Herança vs. Composição

Projeto de Software

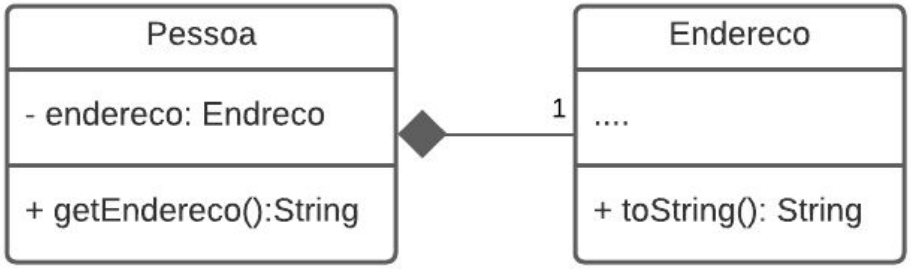
Fabio Morais

Baseado no material dos Professores Nazareno Andrade e Hyggo Almeida

Herança e Composição

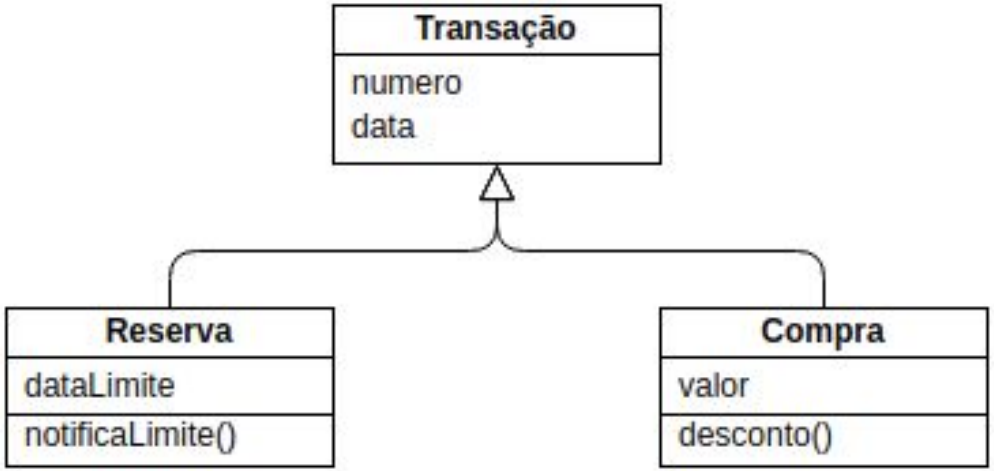
- Dois mecanismos de reutilização de funcionalidades
  - Herança **era** a ferramenta básica de extensão e reuso
  - Problemas de **encapsulamento** e **acoplamento**
- Herança estende **atributos** e **métodos** de uma classe
- **Composição** estende uma classe por **delegação**

Composição



- Delegação de trabalho para outros objetos
  - Relacionamento do tipo **tem-um**
- Exemplo com endereços
  - A classe pessoa tem um endereço, mas as informações de endereço são de responsabilidade da classe Endereco

Herança



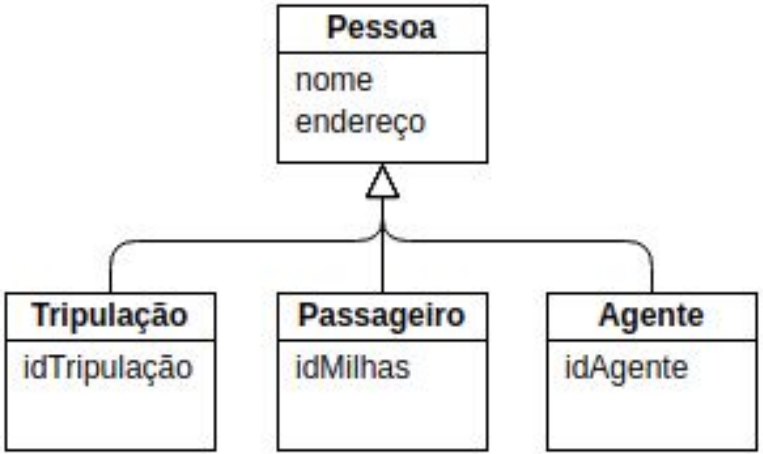
- Captura o que é comum e isola o que é diferente
  - Atributos e métodos comuns vão na superclasse
  - Especializações são realizadas nas subclasses
  - Relacionamento do tipo **é-um**

