# Modelagem de Classes

O texto a seguir está relacionado com as questões 1 a 5

*A empresa tem uma grande frota de carros de passeio, sendo que esses carros apresentam diferentes marcas e modelos. Eventualmente um carro pode ser retirado da frota devido a algum acidente ou simplesmente por ter sido considerado velho demais para o padrão da empresa e tenha sido vendido. Da mesma forma, a empresa eventualmente renova a frota, sendo necessário, portanto, manter o cadastro de veículos da empresa.*

*Os clientes dirigem-se a empresa e solicitam o aluguel de um carro ou carros. No entanto, primeiramente é necessário cadastrá-los, caso ainda não possuam cadastro ou se dados tenham sido alterados.*

*Depois de ter se identificado/cadastrado, o cliente escolherá o carro que deseja alugar (o valor da locação varia de acordo com o ano, marca e modelo). Durante o processo de locação, o cliente deve informar por quanto tempo utilizará o carro, para qual finalidade e por onde desejará trafegar, já que essas informações também influenciam o preço de locação. Antes de liberar o veículo, a empresa exige que o cliente forneça um valor superior ao estabelecido na análise da locação, a título de caução. Caso o cliente não utilize todo o valor de caução até o momento da devolução do veículo, o valor restante lhe será devolvido. Quando o cliente devolve o carro deve-se definir o automóvel como devolvido, registrar a data e hora da devolução e a quilometragem em que se encontra, bem como verificar se o automóvel se encontra nas mesmas condições em que foi alugado. Caso o cliente tenha ocupado o carro por mais tempo que o combinado, deve pagar o aluguel referente ao tempo extra em que permaneceu com o veículo. Da mesma maneira, o cliente deverá pagar por qualquer dano sofrido pelo veículo quando este se encontrava locado. Por outro lado, o cliente pode ser ressarcido de parte do valor que pagou caso o custo do tempo em que esteve de posse de veículo seja inferior ao valor previamente fornecido.*

1. Desenvolva o diagrama de classes de análise. Cada classe deve apresentar no mínimo dois atributos, a interface do objeto e duas operações relacionadas com o negócio.

Diagrama

Descrição gerada automaticamente

2. Transforme o modelo de análise para o modelo de projeto utilizando apenas dependência estrutural. Diagrama

Descrição gerada automaticamente

# Dependência

1. Transforme as dependências estruturais em dependências não estruturais onde for possível

Diagrama

Descrição gerada automaticamente

1. Apresente a estrutura básica de código em JAVA para implementar as associações.

**Cria duas classes simples**

class Cliente { String nome; Pessoa(String nome) { this.nome = nome; } }

class Carro { String nome; Livro(String nome) { this.nome = nome; } }

**Cria uma classe que tem duas outras associadas a ela em java**

class Emprestimo {

Cliente cliente;

Carro carro;

Emprestimo(Cliente cliente, Carro carro) { this.cliente= cliente this.carro = carro; }

}

# Dependência

1. Quais são os tipos de dependências? Como pode ser reduzida a dependência do modelo abaixo?



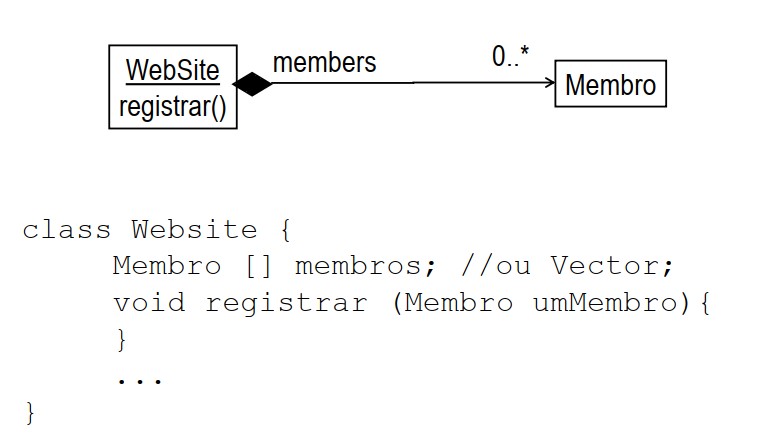
1. **Dependência de Uso:**
   * **Descrição:** Uma classe depende diretamente de outra, usando seus métodos ou propriedades.
2. **Dependência de Agregação:**
   * **Descrição:** Uma classe depende de outra, mas a relação é mais fraca; a classe dependente pode existir independentemente da classe principal.
3. **Dependência de Composição:**
   * **Descrição:** Uma classe depende fortemente de outra; a classe dependente é parte integrante da classe principal.
4. **Dependência de Herança:**
   * **Descrição:** Uma classe depende de outra através da herança; a classe derivada herda propriedades e comportamentos da classe base.
5. **Dependência de Interface:**
   * **Descrição:** Uma classe depende de uma interface, seja por implementação ou pela garantia de um contrato definido.

