Agenda

- Herança
  - Relacionamento **É UM**
  - Utilizando métodos de superclasse

Herança (Motivação)

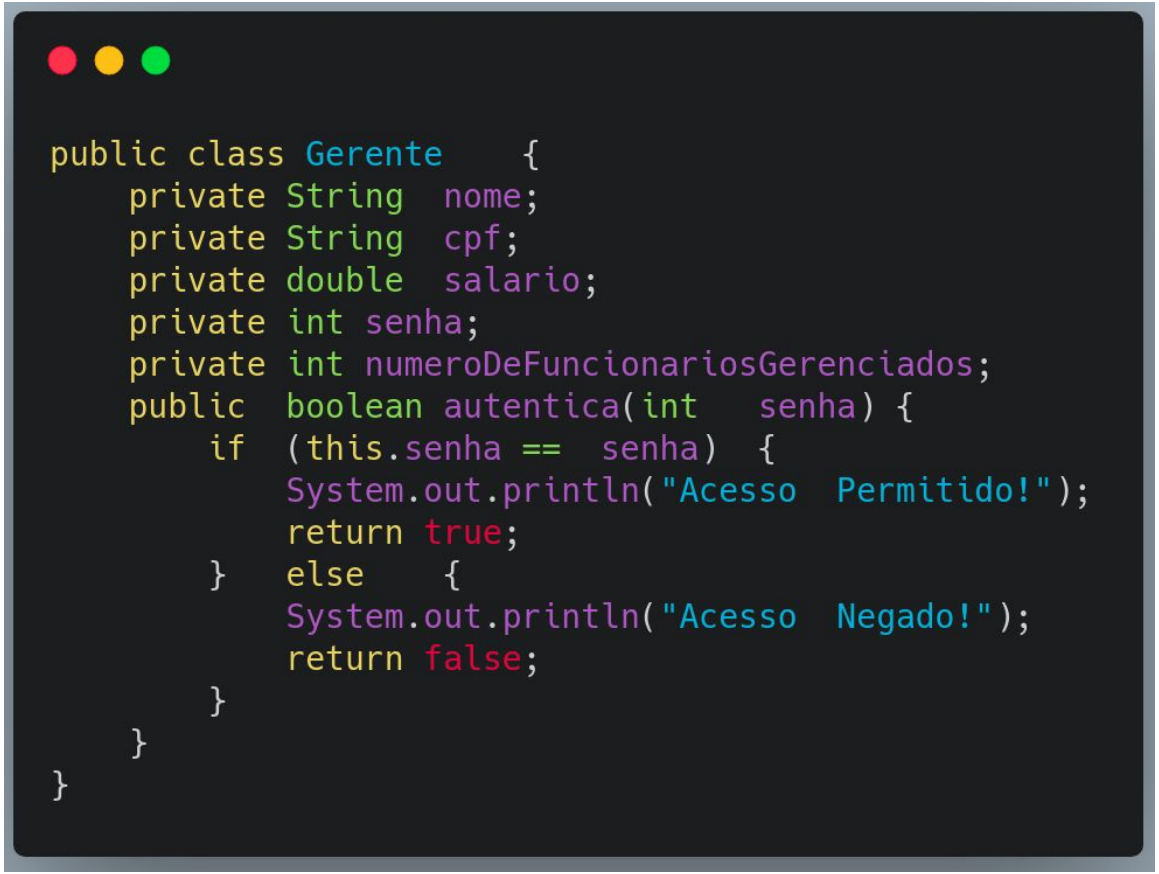
- Suponha um sistema bancário, teremos então duas classes: Funcionário e Gerente.

- Tanto **Funcionário** quanto **Gerente** possuem atributos: **`nome, cpf, salário`**

  Temos que reescrever código?

