

Técnicas de Programação I



Curso Superior de Tecnologia em Desenvolvimento de
Software Multiplataforma

Aula 04

Prof. Claudio Benossi

Estrutura de Decisão

Estrutura de Decisão

if \Rightarrow seleciona um única ação ou um grupo de ações.

```
if (condição) {  
    instruções;  
}
```

if/else \Rightarrow seleciona entre duas ações ou grupo de ações diferentes.

```
if (condição) {  
    instruções 1;  
}  
else {  
    instruções 2;  
}
```

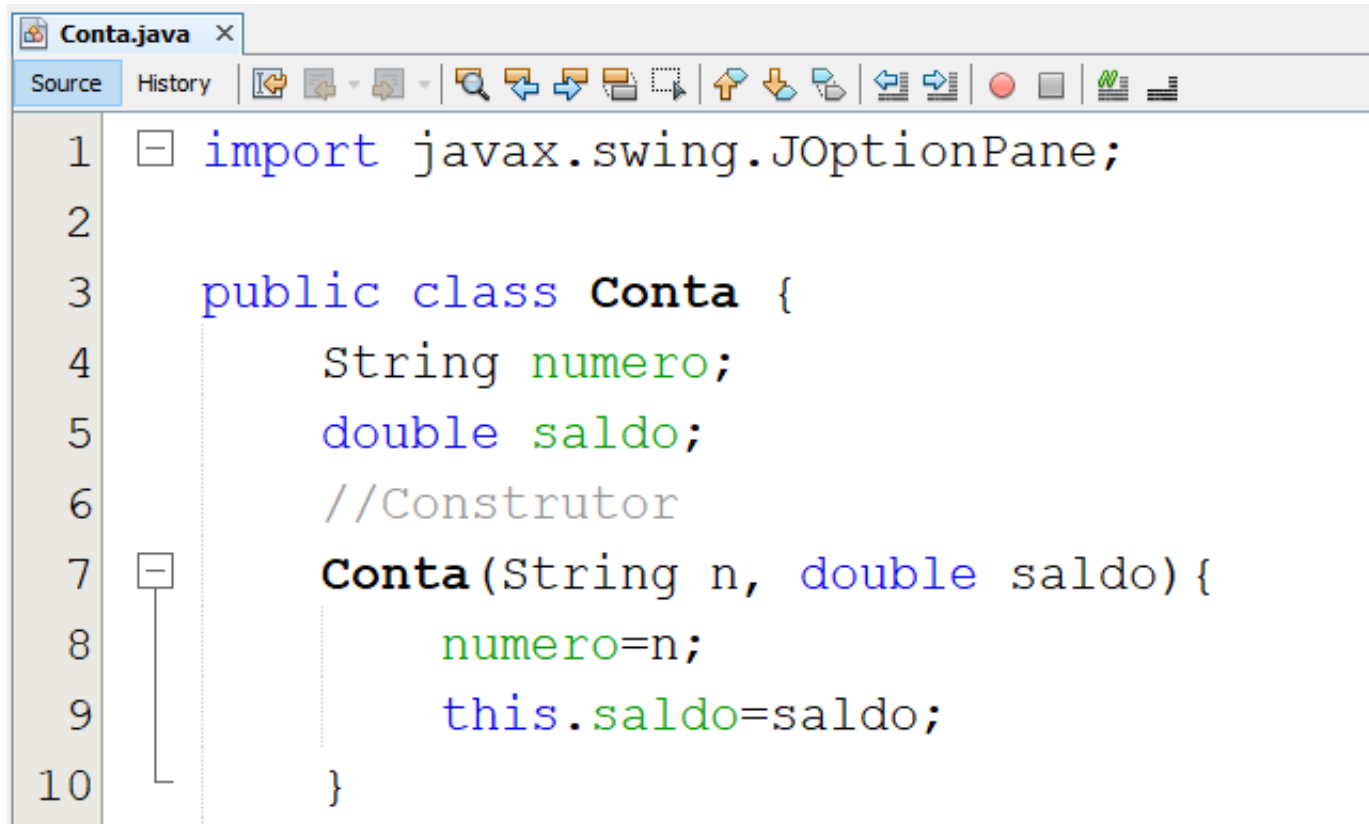
Estrutura de Decisão - Exemplo

Conta
numero: string saldo: double
Conta(n: String, s: double) imprimeDados(): void sacarValor(valor: double): void maiorSaldo(c: Conta): double

O valor só poderá ser retirado da conta caso exista saldo disponível, caso contrário mostre a mensagem “Saldo insuficiente”

Compara o saldo de duas contas e retorna o maior valor, se forem iguais, retornará o valor do objeto c

Estrutura de Decisão - Exemplo



```
Conta.java x
Source History
1  import javax.swing.JOptionPane;
2
3  public class Conta {
4      String numero;
5      double saldo;
6      //Construtor
7      Conta(String n, double saldo){
8          numero=n;
9          this.saldo=saldo;
10     }
```

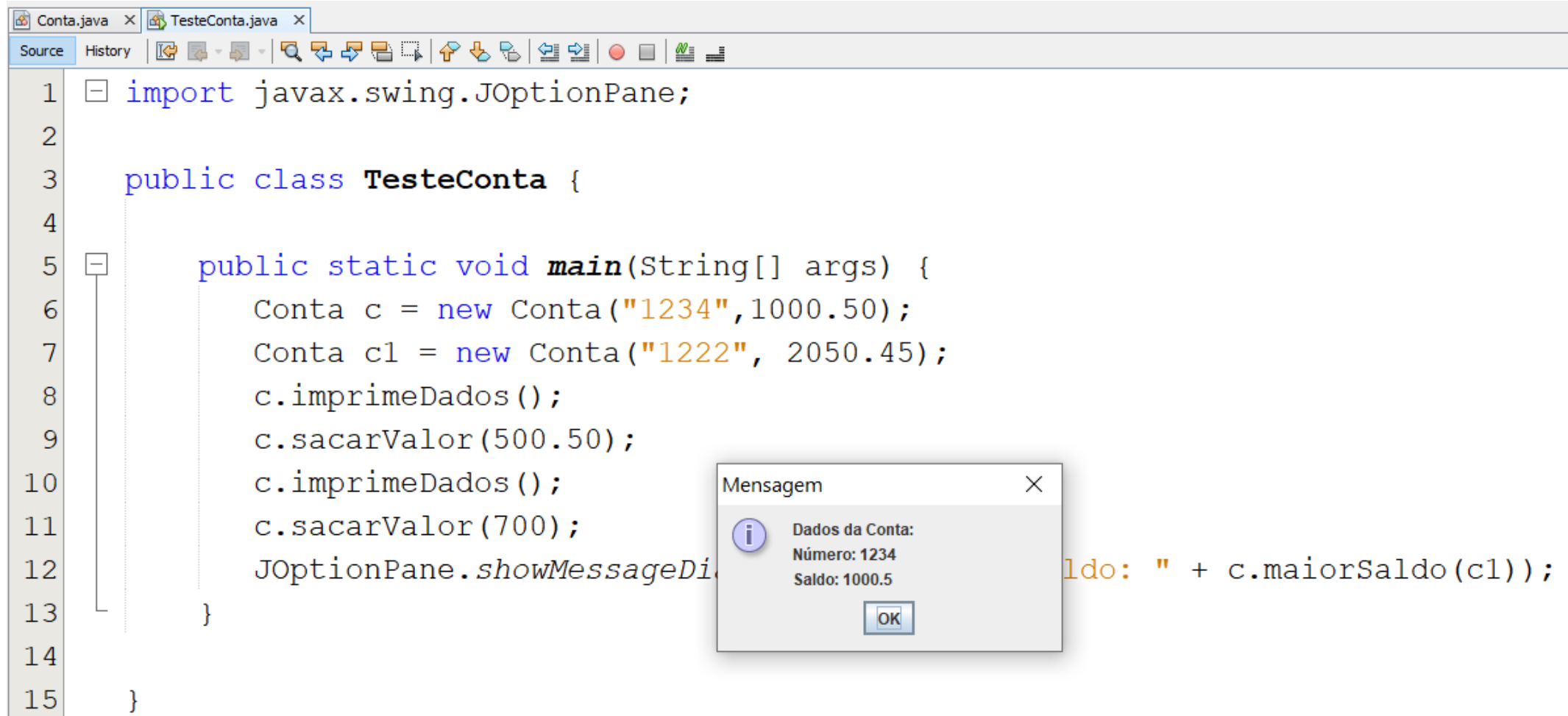
Estrutura de Decisão - Exemplo

```
Conta.java x
Source History
11 //métodos
12 void imprimeDados() {
13     JOptionPane.showMessageDialog(null, "Dados da Conta: " +
14         "\nNúmero: "+numero+
15         "\nSaldo: "+ saldo);
16 }
17 void sacarValor(double valor) {
18     if(valor>saldo) {
19         JOptionPane.showMessageDialog(null, "Saldo insuficiente");
20     }
21     else{
22         saldo=saldo-valor;
23     }
24 }
25 public double maiorSaldo(Conta c) {
26     if(this.saldo > c.saldo)
27         return this.saldo;
28     else
29         return c.saldo;
30 }
```

Estrutura de Decisão - Exemplo

```
Conta.java x TesteConta.java x
Source History
1  import javax.swing.JOptionPane;
2
3  public class TesteConta {
4
5      public static void main(String[] args) {
6          Conta c = new Conta("1234", 1000.50);
7          Conta c1 = new Conta("1222", 2050.45);
8          c.imprimeDados();
9          c.sacarValor(500.50);
10         c.imprimeDados();
11         c.sacarValor(700);
12         JOptionPane.showMessageDialog(null, "Maior saldo: " + c.maiorSaldo(c1));
13     }
14
15 }
```

Estrutura de Decisão - Exemplo



```
1  import javax.swing.JOptionPane;
2
3  public class TesteConta {
4
5      public static void main(String[] args) {
6          Conta c = new Conta("1234", 1000.50);
7          Conta c1 = new Conta("1222", 2050.45);
8          c.imprimeDados();
9          c.sacarValor(500.50);
10         c.imprimeDados();
11         c.sacarValor(700);
12         JOptionPane.showMessageDialog(null, "Número: 1234\nSaldo: 1000.50");
13     }
14
15 }
```

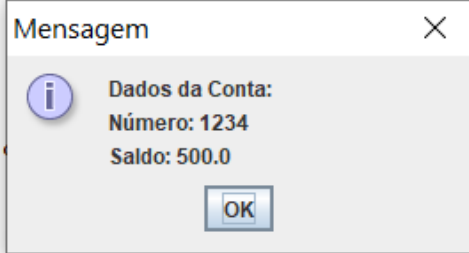
Mensagem

Dados da Conta:
Número: 1234
Saldo: 1000.50

OK

Estrutura de Decisão - Exemplo

```
Conta.java x TesteConta.java x
Source History
1  import javax.swing.JOptionPane;
2
3  public class TesteConta {
4
5      public static void main(String[] args) {
6          Conta c = new Conta("1234",1000.50);
7          Conta c1 = new Conta("1222", 2050.45);
8          c.imprimeDados();
9          c.sacarValor(500.50);
10         c.imprimeDados();
11         c.sacarValor(700);
12         JOptionPane.showMessageDialog(null, "Dados da Conta: \nNúmero: 1234 \nSaldo: 500.0");
13     }
14
15 }
```



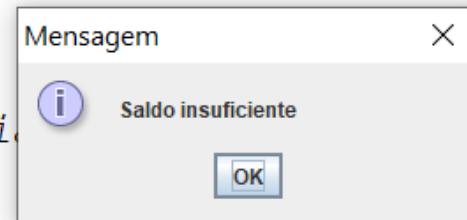
Mensagem

Dados da Conta:
Número: 1234
Saldo: 500.0

OK

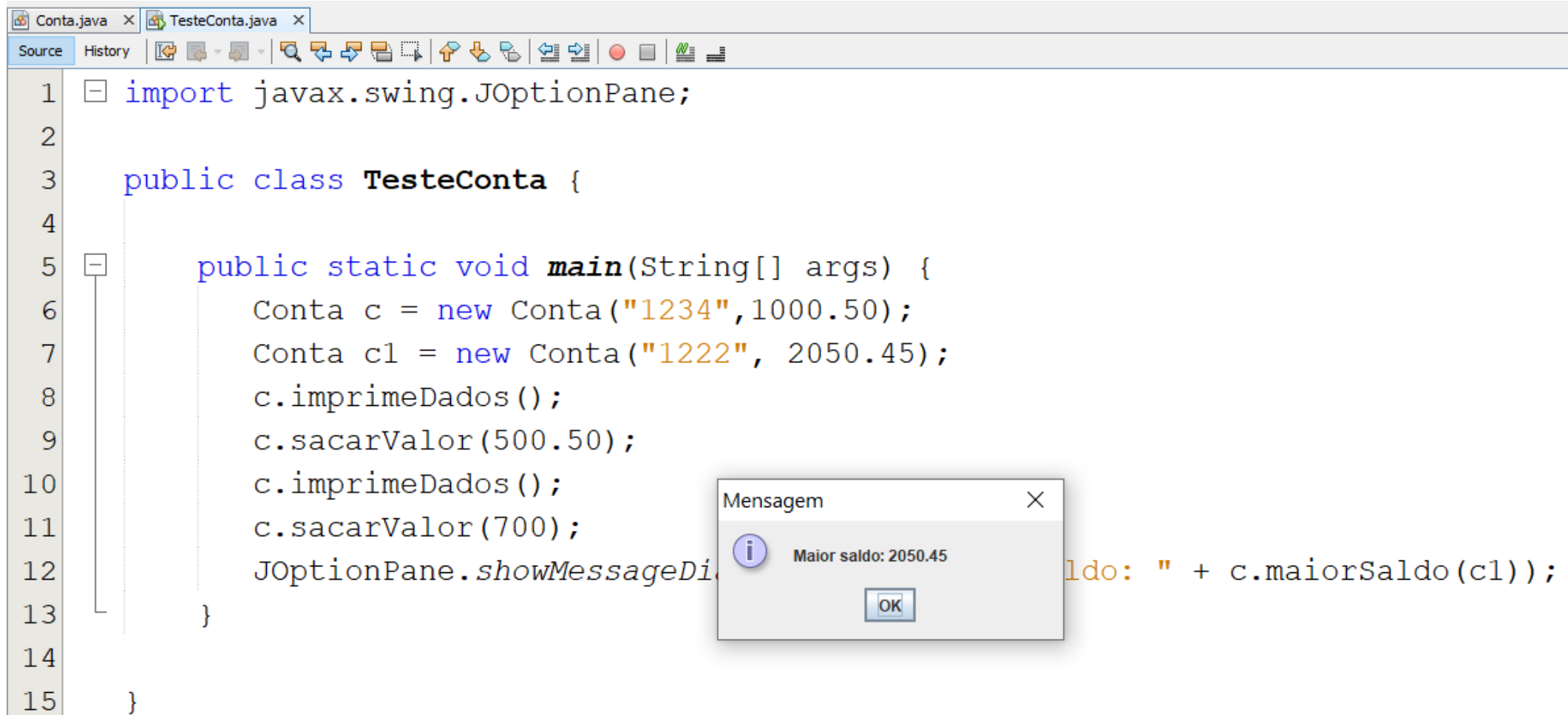
Estrutura de Decisão - Exemplo

```
Conta.java x TesteConta.java x
Source History
1  import javax.swing.JOptionPane;
2
3  public class TesteConta {
4
5      public static void main(String[] args) {
6          Conta c = new Conta("1234", 1000.50);
7          Conta c1 = new Conta("1222", 2050.45);
8          c.imprimeDados();
9          c.sacarValor(500.50);
10         c.imprimeDados();
11         c.sacarValor(700);
12         JOptionPane.showMessageDialog(null, "Saldo insuficiente");
13     }
14
15 }
```



ldo: " + c.maiorSaldo(c1));

