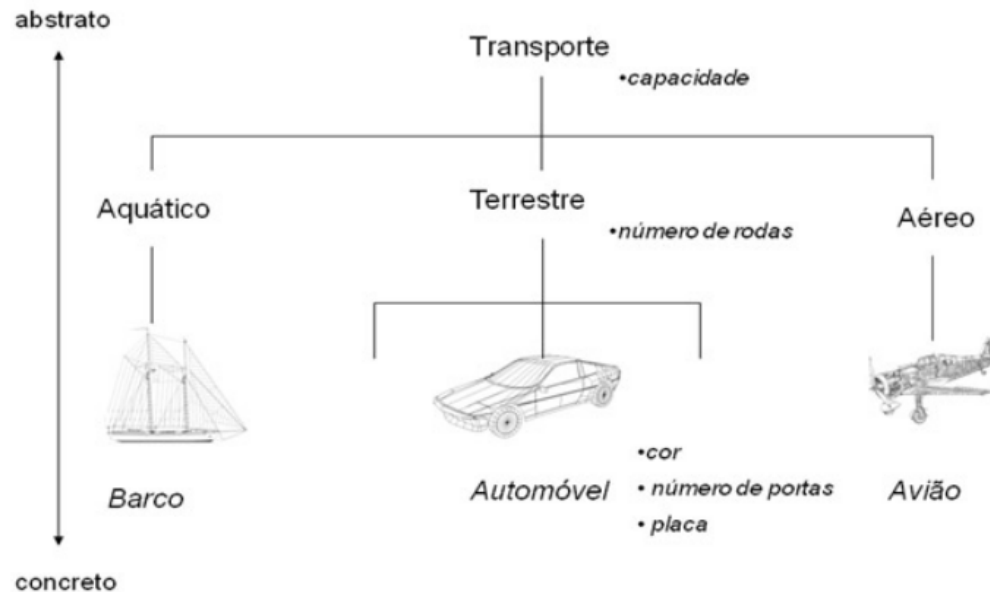


# Programação Orientada a Objetos – Aula 04

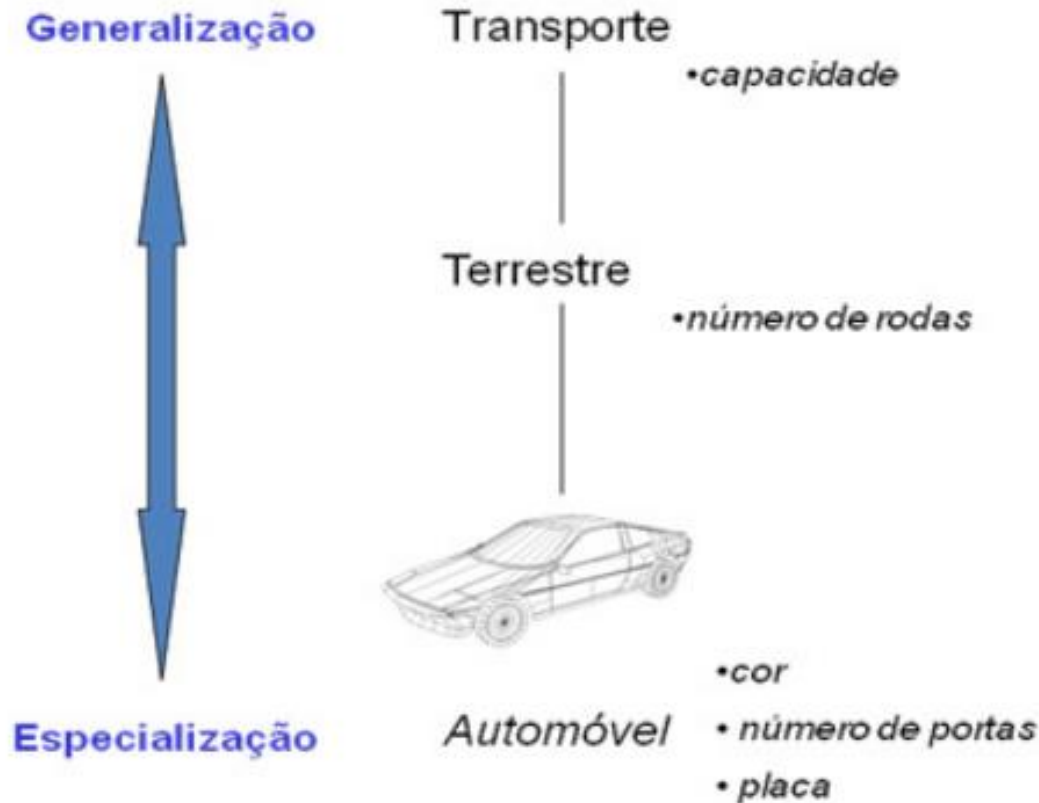
Prof. Dr. Eduardo Takeo Ueda  
*[eduardo.tueda@sp.senac.br](mailto:eduardo.tueda@sp.senac.br)*

# Herança (ou Hierarquia de classes)

- **Herança** é o mecanismo que permite uma classe herdar todos os atributos e métodos de outra classe
- Permite definir a implementação de uma nova classe na definição de uma classe previamente implementada



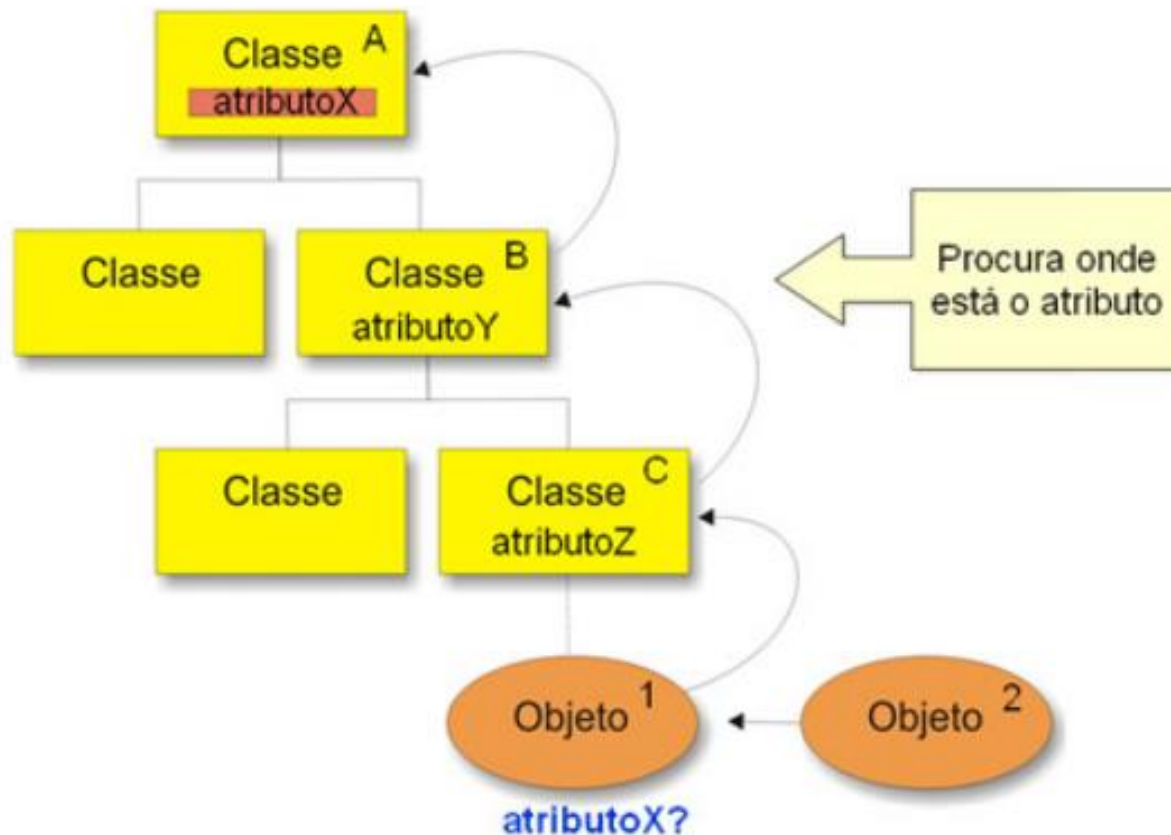
# Generalização e Especialização



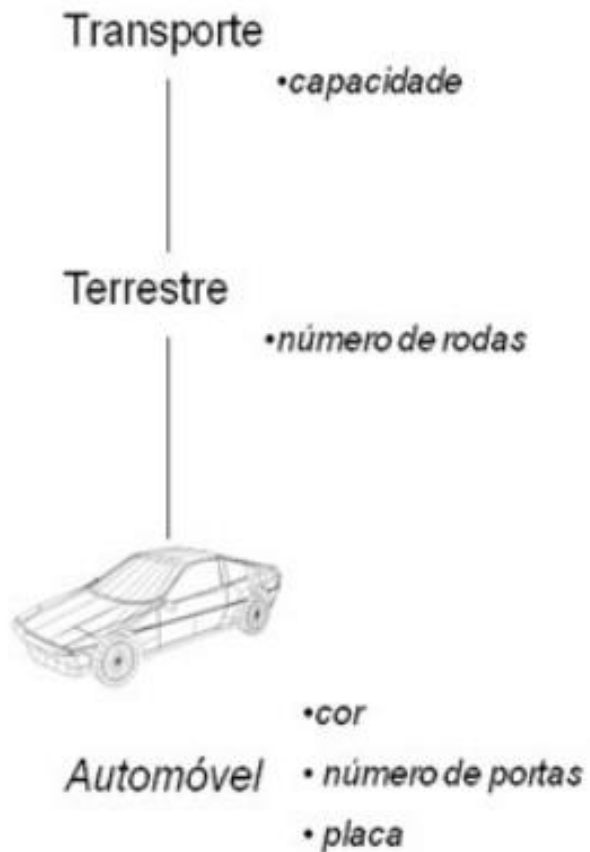
# Herança múltipla e simples

- **Herança múltipla:** é a capacidade de uma classe possuir mais de uma superclasse (classe mãe/pai) e herdar as variáveis e métodos combinados de todas elas
- **Herança simples:** cada classe pode ter apenas uma superclasse, embora uma superclasse possa ter várias subclasses (classes filhas ou derivadas)
- Na linguagem Java, a herança é simples e na codificação usa-se a palavra reservada **extends** para declarar que uma classe é herdeira de outra
- Para simular a herança múltipla em Java, usa-se **Interfaces** (veremos este tópico em breve!)

