LINGUAGEM DE PROGRAMAÇÃO ORIENTADA À OBJETOS – LPOO

Aula 05 – Using Methods

Prof. Danilo Pereira - danilo.pereira1@docente.unip.br

TÓPICOS DA AULA

- **Aprender sobre métodos em Java**
  - *Identificar e entendendo as partes de um método*
  - *Adicionar parâmetros e retorno*

Livro Java Programming – Joyce Farrell (Chapter 3)

ENTENDENDO MÉTODOS EM JAVA

- Um método é um módulo de programa que contém uma série de instruções que realizam uma tarefa

- Já vimos que em algumas classes Java **podem ou não contêm um método main ( )**

- O método main( ) de um programa **pode executar métodos adicionais** e esses métodos podem executar outros. Qualquer classe pode conter um número ilimitado de métodos.

ENTENDENDO MÉTODOS EM JAVA

- Para executar um método, você o invoca ou faz a chamada do método

- Considere a primeira classe simples que criamos anteriormente; ele exibia uma única linha de saída: "Hello World"

- Suponha que você deseja adicionar três linhas de saída a este aplicativo para **exibir o nome e o endereço da sua empresa**.

Uma abordagem seria simplesmente adicionar três novas instruções println( ), conforme mostrado nas instruções sombreadas na Figura a seguir.

ENTENDENDO MÉTODOS EM JAVA

- **Em vez de adicionar as três instruções println ( ), você pode prefira chamar um método que execute as três instruções.**

```
public class First
{
   public static void main(String[] args)
   {
      System.out.println("XYZ Company");
      System.out.println("8900 U.S. Hwy 14");
      System.out.println("Crystal Lake, IL 60014");
      System.out.println("First Java application");
   }
}
```

Figure 3-1 The First class

- Então o programa se pareceria com aquele na Figura 3-1. A linha sombreada **contém a chamada para o método displayAddress ( ).**

```
public class First
{
   public static void main(String[] args)
   {
      displayAddress();
      System.out.println("First Java application");
   }
}
```

Figure 3-2 The First class with a call to the displayAddress() method

ENTENDENDO MÉTODOS EM JAVA

```
public class First
{
   public static void main(String[] args)
   {
      System.out.println("XYZ Company");
      System.out.println("8900 U.S. Hwy 14");
      System.out.println("Crystal Lake, IL 60014");
      System.out.println("First Java application");
   }
}
```