Estrutura de Decisão - Exemplo



```
1  import javax.swing.JOptionPane;
2
3  public class TesteConta {
4
5      public static void main(String[] args) {
6          Conta c = new Conta("1234", 1000.50);
7          Conta c1 = new Conta("1222", 2050.45);
8          c.imprimeDados();
9          c.sacarValor(500.50);
10         c.imprimeDados();
11         c.sacarValor(700);
12         JOptionPane.showMessageDialog(null, "Maior saldo: " + c.maiorSaldo(c1));
13     }
14
15 }
```

Mensagem

Maior saldo: 2050.45

OK

Estrutura de Decisão Aninhada

```
if (condição1){  
    instruções 1  
}  
else{  
    if (condição2){  
        instruções 2  
    }  
    else{  
        instruções 3  
    }  
}
```

```
if (condição1){  
    if (condição2){  
        instruções1  
    }  
    else{  
        instruções2  
    }  
}  
else{  
    instruções3  
}
```

Estrutura de Decisão - Exemplo

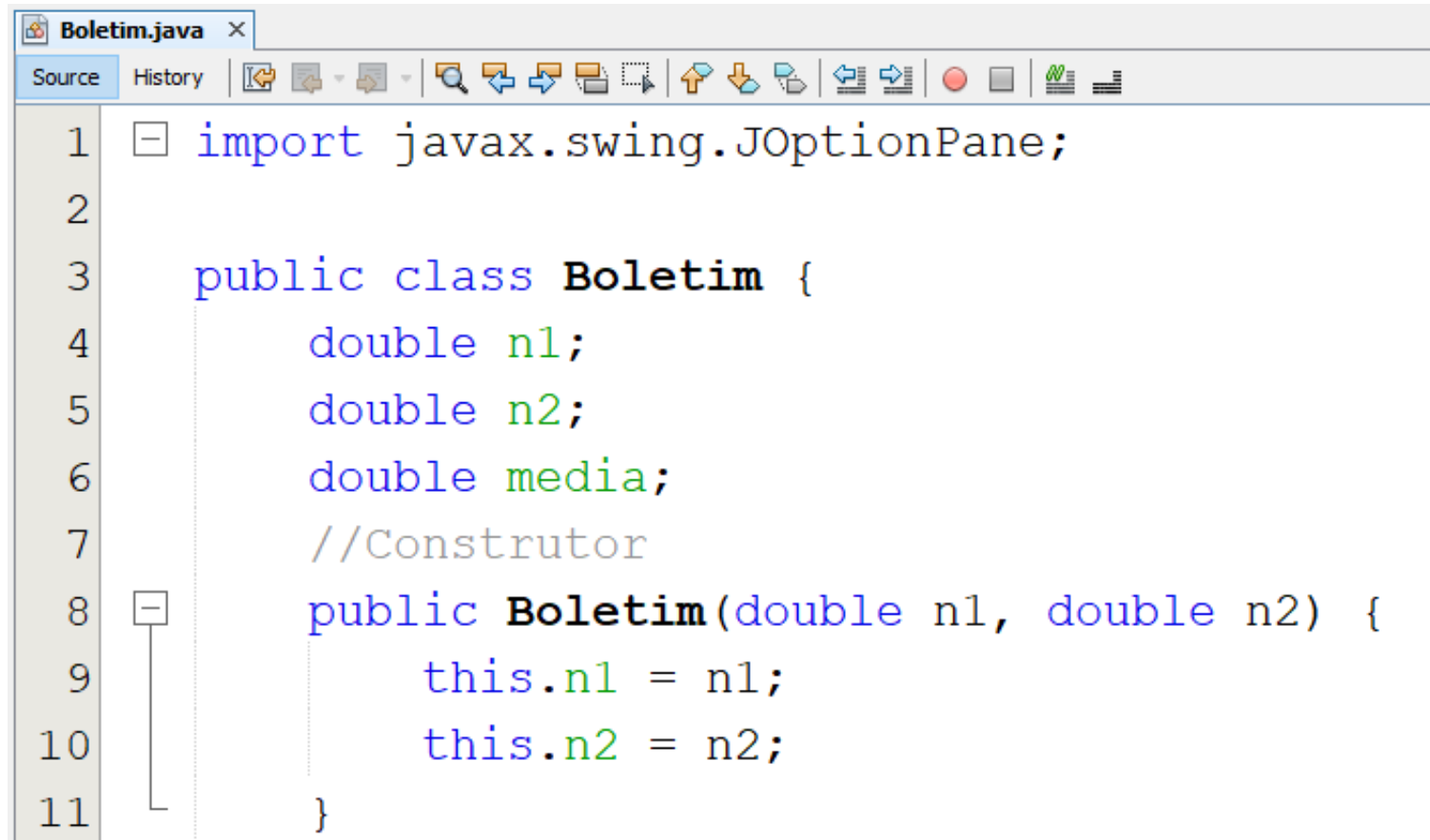
Boletim
n1: double n2: double media: double
Boletim(n1: double, n2: double) imprimeBoletim(): void calculaMedia(): void verificaConceito(): string

Calcula a média:
 $media = (n1 + n2) / 2$

Retorna o conceito, conforme o valor da media, de acordo com a tabela ao lado

MÉDIA	CONCEITO
8,0 ● ——— ● 10,0	A
6,0 ● ——— ○ 8,0	B
4,0 ● ——— ○ 6,0	C
Média abaixo de 4	D

Estrutura de Decisão - Exemplo



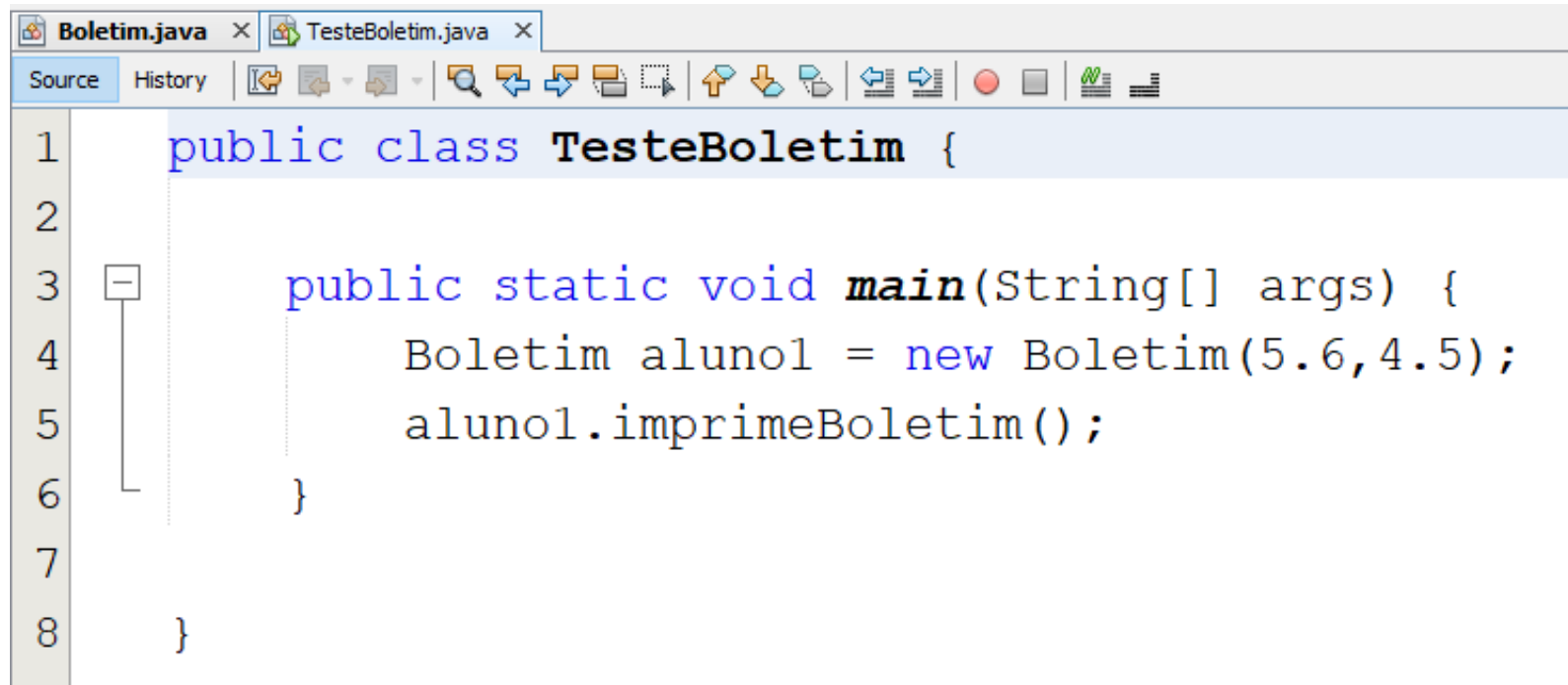
```
Boletim.java x
Source History
1  import javax.swing.JOptionPane;
2
3  public class Boletim {
4      double n1;
5      double n2;
6      double media;
7      //Construtor
8      public Boletim(double n1, double n2) {
9          this.n1 = n1;
10         this.n2 = n2;
11     }
```

Estrutura de Decisão - Exemplo

```
Boletim.java x
Source History
12 //métodos
13 void imprimeBoletim() {
14     calculaMedia();
15     JOptionPane.showMessageDialog(null, "Dados do Boletim: " +
16         "\nNota 1: " + n1 +
17         "\nNota 2: " + n2 +
18         "\nMedia: " + media +
19         "\nConceito: " + verificaConceito());
20 }
21 void calculaMedia() {
22     media = (n1 + n2) / 2;
23 }
```

```
Boletim.java x
Source History
21 void calculaMedia() {
22     media=(n1+n2)/2;
23 }
24 String verificaConceito() {
25     String conceito="";
26     if(media>=8 && media <= 10)
27         conceito="A";
28     else if(media>=6)
29         conceito="B";
30     else if(media>=4)
31         conceito="C";
32     else
33         conceito="D";
34     return conceito;
35 }
36 }
```


Estrutura de Decisão - Exemplo



```
Boletim.java x TesteBoletim.java x
Source History
1 public class TesteBoletim {
2
3     public static void main(String[] args) {
4         Boletim aluno1 = new Boletim(5.6,4.5);
5         aluno1.imprimeBoletim();
6     }
7
8 }
```

Estrutura Switch-Case

- ✓ É uma forma simples para se definir diversos desvios no código a partir de uma única variável.
- ✓ Usada quando se tem várias seleções com muitas alternativas.
- ✓ A partir da versão 7 permite teste com strings.
- ✓ O comando switch testa somente condições simples.



Estrutura Switch-Case



```
escolha (variável)
    caso valor1:
        Instruções1
    caso valor2:
        Instruções2
    caso valorn:
        InstruçõesN
    senão
        Instruções4
fim_escolha
```



```
switch (variável){
    case <valor1>:
        instruções 1;
                                break;
    case <valor2>:
        instruções 2;
                                break;
    case <valorn>:
        instruções n;
                                break;
    default: instruções
              default;
}
```

Estrutura Switch-Case

Observações:

- ✓ Todas as declarações case devem conter valores de um mesmo tipo;
- ✓ O tipo da variável deve ser compatível com os valores das declarações case.
- ✓ A declaração default é opcional.
- ✓ O break finalize o caso, retornando a execução após o comando switch. Case o comando break não seja inserido, todos os outros cases serão testados e executados.

Estrutura Switch-Case

Exemplo:

```
Teste_Switch_Case.java x
Código-Fonte  Histórico
1  import java.util.Scanner;
2
3  public class Teste_Switch_Case {
4      public static void main(String[] args) {
5          Scanner entrada = new Scanner(System.in);
6          System.out.println("Entre com um número entre 1 e 4:");
7          int num = entrada.nextInt();
8          switch (num) {
9              case 1:
10                 System.out.println("Você escolheu 1");
11                 break;
12              case 2:
13                 System.out.println("Você escolheu 2");
14                 break;
15              case 3:
16                 System.out.println("Você escolheu 3");
17                 break;
18              case 4:
19                 System.out.println("Você escolheu 4");
20                 break;
21              default:
22                 System.out.println("Número inválido");
23            }
24        }
25    }
```



Exercícios

01-) Crie um programa que solicite ao usuário informar o número referente ao mês e exibir o nome do mês de acordo com o número informado pelo usuário e caso o número seja < 0 ou > 12 exibir a mensagem de ERRO.

02-) Ler 3 números fracionários do teclado e informar se o primeiro é maior do que a soma dos dois últimos;

03-) Calcular a multa: Leia a velocidade de um carro e a velocidade máxima para a rua:

1. Informe 50 reais se estiver até 10km/h acima;
2. Informe 100 reais se estiver entre 11km/h e 30km/h acima;
3. Informe 300 reais se estiver acima de 31km/h acima;

04-) Ler 3 valores em qualquer ordem e escrever eles em ordem crescente;

05-) Escrever se um ano informado pelo usuário é bissexto ou não.