# Funcionamento da herança para atributos



# Herança em implementações Java



```
Transporte.java  Terrestre.java  Automovel.java
1 package senac.poo.aula4;
2
3 public class Transporte {
4
5     private int capacidade;
6 }
7
```

```
Transporte.java  Terrestre.java  Automovel.java
1 package senac.poo.aula4;
2
3 public class Terrestre extends Transporte{
4
5     private int numRodas;
6 }
7
```

```
Transporte.java  Terrestre.java  Automovel.java
1 package senac.poo.aula4;
2
3 public class Automovel extends Terrestre{
4
5     private String cor;
6     private int numPortas;
7     private String placa;
8 }
9
```

# Atenção

- A classe **Object**: todas as classes em Java descendem de uma classe, chamada **Object**, mesmo que a declaração `extends Object` seja omitida. Assim, a classe **Object** é a classe raiz da hierarquia de todas as classes Java, sendo, portanto, ancestral de todas as classes da linguagem.
- Quando uma classe usa a relação de Herança, podemos dizer que essa classe possui um relacionamento chamado “**é um**” com a classe da qual ela herda. Tal relação também indica que uma classe é do mesmo tipo que outra. Assim, nos exemplos anteriores, podemos dizer que **Automovel** “**é um**” transporte **Terrestre**, assim como que **Terrestre** “**é um**” (tipo de) **Transporte**.

# Herança e o modificador protected

- Funciona como o *private* exceto que as classes filhas também terão acesso ao atributo ou método declarado como *protected*



```
Transporte.java  Terrestre.java  Automovel.java
1 package senac.poo.aula4;
2
3 public class Transporte {
4
5     protected int capacidade;
6 }
7
```

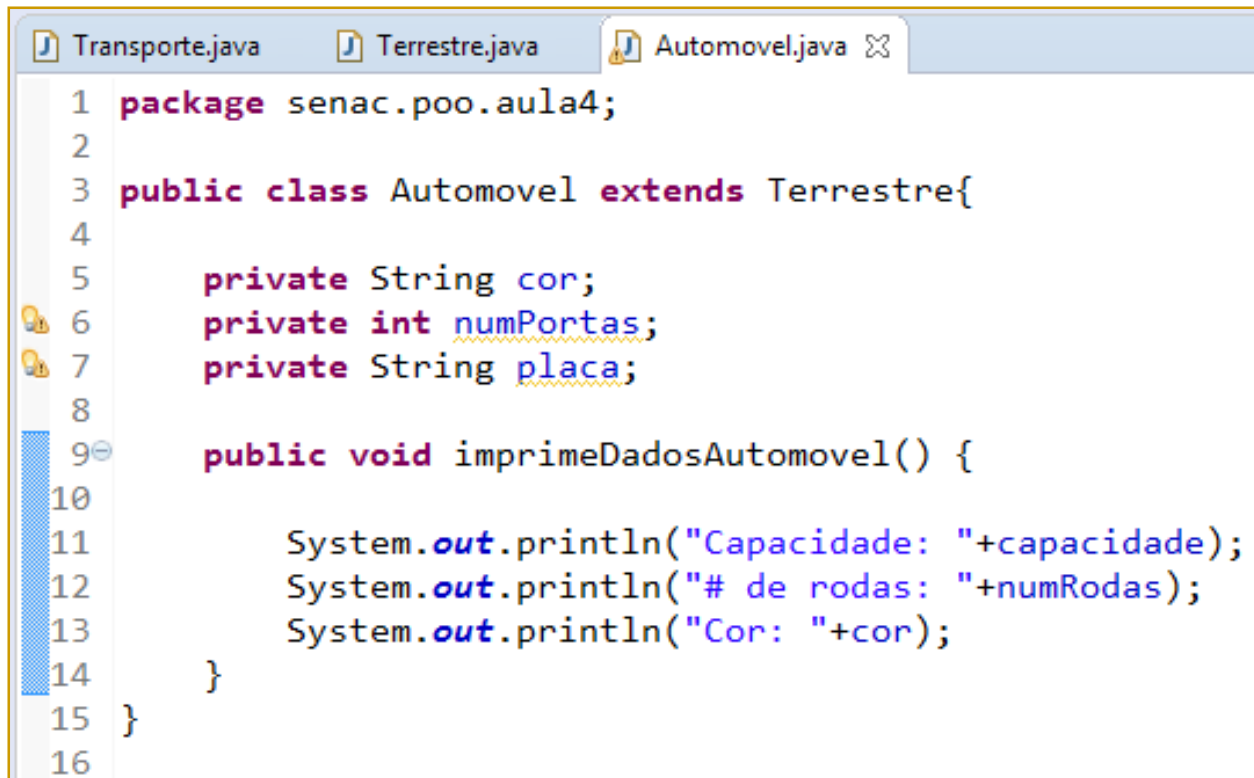
```
Transporte.java  Terrestre.java  Automovel.java
1 package senac.poo.aula4;
2
3 public class Terrestre extends Transporte{
4
5     protected int numRodas;
6 }
7
```

```
Transporte.java  Terrestre.java  Automovel.java
1 package senac.poo.aula4;
2
3 public class Automovel extends Terrestre{
4
5     private String cor;
6     private int numPortas;
7     private String placa;
8 }
9
```



# Herança e o modificador protected

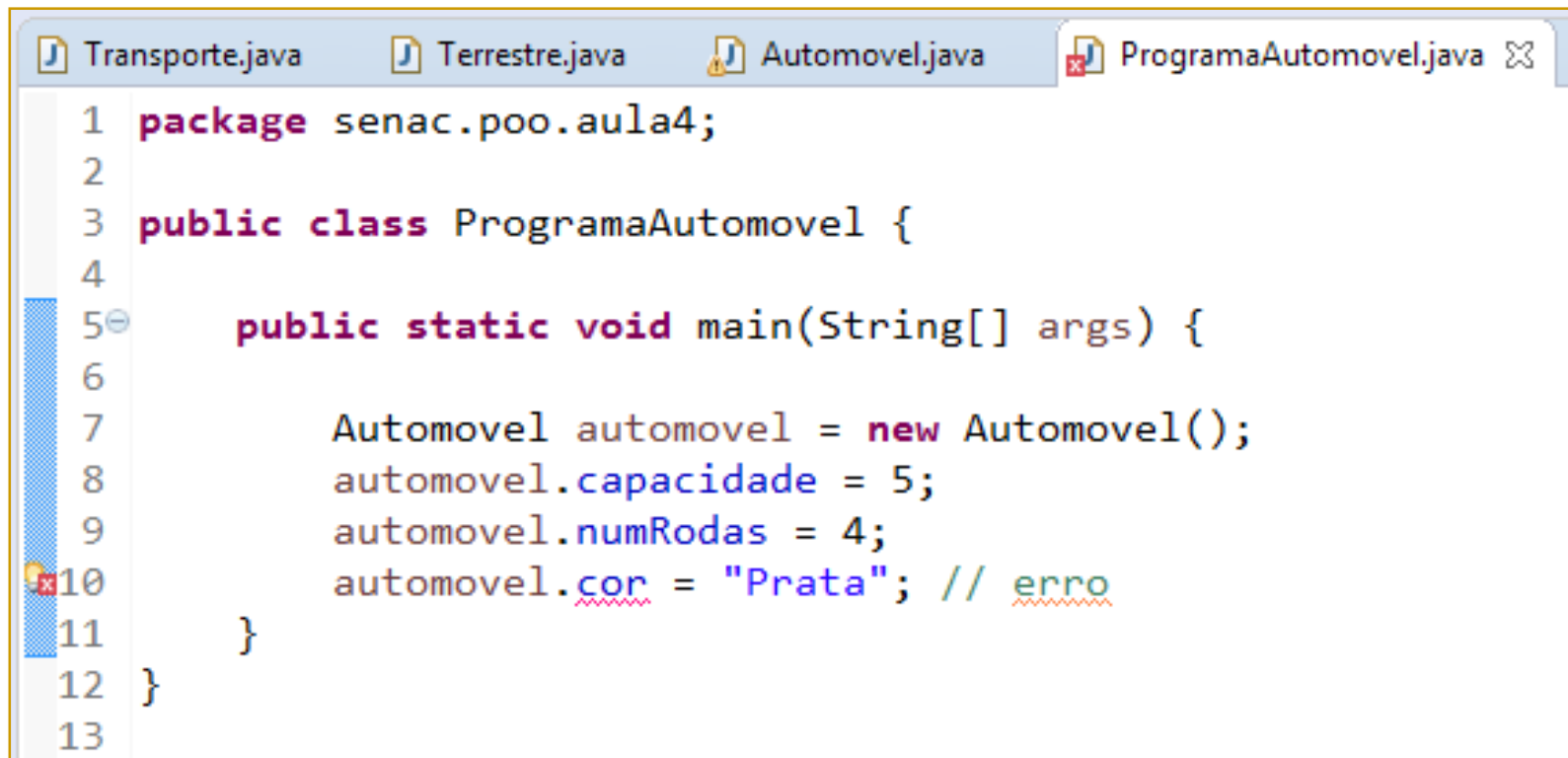
- O objeto Automovel está acessando diretamente os atributos de Terrestre e Transporte



```
1 package senac.poo.aula4;
2
3 public class Automovel extends Terrestre{
4
5     private String cor;
6     private int numPortas;
7     private String placa;
8
9     public void imprimeDadosAutomovel() {
10
11         System.out.println("Capacidade: "+capacidade);
12         System.out.println("# de rodas: "+numRodas);
13         System.out.println("Cor: "+cor);
14     }
15 }
16
```

# Herança e o modificador protected

- A classe Programa não pode acessar os atributos diretamente



The screenshot shows a Java IDE with four tabs: Transporte.java, Terrestre.java, Automovel.java, and ProgramaAutomovel.java. The active tab is ProgramaAutomovel.java, which contains the following code:

```
1 package senac.poo.aula4;
2
3 public class ProgramaAutomovel {
4
5     public static void main(String[] args) {
6
7         Automovel automovel = new Automovel();
8         automovel.capacidade = 5;
9         automovel.numRodas = 4;
10        automovel.cor = "Prata"; // erro
11    }
12 }
13
```

Line 10 has a red squiggly line under the attribute `cor` and a red error icon in the left margin, indicating a compilation error. The comment `// erro` is also present.

