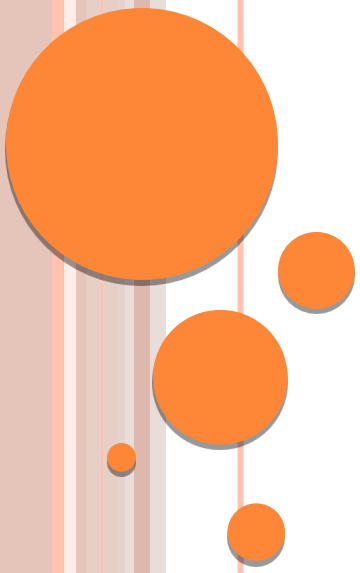


Metodologia e Linguagem de Programação Orientada a Objetos

Drª. Alana Morais

Problema Inicial

1- Implemente a classe **Aluno.java**

- Atributos: nome, matrícula, endereço, turma, sexo.

Problema Inicial

1- Implemente a classe **Aluno.java**

- Atributos: nome, matrícula, endereço, turma, sexo.
- Implemente o método:

matricularAlunoTurma(String turma) .

Problema Inicial

1- Implemente a classe **Aluno.java**

- Atributos: nome, matrícula, endereço, turma, sexo.
- Implemente o método:

matricularAlunoTurma(String turma) .

2- Crie a classe de teste **Teste.java**

Problema Inicial

1- Implemente a classe **Aluno.java**

- Atributos: nome, matrícula, endereço, turma, sexo.
- Implemente o método:

matricularAlunoTurma(String turma) .

2- Crie a classe de teste **Teste.java**

- Adicione uma lista de alunos.

Problema Inicial

1- Implemente a classe **Aluno.java**

- Atributos: nome, matrícula, endereço, turma, sexo.
- Implemente o método:

matricularAlunoTurma(String turma) .

2- Crie a classe de teste **Teste.java**

- Adicione uma lista de alunos.
- Matricule cada aluno em uma turma.

Problema Inicial

1- Implemente a classe **Aluno.java**

- Atributos: nome, matrícula, endereço, turma, sexo.
- Implemente o método:

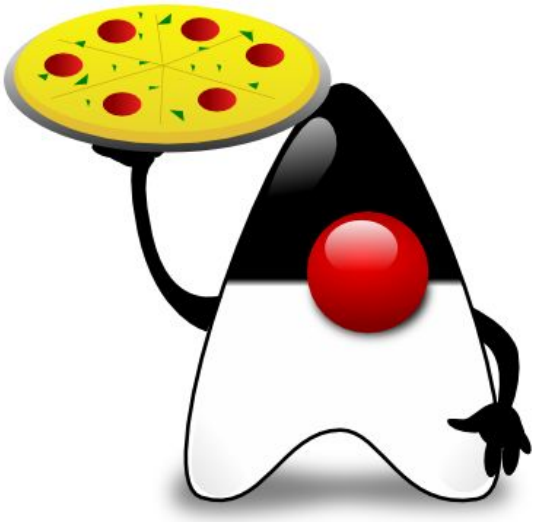
matricularAlunoTurma(String turma) .

2- Crie a classe de teste **Teste.java**

- Adicione uma lista de alunos.
- Matricule cada aluno em uma turma.
- Adicione um **aluno de intercâmbio** e imprima a **faculdade de origem** deste aluno.

O QUE DEVEMOS FAZER???

Que tal voltar para metodologia de Orientação a Objetos? Use os fundamentos da **HERANÇA** em Java.



Herança?? Mas na minha cabeça isso é outra coisa?? Não vejo como usar isso em OO.

O que você lembra com a palavra herança??

- Dinheiro
- Receber algo
- Relacionamento entre as partes
- Posse
- Legado, domínio

O que você lembra com a palavra herança??

- ~~Dinheiro~~
- Receber algo
- Relacionamento entre as partes
- Posse
- Legado, domínio

Mas será que a Herança de Java não tem nenhuma relação com a herança na vida real ?