Um ano é bissexto quando é (divisível por 400) ou é (divisível por 4 e não por 100);

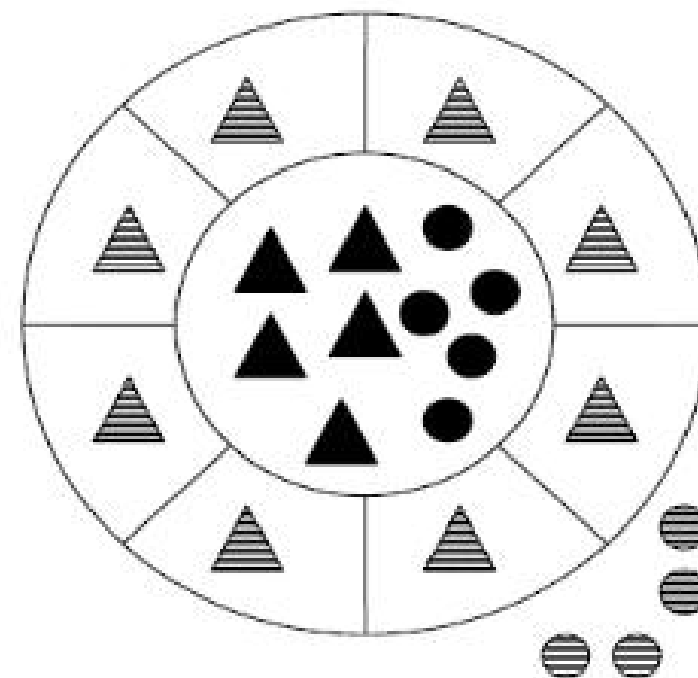
Encapsulamento de Dados

- ❑ O encapsulamento é as vezes referido como **ocultamento de informações**. Os usuários dos objetos não conhecem sua constituição e os utilizam através dos métodos públicos.
- ❑ O encapsulamento elimina dependências diretas na implementação, possibilitando a mudanças sem afetar outros sistemas que utilizem o objeto, desde que as assinaturas dos métodos não sejam alteradas.

Encapsulamento de Dados

CLASSE

- ▲ métodos públicos
- ▲ métodos privados
- dados privados
- dados públicos (não recomendável)



public Menos restrito

↓

private Mais restrito

Encapsulamento de Dados

- ❑ Podemos bloquear o acesso aos atributos da classe.
- ❑ Criamos métodos (*set* e *get*) para acessar os atributos, já que eles não são mais acessíveis diretamente.
- ❑ Nesses métodos, podemos "validar" o que estão tentando armazenar nos atributos.

Modificadores de Acesso

- Há três **modificadores** de acesso: **public**, **protected** e **private**;
- Atributos e Métodos podem ter os **três níveis** de acesso.
- Elementos **públicos** podem ser acessados diretamente por qualquer outra classe, utilizando um ponto (.) após o nome da variável.
- Elementos **privados** e **protegidos** não podem ser acessados diretamente utilizando o ponto.

Modificadores de Acesso

- **Private**

O modificador de acesso `private` é o mais restritivo de todos, variáveis e métodos com esse modificador são visíveis somente dentro da definição da própria classe, acessando-o diretamente ou através de uma instância da mesma classe.

- **Protected**

O modificador de acesso `protected` define que variáveis e métodos com esse modificador podem somente ser acessados por subclasses.

- **Public**

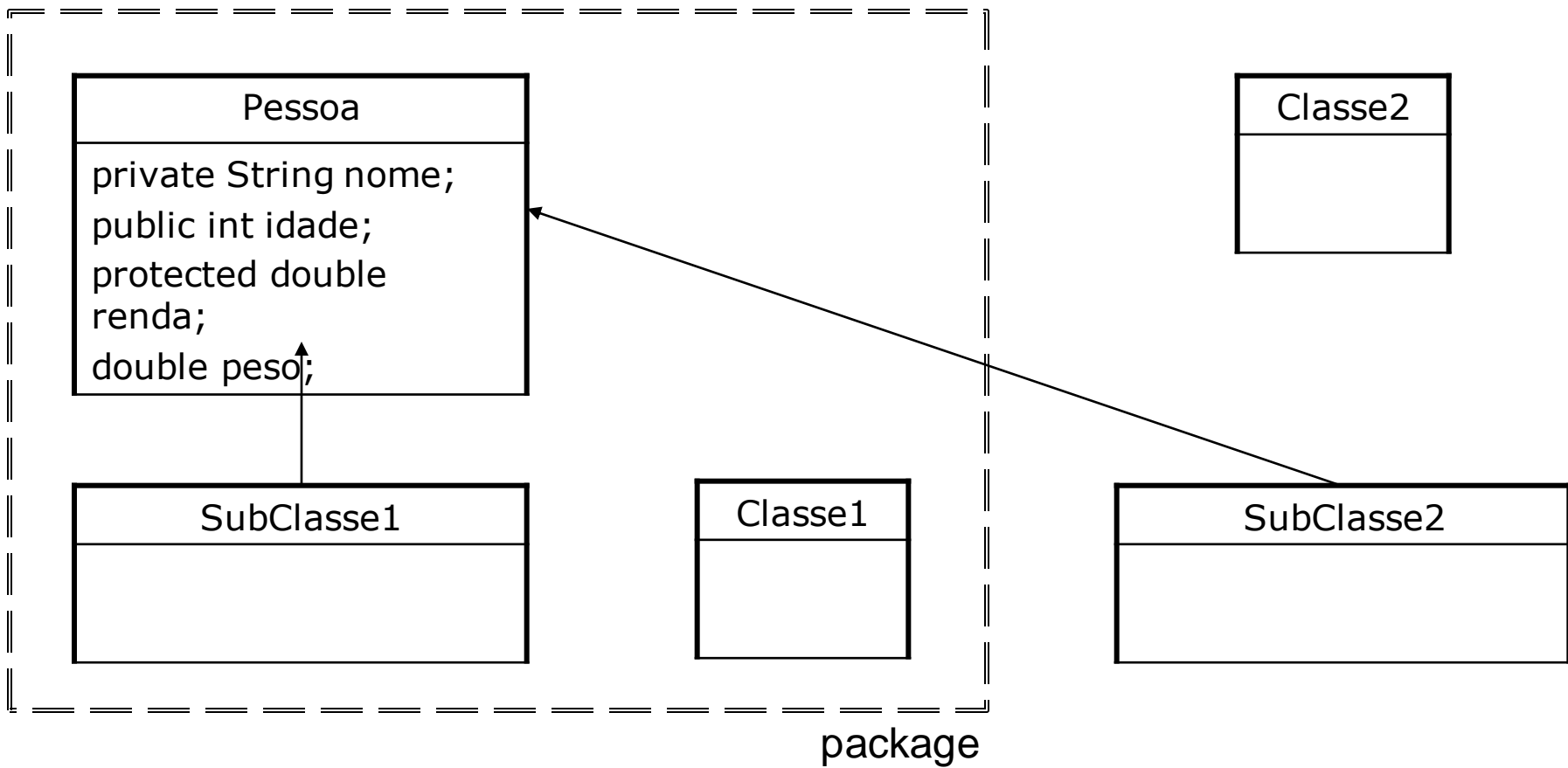
O mais abrangente de todos os tipos de acesso, o modificador `public` declara que elementos com esse modificador são acessíveis de qualquer classe Java.

- **Default**

Define que variáveis ou métodos podem somente ser acessados por classes do mesmo pacote.

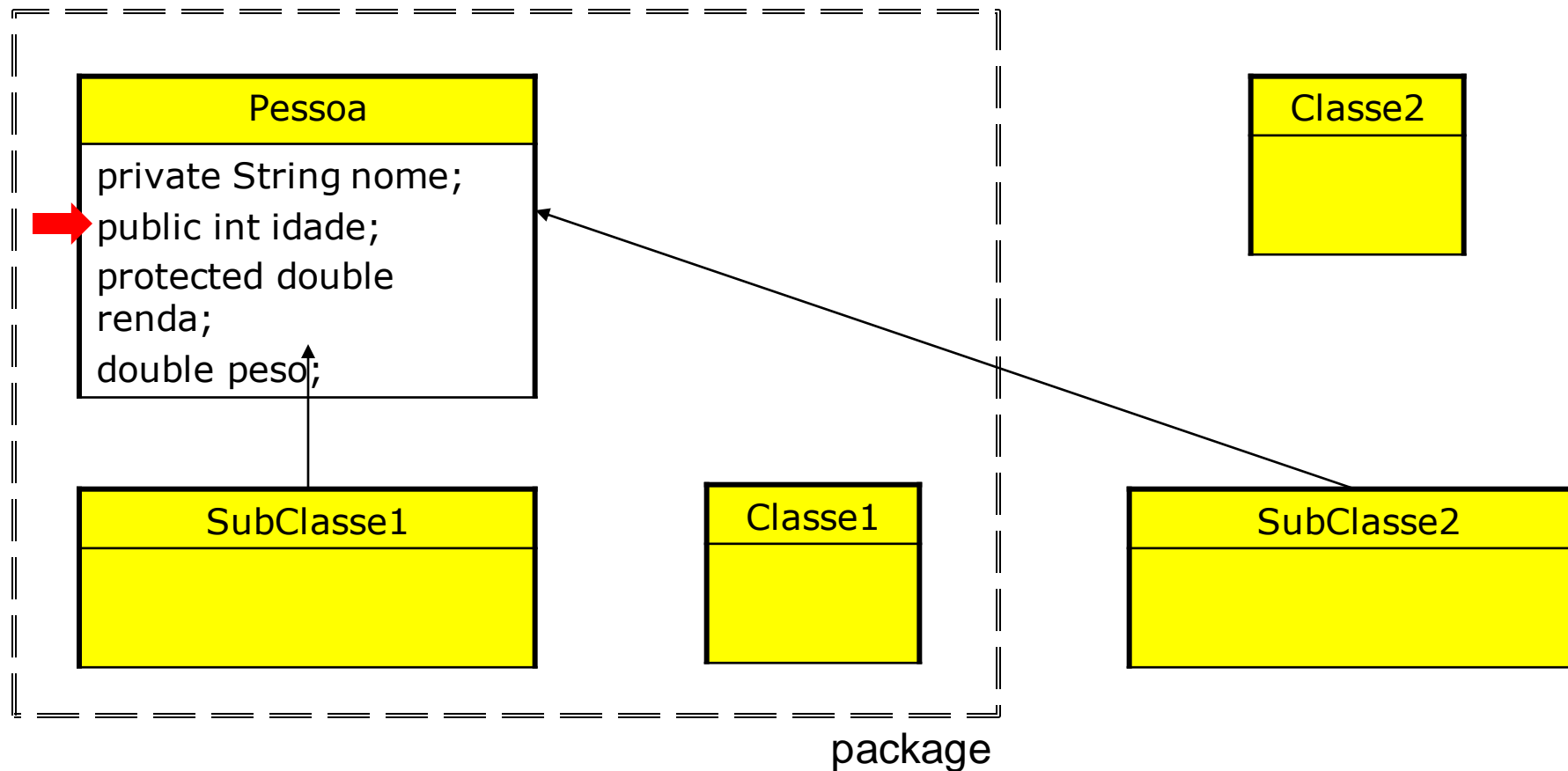


Modificadores de Acesso



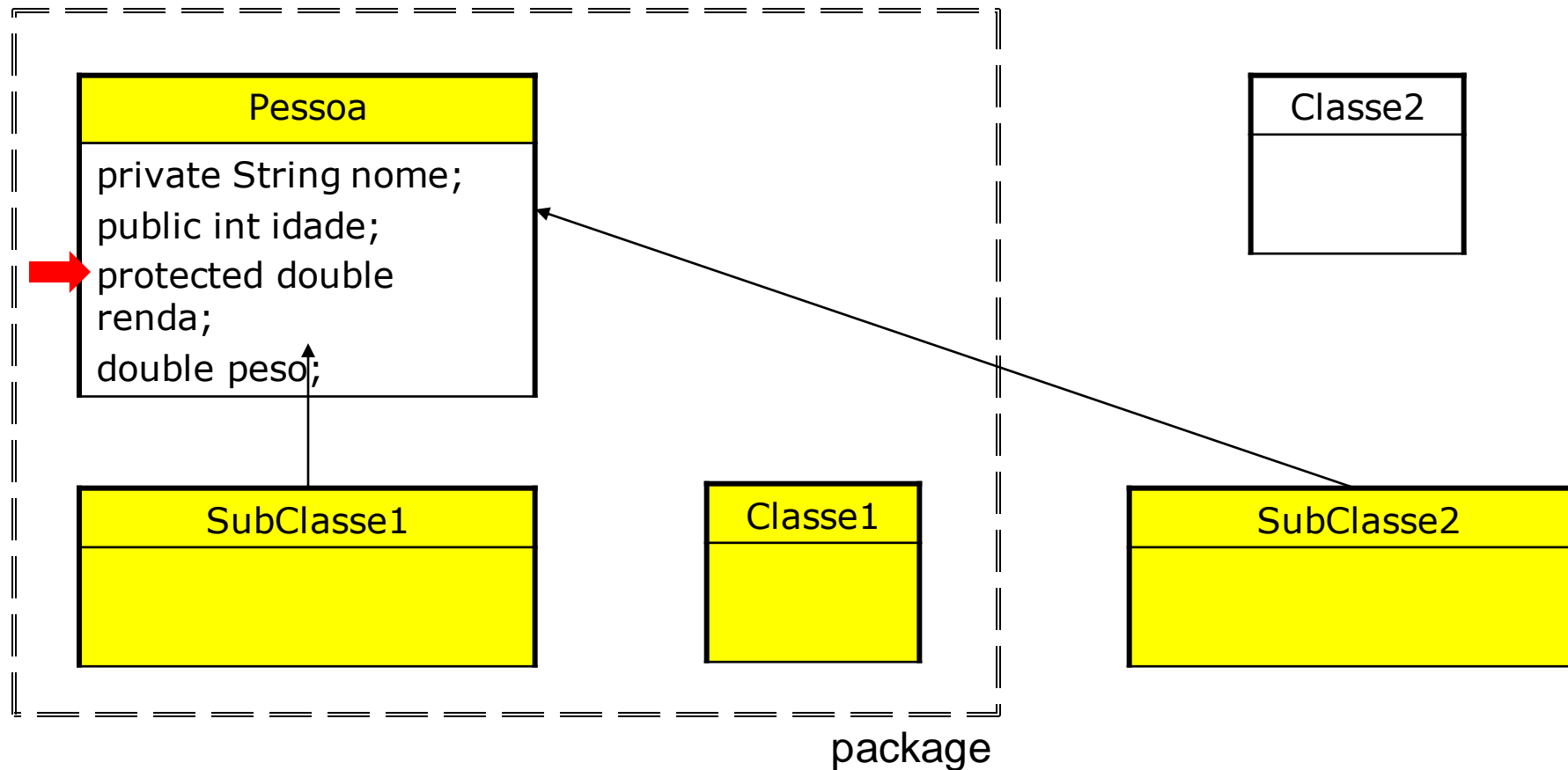
Modificadores de Acesso

Visibilidade de variáveis e métodos com o modificador **public**.



