PROGRAMAÇÃO I

INTRODUÇÃO À ORIENTAÇÃO A OBJETOS

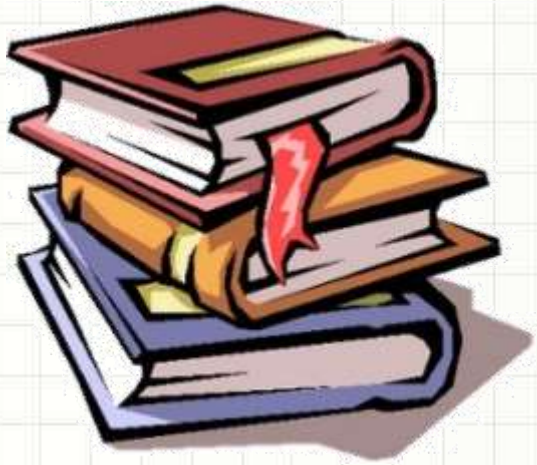
Prof. Dr. Daniel Caetano

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

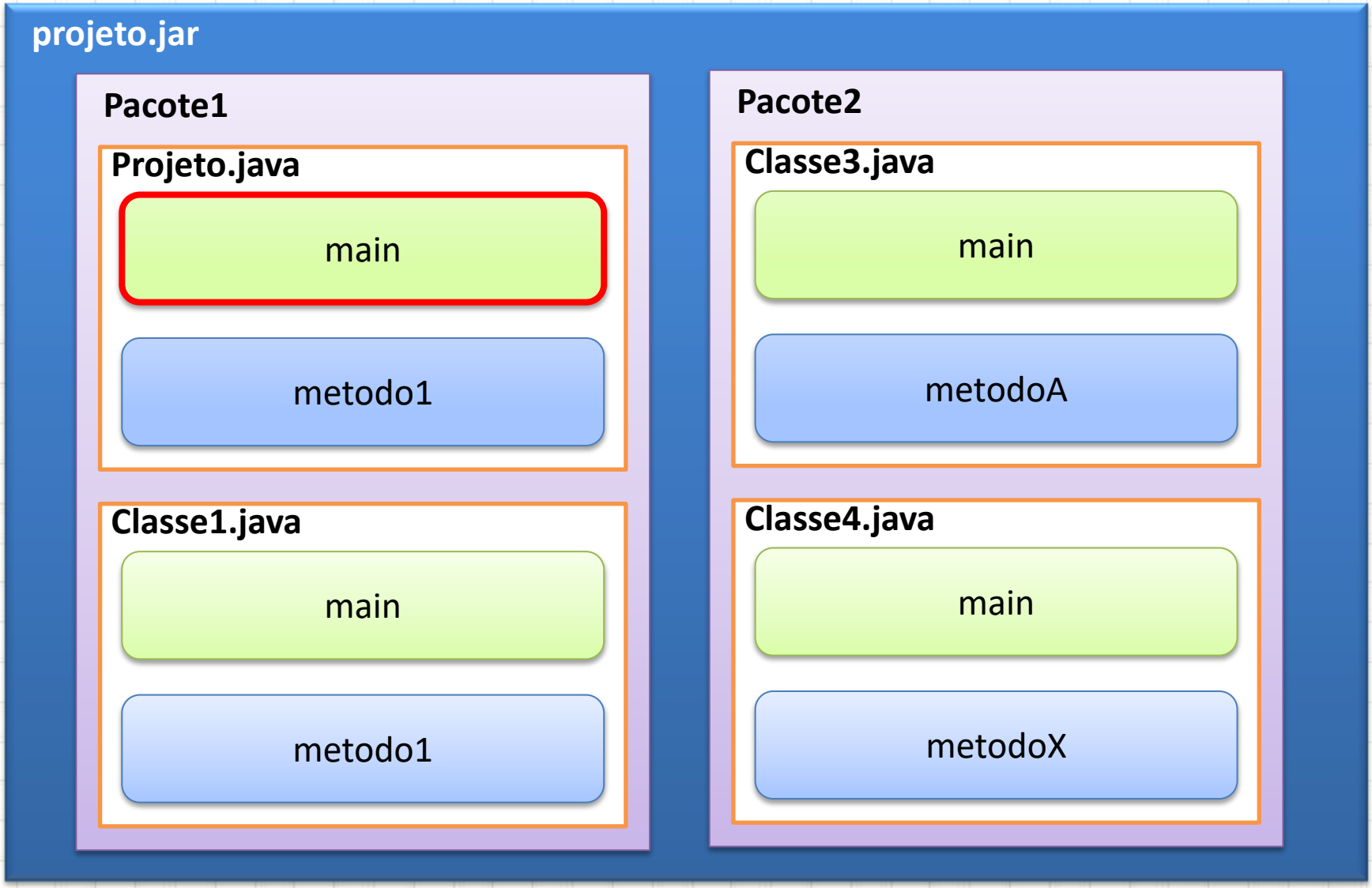
Material de Estudo