Problemas da Herança

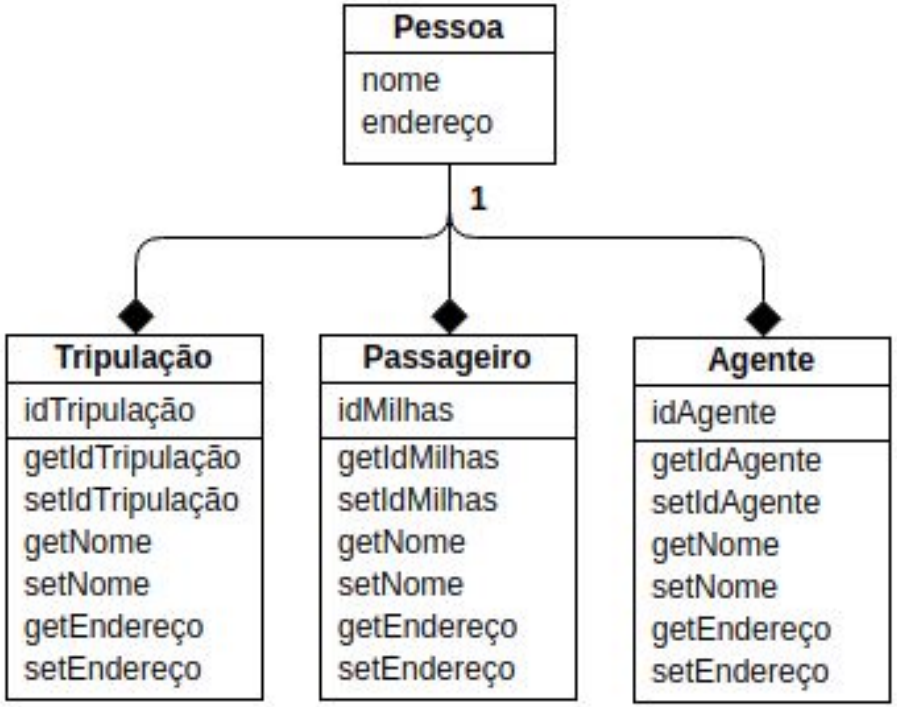
- Fraco **encapsulamento** e forte **acoplamento**
    - Mudanças na superclasse podem afetar as subclasses
- O relacionamento entre superclasse e subclasse é **estático**
    - Não permite que um objeto mude de tipo dinamicamente

Problemas da Herança

Como fazer para uma pessoa assumir diferentes papéis, em diferentes momentos?



