Aula 4

Variáveis compartilhadas de classe

Podemos também passar a variável totalDeContas para dentro da classe e incrementá-la no construtor :

```
class Conta {
   private int totalDeContas;
   //...
   Conta(){
      this.totalDeContas = this.totalDeContas + 1;
   }
}
```

Variáveis compartilhadas de classe

Para solucionar de forma definitiva o problema, precisamos adicionar o atributo static, que torna a variável totalDeContas compartilhada entre todos o objetos da classe:

```
class Conta {
   private static int totalDeContas;
   //...
   Conta(){
      this.totalDeContas = this.totalDeContas + 1;
   }
}
```

Variáveis compartilhadas de classe

Vamos testar a utilização do atributo static.

Recapitulando

O que já aprendemos:

- O que é Java
- Eclipse IDE
- Nosso primeiro código em Java : “Olá Mundo!”
- Variáveis primitivas
- Controle de fluxo
- Laços de repetição
- Orientação a objetos básica
- Modificadores de Acesso e Atributos de Classe:
  - Controlando o Acesso;
  - Encapsulamento;
  - Getters e Setters;
- Construtores:
  - Necessidade de um Construtor;
- Atributo de Classe:
  - O atributo static;

Avante...

O que iremos aprender:

- Herança;
- Reescrita de Métodos;
- Polimorfismo;

Herança

Todo Banco tem contas, clientes e funcionários

Vamos modelar a classe Funcionário :

```
public class Funcionario {
    private String nome;
    private String cpf;
    private double salario;
    // métodos e construtores
}
```

Herança

Além do funcionário temos outros cargos:

- Operadores de Caixa
- Gerentes
- Assistentes
- Diretores

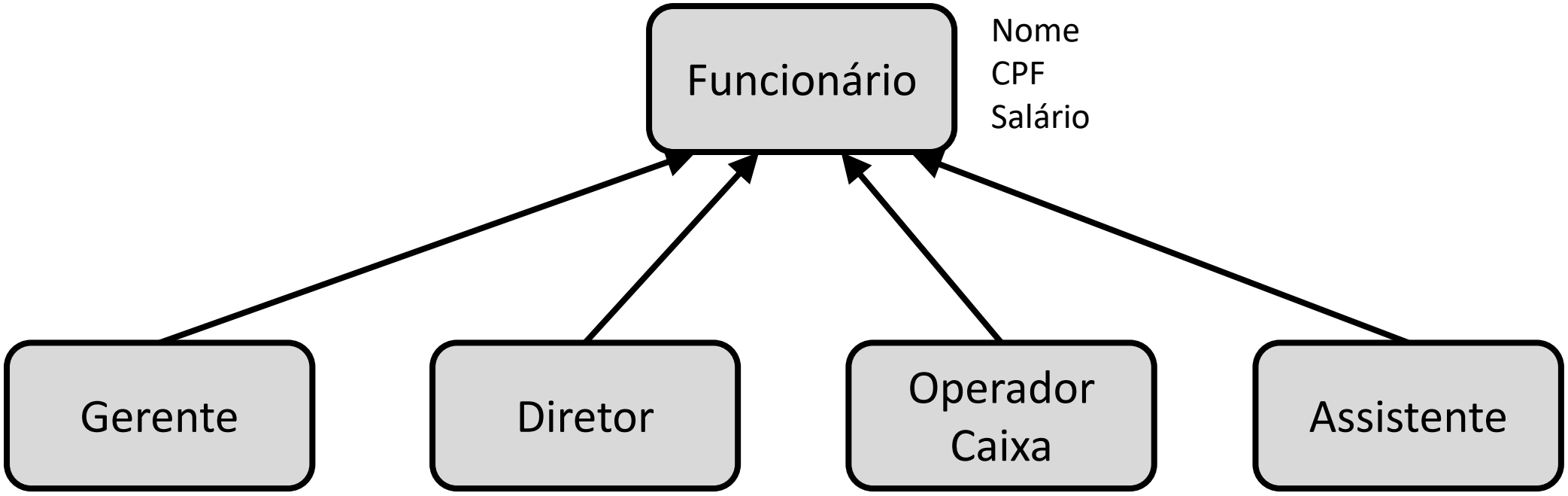
Naturalmente, eles têm informações em comum com os demais funcionários e outras informações exclusivas;

Herança

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

Herança

Podemos ter vários tipos diferentes de funcionários :



Herança

PRECISAMOS MESMO DE OUTRA CLASSE?

