

# Implementação de Herança

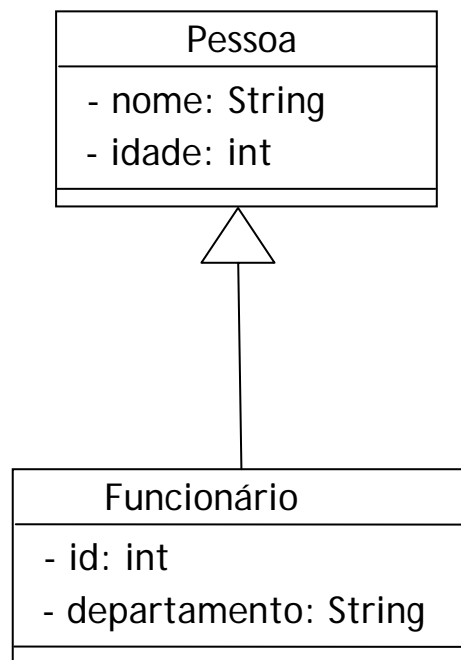
---

## Orientação a Objetos - DCC025

Prof. Edmar Welington Oliveira  
oliveira.edmar@ufjf.edu.br

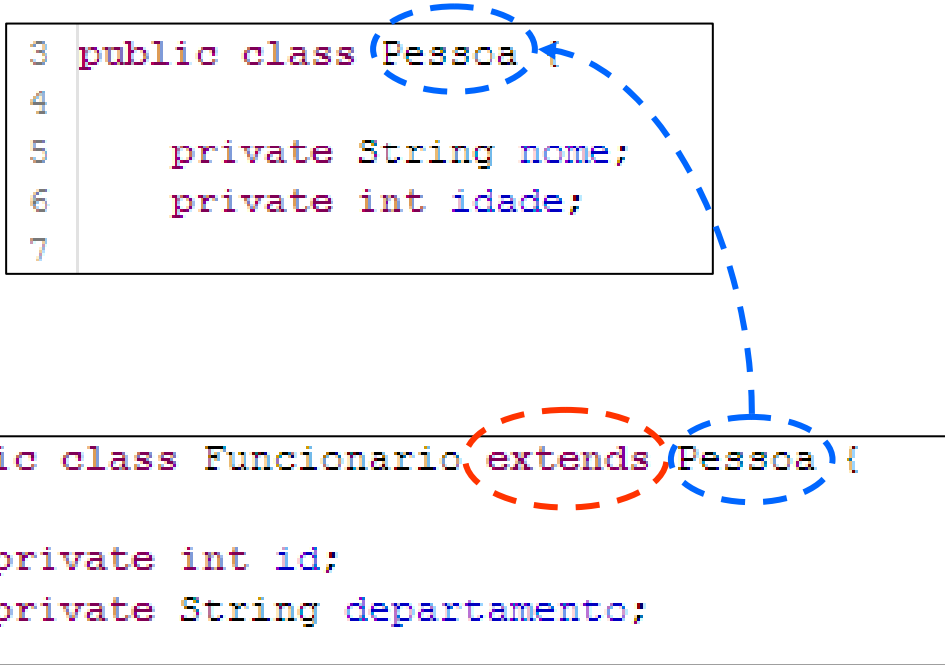
Universidade Federal de Juiz de Fora - UFJF  
Departamento de Ciência da Computação - DCC

# Herança



# Implementação de Herança

```
3 public class Pessoa {  
4  
5     private String nome;  
6     private int idade;  
7 }
```



A diagram illustrating inheritance. A dashed blue arrow points from the `Pessoa` class in the top code block to the `Funcionario` class in the bottom code block. The `extends` keyword in the bottom code block is circled with a dashed red line, and the `Pessoa` class name is circled with a dashed blue line.

```
3 public class Funcionario extends Pessoa {  
4  
5     private int id;  
6     private String departamento;  
7 }
```

Quais seriam os atributos de um  
objeto do tipo Funcionário?