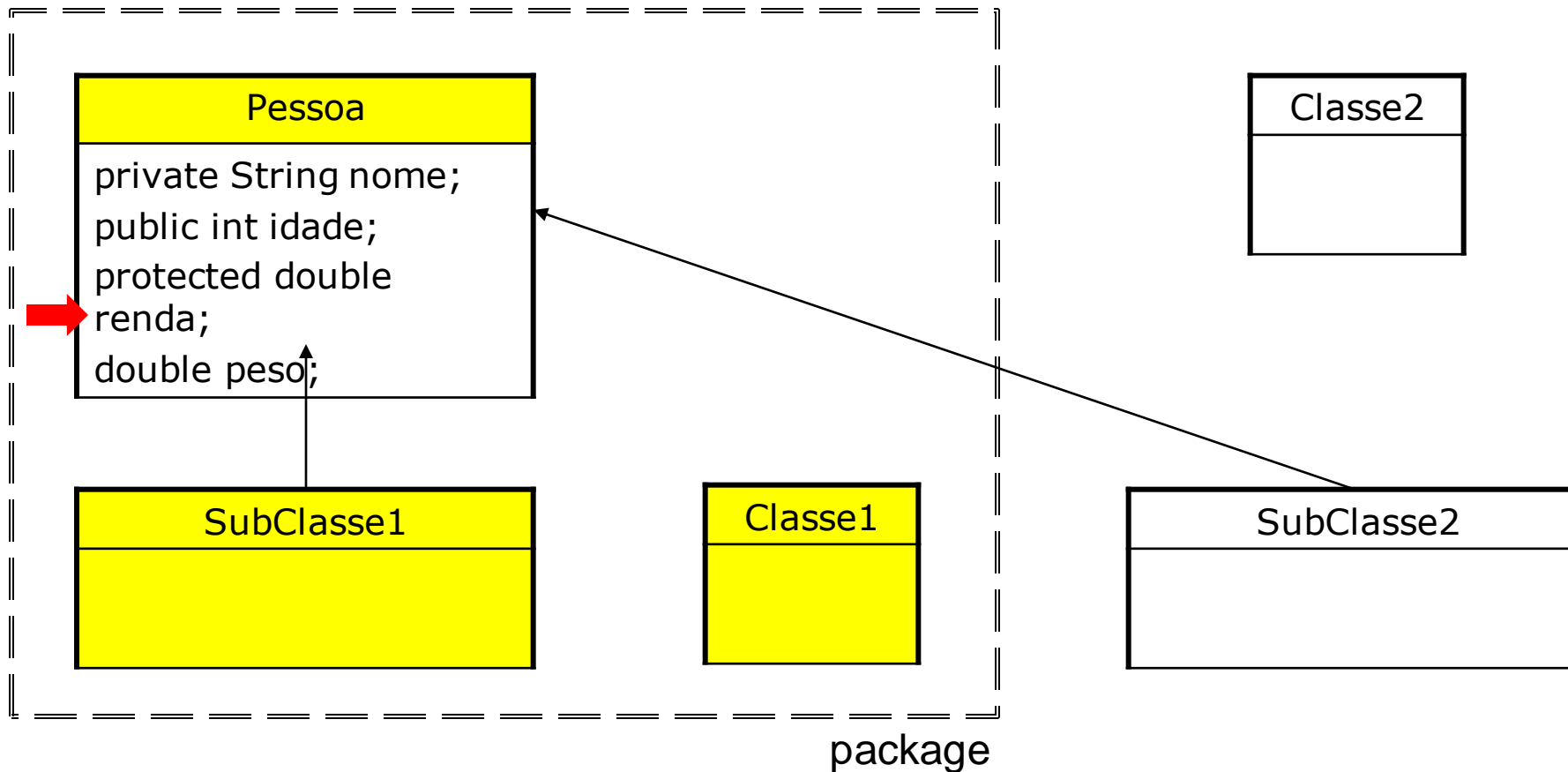
Modificadores de Acesso

Visibilidade de variáveis e métodos com o modificador **protected**.



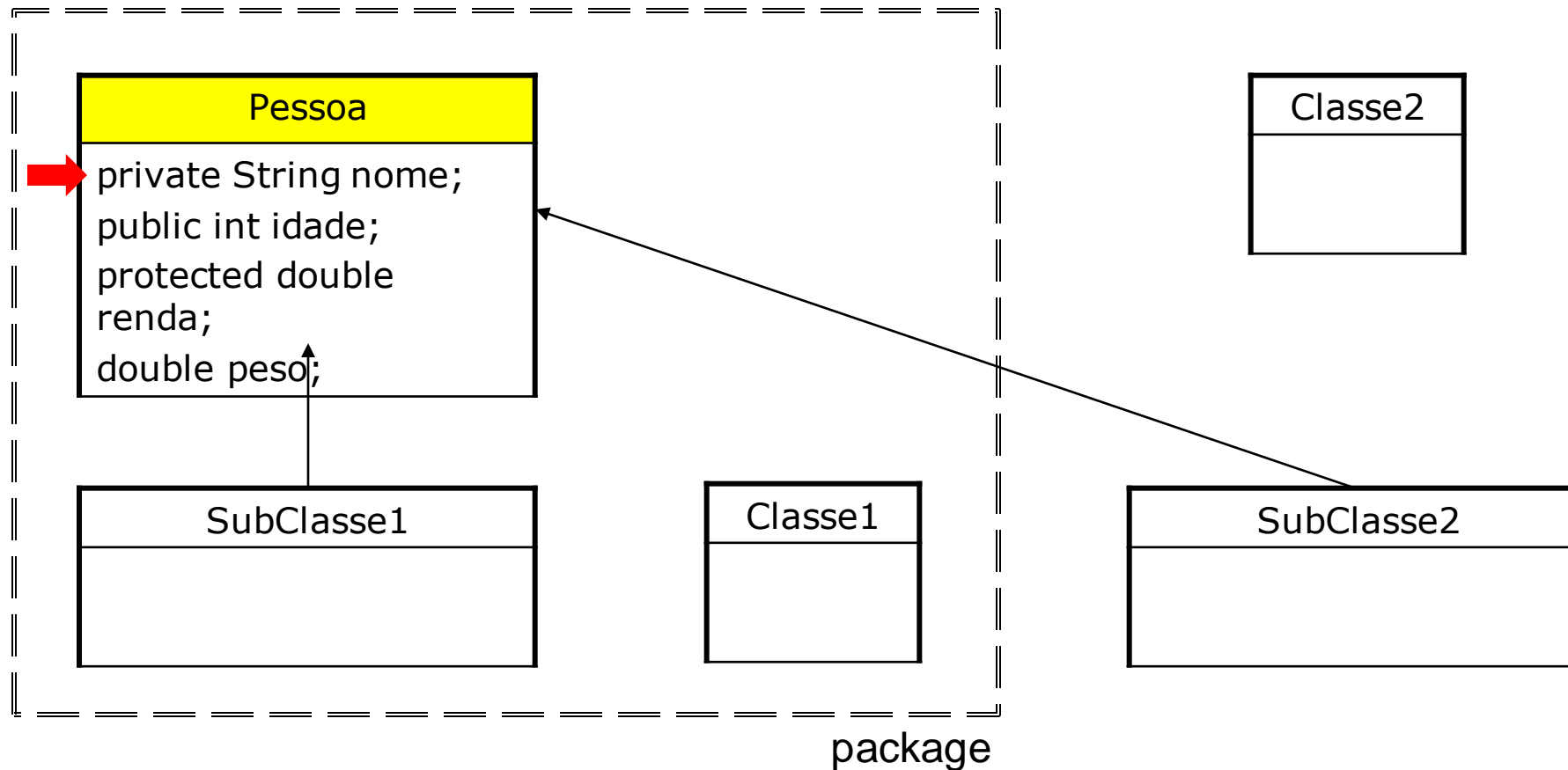
Modificadores de Acesso

Visibilidade de variáveis e métodos com o modificador **default**.



Modificadores de Acesso

Visibilidade de variáveis e métodos com o modificador **private**.



Exemplo

Pessoa
- nome: String + idade: int # renda: double
imprimeDados(): void

Legenda:

(-) indica private
(#) indica protected
(+) indica public

Obs: a ausência de sinal na frente do atributo indica que ele é **default**

```
import javax.swing.JOptionPane;

public class Pessoa {
    //Atributos
    private String nome;
    public int idade;
    protected double renda;

    //métodos
    public void imprimeDados() {
        JOptionPane.showMessageDialog(null, "Nome:" + nome
            + "\nIdade: " + idade
            + "\nRenda: " + renda);
    }
}
```

Exemplo

```
public class TestePessoa {  
    public static void main(String[] args) {  
        Pessoa p1 = new Pessoa();  
  
        p1.nome = "Camila"; //erro de acesso  
        p1.idade = 30; //acesso bem sucedido  
        p1.renda = 1700.55; // acesso bem sucedido  
  
        p1.imprimeDados(); // acesso bem sucedido  
    }  
}
```

Métodos de Acesso

- Servem como métodos de leitura/escrita aos atributos de classes
- Um método de leitura para um atributo deve ser chamado de **getXxx** (onde Xxx é o nome do atributo). Este método não recebe nada como parâmetro, e retorna o mesmo tipo do atributo.
- Já um método de gravação deve ser chamado **setXxx**, não retorna nada (geralmente), e recebe como parâmetro o valor que deve ser armazenado no atributo.

Exemplo

Pessoa
<ul style="list-style-type: none">- nome: String- idade: int- renda: double
Pessoa() Pessoa(n: String, i: int, r: double) getNome(): String getIdade(): int getRenda(): double setNome(String n): void setIdade(int i): void setRenda(double r): void imprimeDados(): void

```
...  
//Métodos de acesso  
public String getNome() {  
    return nome;  
}  
public int getIdade() {  
    return idade;  
}  
public double getRenda() {  
    return renda;  
}  
public void setNome(String n) {  
    nome = n;  
}  
public void setIdade(int i) {  
    idade = i;  
}  
public void setRenda(double r) {  
    renda = r;  
}  
...
```

Exercício 1

De acordo com a classe Funcionário abaixo, crie um construtor sem parâmetros (vazio) que deverá atribuir ao cargo o valor “assistente” e um outro construtor que recebe parâmetros correspondentes aos atributos.

Funcionario
- cracha: int - salario: float - cargo: String
Funcionario() Funcionario(c: int, s: float, car: String) //Métodos de acesso calculaAumento(porcentagem: float) calculaAumento(tempo: int)

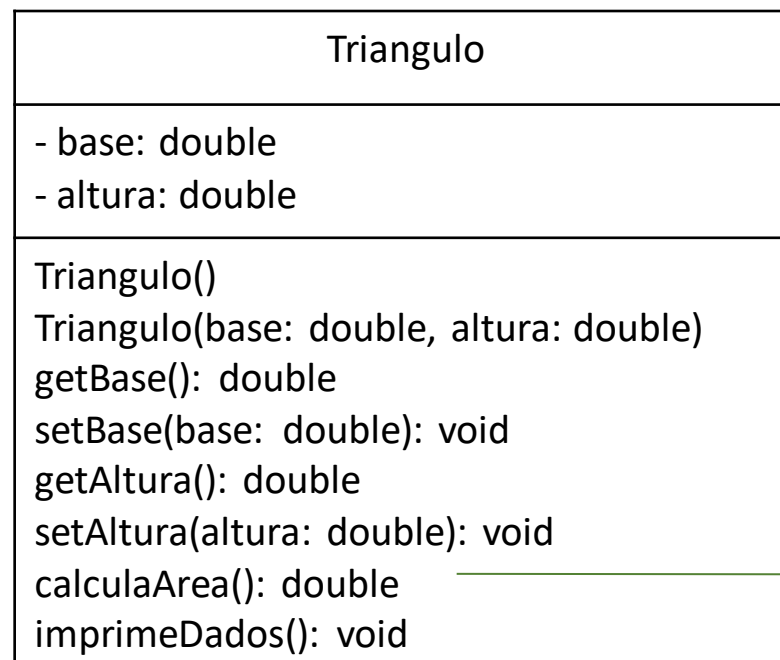
Este método aplica a porcentagem de aumento no salário.

Este método soma R\$150,00 no salário para cada ano trabalhado (recebido por parâmetro).

Exercício 2

Considere o diagrama UML abaixo e altere a classe para acrescentar os modificadores de acesso e os demais métodos necessários.

