

Estruturas de Dados: Além dos Vetores

Agrupando informações com Classes e implementando Pilhas e Filas.

O Problema: Dados Relacionados, mas Separados

Até agora, para guardar dados de um aluno, precisaríamos de variáveis separadas:

```
String nome; int matricula;  
double nota;
```

E para vários alunos? Vetores paralelos!

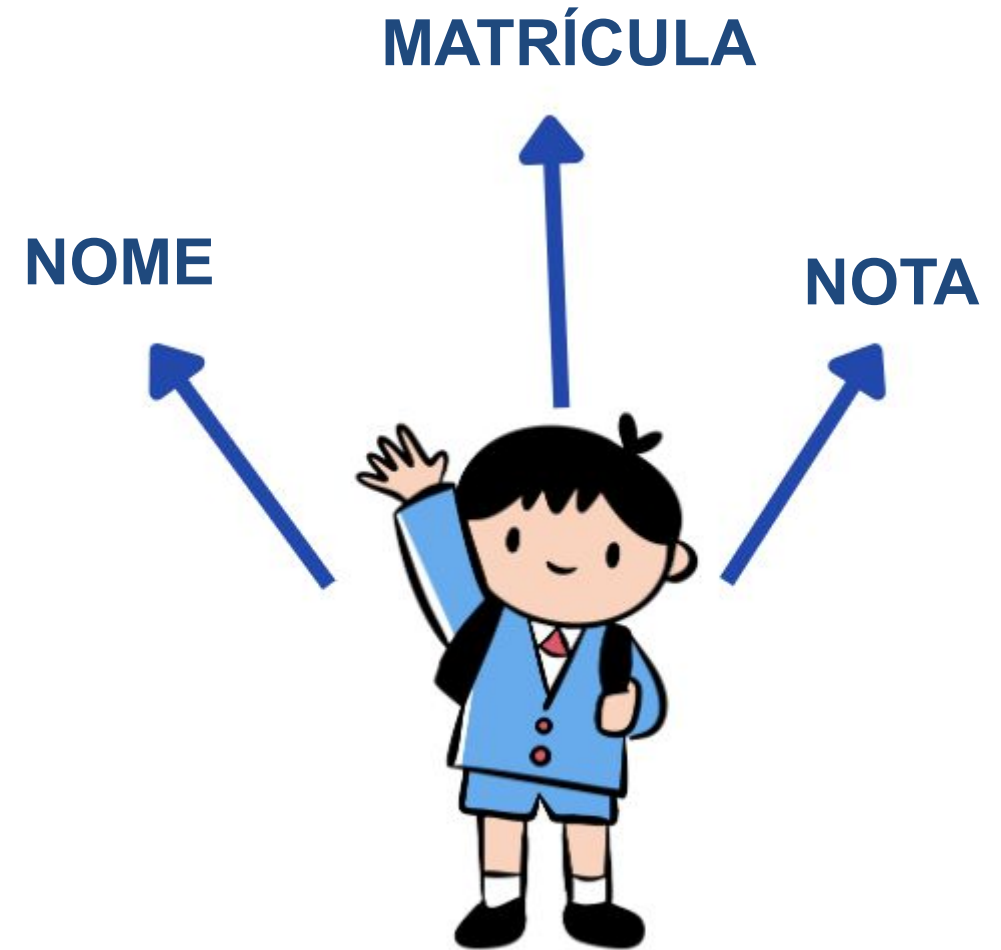
```
// Vetores Paralelos  
String[] nomes = new String[3];  
int[] matriculas = new int[3];  
double[] notas = new double[3];  
  
// Dados do primeiro aluno estão espalhados!  
nomes[0] = "Ana";  
matriculas[0] = 101;  
notas[0] = 8.5;
```

A Ideia Central da Orientação a Objetos

Modelando o Mundo Como Ele É

E se pudéssemos criar nossos próprios tipos de variáveis? Uma variável `aluno` que, dentro dela, já contivesse o nome, a matrícula e a nota?

A Orientação a Objetos (OO) é um paradigma de programação que nos permite agrupar dados (atributos) e os comportamentos (métodos) que operam nesses dados em uma única unidade chamada Objeto.



Os 4 Conceitos Fundamentais

Abstração: Focar no que é essencial, escondendo detalhes complexos. (Ex: Você dirige um carro sem precisar saber como o motor funciona).

Encapsulamento: Agrupar dados e comportamentos em uma "cápsula" segura (o objeto).