Figure 3-1 The `First` class

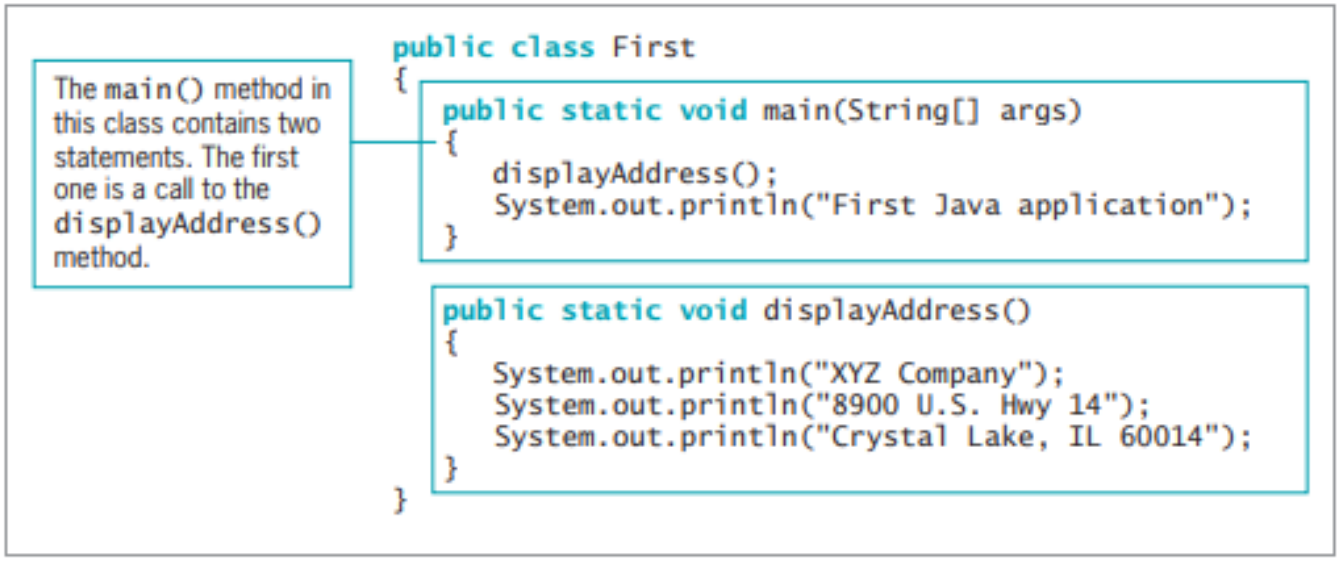


Figure 3-4 `First` class with `main()` calling `displayAddress()`

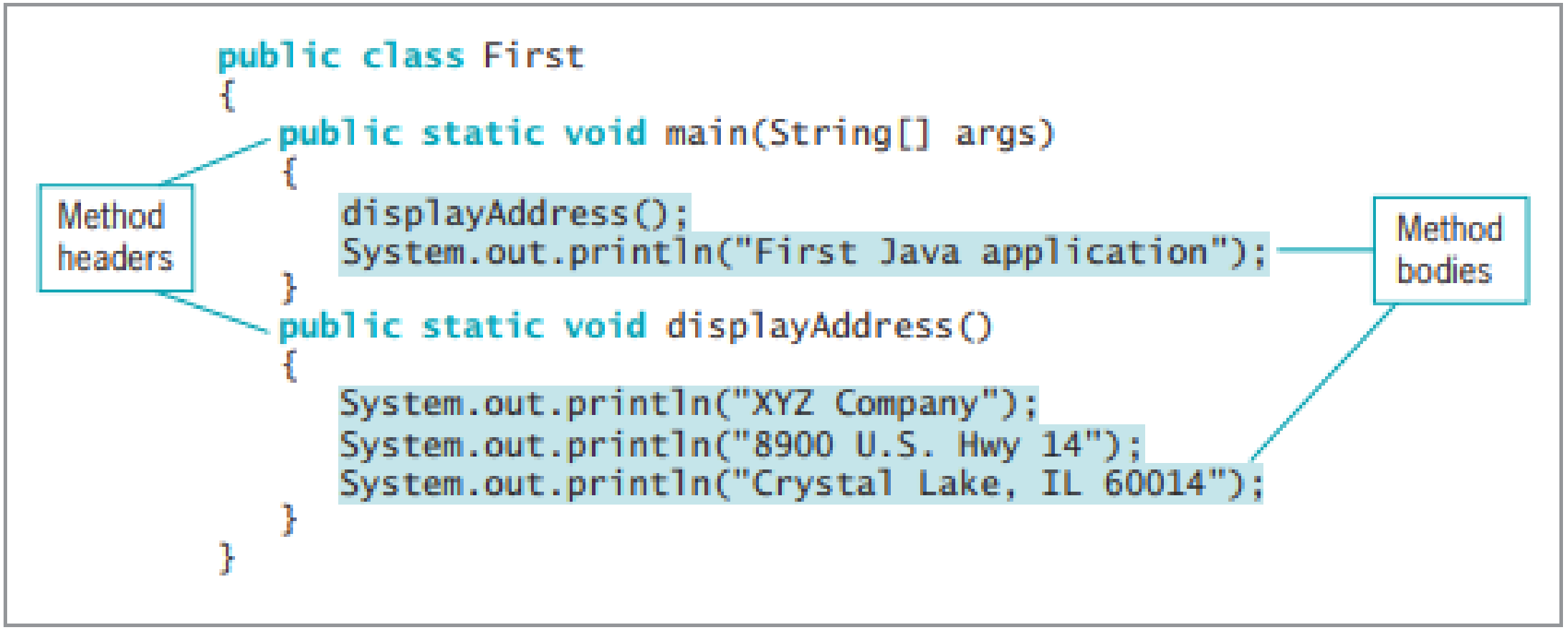
CRIANDO MÉTODOS EM JAVA

- Cada método deve incluir as **2 partes (cabeçalho e o corpo)** apresentadas na Figura 3-6:

  - **Cabeçalho (header)** - o cabeçalho de um método fornece informações sobre como outros métodos pode interagir com ele. ***Um cabeçalho de método também é chamado de declaração.***

  - **Corpo (body)** - O corpo do método contém o declarações que realizam o trabalho do método. ***O corpo de um método é chamado de implementação.***

CRIANDO MÉTODOS EM JAVA



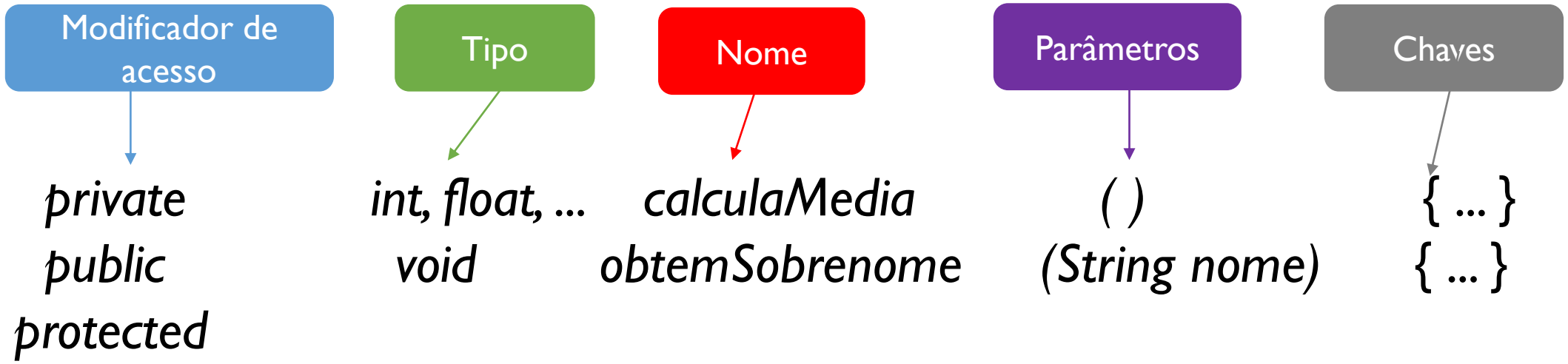

```
public class First
{
    public static void main(String[] args)
    {
        displayAddress();
        System.out.println("First Java application");
    }
    public static void displayAddress()
    {
        System.out.println("XYZ Company");
        System.out.println("8900 U.S. Hwy 14");
        System.out.println("Crystal Lake, IL 60014");
    }
}
```

Figure 3-6 The headers and bodies of the methods in the `First` class

ASSINATURA DO MÉTODO

O cabeçalho (assinatura) do método é a primeira linha de um método. Ele contém o seguinte:

- **Modificador de acesso:** public ou private:
- **Tipo do returno:** identifica o tipo de dados que a variável irá armazenar;
- **Nome:** começando com letra minúscula;
- **Parâmetros (opcional)**
- **Chave**



