Programação Orientada a Objetos

Bacharelado em Engenharia da Computação

Prof. Dr. Eduardo Takeo Ueda

Trabalho 2

Sistema para Controle de Multas de Trânsito

Data limite de entrega: 8 de dezembro de 2024

Suponha que vocês foram contratados pelo DETRAN/SP para implementar um sistema de controle/gerenciamento de multas de trânsito na cidade de São Paulo, utilizando a linguagem Java. Nesse contexto, vocês devem modelar o problema para tratar multas de velocidade, multas de rodízio e multas de tráfego ilegal em corredores de ônibus. Para isso, o sistema deve ler uma base de dados de ocorrências para ser avaliada.

Conforme a avaliação, deverá ser gerada uma multa de acordo com a classificação. Dependendo do tipo de ocorrência a multa pode ter um determinado nível: 0 – sem multa, 1 – multa leve, 2 – multa média, 3 – multa grave. O sistema de software desenvolvido por vocês deverá ter pelo menos as classes da especificação apresentada na Figura 1.

A classe `Ocorrencia` representa uma ocorrência de trânsito. Todos os dados para verificar se uma determinada multa deve ser aplicada estão presentes nessa classe. A classe `BaseDeDados` mantém uma lista de todas as ocorrências, divididas em duas listas: processadas e sem processar, e uma lista das multas, necessária para gerar os relatórios finais. O método `inicializaRegras` deve inicializar manualmente várias regras, por exemplo:

- multas.add(RegraVelocidade(60, "Avenida Washington Luiz"));
- multas.add(RegraVelocidade(70, "Avenida Nações Unidas"));
- multas.add(RegraRodizio(1, "Avenida Bandeirantes", "Avenida 23 de Maio", 1, 1)); *//Rodízio na Avenida 23 de Maio e na Avenida Bandeirantes para veículos leves final de placa 1, às segundas-feiras.*
- multas.add(RegraCorredorOnibus(6, 23, "Avenida Santo Amaro"));
- multas.add(RegraCorredorOnibus(0, 24, "Avenida Vereador José Diniz")).