Herança: Criar novas classes baseadas em classes existentes, aproveitando características.

Polimorfismo: A habilidade de objetos diferentes responderem à mesma mensagem de formas diferentes.

Nosso foco principal será em **Abstração** e **Encapsulamento**!

Classe: A Planta Baixa, O Molde

A class é a nossa forma de definir a "planta baixa" de um objeto. Ela descreve quais atributos (dados) e métodos (comportamentos) um objeto daquele tipo terá.

Analogia:

- Classe: A receita de um bolo.
- Atributos: Os ingredientes na receita (farinha, açúcar, ovos).
- Métodos: Os passos da receita (misturar, assar).

Objeto: A Construção Real

Um **objeto** é a instância concreta criada a partir de uma classe. É o bolo pronto, feito a partir da receita.

Exemplo:

- A classe é Carro.
- Os objetos são: meuGol, seuFusca, oOnixDoVizinho. Todos foram feitos a partir da "planta" Carro, mas cada um é um objeto único e independente.

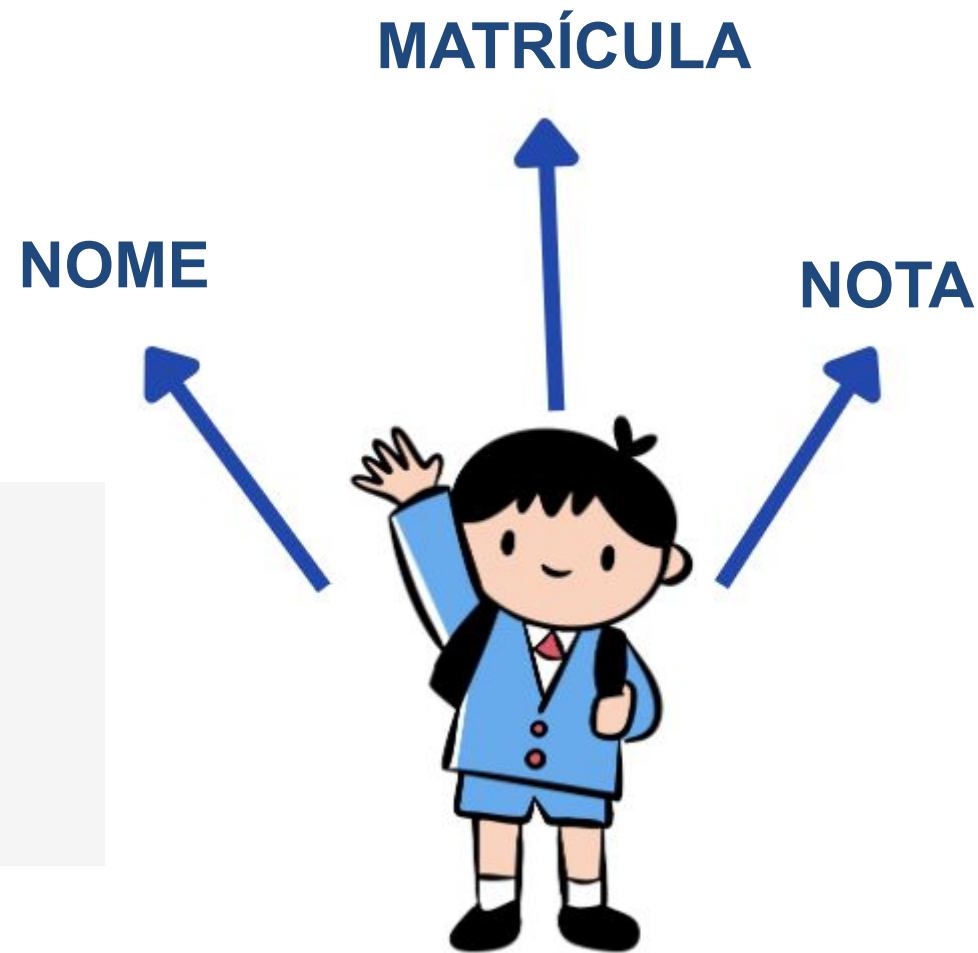
Encapsulamento Visível: Cada objeto **Carro** "carrega dentro de si" sua própria cor, sua própria velocidade, etc. Os dados não ficam espalhados



Nosso Primeiro Molde: A Classe **Aluno**

Objetivo: Vamos transformar aqueles vetores paralelos em uma estrutura organizada