# Construtor de Subclasse

- Chamada a construtor de subclasse
  - Cada construtor de subclasse deve chamar, implícita ou explicitamente, seu construtor de superclasse. Isso assegura que as variáveis de instância herdadas da superclasse sejam inicializadas adequadamente.
  - Sintaxe para chamar de construtor de superclasse
    - Palavra reservada "super"
    - Argumentos de construtor de superclasse

# Exemplo

- Execute:
  - Crie uma classe Pessoa com atributos nome (String) e idade (int)
  - Crie uma classe Funcionário com atributos id (int) e departamento (String)
  - Faça Funcionário estender Pessoa
  - Crie um construtor para a classe Pessoa (com os dois argumentos)
  - Verifique na classe Funcionário algum alerta de erro
  - Analise o alerta: o que deve ser feito para resolver o erro?


# Exemplo

```
3 public class Pessoa {  
4  
5     private String nome;  
6     private int idade;  
7  
8     public Pessoa(String nome, int idade){  
9         this.nome = nome;  
10        this.idade = idade;  
11    }  
12 }
```

```
3 public class Funcionario extends Pessoa {  
4  
5     private int ...  
6     private String ...  
7 }  
8
```

Implicit super constructor Pessoa() is undefined for default constructor. Must define an explicit constructor

1 quick fix available:

 [Add constructor 'Funcionario\(String,int\)'](#)

# Exemplo

```
3 public class Pessoa {  
4  
5     private String nome;  
6     private int idade;  
7  
8     public Pessoa(String nome, int idade){  
9         this.nome = nome;  
10        this.idade = idade;  
11    }  
12 }
```

```
3 public class Funcionario extends Pessoa {  
4  
5     private int ...  
6     private St ...  
7 }  
8
```

Implicit super constructor Pessoa() is undefined for default constructor. Must define an explicit constructor

1 quick fix available:

- [Add constructor 'Funcionario\(String,int\)'](#)

Lembre-se que, para cada classe, o Java cria um construtor padrão para a mesma (se nenhum construtor tiver sido especificado). Contudo, para a classe Funcionário, esse construtor padrão não possui a chamada `super()` - o que gera o erro acima

# Exemplo

```
3 public class Funcionario extends Pessoa {  
4  
5     private int id;  
6     private String departamento;  
7 }  
8
```

Implicit super constructor Pessoa() is undefined for default constructor. Must define an explicit constructor

1 quick fix available:

- [Add constructor "Funcionario\(String,int\)"](#)



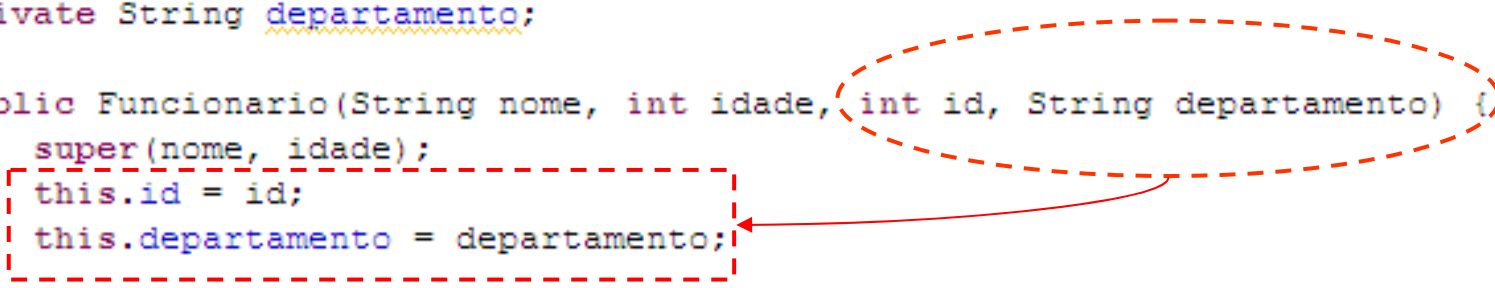
```
3 public class Funcionario extends Pessoa{  
4  
5     private int id;  
6     private String departamento;  
7  
8     public Funcionario(String nome, int idade) {  
9         super(nome, idade);  
10    }  
11 }
```



# Exemplo

```
3 public class Funcionario extends Pessoa {  
4  
5     private int id;  
6     private String departamento;  
7  
8     public Funcionario(String nome, int idade) {  
9         super(nome, idade);  
10    }  
11 }
```

```
3 public class Funcionario extends Pessoa {  
4  
5     private int id;  
6     private String departamento;  
7  
8     public Funcionario(String nome, int idade, int id, String departamento) {  
9         super(nome, idade);  
10        this.id = id;  
11        this.departamento = departamento;  
12    }  
13 }
```






