



# PROGRAMAÇÃO I

## INTRODUÇÃO À ORIENTAÇÃO A OBJETOS

Prof. Dr. Daniel Caetano

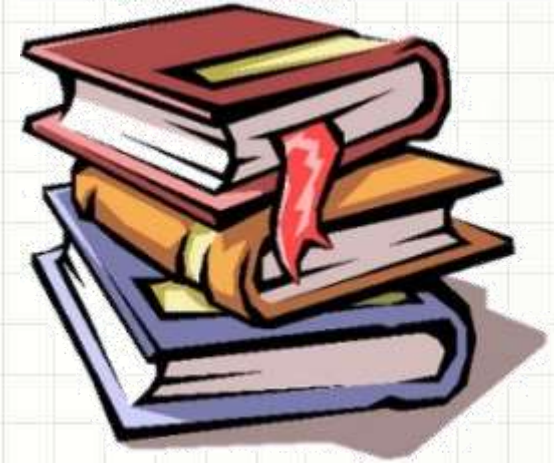
2017 - 1

# Objetivos

- Compreender os conceitos classe e objeto
- Compreender a função dos métodos e atributos e o conceito de encapsulamento
- Compreender os construtores
- Compreender a diferença dos especificadores de acesso
- Aplicar os conceitos na prática



# Material de Estudo



---

## Material

## Acesso ao Material

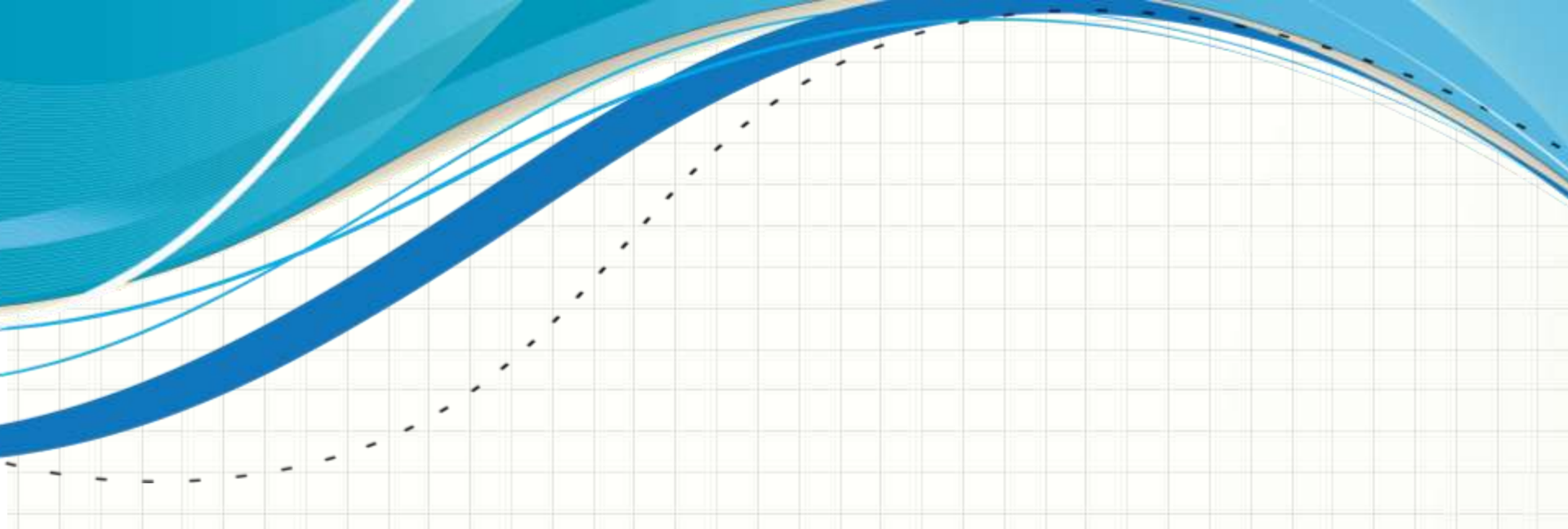
Apresentação

<http://www.caetano.eng.br/>  
(Programação I – Aula 3)

Material Didático

Programação I – Págs 48 a 59

---



# **RETOMANDO A ESTRUTURA DE UM PROGRAMA JAVA**

# Estrutura do Código

projeto.jar

**Pacote1**

**Projeto.java**

main

metodo1

**Classe1.java**

main

metodo1

**Pacote2**

**Classe3.java**

main

metodoA

**Classe4.java**

main

metodoX



# **CLASSES EM JAVA**

# O que são Classes?

- Inicialmente, não nos prendamos em definições formais
- Como foi dito anteriormente, classes são como pequenos programas, que podem ser considerados novos tipos de dados
- Como todo programa, uma classe é composta por algumas variáveis, que chamamos de **atributos** e algumas funções que chamaremos de **métodos**.

# O que são Classes?

- Podemos imaginar uma classe assim:



**Atributos**



# O que são Classes?

- Podemos imaginar uma classe assim:



**Privados**  
**ou Públicos**  
**Atributos**

# O que são Classes?

- Podemos imaginar uma classe assim:



**Métodos**

# O que são Classes?

- Podemos imaginar uma classe assim:

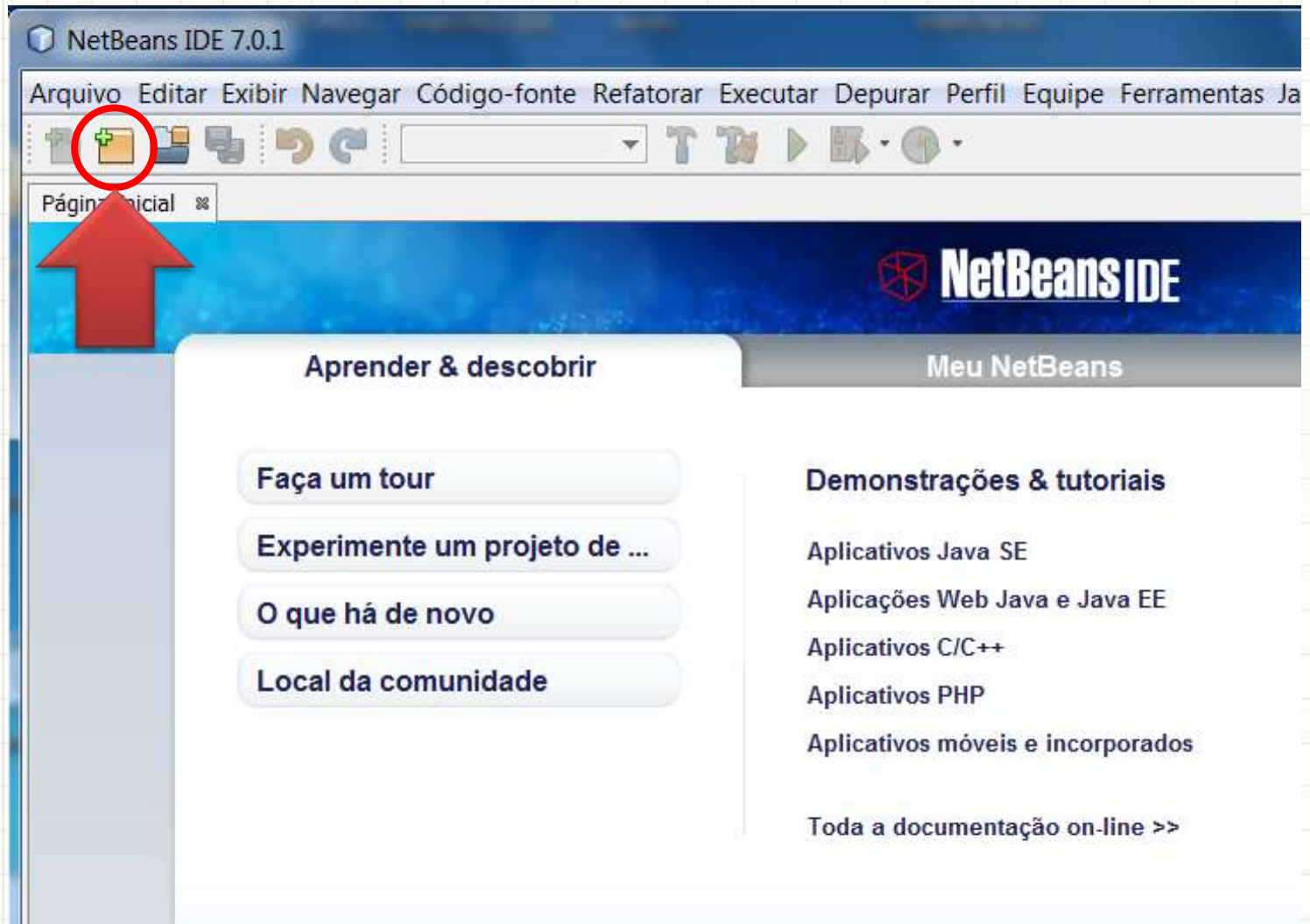


**Privados  
ou Públicos**

**Métodos**

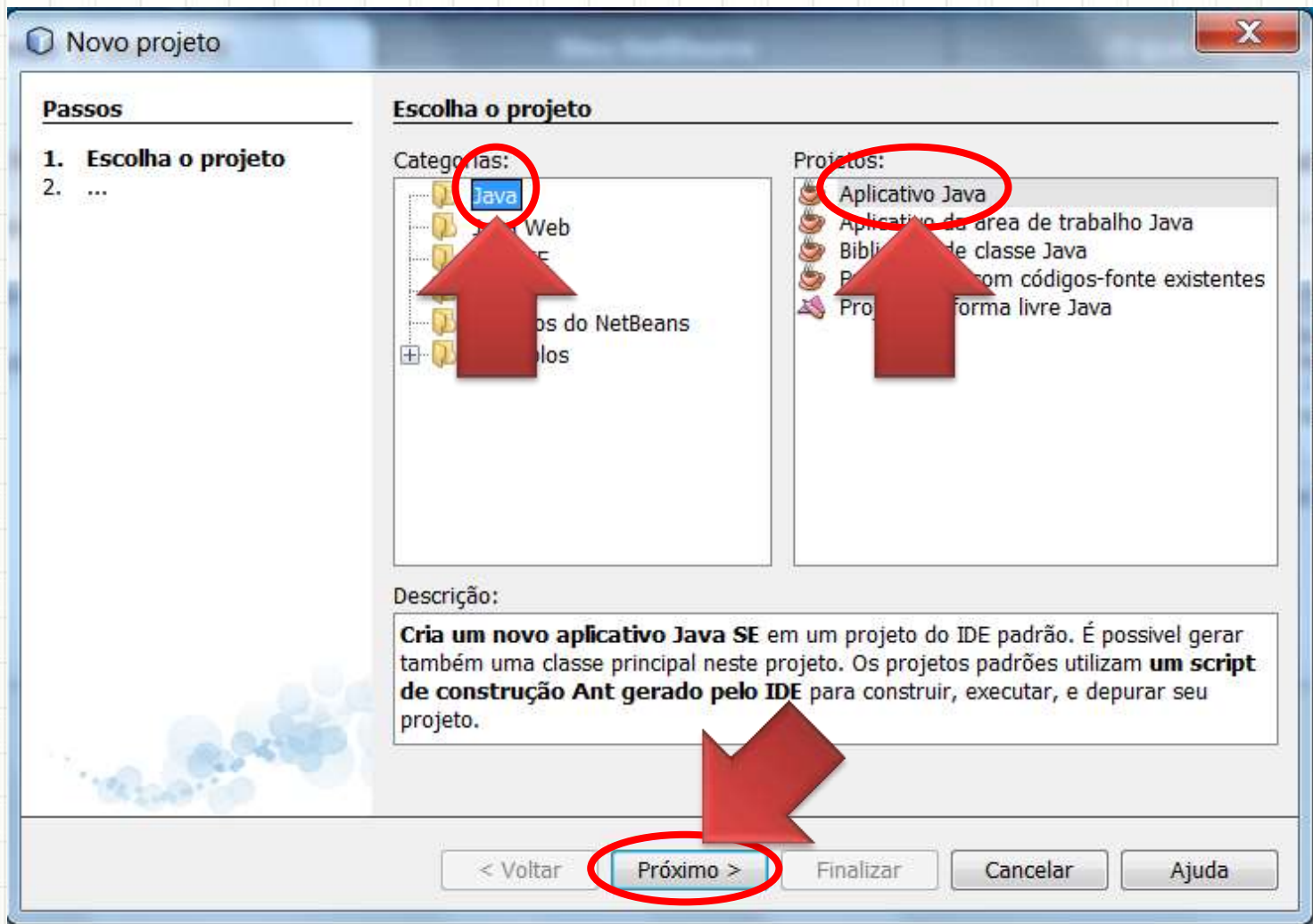
# Definindo uma classe

- Iniciaremos criando um projeto



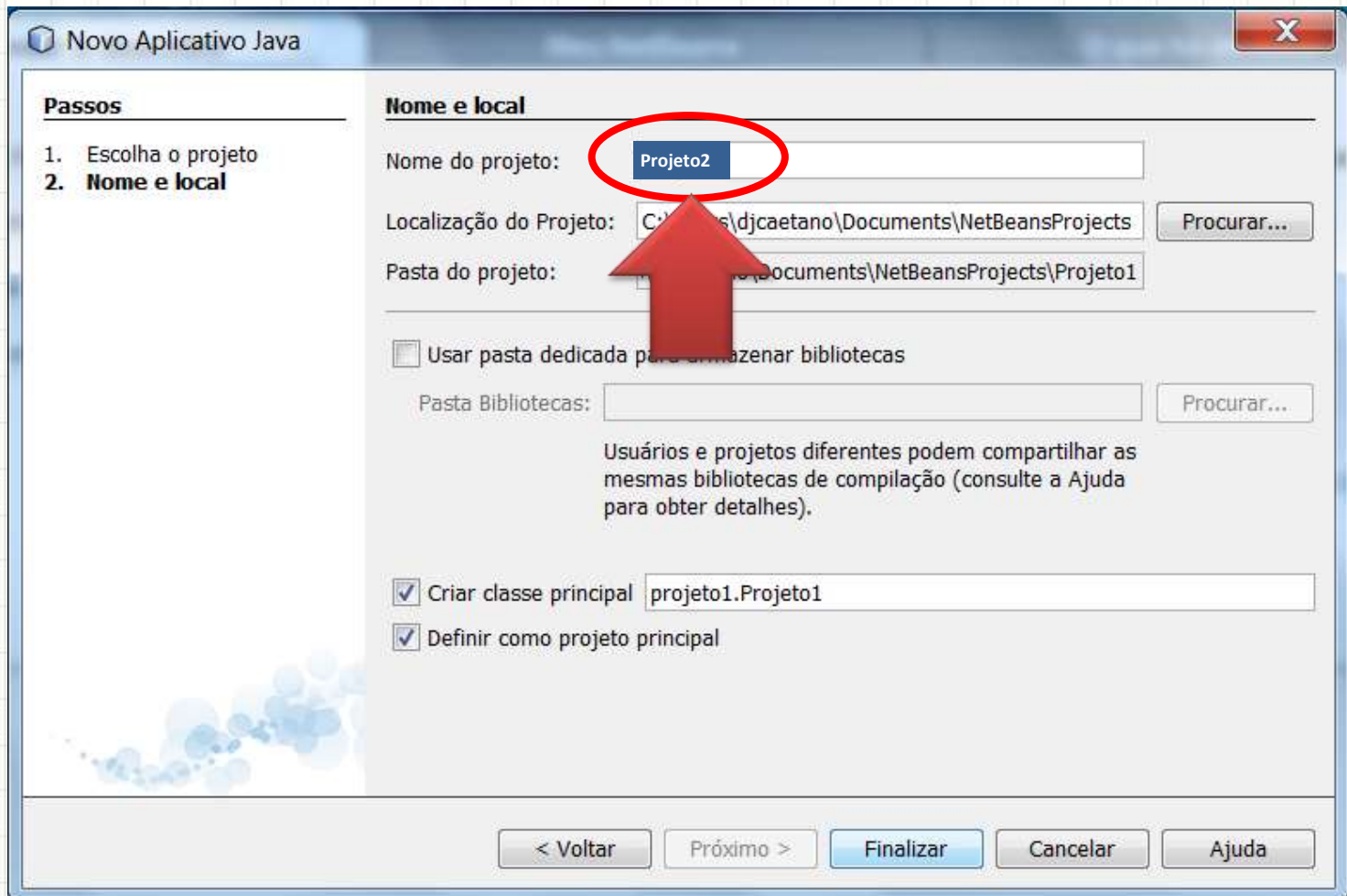
# Definindo uma classe

- Escolha o tipo: Java e Aplicativo Java



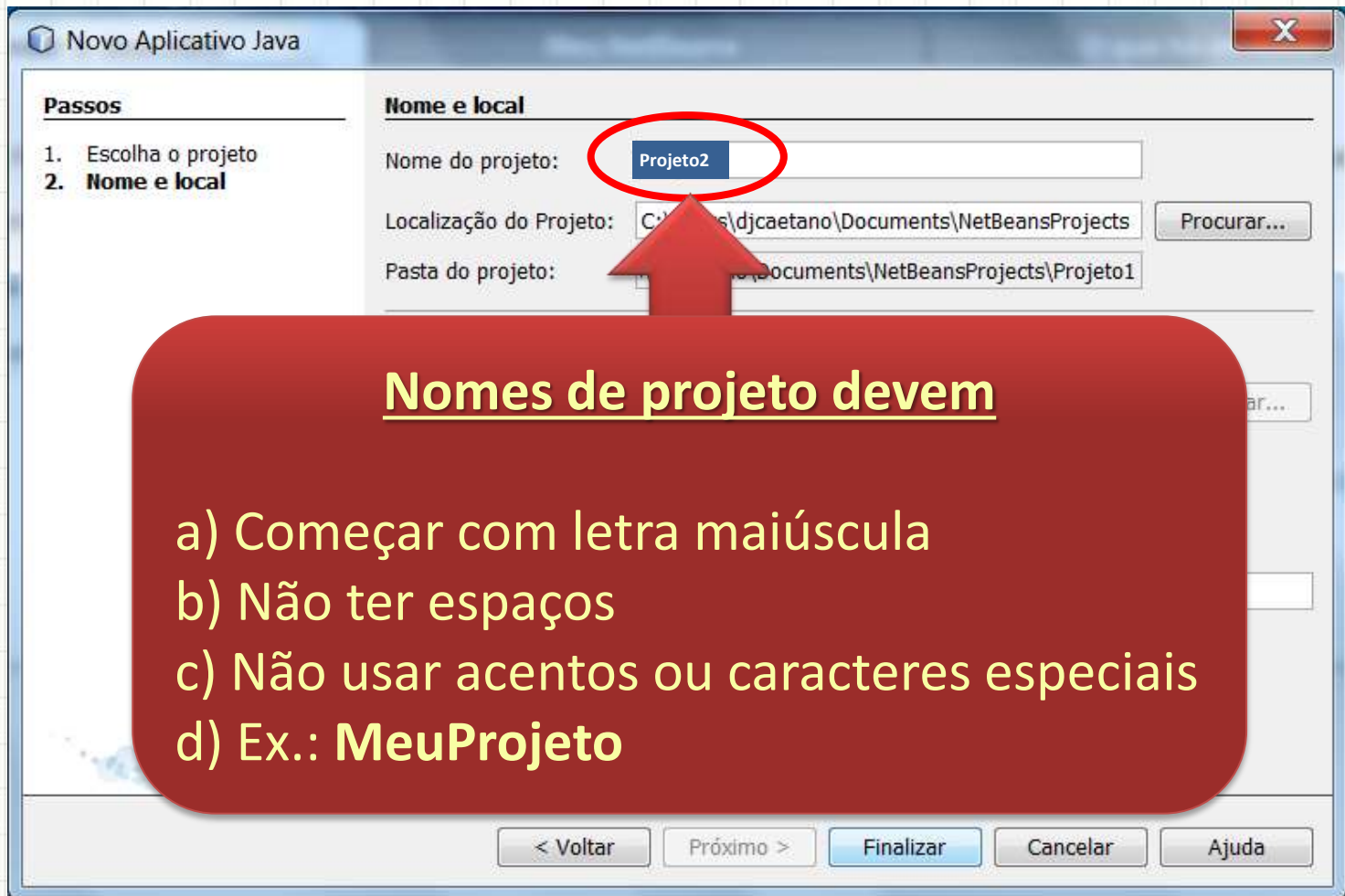
# Definindo uma classe

- Dê um nome ao seu projeto: ex.: **Projeto2**



# Definindo uma classe

- Dê um nome ao seu projeto: ex.: **Projeto2**



**Novo Aplicativo Java**

**Passos**

1. Escolha o projeto
2. **Nome e local**

**Nome e local**

Nome do projeto:

Localização do Projeto: C:\Users\djcaetano\Documents\NetBeansProjects

Pasta do projeto: C:\Users\djcaetano\Documents\NetBeansProjects\Projeto1

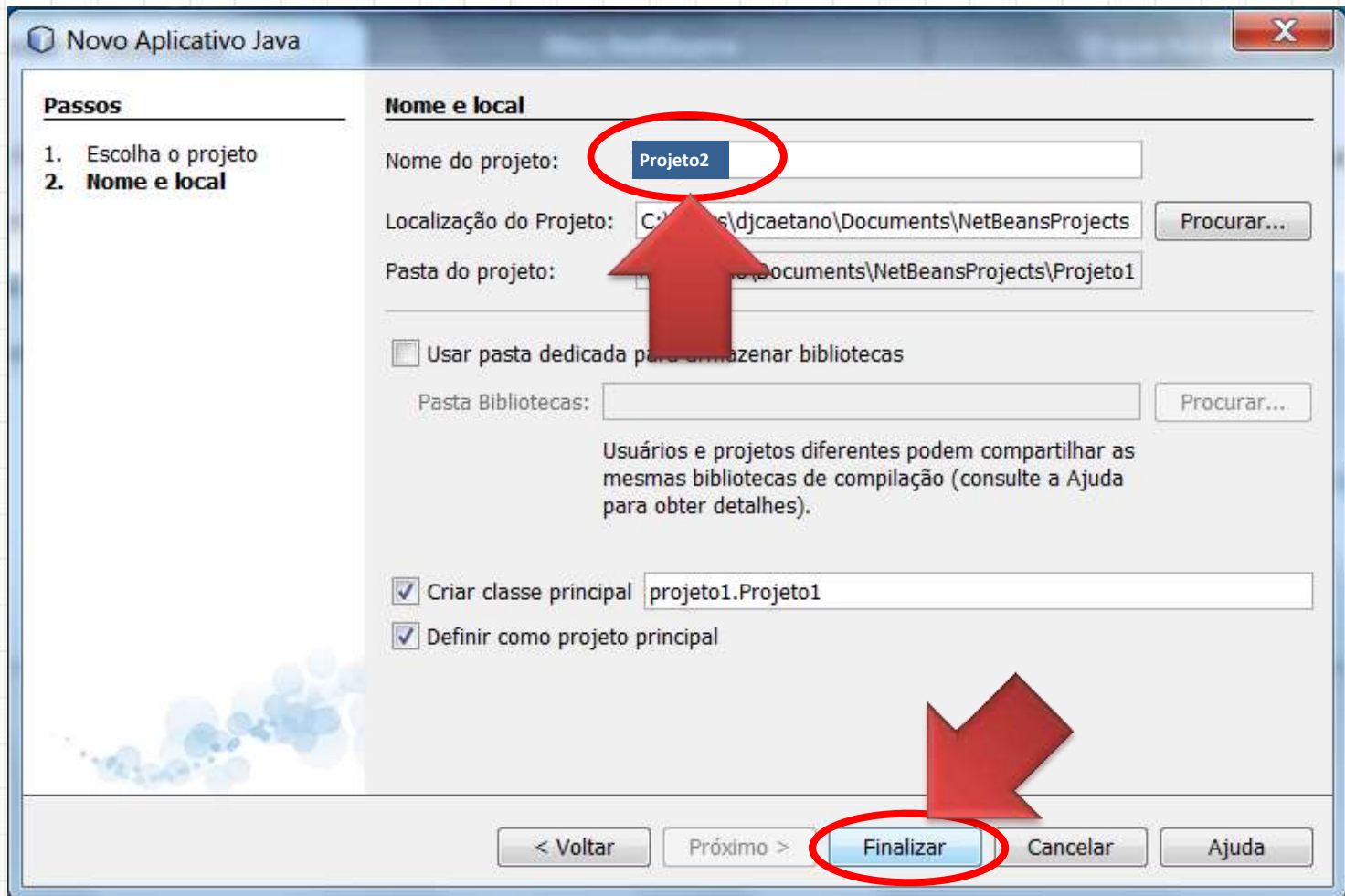
**Nomes de projeto devem**

- a) Começar com letra maiúscula
- b) Não ter espaços
- c) Não usar acentos ou caracteres especiais
- d) Ex.: **MeuProjeto**

