

Subprogramação

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br

Aula de hoje

- Estudaremos a estrutura mais básica de encapsulamento da Orientação a Objetos
 - Métodos

Exemplo

```
import java.util.Scanner;
public class IMC {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);

        System.out.print("Entre com a sua altura em metros: ");
        double altura = teclado.nextDouble();

        System.out.print("Entre com a sua massa em kg: ");
        double massa = teclado.nextDouble();

        double imc = massa / Math.pow(altura, 2);
        System.out.println("Seu IMC é " + imc);
    }
}
```

Parecidos!



```
System.out.print("Entre com a sua altura em metros: ");
double altura = teclado.nextDouble();
```

```
System.out.print("Entre com a sua massa em kg: ");
double massa = teclado.nextDouble();
```

```
double imc = massa / Math.pow(altura, 2);
System.out.println("Seu IMC é " + imc);
```

```
}
```

```
}
```

Exemplo usando método

```
import java.util.Scanner;
public class IMC {
```

```
    public static double leia(String mensagem) {
        Scanner teclado = new Scanner(System.in);
        System.out.print(mensagem);
        return teclado.nextDouble();
    }
```

*Declaração
do método*

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

```
        double altura = leia("Entre com a sua altura em metros: ");
        double massa = leia("Entre com a sua massa em kg: ");
```

```
        double imc = massa / Math.pow(altura, 2);
        System.out.println("Seu IMC é " + imc);
```

*Chamadas
ao método*

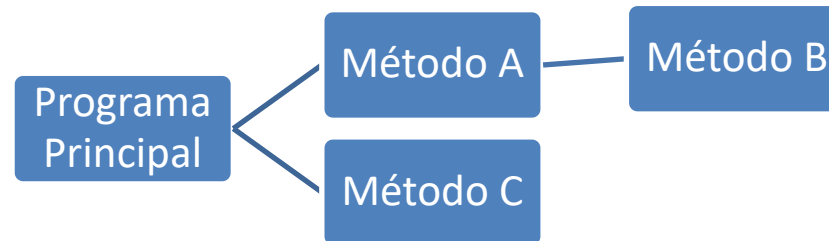
```
    }
}
```

Dividir para conquistar

- Antes: um programa gigante



- Depois: vários programas menores



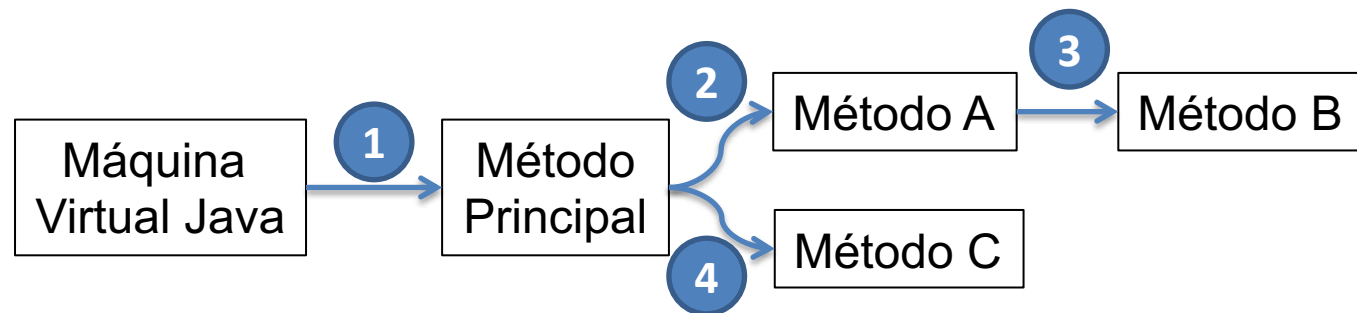
Fluxo de execução

- O programa tem início em um método principal (no caso do Java é o método *main*)
- O método principal chama outros métodos
- Estes métodos podem chamar outros métodos, sucessivamente
- Ao fim da execução de um método, o programa retorna para a instrução seguinte à da chamada ao método

