Orientação a Objetos 1

Tipos de Métodos

Prof. MSc. Vinícius Camargo Andrade

vcandrade@utfpr.edu.br

Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná

Tipos de Métodos

Tipos de Métodos

- *Não retornam valor e não recebem parâmetros;*
- *Não retornam valor mas recebem parâmetros;*
- *Retornam um valor e não recebem parâmetros;*
- *Retornam um valor e recebem parâmetros.*

Não Retornam Valor e Não Recebem Parâmetros

Tipos de Métodos

Não Retornam valor e Não Recebem Parâmetros

Servem para executar uma tarefa completa sem gerar uma dependência externa, como apresentar mensagens, obter dados via teclado, operar sobre variáveis de classe, etc.

Não Retornam valor e Não Recebem Parâmetros

```
public void imprimirDados() {

    System.out.println("Código: " + codigo);
    System.out.println("Nome: " + nome);
}
```

Não Retornam Valor e Recebem Parâmetros

Tipos de Métodos

Não Retornam valor e Recebem Parâmetros

Servem para executar uma tarefa completa sem gerar uma dependência externa, como apresentar mensagens, obter dados via teclado, operar sobre variáveis de classe, etc.

Não Retornam valor e Recebem Parâmetros

Dependem de valores externos (parâmetros) para execução das tarefas

Não Retornam valor e Recebem Parâmetros

```java
public void somar(int a, int b) {
    int resultado = a + b;
    System.out.println("Resultado Soma: " + resultado);
}
```

Retornam Valor e Não Recebem Parâmetros

Tipos de Métodos

Retornam valor e Não Recebem Parâmetros

Servem para executar uma tarefa completa com uma dependência externa (retorno), como informar o valor de determinado atributo;

Retornam valor e Não Recebem Parâmetros

Deve ser declarado o tipo de retorno antes do nome do método;

Retornam valor e Não Recebem Parâmetros

Deve conter a palavra reservada return.

Retornam valor e Não Recebem Parâmetros

```
public int buscarNumeroAgencia() {

    return agencia;
}
```

Retornam Valor e Recebem Parâmetros

Tipos de Métodos

Retornam Valor e Recebem Parâmetros

Servem para executar uma tarefa completa com uma dependência externa (retorno), como informar o valor de determinado atributo

Retornam Valor e Recebem Parâmetros

Deve ser declarado o tipo de retorno antes do nome do método

Retornam Valor e Recebem Parâmetros

*Deve conter a palavra reservada **return**.*

Retornam Valor e Recebem Parâmetros

Dependem de valores externos (parâmetros) para execução das tarefas

Retornam Valor e Recebem Parâmetros

```
public int somar(int a, int b) {

    int resultado = a + b;
    return resultado;
}
```

Invocação de Métodos

Invocação de Métodos

Um mesmo método pode ser invocado quantas vezes forem necessárias.

Invocação de Métodos

Cada objeto conterá os seus próprios métodos, e um não interfere em outros objetos, mesmo que sejam instâncias de mesma classe.

Invocação de Métodos

A invocação de um determinado método é realizada por meio de um ponto (.)

```
nomeObjeto.método();
```

Implementação

```java
public class Carro {

    String modelo;
    String marca;
    int ano;
    String placa;
    String cor;

    public void ligar() {

        System.out.println(modelo + " está ligando.");
    }

    public void desligar() {

        System.out.println(modelo + " está desligando.");
    }

    public void acelerar() {

        System.out.println(modelo + " está acelerando.");
    }

    public void frear() {

        System.out.println(modelo + " está freando.");
    }
}
```

```java
public class CarroTeste {

    public static void main(String[] args) {

        Scanner entradaTeclado = new Scanner(System.in);

        Carro carro1 = new Carro();

        System.out.print("Informe o modelo do carro1: ");
        carro1.modelo = entradaTeclado.nextLine();

        System.out.print("Informe a marca do carro1: ");
        carro1.marca = entradaTeclado.nextLine();

        System.out.print("Informe o ano do carro1: ");
        carro1.ano = entradaTeclado.nextInt();

        System.out.print("Informe a placa do carro1: ");
        entradaTeclado.nextLine(); // limpeza de buffer
        carro1.placa = entradaTeclado.nextLine();

        System.out.print("Informe a cor do carro1: ");
        carro1.cor = entradaTeclado.nextLine();

        carro1.ligar();
        carro1.acelerar();
        carro1.frear();
        carro1.desligar();
    }
}
```

```
run:
Informe o modelo do carro1: Ferrari 458 Italia
Informe a marca do carro1: Ferrari
Informe o ano do carro1: 2015
Informe a placa do carro1: AAA-1234
Informe a cor do carro1: Vermelho
Ferrari 458 Italia está ligando.
Ferrari 458 Italia está acelerando.
Ferrari 458 Italia está freando.
Ferrari 458 Italia está desligando.
BUILD SUCCESSFUL (total time: 1 minute 31 seconds)
```

Exercício 1

Exercício 1

Considerando o exercício da classe Pessoa (aula anterior), implemente os métodos da classe.