# Herança e o super

- A palavra **super** refere-se à classe ancestral imediata da classe, ou seja, a classe mãe ou super-classe
- **super** é usado nos construtores para chamada de construtores em cascata das classes mães

# Herança e o super

```
Transporte.java  Terrestre.java  Automovel.java
1 package senac.poo.aula4;
2
3 public class Transporte {
4
5     protected int capacidade;
6
7     public Transporte(int capacidade) {
8
9         this.capacidade = capacidade;
10    }
11 }
12
```

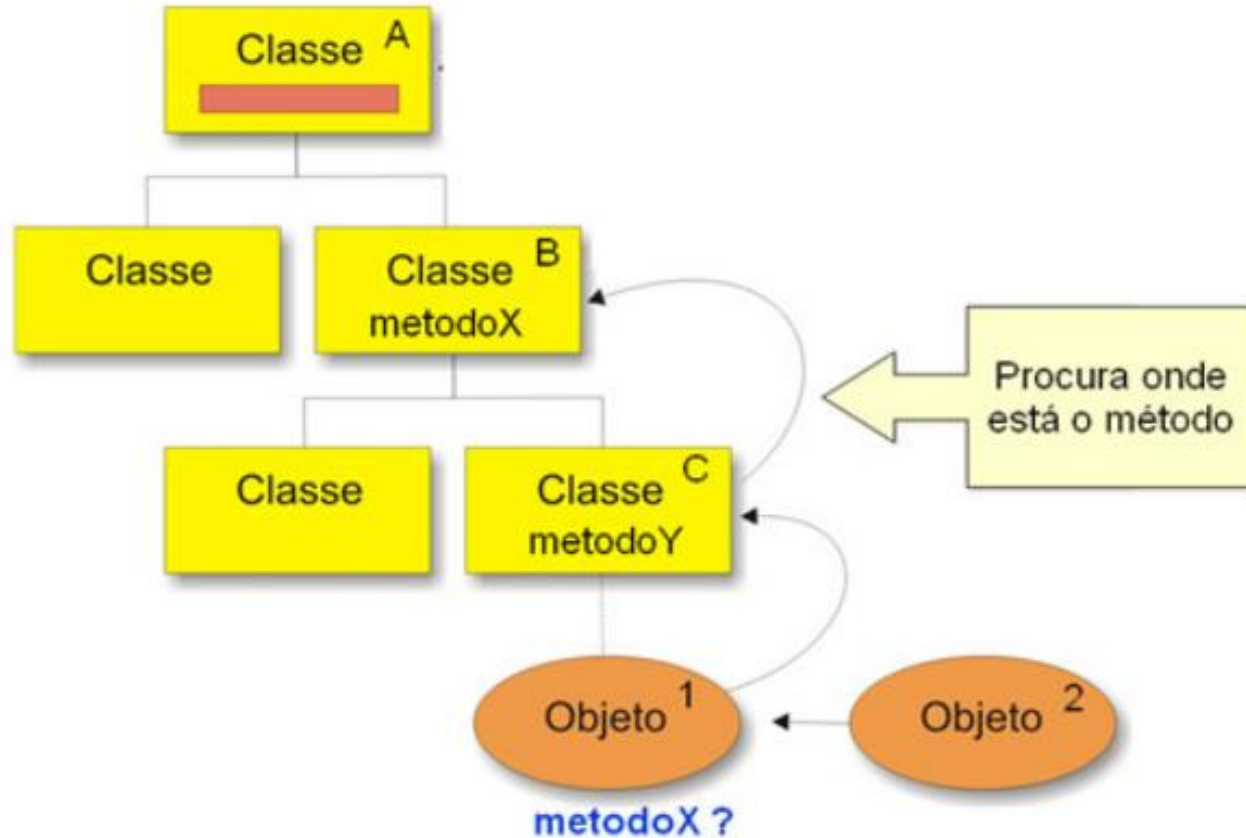
```
Transporte.java  Terrestre.java  Automovel.java
1 package senac.poo.aula4;
2
3 public class Terrestre extends Transporte{
4
5     protected int numRodas;
6
7     public Terrestre(int capacidade, int numRodas) {
8
9         super(capacidade);
10        this.numRodas = numRodas;
11    }
12 }
13
```

```
Transporte.java  Terrestre.java  Automovel.java
1 package senac.poo.aula4;
2
3 public class Automovel extends Terrestre{
4
5     private String cor;
6     private int numPortas;
7     private String placa;
8
9     public Automovel(int capacidd, int numRodas, String cor,
10                     int numPortas, String placa) {
11
12         super(capacidd, numRodas);
13         this.cor = cor;
14         this.numPortas = numPortas;
15         this.placa = placa;
16    }
17 }
18
```

# Atenção novamente

- Apenas comentários são permitidos antes da palavra **super** nos construtores. Assim, não é possível incluir nenhum comando antes de `super()` no código de métodos construtores de classes
- Da mesma maneira que se usa a palavra-chave **this** para acessar os atributos (ou métodos) da própria classe, pode-se usar **super**, para acessar os atributos (ou métodos) da classe mãe