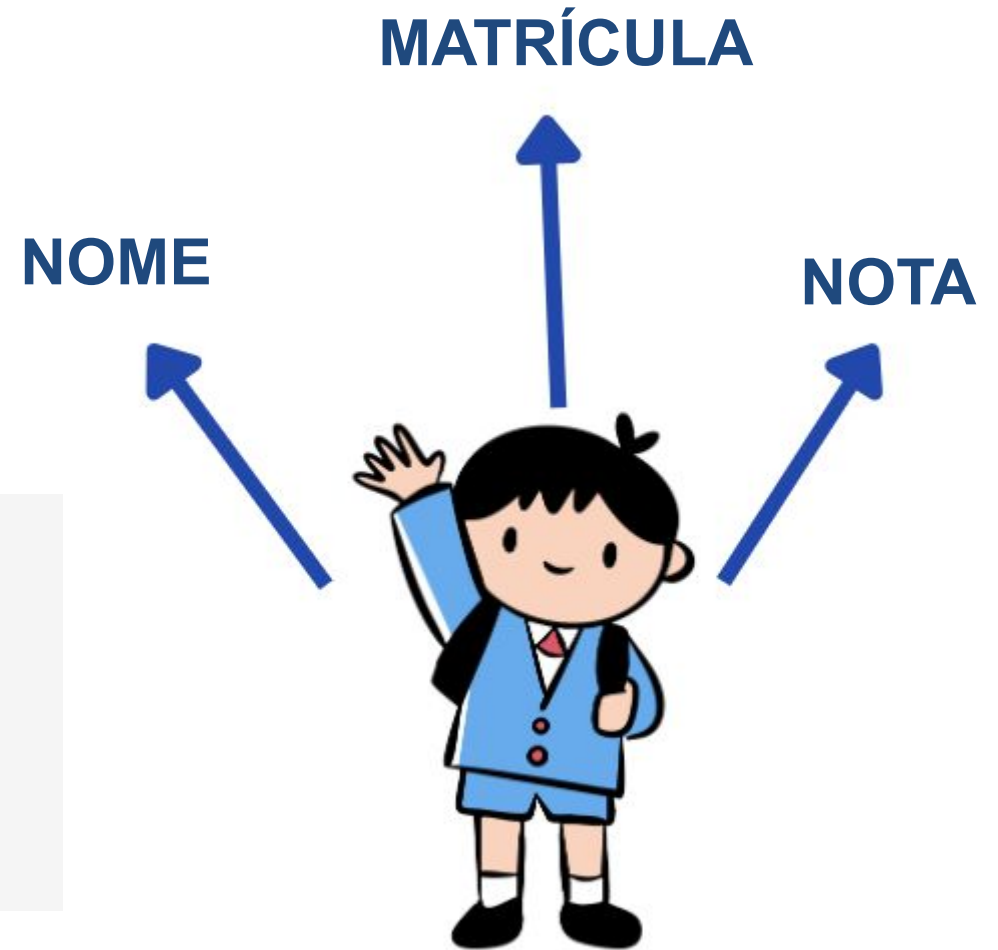
```
public class Aluno {  
    // Atributos (os campos do nosso "registro")  
    String nome;  
    int matricula;  
    double notaFinal;  
}
```



Instanciando e Usando Objetos **Aluno**

Usamos a palavra-chave **new** para construir um objeto a partir da classe. O operador **.** nos dá acesso aos seus atributos internos

```
public static void main(String[] args) {  
    Aluno aluno1 = new Aluno();  
    // Preenchendo os dados do registro  
    aluno1.nome = "Carlos Pereira";  
    aluno1.matricula = 201;  
    aluno1.notaFinal = 7.5;  
    // Acessando os dados  
    System.out.println("Matrícula do aluno: " + aluno1.matricula);  
}
```



Exercício

1. Crie uma nova classe chamada Produto.
2. Adicione a ela os seguintes atributos: String nome, int codigo, double preco.
3. No método main, crie uma instância (objeto) da classe Produto.
4. Peça para o usuário digitar os dados do produto via Scanner.
5. Ao final, exiba os dados do produto que você cadastrou.