Em uma classe Java principal (com método main) crie 2 objetos, cada um deve ser instanciado por um construtor diferente. Para o objeto que utiliza o construtor com parâmetros, defina os valores dos atributos. Para o objeto que utiliza o construtor padrão, após a instanciação do mesmo, solicite ao usuário os valores da base e da altura e altere os valores dos atributos utilizando os métodos de acesso. Para ambos, imprima os dados e sua área.



$$\text{area} = \text{base} * \text{altura} / 2$$

Exercício 3

Torneio
- nome: string - idade: int
Torneio(nome: string, idade: int) getNome(): string getIdade(): int setNome(n: string): void setIdade(i: int): void verificaCategoria(): string imprimeDados(): void

- a) Crie os sets e gets para cada um dos atributos;
- b) Crie um método imprimirDados que imprime o estado do objeto inclusive sua categoria;
- c) O método verificarCategoria que deverá retornar qual a categoria do atleta baseado na tabela abaixo:

Categoria	Idade
Infantil	5 a 7
Juvenil	8 a 10
Adolescente	11 a 15
Adulto	16 a 30
Sênior	Acima de 30

Exercício 4

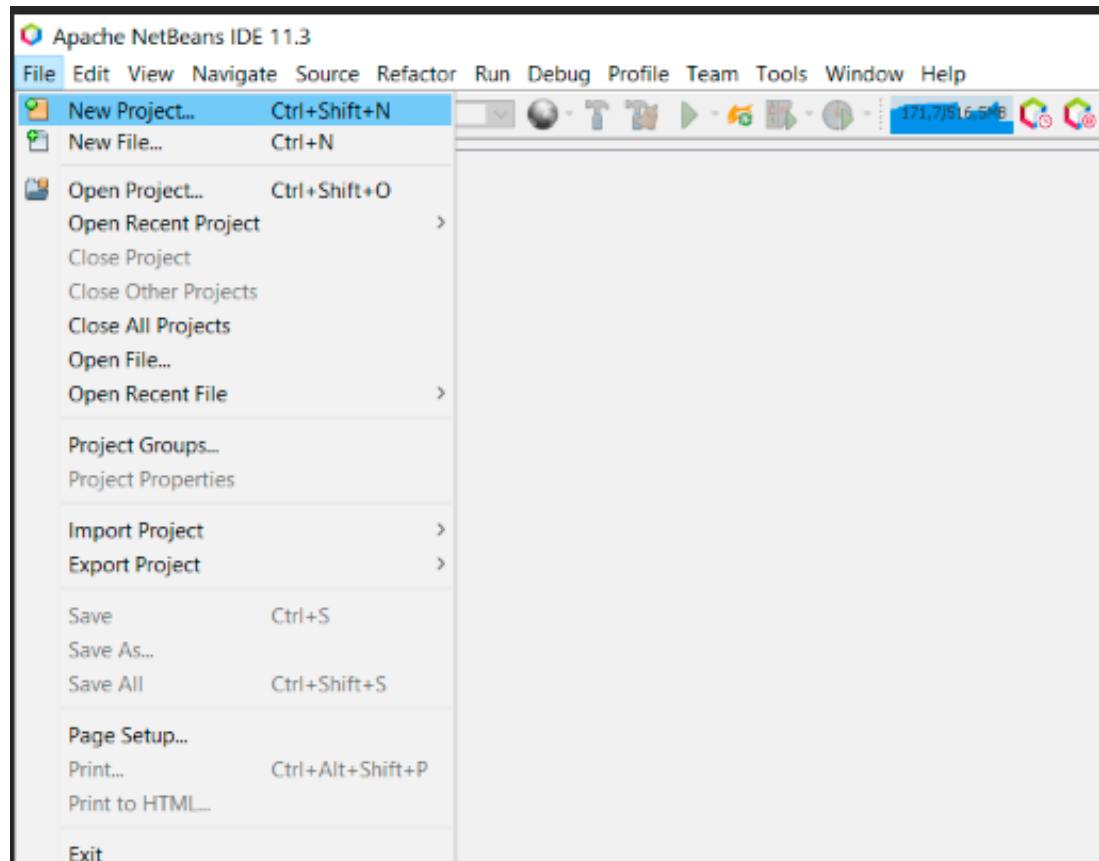
- ▶ Crie uma classe de nome Vendedor conforme diagrama e criar classe com **void main** para instanciar objetos:

Vendedor
<ul style="list-style-type: none"> - vendas: float - salario: float - nome: String - falta: int
Vendedor (v:float, s:float, n:String, f: int) setVendas(v: float): void getVendas(): float setSalario(s: float): void getSalario(): float setNome(n: string): void getNome(): string setFalta(f: int): void getFalta(): int imprimirDados(): void calcularSalario(): void calcularComissao(): float descontoFalta(): float

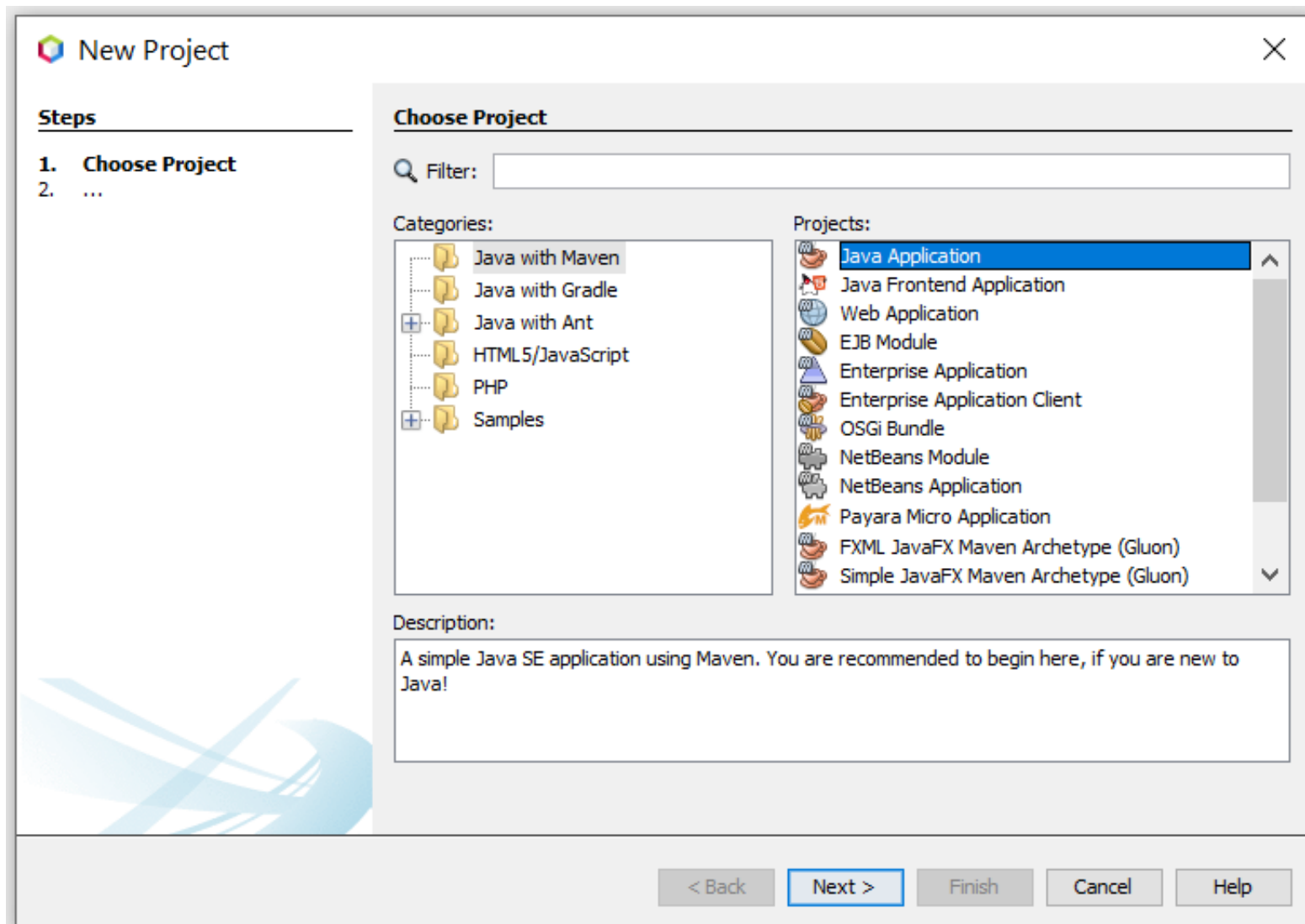
- Crie os sets e gets para cada um dos atributos;
- Crie um método imprimirDados que imprime o estado do objeto;
- O método calcularComissao deverá retornar o valor da comissão, conforme as regras a seguir:
 - venda igual ou acima de 1.000 e menor que 2.000 bônus de 10% sobre o valor das vendas.
 - venda maior ou igual a 2.000 bônus de 15% sobre o valor das vendas.
- O método descontoFalta deverá calcular o desconto das faltas conforme o critério: $\text{desconto} = (\text{salario} / 30) * \text{falta}$
- O método calcularSalario deverá atender ao critério:
 $\text{salario} = (\text{salario} + \text{comissao} - \text{descontoFalta})$

Exercício 4


Novo Projeto



Exercício 4



Exercício 4

 New Java Application ✕

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location: Browse...

Project Folder:

Artifact Id:

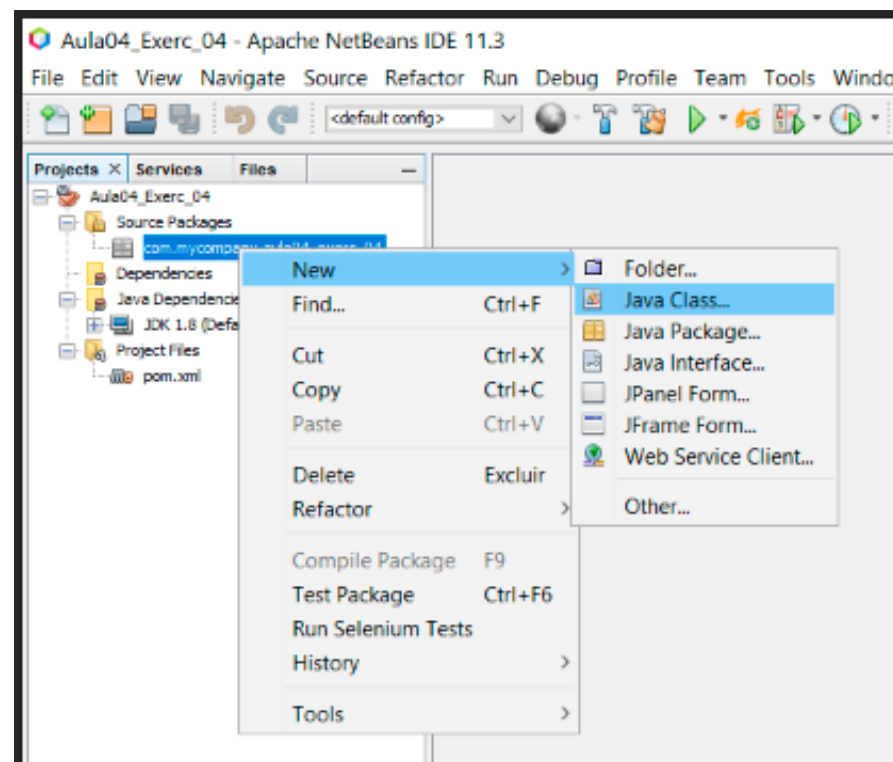
Group Id:

Version:


Package: (Optional)

< Back Next > Finish Cancel Help

Exercício 4



Exercício 4

 New Java Class ✕

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location: ▾

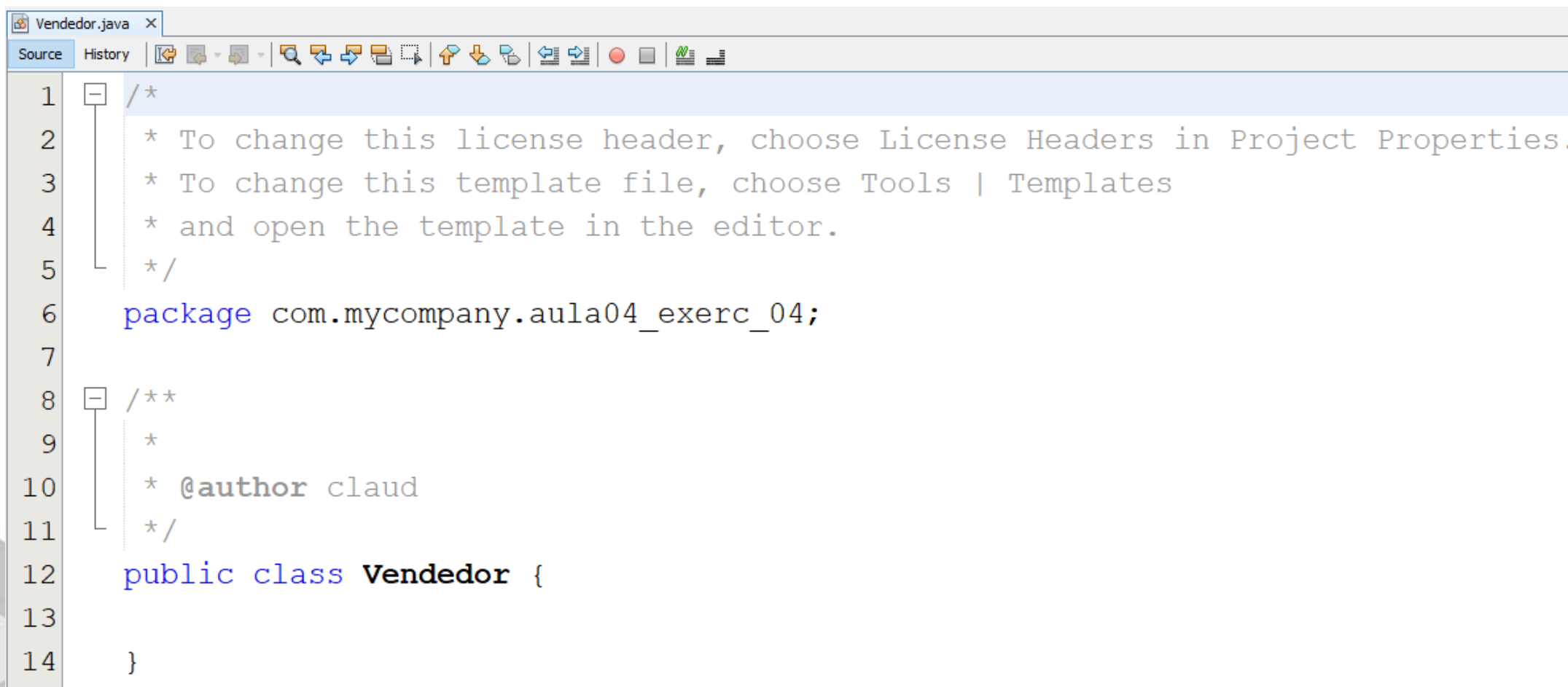
Package: ▾

Created File:

< Back Next > Finish Cancel Help

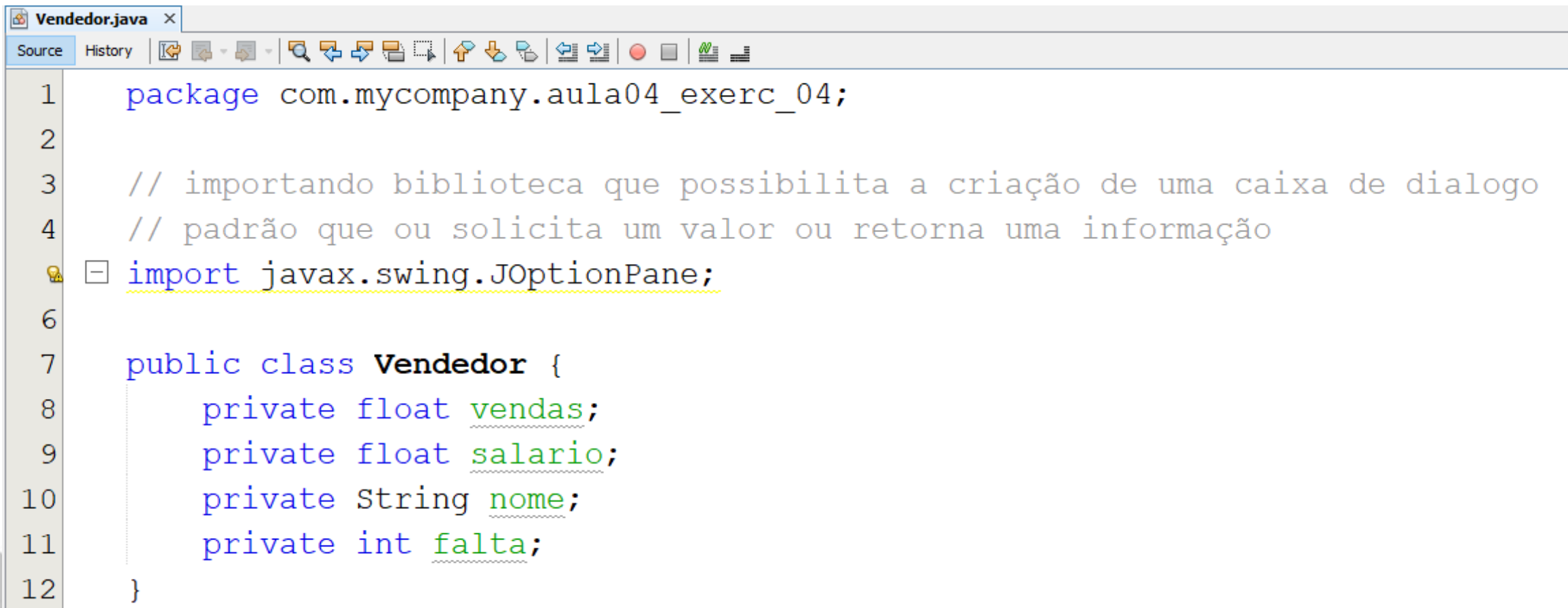
Exercício 4

Vamos limpar o código



```
Vendedor.java x
Source History
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package com.mycompany.aula04_exerc_04;
7
8  /**
9   *
10   * @author claud
11   */
12  public class Vendedor {
13
14  }
```

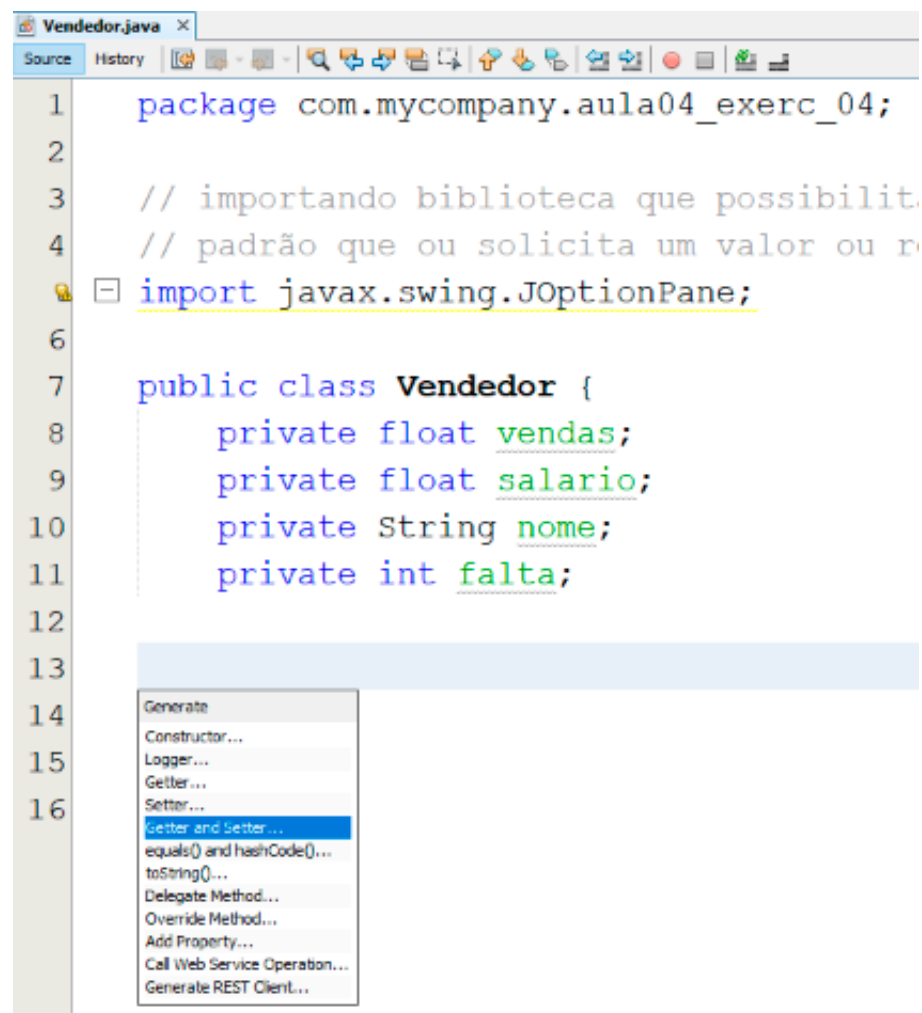

Exercício 4



```
Vendedor.java x
Source History
1 package com.mycompany.aula04_exerc_04;
2
3 // importando biblioteca que possibilita a criação de uma caixa de dialogo
4 // padrão que ou solicita um valor ou retorna uma informação
5 import javax.swing.JOptionPane;
6
7 public class Vendedor {
8     private float vendas;
9     private float salario;
10    private String nome;
11    private int falta;
12 }
```

Exercício 4

Vamos criar os métodos de acesso



```
1 package com.mycompany.aula04_exerc_04;
2
3 // importando biblioteca que possibilita
4 // padrão que ou solicita um valor ou retorna
5 import javax.swing.JOptionPane;
6
7 public class Vendedor {
8     private float vendas;
9     private float salario;
10    private String nome;
11    private int falta;
12
13
14
15
16
```

- Generate
- Constructor...
- Logger...
- Getter...
- Setter...
- Getter and Setter...
- equals() and hashCode()...
- toString()...
- Delegate Method...
- Override Method...
- Add Property...
- Call Web Service Operation...
- Generate REST Client...

Exercício 4

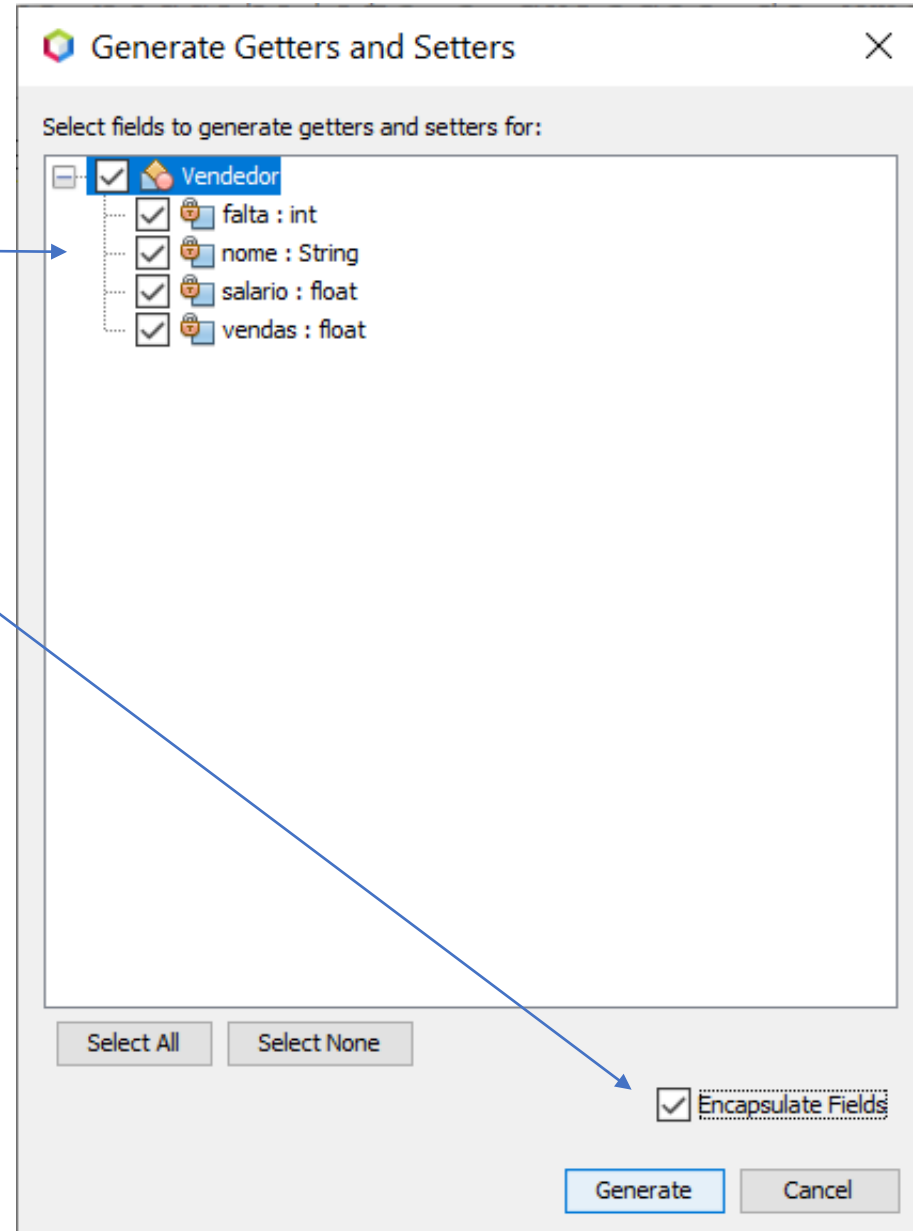
Getter and Setter ...

```
Vendedor.java
Source History
1 package com.mycompany.aula04_exerc_04;
2
3 // importando biblioteca que possibilita
4 // padrão que ou solicita um valor ou retorna
5 import javax.swing.JOptionPane;
6
7 public class Vendedor {
8     private float vendas;
9     private float salario;
10    private String nome;
11    private int falta;
12
13
14
15
16
```

- Generate
- Constructor...
- Logger...
- Getter...
- Setter...
- Getter and Setter...
- equals() and hashCode()...
- toString()...
- Delegate Method...
- Override Method...
- Add Property...
- Call Web Service Operation...
- Generate REST Client...

Exercício 4

Selecione os atributos e
O encapsulamento



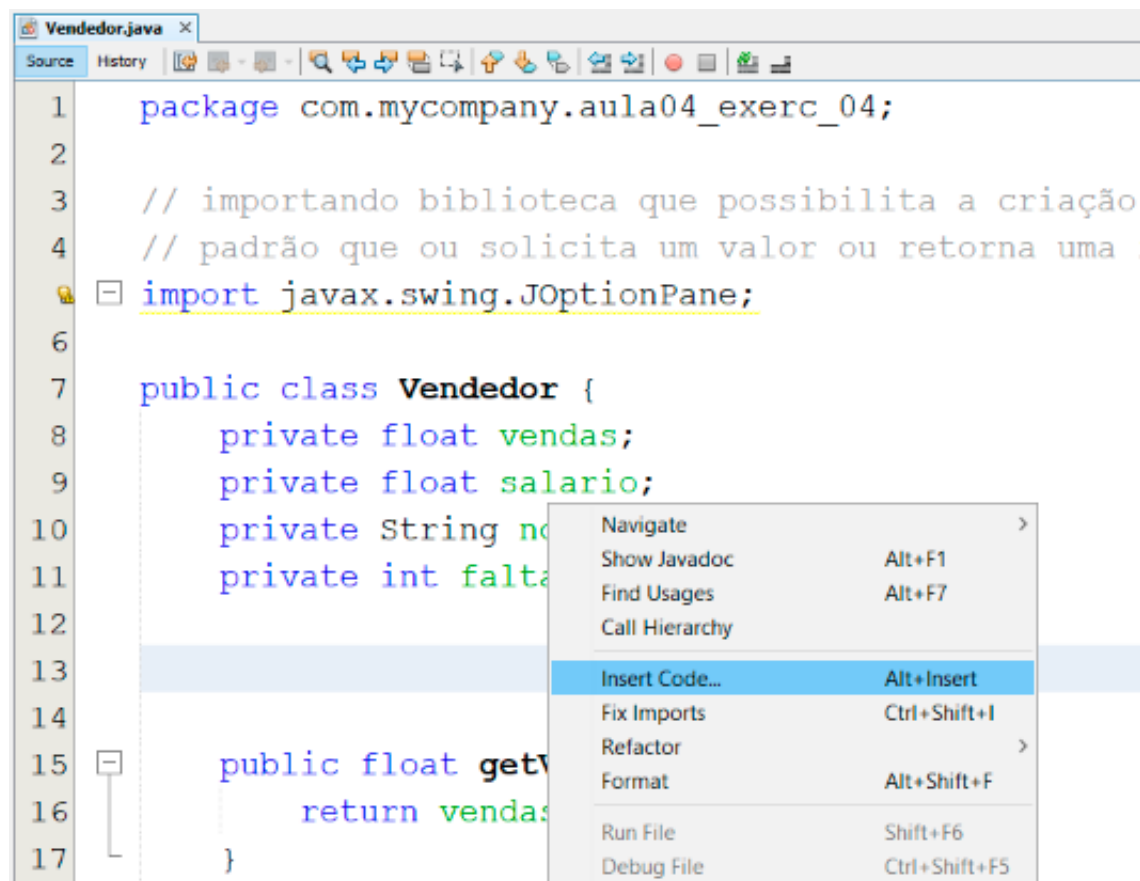
Exercício 4

Pronto, métodos criados ...

```
Vendedor.java x
Source History
7      public class Vendedor {
8          private float vendas;
9          private float salario;
10         private String nome;
11         private int falta;
12
13         public float getVendas() {
14             return vendas;
15         }
16
17         public void setVendas(float vendas) {
18             this.vendas = vendas;
19         }
20
21         public float getSalario() {
22             return salario;
23         }
24
25         public void setSalario(float salario) {
26             this.salario = salario;
27         }
28     }
```

Exercício 4

Vamos criar o método construtor



```
1 package com.mycompany.aula04_exerc_04;
2
3 // importando biblioteca que possibilita a criação
4 // padrão que ou solicita um valor ou retorna uma :
5 import javax.swing.JOptionPane;
6
7 public class Vendedor {
8     private float vendas;
9     private float salario;
10    private String nome;
11    private int faltas;
12
13
14
15    public float getVendas() {
16        return vendas;
17    }
```

Navigate	>
Show Javadoc	Alt+F1
Find Usages	Alt+F7
Call Hierarchy	
Insert Code...	Alt+Insert
Fix Imports	Ctrl+Shift+I
Refactor	>
Format	Alt+Shift+F
Run File	Shift+F6
Debug File	Ctrl+Shift+F5

Exercício 4

- Generate
- Constructor...**
- Logger...
- equals() and hashCode()...
- toString()...
- Delegate Method...
- Override Method...
- Add Property...
- Call Web Service Operation...
- Generate REST Client...