- *andar( )*: *Ao invocar o método andar(), deve ser exibida a mensagem: "<nome da pessoa> está andando".*
- *correr( )*: *Ao invocar o método correr(), deve ser exibida a mensagem: "<nome da pessoa> está correndo".*
- *falar( )*: *Ao invocar o método falar(), deve ser exibida a mensagem: "<nome da pessoa> está falando".*

Método toString( )

Método toString( )

O objetivo do método toString( ) é apresentar uma representação textual de uma instância de um objeto.

Método toString( )

Essa representação textual é útil principalmente em situações de debugging e de logging.

Exemplo

Método toString( )

Exemplo

Observe o exemplo do carro, em que se inicia todos os atributos do objeto carro1 e, posteriormente, imprime todos os valores dos atributos modelo, marca, ano, placa e cor.

```java
public class Carro {

    String modelo;
    String marca;
    int ano;
    String placa;
    String cor;

    public void ligar() {
        System.out.println(modelo + " está ligando.");
    }

    public void desligar() {
        System.out.println(modelo + " está desligando.");
    }

    public void acelerar() {
        System.out.println(modelo + " está acelerando.");
    }

    public void frear() {
        System.out.println(modelo + " está freando.");
    }
}
```

```java
public class CarroTeste {

    public static void main(String[] args) {

        Carro carro1 = new Carro();

        carro1.modelo = "Ferrari 458 Italia";
        carro1.marca = "Ferrari";
        carro1.ano = 2015;
        carro1.placa = "ABC-1234";
        carro1.cor = "Vermelho";

        System.out.println("Modelo: " + carro1.modelo);
        System.out.println("Marca: " + carro1.marca);
        System.out.println("Ano: " + carro1.ano);
        System.out.println("Placa: " + carro1.placa);
        System.out.println("Cor: " + carro1.cor);
    }
}
```

```
run:
Modelo: Ferrari 458 Italia
Marca: Ferrari
Ano: 2015
Placa: ABC-1234
Cor: Vermelho
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
public class Carro {

    String modelo;
    String marca;
    int ano;
    String placa;
    String cor;

    public void ligar() {
        System.out.println(modelo + " está ligando.");
    }

    public void desligar() {
        System.out.println(modelo + " está desligando.");
    }

    public void acelerar() {
        System.out.println(modelo + " está acelerando.");
    }

    public void frear() {
        System.out.println(modelo + " está freando.");
    }
}
```

```java
public class CarroTeste {

    public static void main(String[] args) {

        Carro carro1 = new Carro();

        carro1.modelo = "Ferrari 458 Italia";
        carro1.marca = "Ferrari";
        carro1.ano = 2015;
        carro1.placa = "ABC-1234";
        carro1.cor = "Vermelho";

        System.out.println("Modelo: " + carro1.modelo);
        System.out.println("Marca: " + carro1.marca);
        System.out.println("Ano: " + carro1.ano);
        System.out.println("Placa: " + carro1.placa);
        System.out.println("Cor: " + carro1.cor);
    }
}
```

```
run:
Modelo: Ferrari 458 Italia
Marca: Ferrari
Ano: 2015
Placa: ABC-1234
Cor: Vermelho
BUILD SUCCESSFUL (total time: 0 seconds)
```

Neste caso, seria mais intuitivo enviar apenas o objeto carro1 ao método println( ) para que este imprima os valores dos atributos de carro1.

```java
public class Carro {

    String modelo;
    String marca;
    int ano;
    String placa;
    String cor;

    public void ligar() {
        System.out.println(modelo + " está ligando.");
    }

    public void desligar() {
        System.out.println(modelo + " está desligando.");
    }

    public void acelerar() {
        System.out.println(modelo + " está acelerando.");
    }

    public void frear() {
        System.out.println(modelo + " está freando.");
    }
}
```

```java
public class CarroTeste {

    public static void main(String[] args) {

        Carro carro1 = new Carro();

        carro1.modelo = "Ferrari 458 Italia";
        carro1.marca = "Ferrari";
        carro1.ano = 2015;
        carro1.placa = "ABC-1234";
        carro1.cor = "Vermelho";

        System.out.println(carro1);
    }
}
```

```
run:
orientacaoobjeto.aula02.exemplo02.Carro@1d81eb93
BUILD SUCCESSFUL (total time: 0 seconds)
```

