Sistema de Mobilidade Urbana

1. Introdução

Este documento apresenta uma descrição detalhada das principais classes do sistema de mobilidade urbana, incluindo suas responsabilidades, atributos, métodos, relacionamentos e os princípios de orientação a objetos aplicados (encapsulamento, herança e polimorfismo).

2. Classes do Domínio

2.1 Classe Usuario (abstrata)

A classe **Usuario** representa a base para todos os tipos de usuários do sistema. Ela contém atributos comuns como nome, CPF, e-mail, telefone e senha.

Serve como superclasse para **Passageiro** e **Motorista**, garantindo reutilização e padronização de atributos essenciais.

Herança:

- Passageiro e Motorista estendem Usuario, aplicando **herança simples**.

Encapsulamento:

- Atributos privados e getters garantem controle de acesso.

Polimorfismo:

- A classe é abstrata, permitindo **polimorfismo de subtipo** (runtime).

- Métodos podem ser reutilizados pelas subclasses com comportamento especializado.

2.2 Classe Passageiro

A classe **Passageiro** herda de Usuario e representa o cliente que solicita corridas.

Atributos principais:

- Lista de métodos de pagamento

- Estado de dívida (estaDevendo)

- Valor que deve (valorQueDeve)

O passageiro pode cadastrar vários métodos de pagamento, regularizar dívidas e solicitar corridas através da interação com **ServicoCorrida**.

Relacionamento:

Passageiro possui uma **agregação** com MetodoPagamento (1 o-- 0..*).

O passageiro gerencia internamente seus métodos de pagamento, mas estes não dependem exclusivamente da sua existência no mundo real.

Encapsulamento:

- Atributos privados e acesso controlado via métodos públicos.

2.3 Classe Motorista

A classe **Motorista** representa o profissional que realiza as corridas. Possui atributos como:

- Status do motorista (ONLINE/OFFLINE/EM_CORRIDA)

- Informações de veículo

- Histórico de corridas

Relacionamento:

Motorista se relaciona com ServicoCorrida através de **associação simples**.

Encapsulamento:

- Acesso a atributos controlado por getters.

2.4 Classe MetodoPagamento (interface)

A interface **MetodoPagamento** define o comportamento geral de um pagamento.

Suas implementações incluem:

- PagamentoCartao

- PagamentoPix

- PagamentoDinheiro

Polimorfismo:

- Polimorfismo **de sobrescrita** (runtime).

- Cada classe implementa `processarPagamento(double)` de forma diferente.

Encapsulamento:

- Cada implementação controla seus dados internos.

2.5 Classe Corrida

A classe **Corrida** é responsável por armazenar os dados de uma viagem:

- Origem e destino

- Distância

- Categoria de serviço

- Passageiro

- Motorista

- Método de pagamento

Relacionamentos:

Associações simples com Passageiro, Motorista e MetodoPagamento.

Encapsulamento:

- Atributos privados e métodos acessores garantem segurança.

2.6 Enumeradores

CategoriaServico, **StatusMotorista** e **StatusCorrida** definem estados e classificações utilizadas pelo sistema.

Eles reforçam a legibilidade e evitam uso incorreto de valores.

2.7 Exceções

As exceções específicas representam condições de erro na lógica do sistema:

- NenhumMotoristaDisponivelException

- PassageiroDevendoException

- PagamentoRecusadoException

Elas fortalecem o fluxo de controle.

3. Classe ServicoCorrida

A classe **ServicoCorrida** é responsável por gerenciar a operação de solicitação de corridas.

Ela mantém uma lista de motoristas disponíveis e fornece métodos para procurar motoristas e criar corridas.

Relacionamentos:

- ServicoCorrida --> Motorista: associação simples.

- ServicoCorrida --> Corrida: dependência, pois cria objetos Corrida.

- ServicoCorrida --> Passageiro: associação simples.

- ServicoCorrida --> MetodoPagamento: associação simples.

Encapsulamento:

- A lista de motoristas é privada e só manipulada por métodos públicos.

4. Princípios de Orientação a Objetos Aplicados

4.1 Encapsulamento

Aplicado em:

- Usuario, Passageiro, Motorista: atributos privados + getters/setters.

- Corrida: estado interno protegido.

- Métodos de pagamento: controle privado de dados (saldo, limite etc.).

4.2 Herança

Aplicada na hierarquia:

- Usuario $\rightarrow$ Passageiro

- Usuario $\rightarrow$ Motorista

Permite reutilização de atributos e comportamentos comuns.

4.3 Polimorfismo

Há dois tipos presentes no sistema:

1. Polimorfismo de Subtipo (runtime)

Utilizado com `Usuario` (classe abstrata).

Qualquer método que receba um Usuario aceita Passageiro ou Motorista.

2. Polimorfismo por Sobrescrita (runtime)

Usado em:

- `processarPagamento()` nas implementações de MetodoPagamento

(Cartão, Pix, Dinheiro)

Cada implementação executa o método de forma distinta.

5. Conclusão

A modelagem do sistema segue princípios sólidos de orientação a objetos:

- Encapsulamento garante segurança e controle.
- Herança organiza tipos de usuários.
- Polimorfismo permite flexibilidade e extensão de comportamentos.
- As relações entre classes são coerentes e escaláveis, seguindo o domínio real do sistema.