- A classe Funcionário poderia ser mais genérica:
  - Mantendo nela senha de acesso;
  - O número de funcionários gerenciados;
  - Caso o funcionário não fosse um gerente, deixaríamos estes atributos vazios.
- Essa é uma possibilidade, porém:
  - Podemos começar a ter muito atributos opcionais;
  - A classe ficaria estranha;
  - E em relação aos métodos?
    - A classe Gerente tem o método autentica ,
    - que não faz sentido existir em um funcionário que não é gerente.

Herança

PRECISAMOS MESMO DE OUTRA CLASSE?

- Se tivéssemos um outro tipo de funcionário;
- Com características diferentes do funcionário comum;
- Precisaríamos criar uma outra classe e copiar o código novamente!
- Se fosse necessário adicionar uma nova informação para todos os funcionários:
  - Precisaríamos passar por todas as classes de funcionário e adicionar esse atributo;
  - O problema acontece por não centralizar as informações principais do funcionário em um único lugar;

Herança

Estendendo a classe Funcionário

- Existe um jeito de relacionar uma classe de tal maneira que uma delas herda tudo que a outra tem.
- Em nosso caso, queremos que Gerente possua todos os métodos e atributos de Funcionario.
- Quando criarmos um objeto do tipo Gerente, este possuirá os atributos da classe Funcionario, pois um Gerente é um Funcionario;

Herança

```
public class Gerente extends Funcionario {
    private int senha;
    private int numeroDeFuncionarios;
    public boolean autentica(int senha) {
        if (this.senha == senha) {
            System.out.println("Acesso Permitido!");
            return true;
        } else {
            System.out.println("Acesso Negado!");
            return false;
        }
    }
    // métodos e construtores
}
```

Herança

Super e sub classe

- Todo Gerente é um Funcionário .
- Nomenclatura usual:
  - Funcionario é a superclasse de Gerente
  - Gerente é a subclasse de Funcionario .
- Outra forma é dizer:
  - Funcionario é classe mãe de Gerente
  - Gerente é classe filha de Funcionario

