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

Trabalho #02 - Simulador de Corrida de Veículos

Data de entrega: 03/12/2019 (até 23h55), via moodle.
O trabalho poderá ser individual ou em grupo de até 2 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 um número inteiro, gerado automaticamente) e uma quantidade de rodas (cujos pneus estarão calibrados ou não de acordo com um sorteio[1]). Além disso, os veículos motorizados terão uma quantidade inicial de combustível, 3,5 litros, o valor e se o IPVA (Imposto sobre a Propriedade de Veículos Automotores) do veículo está pago ou não.[2]

O IPVA de um veículo deverá ser calculado utilizando-se o valor base de R$ 500,00 multiplicado pelo valor da alíquota de cada tipo, ou seja, para motocicleta o valor corresponde a 0,75, para carro popular 1,3 e para ferrari 3,15.

Os veículos motorizados podem ser abastecidos e consomem combustível à medida que se deslocam. Eles apenas se movimentam se há combustível suficiente para tal, se os pneus das rodas estiverem todos calibrados e se o IPVA estiver pago. Assume-se que para mover um "*bloco*" de espaço, a motocicleta gasta 0,25 litros de combustível, o carro polular gasta 0,75 litros e a ferrari gasta 2,3 litros. Portanto, um veículo automotivo não deve se movimentar se:

- não possuir a quantidade de combustível suficiente,
- se um ou mais pneus não estiverem calibrados (também válido para bicicleta) e,
- se o IPVA não estiver pago.

Os veículos devem ser desenhados em modo texto[3], e se movem sempre na horizontal da esquerda para direita de acordo com suas respectivas quantidades de "*blocos*" de espaço (unidade de movimento):

- bicicleta: de dois em dois "*blocos*" de espaço,
- motocicleta: de três em três "*blocos*" de espaço,
- carro popular: de cinco em cinco "*blocos*" de espaço,
- ferrari: de dez em dez "*blocos*" de espaço.

[1]Por exemplo, para cada roda, sorteia-se um número de 0 a 100, se for par, calibra-se o iésimo pneu, caso contrário, não.
[2]Também neste caso, soteia-se um número de 0 a 100, se for par, o IPVA já está pago, caso contrário, não.
[3]Devem ser utilizados, para cada veículo, os desenhos em código *ascii* fornecidos pela professora.

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).