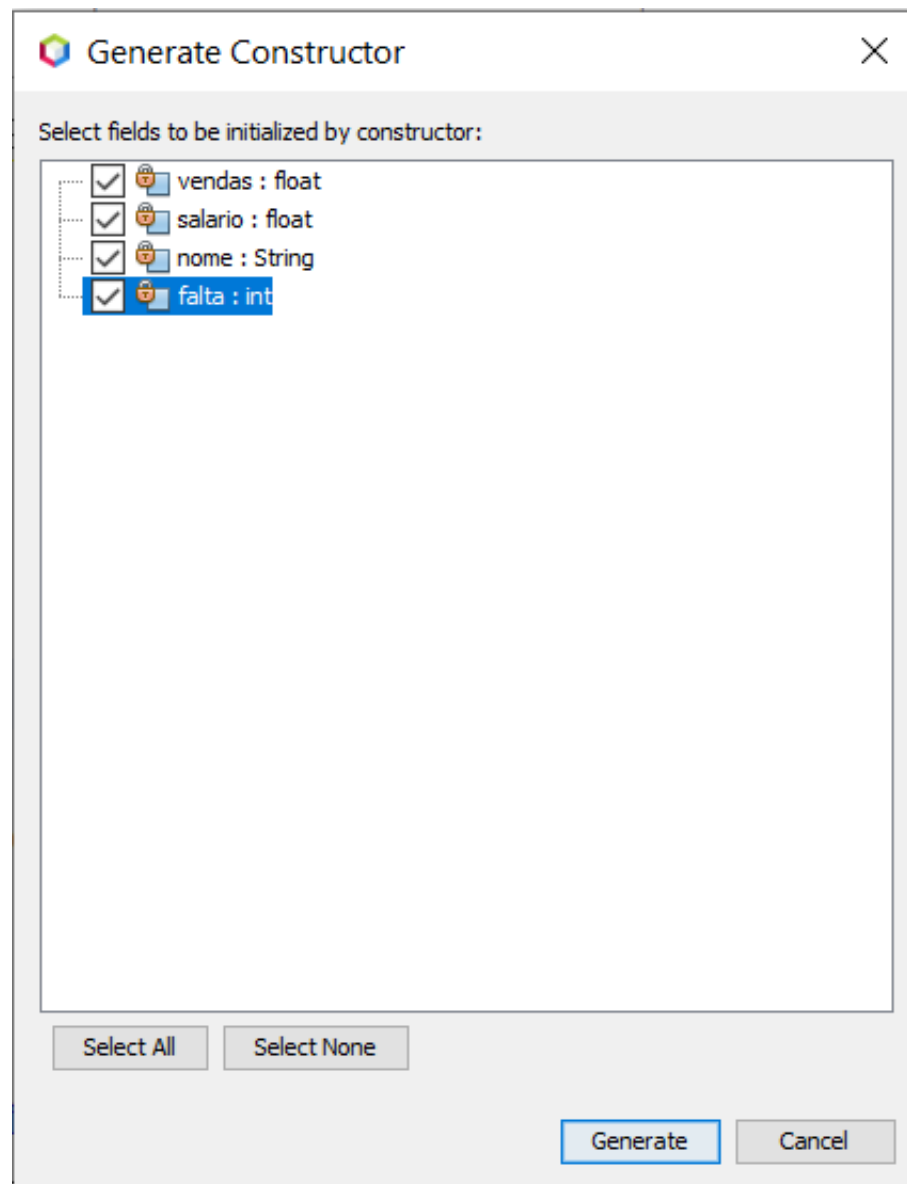
public void

t

v

Exercício 4

Selecione os parâmetros necessários para construir um objeto




Exercício 4

```
Vendedor.java x
Source History
7 public class Vendedor {
8     private float vendas;
9     private float salario;
10    private String nome;
11    private int falta;
12
13    public Vendedor(float vendas, float salario, String nome, int falta) {
14        this.vendas = vendas;
15        this.salario = salario;
16        this.nome = nome;
17        this.falta = falta;
18    }
```

Exercício 4

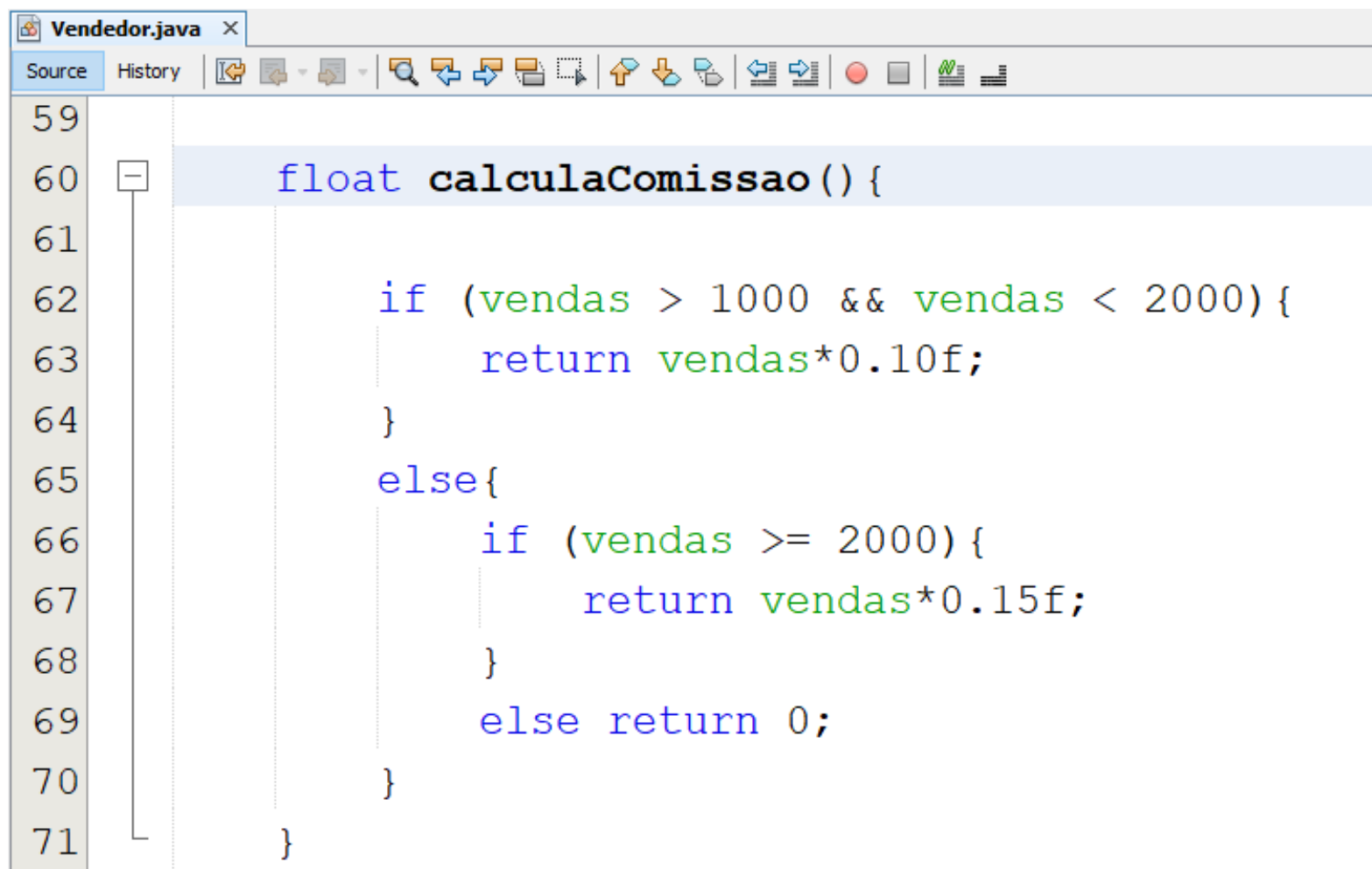
Vamos criar um método para exibir os dados



```
48 public void setFalta(int falta) {
49     this.falta = falta;
50 }
51
52 void imprimirDados() {
53     JOptionPane.showMessageDialog(null, "Dados do Vendedor: " +
54         "\nNome: " + nome +
55         "\nVendas: " + vendas +
56         "\nSalário: " + salario +
57         "\nFaltas: " + falta);
58 }
```

Exercício 4

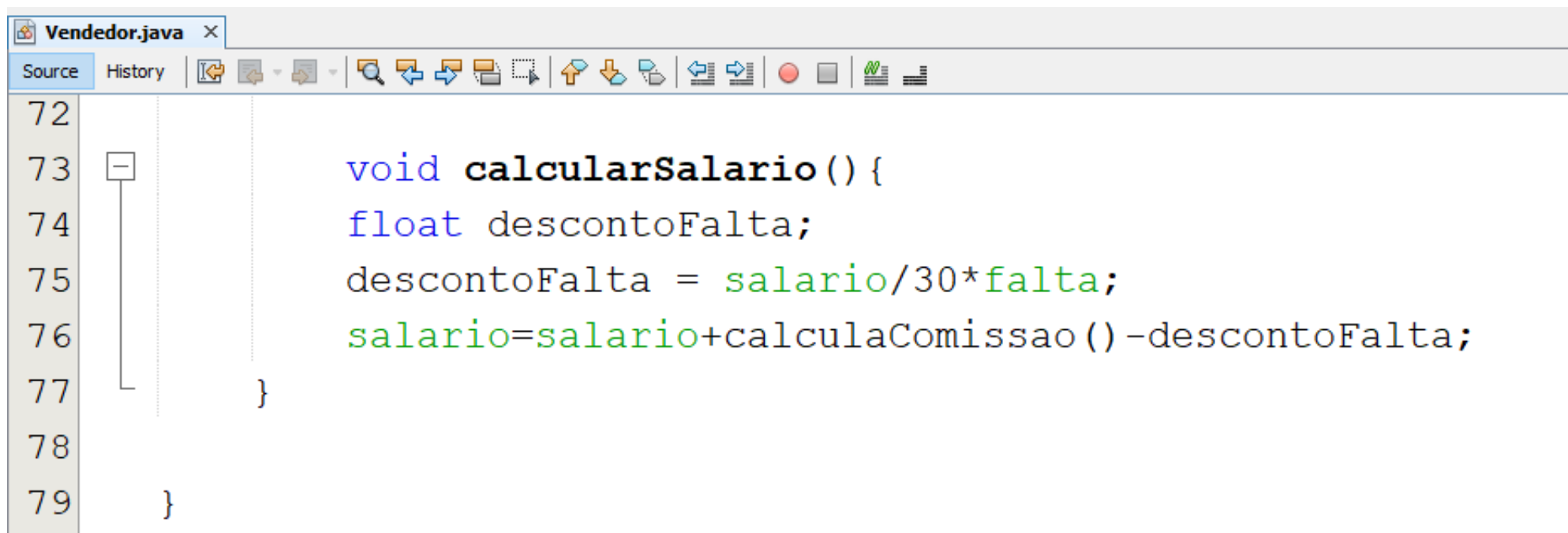
Criar um método para calcular a comissão



```
Vendedor.java x
Source History
59
60 float calculaComissao() {
61
62     if (vendas > 1000 && vendas < 2000) {
63         return vendas*0.10f;
64     }
65     else{
66         if (vendas >= 2000) {
67             return vendas*0.15f;
68         }
69         else return 0;
70     }
71 }
```

Exercício 4

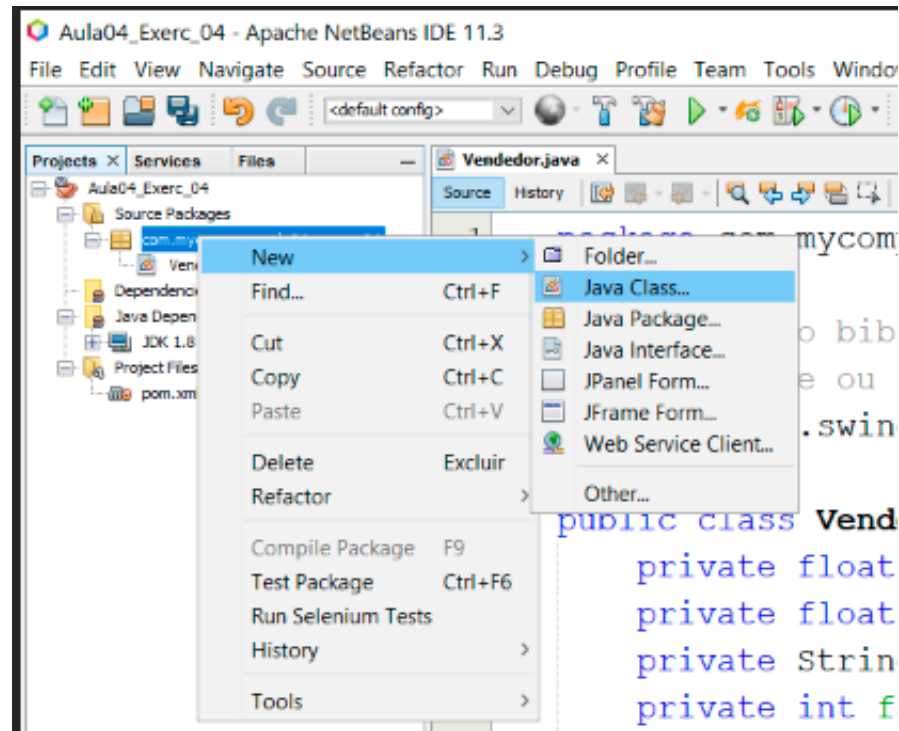
Criar um método para calcular o Salário



```
Vendedor.java x
Source History
72
73 void calcularSalario() {
74     float descontoFalta;
75     descontoFalta = salario/30*falta;
76     salario=salario+calculaComissao()-descontoFalta;
77 }
78
79 }
```

