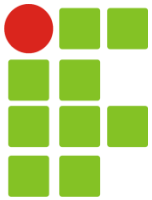


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS JOÃO CÂMARA

PROGRAMAÇÃO ESTRUTURADA E ORIENTADA A OBJETOS - ORIENTAÇÃO A OBJETOS

Nickerson Fonseca Ferreira
nickerson.ferreira@ifrn.edu.br

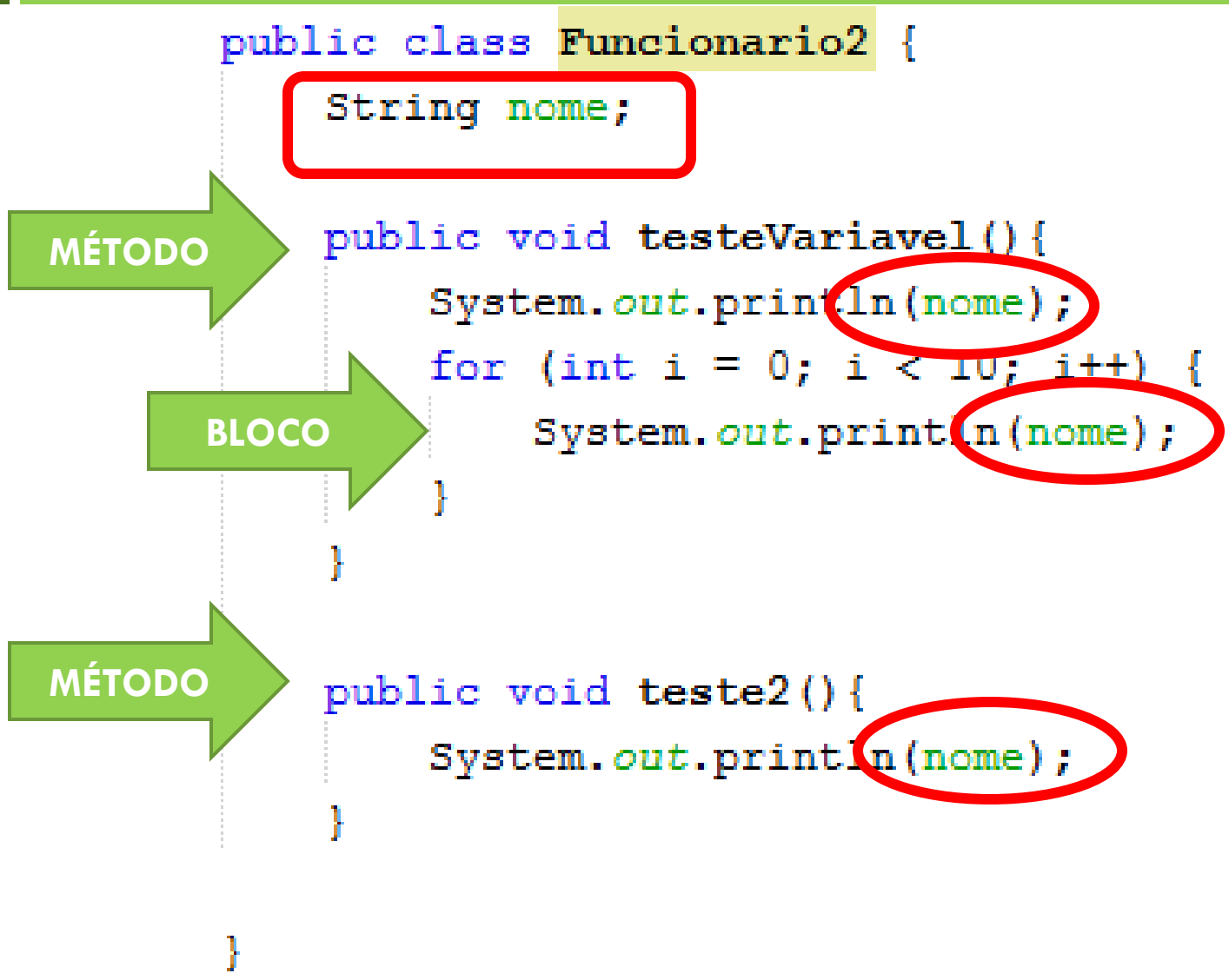
Escopo de variável

- Cada variável possui um escopo.
- Um escopo é a área que a variável é acessível.
- Existem 2 escopos básicos:
  - Global
  - Local
    - Método
    - Bloco

Escopo Global

- Uma variável com escopo global pode ser visualizada/acessada em qualquer ponto da classe.
- São os atributos da classe.

Escopo Global



Escopo Local (Método)

- Uma variável com escopo local (método) pode ser visualizada/acessada somente no método em que foi declarada.