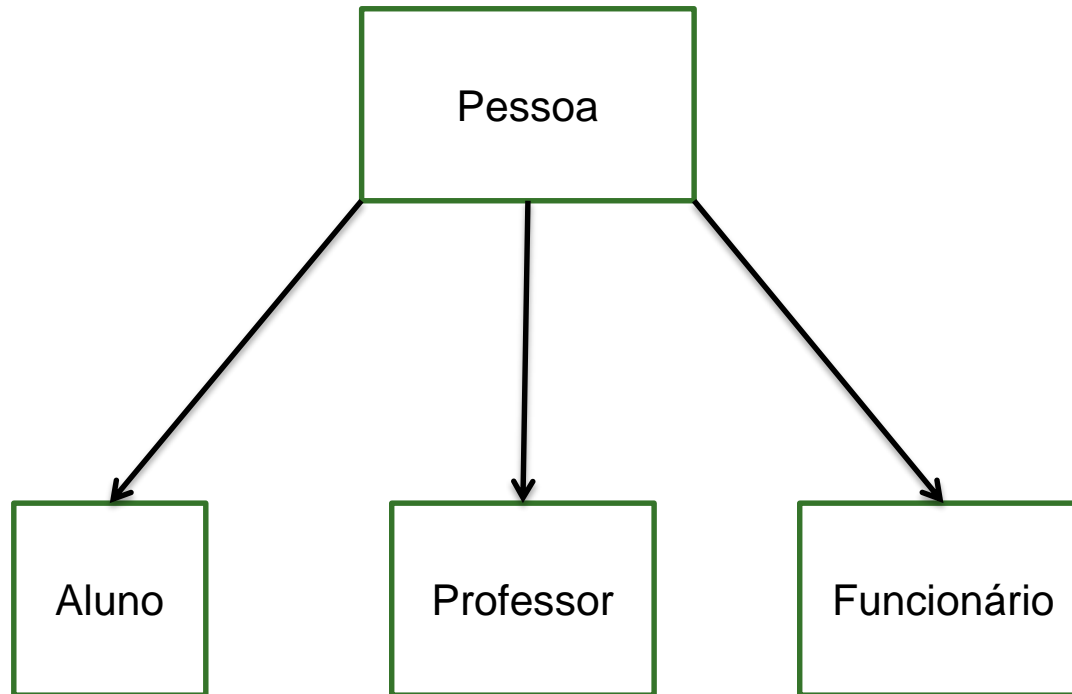
# Funcionamento da herança para métodos



# Níveis de acesso (ou Graus de visibilidade)

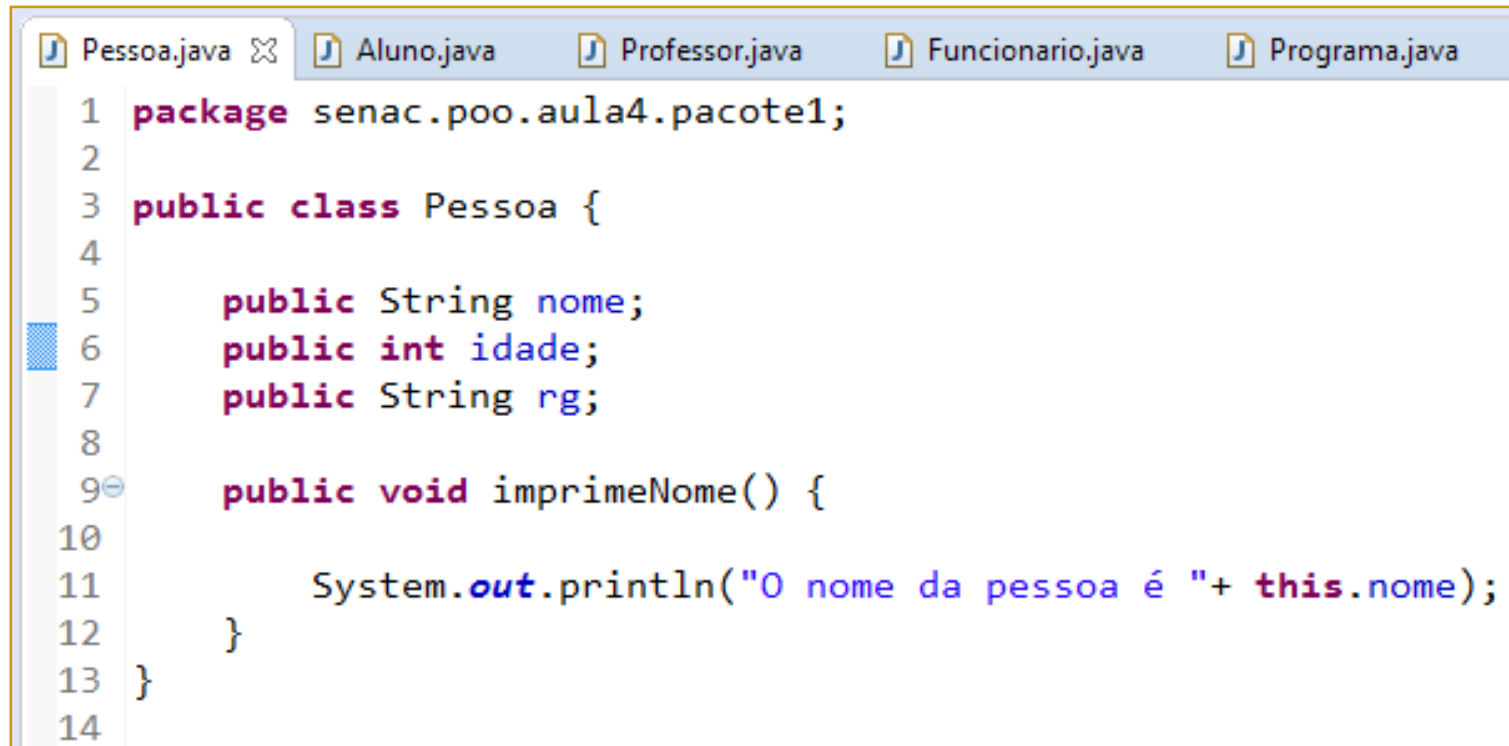
MODIFICADOR	CLASSE	MESMO PACOTE	PACOTE DIFERENTE (SUBCLASSE)	PACOTE DIFERENTE(GLOBAL)
Public				
Protected				
Default				
Private				

# Um pouco mais de herança



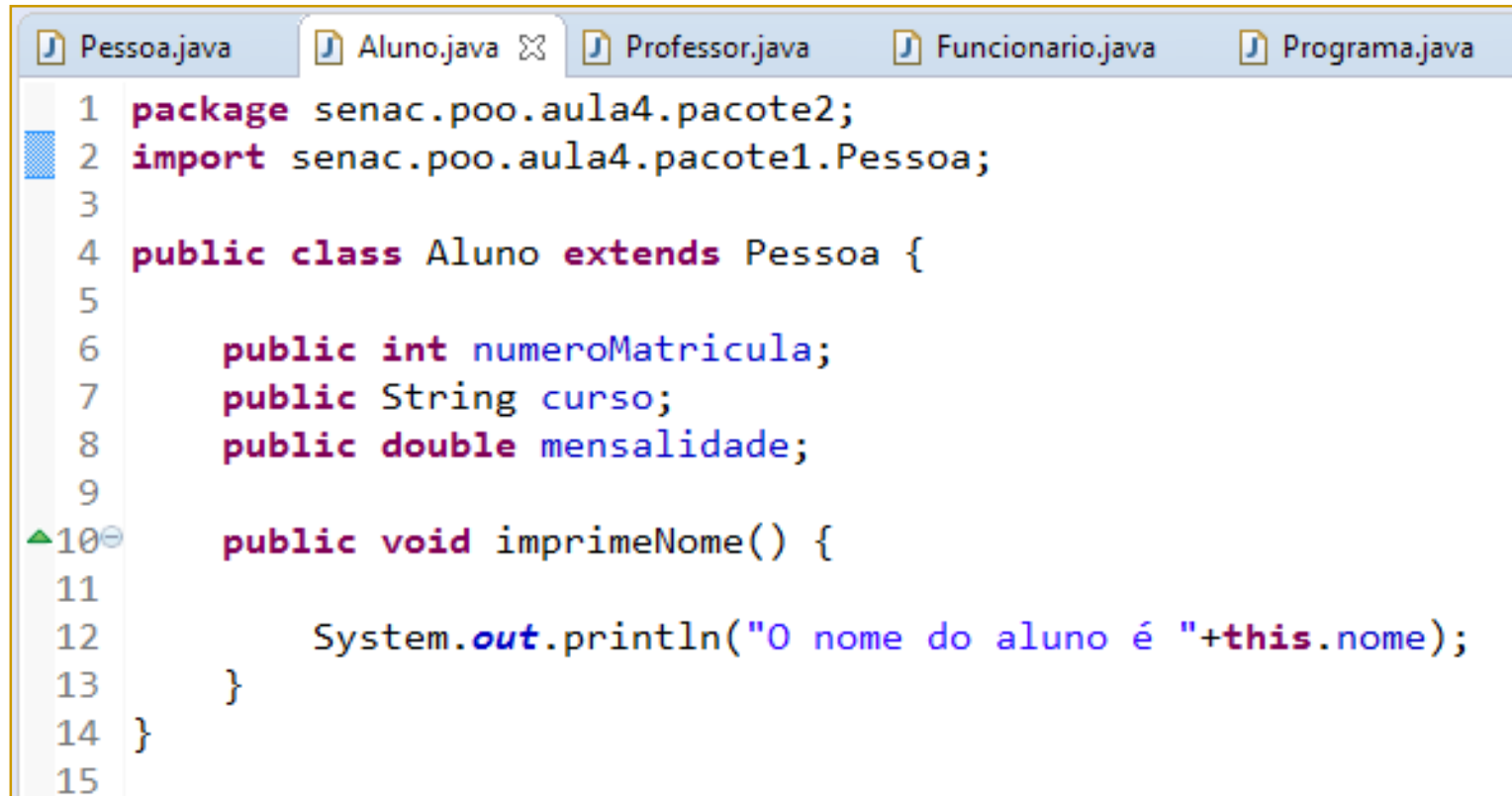


# Superclasse Pessoa



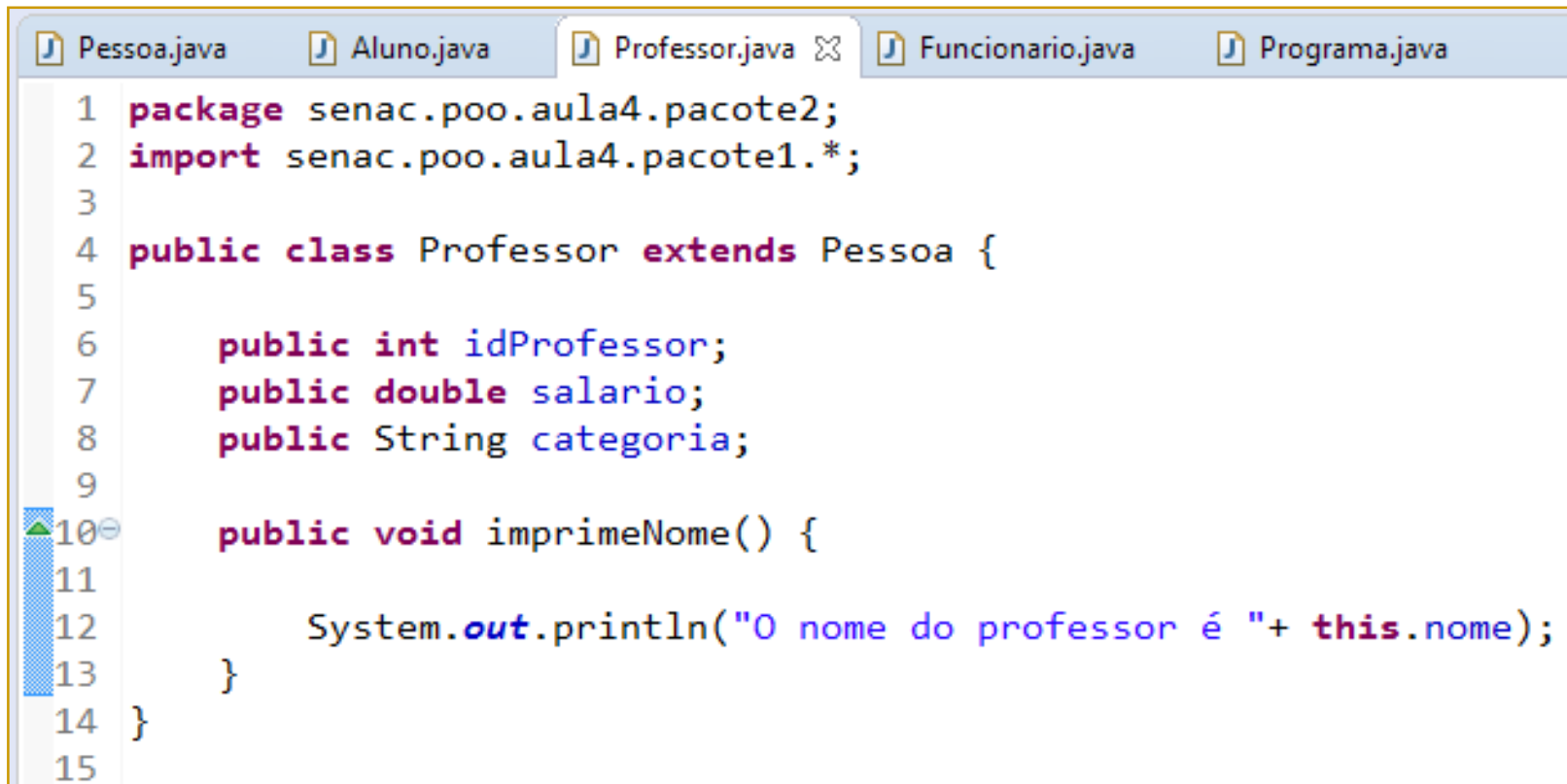
```
1 package senac.poo.aula4.pacote1;
2
3 public class Pessoa {
4
5     public String nome;
6     public int idade;
7     public String rg;
8
9     public void imprimeNome() {
10
11         System.out.println("O nome da pessoa é " + this.nome);
12     }
13 }
14
```