Vamos dar uma pausa no nosso problema!!

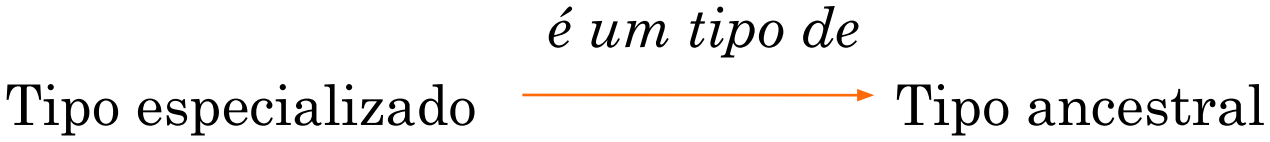
Herança em java

- Herança é uma outra forma de **reutilizar código**.
- Permite a criação de objetos que são parecidos, em comportamento, com outros objetos ancestrais.
- Em orientação a objetos, as relações de **generalização/especialização** são implementadas através do mecanismo de herança.

Herança em java

- Conceito:
  - Herança é a capacidade de especializar tipos de objetos (classes), de forma que os tipos especializados contenham, além de características estruturais e comportamentais já definidas pelos seus "ancestrais", outras definidas para eles próprios.

- Forma:



Herança em Java

- A herança pode ser feita:
  - Com classes já construídas pelo próprio programador.
  - Com classes de terceiros ou classes-padrão da linguagem Java.

- Em Java, a palavra-chave utilizada para criar uma subclasse é **extends**.
  - A sintaxe para uso de **extends** é a seguinte:

```
[Subclasse a ser criada] extends [Superclasse existente]
```

Herança em java

- SuperClasse
  - Em uma relação de herança, é a classe **mais genérica**, cuja estrutura e comportamento são herdados pela outra classe.

- SubClasse:
  - Em uma relação de herança, é a classe **mais específica**, que herda a estrutura e comportamento da outra classe.
  - Classe que faz o extends.

Exemplo

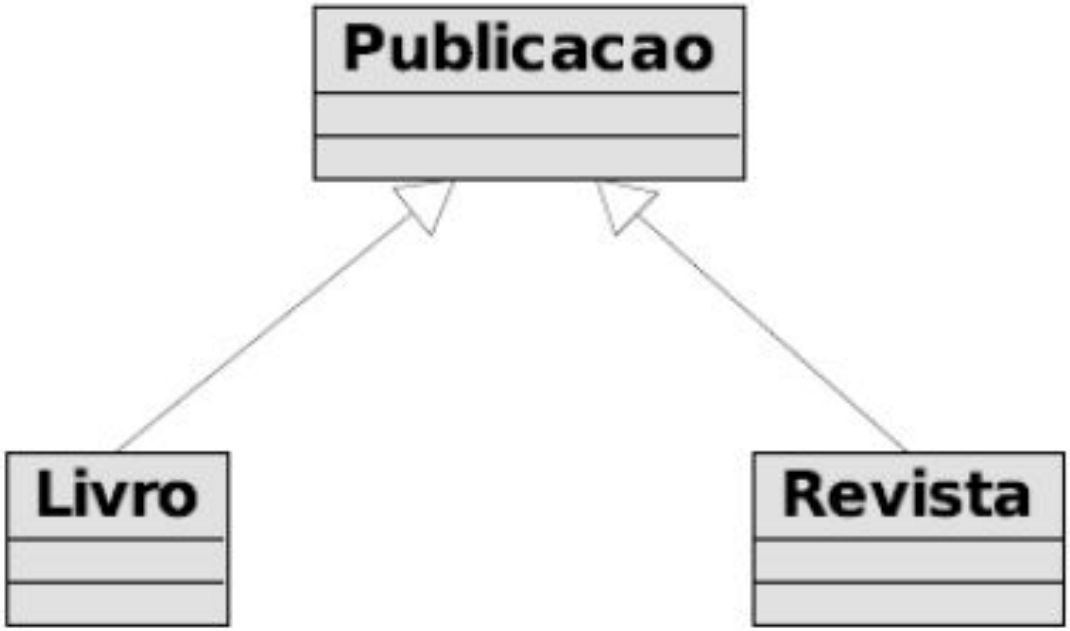
- Implemente as classes de um sistema de uma editora de livros e revistas.
  - Atributos:
    - Livro: nome, data de publicação, editora, isbn.
    - Revista: nome, data de publicação, editora, periodicidade.