Exercício 4

Vamos criar uma classe para testar



Exercício 4

Classe TesteVendedor

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

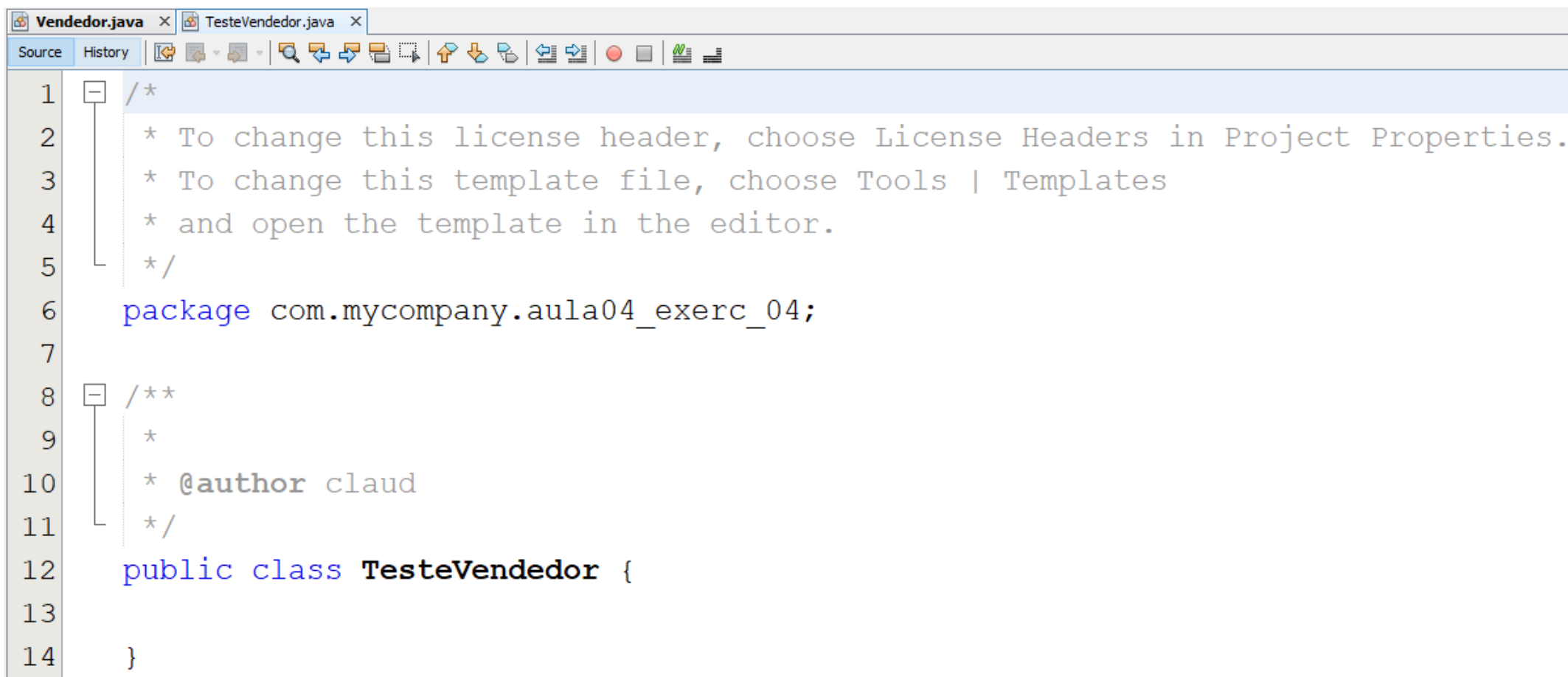
Package:

Created File:

< Back Next > Finish Cancel Help

Exercício 4

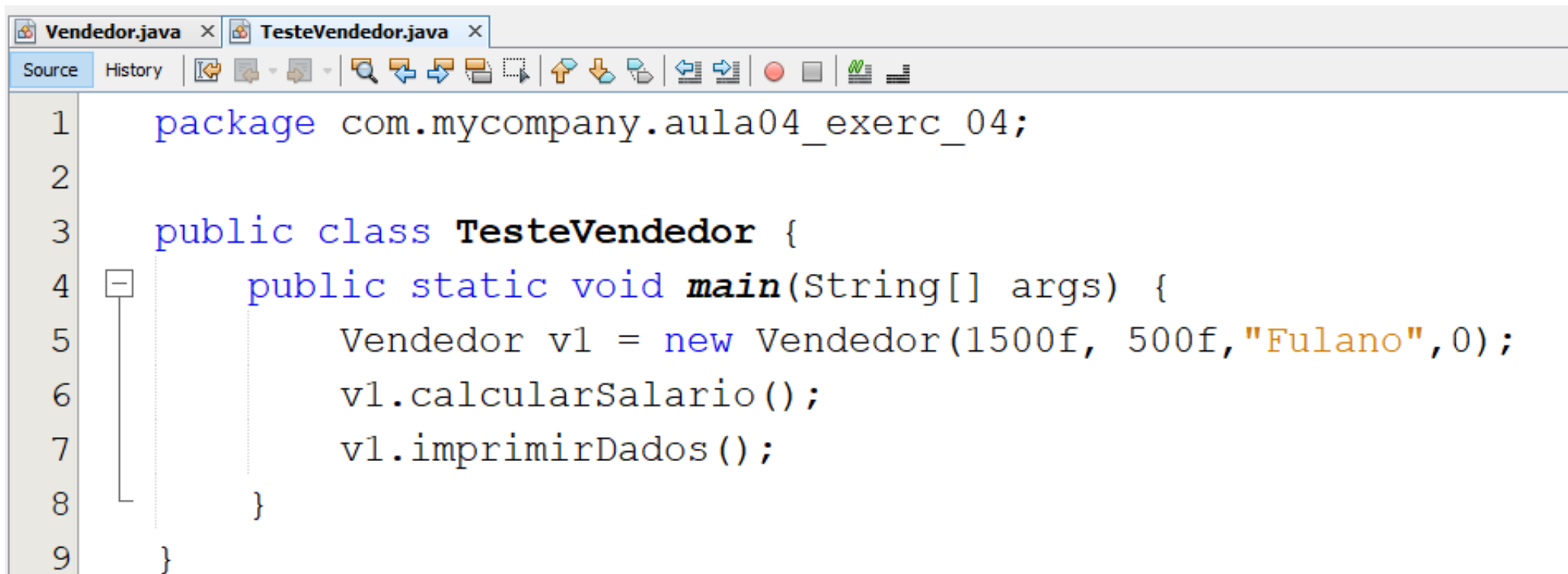
Vamos limpar o código



```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package com.mycompany.aula04_exerc_04;
7
8  /**
9   *
10   * @author claud
11   */
12  public class TesteVendedor {
13
14  }
```

Exercício 4

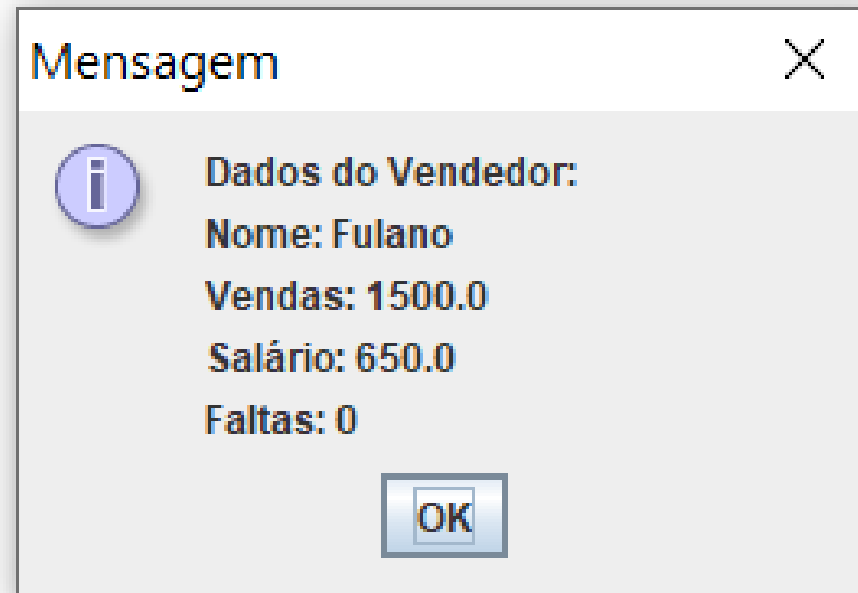
Vamos fazer um teste com dados fixos...



```
1 package com.mycompany.aula04_exerc_04;
2
3 public class TesteVendedor {
4     public static void main(String[] args) {
5         Vendedor v1 = new Vendedor(1500f, 500f, "Fulano", 0);
6         v1.calcularSalario();
7         v1.imprimirDados();
8     }
9 }
```


Exercício 4

Resultado!





Exercício

Vamos analisar o seguinte problema e implementar uma solução para o mesmo usando Java





Exercício

- Uma empresa localizada na zona leste de São Paulo consome 100 KWh por mês;
- Após ter negociado um contrato com uma grande empresa, estima-se que nos próximos 12 meses terá um acréscimo mensal na sua conta de 100 KWh;
- Então o diretor da empresa solicitou para o analista que elaborasse um sistema que através dele fosse capaz de proporcionar uma estimativa financeira de quanto será o custo de energia elétrica por mês;



Exercício

- Para tanto será preciso entender as seguintes regras do negócios:

A. $\text{FORNECIMENTO} = \text{Consumo} * \text{tarifa, ou seja KWh} \times \mathbf{0.28172};$

Exercício

- B. **ICMS** = Se o consumo for até 200 KWh a alíquota é de **12%** e o fator de multiplicação é = **0.136363**.
Caso o consumo seja superior a 200 KWh a alíquota é de **25%** e o fator de multiplicação é = **0.333333**.

O valor do ICMS é o fator de multiplicação * o valor do fornecimento;

Imposto sobre Circulação de Mercadorias e Prestação de Serviços



Exercício

- c. **COFINS** = A alíquota é de **5,033%**. Se o consumo for **até 200 KWh** o fator de multiplicação é = **0.0614722** e caso o consumo seja **superior a 200 KWh** o fator de multiplicação é = **0.0730751**.

O valor do COFINS é o fator de multiplicação * o valor do fornecimento;

Contribuição para o Financiamento da Seguridade Social é uma contribuição federal brasileira



Exercício

- D. **PIS/PASESP** = A alíquota é de **1,0927%**. Se o consumo for até **200 KWh** o fator de multiplicação é = **0.013346** e caso o consumo seja **superior a 200 KWh** o fator de multiplicação é = **0.0158651**.

O valor do PIS/PASESP é o fator de multiplicação * o valor do fornecimento;

O Programa de Integração Social e o Programa de Formação do Patrimônio do Servidor Público



PREVIDÊNCIA SOCIAL

Exercício

- E. **ICMS sobre COFINS** = é a multiplicação dos fatores do COFINS, ICMS e do valor do fornecimento;
- F. **ICMS sobre PIS/PASES** = é a multiplicação dos fatores do PIS/PASESP, ICMS e do valor do fornecimento;
- G. **FATURA** = (Fornecimento + ICMS + COFINS + PIS/PASESP + ICMS_COFINS + ICMS _ PIS_PASESO);

Exercício

Fator de multiplicação é gerado pela cobrança ilegal de imposto sobre o imposto por parte da concessionária de energia elétrica.



$$1 \text{ CV} = 736 \text{ W}$$
$$P_{\text{mec}} = 736 \times 10 = 7360 \text{ W}$$

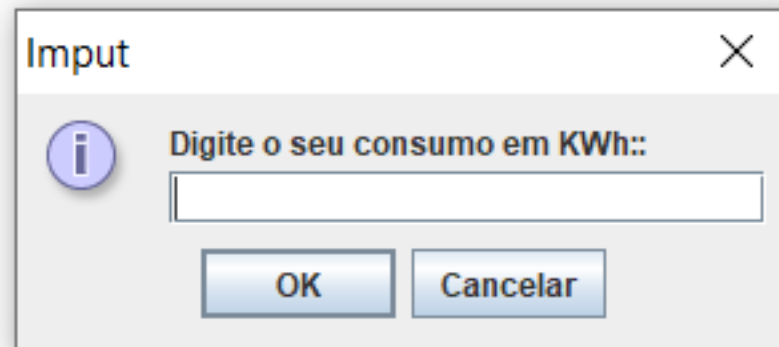
$$\eta(\%) = \frac{P_{\text{mec}}}{P_{\text{elétrica}}} \times 100$$

$$P_{\text{elétrica}} = \frac{7360}{90} \times 100 = 8177,8 \text{ W}$$


Exercício

Nossa tela:

Efetuar o Calculo da Energia Elétrica



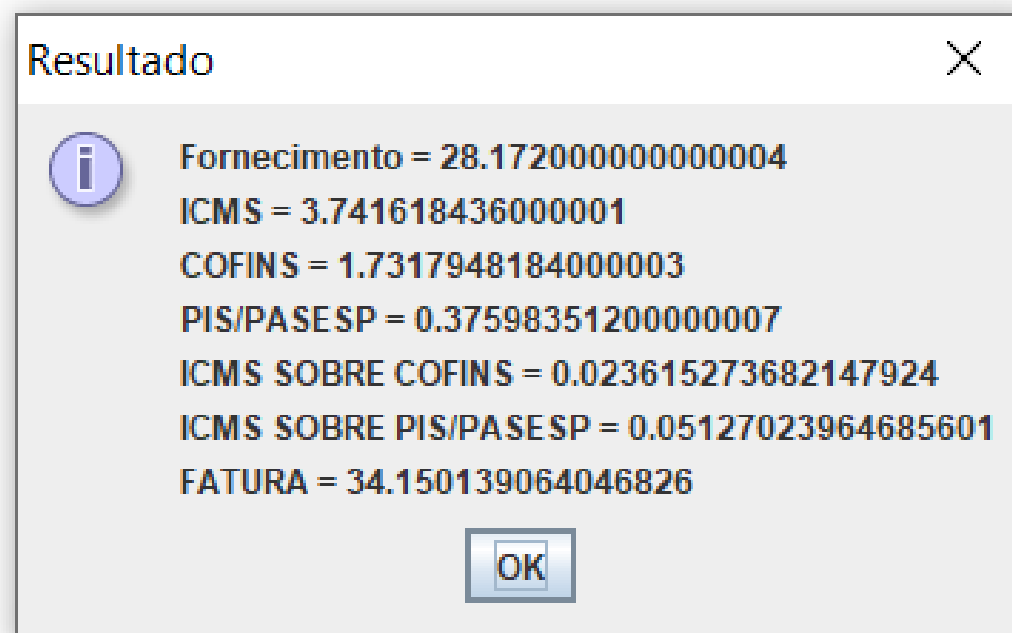
Input

 Digite o seu consumo em KWh:

OK Cancelar

Exercício

Nossa tela com Resultado:



“Nada na vida deve
ser temido, somente
compreendido.
Agora é hora de
compreender mais
para temer menos”



Marié Curié

Obrigado!

Se precisar ...

Prof. Claudio Benossi

Claudio.benossi@fatec.sp.gov.br