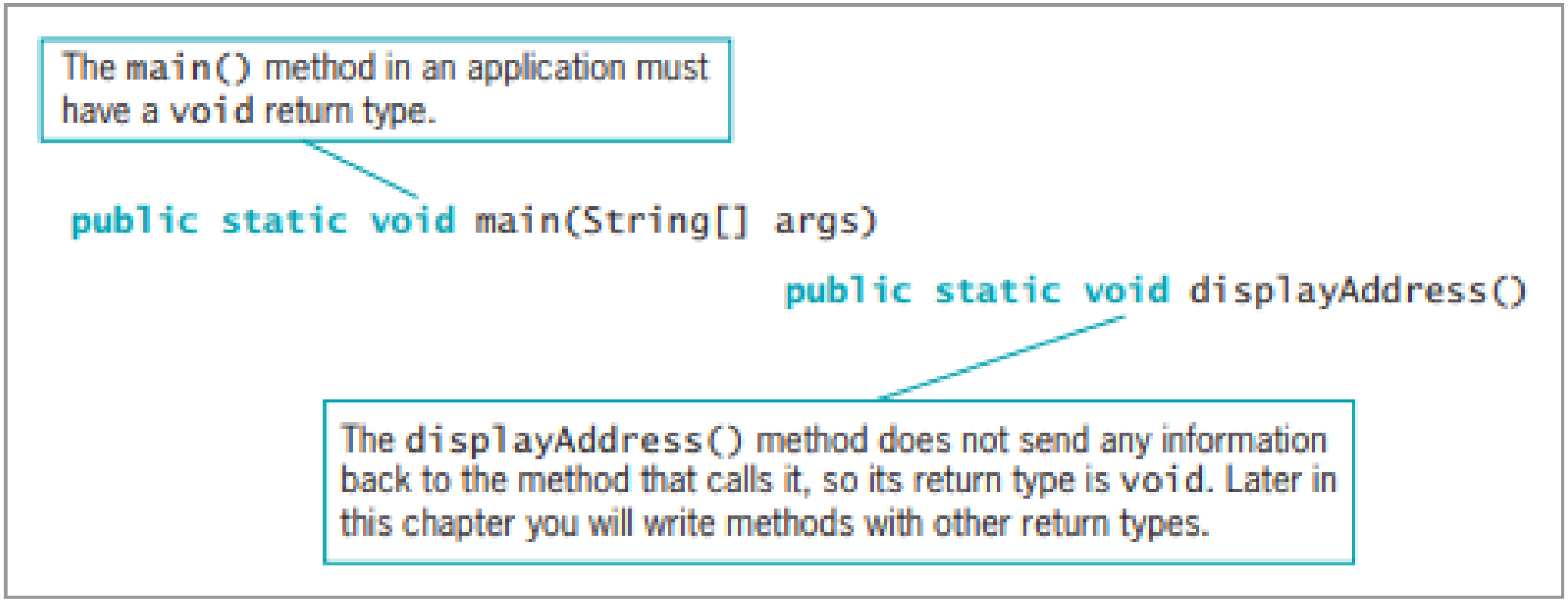
MET**Ó**DOS – TIPOS DE RETORNO

- Um tipo de retorno descreve o **tipo de dados que o método envia de volta para sua chamada método**

- Nem todos os métodos retornam um valor para seus métodos de chamada;

- Os métodos sem retorno tem um tipo de retorno de especifico chamada de **void**.

METÓDOS – TIPOS DE RETORNO



MET**Ó**DOS – PARÂMETROS

- Alguns métodos exigem que os dados sejam enviados a eles quando forem chamados.

- Os dados que você usa em um chamada a um método são chamados de **argumentos ou parâmetros**.

- Métodos que recebem parâmetros são flexíveis porque podem produzir diferentes resultados dependendo de quais dados eles recebem.

METÓDOS – PARÂMETROS

- Como um exemplo da vida real, quando você faz uma reserva em um restaurante, não precisamos criar um método diferente para cada data do ano em todas as horas possíveis do dia.

