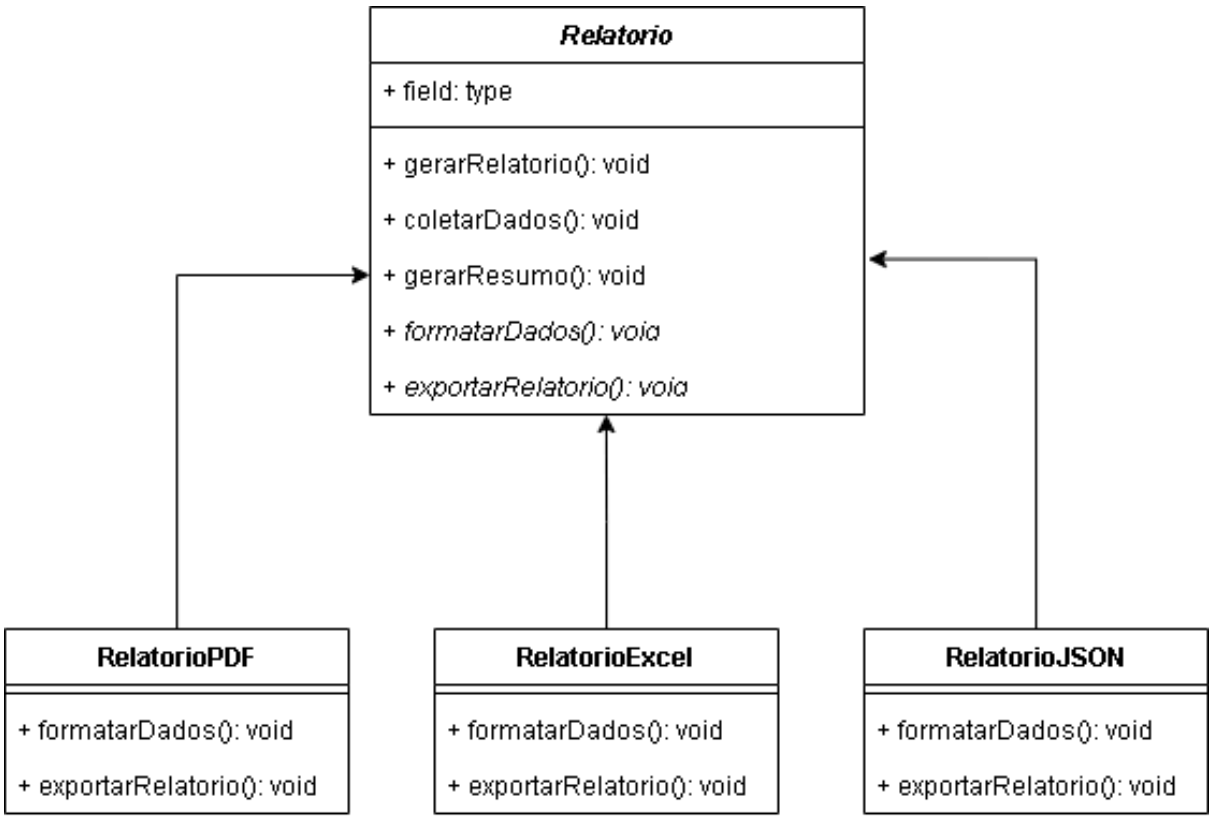
Exercício 1: Sistema de Transporte Urbano com Diferentes Tarifas

Imagine que você está desenvolvendo um sistema para gerar relatórios de vendas em diferentes formatos. Cada formato de relatório (PDF, Excel, JSON) possui uma estrutura básica de geração comum, mas varia nos detalhes de exportação e formatação de dados.