Escopo Local (Método)

MÉTODO

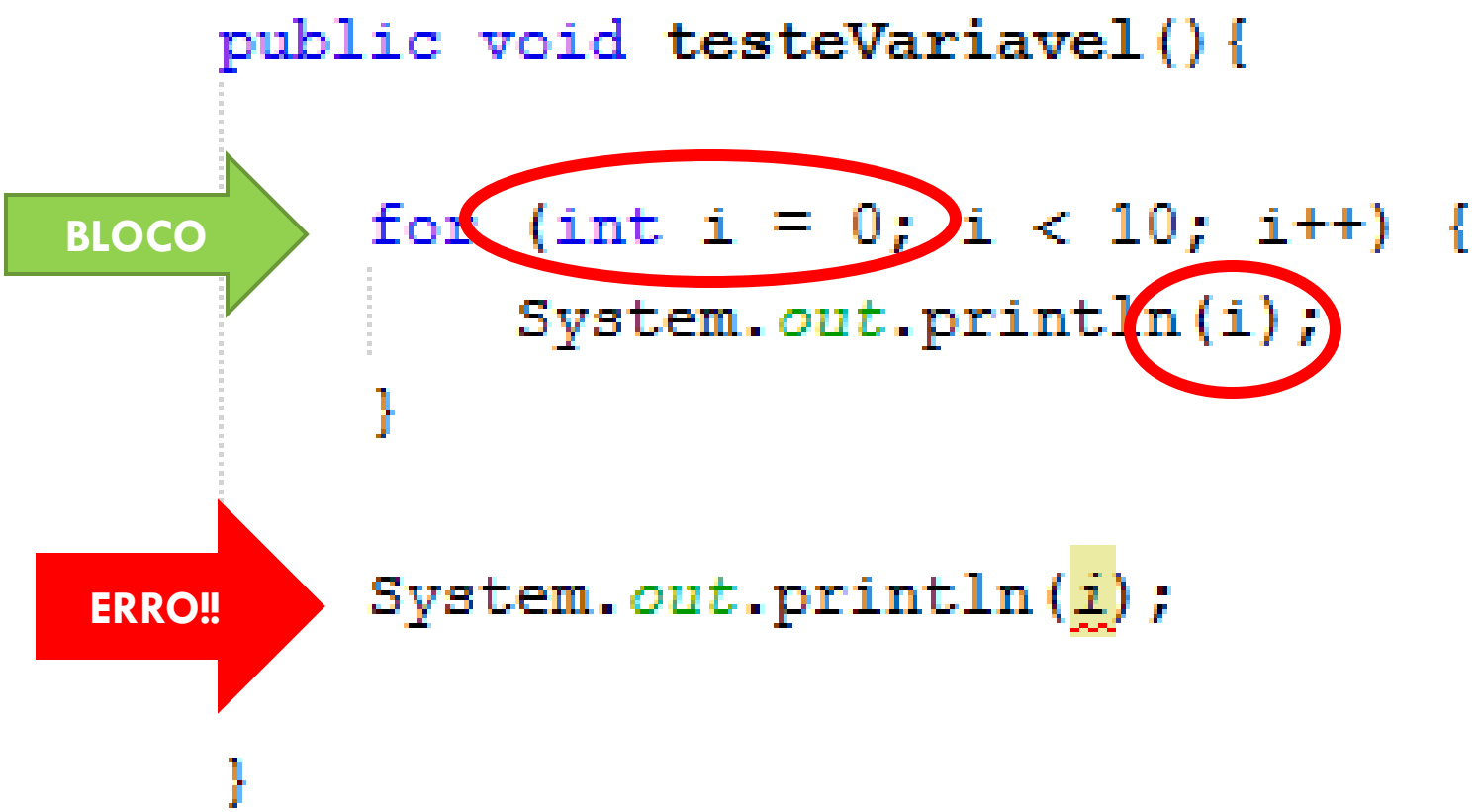
```
public void testeVariavel(){
    String teste = "Testando";
    System.out.println(teste);

    for (int i = 0; i < 10; i++) {
        System.out.println(teste);
    }
}

public void teste2(){
    System.out.println(teste);
}
```

ERRO!!

Escopo Local (Bloco)

- Uma variável com escopo local (bloco) pode ser visualizada/acessada somente no bloco em que foi declarada.

Escopo Local (Bloco)



Encapsulamento

- O encapsulamento serve para esconder os atributos de uma classe.
- Centralizar a forma de acesso aos atributos.
- Por centralizar a forma de acesso, nos ajuda no desenvolvimento de sistemas fáceis de realizar mudanças.
- Como usá-lo ???
  - Primeiramente colocamos os atributos da classe como privados (**private**)
  - Depois criamos um método público (**public**) para centralizar o acesso ao atributo.

