



ORACLE

Academy



Java Foundations

4-1

O que É um Método?

ORACLE
Academy



Objetivos

- Esta lição abrange os seguintes objetivos:
 - Estruturar o código dentro de uma classe
 - Instanciar um objeto
 - Entender os benefícios dos métodos
 - Usar o operador dot (.) para acessar os campos e métodos de um objeto
 - Fornecer argumentos a um método
 - Retornar valores de um método



Classes que Você Encontrará

- No desenvolvimento em Java, você encontrará muitas classes para muitos tipos diferentes de objetos, inclusive...
 - Classes que você mesmo criará
 - Classes criadas por outra pessoa
 - Classes que pertencem ao Java

Classes que Você Encontrará

- Essas classes descrevem...
 - Propriedades (campos) dos objetos
 - Comportamentos (métodos) dos objetos
- O objetivo desta lição é fazer com que você entenda como trabalhar com qualquer classe, seus campos e seus métodos
- As demais lições desta seção exploram classes importantes fornecidas pelo Java
- Começaremos explorando classes e métodos mais detalhadamente

Exercício 1, Parte 1

- Vamos analisar um cenário e ver como podemos modelar os componentes envolvidos:
 - É aniversário do Alex! Você organizou um grupo de oito amigos para comemorar em um restaurante local Quando vocês recebem a conta, ninguém tem certeza do que consumiu Só sabem o total de cada um antes do imposto (5%) e da gorjeta (15%) Mas você é uma pessoa de sorte! Você levou seu laptop e seus amigos pediram que criasse um programa para calcular o total de cada convidado



Exercício 1, Parte 2

- Crie um novo projeto e adicione o arquivo Tip01.java a ele
- É isso que cada um deve antes do imposto (5%) e da gorjeta (15%):

Pessoa 1: \$10	Pessoa 5: \$7
Pessoa 2: \$12	Pessoa 6: \$ 15 (Alex)
Pessoa 3: \$9	Pessoa 7: \$11
Pessoa 4: \$8	Pessoa 8: \$30

Exercício 1, Parte 3

- Seu programa deve produzir a seguinte saída:

```
person1: $12.0  
person2: $14.4  
person3: $10.8  
person4: $9.6  
person5: $8.4  
person6: $18.0  
person7: $13.2  
person8: $36.0
```


Modelando Objetos

- Você pode ter ficado tentado a modelar o total de cada pessoa escrevendo o seguinte:

```
public class Tip01{  
    public static void main(String args[]){  
  
        double person1 = 10;  
        double total1 = person1*(1 +.05 +.15);  
        System.out.println(total1);  
    } //fim do método main  
} //fim da classe Tip01
```

Modelando Mais Objetos

- Quando você precisou modelar dois convidados para o jantar, tentou copiar, colar e renomear:

```
public class Tip01{  
    public static void main(String args[]){  
  
        double person1 = 10;  
        double total1  = person1*(1 +.05 +.15);  
        System.out.println(total1);  
  
        double person2 = 12;  
        double total2  = person2*(1 +.05 +.15);  
        System.out.println(total2);  
    }//fim do método main  
}//fim da classe Tip01
```

Modelando Muitos Objetos

- E se você precisasse fazer o cálculo para 1000 convidados?

```
//Você poderia pensar...  
//Eu realmente preciso copiar, colar e renomear 1000  
//vezes?
```

- E se um dos seus amigos esquecesse a carteira? E se você cometesse um erro em sua fórmula?

```
//Você poderia pensar...  
//Preciso fazer 1000 edições?!  
//Tem que haver uma maneira melhor!!!
```



Variáveis Oferecem Flexibilidade

- Se a taxa do imposto ou a porcentagem precisar ser alterada...
 - Não precisaremos fazer 1000 edições
 - Simplesmente editaremos cada variável uma única vez

```
double tax = 0.05;  
double tip = 0.15;
```

```
double person1 = 10;  
double total1 = person1*(1 +tax +tip);  
System.out.println(total1);
```

```
double person2 = 12;  
double total2 = person2*(1 +tax +tip);  
System.out.println(total2);
```

Os Métodos Oferecem uma Flexibilidade Semelhante

- Os mesmos comportamentos de impressão e cálculo são repetidos
- Em vez disso, essa lógica pode ser escrita uma única vez em um método

```
double tax = 0.05;  
double tip = 0.15;
```

```
double person1 = 10;  
double total1 = person1*(1 +tax +tip);  
System.out.println(total1);
```

```
double person2 = 12;  
double total2 = person2*(1 +tax +tip);  
System.out.println(total2);
```