# Subclasse Aluno



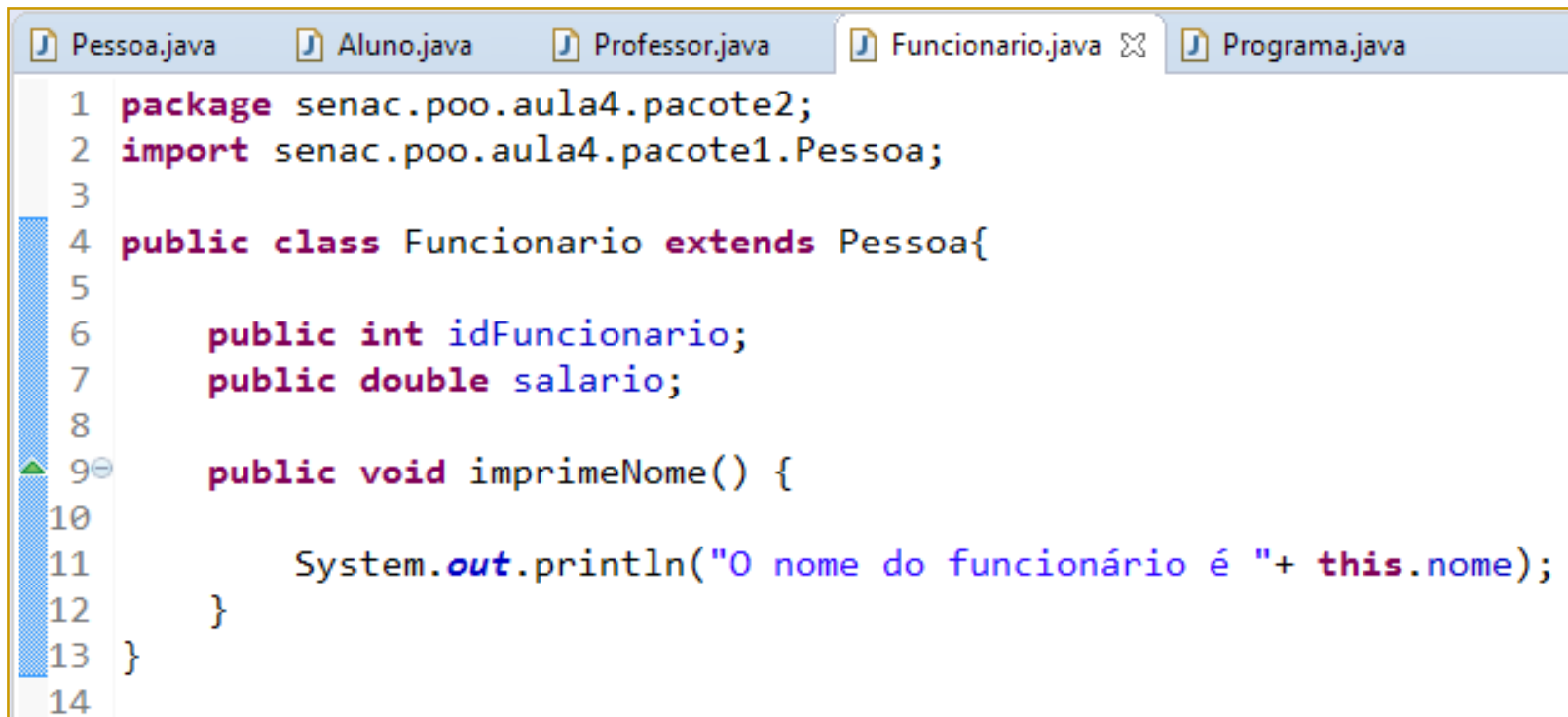
```
1 package senac.poo.aula4.pacote2;
2 import senac.poo.aula4.pacote1.Pessoa;
3
4 public class Aluno extends Pessoa {
5
6     public int numeroMatricula;
7     public String curso;
8     public double mensalidade;
9
10    public void imprimeNome() {
11
12        System.out.println("O nome do aluno é "+this.nome);
13    }
14 }
15
```

# Subclasse Professor



```
1 package senac.poo.aula4.pacote2;
2 import senac.poo.aula4.pacote1.*;
3
4 public class Professor extends Pessoa {
5
6     public int idProfessor;
7     public double salario;
8     public String categoria;
9
10    public void imprimeNome() {
11
12        System.out.println("O nome do professor é "+ this.nome);
13    }
14 }
15
```

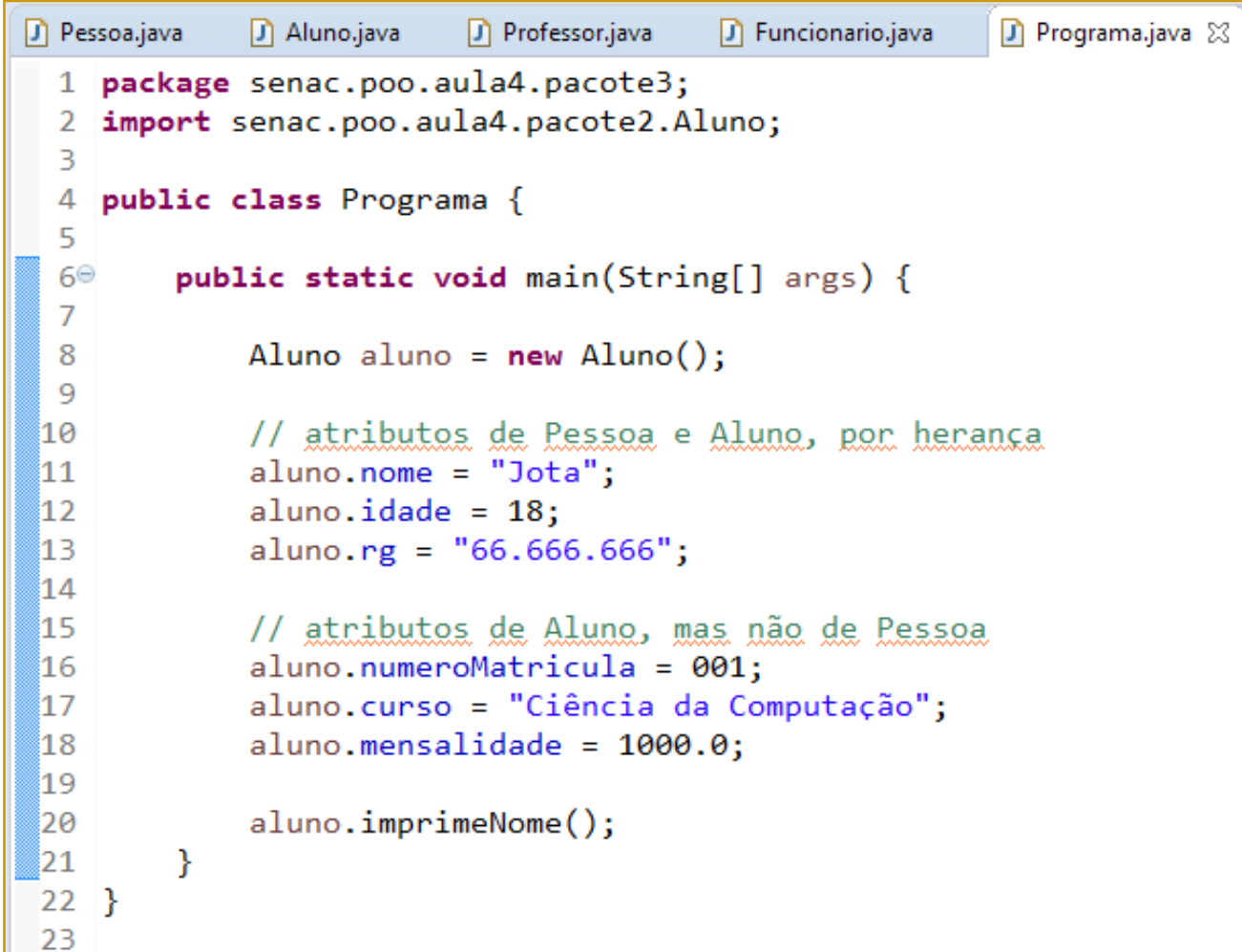
# Subclasse Funcionario



```
Pessoa.java  Aluno.java  Professor.java  Funcionario.java  Programa.java

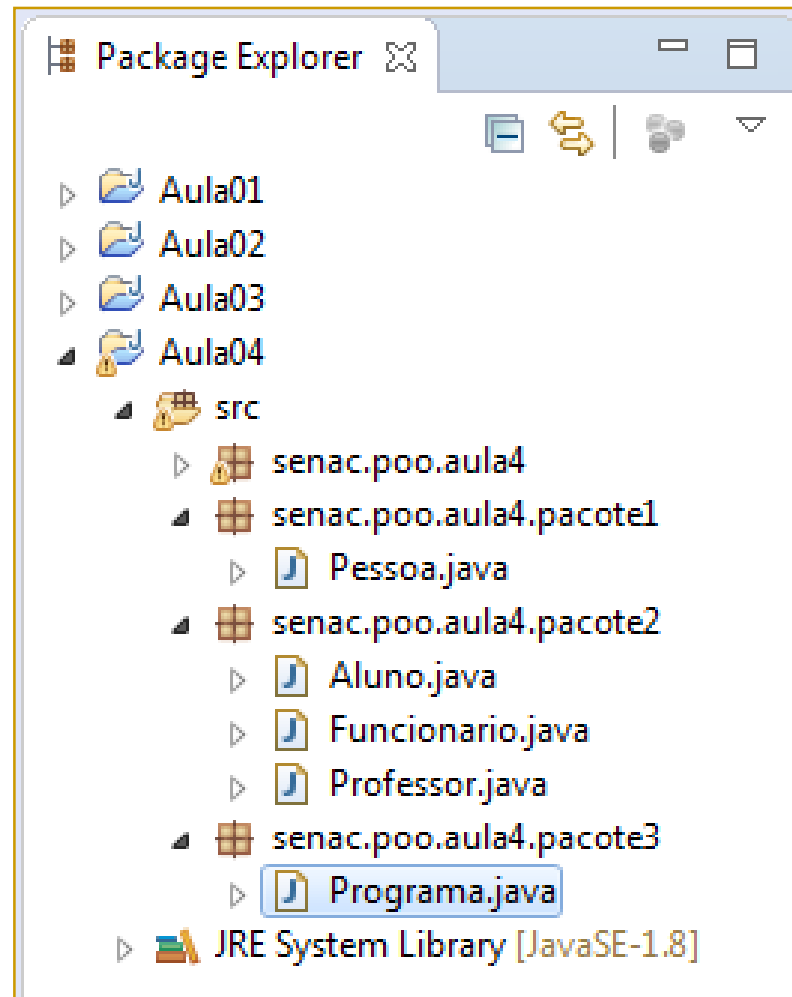
1 package senac.poo.aula4.pacote2;
2 import senac.poo.aula4.pacote1.Pessoa;
3
4 public class Funcionario extends Pessoa{
5
6     public int idFuncionario;
7     public double salario;
8
9     public void imprimeNome() {
10
11         System.out.println("O nome do funcionário é "+ this.nome);
12     }
13 }
14
```