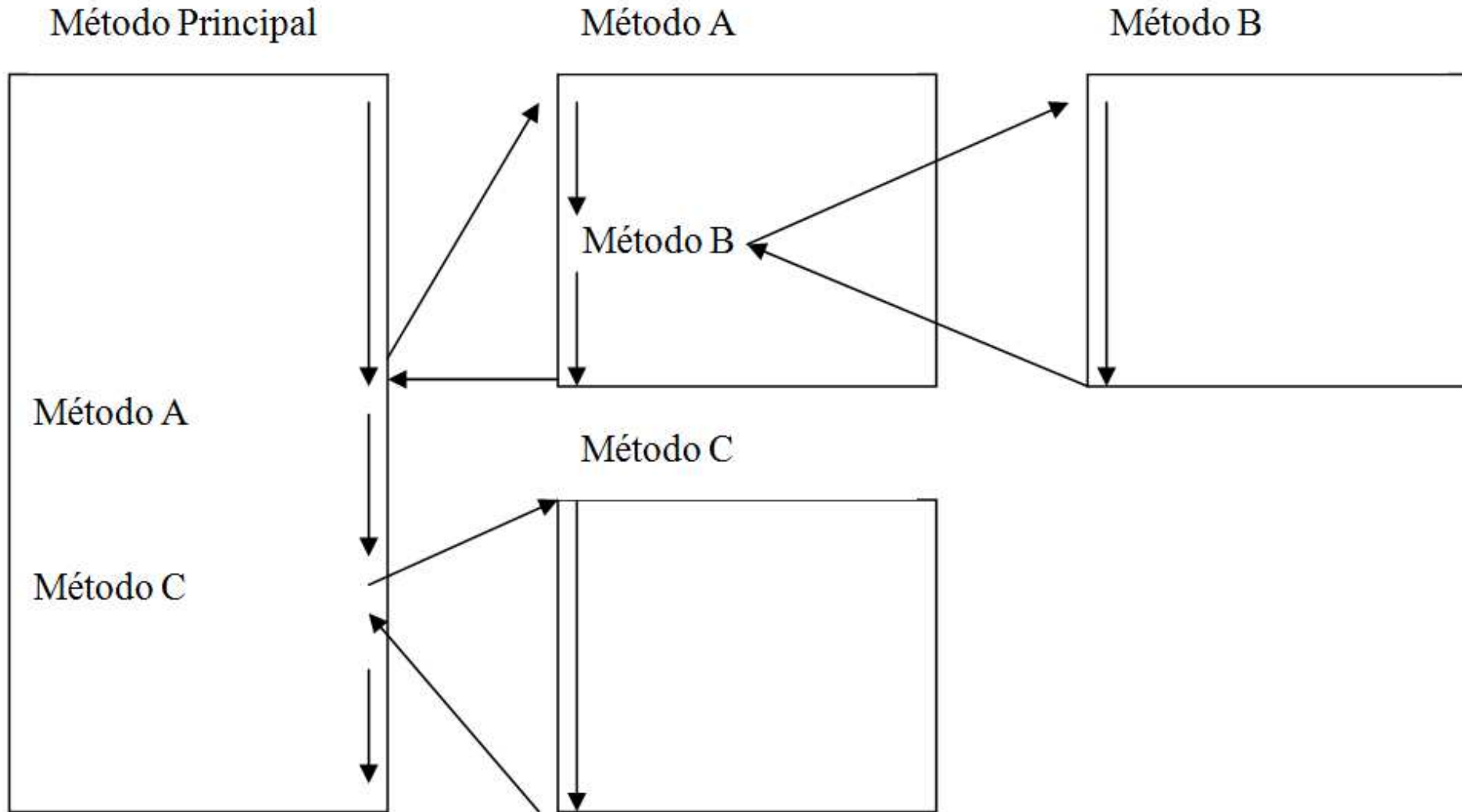
Programa

Método Principal
Método A
Método B
Método C

Possível sequencia de chamadas

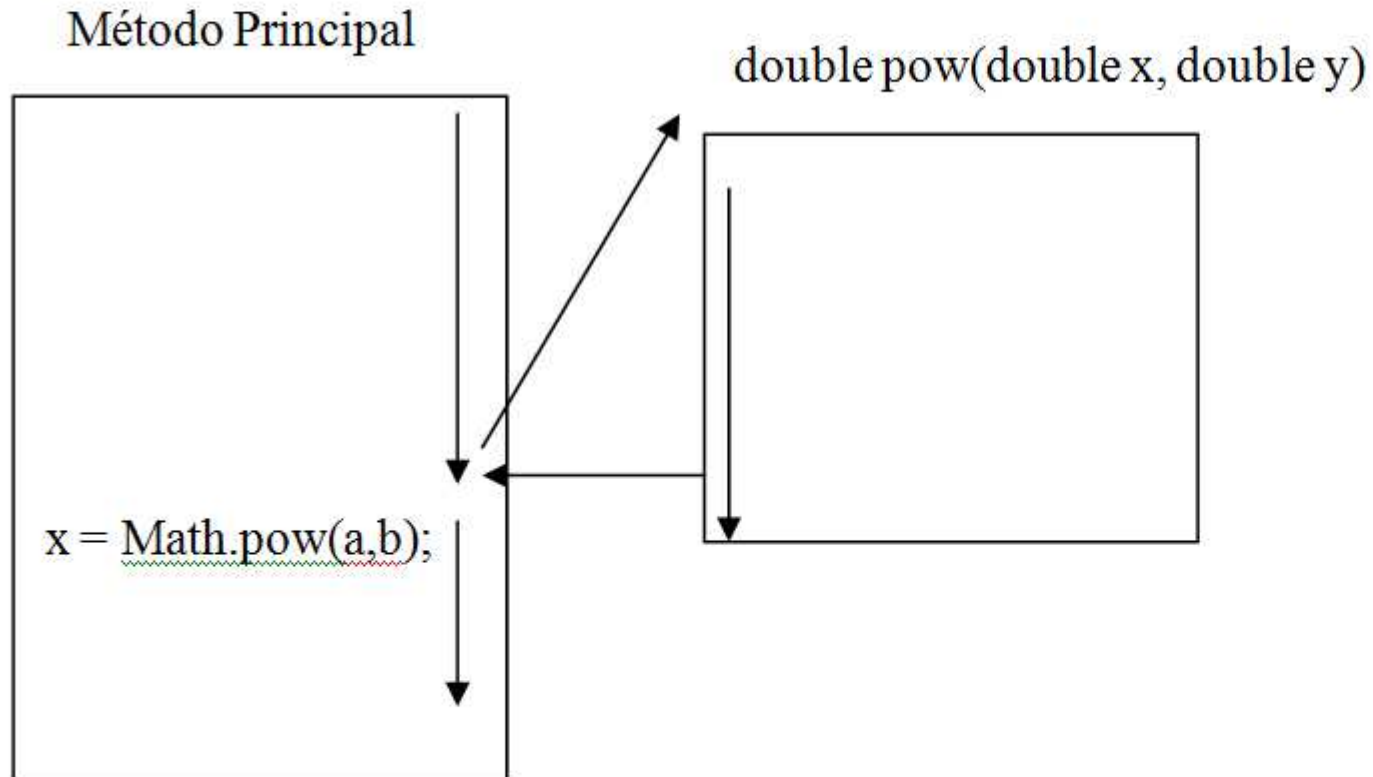


Fluxo de execução



Fluxo de execução

- É equivalente ao que acontece quando chamamos um método predefinido do Java



Vantagens

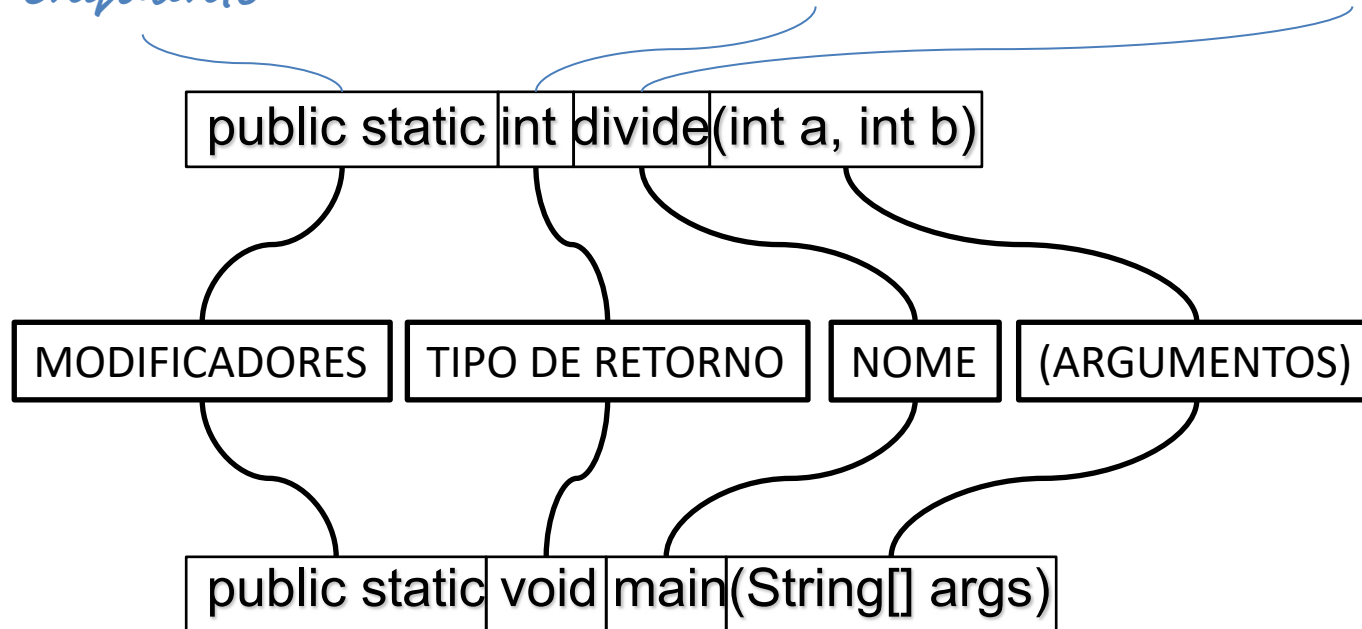
- Economia de código
 - Quanto mais repetição, mais economia
- Facilidade na correção de defeitos
 - Corrigir o defeito em um único local
- Legibilidade do código