- Em vez disso, você pode fornecer a data e hora **como parâmetros,** para a pessoa que faz a chamada do método.

- O método que faz o registro da reserva, é então realizado da mesma maneira, não importa o que data e hora são fornecidas.

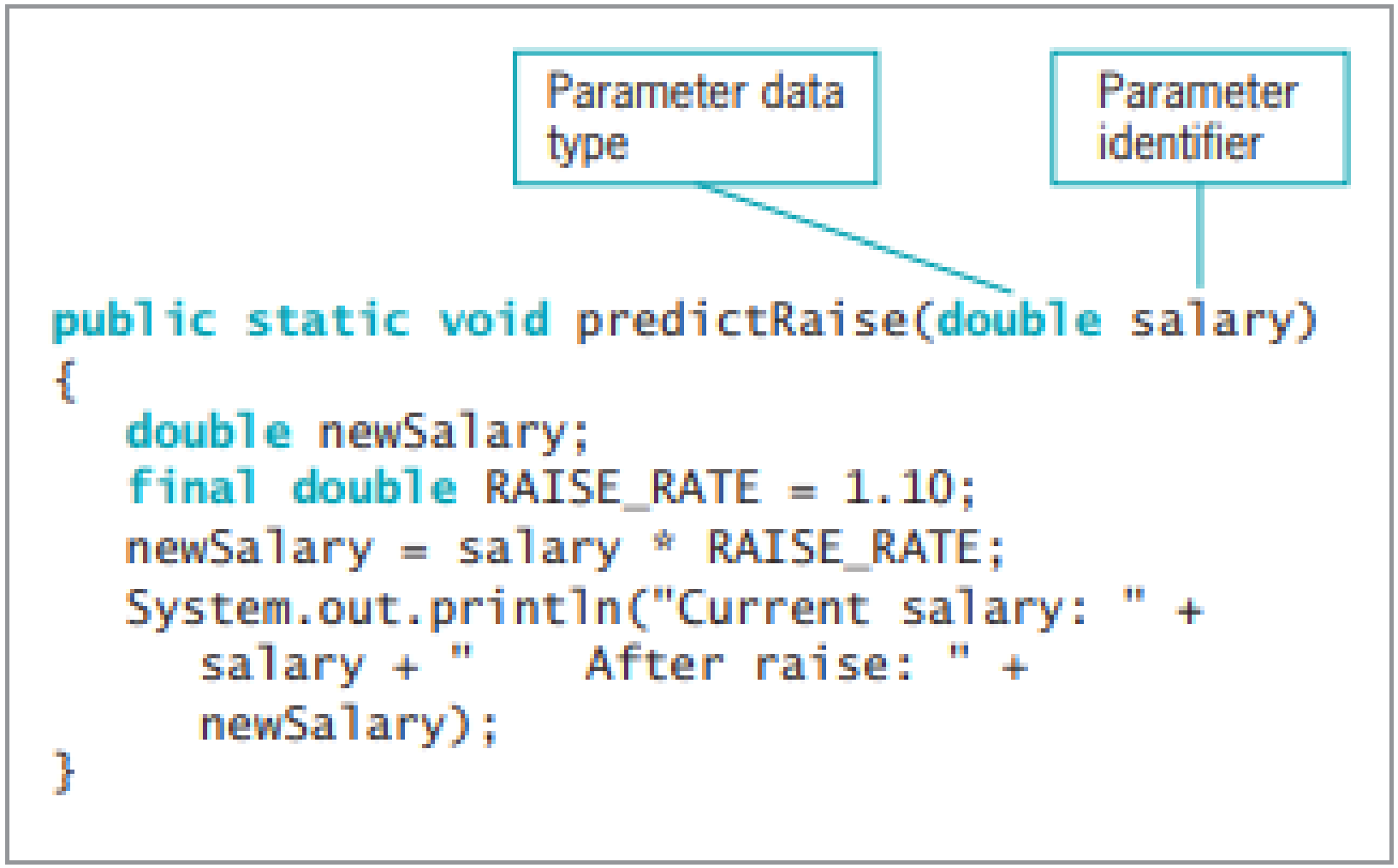
CRIANDO MÉTODO COM APENAS UM PARÂMETRO

- Quando um método pode receber um parâmetro, sua declaração contém os mesmos elementos que um que não aceita um parâmetro
  - modificadores de acesso (opcional)
  - o tipo de retorno para o método,
  - nome do método
  - conjunto de parênteses que inclui dois itens:
    - tipo do parâmetro e o nome

public static void predictRaise(double salary)

No cabeçalho do método para predictRaise (), o parâmetro salary indica que o método receberá um valor do tipo double, e que dentro método, o valor passado será conhecido como salary.

CRIANDO MÉTODO COM APENAS UM PARÂMETRO



PRATICANDO ...

1) Criar um método **sem retorno** chamado **formatarNome**, que recebe com parâmetro:

- nomeCompleto – String

O método será responsável em exibir na tela (System.out.println) o nome do usuário em letra maiúscula.

2) Criar um método **sem retorno** chamado **calculaQtdeCaracteres**, que recebe com parâmetro

- uma frase

O método será responsável em exibir na tela (Scanner ou System.out.println) a quantidade de caracteres da frase fornecedida.

METÓDOS – MULTIPLOS PARAMÊTROS

- Normalmente, vamos precisar criar métodos que devera receber diferentes parâmetros.

- Um método pode conter, nenhum, um ou mais parâmetros de diferentes tipos de dados .Veja os exemplos

public void exibeInformacoesUsuario ***( )*** { ... }

public void formataNome ***(String nome)*** { ... }

*public void cadastroUsuario****(String nome, int cpf, int rg, byte idade)*** { ... }

METÓDOS – CRIANDO MÉTODOS COM RETORNO