< Voltar   Próximo >   Finalizar   Cancelar   Ajuda

# Definindo uma classe

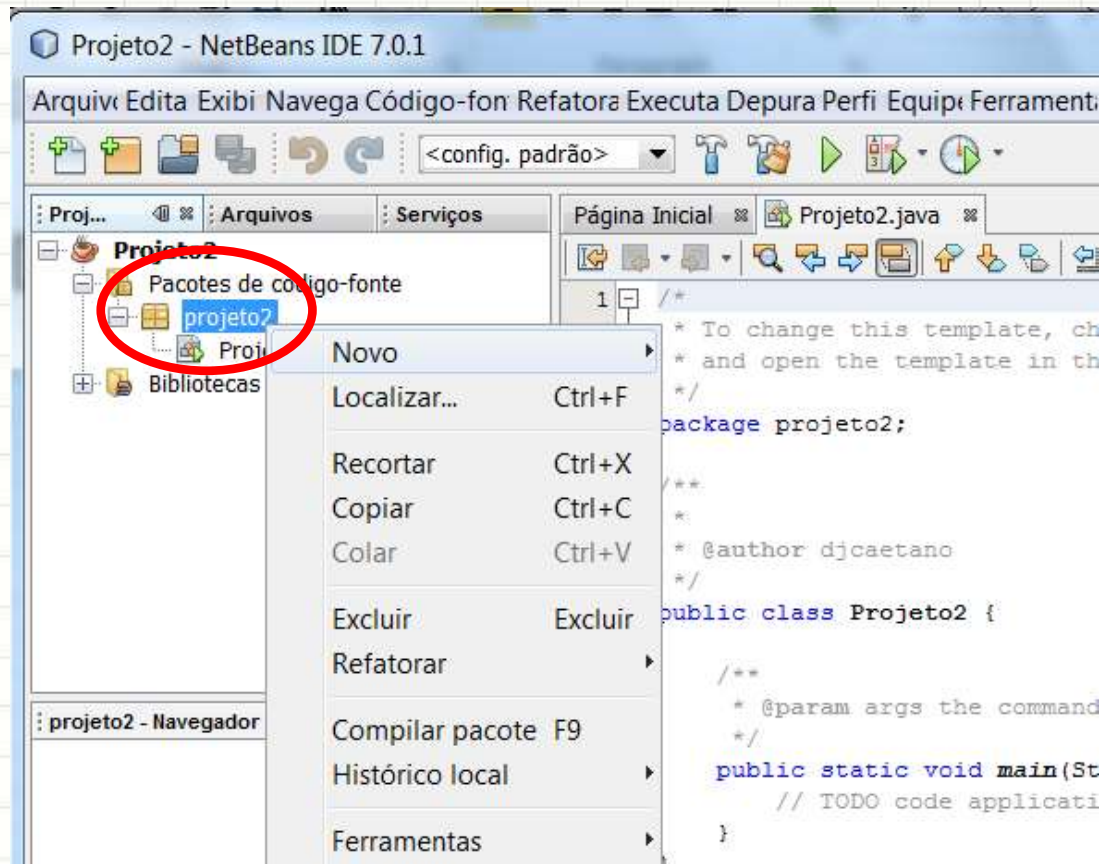
- Dê um nome ao seu projeto: ex.: **Projeto2**





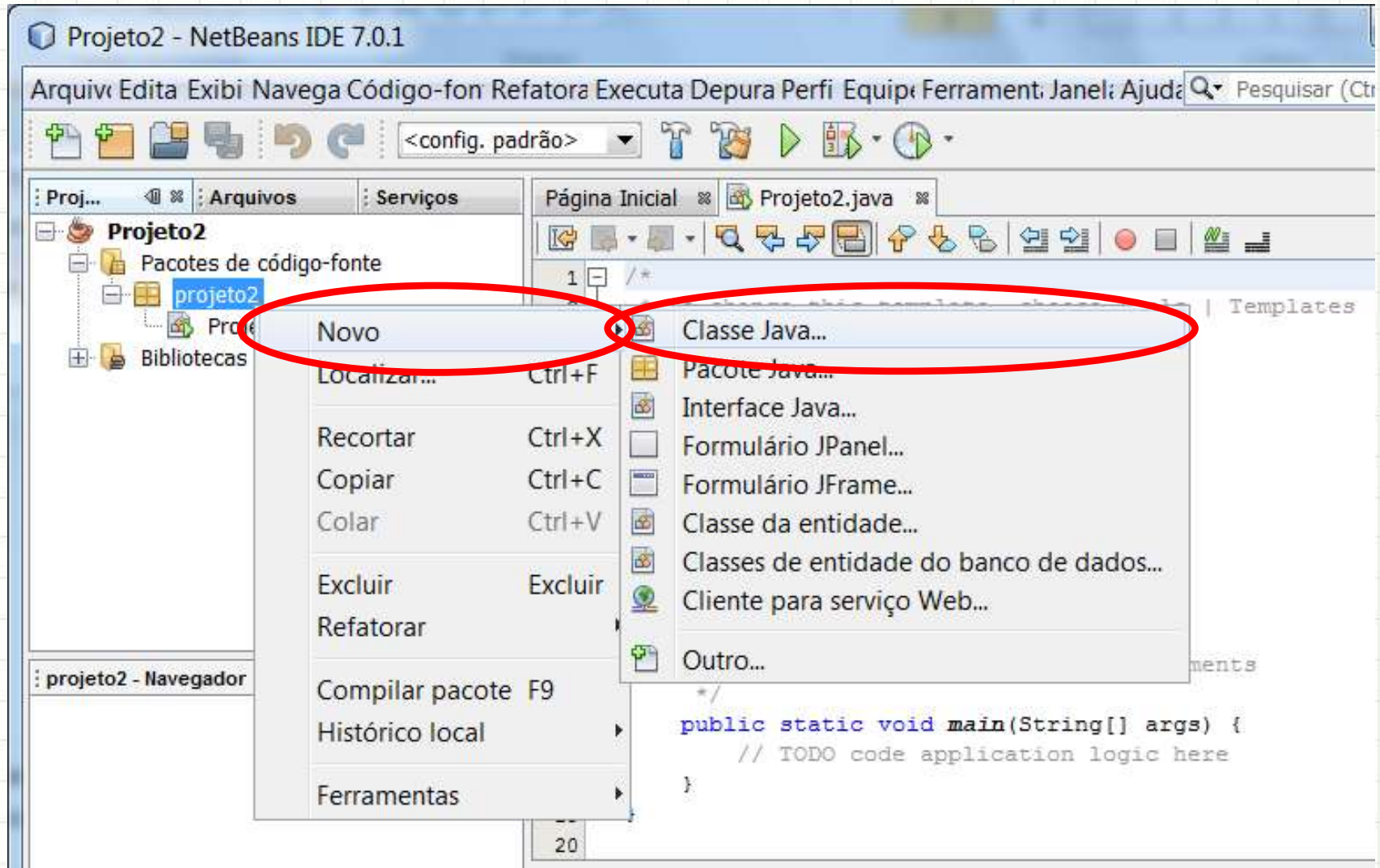
# Definindo uma Classe

- As classes devem ficar dentro de pacotes; Assim, clique com o botão direito no ícone do pacote que tem o nome do projeto (**projeto2**)



# Definindo uma Classe

- Agora selecione **novo > classe java**



# Definindo uma Classe

- Agora dê um nome à classe: **Produto**

Novo Classe Java

**Passos**

1. Escolha o tipo de arquivo
2. **Nome e local**

**Nome e local**

Nome da classe: **Produto**

Projeto: P

Localização: Pa e código-fonte

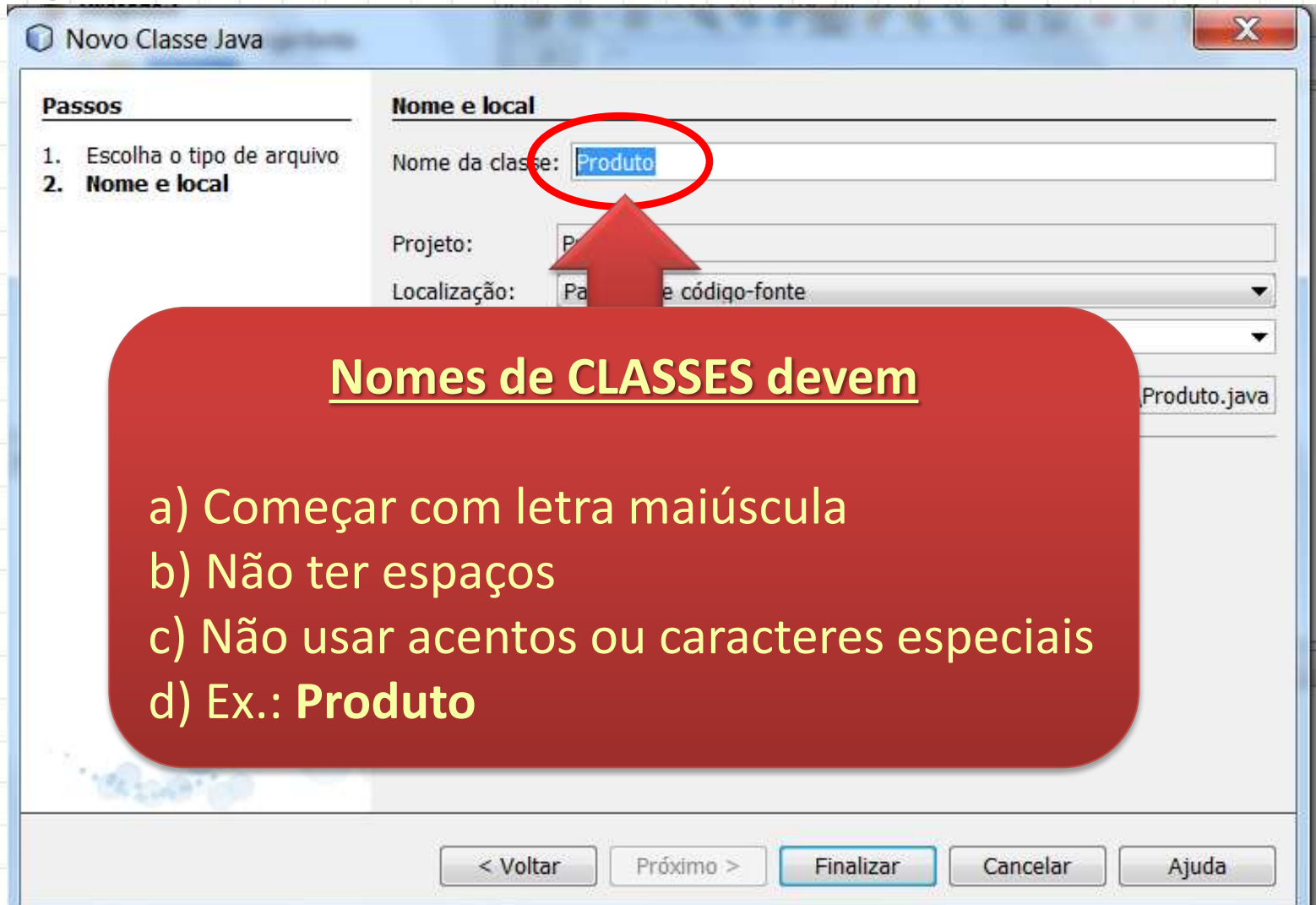
Pacote: pr

Arquivo criado: c:\Documents\NetBeansProjects\Projeto2\src\projeto2\Produto.java

< Voltar   Próximo >   **Finalizar**   Cancelar   Ajuda

# Definindo uma Classe

- Agora dê um nome à classe: **Produto**

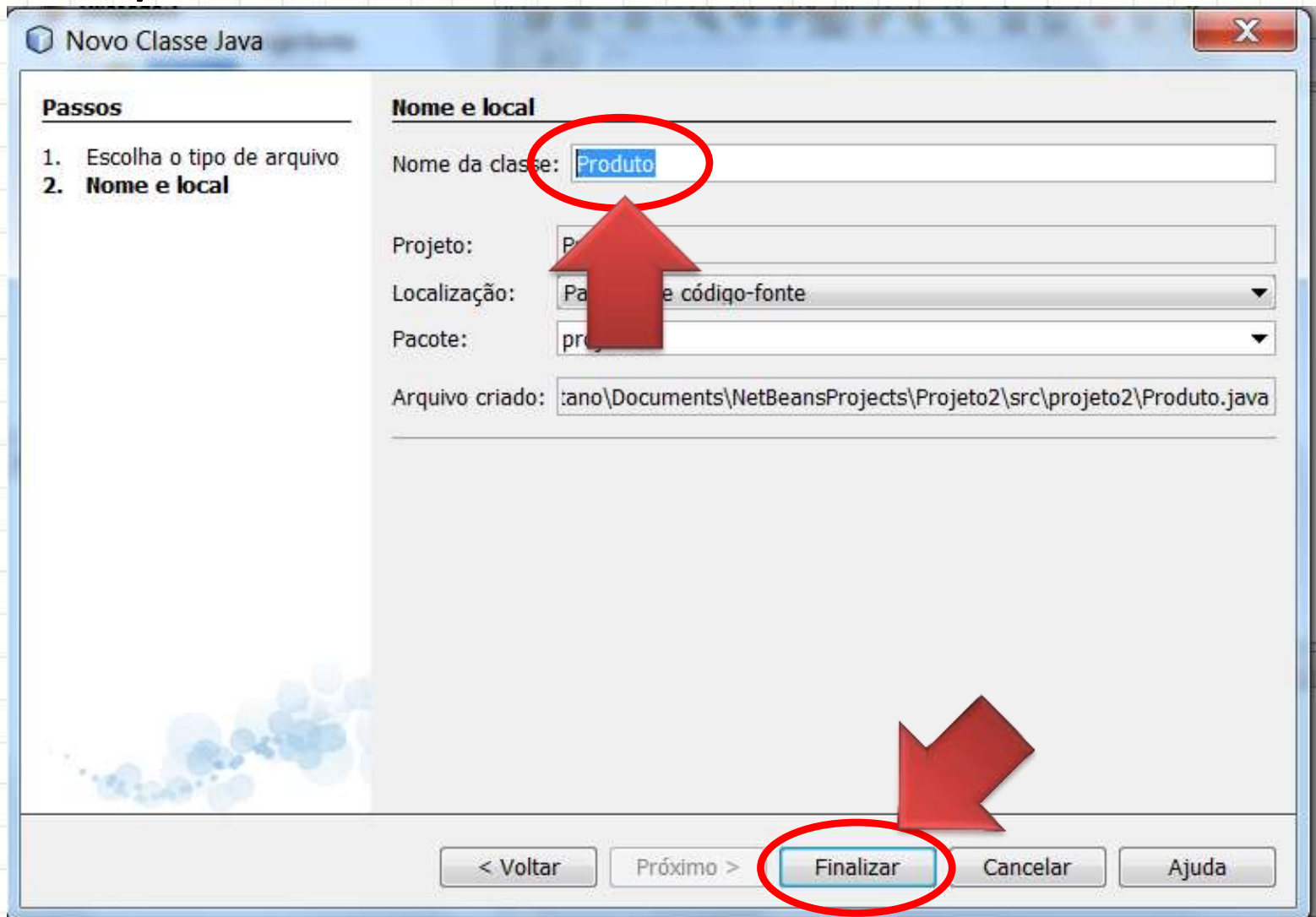


**Nomes de CLASSES devem**

- a) Começar com letra maiúscula
- b) Não ter espaços
- c) Não usar acentos ou caracteres especiais
- d) Ex.: **Produto**