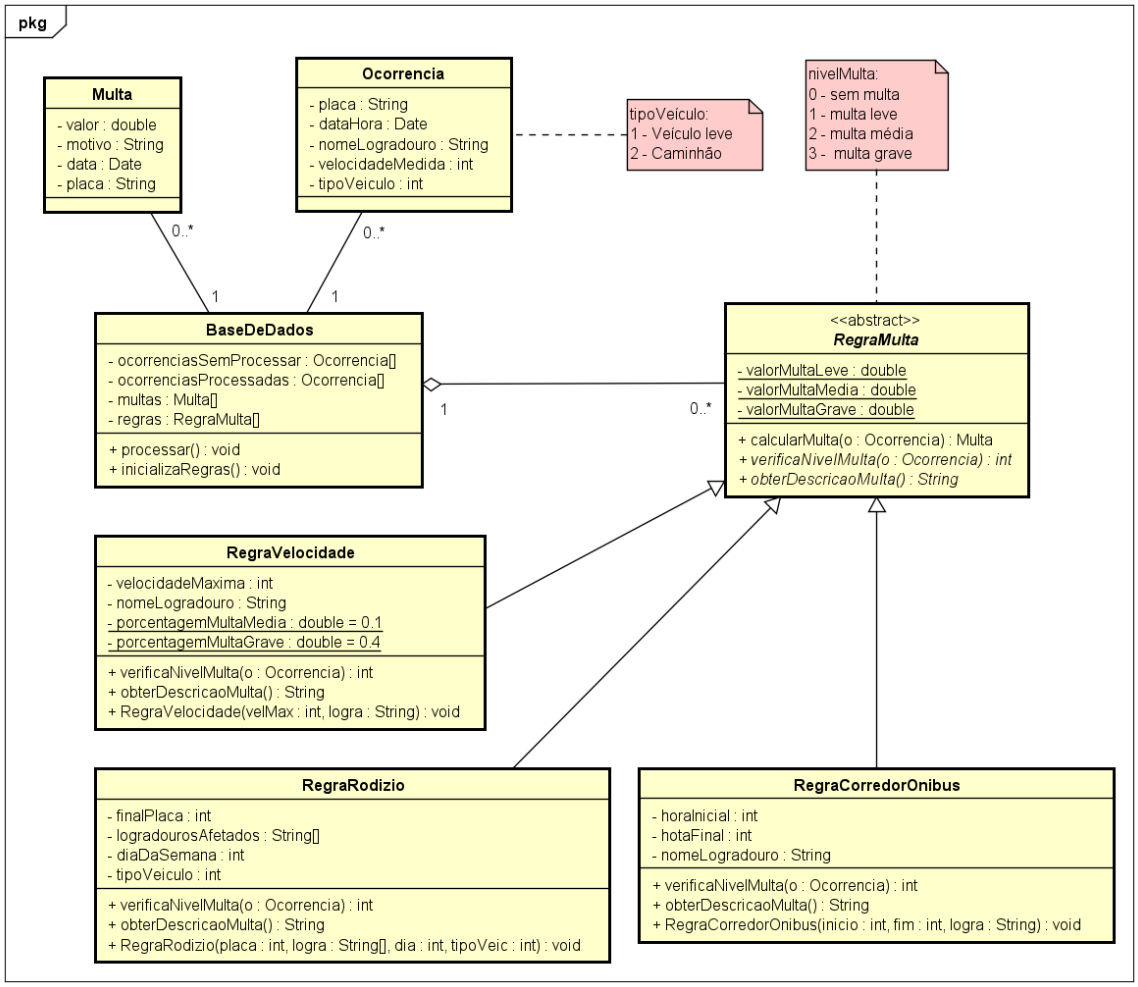

Figura 1: Diagrama de classes

A classe abstrata `RegraMulta`, representa uma multa genérica e possui atributos estáticos, que representam os valores das multas (pois eles não mudam). Ela também possui três métodos: `calcularMulta`, que cria um objeto `Multa` dependendo do nível da multa, `verificaNivelMulta` que retorna um indicador do nível da multa e `obterDescricaoMulta` que retorna um texto com a explicação do tipo de multa. As classes que estendem à classe `RegraMulta` contêm as informações necessárias para verificar cada tipo de multa e devem implementar as regras nos seus respectivos métodos `verificarNivelMulta`. Por sua vez, a classe `Multa` é apenas um repositório de informação de uma multa.

O seu sistema deverá ter uma interface para registrar ocorrências, visualizar as ocorrências cadastradas e visualizar as multas calculadas, podendo filtrar as multas por data e por placa. Para simplificar, considere um número reduzido de logradouros (10), cuidando para que cada regra ocorra pelo menos uma vez. Sempre que precisar digitar um logradouro, liste-os para evitar erros de digitação.

É permitido adicionar novos métodos ou atributos que vocês considerarem necessários, mas não é permitido remover os métodos ou atributos apresentados na especificação UML, conforme o diagrama de classes.

Na correção da implementação Java do sistema de controle de multas serão considerados os seguintes itens: **(1)** implementação das classes conforme a especificação UML fornecida, **(2)** inicialização de regras programaticamente (mínimo: 10 logradouros e 25 regras), **(3)** importação/leitura das ocorrências de um arquivo de texto, **(4)** pesquisa/busca de multas por data e placa, **(5)** interface para cadastro de ocorrências, **(6)** interface para cadastro de novas regras, **(7)** implementação de validações adequadas para as entradas do usuário nas interfaces de cadastro (ocorrências e regras), **(8)** aplicação de comentários no código, **(9)** criação e compartilhamento de um repositório no GitHub para o sistema, **(10)** criação de um arquivo README contendo nomes dos integrantes do grupo, *link* de acesso ao repositório no GitHub, orientações de como executar o código, etc.

Além do código-fonte Java os integrantes do grupo deverão preparar um vídeo, com duração mínima de 10 e máxima de 20 minutos, com uma explicação detalhada do sistema de software implementado. É importante que fique evidente que tal explicação foi apresentada pelos integrantes do grupo, e não por terceiros. Pode ser feito o *upload* do vídeo no Blackboard ou em uma plataforma, como o **YouTube**, e compartilhado o *link* para acesso.

Para a atribuição de nota ao trabalho o peso da implementação, em linguagem Java, é de 50%, e para o vídeo também é 50%, ou seja, tanto a implementação quanto o vídeo valem 5 pontos. Vale destacar que, todos os integrantes do grupo devem apresentar. Caso algum integrante não apresente a nota será descontada de forma proporcional. Por exemplo, em um grupo com 4 integrantes se apenas 2 apresentarem a nota máxima do vídeo será de $2,5$ pontos (metade de 5 pontos).

Recomendações importantes

1. O trabalho deve ser desenvolvido em grupo com no mínimo 2 e máximo 4 alunos.

2. A implementação de interface gráfica para o sistema é opcional, entretanto representa um **bônus** no trabalho e vale 2 pontos extras.

3. Com relação ao uso do Git e GitHub o professor deve ser adicionado no time para acompanhar o desenvolvimento e contabilizar os *commits* (cada aluno deve realizar no mínimo 3 *commits*).

4. Basta apenas um dos integrantes do grupo realizar a submissão do trabalho no Blackboard, lembrando que entregas via *email* não serão aceitas.

5. Não deixe para iniciar a implementação na última hora, como na semana que precede a data limite de entrega.

6. Cuidado, pois implementações que forem confirmados como **plágio** serão penalizados com nota 0 (zero).

Bom trabalho!