Usando Composição

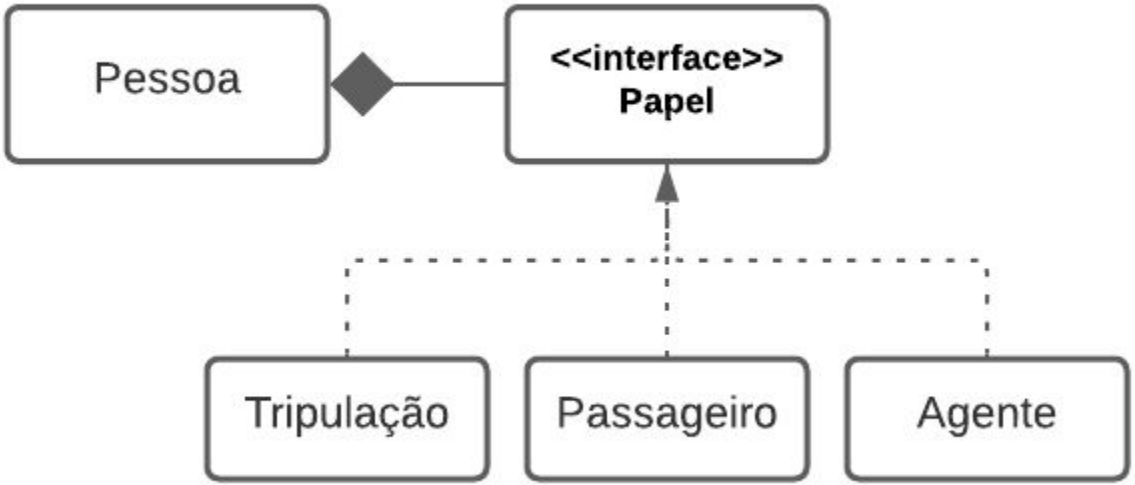


Cada papel tem uma Pessoa

- Delegação de funcionalidades (ex. setEndereco)
- Semelhante a uma subclasse delegar para uma superclasse

Outra solução com Composição

- Pessoa delega responsabilidades a Papel



Benefícios da Composição

- Objetos só são acessados por sua interface
- Menor dependência de implementação
  - Melhor distribuição de responsabilidades
  - Maior encapsulamento das classes
- Sempre pode ser usada no lugar de herança

Problema da Composição

- Mais difícil de entender 😬
  - Software muito dinâmico e parametrizado
  - A relação não é estática e visível explicitamente no código

Princípio de Orientação a Objetos

Sempre prefira **composição** ao invés de **herança**!

(ou use herança com moderação)

Quando usar Herança?

1. A subclasse “**é um tipo especial de**” e não “um papel assumido por”

2. Uma instância da subclasse **nunca tem que mudar** de classe

3. A subclasse estende a superclasse mas **não sobrescreve ou anula** os atributos e/ou métodos

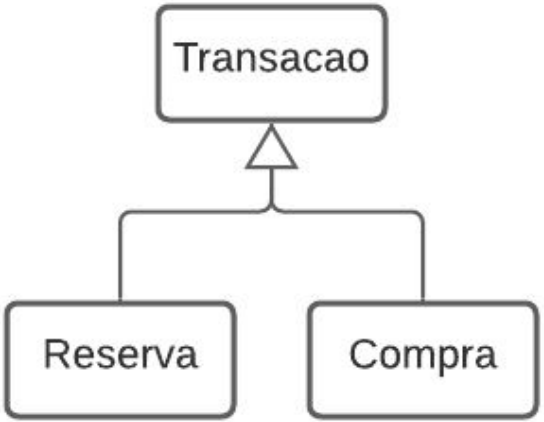
Quando usar Herança?

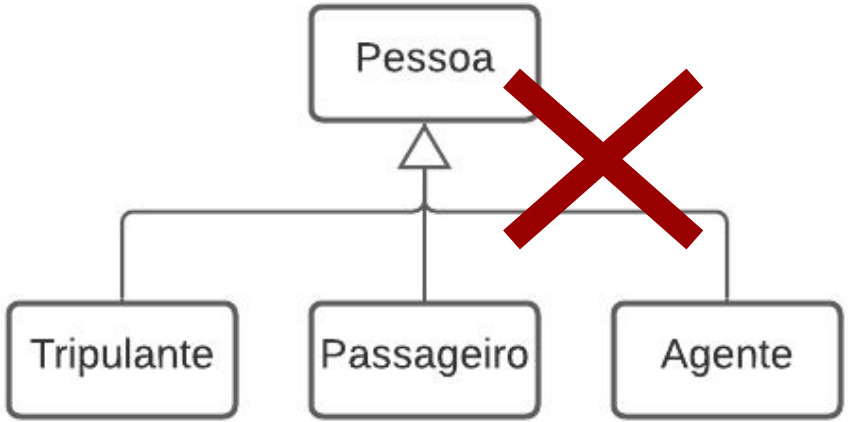
4. A superclasse **não é uma classe utilitária**

5. Para classes do domínio do problema, a subclasse expressa **tipos especiais** de **papéis**, **transações** ou **dispositivos**

Quando usar Herança?