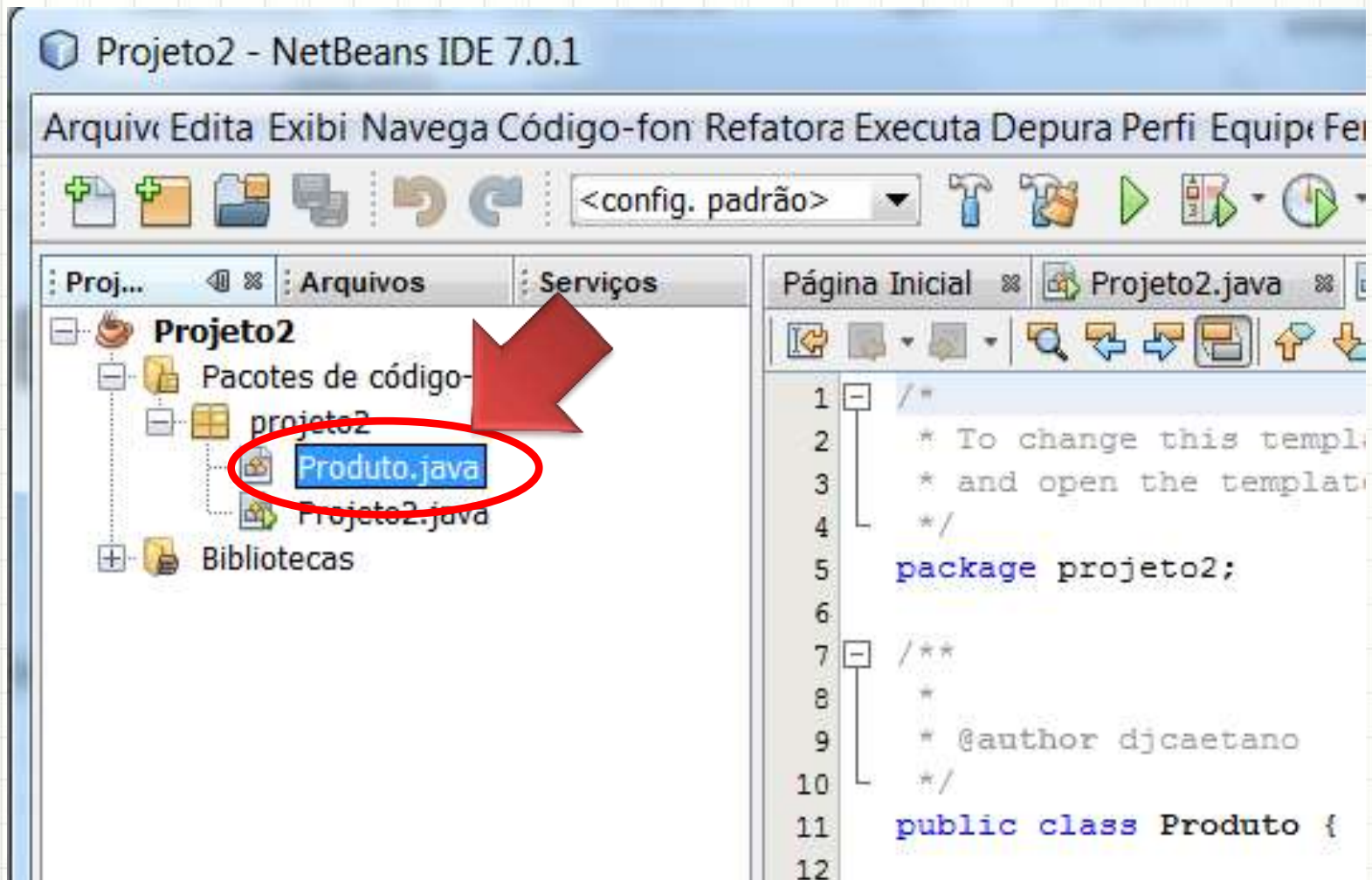
# Definindo uma Classe

- E clique em finalizar



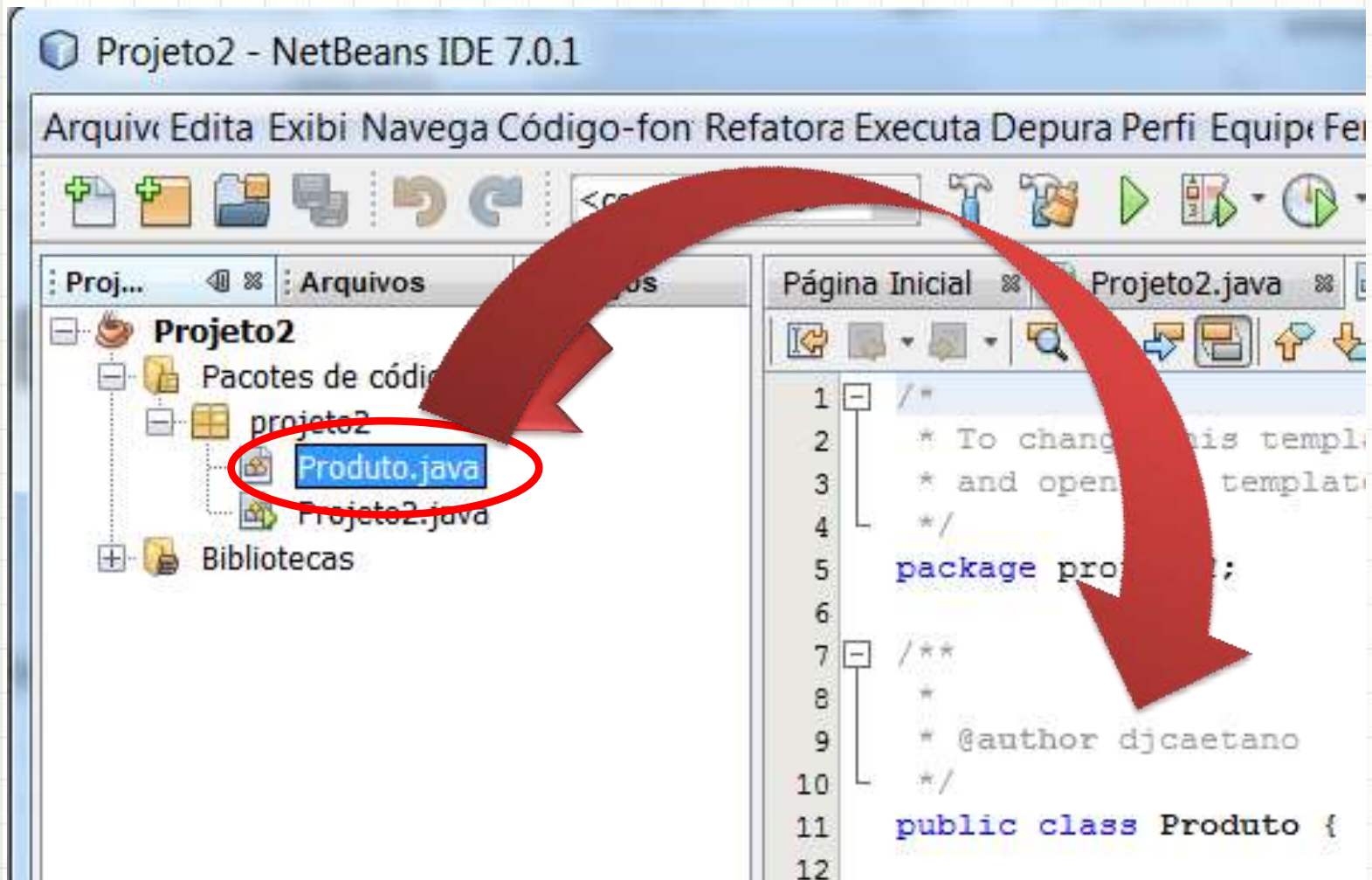
# Definindo uma Classe

- Observe a classe na área de projeto



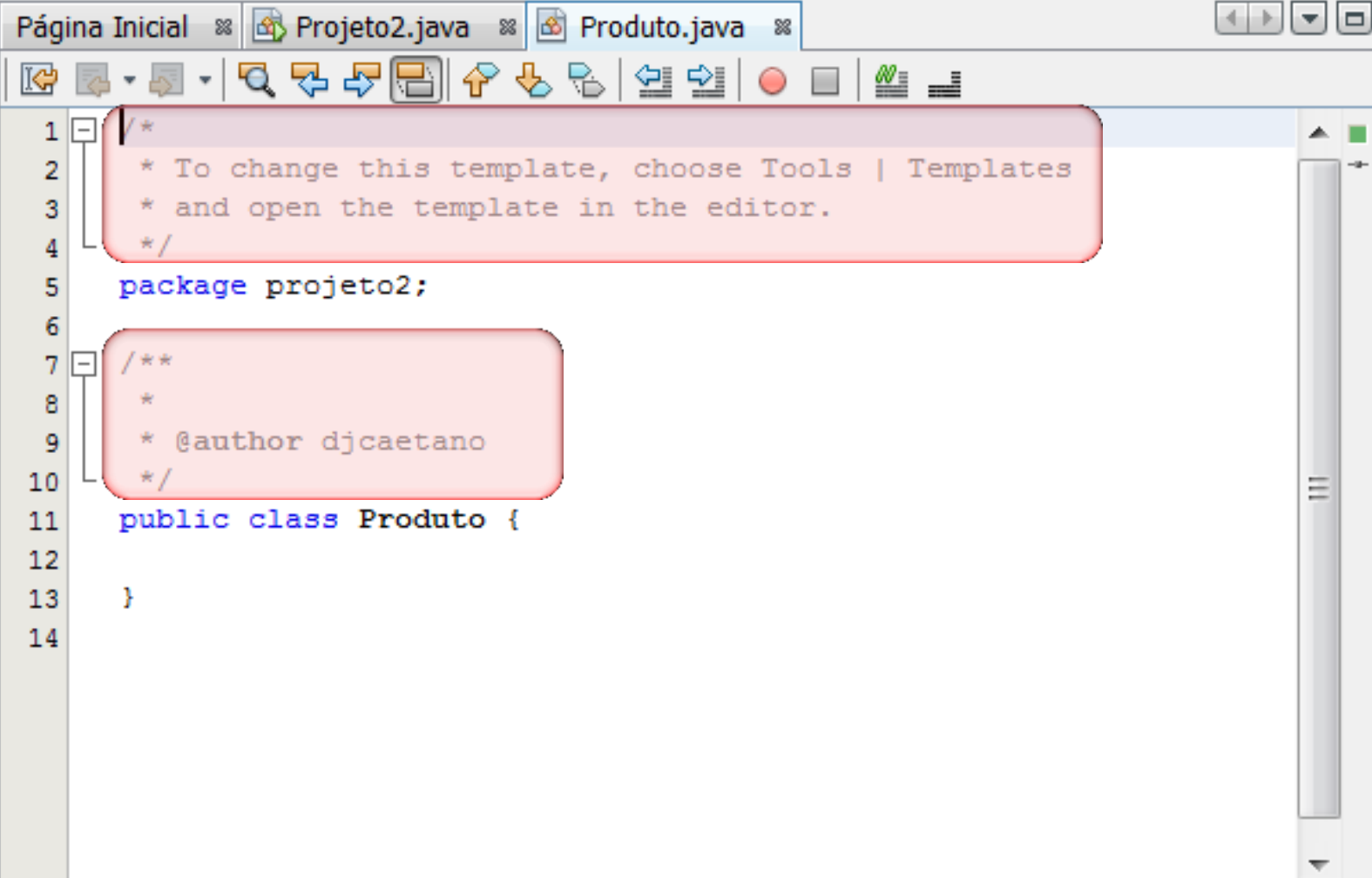
# Definindo uma Classe

- Clique 2 vezes nesse ícone para ver o código



# Limpando a área

- Vamos apagar os comentários do NetBeans

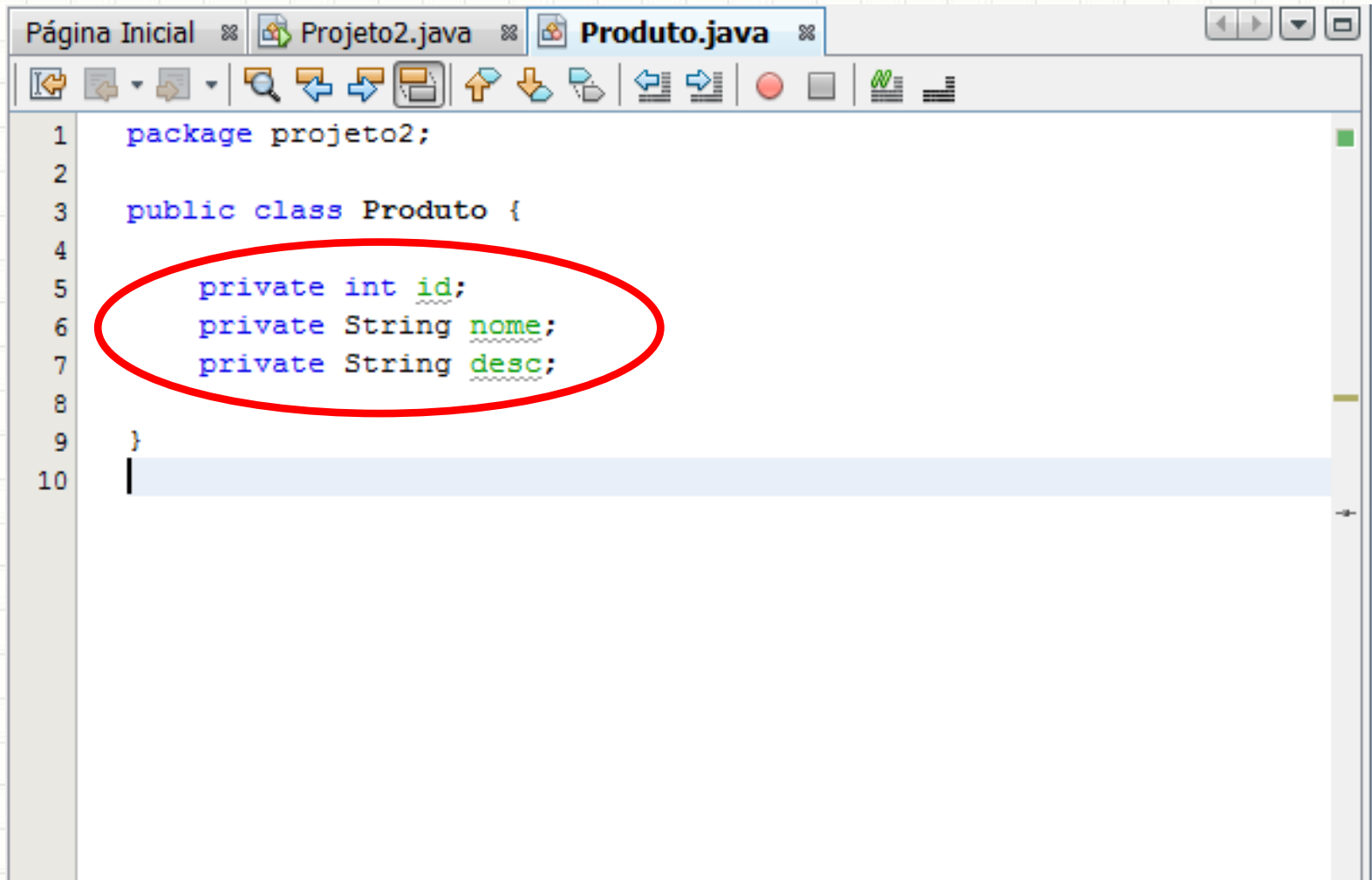


```
Página Inicial ✖ Projeto2.java ✖ Produto.java ✖
1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5  package projeto2;
6
7  /**
8   *
9   * @author djcaetano
10  */
11  public class Produto {
12
13  }
14
```



# Adicionando Atributos

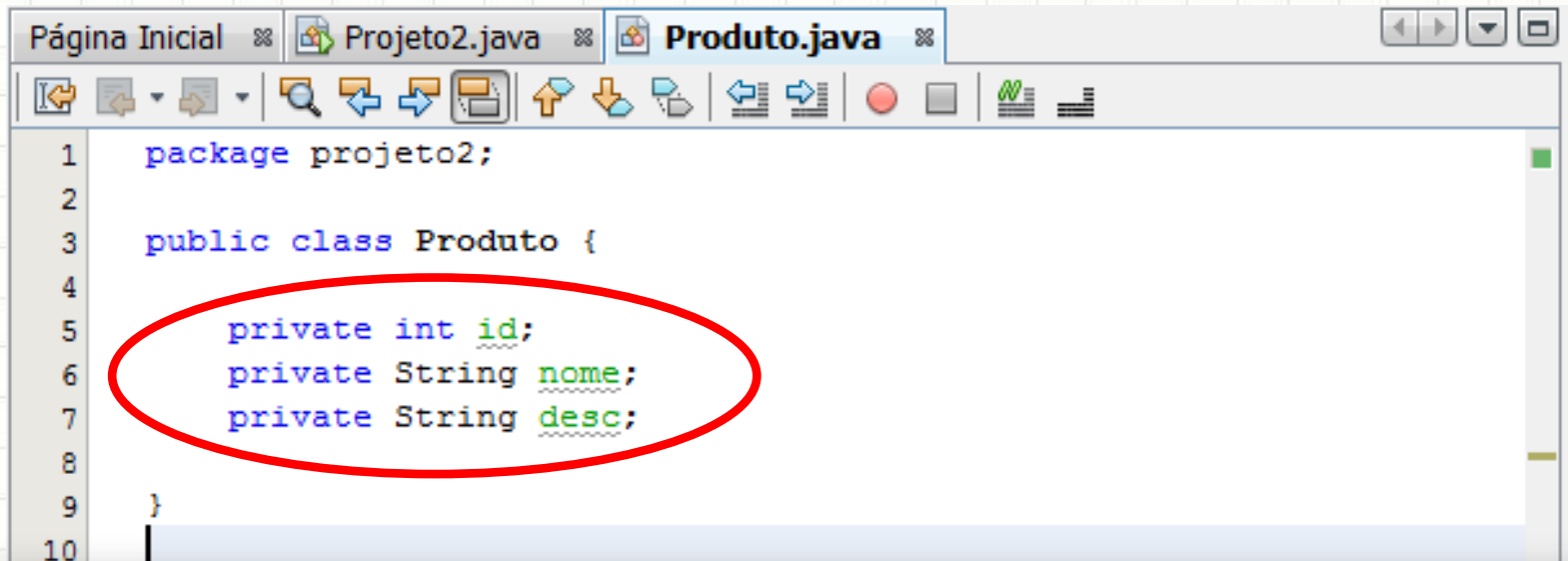
- Vamos adicionar **atributos** no produto



```
1 package projeto2;
2
3 public class Produto {
4
5     private int id;
6     private String nome;
7     private String desc;
8
9 }
10
```

# Adicionando Atributos

- Vamos adicionar **atributos** no produto

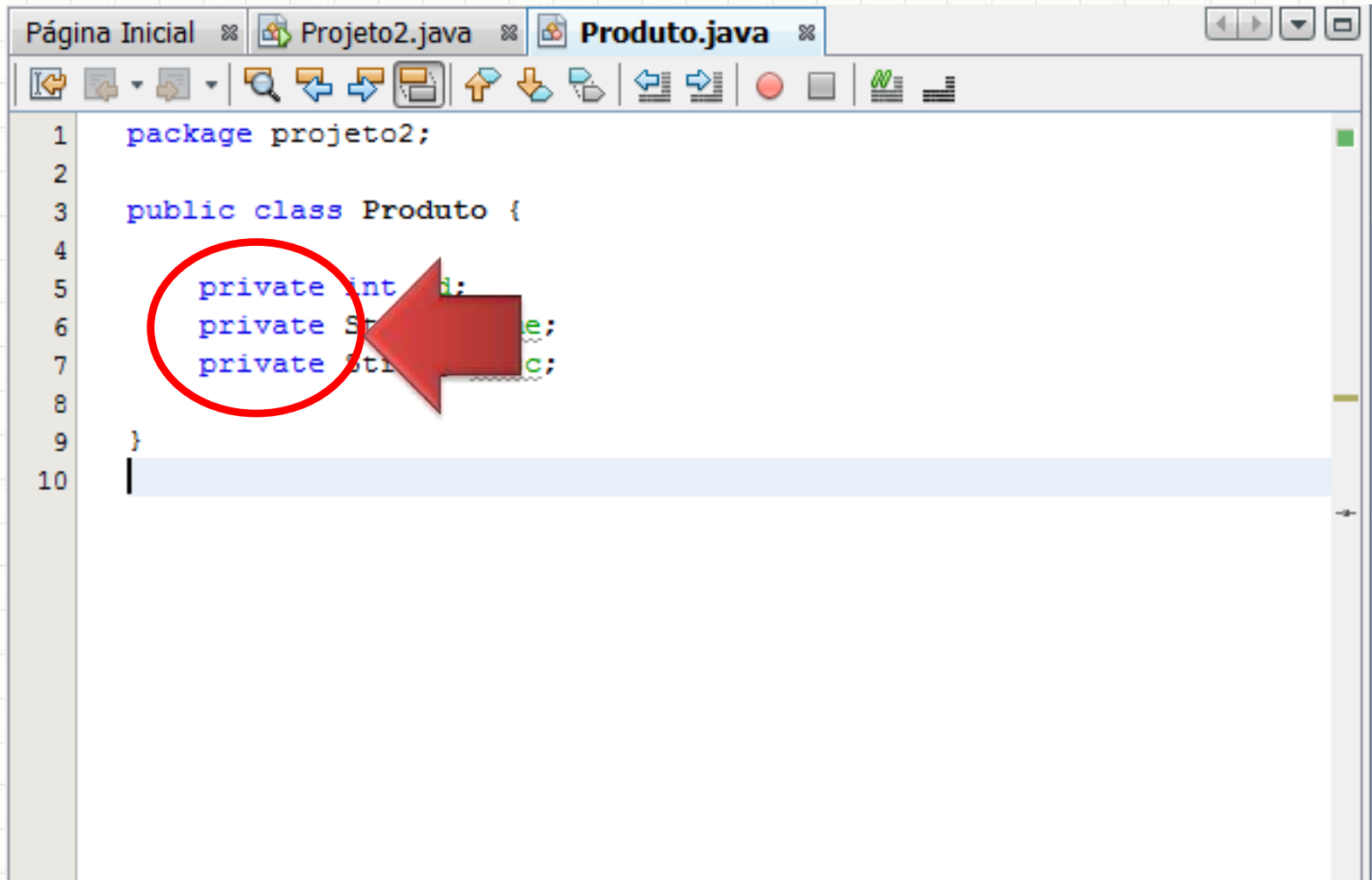


```
1 package projeto2;
2
3 public class Produto {
4
5     private int id;
6     private String nome;
7     private String desc;
8
9 }
10
```

```
private int id;
private String nome;
private String desc;
```

# Adicionando Atributos

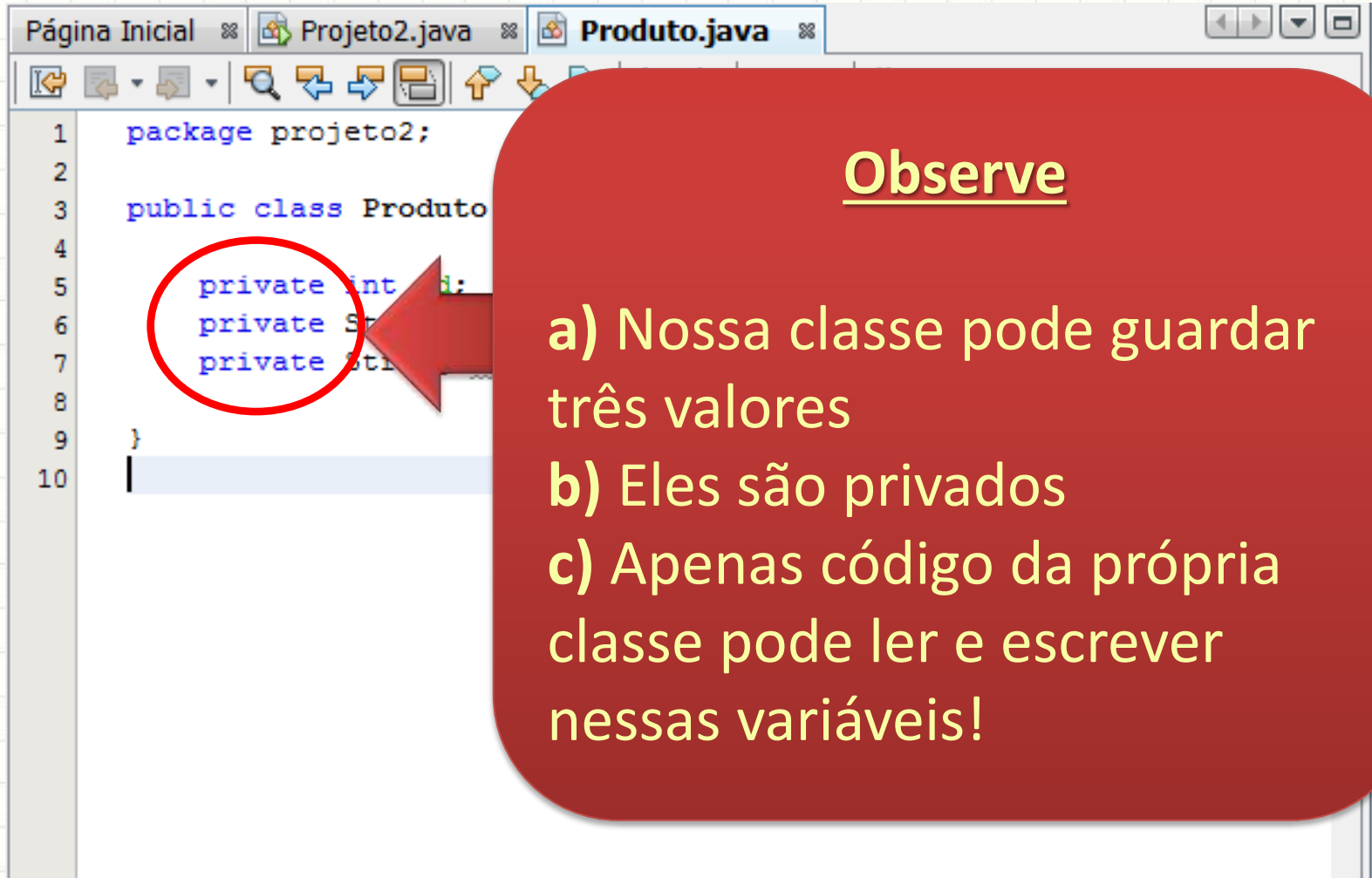
- Vamos adicionar **atributos** no produto



```
1 package projeto2;
2
3 public class Produto {
4
5     private int id;
6     private String nome;
7     private String descricao;
8
9 }
10
```

# Adicionando Atributos

- Vamos adicionar **atributos** no produto



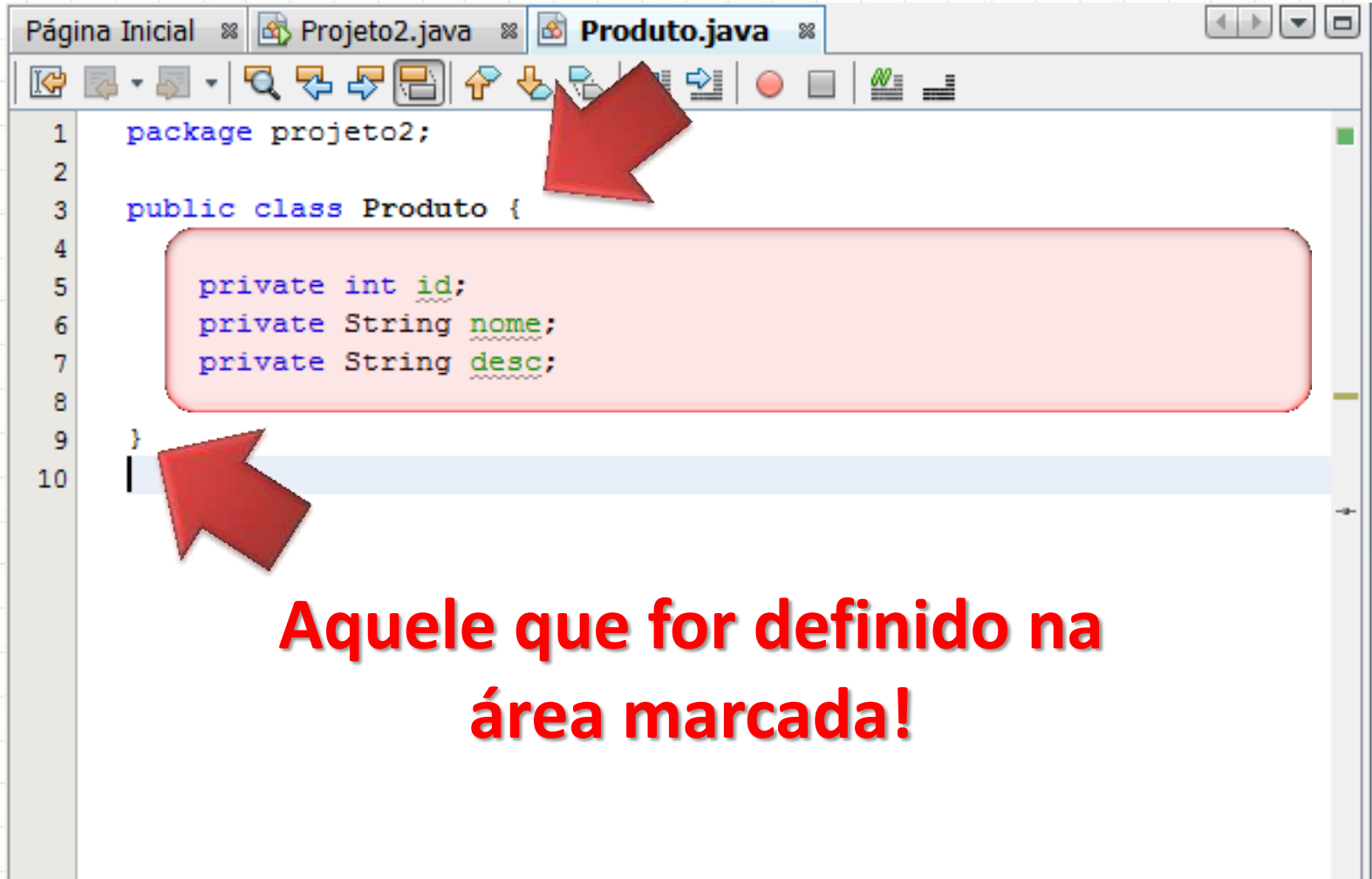
```
1 package projeto2;  
2  
3 public class Produto  
4  
5     private int id;  
6     private String nome;  
7     private String descricao;  
8  
9 }  
10
```

Observe

- a) Nossa classe pode guardar três valores
- b) Eles são privados
- c) Apenas código da própria classe pode ler e escrever nessas variáveis!

# Escopo e Visibilidade

- O que é “código da própria classe”?