Exemplo

```
01. class Publicacao {
02.     private String nome, dataPublicacao, editora;
03. }
04.
05. class Livro extends Publicacao {
06.     private String ISBN;
07. }
08.
09. class Revista extends Publicacao {
10.     private String periodicidade;
11. }
```

Exemplo

- Diagrama



Exemplo

- **Superclasse**: Publicacao.java

- **Subclasses**: Livro.java e Revista.java

Voltemos para o Problema dos Alunos!!

O QUE A HERANÇA VAI REPRESENTAR DE FATO??

- Podemos aproveitar métodos já implementados.
  - Exceto o construtor (chamado pelo método super( ))
  - Mas eu posso chamar o construtor da superclasse na subclasse.

- O construtor de uma subclasse tem, **obrigatoriamente**, que chamar um construtor de sua superclasse em seu corpo.

Como regra geral temos:

- As subclasses herdam variáveis e métodos cuja visibilidade seja **pública** ou **protegida**.
- As subclasses herdam da superclasse as variáveis e os métodos **sem** especificação de visibilidade desde que a classe esteja no mesmo pacote.
- As subclasses NÃO herdam membros da superclasse se a subclasse declarar uma variável ou método de mesmo nome.
  - Significa sobrescrever variável ou sobrescrever o método.

Entretanto...

- A subclasse não está limitada as variáveis e aos métodos da superclasse.
  - Para caracterizá-la melhor novas variáveis ou métodos podem ser adicionadas à subclasse, assim como, podem ser redefinidos (sobrescritos).

Curiosidade!!

- Uma característica da linguagem Java é que qualquer objeto que não herde explicitamente de outra classe, automaticamente será subclasse direta de **java.lang.Object**, que é a superclasse no nível mais alto da linguagem.

O que você lembra com a palavra herança??

- ~~Dinheiro~~

- Receber algo 
  - Métodos, atributos.

- Relacionamento entre as partes 
  - Especialização/Generização

- Posse,Legado, domínio 
  - Métodos, atributos

- Exemplo
  - class Veículo {...}
  - class Carro extends Veículo{...}
  - class Bicicleta extends Veículo{...}