Porém, como saída, tem-se o endereço de memória que o objeto carro1 está alocado, e não os valores de seus atributos.

```java
public class Carro {

    String modelo;
    String marca;
    int ano;
    String placa;
    String cor;

    public void ligar() {
        System.out.println(modelo + " está ligando.");
    }

    public void desligar() {
        System.out.println(modelo + " está desligando.");
    }

    public void acelerar() {
        System.out.println(modelo + " está acelerando.");
    }

    public void frear() {
        System.out.println(modelo + " está freando.");
    }
}
```

```java
public class CarroTeste {

    public static void main(String[] args) {

        Carro carro1 = new Carro();

        carro1.modelo = "Ferrari 458 Italia";
        carro1.marca = "Ferrari";
        carro1.ano = 2015;
        carro1.placa = "ABC-1234";
        carro1.cor = "Vermelho";

        System.out.println(carro1);
    }
}
```

```
run:
orientacaoobjeto.aula02.exemplo02.Carro@1d81eb93
BUILD SUCCESSFUL (total time: 0 seconds)
```

Para isso, implementa-se o método toString que retorna uma String. Neste caso, concatenando todos os atributos de Carro.

```java
public class Carro {

    String modelo;
    String marca;
    int ano;
    String placa;
    String cor;

    public void ligar() {
        System.out.println(modelo + " está ligando.");
    }

    public void desligar() {
        System.out.println(modelo + " está desligando.");
    }

    public void acelerar() {
        System.out.println(modelo + " está acelerando.");
    }

    public void frear() {
        System.out.println(modelo + " está freando.");
    }

    public String toString() {
        return "Modelo: " + modelo + "\n"
             + "Marca: " + marca + "\n"
             + "Ano: " + ano + "\n"
             + "Placa: " + placa + "\n"
             + "Cor: " + cor;
    }
}
```

```java
public class CarroTeste {

    public static void main(String[] args) {

        Carro carro1 = new Carro();

        carro1.modelo = "Ferrari 458 Italia";
        carro1.marca = "Ferrari";
        carro1.ano = 2015;
        carro1.placa = "ABC-1234";
        carro1.cor = "Vermelho";

        System.out.println(carro1);
    }
}
```

```
run:
Modelo: Ferrari 458 Italia
Marca: Ferrari
Ano: 2015
Placa: ABC-1234
Cor: Vermelho
BUILD SUCCESSFUL (total time: 0 seconds)
```

Neste caso, não há necessidade de invocar explicitamente o método toString( ), apenas envia-se o objeto ao método println( )

```java
public class Carro {

    String modelo;
    String marca;
    int ano;
    String placa;
    String cor;

    public void ligar() {
        System.out.println(modelo + " está ligando.");
    }

    public void desligar() {
        System.out.println(modelo + " está desligando.");
    }

    public void acelerar() {
        System.out.println(modelo + " está acelerando.");
    }

    public void frear() {
        System.out.println(modelo + " está freando.");
    }

    public String toString() {
        return "Modelo: " + modelo + "\n"
             + "Marca: " + marca + "\n"
             + "Ano: " + ano + "\n"
             + "Placa: " + placa + "\n"
             + "Cor: " + cor;
    }
}
```

```java
public class CarroTeste {

    public static void main(String[] args) {

        Carro carro1 = new Carro();

        carro1.modelo = "Ferrari 458 Italia";
        carro1.marca = "Ferrari";
        carro1.ano = 2015;
        carro1.placa = "ABC-1234";
        carro1.cor = "Vermelho";

        System.out.println(carro1);
    }
}
```

```
Modelo: Ferrari 458 Italia
Marca: Ferrari
Ano: 2015
Placa: ABC-1234
Cor: Vermelho
BUILD SUCCESSFUL (total time: 0 seconds)
```

Exercício 2

Exercício 2

Implemente o método toString( ) na classe Pessoa.

O método deve retornar a concatenação de todos os atributos da classe Pessoa.

Exercício 3

Exercício 3

Desenvolva um sistema que controla as ações de N carros. Sabe-se que cada carro possui um modelo, marca, ano, placa, cor, ligado e velocidade, que determina se o mesmo está ligado ou desligado. Além disso, o carro pode ligar, desligar, acelerar e frear.

Para criar novos carros, utilize uma classe de teste (CarroTeste.java).

Os atributos ligado e velocidade dessem obrigatoriamente ser iniciados com false e 0, respectivamente.

Exercício 3

Métodos:

- *ligar( ): atribuir valor true para a variável ligado se o carro estiver desligado;*
- *desligar( ): atribuir false para a variável ligado apenas se a velocidade for 0 e se o carro estiver ligado;*
- *acelerar( ): se o carro estiver ligado, aumentar a velocidade em 10 km/h. O carro não pode exceder a velocidade máxima de 200 km/h;*
- *frear( ): se o carro estiver ligado, diminuir a velocidade em 10 km/h. O carro não pode ficar com sua velocidade negativa.*