# Classe Programa

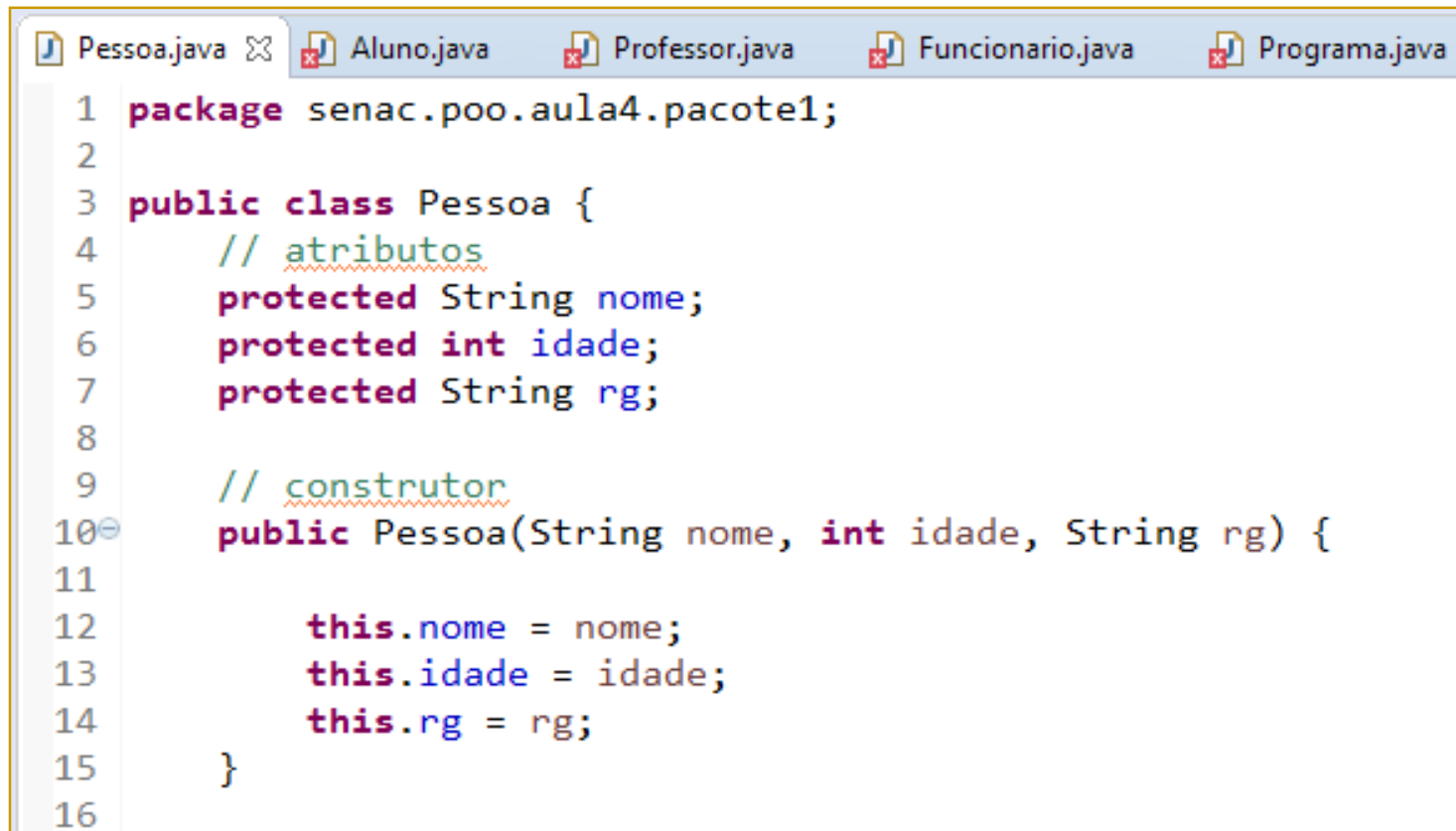


```
1 package senac.poo.aula4.pacote3;
2 import senac.poo.aula4.pacote2.Aluno;
3
4 public class Programa {
5
6     public static void main(String[] args) {
7
8         Aluno aluno = new Aluno();
9
10        // atributos de Pessoa e Aluno, por herança
11        aluno.nome = "Jota";
12        aluno.idade = 18;
13        aluno.rg = "66.666.666";
14
15        // atributos de Aluno, mas não de Pessoa
16        aluno.numeroMatricula = 001;
17        aluno.curso = "Ciência da Computação";
18        aluno.mensalidade = 1000.0;
19
20        aluno.imprimeNome();
21    }
22 }
23
```

