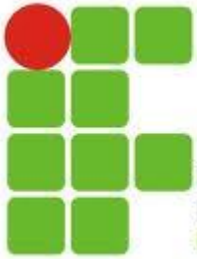


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

# AULA 16
# HERANÇA


**Disciplina: Programação Orientada a Objetos**

**Professora: Alba Lopes**

**alba.lopes@ifrn.edu.br**

# Repetição de Código

- Tomemos como exemplo a classe Funcionario, que representa o funcionário de um banco:

```
public class Funcionario {
    private String nome;
    private String cpf;
    private double salario;

    public String getNome() {
        return nome;
    }

    public void setNome(String nomeFuncionario) {
        nome = nomeFuncionario;
    }

    public String getCpf() {
        return cpf;
    }
```

# Repetição de Código

- Além de um funcionário comum, há também outros cargos, como os gerentes.
  - Os gerentes guardam a mesma informação que um funcionário comum, mas possuem outras informações, além de ter funcionalidades um pouco diferentes
  - Vamos supor que, nesse banco, o gerente possui também uma senha numérica que permite o acesso ao sistema interno do banco

# Repetição de Código

- Classe Gerente:

```
public class Gerente {
    private String nome;
    private String cpf;
    private double salario;

    private int senha;

    public boolean autentica(int testarSenha) {
        if (testarSenha == senha) {
            System.out.println("Acesso Permitido!");
            return true;
        } else {
            System.out.println("Acesso Negado!");
            return false;
        }
    }

     public String getNome() {
        return nome;
    }
```

# Repetição de Código

- Ao invés de criar duas classes diferentes, uma para Funcionario e outra para gerente, poderíamos ter deixado a classe Funcionario mais genérica, mantendo nela senha de acesso.
  - Caso o funcionário não fosse um gerente, deixaríamos este atributo vazio (não atribuiríamos valor a ele).
- Mas e em relação aos métodos?
  - A classe Gerente tem o método autentica, que não faz sentido ser acionado em um funcionário que não é gerente.

# Repetição de Código

- Se tivéssemos um outro tipo de funcionário, que tem características diferentes do funcionário comum, precisaríamos criar uma outra classe, e copiar o código novamente!
- Ou ainda, se um precisássemos adicionar uma nova informação (ex: data de nascimento) para todos os funcionários?
  - Todas as classes teriam que ser alteradas para receber essa informação
- SOLUÇÃO: Centralizar as informações principais do funcionário em um único lugar!

# Herança

- Existe uma maneira, em Java, de relacionarmos uma classe de tal maneira que uma delas herda tudo que a outra tem.
- Isto é uma relação de classe mãe e classe filha.
- No nosso caso, gostaríamos de fazer com que o Gerente tivesse tudo que um Funcionario tem:
  - Gostaríamos que ela fosse uma extensão de Funcionario
- Fazemos isto através da palavra chave **extends**

# HERANÇA

```
public class Gerente extends Funcionario{

    private int senha;

    public boolean autentica(int testarSenha) {
        if (testarSenha == senha) {
            System.out.println("Acesso Permitido!");
            return true;
        } else {
            System.out.println("Acesso Negado!");
            return false;
        }
    }


}
```

# Herança

- Todo momento que criarmos um objeto do tipo Gerente, este objeto possuirá também os atributos definidos na classe Funcionario, pois agora um Gerente é um Funcionario!