Quando Usar Métodos

- É uma boa ideia escrever um método se você...
- Percebe que está precisando repetir muitas linhas de código, inclusive cálculos
- Precisa descrever o comportamento de um objeto

Como Usar um Método Main

- O método main é conhecido como um orientador
 - Utilize-o para orientar os eventos de um programa
 - Utilize-o para acessar campos e métodos, bem como outras classes
- O método main não descreve o comportamento de qualquer objeto em especial
 - Mantenha-o separado de suas classes de objetos
 - Use um único método main para cada aplicativo

Qual é a Aparência das Classes de Objetos?

- Vamos alterar o código para ajustá-lo ao formato a seguir
- Vejamos o que precisamos fazer para que nosso código tenha este formato:

```
1 public class Calculator{  
2  
3  
4     Properties  
5  
6  
7  
8     Behaviours  
9  
10  
11 }
```

Etapa 1) Mover Campos do Método Main


```
public class Calculator{  
    //Fields  
    public double tax = 0.05;  
    public double tip = 0.15;  
    public double originalPrice = 10;  
  
    public static void main(String args[]){  
        //o dobro do imposto 0.05;  
        //o dobro da gorjeta = 0.15;  
  
        //o dobro da person1 = 10;  
        double total1 = person1*(1 + tax + tip);  
        System.out.println(total1);  
    }//fim do método main  
}//fim da classe Calculator
```

Variáveis locais
se tornam
campos

Etapa 2) Mover Comportamentos Repetidos do Método Main

```
public class Calculator{  
    //Campos  
    public double tax = 0.05;  
    public double tip = 0.15;  
    public double originalPrice = 10;  
  
    //Métodos  
    public void findTotal(){  
        //Calcula o total depois da taxa e da gorjeta  
        //Imprime esse valor  
    } //fim do método findTotal  
  
    public static void main(String args[]){  
        //double total1 = person1*(1 + tax + tip);  
        //System.out.println(total1);  
    } //fim do método main  
} //fim da classe Calculator
```

Escreveremos esse
método no
próximo exercício



Etapa 3) Remover o Método Main

```
public class Calculator{
    //Campos
    public double tax = 0.05;
    public double tip = 0.15;
    public double originalPrice = 10;

    //Métodos
    public void findTotal(){
        //Calcula o total depois da taxa e da gorjeta
        //Imprime esse valor
    } //fim do método findTotal

    //public static void main(String args[]){
        //double total1 = person1*(1 + tax + tip);
        //System.out.println(total1);
    //} //fim do método main
} //fim da classe Calculator
```

Sucesso!

```
public class Calculator{  
    //Campos  
    public double tax = 0.05;  
    public double tip = 0.15;  
    public double originalPrice = 10;  
  
    //Métodos  
    public void findTotal(){  
        //Calcula o total depois da taxa e da gorjeta  
        //Imprime esse valor  
    } //fim do método findTotal  
} //fim da classe Calculator
```

Onde Insiro o Método Main?

```
public class CalculatorTest {  
    public static void main(String args[]){  
        //Crie uma instância do objeto Calculator  
        Calculator calc = new Calculator();  
  
        calc.tip = 0.10;    //Alterando um campo  
        calc.findTotal();  //Chamando um método  
    }//fim do método main  
}//fim da classe CalculatorTest
```

- Coloque o método main em outra classe, como uma classe de teste
- O método main orienta a ação do programa:
 - Ele cria instâncias de objetos
 - Ele chama os campos e os métodos de uma instância usando o operador dot(.)