 - Podemos dar nomes mais intuitivos a blocos de código
 - É como se criássemos nossos próprios comandos
- Melhor tratamento de complexidade
 - Estratégia de “dividir para conquistar” nos permite lidar melhor com a complexidade de programas grandes
 - Abordagem *top-down* ajuda a pensar!

Sintaxe de um método

Vamos usar esses modificadores por enquanto

Qualquer tipo da linguagem

Mesma regra de nome de variável



Significa que não tem retorno

Mesma regra de declaração de variáveis, separando por vírgula cada argumento

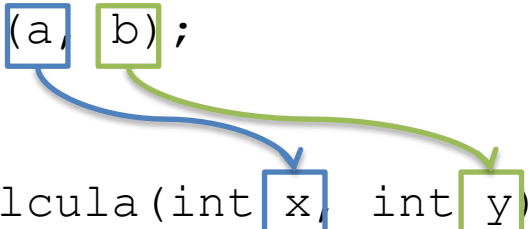
Acesso a variáveis

- Um método não consegue acessar as variáveis de outros métodos
 - Cada método pode criar as suas próprias variáveis locais
 - Os parâmetros para a execução de um método devem ser definidos como argumentos do método
- Passagem por valor
 - Java **copiará o valor** de cada argumento para a respectiva variável
 - Os nomes das variáveis podem ser diferentes

```

z = calcula(a, b);
public static double calcula(int x, int y)