- Vimos ate agora, como podemos criar métodos sem retorno (void) e também receber um ou mais parâmetros ou até mesmo não receber nenhum parâmetro.

- Entretanto, muitas vezes precisamos fazer métodos que tem como a função retorna um informação no final da execução

- O tipo de retorno de um método pode ser qualquer tipo usado em Java, o que inclui os tipos primitivos e bem como tipos de classe (incluindo tipos de classe que você cria).

MÉTODOS – CRIANDO MÉTODOS COM RETORNO

- Vejamos alguns exemplo de métodos com retorno:

public ***String*** *buscaEnderecoCompleto(String cep) { ... }*

public ***int*** *calculaQtdeCaracteres (String frase) { ... }*

public ***Integer*** *calculaQtdeCaracteres (String frase) { ... }*

public ***float*** *calculaMediaPonderada (double nota1, double nota2, double peso) { ... }*

public ***double*** *calculaDesconto (double preco, double porcentagem) { ... }*

public ***boolean*** *validaCPF (String cfp) { ... }*

JAVA METHOD OVERLOADING

- Com a **sobrecarga (*overloading*) de método**, vários métodos podem ter ***o mesmo nome com parâmetros diferentes:***

Exemplo:

public int ***myMethod****(int x)*
public float ***myMethod****(float x)*
public double ***myMethod****(double x, double y)*

Vamos analisar alguns exemplos:

https://www.w3schools.com/java/java_methods_overloading.asp

MÉTODOS - RESUMO

- Como vimos anteriormente podemos criar métodos de diferentes maneiras:

- **Sem retorno (void) –** *public **void** calculaSalario( )*

- **Com retorno –** *public **double** calculaDesconto( )*

- **Sem parâmetros –** *public void imprimeRelatorio**( )***

- **Com parâmetros**
  - **Único:** *public int calculaIdade**(Data dataNascimento)***
  - **Multiplos:** *public boolean verificaValidade **(int idProduto, Data dataValidade)***