# Note que temos 3 pacotes

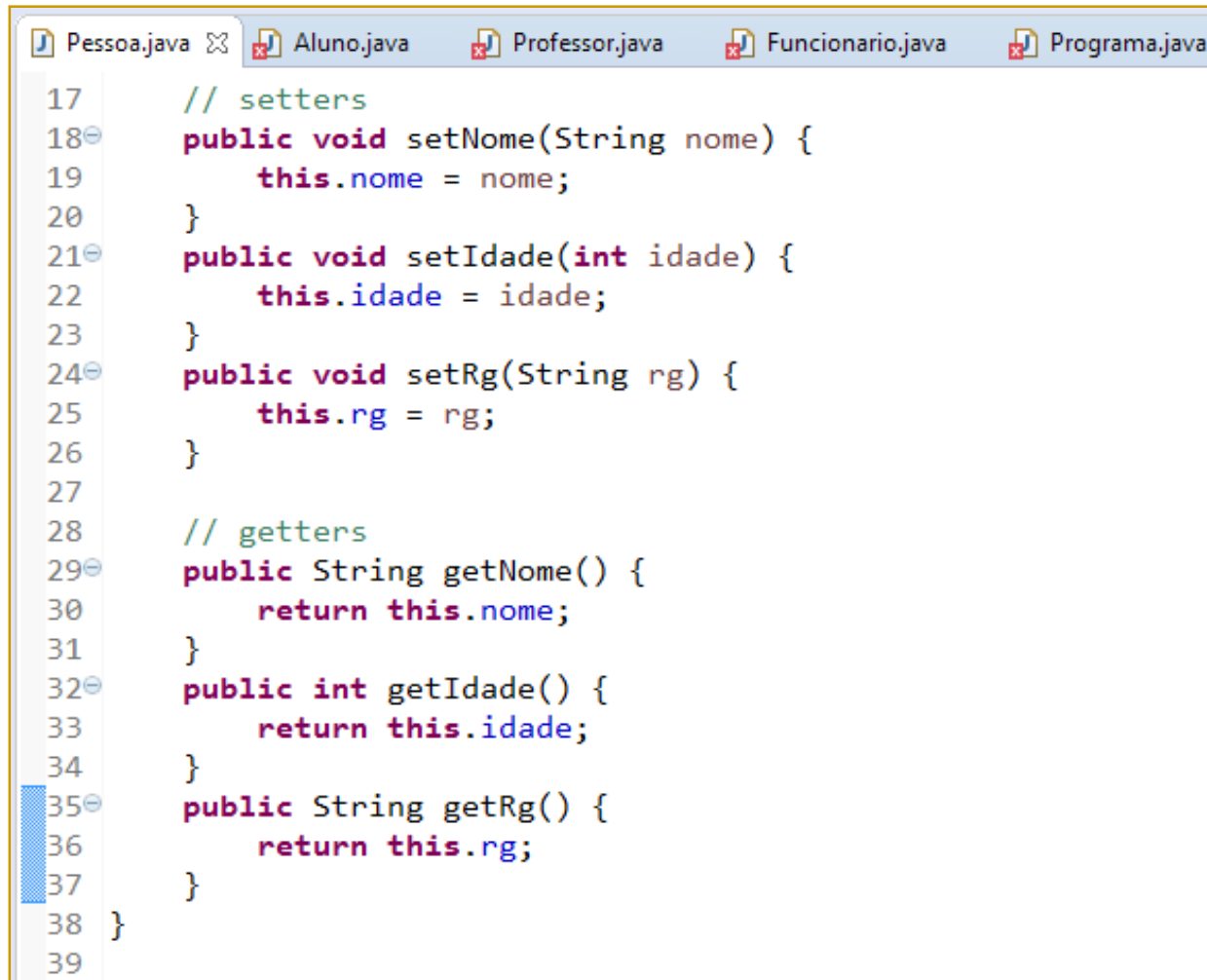


# Encapsulando melhor a classe Pessoa



```
1 package senac.poo.aula4.pacote1;
2
3 public class Pessoa {
4     // atributos
5     protected String nome;
6     protected int idade;
7     protected String rg;
8
9     // construtor
10    public Pessoa(String nome, int idade, String rg) {
11
12        this.nome = nome;
13        this.idade = idade;
14        this.rg = rg;
15    }
16
```

# Encapsulando melhor a classe Pessoa

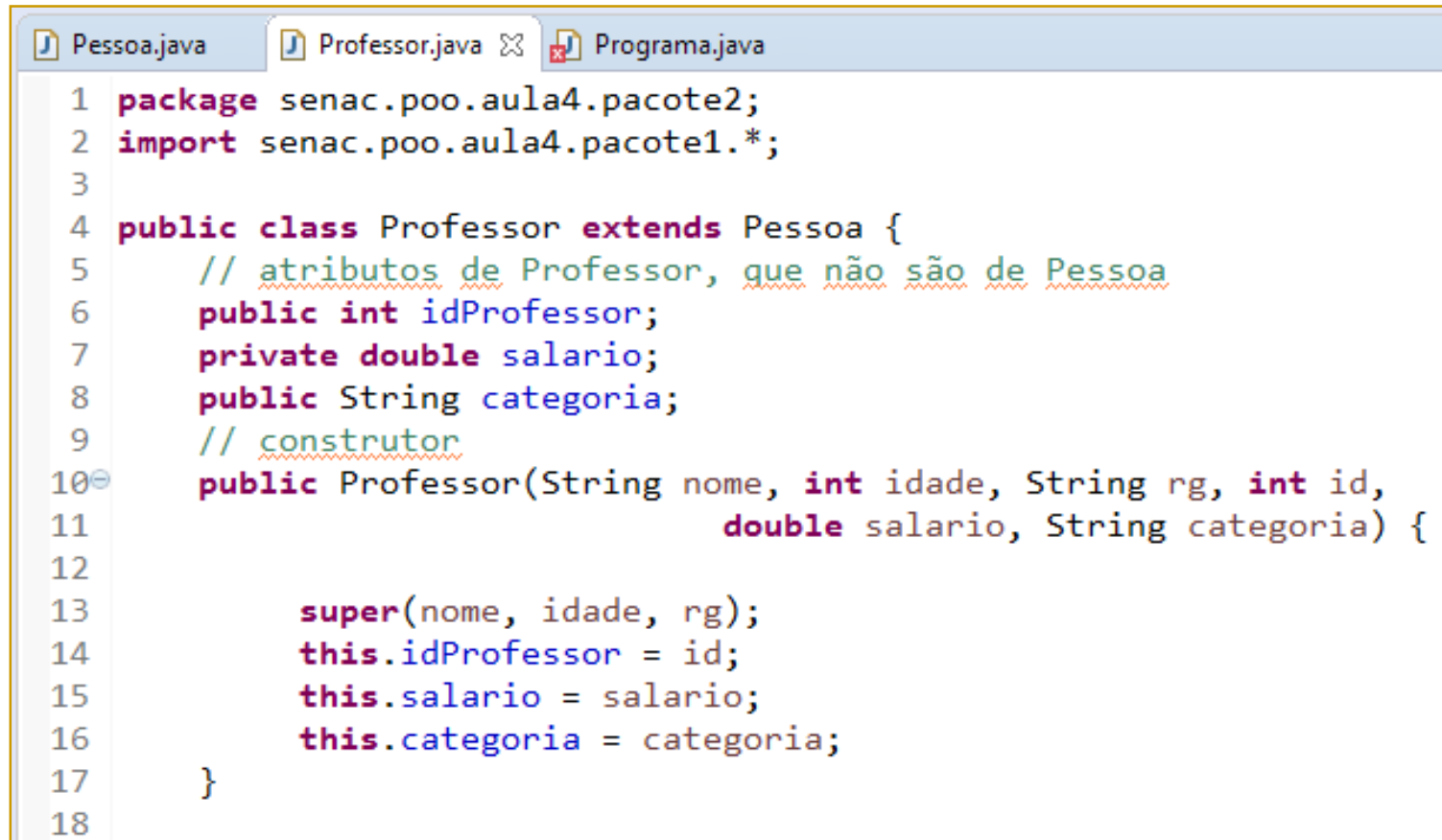


The screenshot shows a Java IDE with five tabs: Pessoa.java, Aluno.java, Professor.java, Funcionario.java, and Programa.java. The Pessoa.java tab is active, displaying the following code:

```
17 // setters
18 public void setNome(String nome) {
19     this.nome = nome;
20 }
21 public void setIdade(int idade) {
22     this.idade = idade;
23 }
24 public void setRg(String rg) {
25     this.rg = rg;
26 }
27
28 // getters
29 public String getNome() {
30     return this.nome;
31 }
32 public int getIdade() {
33     return this.idade;
34 }
35 public String getRg() {
36     return this.rg;
37 }
38 }
39
```

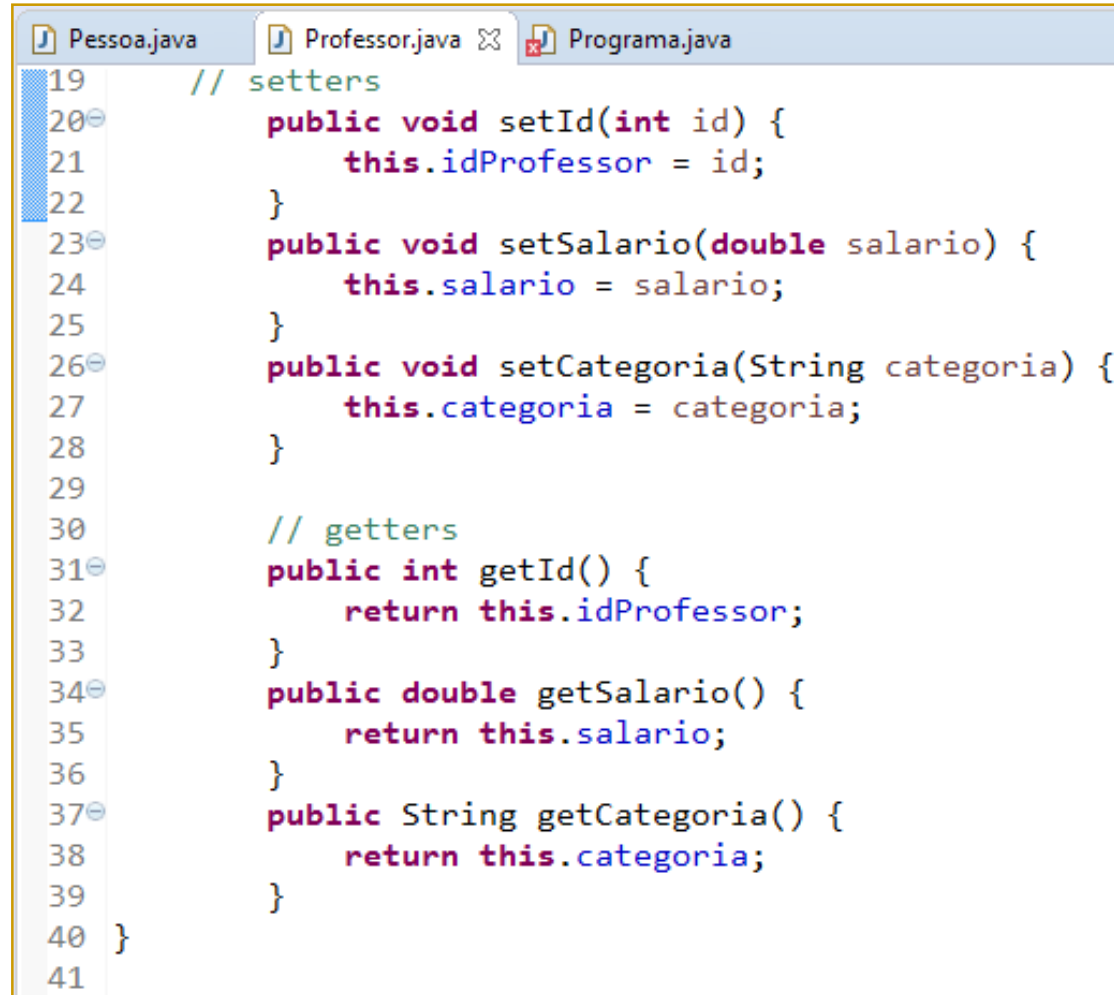


# Encapsulando melhor a classe Professor



```
1 package senac.poo.aula4.pacote2;
2 import senac.poo.aula4.pacote1.*;
3
4 public class Professor extends Pessoa {
5     // atributos de Professor, que não são de Pessoa
6     public int idProfessor;
7     private double salario;
8     public String categoria;
9     // construtor
10    public Professor(String nome, int idade, String rg, int id,
11                    double salario, String categoria) {
12
13        super(nome, idade, rg);
14        this.idProfessor = id;
15        this.salario = salario;
16        this.categoria = categoria;
17    }
18 }
```

