



C125 – Programação Orientada a Objetos com  
Java

# Construtores e o Modificador Static

Prof. Phyllipe Lima  
phyllipe@inatel.br



1



## Agenda



- ☕ Entender os Construtores
- ☕ Utilizar membros e métodos estáticos

2



## Construtores



☕ Já sabemos que a palavra chave “new” invoca o construtor de uma classe e cria a sua instância!

```
public class Pessoa {

    private String nome;
    private int idade;

    //Construtor
    public Pessoa() {
        System.out.println("Criando Instância de Pessoa");
    }
}
```

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

3

3



## Construtores



☕ Quando não declaramos o construtor explicitamente, o Java invoca o construtor implícito. Porém, se passamos a deixar um construtor visível, o padrão não é mais fornecido.

```
public class Pessoa {

    private String nome;
    private int idade;

    //Construtor
    //Não existe mais o construtor implícito agora
    public Pessoa() {
        System.out.println("Criando Instância de Pessoa");
    }
}
```

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

4

4



## Construtores



☕ Os construtores podem receber parâmetros, facilitando a inicialização de instâncias.

```
public class Pessoa {
```

```
    private String nome;
    private int idade;
```

```
    //Construtor recebendo parametros
    //E inicializando os membros da classe
    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }
}
```

```
public class Main {
```

```
    public static void main(String args[]) {
        Pessoa p = new Pessoa("Capiroto", 54);
    }
}
```

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

5

5



## Construtores



☕ Perceba que no exemplo anterior, o construtor força o cliente a passar dois parâmetros durante a instanciação. Não é mais possível invocar o construtor vazio!

```
public class Main {
```

```
    public static void main(String args[]) {
```

```
        //Não compila
        Pessoa p = new Pessoa();
    }
```

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

6

6

## Construtores



- ☕ Mas e se ainda desejamos oferecer uma opção de construtor vazio?
- ☕ Podemos fazer a sobrecarga de construtores. Basta termos parâmetros diferentes!
- ☕ O Compilador sabe qual construtor invocar, desde que os parâmetros sejam diferentes

## Construtores

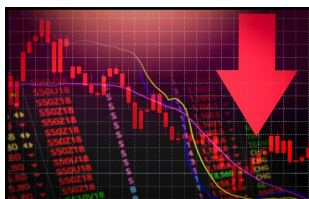


```
//Construtor recebendo parametros
public Pessoa(String nome, int idade) {
    this.nome = nome;
    this.idade = idade;
}

//Construtor Vazio
public Pessoa() {
}

//Recebendo apenas o nome
public Pessoa(String nome) {
    this.nome = nome;
}

public static void main(String args[]) {
    //Invoca Construtor Vazio (sem parametros)
    Pessoa p = new Pessoa();
    //Invoca Construtor com um parâmetro String
    Pessoa p1 = new Pessoa("Capiroto");
    //Invoca Construtor com dois parâmetros
    Pessoa p2 = new Pessoa("Tinhoso", 56);
}
```



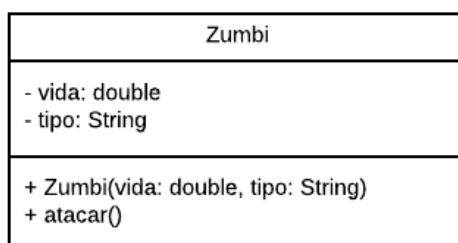
## Exercício 1 – Circuit Breaker



☕ Continuando a combater a crise econômica, agora os bancos querem que você adeque as classes Conta e Cliente com os seguintes construtores

- ☕ Para a Classe Cliente crie um construtor que já receba todos os parâmetros. Crie também um construtor vazio
- ☕ Para a Classe Conta crie um construtor que receba o número, saldo e um *array* de Clientes. Dentro desse construtor coloque um limite inicial de R\$ 350,00.

## UML – Construtor e Private



- ☕ O símbolo (-) denota que é um membro/método **private**
- ☕ O construtor é denotado pelo próprio nome da classe
- ☕ Geralmente os **getters** and **setters** são omitidos

## Exercício 2 – Zombicide



☕ Resgatando os velhos tempos 😊



C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

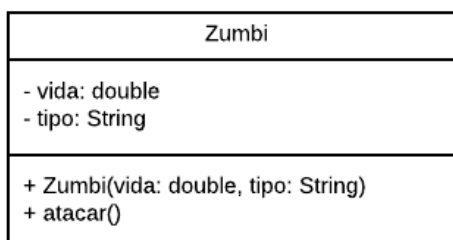
11

11

## Exercício 2 – Zombicide, o Retorno



☕ Crie uma classe Zumbi de acordo com o UML abaixo. Crie *getters* e *setters*. Em seguida teste seu funcionamento!



C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

12

12

## Modificador *Static*



☕ Imagine que estamos modelando um jogo estilo Arkanoid, e queremos saber quantos blocos existem no jogo



C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

13

13

## Modificador *Static*



☕ Como podemos fazer isso? Precisamos saber quantos foram criados. Vamos colocar um contador dentro do construtor de uma classe Bloco.