- Sabemos que todo **Gerente** no fim das contas **é um Funcionário** de um banco. Mas Gerentes possuem privilégios que demais Funcionários não possuem.


public class Funcionario {
private String nome;
private String cpf;
private double salario;
// métodos devem vir aqui
}



```
public class Gerente    {
    private String  nome;
    private String  cpf;
    private double  salario;
    private int senha;
    private int numeroDeFuncionariosGerenciados;
    public  boolean autentica(int   senha) {
        if  (this.senha ==  senha)  {
            System.out.println("Acesso  Permitido!");
            return true;
        }   else    {
            System.out.println("Acesso  Negado!");
            return false;
        }
    }
}
```

Herança (Relacionamento “é um”)

Gerente **é um** Funcionário
Caixa **é um** Funcionário
Analista **é um** Funcionário

Aluno **é uma** Pessoa
Professor **é uma** Pessoa

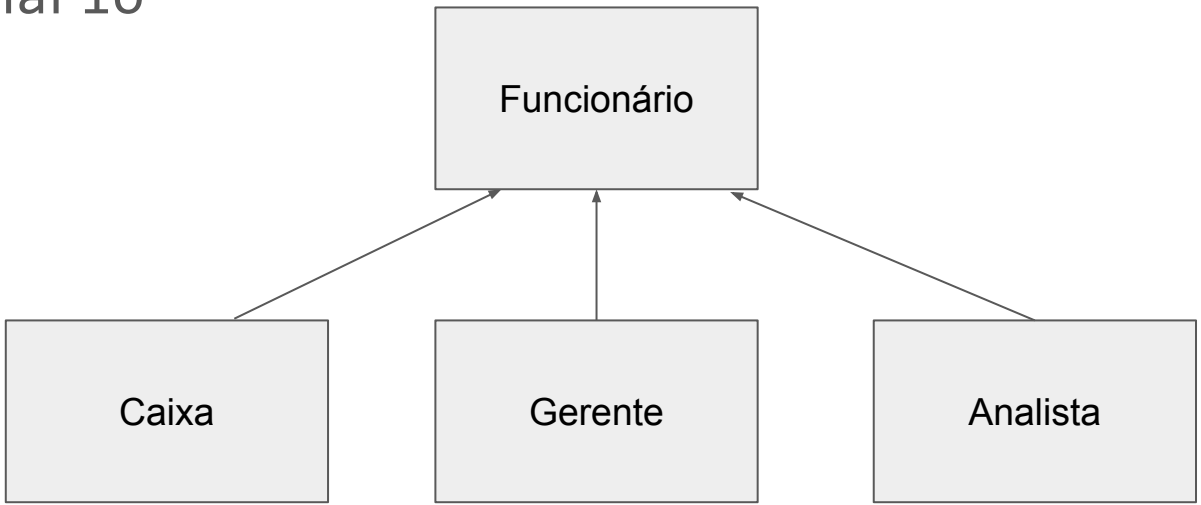
Cachorro **é um** Animal
Gato **é um** Animal

Conta Corrente **é uma** Conta
Carro **é um** Animal ? **X**

Herança (Relacionamento “é um”)

- Funcionário é **superclasse** de Caixa, Gerente e Analista

- Caixa, Gerente e Analista são **subclasses** de Funcionário



Herança (Relacionamento “é um”)

- Em Java criamos esse tipo de relacionamento com a palavra chave ***extends***

- Uma classe Filha(subclasse) herda todos os métodos e atributos da sua classe Pai (superclasse).