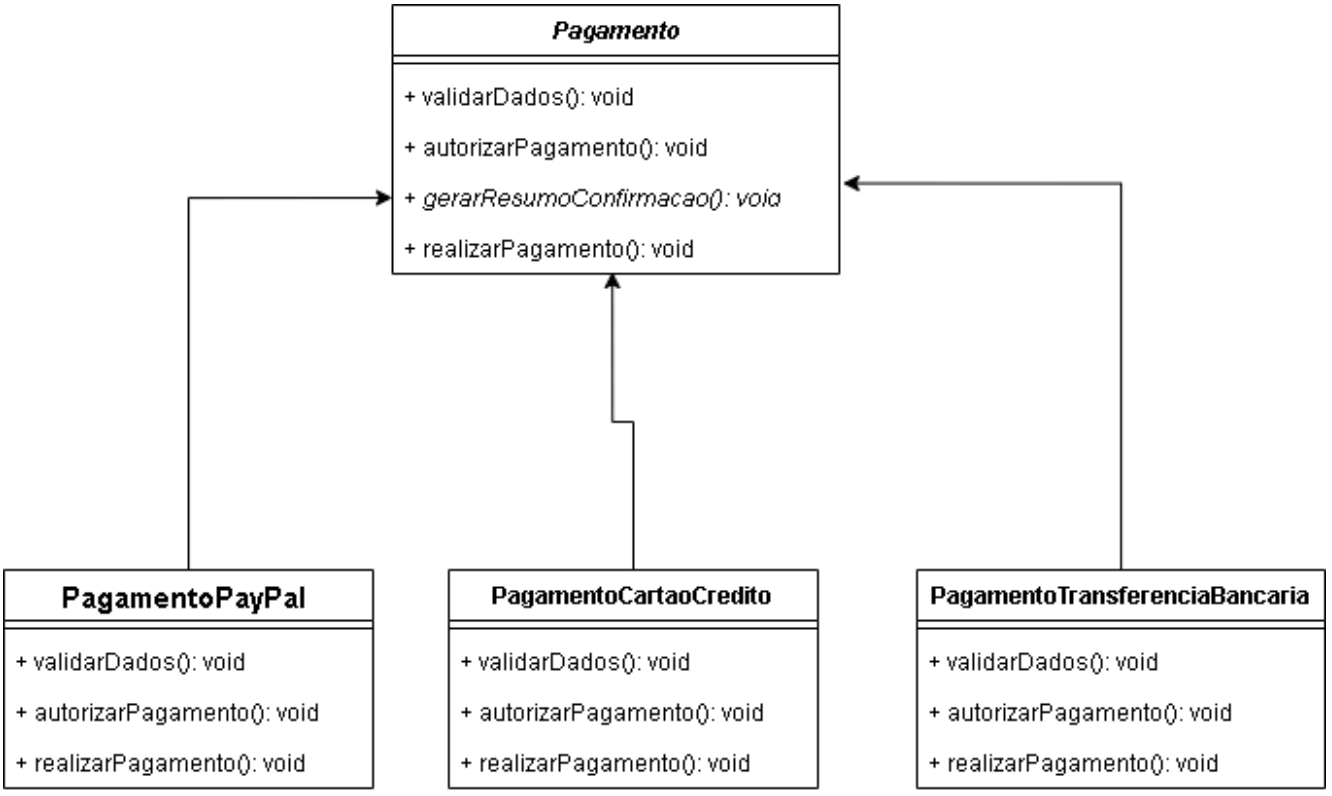
- **Objetivo**: Implemente um sistema que permita a geração de relatórios de vendas em diferentes formatos, mantendo uma estrutura de passos fixa.
- **Desafio**: Cada formato de relatório precisa de detalhes de formatação específicos, mas deve seguir uma sequência de geração de dados, resumo e exportação.
- **Tarefa**: Crie uma classe base com um método de geração de relatório que defina os passos fixos, permitindo a extensão para diferentes formatos. Use o padrão *Template Method* para que as subclasses implementem apenas os detalhes específicos do formato.



Exercício 2: Sistema de Pedidos com Diferentes Processos de Pagamento

Imagine que você está criando um sistema de pedidos para uma loja que aceita vários métodos de pagamento, como cartão de crédito, PayPal e transferência bancária. Cada método de pagamento tem um processo específico, mas todos seguem uma estrutura comum de validação, autorização e confirmação do pagamento.