Variáveis de Objetos

```
int      age  = 22;  
String   str  = "Happy Birthday!";  
Scanner  sc   = new Scanner();  
Calculator calc = new Calculator();
```

tipo *nome* *valor*

- Os objetos, como primitivas, são representados por variáveis
- A maioria dos objetos requer a palavra-chave `new` quando eles são inicializados para criar novas instâncias
 - Isso denomina-se instanciar um objeto
 - Existem algumas exceções, como objetos de `String`, que não requerem a palavra-chave `new`

Usando o Operador dot

- Insira o operador dot (.) depois do nome de uma variável para acessar seus campos ou métodos

```
public class CalculatorTest {  
    public static void main(String args[]){  
        Calculator calc = new Calculator();  
        calc.printTip();           //imprime 0.15  
        calc.tip = 0.10;  
        calc.printTip();           //imprime 0.10  
    }//fim do método main  
}//fim da classe CalculatorTest
```

```
public class Calculator{  
    public double tip = 0.15;      //valor inicializado 0.15  
    public void printTip(){  
        System.out.println(tip);  
    }//fim do método printTip  
}//fim da classe Calculator
```

Exercício 2, Parte 1

- Crie um novo projeto e adicione os arquivos CalculatorTest2.java e Calculator2.java a ele
- Complete o método findTotal(), que deve:
 - Calcular um total com base nos campos tax, tip e originalPrice
 - Imprimir o total de uma pessoa



Exercício 2, Parte 2

- No método main:
 - Instancie um objeto Calculator denominado calc
 - Observe seu JDE depois de digitar "calc"
 - Acesse os campos e os métodos desse objeto para imprimir o total de cada pessoa na festa de aniversário
- Altere tip e tax se preferir valores diferentes



O que Você Pode Ter Escrito

- Você pode ter escrito seu programa desta forma:
 - São necessárias duas linhas para cada pessoa
 - E mais se você decidir imprimir nomes ou alterar os valores do imposto/gorjeta

```
public class CalculatorTest{  
    public static void main(String args[]){  
        Calculator calc = new Calculator();  
  
        calc.originalPrice = 10;  
        calc.findTotal();  
        calc.originalPrice = 12;  
        calc.findTotal();  
    }//fim do método main  
}//fim da classe CalculatorTest
```

Tornando-se Mais Flexível

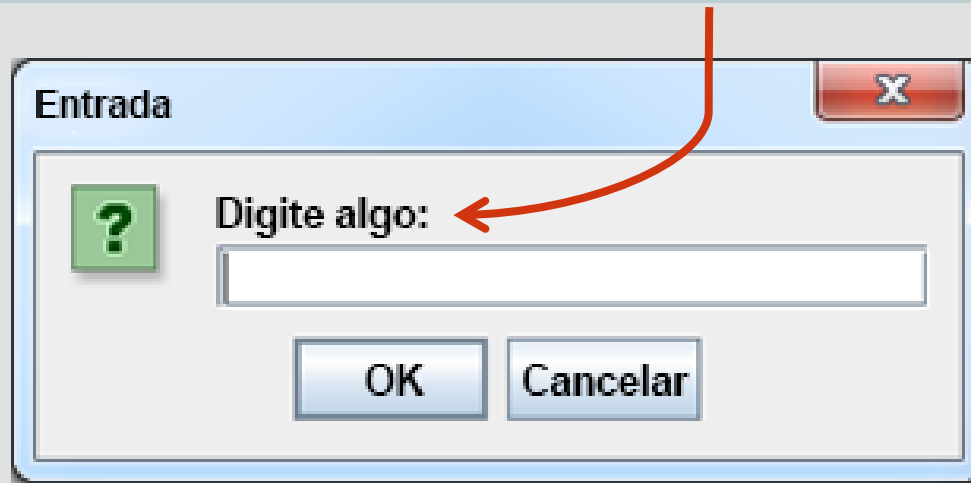
- Mas é possível fazer o mesmo trabalho em uma única linha
- Também é perigoso escrever programas que acessam campos diretamente
 - Você aprenderá sobre isso mais adiante
 - O objetivo desta lição é apenas preparar você para trabalhar com classes importantes fornecidas pelo Java

```
calc.originalPrice = 10;           //Perigoso  
calc.findTotal();
```

Lembre-se de JOptionPane

- Quando adicionamos a literal de String "digite algo:" à chamada do método, estamos fornecendo argumentos ao método
- Esse argumento altera o JOptionPane resultante

```
JOptionPane.showInputDialog("Digite algo:");
```



Quando os Métodos Podem Aceitar Argumentos?