Herança

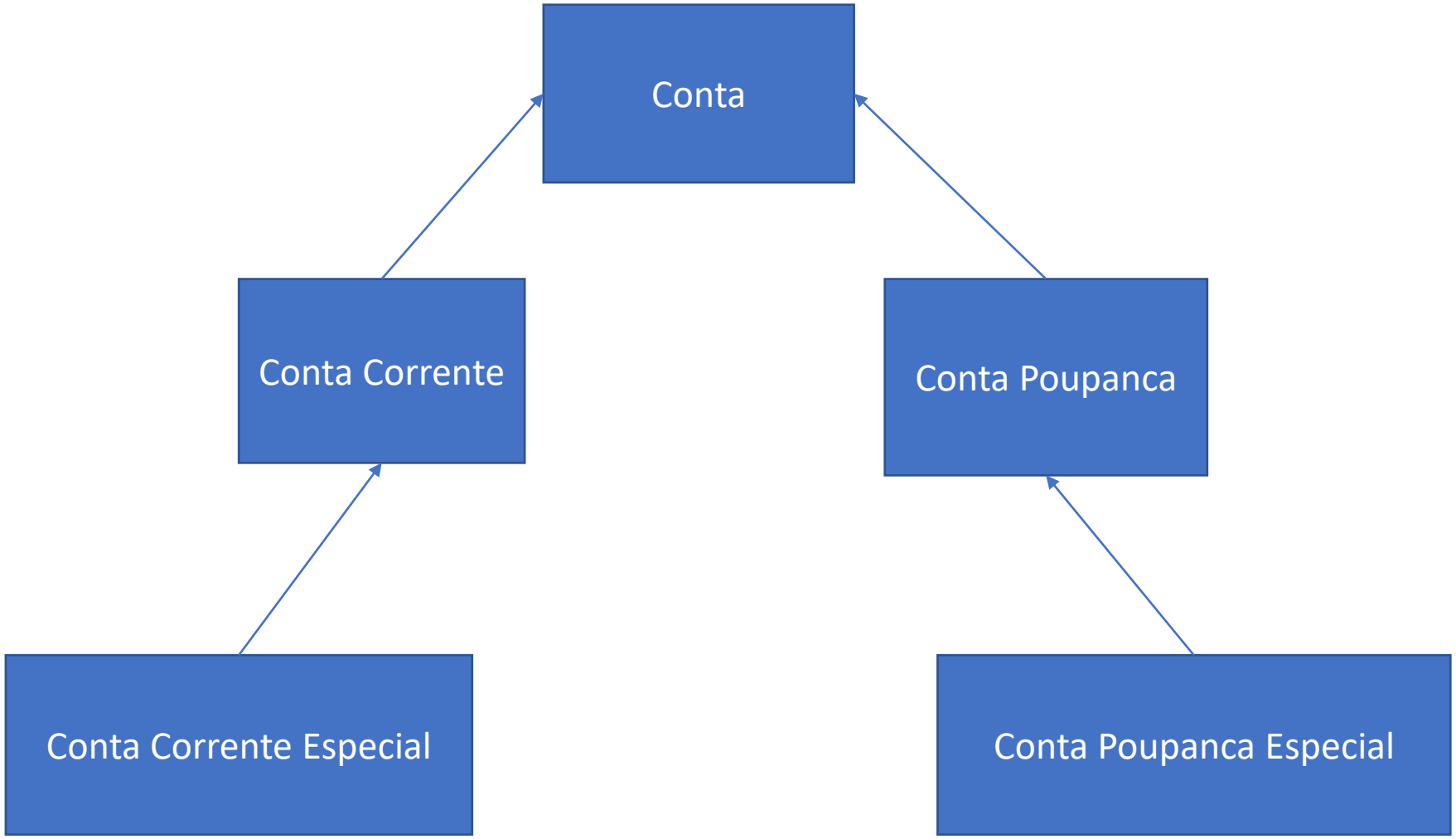
Herança:

- Um dos pilares da Orientação a Objetos;
- Relacionamento entre uma classe base (super classe) e uma classe derivada(sub classe)
- A classe derivada herda atributos e métodos da classe base.
- Usada na intenção de:
  - Criar um padrão de objeto;
  - Reaproveitar código ou comportamento generalizado;
  - Especializar operações ou atributos.

Exercícios

Vamos implementar uma hierarquia de contas?

- Implemente a classe ContaPoupanca como sub classe da classe Conta:
  - Esta conta possui um atributo próprio chamado rendimento;
- Implemente a classe ContaCorrente como sub classe da classe Conta:
  - Esta conta possui como atributo a tarifa e o limite, que representa o cheque especial;
- Construa um objeto de cada uma dessas contas na classe principal;
- Experimente inserir dados nos atributos herdados dessas classes e para comprovar que está funcionando, imprima o modelo da classe;
- Implemente uma ContaPoupancaEspecial que seja subclasse da ContaPoupanca
  - Terá como atributo o cartão de débito em poupança;
- Implemente uma ContaCorrenteEspecial que seja subclasse da ContaCorrente.
  - Terá como atributo um cartão de crédito e investimento;


Conta
Conta Corrente
Conta Poupanca
Conta Corrente Especial
Conta Poupanca Especial