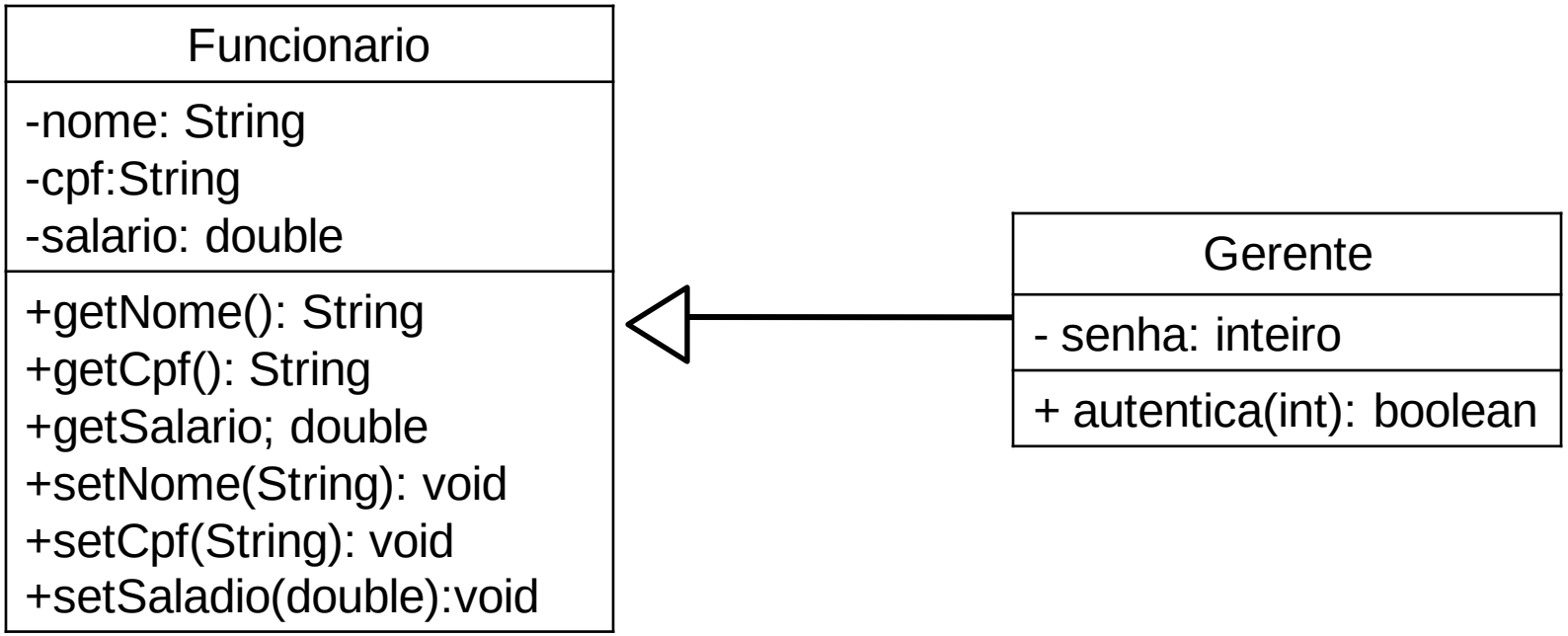
# Herança

- Termos utilizados:

| Classes que fornecem Herança | Classes que herdam de outras |
|---|---|
| Superclasse | Subclasse |
| Pai | Filha |
| Tipo | Subtipo |