- Você perceberá que muitos métodos são afetados por argumentos
 - Mas os métodos devem ser escritos de forma que aceitem argumentos
 - Caso contrário, o compilador reclamará
 - O método de cálculo é escrito para não aceitar argumentos

```
Calculator calc = new Calculator();  
calc.calculate();           //Bom  
calc.calculate(3, 2.0);     //Fraco
```


```
public void calculate(){  
    //Como eu calculo?  
} //fim do método calculate
```


Argumento do Método

- Mas esse método de cálculo é escrito para aceitar dois argumentos:
 - O primeiro argumento deve ser int
 - O segundo deve ser double

```
Calculator calc = new Calculator();  
calc.calculate(3, 2.0);
```

```
public void calculate(int x, double y){  
    System.out.println(x/y);           //imprime 1.5  
} //fim do método calculate
```



- É atribuído o valor 3 à variável int x
- É atribuído o valor 2 à variável double y

A Ordem dos Argumentos É Importante

- O que acontece se invertermos a ordem de nossos argumentos?

```
Calculator calc = new Calculator();  
calc.calculate(2.0, 3);
```

- Recebemos um erro do compilador:
 - int x não pode receber um valor double
 - O primeiro argumento deve ser int

```
public void calculate(int x, double y){  
    System.out.println(x/y);  
}//fim do método calculate
```

2,0

3

Exercício 3, Parte 1

- Crie um novo projeto e adicione os arquivos CalculatorTest3.java e Calculator3.java a ele
- No método main:
 - Use uma instância do objeto Calculator e forneça argumentos para findTotal(), a fim de imprimir o total de cada pessoa
 - **Dica:** observe o método findTotal() na classe Calculator para descobrir quantos argumentos esse método aceita



Exercício 3, Parte 2

- A quem cada total pertence?
- Modifique o método findTotal() para aceitar mais um argumento name de String
- Concatene a instrução de impressão para incluir name
- Observe a reclamação de seu IDE no método main e revise suas chamadas do método findTotal()



Argumentos e Parâmetros do Método

- Um argumento é um valor que é passado ao chamar um método:

```
Calculator calc = new Calculator();  
calc.calculate(3, 2.0);    //deve imprimir 1.5
```

Argumentos

- Um parâmetro é uma variável que é definida na declaração do método:

```
public void calculate(int x, double y){  
    System.out.println(x/y);  
} //fim do método calculate
```

Parâmetros



Parâmetros dos Métodos: Exemplos

- Os métodos podem ter qualquer número ou tipo de parâmetros:

```
public void calculate0(){  
    System.out.println("Nenhum parâmetro");  
}//fim do método calculate0
```

```
public void calculate1(int x){  
    System.out.println(x/2.0);  
}//fim do método calculate1
```

```
public void calculate2(int x, double y){  
    System.out.println(x/y);  
}//fim do método calculate2
```

```
public void calculate3(int x, double y, int z){  
    System.out.println(x/y +z);  
}//fim do método calculate3
```

O Escopo dos Parâmetros

- Os métodos precisam ser informados sobre o que fazer com os argumentos que recebem
- E você faz isso usando parâmetros do método
 - Os parâmetros do método são variáveis que existem dentro de todo o escopo de um método
 - Eles são criados dentro da declaração do método
 - O escopo refere-se ao **{bloco de código}** que pertence a um método seguido de sua declaração

```
public void calculate(int x, double y){  
    System.out.println(x/y);  
}//fim do método calculate
```


Fazendo Referência a Parâmetros do Método

- É possível fazer referência a uma variável em qualquer parte dentro de seu bloco atual depois que ela for declarada
- Não é possível fazer referência a uma variável fora do bloco em que ela foi declarada ou antes de ela ser declarada

```
public void calculate(int x, double y){  
    System.out.println(x/y);    Escopo de x  
} //fim do método calculate
```



```
public void calculate2(){  
    System.out.println(2*x);    Não escopo de x  
} //fim do método calculate2
```

Encontrando o Grande Total: Cenário

- Seus amigos estão impressionados com tudo o que você está aprendendo no curso Noções Básicas de Java! Alex pergunta: "Qual deve ser o total da mesa inteira?" Saber a resposta dessa pergunta ajudaria a ter certeza de que todos contribuíram e de que o garçom recebeu o valor correto
- Como isso pode ser incluído no seu código?



Somando Totais

- Outra maneira de pensar nisso seria:
 - Calculei um valor dentro de um método...
 - Mas ele está armazenado como uma variável que não pode existir fora do escopo de seu bloco de método...
 - Como retiro esse valor dali?

```
public void findTotal(double price, String name){  
    double total = price * (1 + tax + tip);  
    System.out.println(name + ": $ " + total);  
} //fim do método findTotal
```

Somando Totais

```
public class CalculatorTest
```

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

```
public class Calculator
```

```
    public void findTotal()
```

```
        double total
```

Ha ha! Tente descobrir
meu valor!