The screenshot shows an IDE window with the file 'Produto.java' open. The code is as follows:

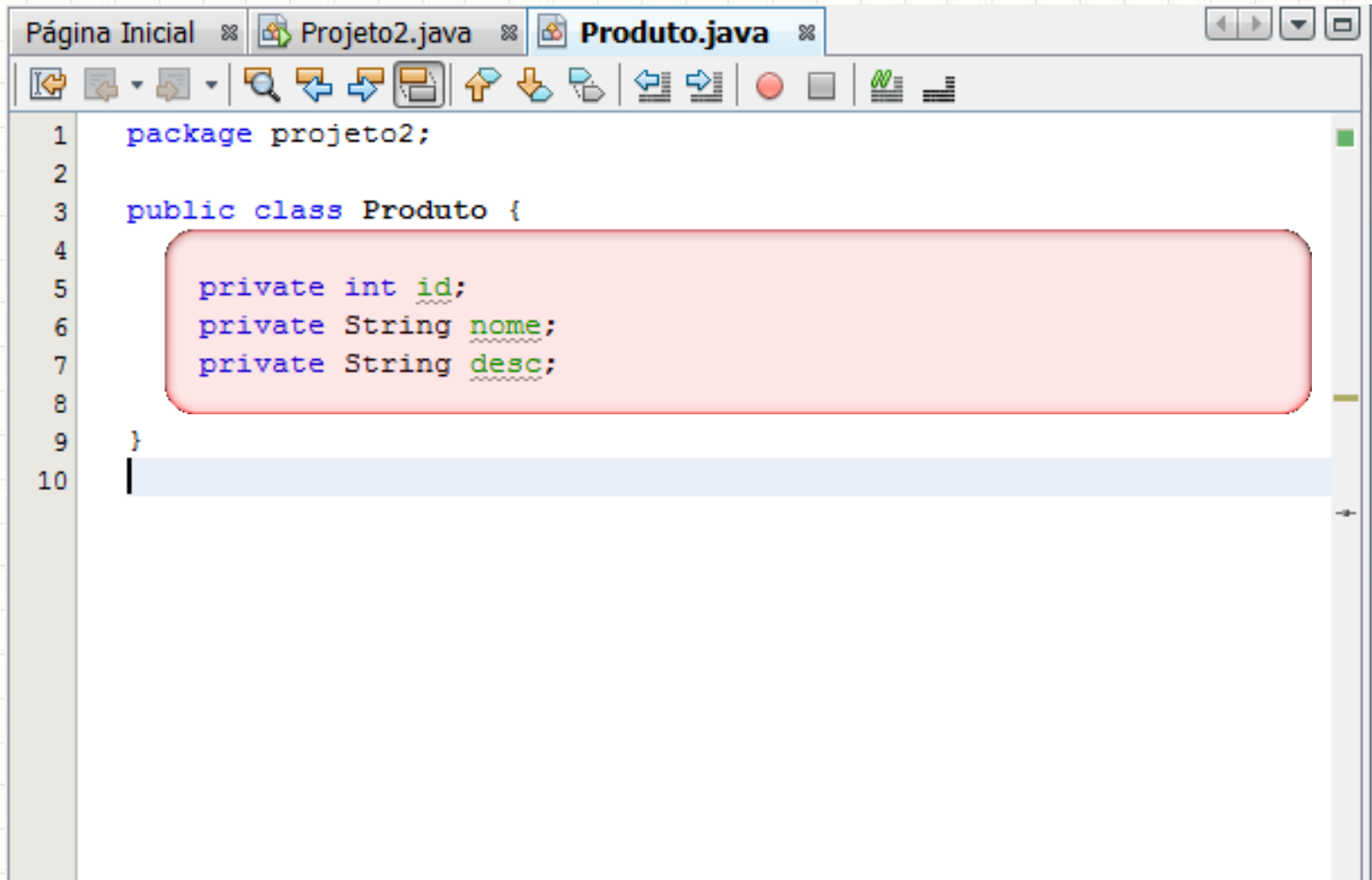
```
1 package projeto2;  
2  
3 public class Produto {  
4     private int id;  
5     private String nome;  
6     private String desc;  
7  
8  
9 }  
10
```

Red arrows point to the package declaration, the class declaration, and the closing brace. A red rounded rectangle highlights the private field declarations from line 5 to line 7.

**Aquele que for definido na área marcada!**

# Setters e Getters

- Como modificar estes valores?



The image shows a screenshot of an IDE window with the following content:

```
Página Inicial ✖ Projeto2.java ✖ Produto.java ✖  
1 package projeto2;  
2  
3 public class Produto {  
4     private int id;  
5     private String nome;  
6     private String desc;  
7  
8  
9 }  
10
```

The code defines a class named `Produto` within the package `projeto2`. It contains three private fields: `id` (an integer), `nome` (a String), and `desc` (a String). The fields are highlighted with a red rounded rectangle.

# Setters e Getters

- Vamos acrescentar métodos públicos?
- Os métodos que servem para modificar atributos são chamados de **setters**.
- Se o nome do atributo é **idade**, o nome do *setter* será **setIdade()**.
- Se o nome do atributo é **sexo**, o nome do *setter* será **setSexo()**.

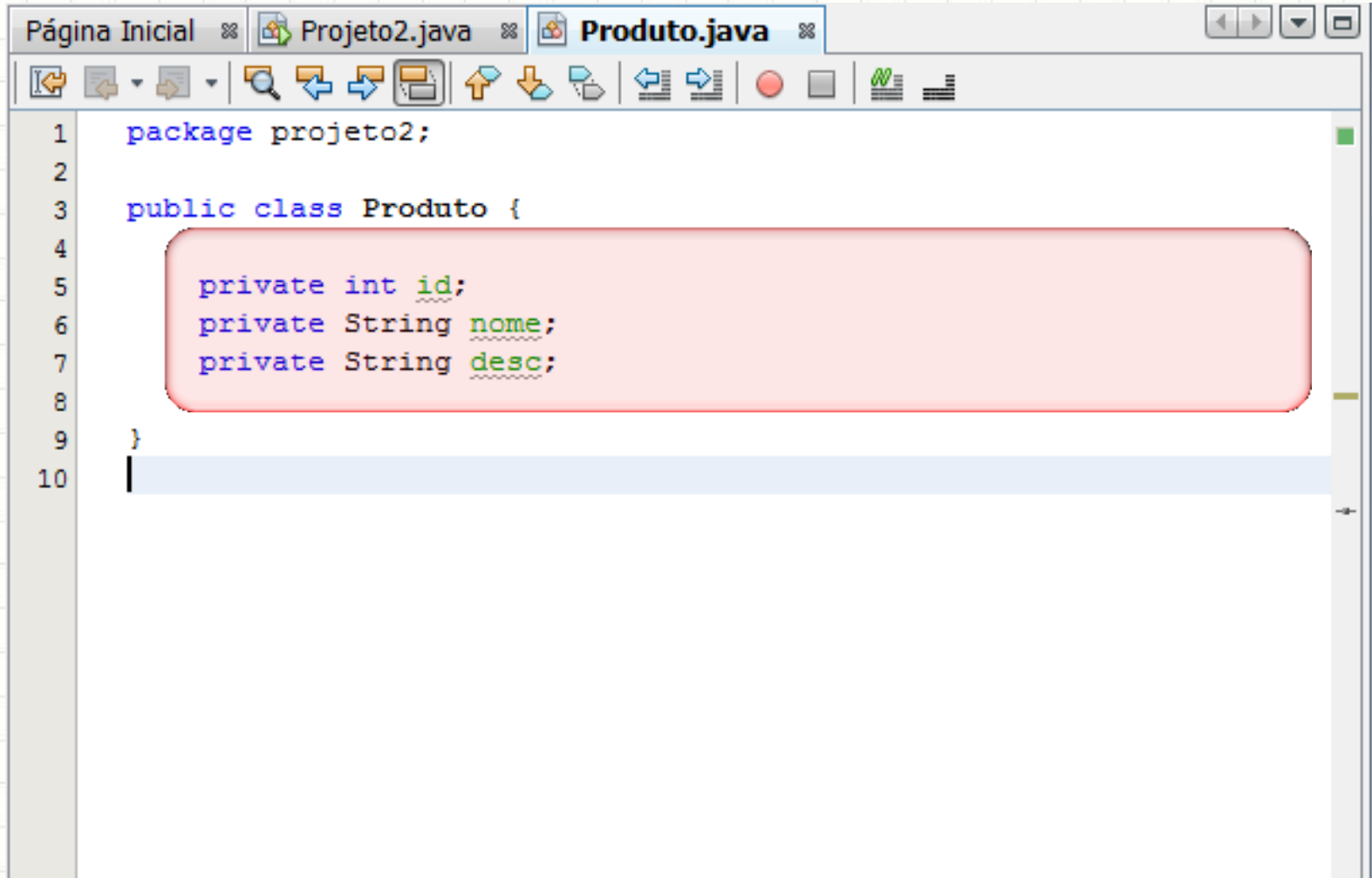
# *Setters e Getters*

- Entre parênteses devemos indicar o novo valor desejado para o atributo
- Ex.: **setIdade(23)**
- Como a criação desses métodos é totalmente mecânica, o NetBeans a realiza por nós



# Setters e Getters – Criando Setters

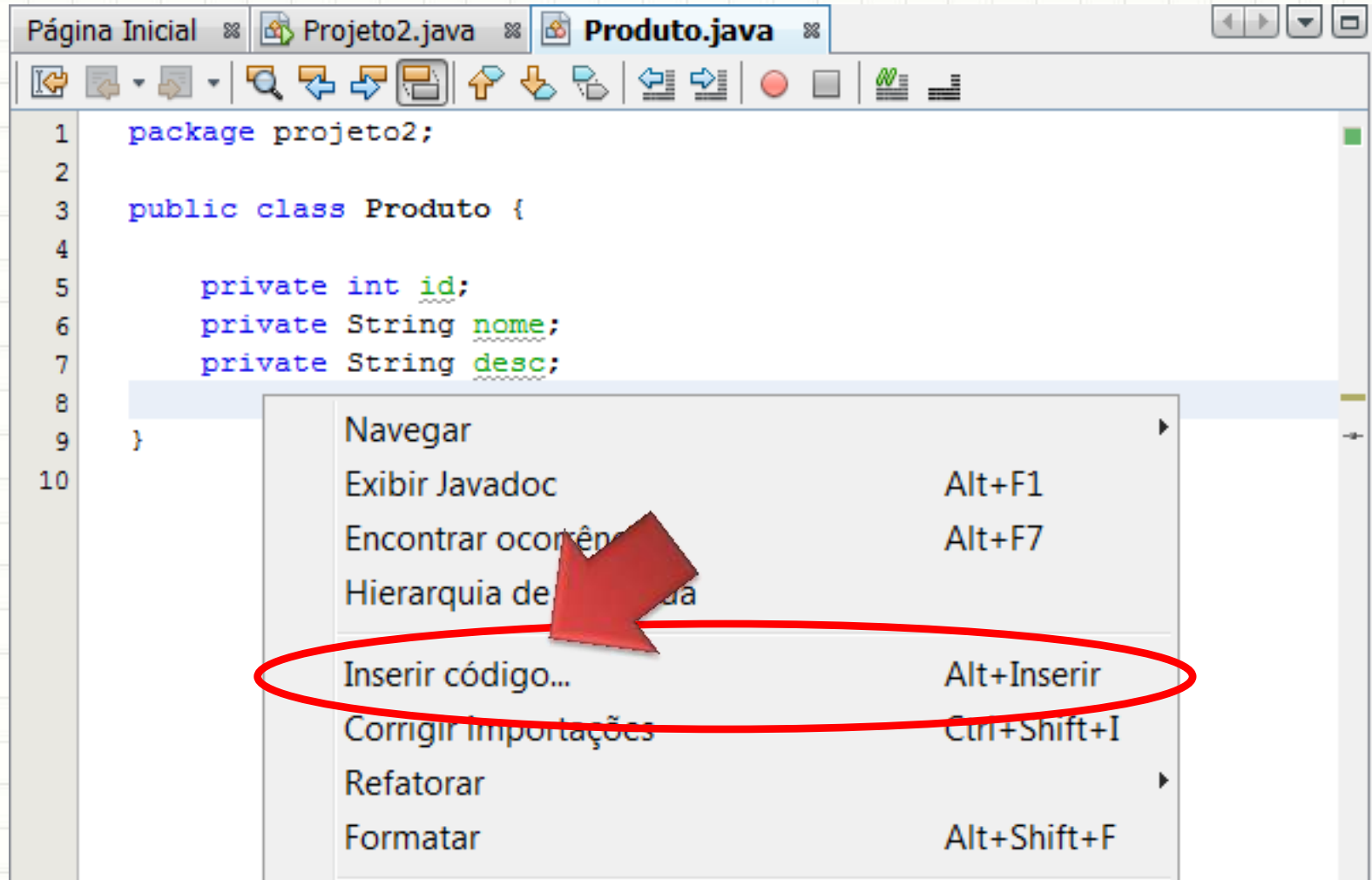
- Clique com o botão direito na área da classe



```
1 package projeto2;
2
3 public class Produto {
4     private int id;
5     private String nome;
6     private String desc;
7
8
9 }
10
```

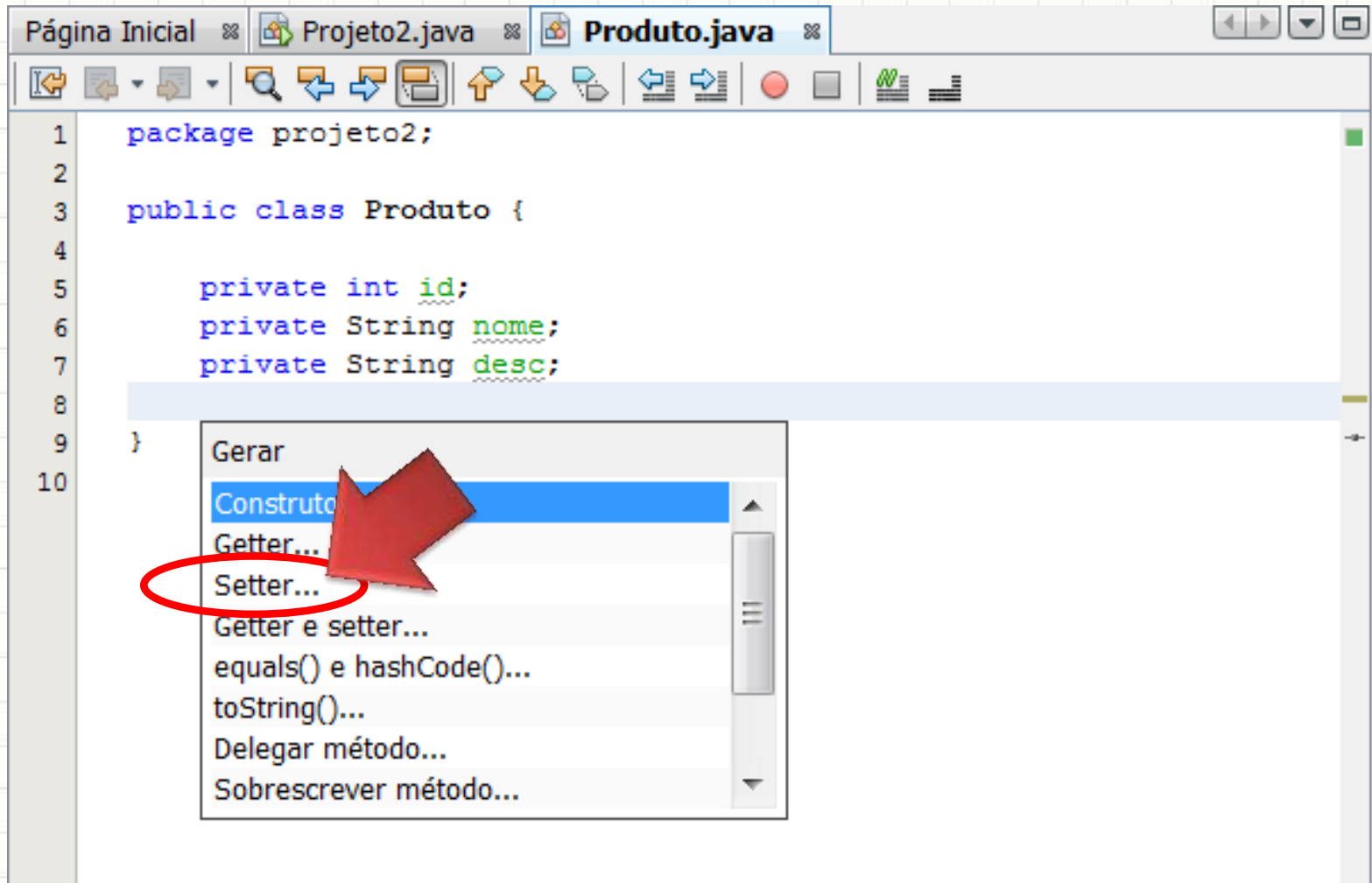
# Setters e Getters – Criando Setters

- Selecione a opção **inserir código**



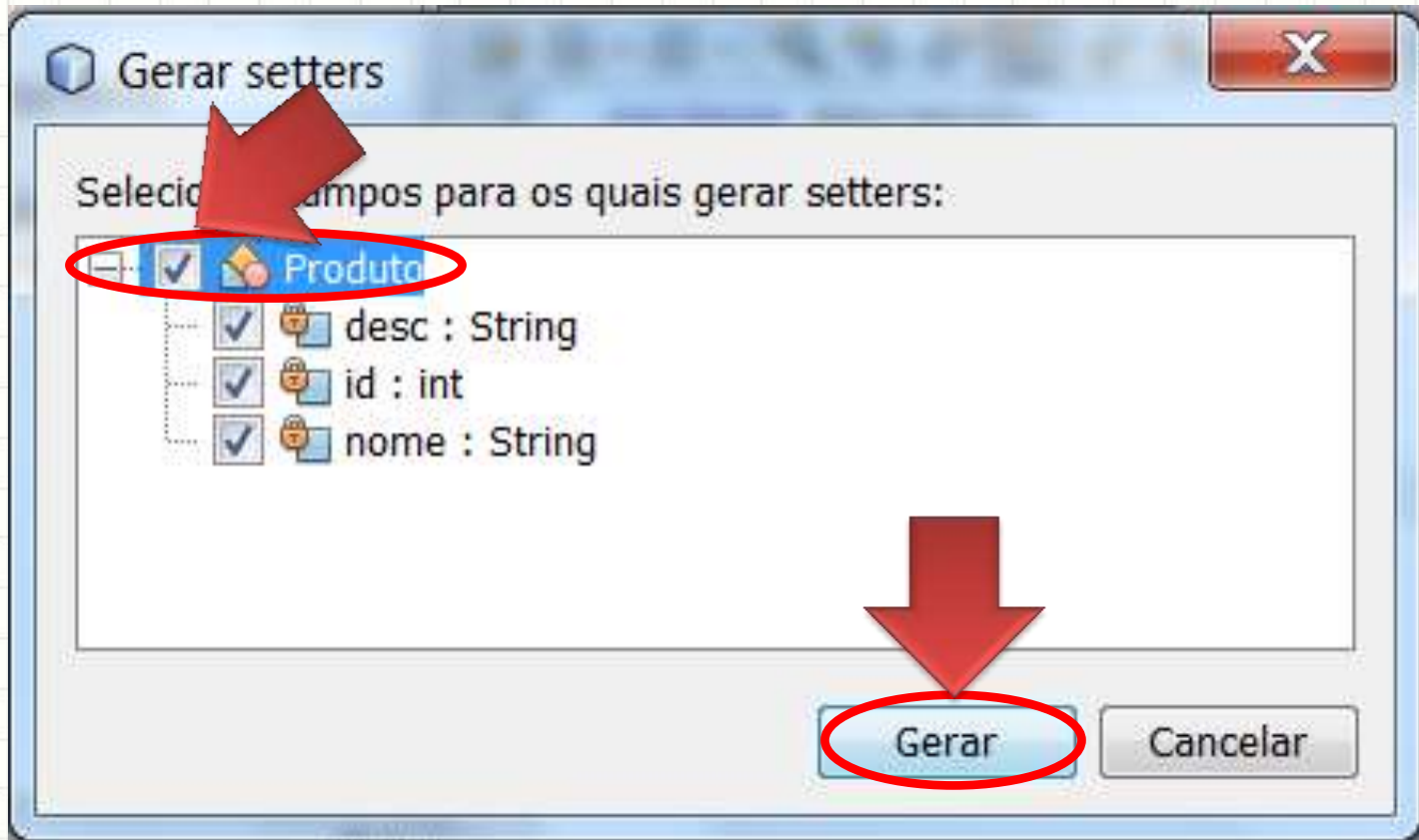
# Setters e Getters – Criando Setters

- No menu, selecione **Setter...**



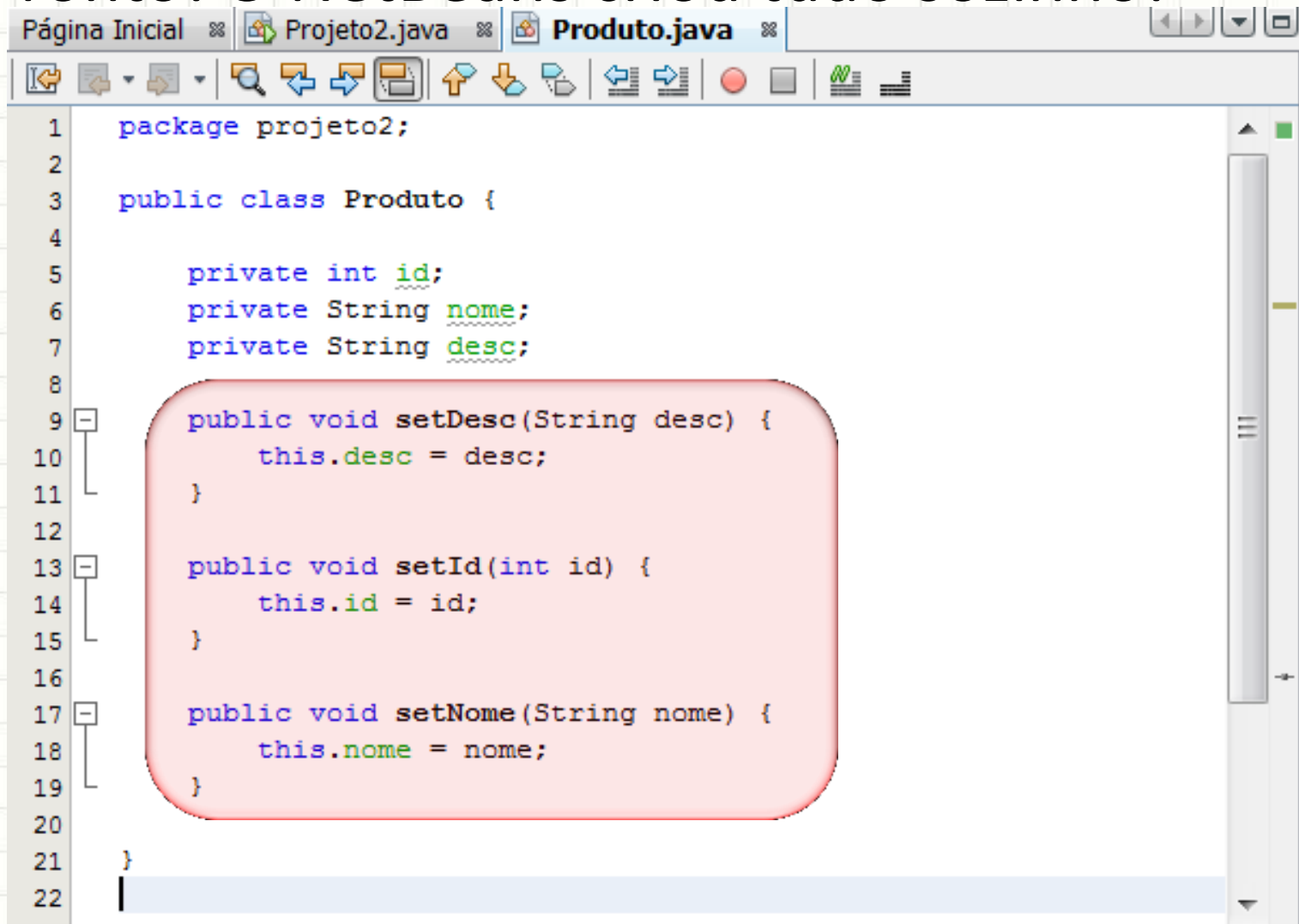
# Setters e Getters – Criando Setters

- Selecione a caixinha de **Produto** para marcar todos os atributos ao mesmo tempo



# Setters e Getters – Criando Setters

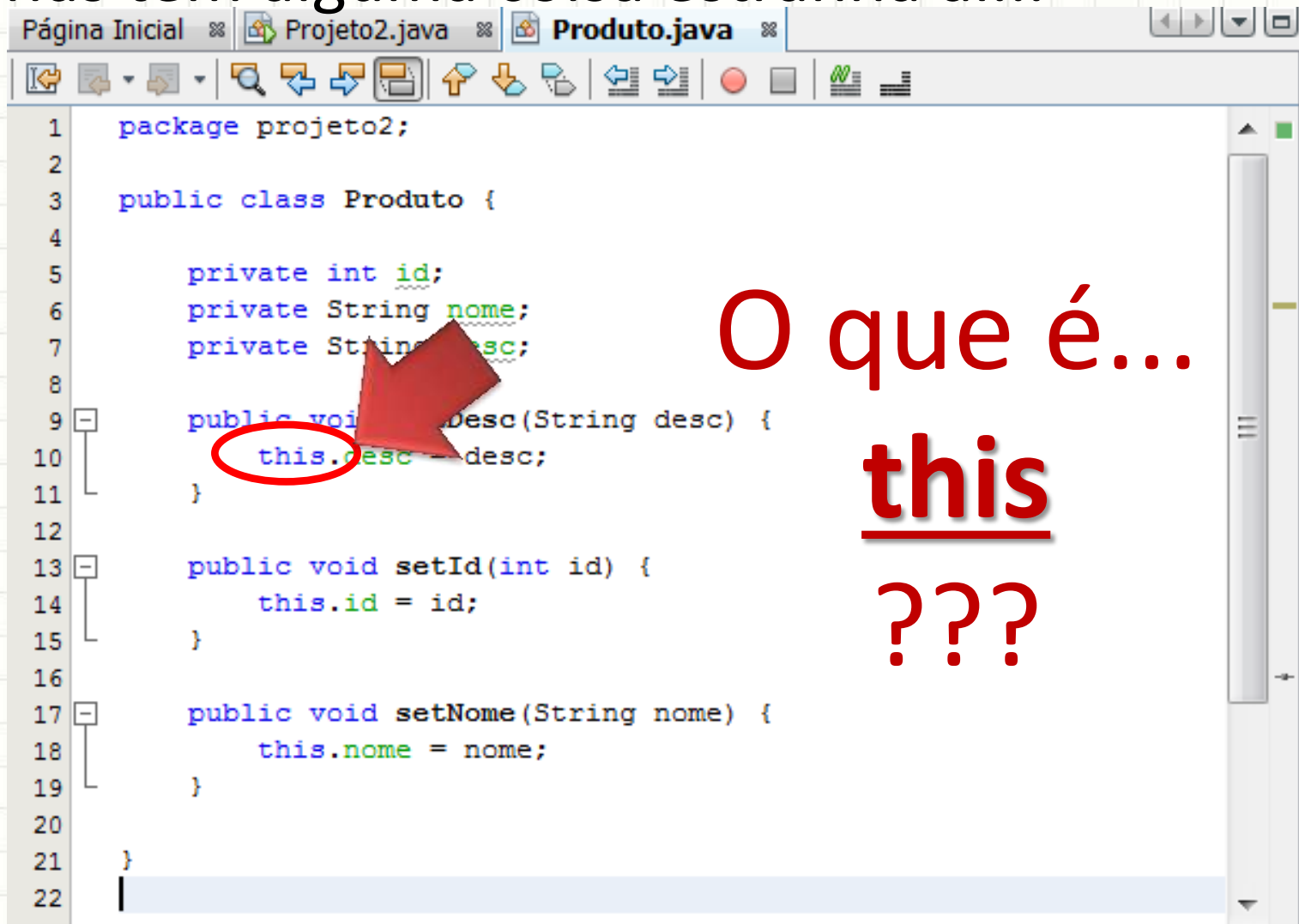
- Pronto! O NetBeans criou tudo sozinho!



```
1 package projeto2;
2
3 public class Produto {
4
5     private int id;
6     private String nome;
7     private String desc;
8
9     public void setDesc(String desc) {
10         this.desc = desc;
11     }
12
13     public void setId(int id) {
14         this.id = id;
15     }
16
17     public void setNome(String nome) {
18         this.nome = nome;
19     }
20
21 }
22
```

# Setters e Getters – Criando Setters

- Mas tem alguma coisa estranha aí...