# HERANÇA

- Exemplo de notação UML

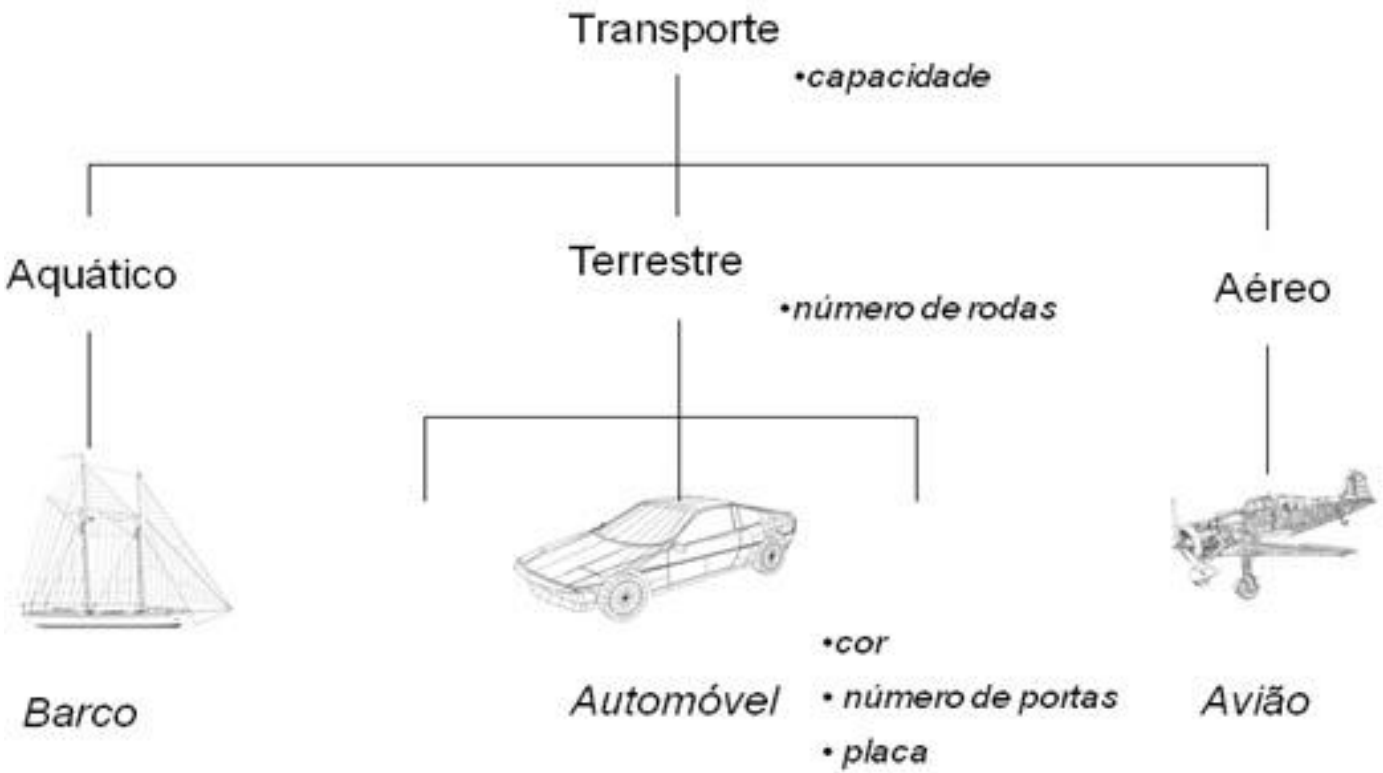


# Herança

- Exemplo de Teste da classe:

```
public class TestarHeranca {
    public static void main(String [] args){
        Gerente gerenteBanco = new Gerente();
        gerenteBanco.setNome("João da Silva");
        gerenteBanco.setCpf("123456789101");
        gerenteBanco.setSenha(123456);
    }
}
```

# Exemplo 1


Transporte
•capacidade
Aquático
Terrestre
•número de rodas
Aéreo
Barco
Automóvel
•cor
• número de portas
• placa
Avião

# Exemplo 1

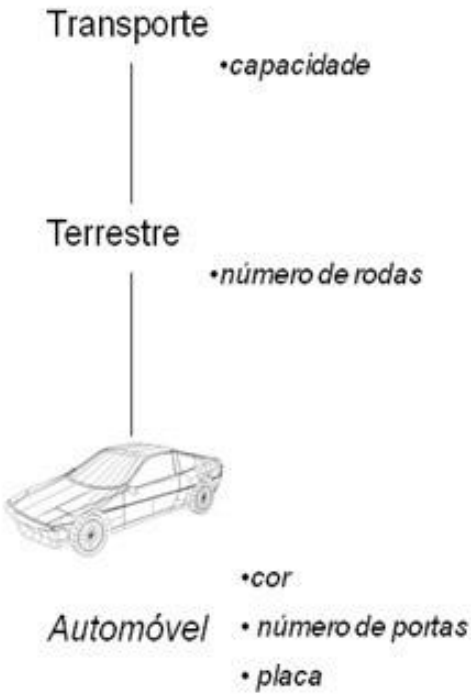
- Usando a lógica de herança responda:
  - Quantos e quais são os atributos da classe Terrestre?
  - E da classe Automóvel?

- Respondendo:
  - a classe Terrestre possui dois atributos:
    - capacidade (que é herdado de Transporte) e número de rodas.
  - Já a classe Automóvel possui cinco atributos:
    - capacidade (herdado de Transporte), número de rodas (herdado de Terrestre), cor, número de portas e placa.