Somando Totais

- Se você pensou em escrever seu programa desta forma:

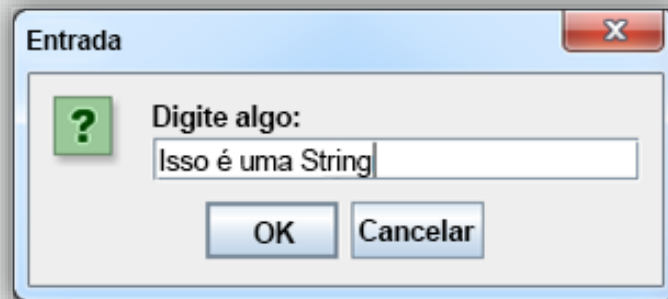
```
public class CalculatorTest{  
    public static void main(String args[]){  
        Calculator calc = new Calculator();  
        calc.findTotal(10);  
        calc.findTotal(12);  
        System.out.println(calc.findTotal(10) +  
            calc.findTotal(12));  
    } //fim do método main  
} //fim da classe CalculatorTest
```

- Você está parcialmente correto
- Mas seu IDE informa o erro a seguir:
 - 'void' type not allowed here

O que É um Tipo Void?

- `showInputDialog()` é um método de tipo de String
 - Ele retorna um valor que pode ser armazenado como uma String

```
String input = JOptionPane.showInputDialog("Digite algo:");
```



- Os métodos do tipo void não retornam valores
 - Não há valores para serem armazenados depois que um método void é chamado

```
System.out.println("println é um método do tipo void");
```

Tipos de Retorno do Método

- As variáveis podem ter valores de muitos tipos diferentes:

int *double* *long* *char* *float* *byte*
short *String* *boolean* *Calculadora*

- As chamadas de método também retornam valores de muitos tipos diferentes:

int *double* *long* *char* *float* *byte*
short *String* *boolean* *Calculadora*

- Como fazer com que um método retorne um valor:
 - Declare o método para ser do tipo de retorno não void
 - Use a palavra-chave `return` dentro de um método, seguida de um valor



Tipos de Retorno do Método: Exemplos

- Os métodos devem retornar dados que correspondam ao respectivo tipo de retorno:

```
public void printString(){  
    System.out.println("Hello");  
}//fim do método printString
```

```
public String returnString(){  
    return("Hello");  
}//fim do método returnString
```

```
public int sum(int x, int y){  
    return(x + y);  
}//fim do método sum
```

```
public boolean isGreater(int x, int y){  
    return(x > y);  
}//fim do método isGreater
```