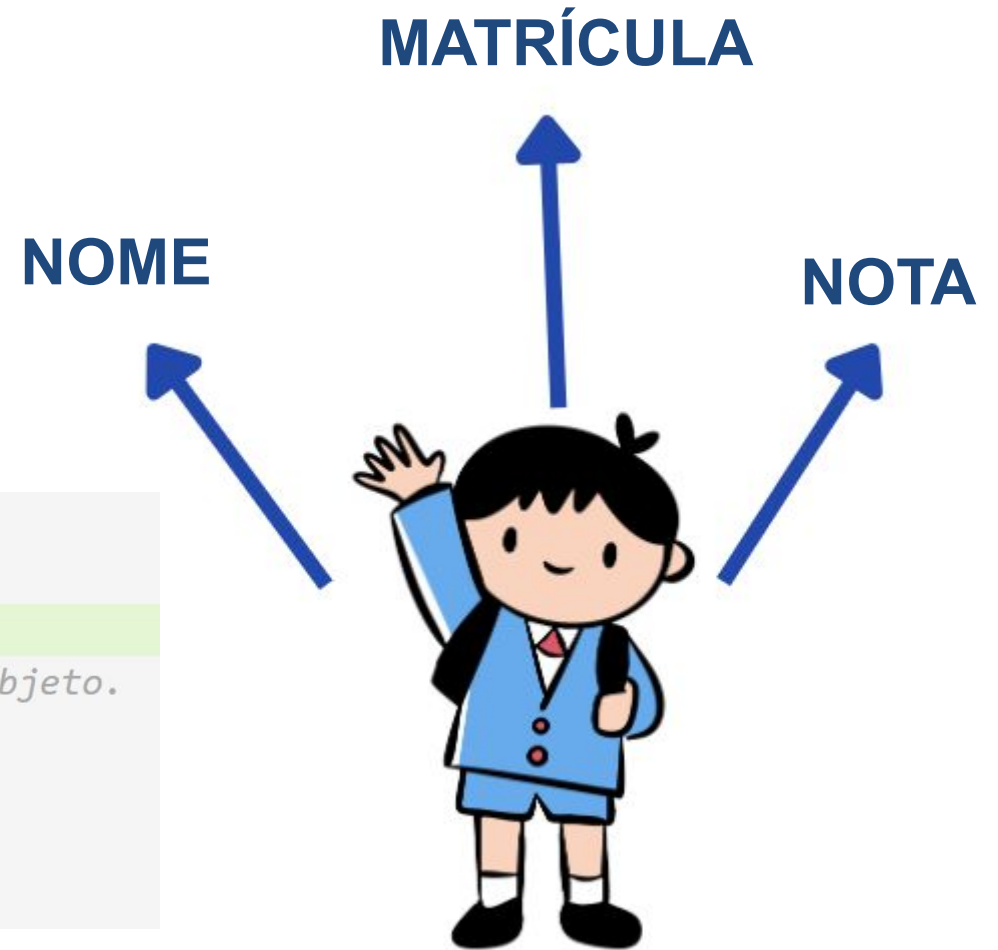


Evoluindo: Vetores de Objetos

Agora, em vez de 3 vetores paralelos, podemos ter UM único vetor do nosso novo tipo!

```
// Um vetor que pode guardar 30 registros do tipo Aluno
Aluno[] minhaTurma = new Aluno[30];

// Importante: O vetor está vazio! Precisamos criar cada objeto.
minhaTurma[0] = new Aluno();
minhaTurma[0].nome = "Juliana Lima";
minhaTurma[0].matricula = 202;
// ... e assim por diante
```



Exercício

1. Usando a classe Aluno que definimos, crie um programa principal.
2. Declare um vetor de Aluno com capacidade para 3 alunos.
3. Use um laço for para percorrer o vetor. A cada iteração:
 - a. Crie um novo objeto Aluno (`new Aluno()`) e o coloque na posição atual do vetor.
 - b. Peça ao usuário para digitar o nome, matrícula e nota daquele aluno.
4. Ao final do cadastro, use outro laço for para exibir os dados de todos os alunos da turma.



Um Objeto Não Tem Apenas Dados, Tem Ações!