# Exemplo

- Execute
  - Na classe Funcionário
    - Altere a chamada ao construtor da superclasse omitindo os parâmetros
    - Verifique se algum alerta de erro foi emitido
    - Analise o alerta
    - Verifique as possibilidade para “consertar” o erro

# Exemplo

 The constructor Pessoa() is undefined

3 quick fixes available:

-  [Add arguments to match 'Pessoa\(String, int\)'](#)
-  [Change constructor 'Pessoa\(String, int\)': Remove parameters 'String, int'](#)
-  [Create constructor 'Pessoa\(\)'](#)

```
public Funcionario(String nome, int idade, int id, String departamento) {  
    super(departamento, id);  
    this.id = id;  
    this.departamento = departamento;  
}
```

Problema

```
3 public class Pessoa {  
4  
5     private String nome;  
6     private int idade;  
7  
8     public Pessoa(String nome, int idade){  
9         this.nome = nome;  
10        this.idade = idade;  
11    }  
12 }
```

# Exemplo

```
3 public class Funcionario extends Pessoa{
4
5     private int id;
6     private String departamento;
7
8     public Funcionario(int id, String departamento, String nome, int idade) {
9         super(nome, idade);
10        this.id = id;
11        this.departamento = departamento;
12    }
13 }
14 }
```

# Chamada a Construtor Padrão

- Problema

- Se o construtor da subclasse Funcionario não invocasse o construtor de Pessoa explicitamente (com o uso de `super()`), o Java tentaria invocar o construtor sem argumentos padrão da classe Pessoa - **mas a classe não possui esse construtor (pelo nosso exemplo)**
  - Resultado: o compilador emitiria um erro.

# Exemplo

```
3 public class Pessoa {  
4  
5     private String nome;  
6     private int idade;  
7 }
```

```
3 public class Funcionario extends Pessoa{  
4  
5     private int id;  
6     private String departamento;  
7  
8 }
```

Essas classes não possuem construtor (ou melhor, possuem o construtor padrão criado pelo Java). Embora não se tenha a palavra reservada `super()` na classe `Funcionario`, não é gerado um erro, pois o Java:

Da classe `Funcionario` chama o construtor padrão da classe `Pessoa`

# Exemplo

```
3 public class Pessoa {  
4  
5     private String nome;  
6     private int idade;  
7 }
```

```
3 public class Funcionario extends Pessoa{  
4  
5     private int id;  
6     private String departamento;  
7  
8     public Funcionario(int id, String departamento){  
9         super();  
10    }  
11 }
```

Porque não é gerado um erro no código da classe Funcionário após a inclusão da palavra reservada `super()`?

# Exemplo

```
3 public class Pessoa {
4
5     private String nome;
6     private int idade;
7
8     public Pessoa(String nome, int idade){
9         this.nome = nome;
10        this.idade = idade;
11    }
12 }
```

```
3 public class Funcionario extends Pessoa{
4
5     private int id;
6     private String departamento;
7
8     public Funcionario(int id, String departamento){
9         super();
10    }
11 }
```