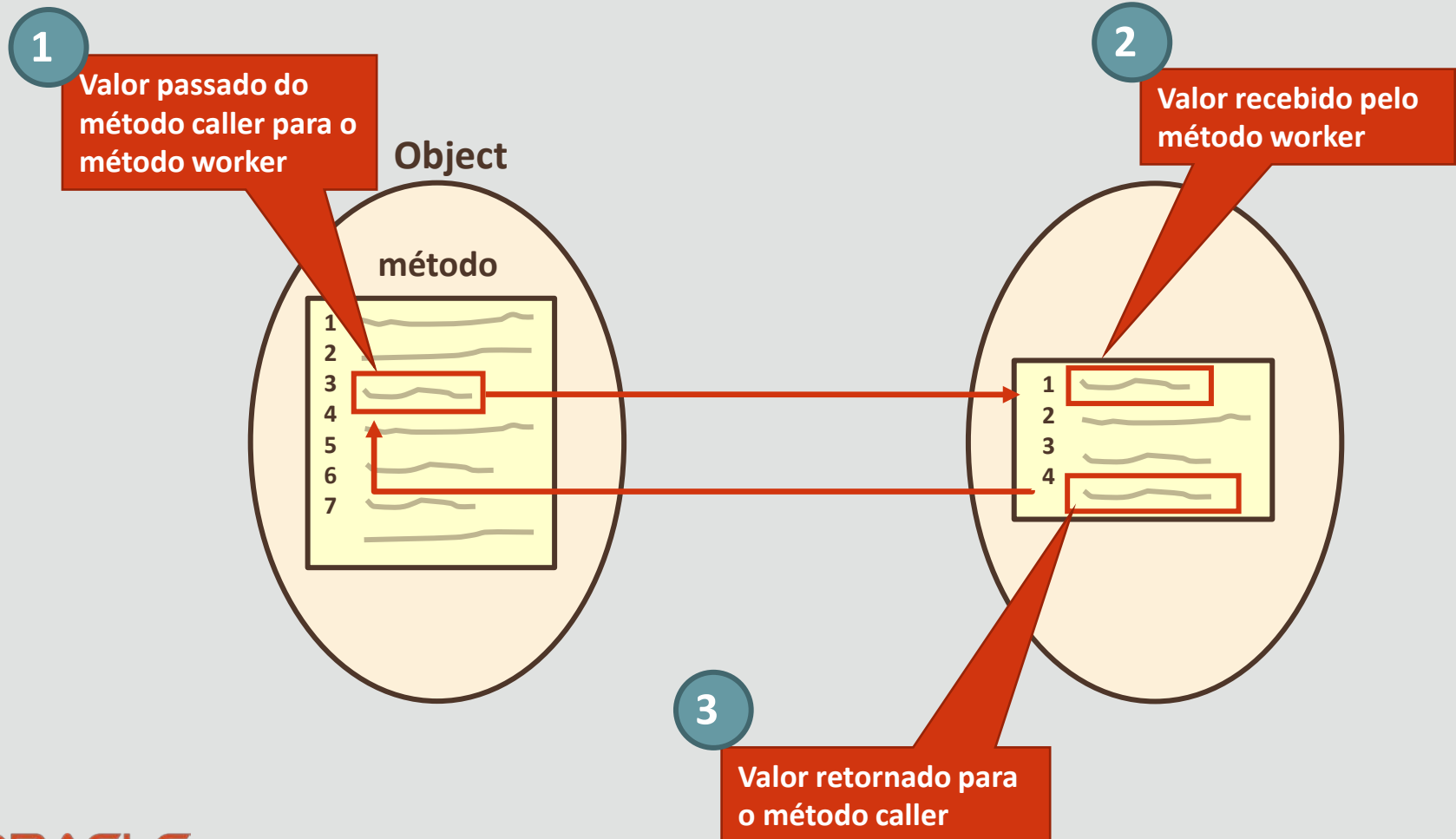

Retorno do Método

- Os dois exemplos a seguir produzem resultados equivalentes:

```
public static void main(String[] args){  
    int num1 = 1, num2 = 2;  
    int result = num1 + num2;  
    System.out.println(result);  
} //fim do método main
```

```
public static void main(String[] args){  
    int num1 = 1, num2 = 2;  
    int result = sum(num1, num2);  
    System.out.println(result);  
} //fim do método main  
public static int sum(int x, int y){  
    return(x + y);  
} //fim do método sum
```

Passando Argumentos e Retornando Valores



Exercício 4, Parte 1

- Edite sua solução do Exercício 3
 - Ou crie um novo projeto e adicione os arquivos CalculatorTest4.java e Calculator4.java a ele
- Encontre e imprima o total da mesa inteira, incluindo o imposto e a gorjeta
 - Você precisará editar findTotal() para que ele retorne seu valor calculado



Exercício 4, Parte 2

- Person8 esqueceu a carteira
- E o jantar do Alex deveria ser um presente de aniversário
- Modifique findTotal() para que o custo das refeições seja dividido igualmente com o restante dos convidados
- Recalcule o total de toda a mesa
- Esse número não deve ter mudado



Resumo sobre a Sintaxe do Método

O tipo de retorno do método
Nome do método
Parâmetros

```
public double calculate(int x, double y){  
    double quotient = x/y;  
    return quotient;  
} //fim do método calculate
```

Implementação

The diagram illustrates the components of a Java method signature and its implementation. Red brackets and lines connect labels to specific parts of the code: 'O tipo de retorno do método' points to 'public double'; 'Nome do método' points to 'calculate'; 'Parâmetros' points to '(int x, double y)'; and 'Implementação' points to the block of code inside the curly braces.

Resumo

- Nesta lição, você deverá ter aprendido a:
 - Estruturar o código dentro de uma classe
 - Instanciar um objeto
 - Entender os benefícios dos métodos
 - Usar o operador dot para acessar os campos e métodos de um objeto
 - Fornecer argumentos a um método
 - Retornar valores de um método





ORACLE

Academy