Podemos adicionar funções (métodos) dentro da nossa classe para que o próprio objeto saiba como realizar ações com seus dados.

```
✓ public class Aluno {  
    // Atributos (os campos do nosso "registro")  
    String nome;  
    int matricula;  
    double notaFinal;  
    // Um método (comportamento) do Aluno  
    ✓ void verificarAprovacao() {  
    ✓     if (notaFinal >= 7.0) {  
    ✓         System.out.println(nome + " está APROVADO(A)!");  
    ✓     } else {  
        System.out.println(nome + " está em RECUPERAÇÃO.");  
    }  
    }  
}
```

Exercício

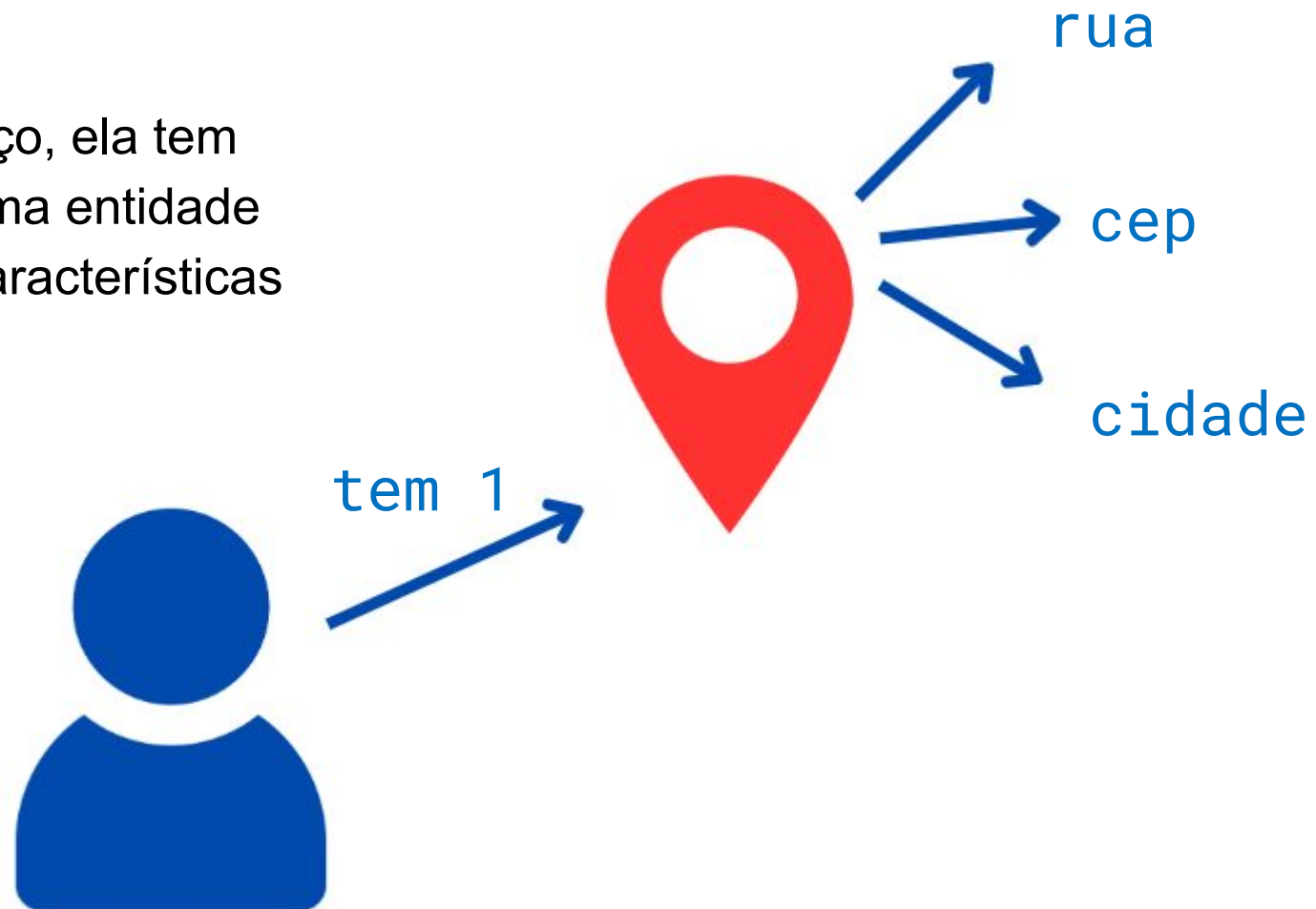
1. Adicione o método `verificarAprovacao()` à sua classe `Aluno`, como mostrado no slide anterior.
2. Modifique o programa do exercício 2. No laço final que exhibe os dados, em vez de usar `System.out.println` para cada dado, simplesmente chame o método do objeto.

Exemplo: `minhaTurma[i].verificarAprovacao();`



Objetos Dentro de Objetos: Composição

Uma Pessoa não é um endereço, ela tem um Endereco. O endereço é uma entidade separada com suas próprias características (rua, CEP, cidade).