☕ A cada nova instância faremos um incremento

☕ Será que funciona?


C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

14

14



## Modificador *Static*



```

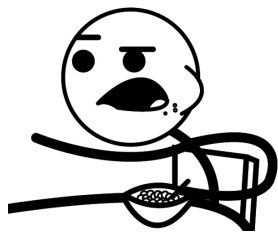
public class Bloco {
    private int numBlocos = 0;

    public Bloco() {
        numBlocos++;
    }
}

public static void main(String[] args) {
    Bloco bloco1 = new Bloco();
    Bloco bloco2 = new Bloco();


    System.out.println(bloco2.getNumBlocos());
}

```




C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

15



## Modificador *Static*



```

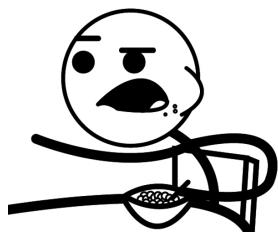
public class Bloco {
    private int numBlocos = 0;

    public Bloco() {
        numBlocos++;
    }
}

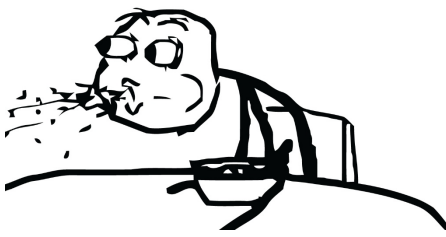
public static void main(String[] args) {
    Bloco bloco1 = new Bloco();
    Bloco bloco2 = new Bloco();

    System.out.println(bloco2.getNumBlocos());
}

```



1



C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

16

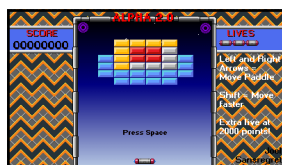




## Modificador *Static*



- ☕ Cada instância tem o seu próprio numBlocos, assim esse número sempre será incrementando uma única vez.
- ☕ Por isso eles são chamados membros da instância.
- ☕ Precisamos de algo compartilhado entre as instâncias, ou seja, algo que pertença a classe e não a instância.
- ☕ Vamos utilizar o modificador ***static***



## Modificador *Static*



```
public class Bloco {
    private static int numBlocos = 0;

    public Bloco() {
        numBlocos++;
    }

    public static int getNumBlocos() {
        return numBlocos;
    }
}
```

☕ Qualquer instância da classe Bloco compartilha a mesma variável numBlocos. Essa variável pertence a classe.

☕ O acesso é dado via classe



## Modificador *Static*



```
public class Main {

    public static void main(String[] args) {
        Bloco bloco1 = new Bloco();
        Bloco bloco2 = new Bloco();

        System.out.println(Bloco.getNumBlocos());
    }
}
```

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

19

19



## Modificador *Static*



```
public class Main {

    public static void main(String[] args) {
        Bloco bloco1 = new Bloco();
        Bloco bloco2 = new Bloco();

        System.out.println(Bloco.getNumBlocos());
    }
}
```

2

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

20

20



## Modificador *Static*



E se esse membro não fosse ***private***?

```
public class Bloco {
```

```
    static int numBlocos = 0;
```

```
    public Bloco() {
        numBlocos++;
    }
```

```
public class Main {
```

```
    public static void main(String[] args) {
        Bloco bloco1 = new Bloco();
        Bloco bloco2 = new Bloco();
```

```
        System.out.println(Bloco.numBlocos);
```

```
    }
```

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

21

21



## Modificador *Static*



E se esse membro não fosse ***private***?

```
public class Bloco {
```

```
    static int numBlocos = 0;
```

```
    public Bloco() {
        numBlocos++;
    }
```

```
public class Main {
```

```
    public static void main(String[] args) {
        Bloco bloco1 = new Bloco();
        Bloco bloco2 = new Bloco();
```

```
        System.out.println(Bloco.numBlocos);
```

```
    }
```

2

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

22

22



## Exercício 3 - Arkanoid



- ☕ Simule um jogo de Arkanoid. Construa duas classes, Jogador e Bloco.
- ☕ A cada novo bloco, incremente o número de blocos em jogo
- ☕ A cada bloco destruído, decmente o número de blocos em jogo e aumente a pontuação do Jogador.

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

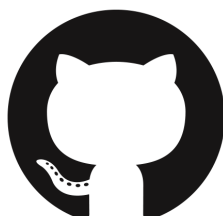
23

23

## Resolução dos Exercícios




<https://github.com/phillima-inatel/C125>




C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES


24


24






## Material Complementar



 Vídeo aula sobre variáveis estáticas (static)  
<https://youtu.be/qiufwvPDkts>

 Capítulo 5 da apostila FJ-11

-  Modificadores de Acesso e **Atributos**(Membros) de Classe
-  A partir do item 5.4



C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

25

25



# Inatel

C125 – Programação Orientada a Objetos com Java



## Construtores e o Modificador Static

Prof. Phyllipe Lima  
phyllipe@inatel.br

26