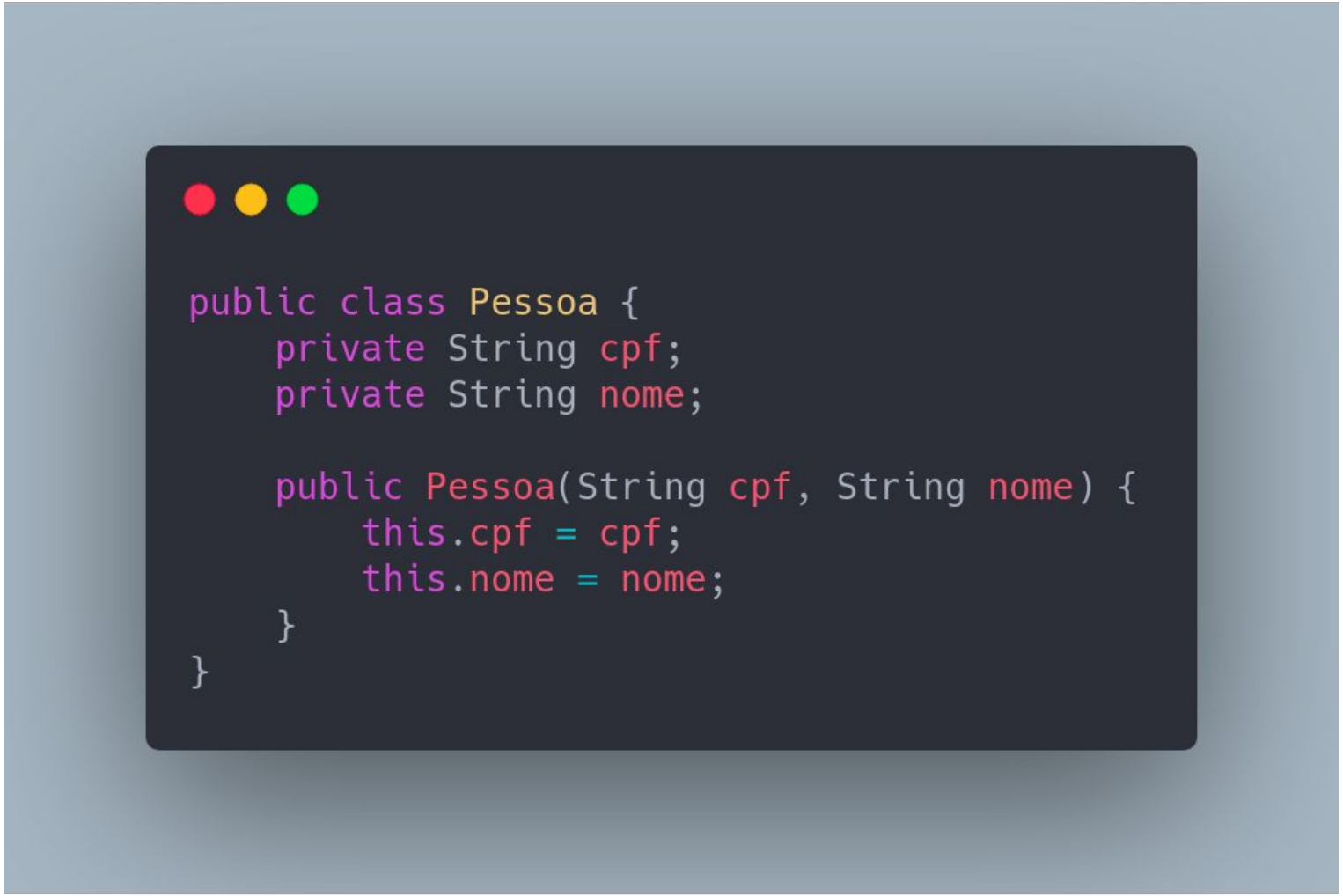
- Métodos e atributos da classe Pai declarados como *private*, não são acessíveis nas classes Filhas. A menos que estejam com modificadores *protected* nos seus atributos

Herança (Chamando métodos da classe Pai)

- Quando queremos chamar o método de uma classe Pai numa classe Filha, utilizamos a palavra-chave ***super***

  ex:

```
super.metodoX();
super(nome, cpf); // construtor
```


public class Pessoa {
private String cpf;
private String nome;
public Pessoa(String cpf, String nome) {
this.cpf = cpf;
this.nome = nome;
}
}

```java
public class Aluno extends Pessoa{
    private String matricula;

    public Aluno(String cpf, String nome, String matricula) {
        super(cpf, nome);  //<-- CHAMANDO O CONSTRUTOR DA CLASSE PAI
        this.matricula = matricula;
    }
}
```