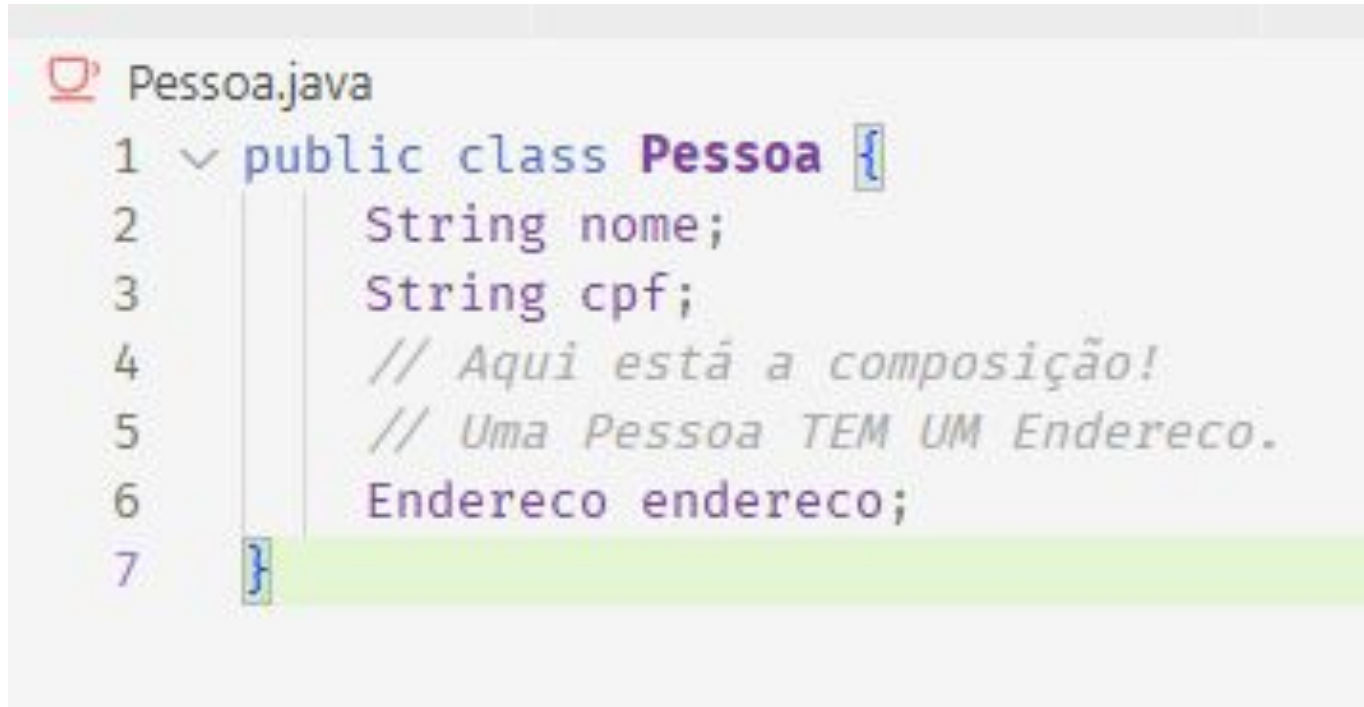
Criando a Parte: Endereco.java

Primeiro, criamos a classe mais "interna", que será usada por outra.

```
Endereço.java
1  public class Endereco {
2      String rua;
3      String cep;
4      String cidade;
5      String estado;
6      int numero;
7  }
```

Usando a Parte no Todo: Pessoa.java

Agora, a classe Pessoa pode ter um atributo do tipo Endereco.



```
Pessoa.java
1  public class Pessoa {
2      String nome;
3      String cpf;
4      // Aqui está a composição!
5      // Uma Pessoa TEM UM Endereco.
6      Endereco endereco;
7  }
```


Construindo o Objeto Composto

Para criar uma Pessoa, primeiro precisamos criar o objeto Endereco que fará parte dela.

```
public class Programa {  
    public static void main(String[] args) {  
        // 1. Cria o objeto Endereco  
        Endereco endDoJoao = new Endereco();  
        endDoJoao.rua = "Rua das Flores";  
        endDoJoao.numero = 123;  
        endDoJoao.cep = "25680-000";  
        endDoJoao.cidade = "Petrópolis";  
  
        // 2. Cria o objeto Pessoa  
        Pessoa joao = new Pessoa();  
        joao.nome = "João Carlos";  
        joao.cpf = "111.222.333-44";  
  
        // 3. Conecta os dois objetos  
        joao.endereco = endDoJoao;  
  
        // Acessando dados através da composição  
        System.out.println("O " + joao.nome + " mora na cidade de " + joao.endereco.cidade);  
    }  
}
```