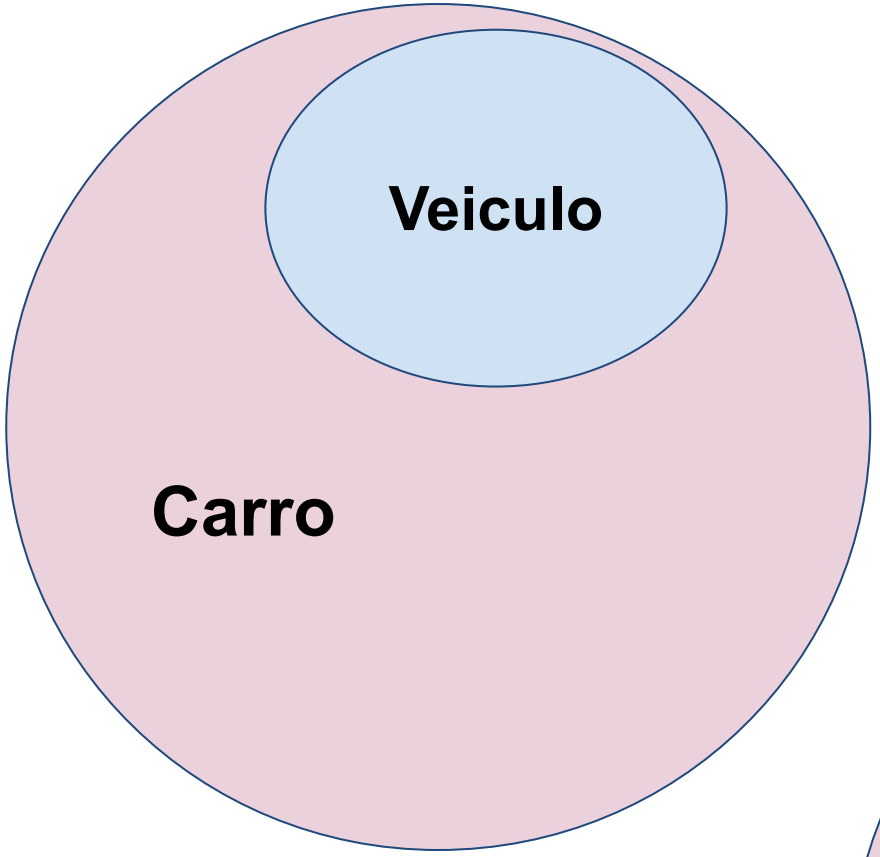
- É legal:
  - Veículo v1 = new Veículo();
  - Veículo v2 = new Car();
  - Veículo v3 = new Bicicleta();

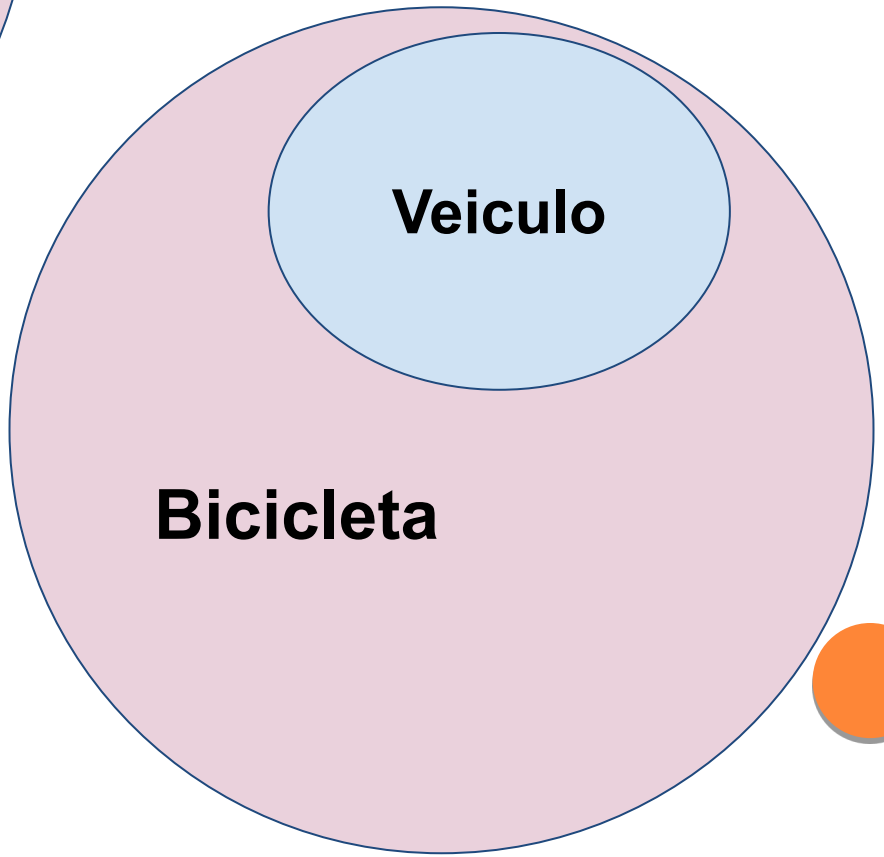
- **Não** faz sentido:
  - Carro c1 = new Veículo();
  - Carro c2 = new Bicicleta();

