

Classe Funcionario tem Endereco

Criar um projeto -> Aula03

Criar uma classe Funcionario contendo 2 atributos (código, nome).

```
package entity;

    public class Funcionario {

        private Integer codigo;
        private String nome;
    }
```

Criar uma classe Endereco contendo os seguintes atributos: código, bairro, cidade, os construtores, toString e getters e setters.

```
package entity;

public class Endereco {

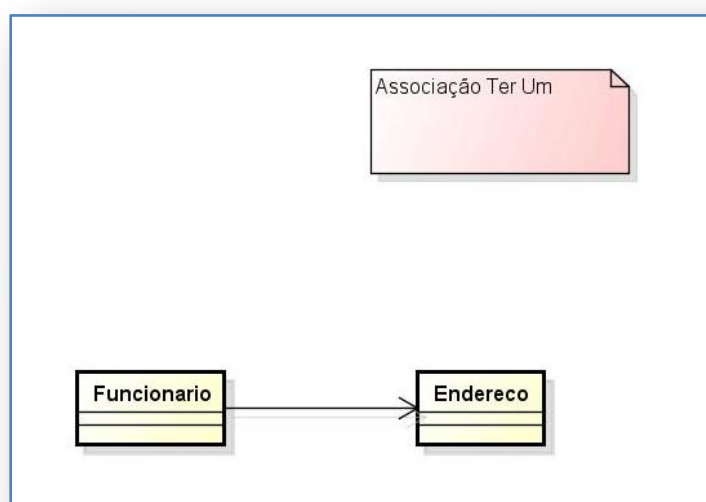
    private Integer codigo;
    private String bairro;
    private String cidade;

    public Endereco() {
    }

    public Endereco(Integer codigo, String bairro, String
cidade) {
        super();
        this.codigo = codigo;
        this.bairro = bairro;
        this.cidade = cidade;
    }

    @Override
```

```
public String toString() {  
    return "Endereco [codigo=" + codigo + ", bairro=" +  
    bairro + ", cidade=" + cidade + "];"  
}  
  
public Integer getCodigo() {  
    return codigo;  
}  
}  
public void setCodigo(Integer codigo) {  
    this.codigo = codigo;  
}  
public String getBairro() {  
    return bairro;  
}  
public void setBairro(String bairro) {  
    this.bairro = bairro;  
}  
public String getCidade() {  
    return cidade;  
}  
public void setCidade(String cidade) {  
    this.cidade = cidade;  
}  
}
```



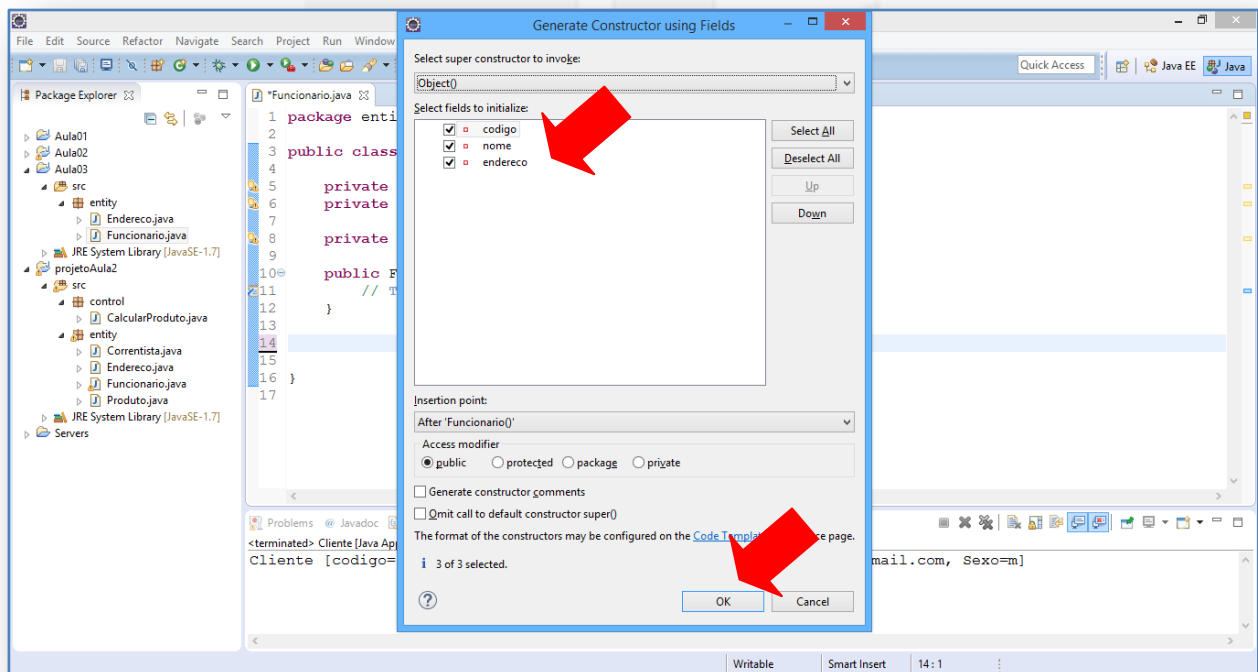
Voltando na classe Funcionario, criar o relacionamento do funcionário com endereço. Acrescentamos o atributo endereço na classe. Ficando dessa forma:

```
package entity;
```

```
public class Funcionario {  
  
    private Integer codigo;  
    private String nome;  
  
    private Endereco endereco;
```