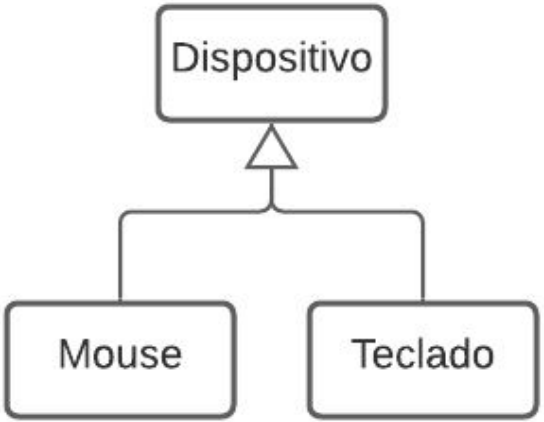
O objeto é “**um tipo especial de**” e não um “papel assumido por”

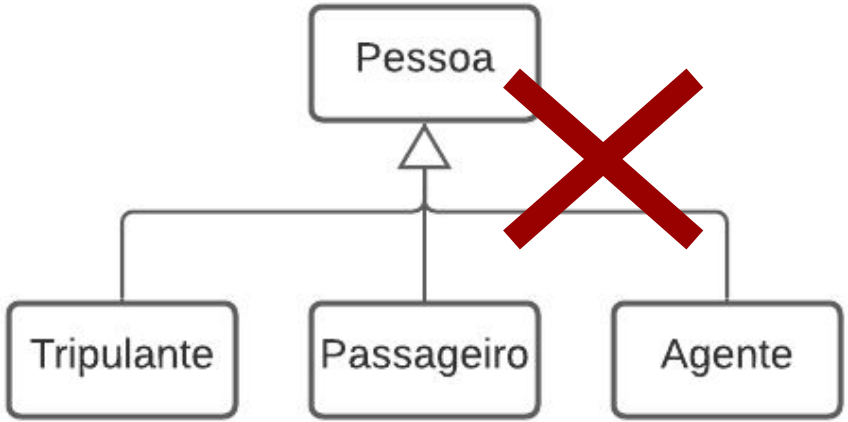




Quando usar Herança?

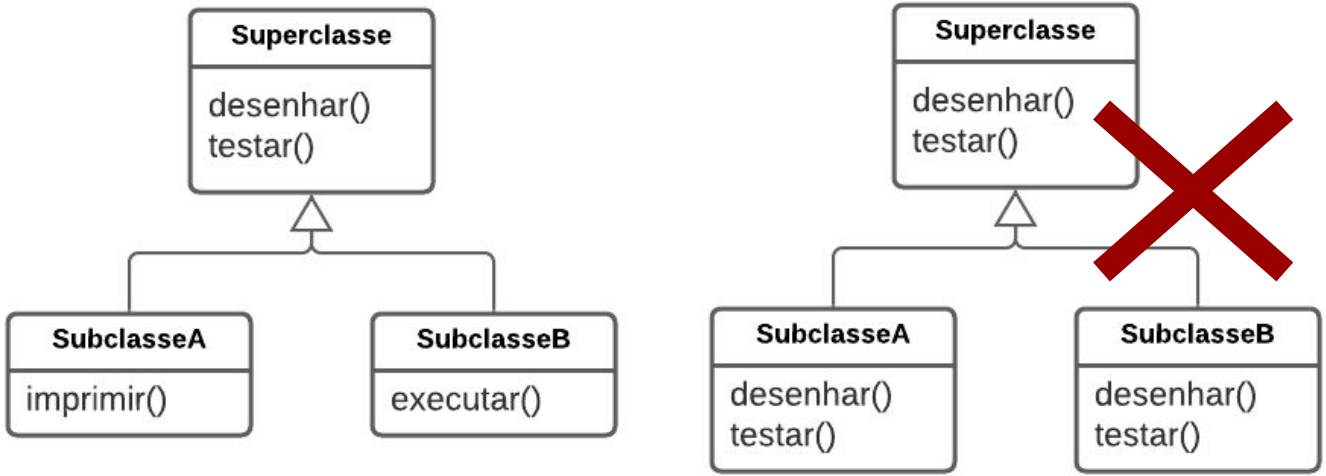
O objeto **nunca tem que mudar** para outra classe