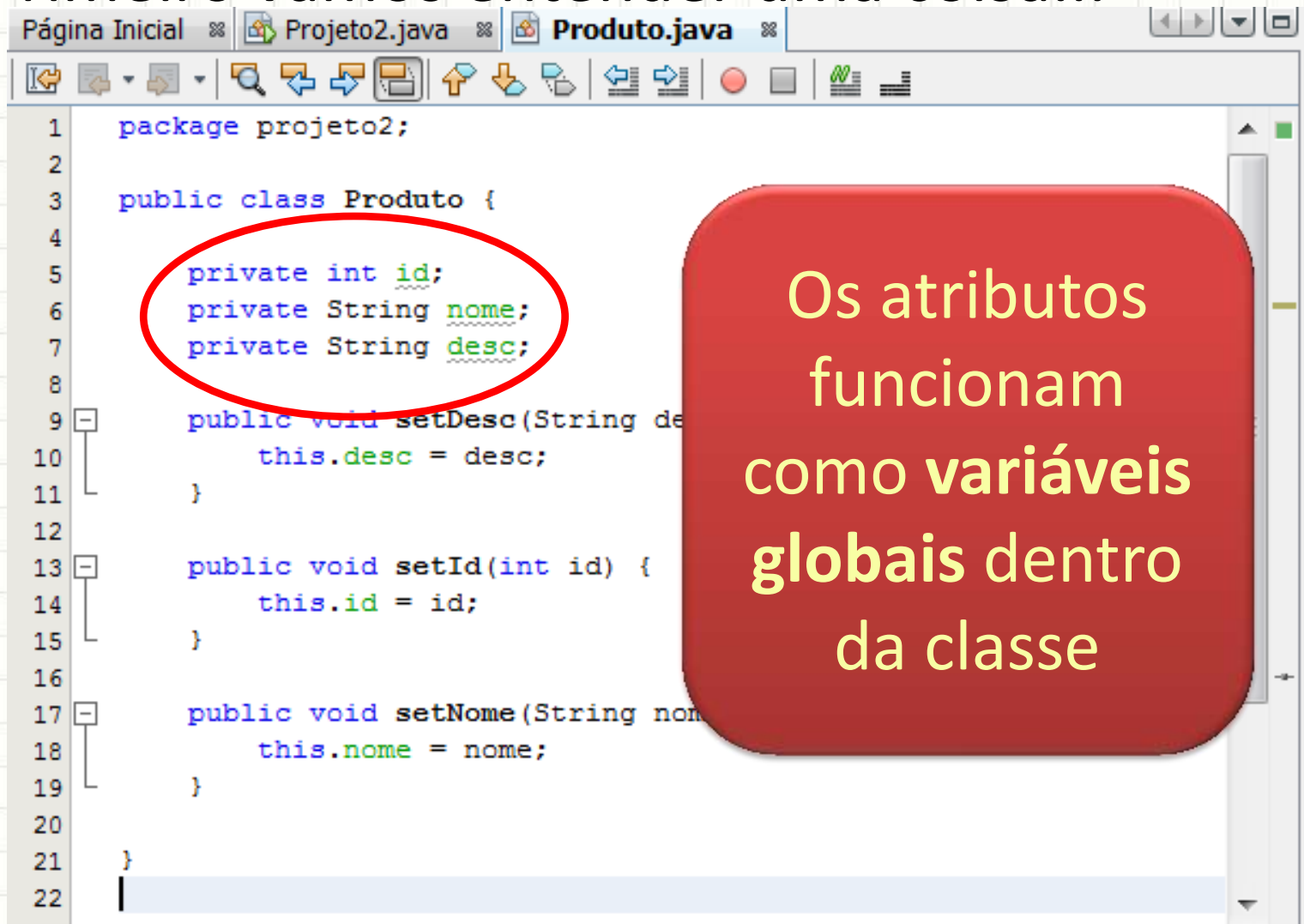


```
1 package projeto2;
2
3 public class Produto {
4
5     private int id;
6     private String nome;
7     private String desc;
8
9     public void setDesc(String desc) {
10         this.desc = desc;
11     }
12
13     public void setId(int id) {
14         this.id = id;
15     }
16
17     public void setNome(String nome) {
18         this.nome = nome;
19     }
20
21 }
22
```

O que é...  
this  
???

# Setters e Getters – Criando Setters

- Primeiro vamos entender uma coisa...

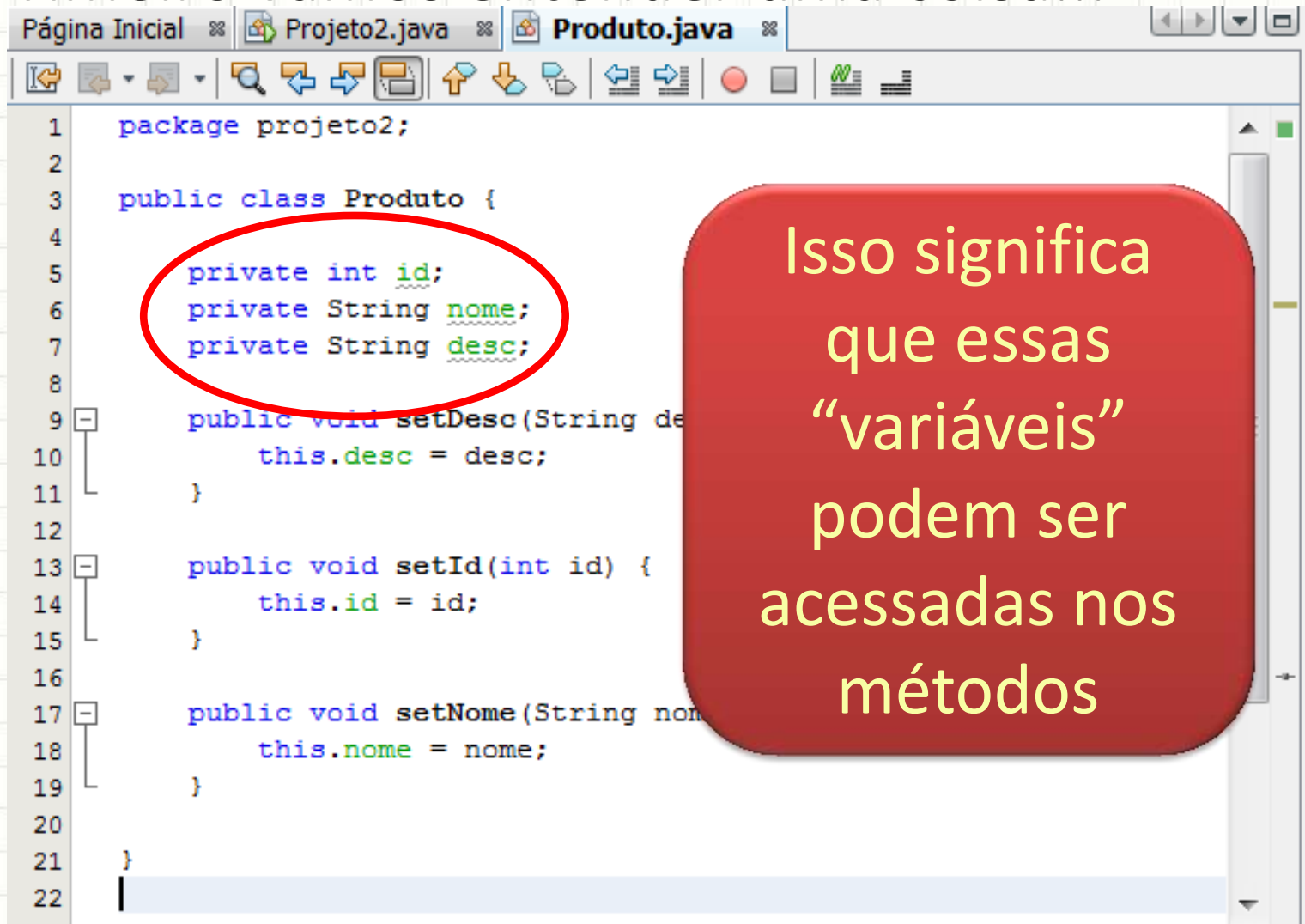


```
1 package projeto2;
2
3 public class Produto {
4     private int id;
5     private String nome;
6     private String desc;
7
8
9     public void setDesc(String desc) {
10         this.desc = desc;
11     }
12
13     public void setId(int id) {
14         this.id = id;
15     }
16
17     public void setNome(String nome) {
18         this.nome = nome;
19     }
20
21 }
22
```

Os atributos funcionam como variáveis globais dentro da classe

# Setters e Getters – Criando Setters

- Primeiro vamos entender uma coisa...



```
1 package projeto2;
2
3 public class Produto {
4     private int id;
5     private String nome;
6     private String desc;
7
8
9     public void setDesc(String desc) {
10         this.desc = desc;
11     }
12
13     public void setId(int id) {
14         this.id = id;
15     }
16
17     public void setNome(String nome) {
18         this.nome = nome;
19     }
20
21 }
22
```

Isso significa que essas “variáveis” podem ser acessadas nos métodos



# Setters e Getters

- Primeiro vamos em

```
Página Inicial  Projeto2.java
1 package projeto2;
2
3 public class Produto {
4
5     private int id;
6     private String nome;
7     private String desc;
8
9     public void setDesc(String desc) {
10         this.desc = desc;
11     }
12
13     public void setId(int id) {
14         this.id = id;
15     }
16
17     public void setName(String nome) {
18         this.nome = nome;
19     }
20
21 }
22
```

Ocorre que, neste caso, foram definidos parâmetros com os **mesmos** nomes dos atributos... Observe!

# Setters e Getters

- Primeiro vamos criar

```
Página Inicial ✖ Projeto2.java ✖  
1 package projeto2;  
2  
3 public class Produto {  
4  
5     private int id;  
6     private String nome;  
7     private String desc;  
8  
9     public void setDesc(String desc) {  
10         this.desc = desc;  
11     }  
12  
13     public void setId(int id) {  
14         this.id = id;  
15     }  
16  
17     public void setNome(String nome) {  
18         this.nome = nome;  
19     }  
20  
21 }  
22
```

Isso cria confusão: se, no método `setDesc`, escrevermos, `desc = "Olá"` Estamos mudando o valor do atributo ou do parâmetro?

O Java soluciona isso da seguinte forma: sempre que houver confusão entre nomes de variáveis de um método com os nomes de atributos de uma classe, devemos usar a palavra this para nos referir ao atributo.

```
4
5     private int id;
6     private String nome;
7     private String desc;
8
9     public void setDesc(String desc) {
10         this.desc = desc;
11     }
12
13     public void setId(int id) {
14         this.id = id;
15     }
16
17     public void setNome(String nome) {
18         this.nome = nome;
19     }
20
21 }
22
```

O Java soluciona isso da seguinte forma: sempre que houver confusão entre nomes de variáveis de um método com os nomes de atributos de uma classe, devemos usar a palavra this para nos referir ao atributo.

```
4
5     private int id;
6     private String nome;
7     private String desc;
8
9     public void setDesc(String desc) {
10         this.desc = desc;
11     }
12
13     public void setId(int id) {
14         this.id = id;
15     }
16
17     public void setName(String nome) {
18         this.nome = nome;
19     }
20
21 }
22
```

O Java soluciona isso da seguinte forma: sempre que houver confusão entre nomes de variáveis de um método com os nomes de atributos de uma classe, devemos usar a palavra this para nos referir ao atributo.

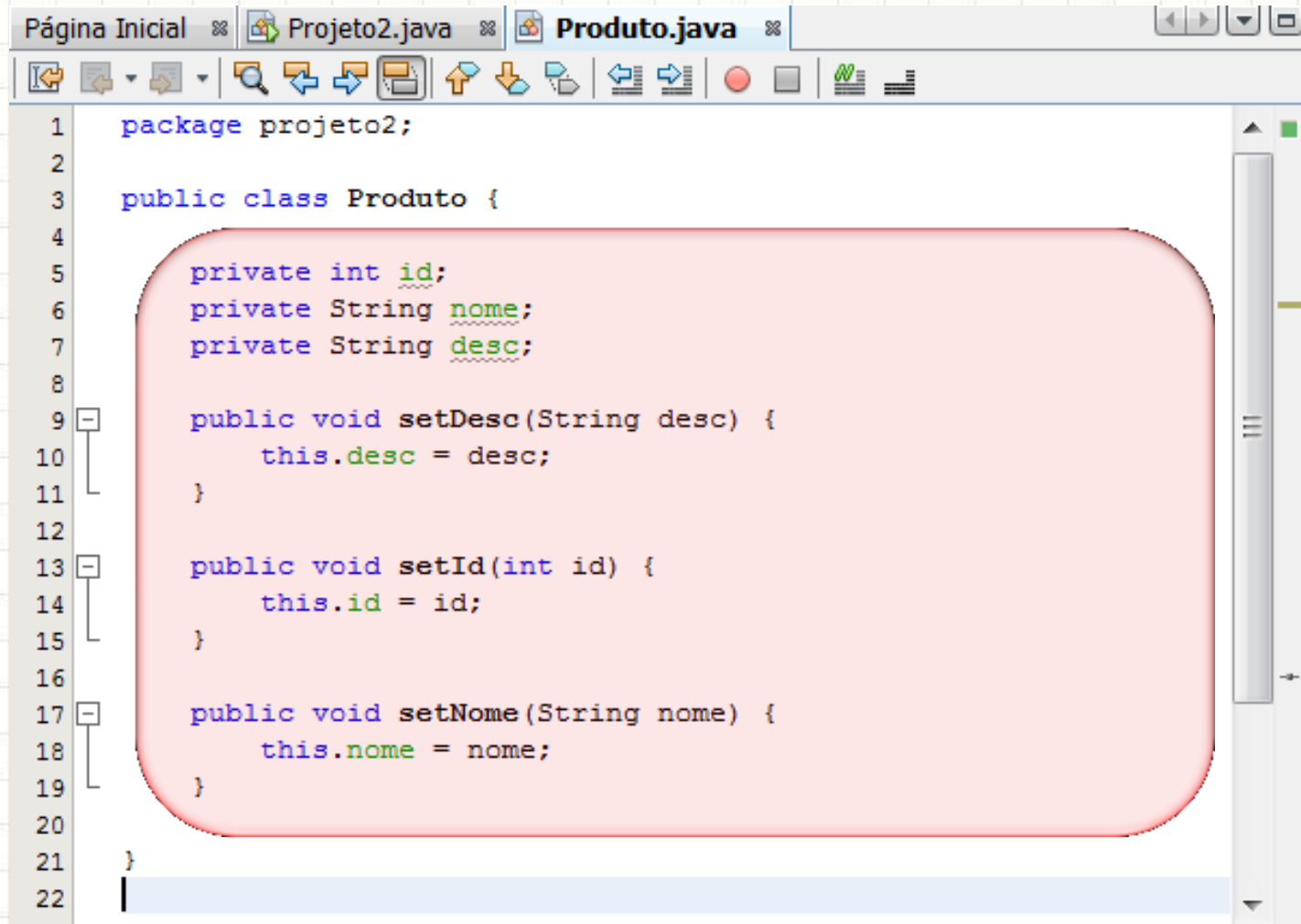
```
4
5     private int id;
6     private String nome;
7     private String desc;
8
9     public void setDesc(String desc) {
10         this.desc = desc;
11     }
12
13     public void setId(int id) {
14         this.id = id;
15     }
16
17     public void setName(String nome) {
18         this.nome = nome;
19     }
20
21 }
22
```

# Setters e Getters

- Agora já sabemos como **mudar os valores** dos atributos. Mas como ler estes valores?
- Os métodos que servem para ler atributos são chamados de **getters**.
- Se o nome do atributo é **idade**, o nome do *getter* será **getIdade**.
- Se o nome do atributo é **sexo**, o nome do *getter* será **getSexo**.
- Deu para pegar a idéia? 😊

# Setters e Getters – Criando Getters

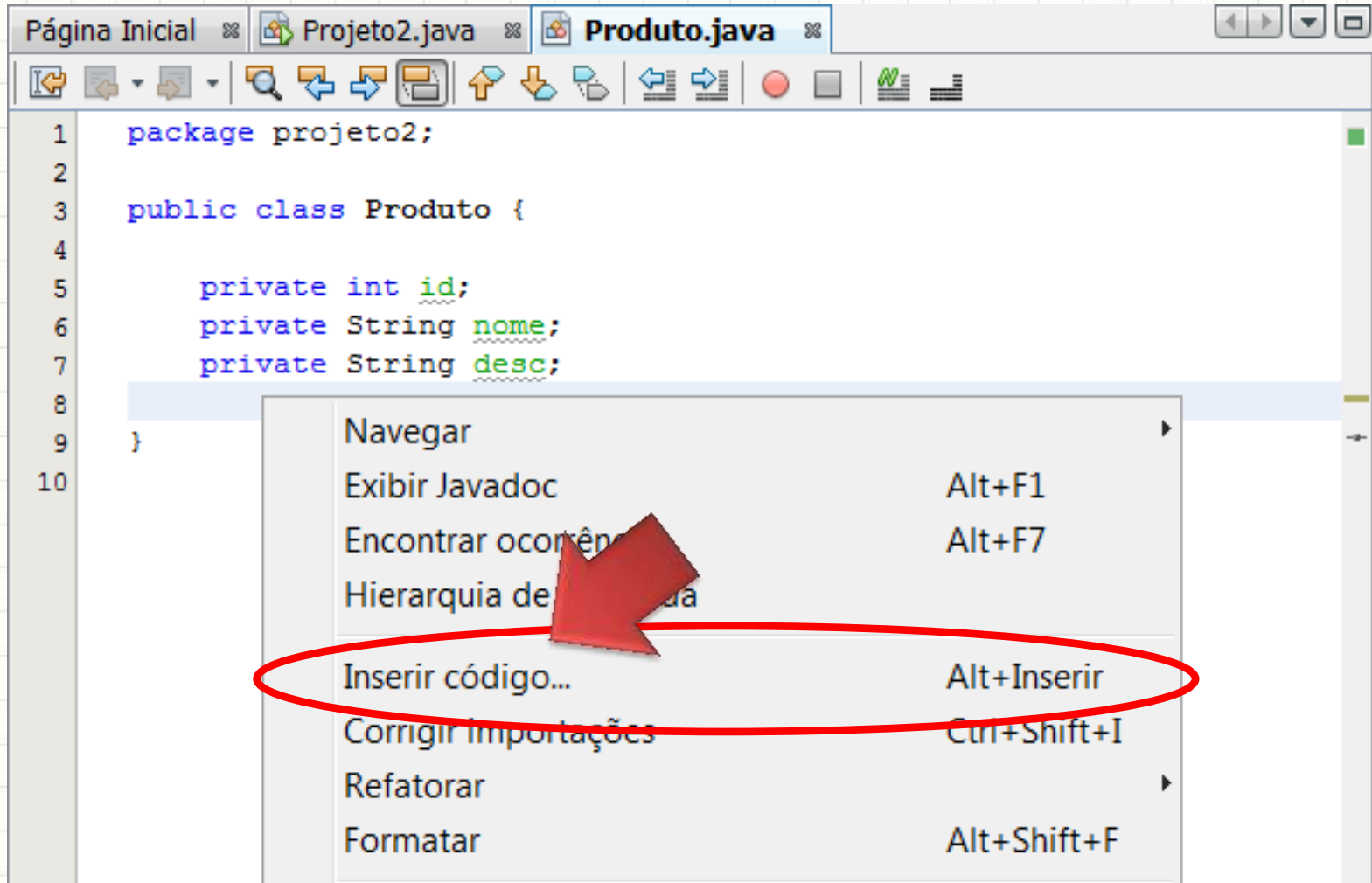
- Clique com o botão direito na área da classe



```
1 package projeto2;
2
3 public class Produto {
4
5     private int id;
6     private String nome;
7     private String desc;
8
9     public void setDesc(String desc) {
10         this.desc = desc;
11     }
12
13     public void setId(int id) {
14         this.id = id;
15     }
16
17     public void setNome(String nome) {
18         this.nome = nome;
19     }
20
21 }
22
```