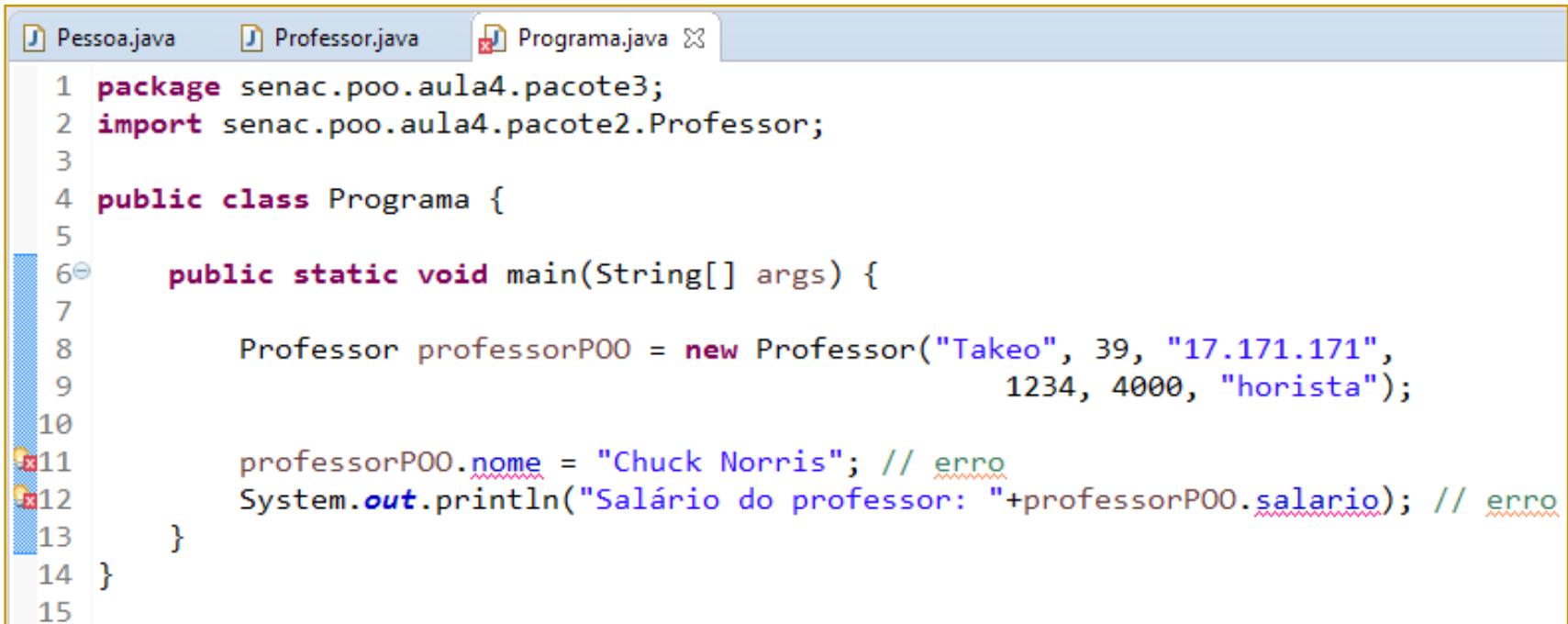
# Encapsulando melhor a classe Professor



The screenshot shows an IDE with three tabs: Pessoa.java, Professor.java (active), and Programa.java. The Professor.java file contains the following code:

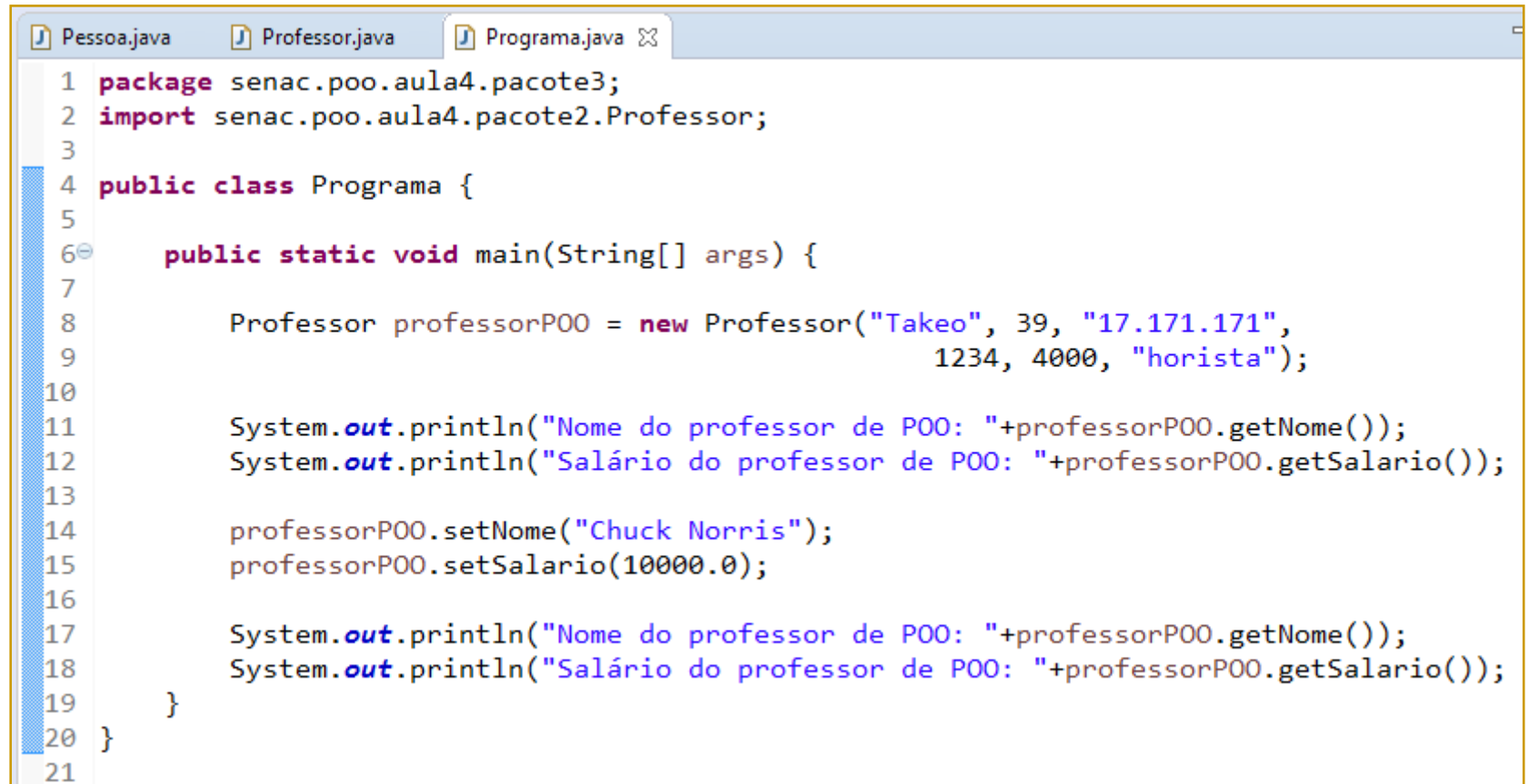
```
19 // setters
20 public void setId(int id) {
21     this.idProfessor = id;
22 }
23 public void setSalario(double salario) {
24     this.salario = salario;
25 }
26 public void setCategoria(String categoria) {
27     this.categoria = categoria;
28 }
29
30 // getters
31 public int getId() {
32     return this.idProfessor;
33 }
34 public double getSalario() {
35     return this.salario;
36 }
37 public String getCategoria() {
38     return this.categoria;
39 }
40 }
41
```

# Classe Programa



```
1 package senac.poo.aula4.pacote3;
2 import senac.poo.aula4.pacote2.Professor;
3
4 public class Programa {
5
6     public static void main(String[] args) {
7
8         Professor professorP00 = new Professor("Takeo", 39, "17.171.171",
9             1234, 4000, "horista");
10
11         professorP00.nome = "Chuck Norris"; // erro
12         System.out.println("Salário do professor: "+professorP00.salario); // erro
13     }
14 }
15
```

# Classe Programa



```
1 package senac.poo.aula4.pacote3;
2 import senac.poo.aula4.pacote2.Professor;
3
4 public class Programa {
5
6     public static void main(String[] args) {
7
8         Professor professorP00 = new Professor("Takeo", 39, "17.171.171",
9             1234, 4000, "horista");
10
11         System.out.println("Nome do professor de P00: "+professorP00.getNome());
12         System.out.println("Salário do professor de P00: "+professorP00.getSalario());
13
14         professorP00.setNome("Chuck Norris");
15         professorP00.setSalario(10000.0);
16
17         System.out.println("Nome do professor de P00: "+professorP00.getNome());
18         System.out.println("Salário do professor de P00: "+professorP00.getSalario());
19     }
20 }
21
```