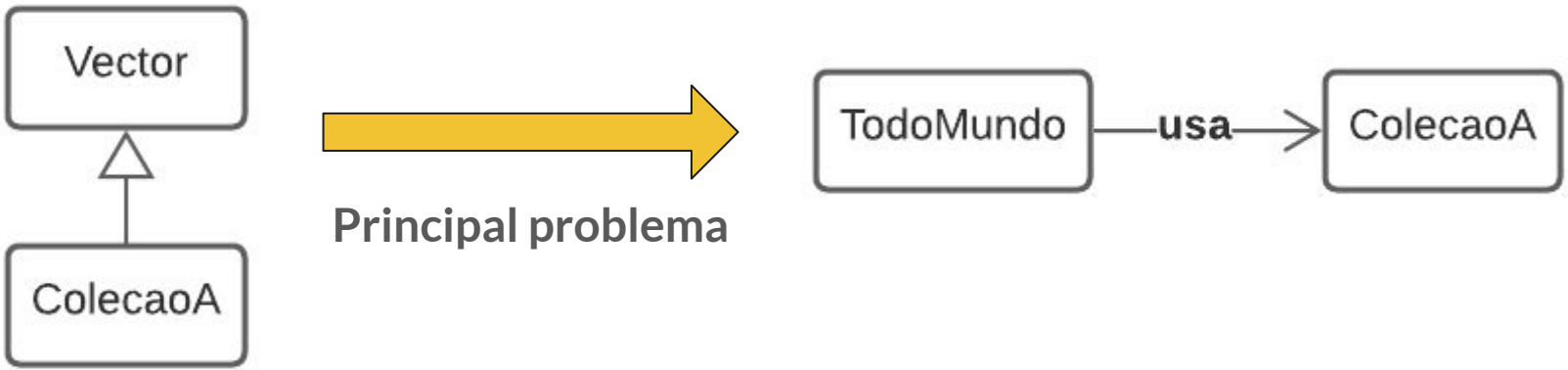
Quando usar Herança?

As subclasses **não sobrescrevem** os métodos da superclasse



Quando usar Herança?

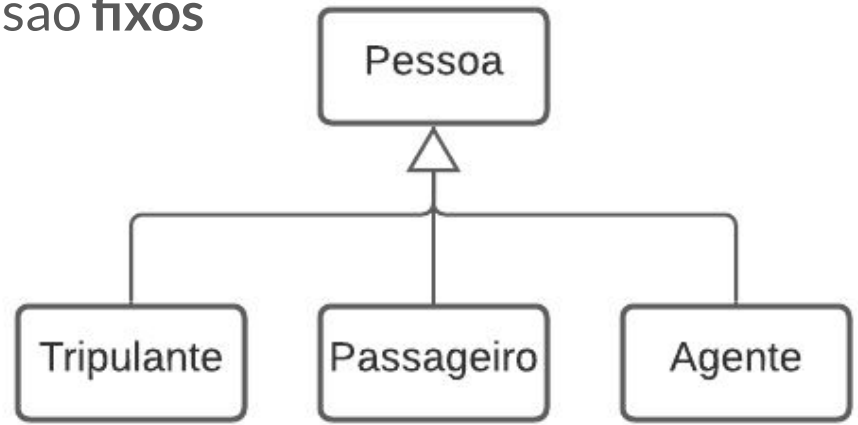
A superclasse **não é uma classe utilitária**



Quando usar Herança?

No contexto do problema **papéis representam tipos especiais**

- Tripulante, passageiro e agente são **fixos**



Herança vs. Composição

Projeto de Software