Pode ser reduzido da seguinte maneira:

Diagrama

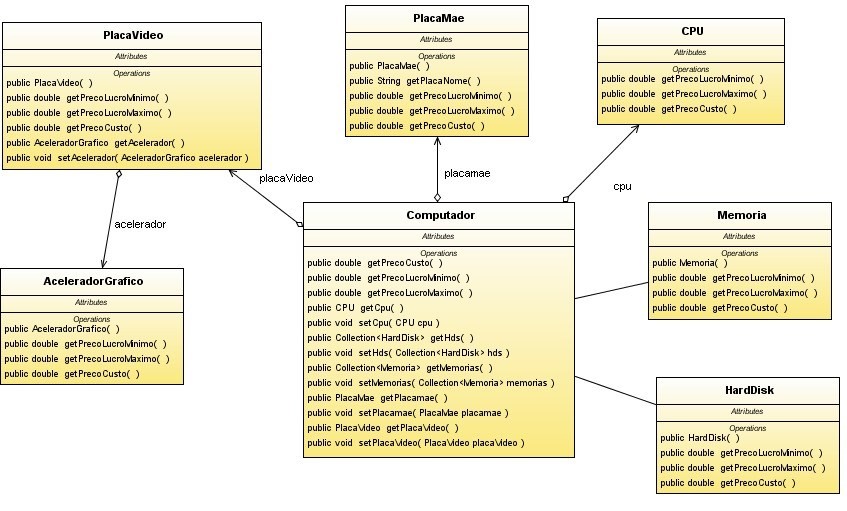
Descrição gerada automaticamente

1. Sob a ótica do reuso, qual é o problema modelo abaixo?



O problema é o acoplamento forte gerado nessa relação entre as classes, que limita o reuso visto que se fizer uma alteração na classe impactará diretamente a outra classe, e a manutenção se tornará cada vez mais complexa

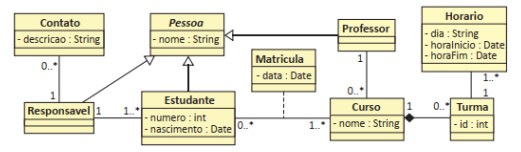
1. Dado o modelo abaixo, qual seria o problema dele? Justifique sua resposta

Nesse caso especifico o problema é que o sistema tem um conjunto de classes que se comunicam umas com as outras e o acoplamento por conta disso é muito forte. E pode levar a uma série de desafios, incluindo dificuldade de manutenção, reutilização limitada e complexidade na compreensão do código.

1. Dada a questão 7, qual padrão de projeto poderia ser aplicado no modelo?

O padrão Mediator poderia ser aplicado a esse modelo porque ele gerencia as interações entre as classe e o acoplamento forte é reduzido, porque agora as classes devem se comunicar por meio do Mediator, e não mais por relações diretas entre elas, o que facilita a reutilização de classes nos diferentes contextos, torna o sistema mais modular e fácil de manter

1. Dado o modelo de classes a seguir, transforme-o no modelo ER



TRANSFORMAÇÃO:

Diagrama

Descrição gerada automaticamente

1. Analise o cenário abaixo e realize os exercícios na sequência.

Em um sistema de hotel, uma classe consumidora necessita dos serviços de três classes fornecedoras:

* + - Atualização do telefone e e-mail do Cliente;
    - Obtenção e atualização do status do Quarto (disponível ou indisponível);
    - Criação, cancelamento e atualização da Reserva.