# Setters e Getters – Criando Getters

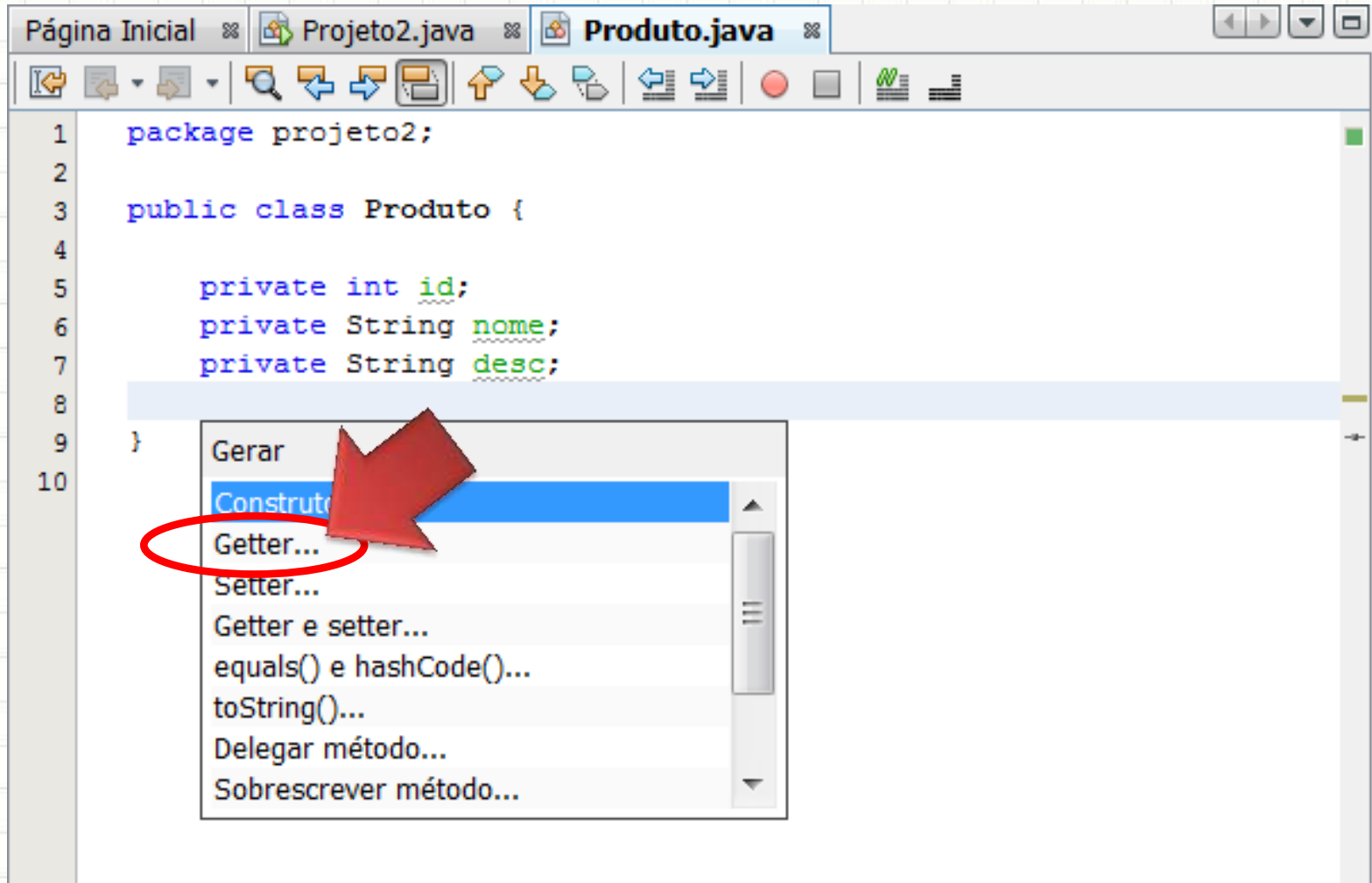
- Selecione a opção **inserir código**





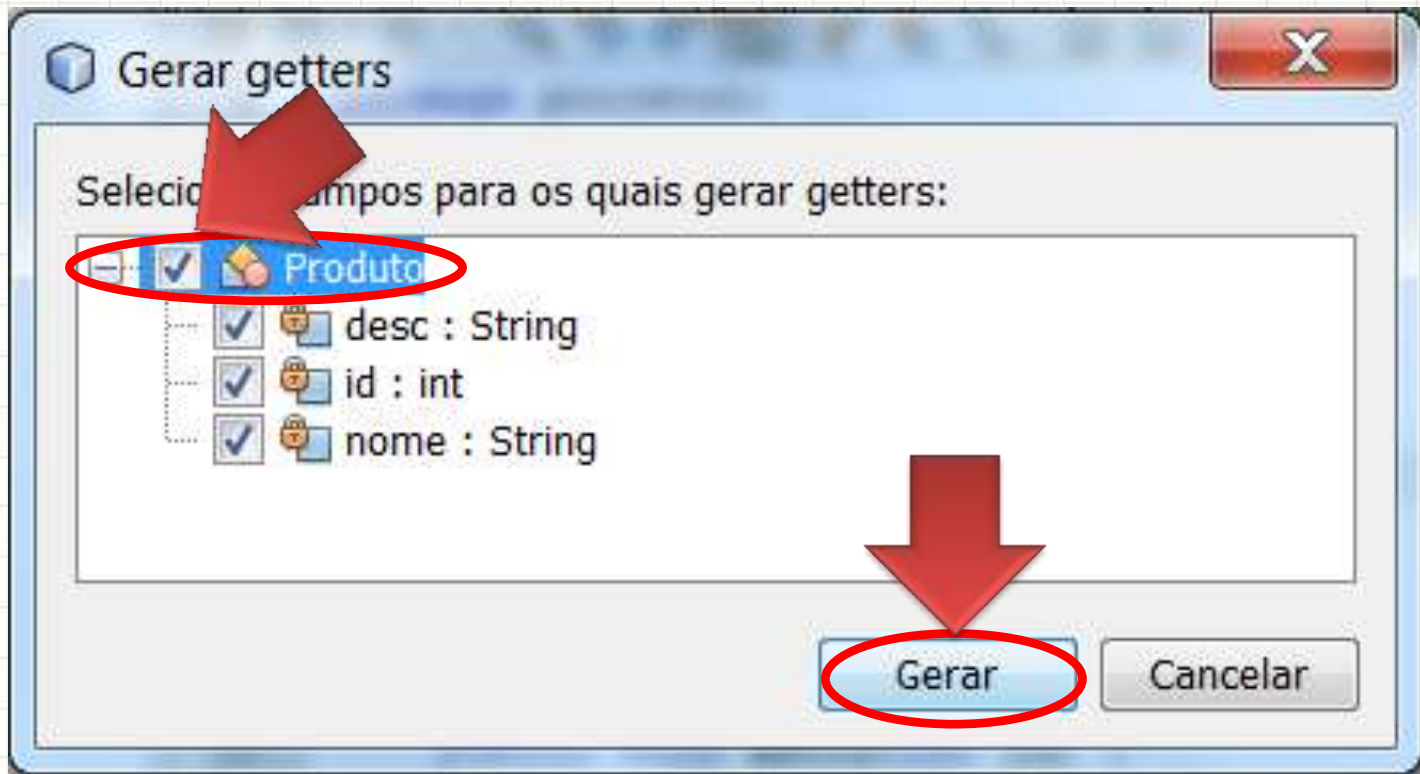
# Setters e Getters – Criando Getters

- No menu, selecione **Getter...**



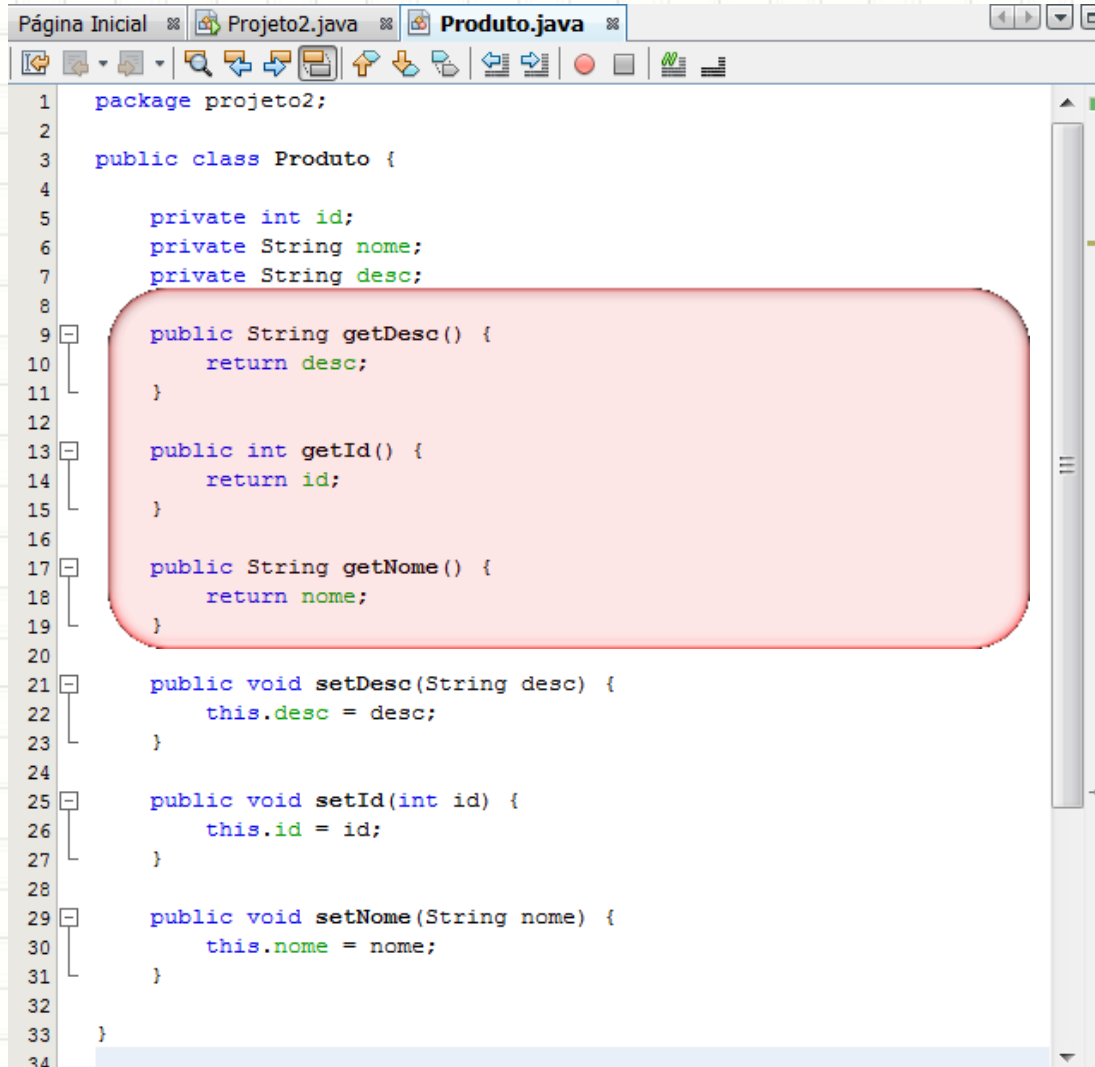
# Setters e Getters – Criando Getters

- Selecione a caixinha de **Produto** para marcar todos os atributos ao mesmo tempo



# Setters e Getters – Criando Getters

- Pronto! O NetBeans criou tudo sozinho!



```
Página Inicial  Projeto2.java  Produto.java
1  package projeto2;
2
3  public class Produto {
4
5      private int id;
6      private String nome;
7      private String desc;
8
9      public String getDesc() {
10         return desc;
11     }
12
13     public int getId() {
14         return id;
15     }
16
17     public String getNome() {
18         return nome;
19     }
20
21     public void setDesc(String desc) {
22         this.desc = desc;
23     }
24
25     public void setId(int id) {
26         this.id = id;
27     }
28
29     public void setNome(String nome) {
30         this.nome = nome;
31     }
32
33 }
34
```

# Setters e Getters – Criando Getters

- Observe os *getters* de perto.
- Eles simplesmente retornam o valor do atributo!
- Observe que, neste caso, como não há confusão, não é necessário usar a palavra **this**

```
package projeto2;

public class Produto {

    private int id;
    private String nome;
    private String desc;

    public String getDesc() {
        return desc;
    }


    public int getId() {
        return id;
    }

    public String getNome() {
        return nome;
    }
}
```

# Setters e Getters

- Qual a vantagem de usar *getters* e *setters*?
  1. Se você não quiser que um atributo seja modificado por outras classes, remova o *setter* daquela variável
  2. Se você não quiser que um atributo seja lido por outras classes, remova o *getter* daquela variável
  3. O *setter* permite **validar** os dados antes de armazená-los, evitando que dados incorretos sejam colocados nos atributos
  4. O *getter* permite esconder o formato (tipo de dado) com que um atributo está armazenado

# Setters e Getters

- Qual a vantagem de usar *getters* e *setters*?
  1. Se você não quiser que um atributo seja modificado por outras classes, remova o *setter* daquela variável
  2. ido por variável
  3. e armazená-los, evitando que dados incorretos sejam colocados nos atributos
  4. O *getter* permite esconder o formato (tipo de dado) com que um atributo está armazenado



# **ESPECIFICADORES DE ACESSO**

# Especificadores de Acesso

- Você deve ter reparado as palavras:
  - **public** e **private**
  - Já falamos delas!
  - Modificadores para **classes**, **atributos** e **métodos**
- **Public**: elemento acessível em todo o projeto
- **Private**: elemento só acessível localmente
  - Classe: no arquivo
  - Atributo e método: na classe



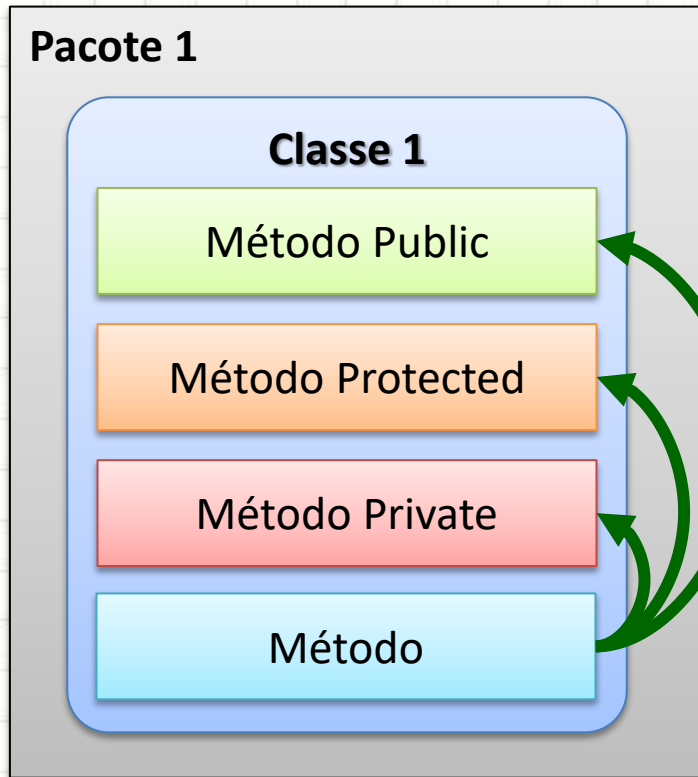
# Especificadores de Acesso

- Existe um terceiro especificador: **protected**
  - Um misto de **public** e **private**, ao mesmo tempo!
  - Relacionado aos pacotes
- Dentro do mesmo pacote
  - **protected** é o mesmo que **public**
- Em pacotes distintos
  - **protected** é o mesmo que **private**

**ATENÇÃO:** não indicar um especificador de acesso é parecido com `protected`... Depois veremos diferença!

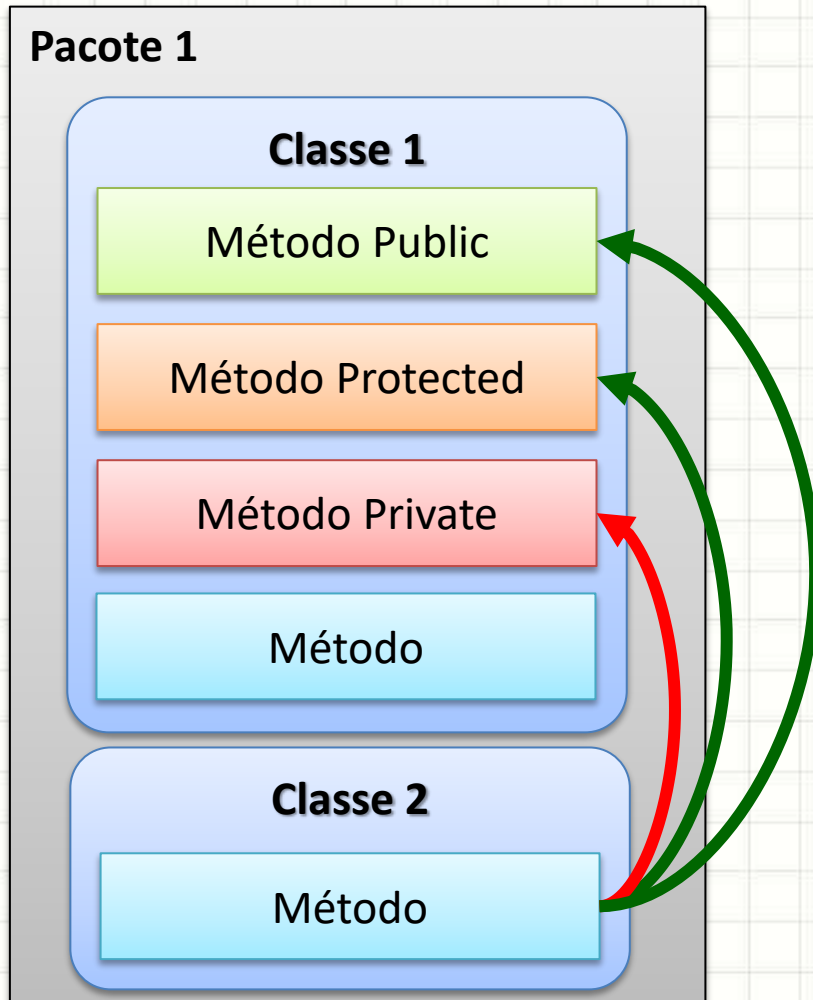
# Especificadores de Acesso

- Compreenda os acessos



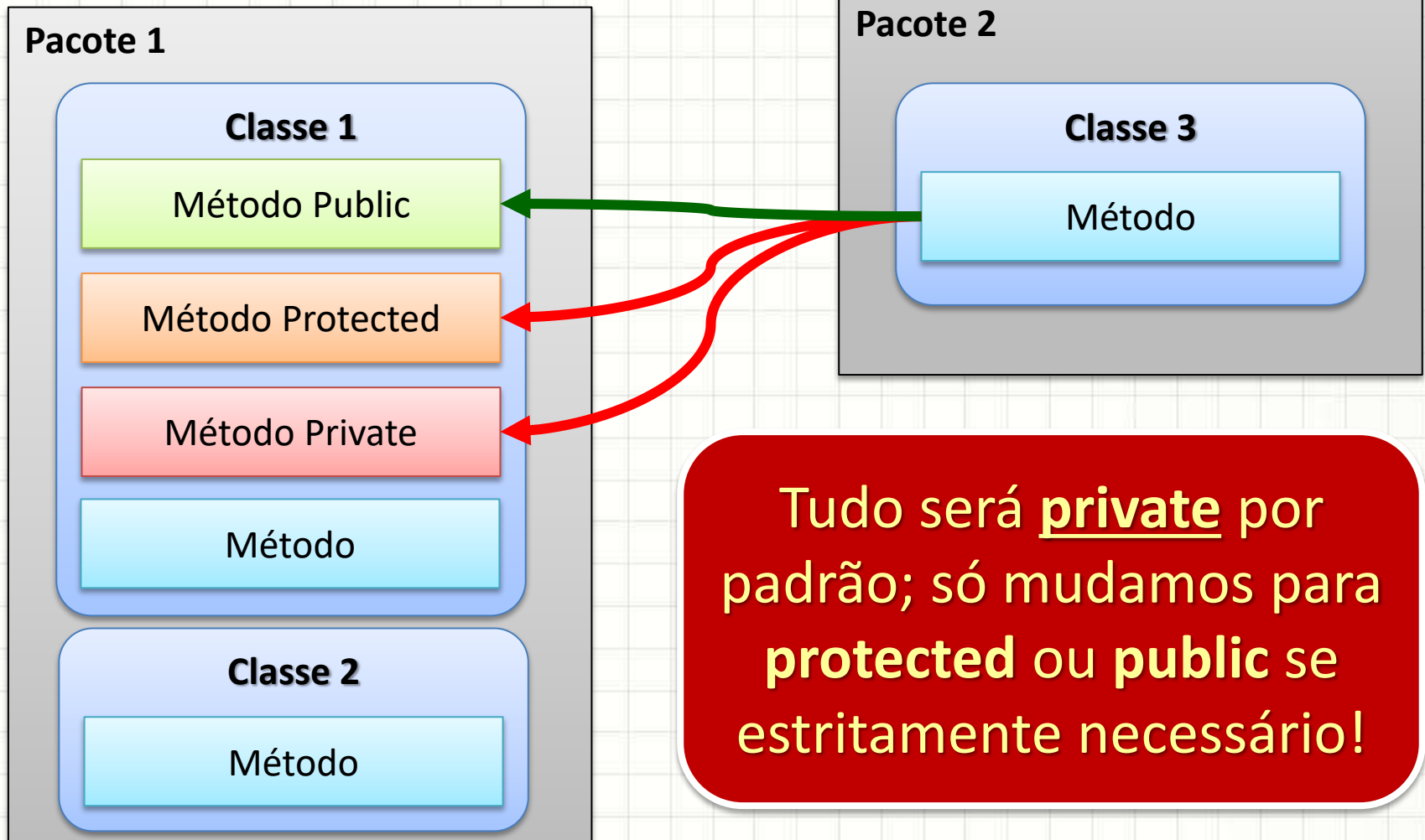
# Especificadores de Acesso

- Compreenda os acessos



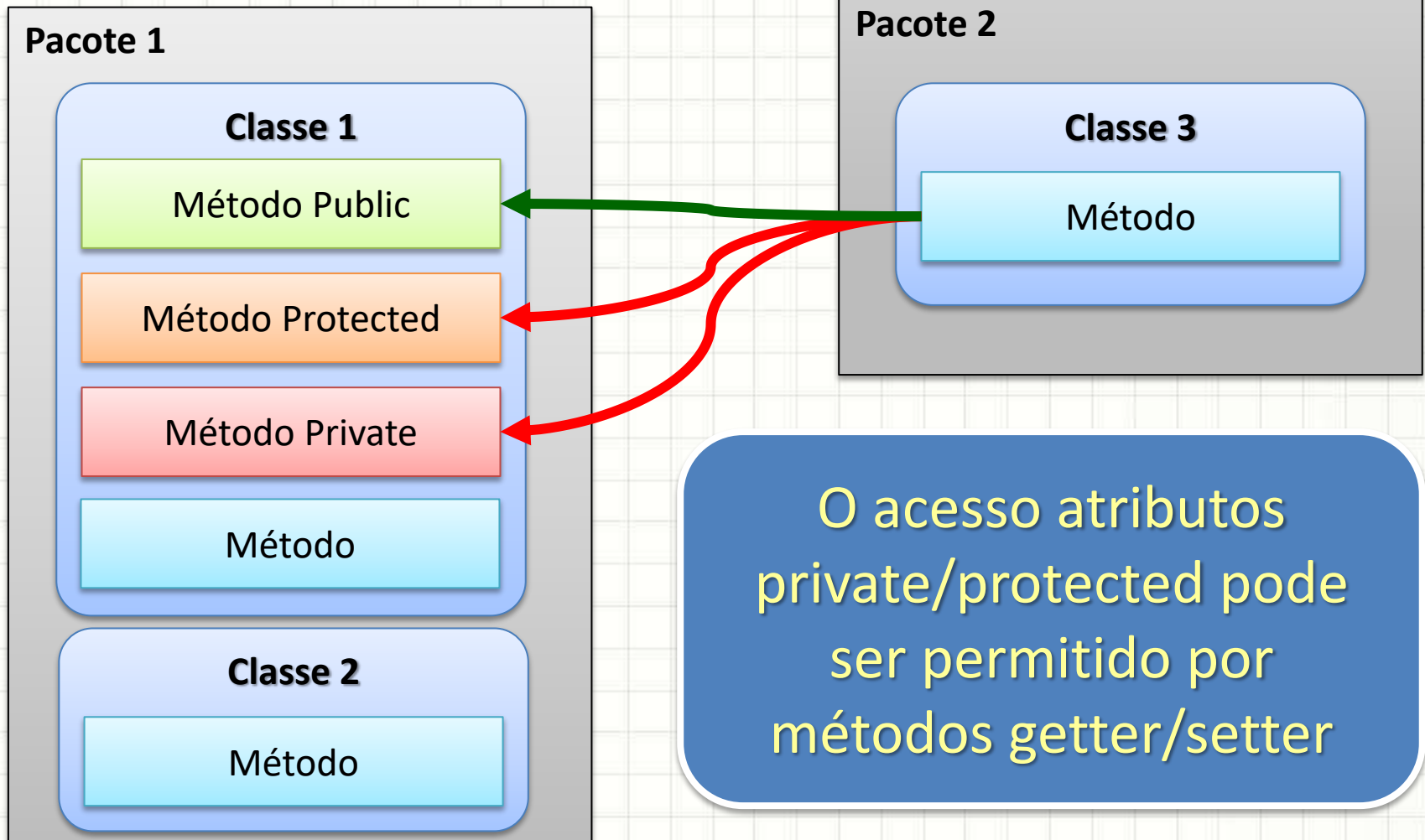
# Especificadores de Acesso

- Compreenda os acessos



# Especificadores de Acesso

- Compreenda os acessos





# OBJETOS EM JAVA

# O que são Objetos?

- Ok, aprendemos a criar uma classe, mas como usamos isso?
- Bem, como foi apresentado na aula anterior, uma das funções das classes é permitir a definição de tipos de dados não nativos, para **guardar dados de formato específico**
- Assim, a classe define apenas o **tipo** do dado; definir uma classe não significa que já podemos guardar informações

# O que são Objetos?

- Uma classe é como uma planta baixa de uma casa: diz como uma casa é, mas não dá pra morar nela!