- Exemplo
  - class Veículo {...}
  - class Carro extends Veículo{...}
  - class Bicicleta extends Veículo{...}

- É legal:
  - Veículo v1 = new Veículo();
  - Veículo v2 = new Car();
  - Veículo v3 = new Bicicleta();

- **Não** faz sentido:
  - ~~Carro c1 = new Veículo();~~
  - ~~Carro c2 = new Bicicleta();~~


Veiculo
Carro


Veiculo
Bicicleta

HERANÇA
x
COMPOSIÇÃO

Composição

- Conceito:
  - É a técnica de construir um tipo não pela derivação partindo de outra classe, mas pela junção de vários outros objetos de menor complexidade que fornecem ao objeto composto determinada funcionalidade quando em conjunto.
  - Não há nenhuma palavra-chave ou recurso especial para utilizar composição em Java.
  - Instanciar objetos de outras classes em uma terceira.

Composição

- Exemplo de composição:

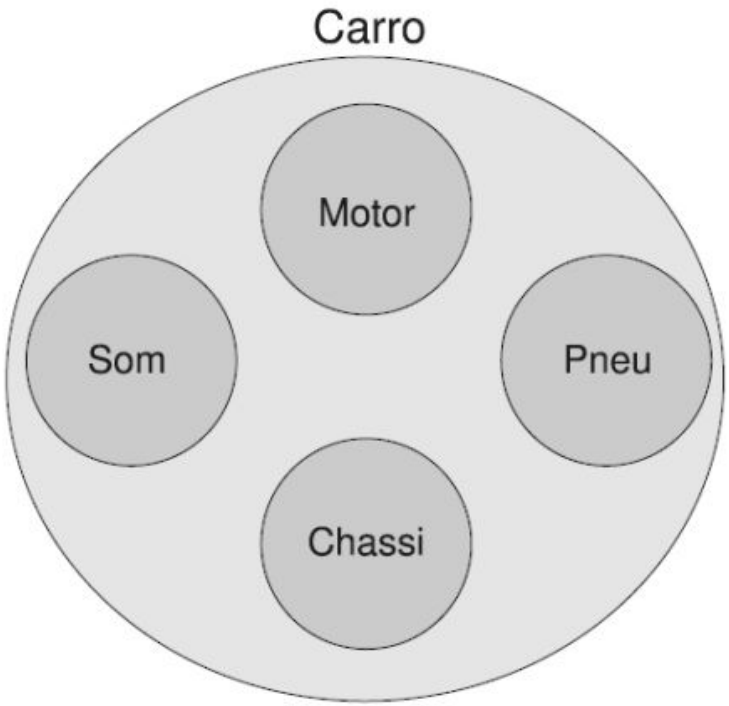


Figura 4.1: Composição entre diferentes níveis

Carro —*é composto por*→ {Motor, Som, Pneu, Chassi}

Composição

Programa 4.1 Composição entre classes

```
01. // Composição entre classes
02.
03. class Carro {
04.     private String modelo, fabricante;
05.     private Chassi chassi;
06.     private Motor motor;
07.     private Pneu[] pneus;
08. }
09. class Chassi {
10.     private String numero, dataFabricacao;
11. }
12.
13. class Motor {
14.     private int potencia;
15.     private String numero;
16. }
17.
18. class Pneu {
19.     private int raio;
20.     private String fabricante;
21. }
22. class Som {
23.     private int potencia;
24.     private String marca;
25. }
```

Exercício

- Desenhe um diagrama de classes UML e implemente a seguinte situação:
  - Funcionário tem nome, matrícula, sexo e data de nascimento e recebe salário.
  - Professor é um Funcionário que tem número de horas-aula e leciona.
  - Contador é um Funcionário que tem inscrição no Conselho de Contabilidade (CC) e executa a contabilidade.
  - Coordenador é um Professor que tem cargo e é responsável por criar turmas.

- Não se esqueça da classe de teste.

Dúvidas??

alanamm.prof@gmail.com