```

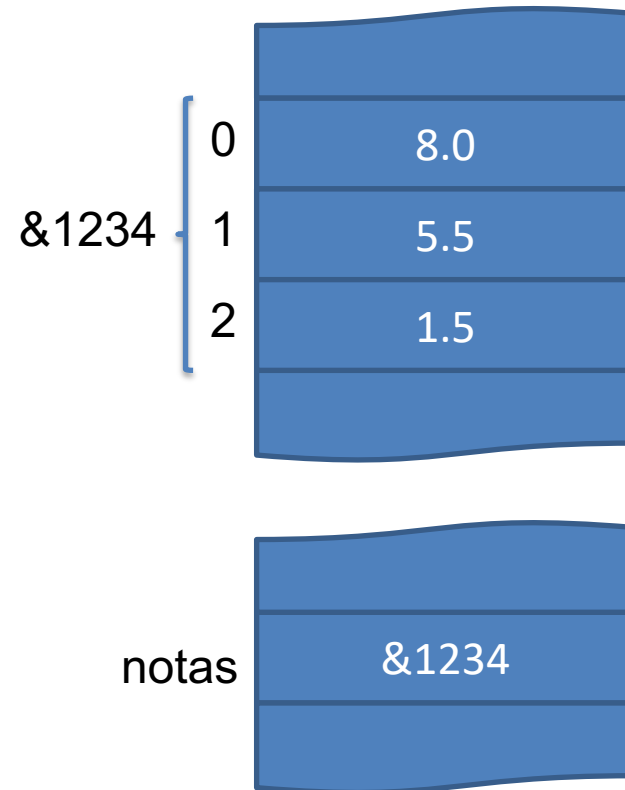


Exemplo

```
public class Troca {
    public static void troca(int x, int y) {
        int aux = x;
        x = y;
        y = aux;
    }
    public static float media(int x, int y) {
        return (x + y) / 2f;
    }
    public static void main(String[] args) {
        int a = 5;
        int b = 7;
        troca(a, b);
        System.out.println("a: " + a + ", b: " + b);
        System.out.println("média: " + media(a,b));
    }
}
```

Passagem de ponteiro por valor

- Variáveis compostas são, na verdade, ponteiros.
- Seus endereços são passados por valor
 - Se criar uma nova variável, o efeito não é notado fora do método
 - Se trocar o valor de uma posição da variável, o efeito é notado fora do método



Exemplo 1

```

public class Array {
    public static void mostra(int[] array) {
        System.out.println(array[0] + ", " + array[1]);
    }

    public static void troca(int[] array) {
        array = new int[2];
        array[0] = 20;
        array[1] = 10;
    }

    public static void main(String[] args) {
        int[] array = { 10, 20 };
        mostra(array);
        troca(array);
        mostra(array);
    }
}

```

Exemplo 2

```

public class Array {
    public static void mostra(int[] array) {
        System.out.println(array[0] + ", " + array[1]);
    }

    public static void troca(int[] array) {
        int tmp = array[0];
        array[0] = array[1];
        array[1] = tmp;
    }

    public static void main(String[] args) {
        int[] array = { 10, 20 };
        mostra(array);
        troca(array);
        mostra(array);
    }
}

```

Sobrecarga de métodos

- Uma classe pode ter **dois ou mais métodos com o mesmo nome**, desde que os tipos de seus argumentos sejam distintos
- Isso é útil quando queremos implementar um método em função de outro
- Exemplo baseado na classe String:

```
public int indexOf(String substring) {
    return indexOf(substring, 0);
}
```


Métodos sem argumentos

- Não é necessário ter argumentos nos métodos
 - Nestes casos, é obrigatório ter () depois do nome do método
 - A chamada ao método também precisa conter ()
- Exemplo de declaração:

```
public static void pulaLinha() {
    System.out.println();
}
```

- Exemplo de chamada:

```
pulaLinha();
```

Exercício

- Faça uma calculadora que forneça as seguintes opções para o usuário, usando métodos sempre que possível
- A calculadora deve operar sempre sobre o valor corrente na memória

Estado da memória: 0

Opções:

- (1) Somar
- (2) Subtrair
- (3) Multiplicar
- (4) Dividir
- (5) Limpar memória
- (6) Sair do programa

Qual opção você deseja?

Subprogramação

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br