# EXEMPLO 1

**Veja o código completo de cada uma das classes no pacote aula16_exemplos.Transporte**

- Como seria o código?




```
public class Transporte{
        private int capacidade;
}
```

```
public class Terrestre extends Transporte{
        private int numRodas;
}
```

```
public class Automovel extends Terrestre{
        private String cor;
        private int numPortas;
        private String placa;
}
```

# EXEMPLO 1

```
public class TestarTransporte {

    public static void main(String [] args){

        Transporte t1 = new Transporte();
        t1.setCapacidade(300);
        System.out.println("A capacidade de T1 é: " + t1.getCapacidade());

        Terrestre t2 = new Terrestre();
        t2.setCapacidade(30);
        System.out.println("A capacidade de T2 é: " + t2.getCapacidade());


        Automovel t3 = new Automovel();
        t3.setCapacidade(5);
        System.out.println("A capacidade de T3 é: " + t3.getCapacidade());

    }

}
```

# Exercício

- No projeto **Exemplos_e_Exercicios_POO**, crie um pacote **aula16_exercicios**

1. Crie uma classe **Animal** que obedeça à seguinte descrição:
   - possua os atributos **nome** (String), **comprimento (**float**)**, **número de patas** (int), **cor** (String), **ambiente** (String) e **velocidade média** (float)
   - Crie um método construtor que receba por parâmetro os valores iniciais de cada um dos atributos e atribua-os aos seus respectivos atributos.
   - Crie os métodos **get** e **set** para cada um dos atributos.
   - Crie um método **dados**, sem parâmetro e do tipo **void**, que, quando chamado, imprime na tela uma espécie de relatório informando os dados do animal.

# Exercício

2. Crie uma classe **Peixe** que herde da classe Animal e obedeça à seguinte descrição:
   - possua um atributo **caracteristica**(String)
   - Crie um método construtor que receba por parâmetro os valores iniciais de cada um dos atributos (incluindo os atributos da classe **Animal**) e atribua-os aos seus respectivos atributos.
   - Crie ainda os métodos **get** e **set** para o atributo **caracteristica**.
   - Crie um método **dadosPeixe** sem parâmetro e do tipo void, que, quando chamado, imprime na tela uma espécie de relatório informando os dados do peixe (incluindo os dados do Animal e mais a característica).

# Exercício

3. Crie uma classe **Mamifero** que herde da classe Animal e obedeça à seguinte descrição:
   - possua um atributo **alimento**(String)
   - Crie um método construtor que receba por parâmetro os valores iniciais de cada um dos atributos (incluindo os atributos da classe **Animal**) e atribua-os aos seus respectivos atributos.
   - Crie ainda os métodos **get** e **set** para o atributo **alimento**.
   - Crie um método **dadosMamifero** sem parâmetro e do tipo void, que, quando chamado, imprime na tela uma espécie de relatório informando os dados do mamifero (incluindo os dados do Animal e mais o alimento).

# Exercício

4. Crie uma classe **TestarAnimais** possua um método main para testar as classes criadas.
   a) Crie um objeto **camelo** do tipo Mamífero e atribua os seguintes valores para seus atributos:
      - Nome: Camelo
      - Comprimento: 150 cm
      - Patas: 4
      - Cor: Amarelo
      - Ambiente: Terra
      - Velocidade: 2.0 m/s

# Exercício

4. (cont.)
   b) Crie um objeto **tubarao** do tipo Peixe e atribua os seguintes valores para seus atributos
      - Nome: Tubarão
      - Comprimento: 300 cm
      - Patas: 0
      - Cor: Cinzento
      - Ambiente: Mar
      - Velocidade: 1.5 m/s
      - Caracteristica: Barbatanas e cauda

# Exercício

4. (cont.)
   c) Crie um objeto **ursocanada** do tipo Mamifero e atribua os seguintes valores para seus atributos:
      - Nome: Urso-do-canadá
      - Comprimento: 180 cm
      - Patas: 4
      - Cor: Vermelho
      - Ambiente: Terra
      - Velocidade: 0.5 m/s
      - Alimento: Mel

# Exercício

4. (cont.)
   d) Chame os método para imprimir os dados de cada um dos objetos criados.

# Referências


- Caelum - Java e Orientação a Objetos
- Métropole Digital – Programação Orientada a Objetos: http://www.metropoledigital.ufrn.br/aulas/disciplinas/poo/aula_10.html