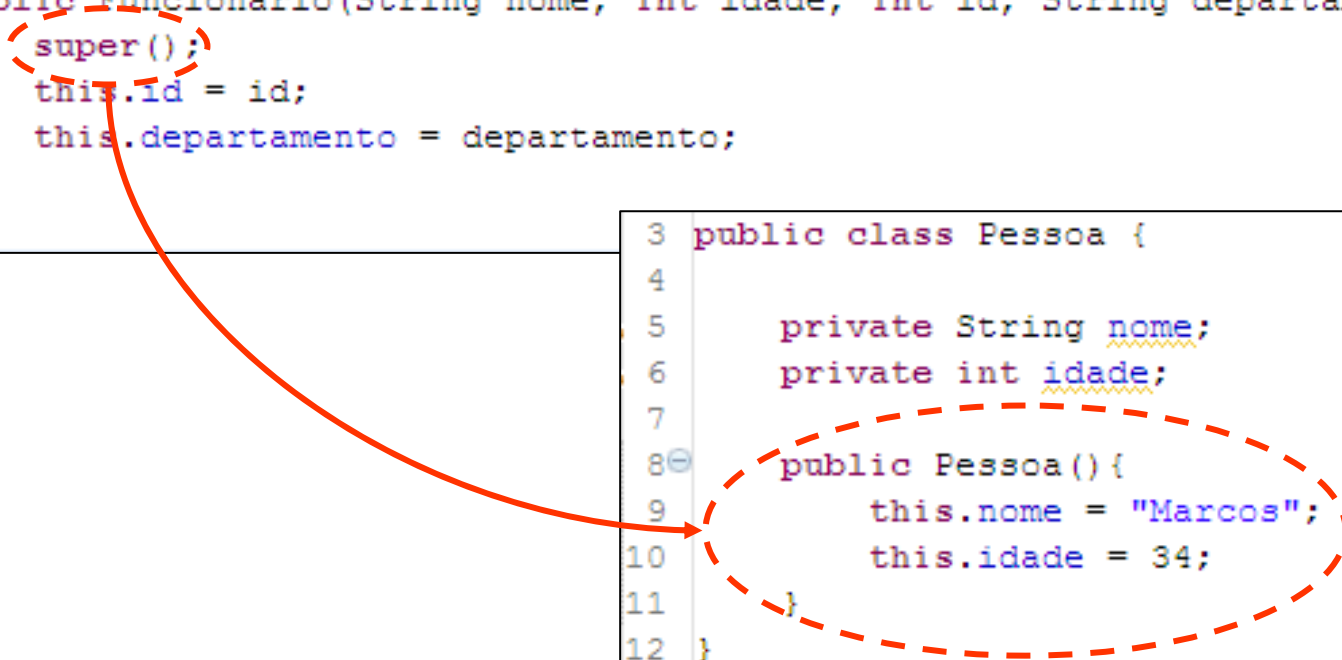
Porque não é gerado um erro no código da classe Funcionário após a inclusão de um construtor na classe Pessoa? Há duas formas para corrigir o erro



# Correção 01

```
3 public class Funcionario extends Pessoa {  
4  
5     private int id;  
6     private String departamento;  
7  
8     public Funcionario(String nome, int idade, int id, String departamento) {  
9         super();  
10        this.id = id;  
11        this.departamento = departamento;  
12    }  
13 }
```

```
3 public class Pessoa {  
4  
5     private String nome;  
6     private int idade;  
7  
8     public Pessoa() {  
9         this.nome = "Marcos";  
10        this.idade = 34;  
11    }  
12 }
```



## Correção 02

```
3 public class Pessoa {  
4  
5     private String nome;  
6     private int idade;  
7  
8     public Pessoa(String nome, int idade){  
9         this.nome = nome;  
10        this.idade = idade;  
11    }  
12 }
```

```
3 public class Funcionario extends Pessoa{  
4  
5     private int id;  
6     private String departamento;  
7  
8     public Funcionario(int id, String departamento, int idade, String nome){  
9         super(nome, idade);  
10        this.departamento = departamento;  
11        this.id = id;  
12    }  
13 }
```

# Construtor de Subclasse

- Chamada a construtor
  - A chamada de construtor de superclasse deve ser a primeira instrução no corpo do construtor de subclasse. Primeiro deve-se inicializar os campos das superclasses e, após, inicializar os construtores da subclasse.

# Construtores em Subclasses

- Construtores em subclasse
  - Instanciar um objeto de subclasse inicia uma cadeia de chamadas de construtor, em que o construtor da subclasse, antes de realizar suas próprias ações, invoca o construtor de sua superclasse direta de forma explícita (via referência `super`) ou implicitamente (chamando o construtor padrão ou o construtor sem argumentos da superclasse).
  - De forma semelhante, se a superclasse é derivada de outra classe, o construtor da superclasse invoca o construtor da próxima classe no topo da hierarquia, e assim por diante
  - OBS
    - O último construtor chamado é sempre o construtor da classe `Object`.
    - O corpo do construtor da subclasse original termina a execução por último.

# Implementação de Herança

---

## Orientação a Objetos - DCC025

Prof. Edmar Welington Oliveira  
oliveira.edmar@ufjf.edu.br

Universidade Federal de Juiz de Fora - UFJF  
Departamento de Ciência da Computação - DCC