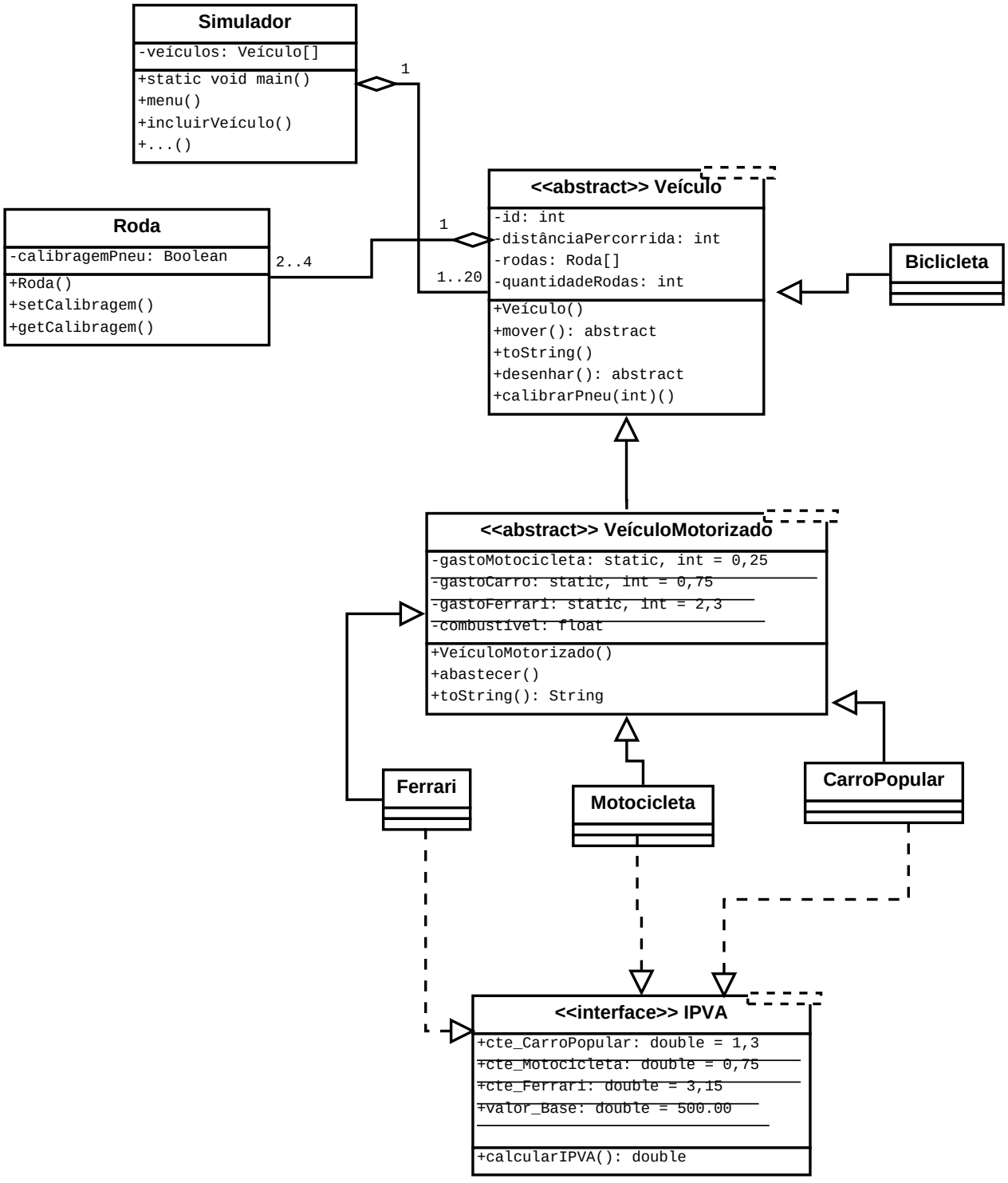


Figura 1: Esboço do diagrama UML a ser seguido.

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* (inteiro) automático para o veículo e assumir que cada pneu está vazio ou não de acordo com um sorteio. Para os automotivos sortear se o IPVA estará pago ou não e calcular o valor do IPVA.]

(b) Remover um veículo

[deve-se informar o *id* do veículo]

(c) Abastecer veículo

[deve-se informar o *id* do veículo e a quantidade de combustível em litros]

(d) Movimentar um veículo

(e) Movimentar veículos por tipo

(f) Imprimir todos os dados de todos os veículos

(g) Imprimir dados de veículos por tipo

(h) Esvaziar um pneu específico,

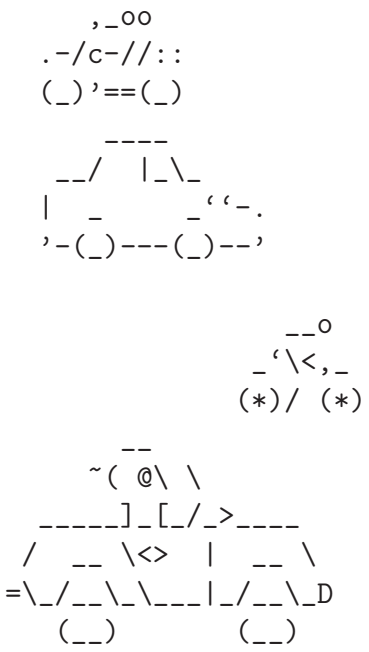
[Solicitar o *id* do veículo e qual pneu será esvaziado (primeiro, segundo,...), caso não seja dada uma entrada correta, repetir a leitura.]

(i) Calibrar um pneu específico

(j) Calibrar todos os pneus por tipo do veículo

(k) Imprimir pista de corrida

[imprime na ordem em que estão no *array*, os veículos com seus respectivos *"blocos"* de espaços percorridos. Para cada tipo de veículo, utilizar os desenhos *ascii* correspondentes, como mostra o exemplo abaixo, que corresponde a uma motocicleta, um carro, uma bicicleta e uma ferrari respectivamente:]



(l) 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).

Detalhamento de itens a serem avaliados:

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`	
Fazer uso do operador `instanceof`	
Ter pelo menos um atributo `static`	
Criar relacionamento entre classes (Agregação ou Composição)	
Fazer uso de classe abstrata	
Fazer uso de interface	
Fazer uso do conceito de herança e polimorfismo	
Não utilizar modificador `protected`	
Não apresentar erro em tempo de execução	
Apresentar o diagrama UML	