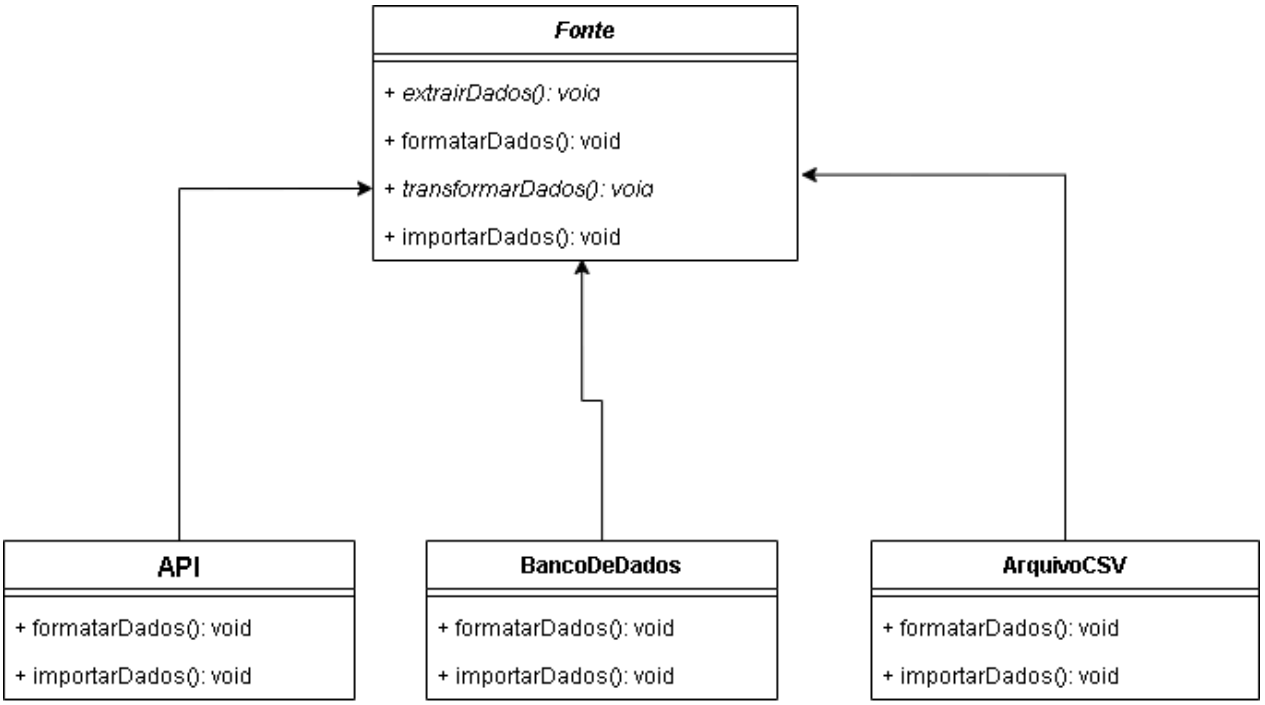
- **Objetivo**: Desenvolver um sistema que permita o processamento de diferentes tipos de pagamento, aplicando uma sequência comum de passos.
- **Desafio**: Cada método de pagamento possui detalhes únicos para autorização e validação, mas todos compartilham o mesmo fluxo.
- **Tarefa**: Crie uma classe base que define a sequência de passos do processo de pagamento. As subclasses devem especializar apenas as etapas específicas de validação e autorização, implementando o *Template Method* para permitir que o sistema expanda com novos métodos de pagamento.



Exercício 3: Sistema de Importação de Dados com Diferentes Fontes

Imagine que você está desenvolvendo um sistema de importação de dados para uma aplicação que coleta informações de diferentes fontes (API, banco de dados, arquivo CSV). Cada fonte tem suas próprias especificidades de conexão, extração e transformação de dados.

- **Objetivo**: Implemente um sistema de importação de dados que permita conectar-se a diferentes fontes seguindo um fluxo de etapas padrão.
- **Desafio**: Cada fonte de dados precisa de uma implementação específica para conectar, extrair e formatar os dados, mas o fluxo de importação deve seguir a mesma estrutura.
- **Tarefa**: Crie uma classe base que define o método de importação como um *Template Method*, com etapas de conexão, extração e transformação de dados. As subclasses devem implementar os detalhes de cada fonte de dados, mantendo o fluxo de importação consistente.



Exercício 4: Sistema de Notificações com Diferentes Canais de Comunicação

Imagine que você está desenvolvendo um sistema de notificações que envia mensagens através de diferentes canais (e-mail, SMS, push notification). Cada canal de comunicação tem seu próprio processo para configurar, formatar e enviar uma mensagem.

- **Objetivo**: Desenvolver um sistema de notificação que permite enviar mensagens através de diferentes canais, seguindo um fluxo estruturado.
- **Desafio**: Cada canal de comunicação possui configurações e processos específicos, mas o fluxo de envio de notificação (configuração, formatação, envio) deve ser o mesmo.
- **Tarefa**: Crie uma classe base que define o método de envio como um *Template Method*, com passos de configuração, formatação e envio. As subclasses devem implementar as particularidades de cada canal de comunicação, permitindo que novos canais sejam adicionados facilmente no futuro.