# O que são Objetos?

- U
- u
- p

**É preciso construir**

o dá





# O que são Objetos?

- U  
u  
p

Em um OBJETO

o dá



# Como criar objetos?

- Operador **new**



**Classe**



**Objeto**

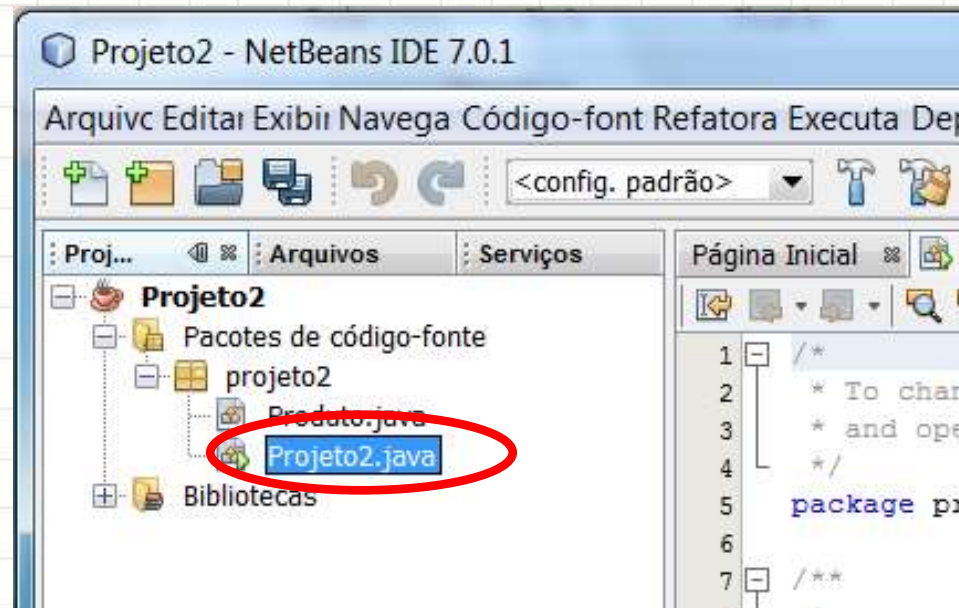
- Ex.: classe Cliente

`Cliente oCliente = new Cliente();`

O objeto **oCliente** é uma instância da classe **Cliente**

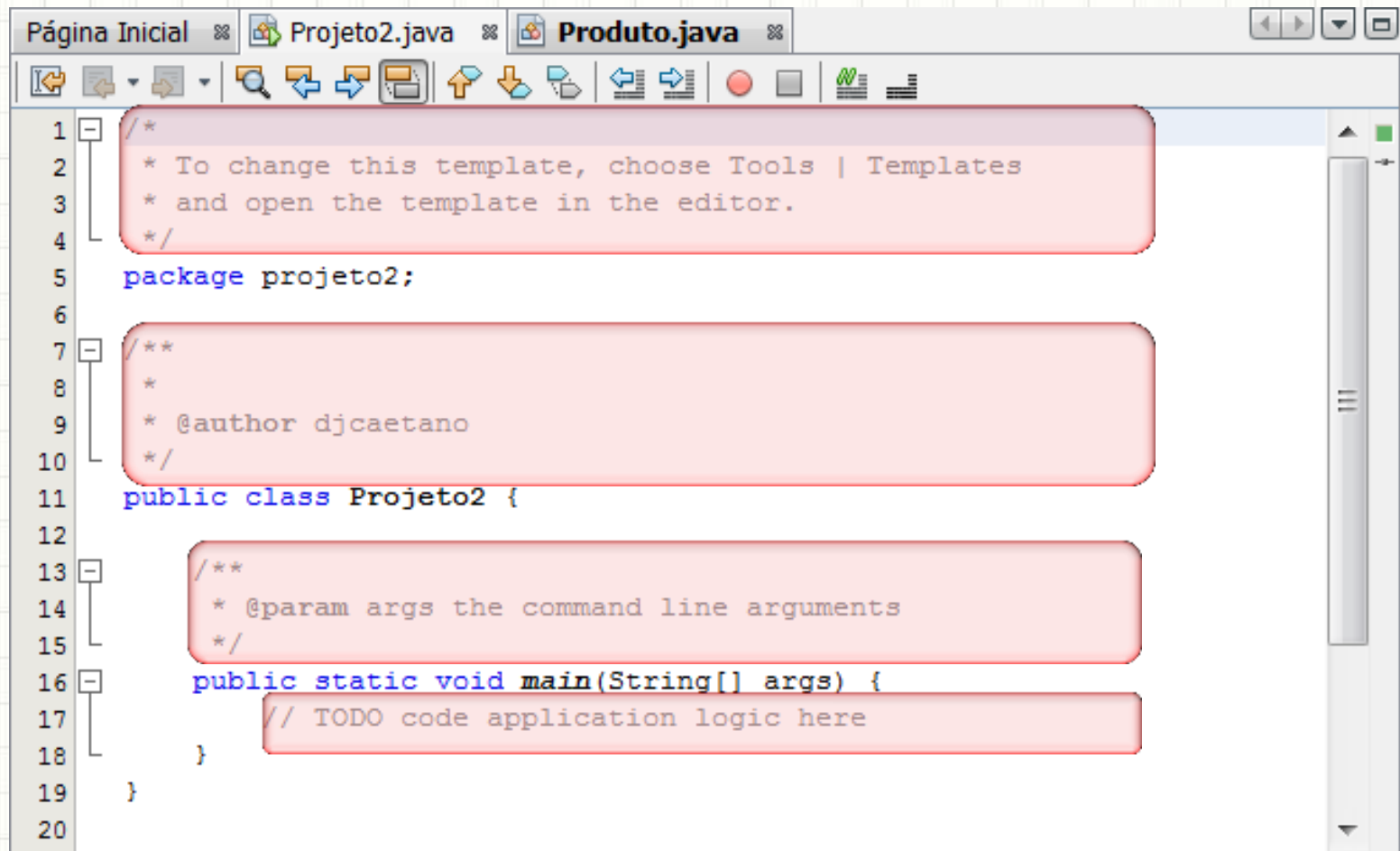
# Como criar objetos?

- Vamos começar do começo...
- Clique duas vezes no ícone do arquivo da classe principal (aquela com o nome do projeto):



# Como criar objetos?

- Apague os comentários do NetBeans...



```
Página Inicial x Projeto2.java x Produto.java x
package projeto2;

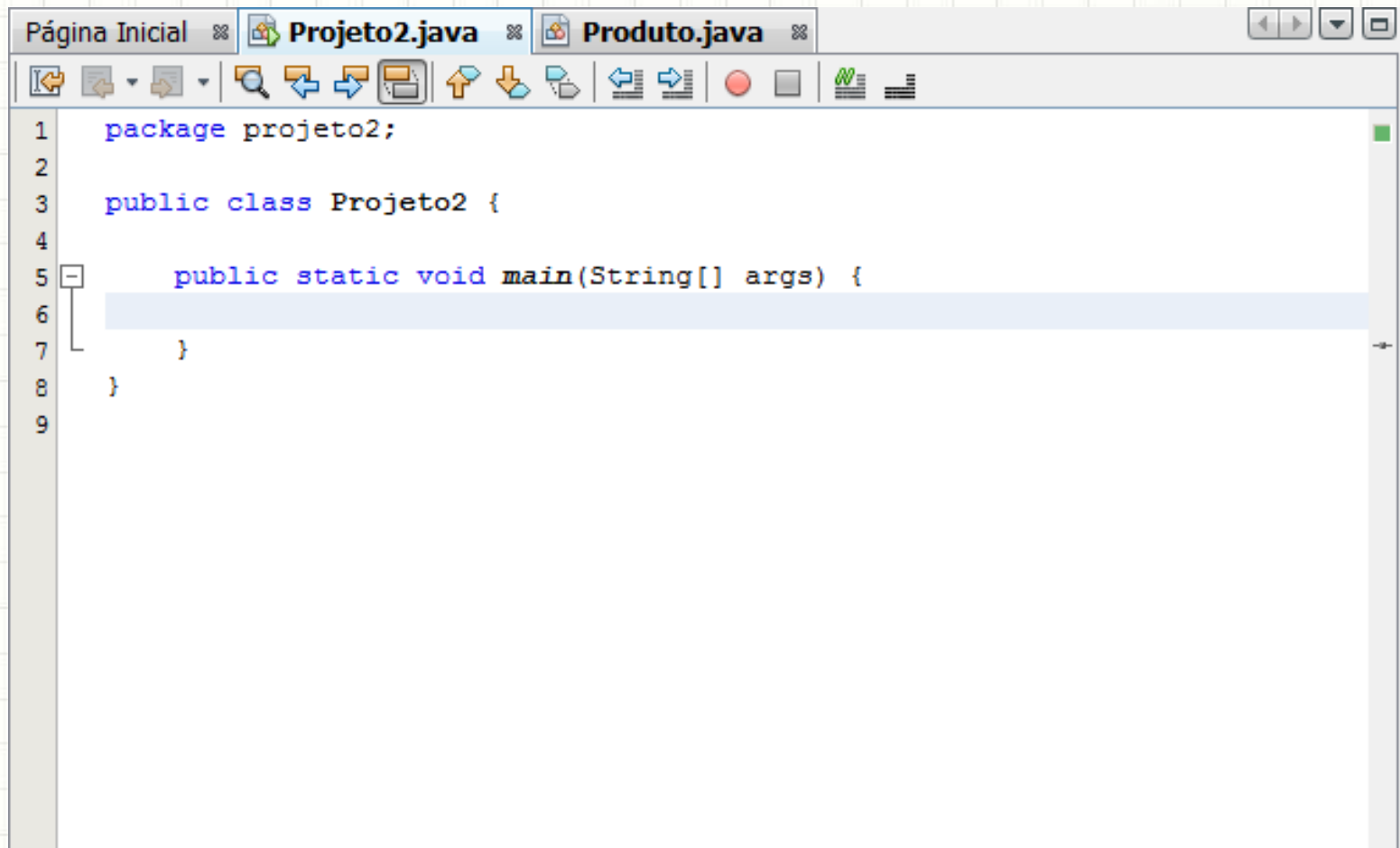
/**
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author djcaetano
 */
public class Projeto2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

# Como criar objetos?

- Apague os comentários do NetBeans...

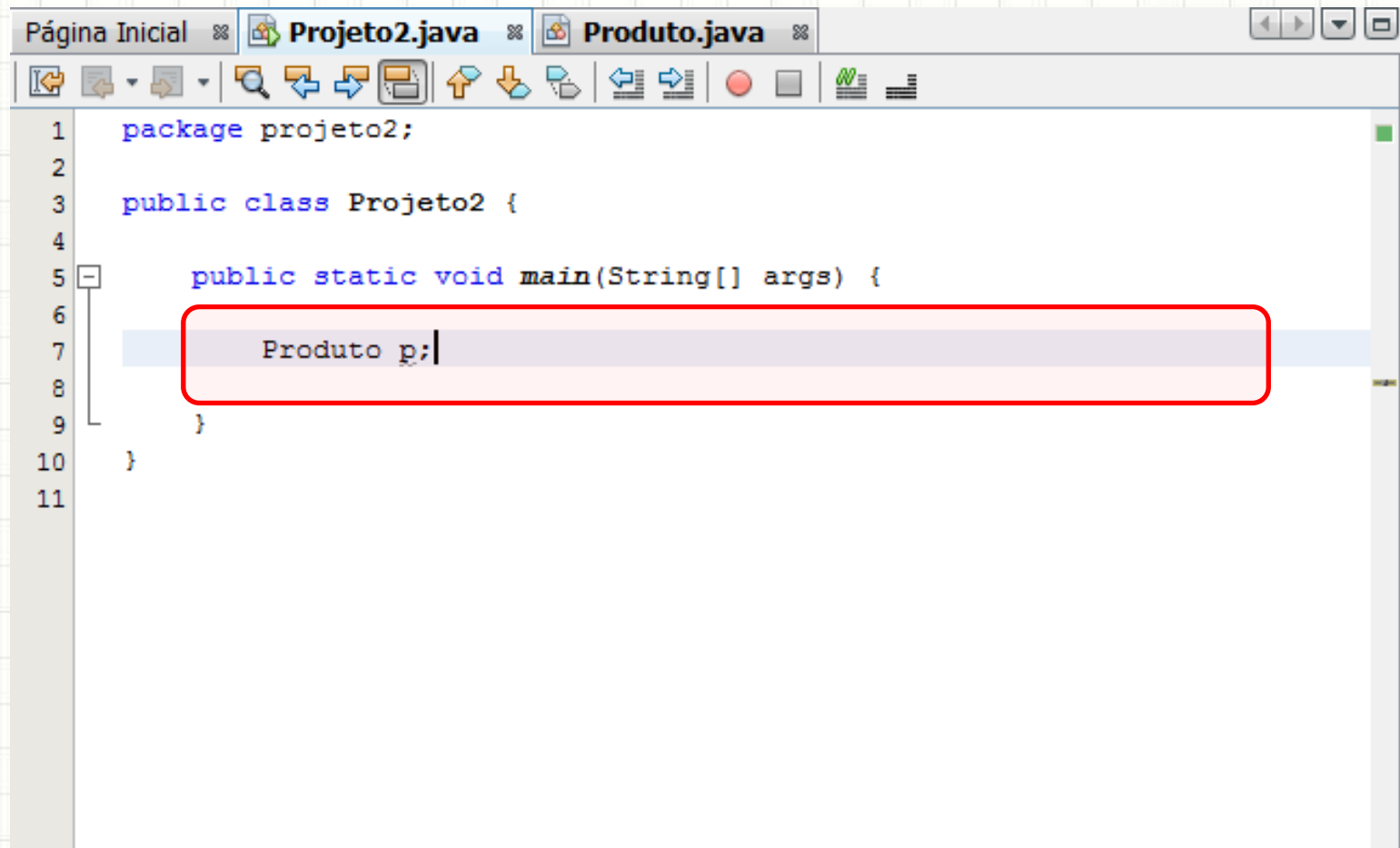


The screenshot shows the NetBeans IDE interface. The top toolbar contains various icons for navigation and editing. The main editor window displays the following Java code:

```
1 package projeto2;  
2  
3 public class Projeto2 {  
4  
5     public static void main(String[] args) {  
6  
7     }  
8 }  
9
```

# Como criar objetos?

- Vamos declarar uma variável para o produto

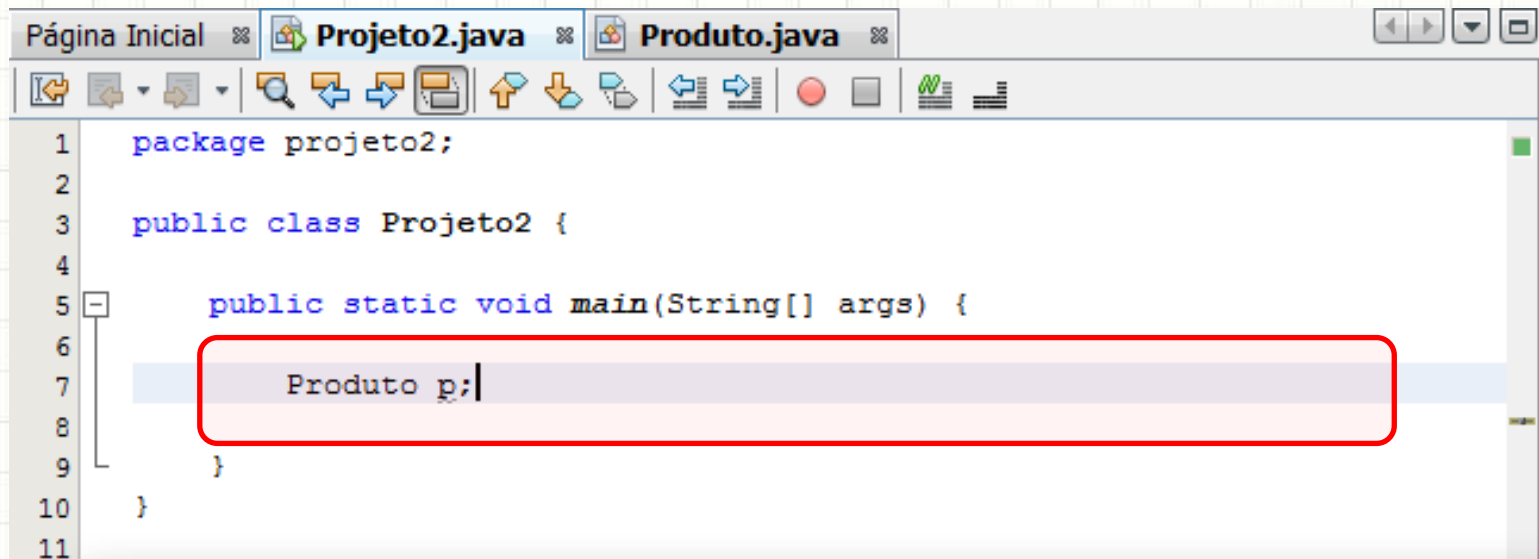


```
1 package projeto2;
2
3 public class Projeto2 {
4
5     public static void main(String[] args) {
6         Produto p;
7     }
8 }
9
10
11
```



# Como criar objetos?

- Vamos declarar uma variável para o produto

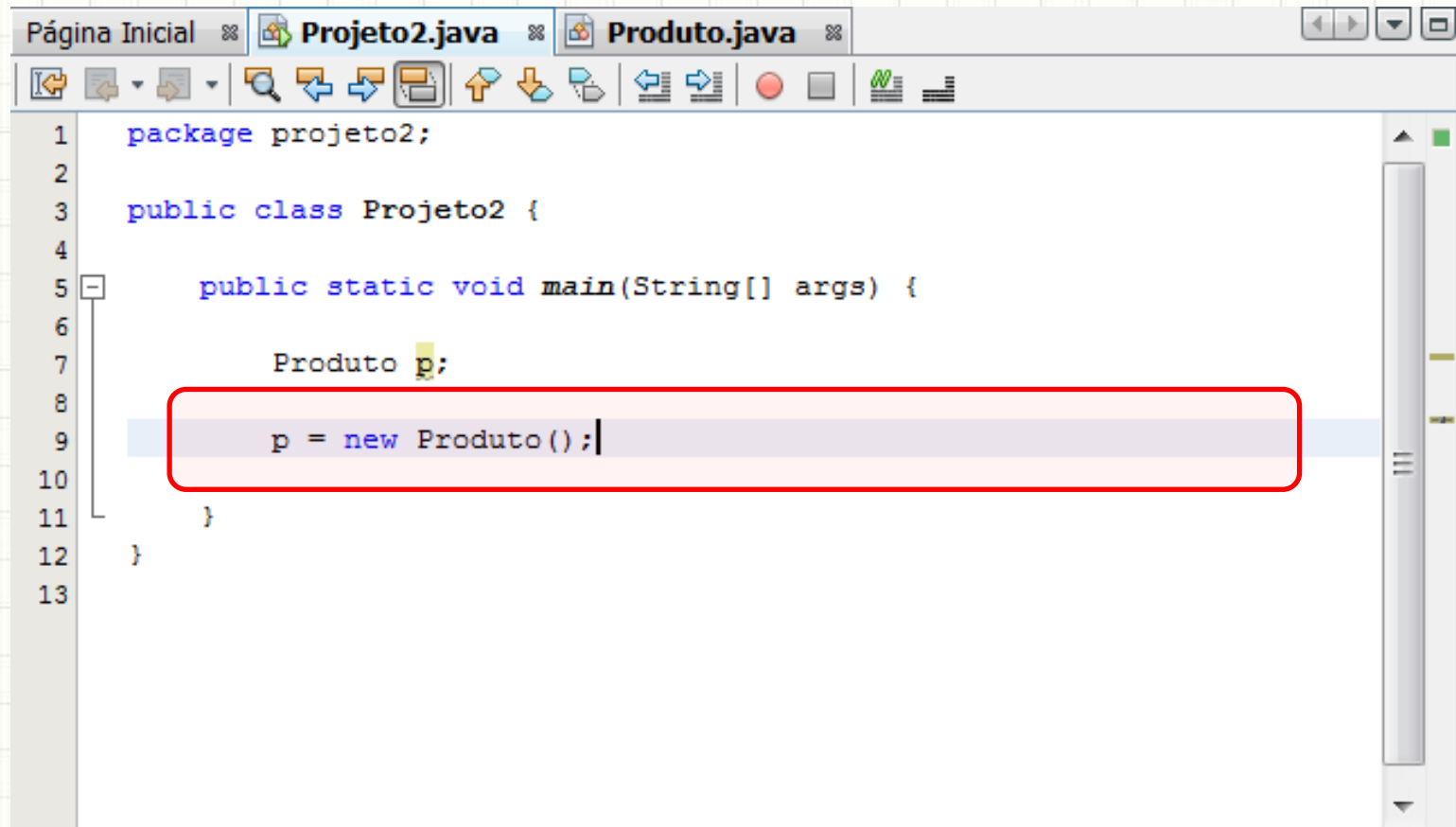


```
1 package projeto2;
2
3 public class Projeto2 {
4
5     public static void main(String[] args) {
6
7         Produto p;
8
9     }
10 }
11
```

Produto p;

# Como criar objetos?

- Agora vamos **criar o objeto**



The screenshot shows an IDE window with two tabs: 'Página Inicial' and 'Projeto2.java'. The code in 'Projeto2.java' is as follows:

```
1 package projeto2;
2
3 public class Projeto2 {
4
5     public static void main(String[] args) {
6
7         Produto p;
8
9         p = new Produto();
10
11     }
12 }
13
```

The line `p = new Produto();` is highlighted with a red box, indicating the creation of a new object.

# Como criar objetos?

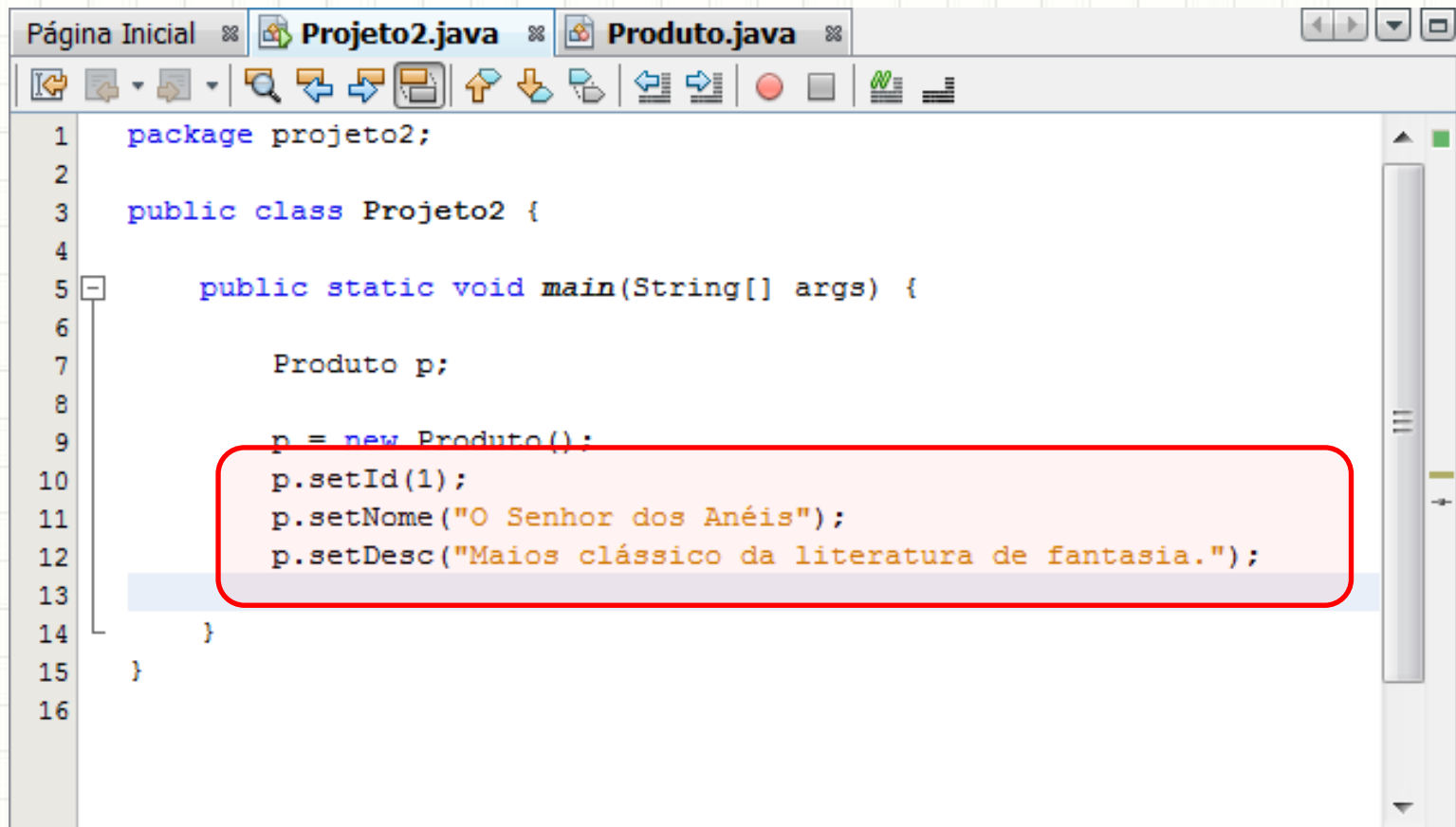
- Agora vamos **criar o objeto**

