Justificativa para as classes Veiculo e Caminhao:

1. **Single Responsibility Principle (Princípio da Responsabilidade Única)**:
    - A interface **Veiculo** tem a responsabilidade de representar um veículo alugável e definir métodos para calcular o valor do aluguel e o imposto.
    - A classe **Caminhao** tem a responsabilidade de representar um tipo específico de veículo (caminhão) e implementar os cálculos específicos para esse tipo de veículo.
2. **Open-Closed Principle (Princípio Aberto/Fechado)**:
    - Ambas as classes estão abertas para extensão, pois novos tipos de veículos podem ser adicionados estendendo a classe **VeiculoAlugavel** sem a necessidade de modificar as classes existentes.
    - Por exemplo, se quisermos adicionar um novo tipo de veículo, como **Van**, podemos estender a classe **VeiculoAlugavel** e implementar os cálculos específicos para esse tipo de veículo.
3. **Liskov Substitution Principle (Princípio da Substituição de Liskov)**:
    - A classe **Caminhao** é uma especialização da classe **VeiculoAlugavel**, portanto, pode ser substituída por objetos do tipo **VeiculoAlugavel** sem afetar o comportamento do sistema.
    - Isso permite que os objetos do tipo **Veiculo** sejam tratados de maneira uniforme, independentemente do tipo específico de veículo.
4. **Interface Segregation Principle (Princípio da Segregação de Interface)**:
    - A interface **Veiculo** contém apenas os métodos necessários para calcular o valor do aluguel e o imposto, não incluindo métodos irrelevantes.
    - Isso garante que as interfaces sejam coesas e específicas para os requisitos dos clientes.
5. **Dependency Inversion Principle (Princípio da Inversão de Dependência)**:
    - Ambas as classes dependem de abstrações (interfaces) em vez de implementações concretas, o que torna o código mais flexível e fácil de manter.
    - Por exemplo, a classe **Caminhao** depende da interface **Veiculo**, não de uma implementação específica, o que permite que diferentes tipos de veículos possam ser utilizados sem modificar o código da classe **Caminhao**.

Teste de Coesão (LCOM) para as classes Veiculo e Caminhao:

Para a classe **VeiculoAlugavel**, temos dois métodos:

1. **calcularImposto()**
2. **calcularValorAluguel()**

Esses métodos não acessam atributos da classe **VeiculoAlugavel**, apenas utilizam os valores **valorBase** e **taxaImposto** que são fornecidos no construtor.

Métodos: **calcularImposto()**, **calcularValorAluguel()**

Atributos acessados: **valorBase**, **taxaImposto**

Como ambos os métodos acessam os mesmos atributos, temos a interseção não vazia. Portanto, o valor de LCOM para **VeiculoAlugavel** é 0.

Para a classe **Caminhao**:

A classe **Caminhao** não tem métodos adicionais além dos herdados de **VeiculoAlugavel**, portanto, não há novos métodos para analisar.

Para a classe **Caminhao**, como herda diretamente de **VeiculoAlugavel** e não adiciona métodos novos, o valor de LCOM será o mesmo que **VeiculoAlugavel**, ou seja, 0.

Teste de Acoplamento (CBO) entre as classes Veiculo e Caminhao:

Para a classe **VeiculoAlugavel**:

1. Chama um método de outra classe - 0
2. Acessa um atributo público de outra classe - 0
3. Herda de outra classe - 0
4. Declara uma variável local, um parâmetro ou um tipo de retorno do tipo de outra classe - 0
5. Captura uma exceção do tipo de outra classe - 0
6. Levanta uma exceção do tipo de outra classe - 0
7. Cria um objeto do tipo de outra classe - 0

Portanto, para a classe **VeiculoAlugavel**, CBO(VeiculoAlugavel) = 0.

Para a classe **Caminhao**:

1. Chama um método de outra classe - 0
2. Acessa um atributo público de outra classe - 0
3. Herda de outra classe (VeiculoAlugavel) - 1
4. Declara uma variável local, um parâmetro ou um tipo de retorno do tipo de outra classe - 0
5. Captura uma exceção do tipo de outra classe - 0
6. Levanta uma exceção do tipo de outra classe - 0
7. Cria um objeto do tipo de outra classe (VeiculoAlugavel) - 1

Portanto, para a classe **Caminhao**, CBO(Caminhao) = 2.

Isso significa que **VeiculoAlugavel** não tem dependências com outras classes, enquanto **Caminhao** depende da classe **VeiculoAlugavel**.