A classe Endereco virou um atributo da classe Funcionario. Dessa forma indicamos que Funcionario tem um Endereco. Agora criar os outros métodos restantes (construtores, toString e getters e setters).

```
public Funcionario() {  
    // TODO Auto-generated constructor stub  
}
```



```
public Funcionario(Integer codigo, String nome, Endereco  
endereco) {  
    super();  
    this.codigo = codigo;  
    this.nome = nome;
```

```
        this.endereco = endereco;  
    }
```

Criar um construtor cheio completo, com todos os atributos, incluindo endereço e criar um outro construtor cheio sem o atributo endereço.

```
public Funcionario(Integer codigo, String nome) {  
    super();  
    this.codigo = codigo;  
    this.nome = nome;  
}
```

O toString será completo, contendo o endereço, pois é a saída. Imprimirá funcionário contendo endereço.

```
@Override  
public String toString() {  
    return "Funcionario [codigo=" + codigo + ", nome=" +  
nome + ", endereco=" + endereco + "];"  
}
```

E os getters e setters de todos os atributos.

```
public Integer getCodigo() {  
    return codigo;  
}  
public void setCodigo(Integer codigo) {  
    this.codigo = codigo;  
}  
public String getNome() {  
    return nome;  
}  
public void setNome(String nome) {  
    this.nome = nome;  
}  
public Endereco getEndereco() {  
    return endereco;  
}  
public void setEndereco(Endereco endereco) {
```

```
        this.endereco = endereco;  
    }
```

A classe completa ficará da seguinte forma:

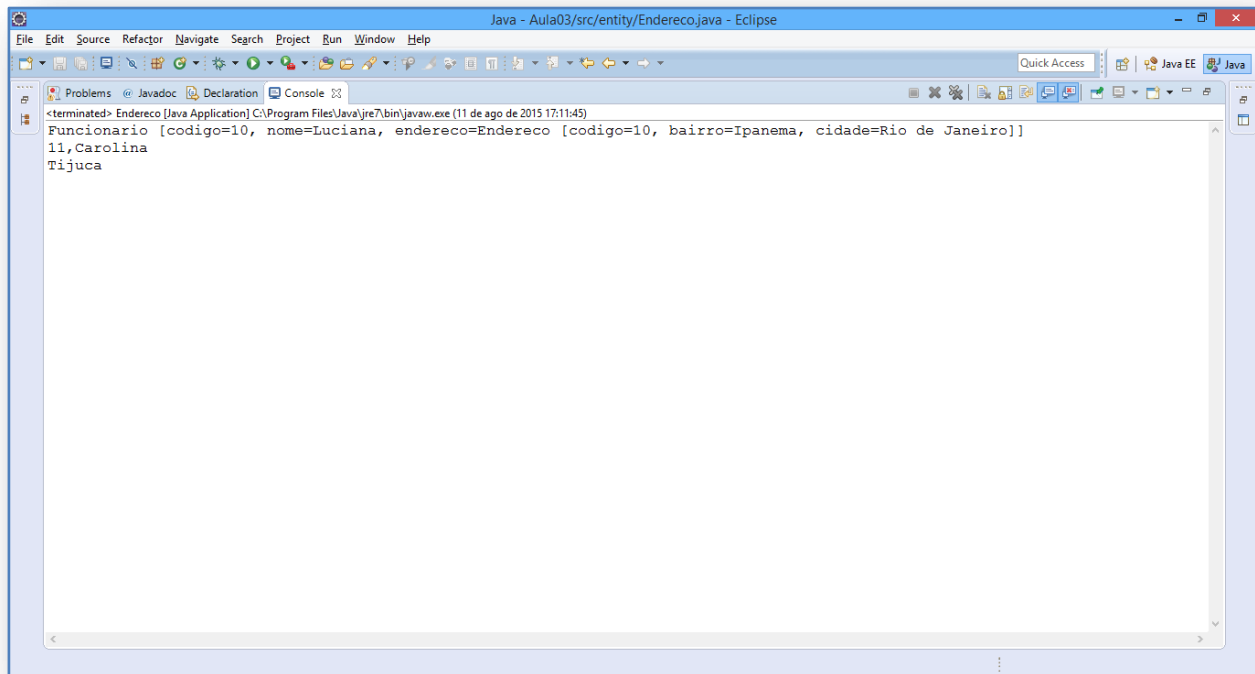
```
package entity;  
  
public class Funcionario {  
  
    private Integer codigo;  
    private String nome;  
  
    private Endereco endereco;  
  
    public Funcionario() {  
        // TODO Auto-generated constructor stub  
    }  
  
    public Funcionario(Integer codigo, String nome) {  
        super();  
        this.codigo = codigo;  
        this.nome = nome;  
    }  
  
    public Funcionario(Integer codigo, String nome, Endereco  
endereco) {  
        super();  
        this.codigo = codigo;  
        this.nome = nome;  
        this.endereco = endereco;  
    }  
  
    @Override  
    public String toString() {  
        return "Funcionario [codigo=" + codigo + ", nome=" +  
nome + ", endereco=" + endereco + "];"  
    }  
  
    public Integer getCodigo() {  
        return codigo;  
    }  
    public void setCodigo(Integer codigo) {
```

```
        this.codigo = codigo;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public Endereco getEndereco() {
        return endereco;
    }
    public void setEndereco(Endereco endereco) {
        this.endereco = endereco;
    }
}
```

Para testar a classe Funcionario fazer um método main.

```
    public static void main(String[] args) {
        Funcionario f1 = new Funcionario(10, "Luciana", new
Endereco(10, "Ipanema", "Rio de Janeiro"));
        Funcionario f2 = new Funcionario();
        f2.setCodigo(11);
        f2.setNome("Carolina");
        f2.setEndereco(new Endereco(11, "Tijuca", "Rio de
Janeiro"));
        System.out.println(f1);
        System.out.println(f2.getCodigo() + "," + f2.getNome() );
        System.out.println(f2.getEndereco().getBairro());
    }
```

Criamos o objeto do funcionário "f1" e passamos os dados através do construtor cheio. Depois criamos o funcionário "f2" e usamos o construtor vazio, passando os dados através do "set" de cada atributo. Pedimos a impressão do "f1" completa, através do toString e "f2" imprimiremos somente o que foi pedido (código, nome) e (bairro). A impressão no console fica da seguinte forma:



```
Java - Aula03/src/entity/Endereco.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java EE Java
Problems Javadoc Declaration Console
<terminated> Endereco [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (11 de ago de 2015 17:11:45)
Funcionario [codigo=10, nome=Luciana, endereco=Endereco [codigo=10, bairro=Ipanema, cidade=Rio de Janeiro]]
11,Carolina
Tijuca
```

COTI
Infor mática

Escola de Nerds

www.cotiinformatica.com.br