* 1. Modele uma solução para o cenário apresentado, aplicando o padrão de projeto Observer. Justifique a resposta.

Diagrama

Descrição gerada automaticamente

* 1. Modele uma solução para o cenário apresentado, aplicando o padrão de projeto Façade.

Diagrama

Descrição gerada automaticamente

* 1. O sistema de processamento de pagamentos online. O sistema deve ser capaz de processar pagamentos usando diferentes métodos, como cartão de crédito, transferência bancária e carteira digital. A

lém disso, os usuários devem ter a flexibilidade de escolher o método de pagamento desejado. Implemente o sistema utilizando o padrão de projeto Strategy. Crie interfaces para as estratégias de processamento de pagamento e implementações concretas para cada método de pagamento. A classe principal do sistema deve aceitar diferentes estratégias e processar pagamentos conforme a estratégia escolhida.

Considere os seguintes requisitos:

* + - O sistema deve oferecer suporte aos seguintes métodos de pagamento: Cartão de Crédito, Transferência Bancária, Pix e Carteira Digital.
    - Cada método de pagamento pode ter lógicas específicas de processamento (por exemplo, validação de cartão de crédito, confirmação de transferência bancária, verificação de saldo na carteira digital).
    - O usuário deve poder escolher o método de pagamento a ser utilizado em tempo de execução.
    - O sistema deve exibir uma mensagem indicando o sucesso ou falha do processamento do pagamento.

Diagrama

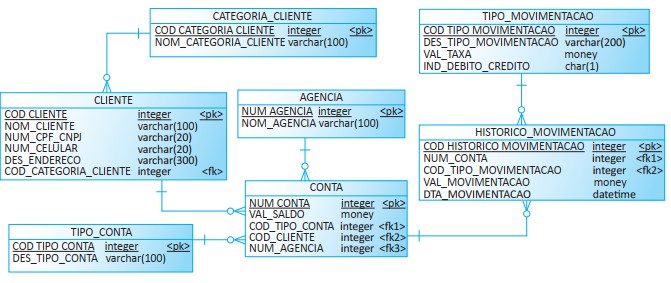
Descrição gerada automaticamente

1. Qual é a categoria de cada um padrão de projeto utilizados neste exercício?

Observer  **Padrões Comportamentais**

Façade  \_**Padrões Estruturais**

1. Dado o modelo ER, faça o modelo de classes correspondente.



Resolução: Diagrama

Descrição gerada automaticamente

1. Você foi designado para desenvolver um sistema de ordenação de listas. O sistema deve ser capaz de ordenar listas de diferentes tipos de objetos, como números inteiros, strings, ou objetos customizados. Além disso, o usuário deve poder escolher o algoritmo de ordenação a ser utilizado.

Implemente o sistema utilizando o padrão de projeto Strategy. Crie interfaces para as estratégias de ordenação e implementações concretas para diferentes algoritmos de ordenação. A classe principal do sistema deve aceitar diferentes estratégias e ordenar listas conforme a estratégia escolhida.

Considere os seguintes requisitos:

* 1. O sistema deve oferecer suporte aos seguintes algoritmos de ordenação: Bubble Sort, Quick Sort e Merge Sort.
  2. O sistema deve permitir a ordenação de listas de inteiros, strings e objetos customizados (por exemplo, objetos que possuam um atributo numérico a ser utilizado na ordenação).
  3. O usuário deve poder escolher a estratégia de ordenação a ser utilizada em tempo de execução.
  4. O sistema deve exibir a lista antes e depois da ordenação.

Crie um exemplo de uso em programação Java, ordenando diferentes tipos de listas com diferentes algoritmos.