Encapsulamento

- Vamos pensar numa classe Cliente:
  - Ela possui o atributo CPF, entre outros.
  - Necessita de um método para mudar o CPF.

```
class Cliente {
  private String nome;
  private String endereco;
  private String cpf;
  private int idade;

  public void mudaCPF(String cpf) {
    this.cpf = cpf;
  }
}
```

Encapsulamento

- Surge a necessidade de, antes de mudar o CPF, validá-lo.
- E ai

```
class Cliente {
  private String nome;
  private String endereco;
  private String cpf;
  private int idade;

  public void mudaCPF(String cpf) {
    validaCPF(cpf);
    this.cpf = cpf;
  }

  private void validaCPF(String cpf) {
    // série de regras aqui, falha caso não seja válido
  }
}
```

Métodos Get e Set

- Ao colocar um atributo como privado é necessário criar formas de acessá-lo.
- Tanto para recuperar quanto para alterar seu valor.
- Para isso, criamos os métodos Get e Set para esse(s) atributo(s).

```
class Conta {

  private double saldo;

  public double getSaldo() {
    return this.saldo;
  }

  public void setSaldo(double saldo) {
    this.saldo = saldo;
  }
}
```

Static

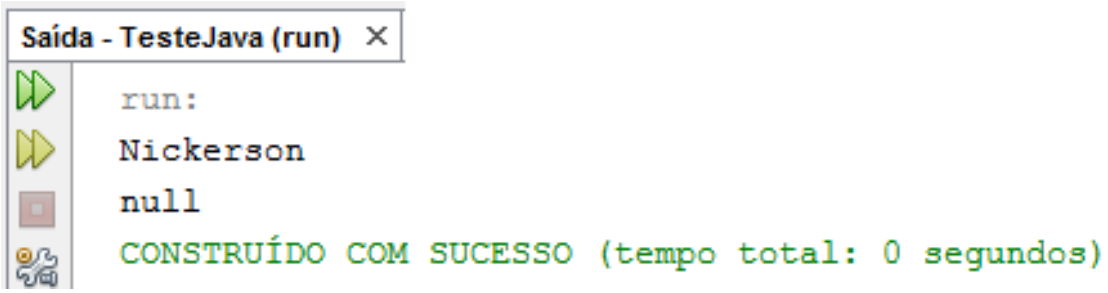
- Cada objeto possui sua própria cópia de todos os atributos da classe.
- Os membros declarados como static são conhecidos como membros de classe.
- Os membros de classe servem para compartilhar APENAS uma cópia de uma variável por todos os objetos.
- Para declarar um membro de classe identificamos com a palavra reservada static.
  - Ex: static String nome;

    private static String profissao;

Static


```java
public class Pessoa {

    String nome;
    int idade;
    String nacionalidade;
    String profissao;
}
```

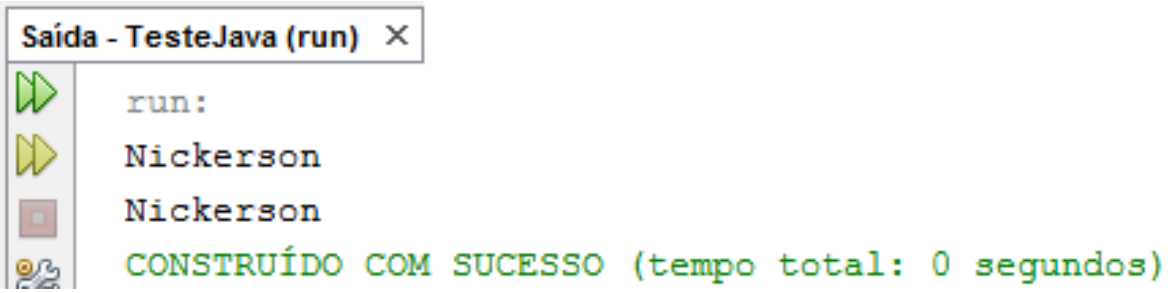
```java
public class TestePessoa {

    public static void main(String[] args) {
        Pessoa p1 = new Pessoa();
        p1.nome = "Nickerson";
        System.out.println(p1.nome);

        Pessoa p2 = new Pessoa();
        System.out.println(p2.nome);
    }

}
```



Static


```java
public class Pessoa {

    static String nome;
    int idade;
    String nacionalidade;
    String profissao;


}
```

```java
public class TestePessoa {

    public static void main(String[] args) {
        Pessoa p1 = new Pessoa();
        p1.nome = "Nickerson";
        System.out.println(p1.nome);

        Pessoa p2 = new Pessoa();
        System.out.println(p2.nome);
    }

}
```



Static

- Podemos utilizar também em métodos.
- Como se tornarão membros de classe, não é necessário instanciar a classe para chamar o método.
- Ex: Math.random();

```java
public class Pessoa {

    static String nome;
    int idade;
    String nacionalidade;
    String profissao;

    public static void testarMetodoStatic(){

        System.out.println("Testando o método estático...");

    }

}
```

```java
public class TestePessoa {

    public static void main(String[] args) {

        Pessoa.testarMetodoStatic();

    }

}
```

Referências

- **Apostila Caelum:** https://www.caelum.com.br/apostila-java-orientacao-objetos/orientacao-a-objetos-basica
- H.M. Deitel, P.J. Deitel, **Java Como programar.**