Exercício

1. Crie uma classe Autor com os atributos nome e nacionalidade.
2. Crie uma classe Livro com os atributos titulo, anoPublicacao e um atributo do tipo Autor.
3. No método main, crie um objeto Autor, preencha seus dados.
4. Depois, crie um objeto Livro, preencha seus dados e associe o autor criado a ele.
5. Exiba no console uma frase como: "O livro 'Dom Casmurro' foi escrito por Machado de Assis."



O que é Aquele Parênteses? ()

- Conceito: A parte Produto() é uma chamada a um método especial chamado Construtor.
- Definição: Um construtor é um "bloco de código" que é executado exatamente no momento em que um objeto é criado com a palavra-chave new.
- O Construtor Padrão: Se nós não escrevemos nenhum construtor, o Java nos dá um invisível e sem argumentos, que não faz nada além de criar o objeto. É o que temos usado até agora.

Assumindo o Controle da Construção

Podemos escrever nosso próprio construtor para obrigar que dados essenciais sejam fornecidos no momento da criação.

Regras de um Construtor: Tem o mesmo nome da classe.

Não tem tipo de retorno (nem mesmo void).

```
public class Produto {  
    String nome;  
    int codigo;  
    double preco;  
  
    // Nosso construtor personalizado!  
    public Produto(String nome, int codigo, double preco) {  
        // A palavra "this" se refere ao atributo do objeto  
        // Distingue o atributo "nome" do parâmetro "nome"  
        this.nome = nome;  
        this.codigo = codigo;  
        this.preco = preco;  
    }  
}
```

Criação de Objetos: Segura e Eficiente

Agora que a classe Produto tem um construtor que exige 3 argumentos, a única forma de criar um objeto é fornecendo esses dados.

```
public static void main(String[] args) {  
    // O jeito antigo não funciona mais!  
    // Produto p1 = new Produto(); // <-- ISSO AGORA DÁ ERRO!  
  
    // O jeito novo, correto e seguro:  
    Produto p1 = new Produto("Teclado", 55, 150.00);  
    Produto p2 = new Produto("Mouse", 56, 80.00);  
  
    System.out.println("Produto: " + p1.nome); // O objeto já nasce pronto!  
    System.out.println("Produto: " + p2.nome); // O objeto já nasce pronto!  
}
```