```
p = new Produto(); |
```

```
4  
5 public static void main(String[] args) {  
6  
7     Produto p;  
8  
9     p = new Produto(); |  
10  
11 }  
12 }  
13
```

# Como criar objetos?

- E armazenar algumas informações



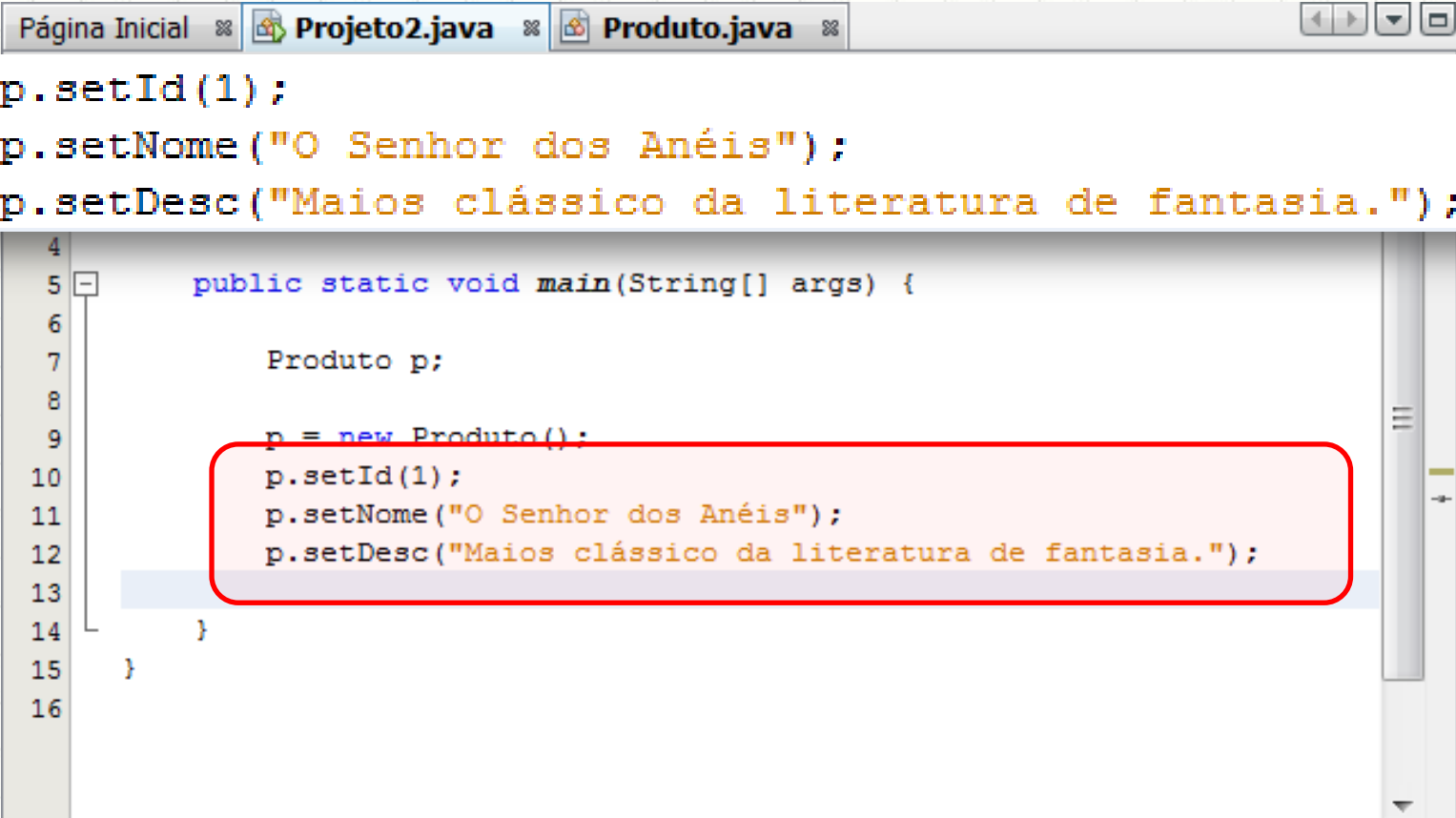
The screenshot shows an IDE window with two tabs: 'Página Inicial' and 'Projeto2.java'. The code in 'Projeto2.java' is as follows:

```
1 package projeto2;
2
3 public class Projeto2 {
4
5     public static void main(String[] args) {
6
7         Produto p;
8
9         p = new Produto();
10        p.setId(1);
11        p.setNome("O Senhor dos Anéis");
12        p.setDesc("Maiores clássico da literatura de fantasia.");
13
14    }
15 }
16
```

The lines 9 through 12 are highlighted with a red rounded rectangle, indicating the object creation and initialization process.

# Como criar objetos?

- E armazenar algumas informações



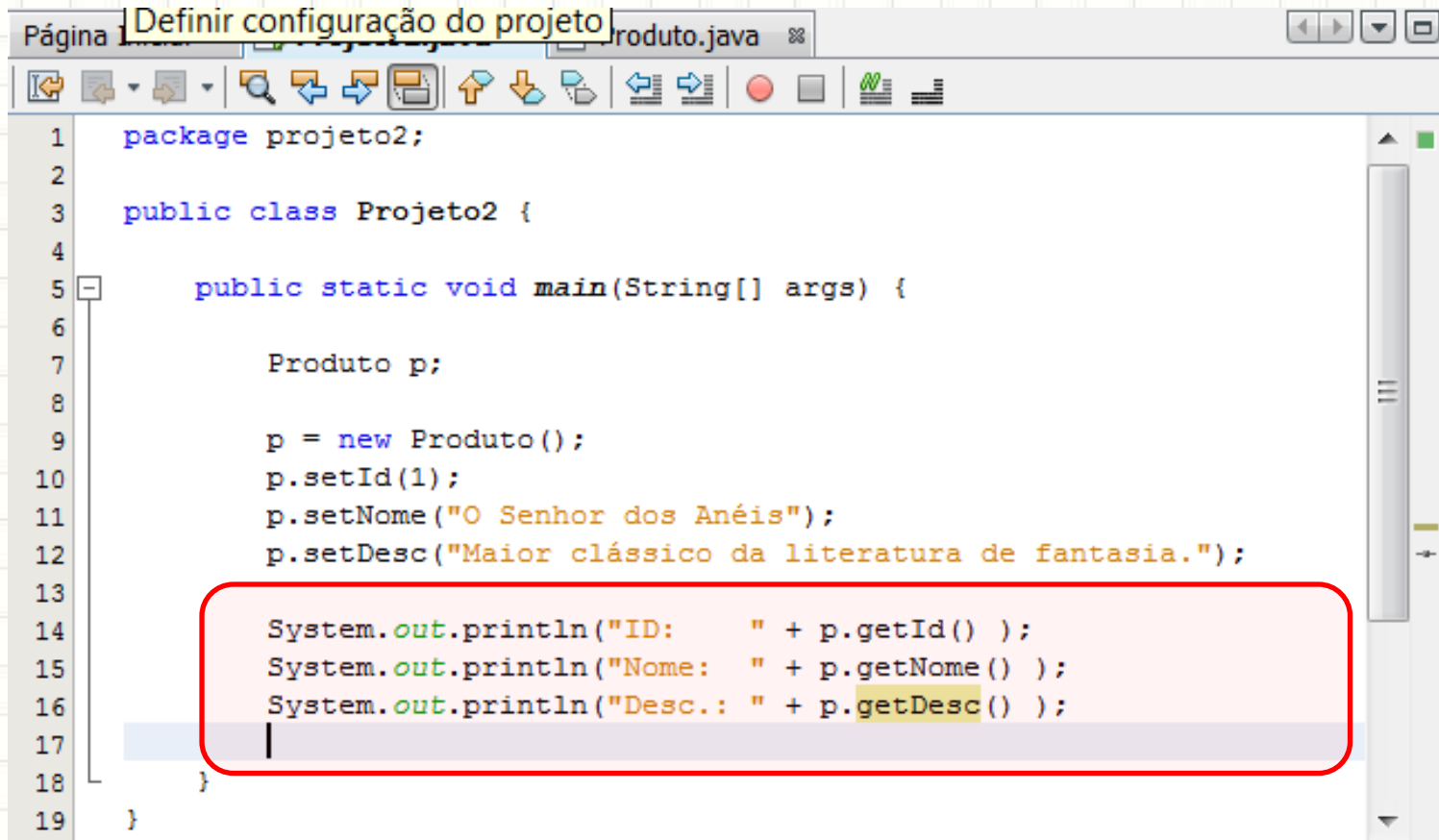
```
Página Inicial ✖ Projeto2.java ✖ Produto.java ✖
```

```
p.setId(1);  
p.setNome("O Senhor dos Anéis");  
p.setDesc("Maios clássico da literatura de fantasia.");
```

```
4  
5 public static void main(String[] args) {  
6  
7     Produto p;  
8  
9     p = new Produto();  
10    p.setId(1);  
11    p.setNome("O Senhor dos Anéis");  
12    p.setDesc("Maios clássico da literatura de fantasia.");  
13  
14 }  
15 }  
16
```

# Como criar objetos?

- E, agora, vamos imprimir alguns dados...



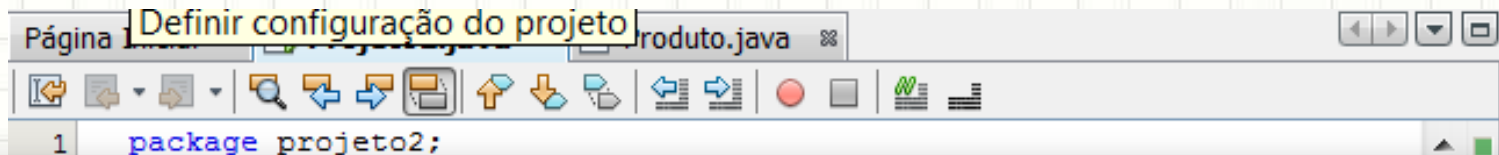
The screenshot shows an IDE window titled 'Definir configuração do projeto' with a tab for 'produto.java'. The code is as follows:

```
1 package projeto2;
2
3 public class Projeto2 {
4
5     public static void main(String[] args) {
6
7         Produto p;
8
9         p = new Produto();
10        p.setId(1);
11        p.setNome("O Senhor dos Anéis");
12        p.setDesc("Maior clássico da literatura de fantasia.");
13
14        System.out.println("ID:      " + p.getId() );
15        System.out.println("Nome:   " + p.getNome() );
16        System.out.println("Desc.: " + p.getDesc() );
17
18    }
19 }
```

The lines 14-16, which contain the print statements, are highlighted with a red rounded rectangle.

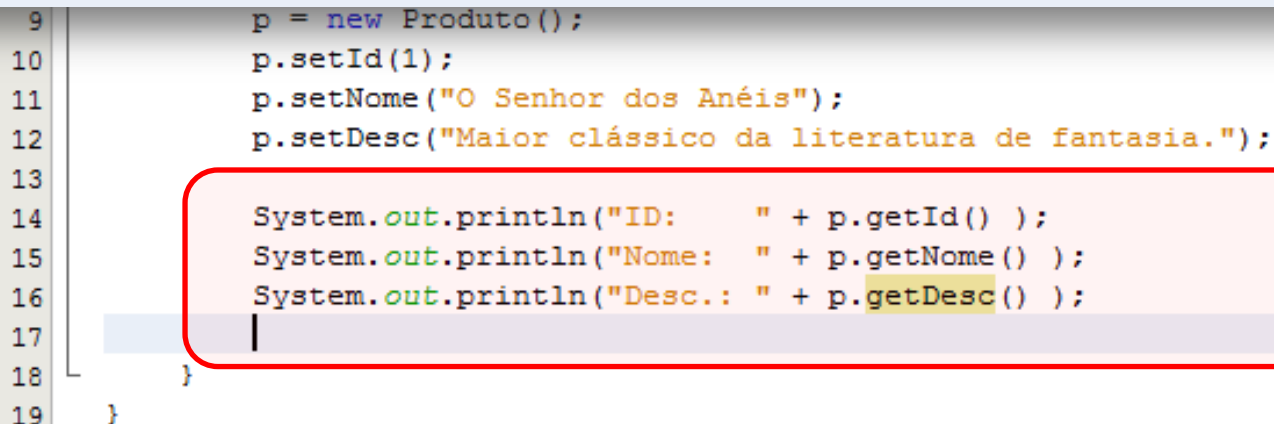
# Como criar objetos?

- E, agora, vamos imprimir alguns dados...



```
Página Definir configuração do projeto produto.java x
1 package projeto2;
```

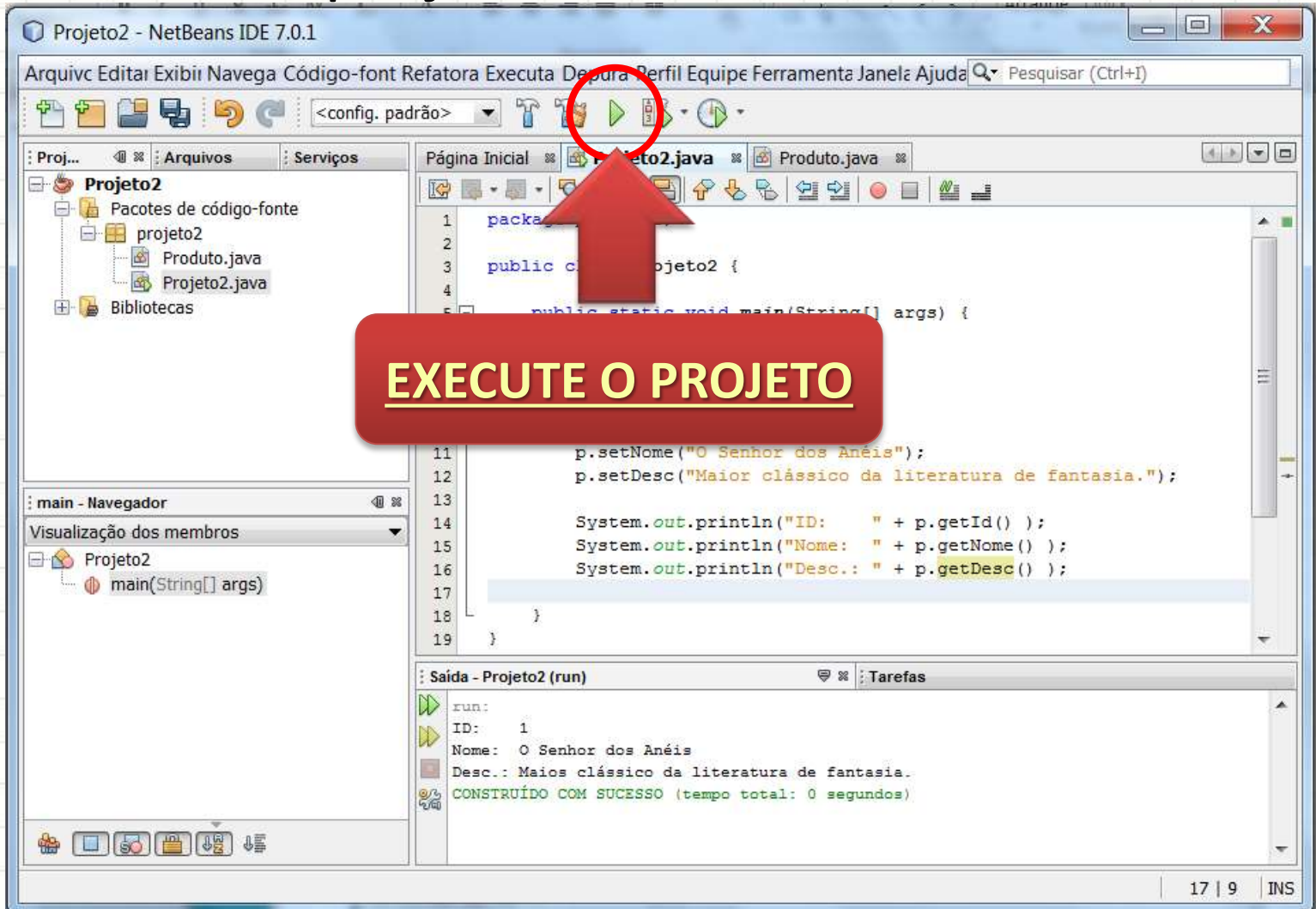
```
System.out.println("ID:      " + p.getId() );
System.out.println("Nome:    " + p.getNome() );
System.out.println("Desc.:  " + p.getDesc() );
```



```
9     p = new Produto();
10    p.setId(1);
11    p.setNome("O Senhor dos Anéis");
12    p.setDesc("Maior clássico da literatura de fantasia.");
13
14    System.out.println("ID:      " + p.getId() );
15    System.out.println("Nome:    " + p.getNome() );
16    System.out.println("Desc.:  " + p.getDesc() );
17
18    }
19 }
```

# Como criar objetos?

- Execute o projeto...



Projeto2 - NetBeans IDE 7.0.1

Arquivar Editar Exibir Navegar Código-fonte Refatora Executa Depura Perfil Equipe Ferramentas Janela Ajuda Pesquisar (Ctrl+I)

Projeto2

- Pacotes de código-fonte
  - projeto2
    - Produto.java
    - Projeto2.java
  - Bibliotecas

main - Navegador

Visualização dos membros

- Projeto2
  - main(String[] args)

```
1 package projeto2;
2
3 public class Projeto2 {
4
5     public static void main(String[] args) {
6
7         Produto p = new Produto(1, "O Senhor dos Anéis", "Maior clássico da literatura de fantasia.");
8         p.setNome("O Senhor dos Anéis");
9         p.setDesc("Maior clássico da literatura de fantasia.");
10
11         System.out.println("ID: " + p.getId() );
12         System.out.println("Nome: " + p.getNome() );
13         System.out.println("Desc.: " + p.getDesc() );
14     }
15 }
16
17
18
19
```

Saída - Projeto2 (run)

```
run:
ID: 1
Nome: O Senhor dos Anéis
Desc.: Maior clássico da literatura de fantasia.
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

17 | 9 | INS



# Como criar objetos?

- E veja o resultado...

**VEJA O RESULTADO**

```
package projeto2;

public class Projeto2 {

    public static void main(String[] args) {

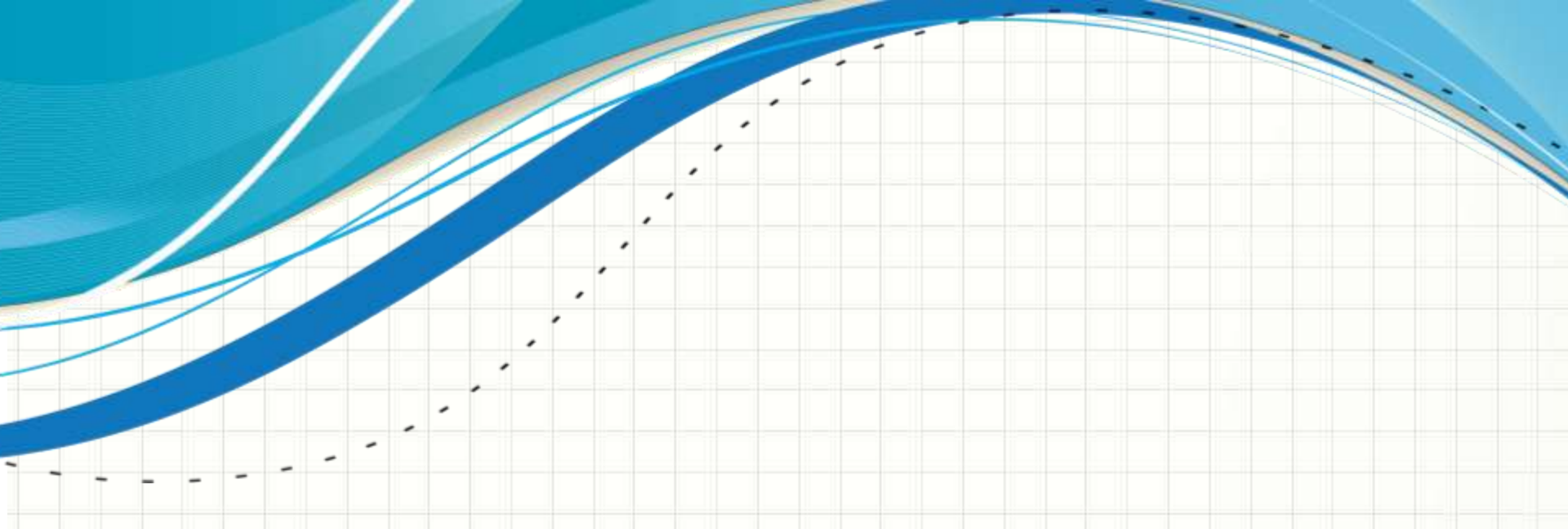
        Produto p;

        p = new Produto(1, "O Senhor dos Anéis", "Maios clássico da literatura de fantasia.");

        System.out.println("ID: " + p.getId() );
        System.out.println("Nome: " + p.getNome() );
        System.out.println("Desc.: " + p.getDesc() );
    }
}
```

main - Navegador  
Visualização dos membros  
Projeto2  
main(String[] args)

Saida - Projeto2 (run)  
run:  
ID: 1  
Nome: O Senhor dos Anéis  
Desc.: Maios clássico da literatura de fantasia  
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)



# **CONSTRUTOR E DESTRUTOR**

# Construtor e Destrutor

- Seria útil ter um “método de configuração”
  - Define os valores iniciais dos atributos
- Método construtor
  - Tem o mesmo nome da classe
  - Obrigatoriamente público
  - Não retorna valor
  - É executado quando um objeto é instanciado
    - Automático!

# Construtor e Destrutor

- A ideia do destrutor seria similar
  - Execução quando objeto é deletado
- Não existe (?) em Java
  - Método **finalize()** → sem garantia de execução
    - Nunca recebe parâmetros
  - Executa durante o Garbage Collector
    - Pode ser apenas quando o programa fechar!

# Exemplos de Construtor

- Declarando um construtor com parâmetro

```
public Cliente(String cpf) {  
    ...  
}
```

- É possível passar parâmetros de criação

```
Cliente oCliente;  
oCliente = new Cliente("012.345.678-90");
```

# Exemplos de Construtor

- Exemplos já vistos em aula

```
Scanner teclado;
```

```
teclado = new Scanner(System.in);
```

```
String umTexto;
```

```
umTexto = new String("Olá mundo!");
```



**PERGUNTAS?**



# PARTE PRÁTICA



# Implementando uma Classe

- Vamos implementar a classe **Produto** vista em aula, com os atributos:
  - **id**: inteiro
  - **nome**: string
  - **desc**: string
- Obviamente vamos usar o main da classe do projeto para instanciar o objeto da classe **Produto**.

# Implementando uma Classe

- Vamos criar a classe Pessoa
  - **nome**: string
  - **idade**: int
- Obviamente vamos usar o main da classe do projeto para instanciar o objeto da classe **Pessoa**.



**ATIVIDADES**

# Atividade 1

- Implemente um método construtor que aceite os parâmetros **id**, **nome** e **desc** e configure o objeto com base nesses valores
- Modifique a função main do projeto para usar o novo construtor.

# Atividade 2

- Crie um novo projeto chamado **Aventura**
- Crie uma classe chamada **Item** que contenha os seguintes atributos:
  - id: int
  - nome: String
  - descricao: String
- Crie uma nova classe chamada **Sala** que contenha os seguintes atributos:
  - id: int
  - descricao: String