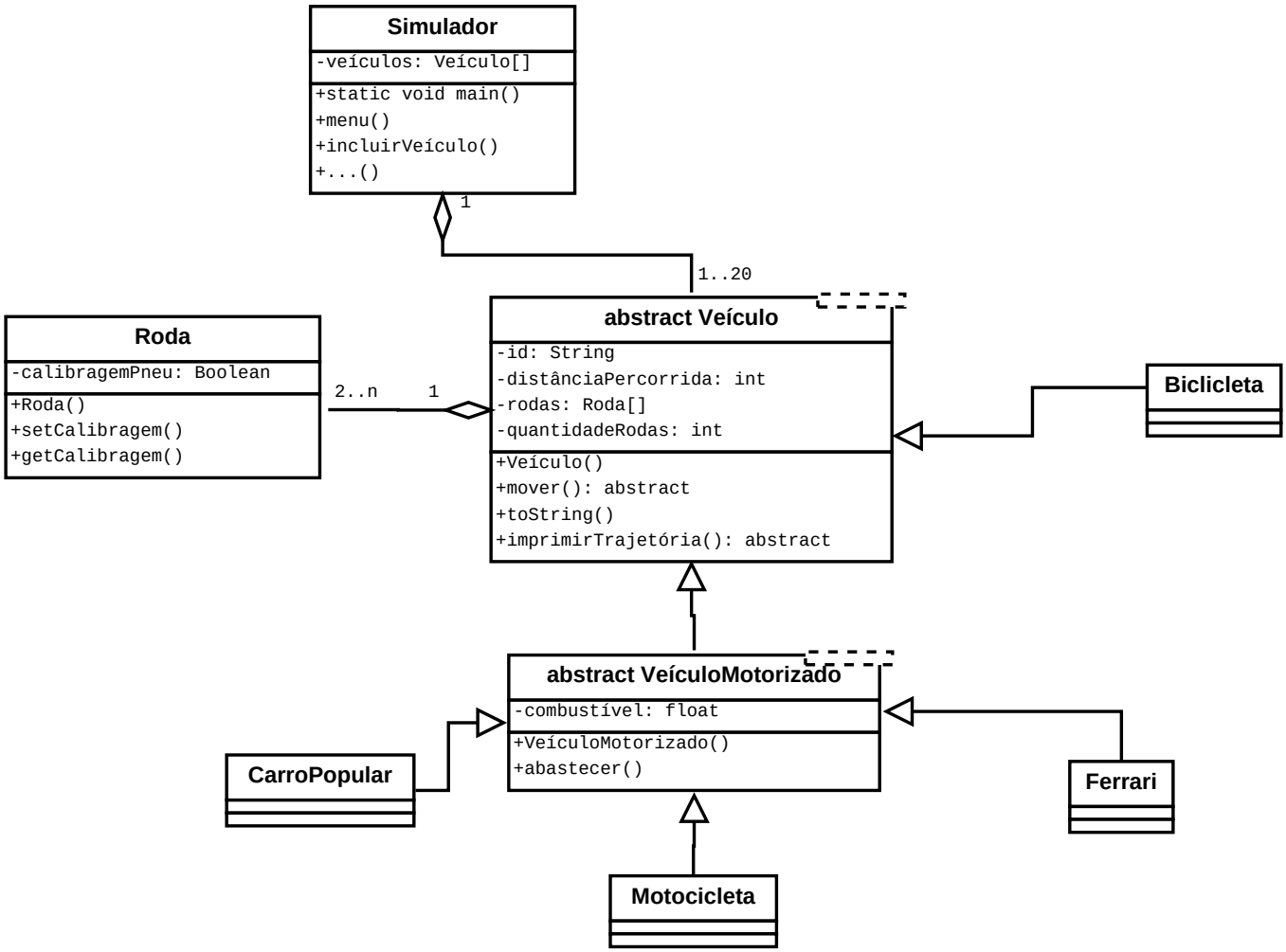

Figura 1: Esboço do diagrama UML a ser seguido.

Detalhamento de itens a serem avaliados e seus respectivos pontos descontados:

Item	Atendeu?
Respeitar o princípio do encapsulamento de dados	
Usar modificadores de acesso adequados (`private` e `public`)	
Criar getters e setters que forem necessários	
Criar métodos construtores parametrizados	
Fazer sobrecarga de pelo menos um método (qualquer um)	
Ter pelo menos um atributo `final`	
Fazer uso da palavra reservada `this`	
Ter pelo menos um atributo `static`	
Criar relacionamento entre classes (Agregação ou Composição)	
Fazer uso de classe abstrata	
Fazer uso do conceito de herança e polimorfismo	
Não utilizar modificador `protected`	
Não apresentar erro em tempo de execução	
Apresentar o diagrama UML	