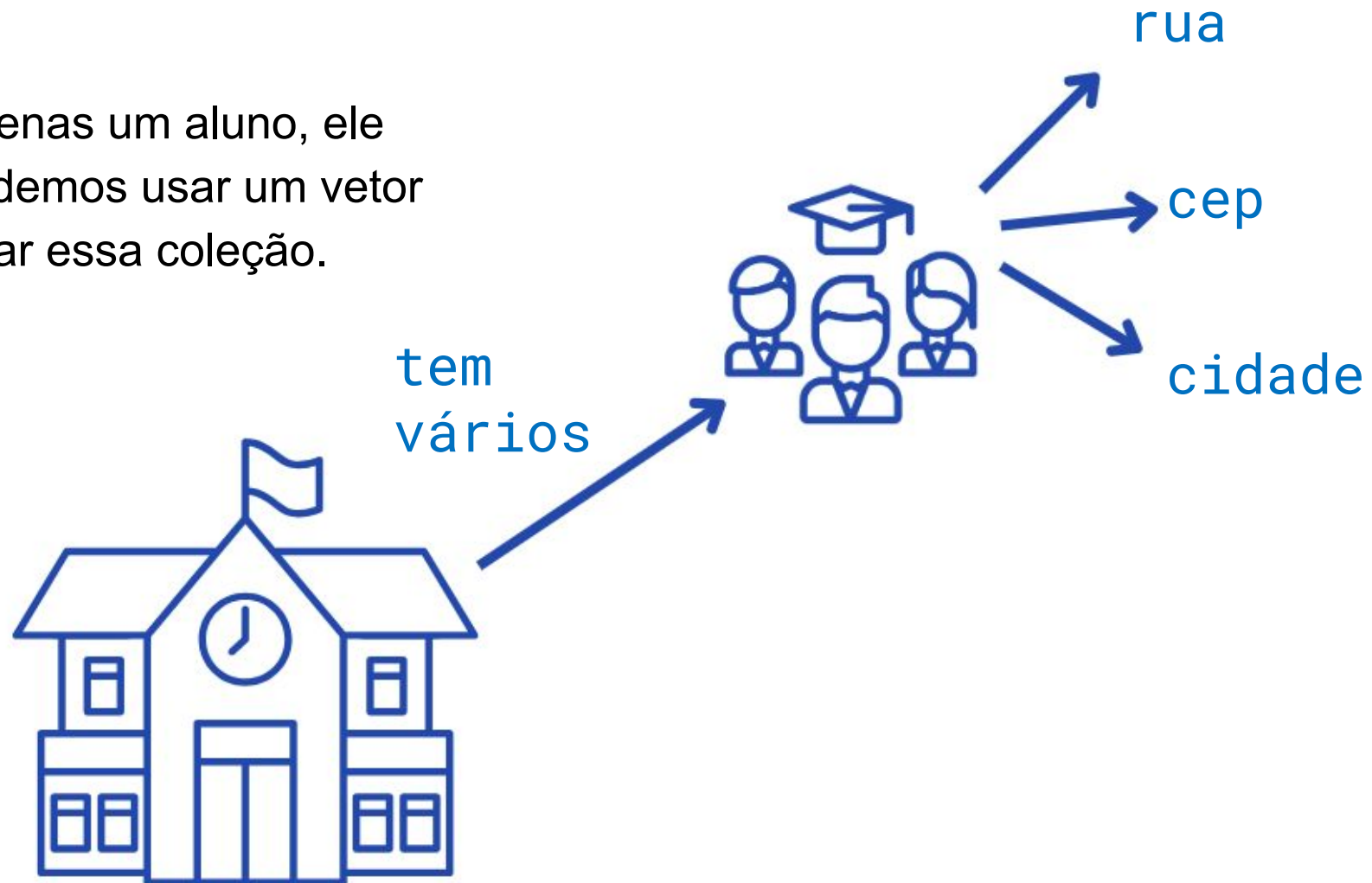
Exercício

1. Pegue a classe Autor que você criou (nome, nacionalidade).
2. Adicione a ela um construtor que receba o nome e a nacionalidade como parâmetros.
3. Modifique o programa principal para criar dois objetos Autor usando o novo construtor em uma única linha para cada.
4. Desafio: Adicione um construtor à classe Livro que receba o título, anoPublicacao e um objeto Autor já criado



Objetos Dentro de Objetos: Composição

Um Curso não tem apenas um aluno, ele tem vários Alunos. Podemos usar um vetor (array) para representar essa coleção.



Modelando o Curso

A classe Curso terá um vetor de Aluno.

```
Curso > J Curso.java > ...
1  package Curso;
2  public class Curso {
3      String nomeDoCurso;
4      String sigla;
5      Aluno[] alunosMatriculados;
6      int vagasOcupadas = 0; // Para controlar o preenchimento do vetor
7
8      // Construtor para inicializar o curso com um número de vagas
9      public Curso(String nome, int totalVagas) {
10         this.nomeDoCurso = nome;
11         this.alunosMatriculados = new Aluno[totalVagas];
12     }
13
14     // Método para matricular um aluno
15     public void matricular(Aluno novoAluno) {
16         if (vagasOcupadas < alunosMatriculados.length) {
17             alunosMatriculados[vagasOcupadas] = novoAluno;
18             vagasOcupadas++;
19             System.out.println("Matrícula de " + novoAluno.nome + " realizada!");
20         } else {
21             System.out.println("Não há vagas disponíveis!");
22         }
23     }
24 }
25
```


Exercício

1. Crie uma classe Jogador com os atributos nome e posicao (ex: "Atacante").
2. Crie uma classe TimeDeFutebol com atributos nomeDoTime e um vetor de Jogador (com capacidade para 11 jogadores).
3. Crie um método contratarJogador(Jogador novoJogador) na classe TimeDeFutebol.
4. No main, crie um objeto TimeDeFutebol. Depois, crie 2 ou 3 objetos Jogador e use o método contratarJogador para adicioná-los ao time.
5. Crie um método exibirEscalacao() no time que mostre o nome e a posição de todos os jogadores contratados.