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



Classes em Java

O que são Classes?

- Inicialmente, não nos prendamos em definições formis
- 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?

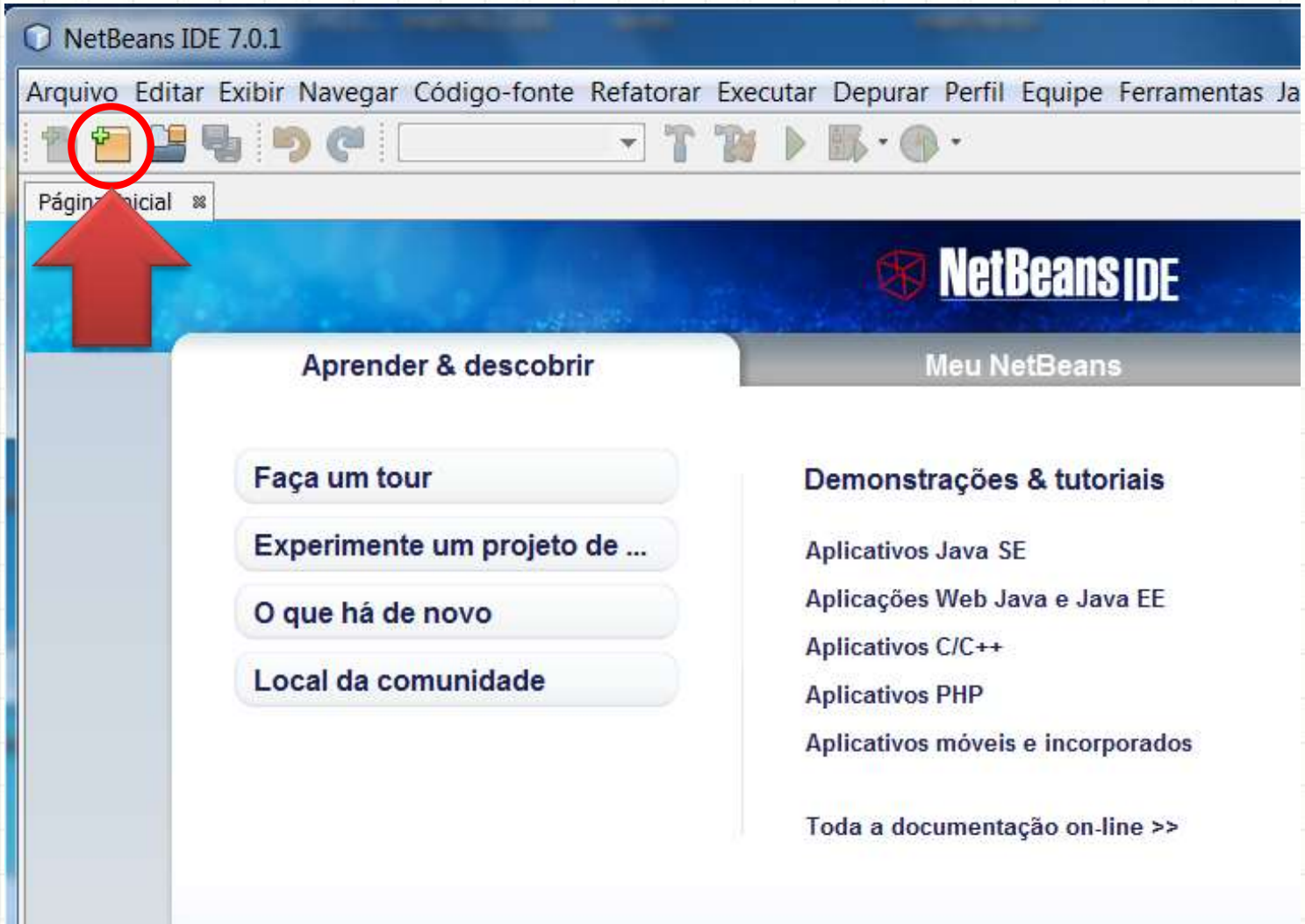
- Podemos imaginar uma classe assim:

Privados ou Públicos

Métodos

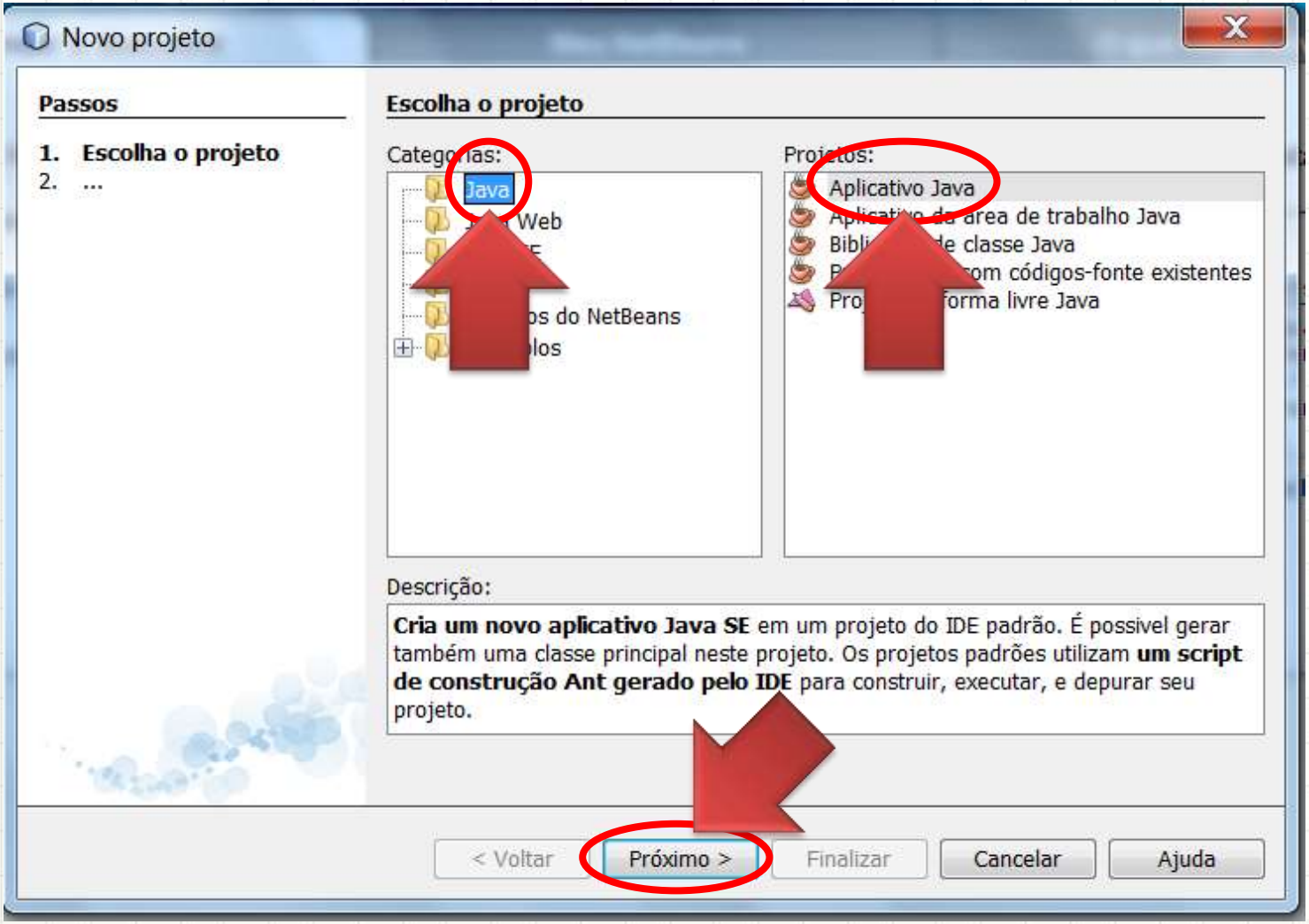
Definindo uma classe

- Iniciaremos criando um projeto



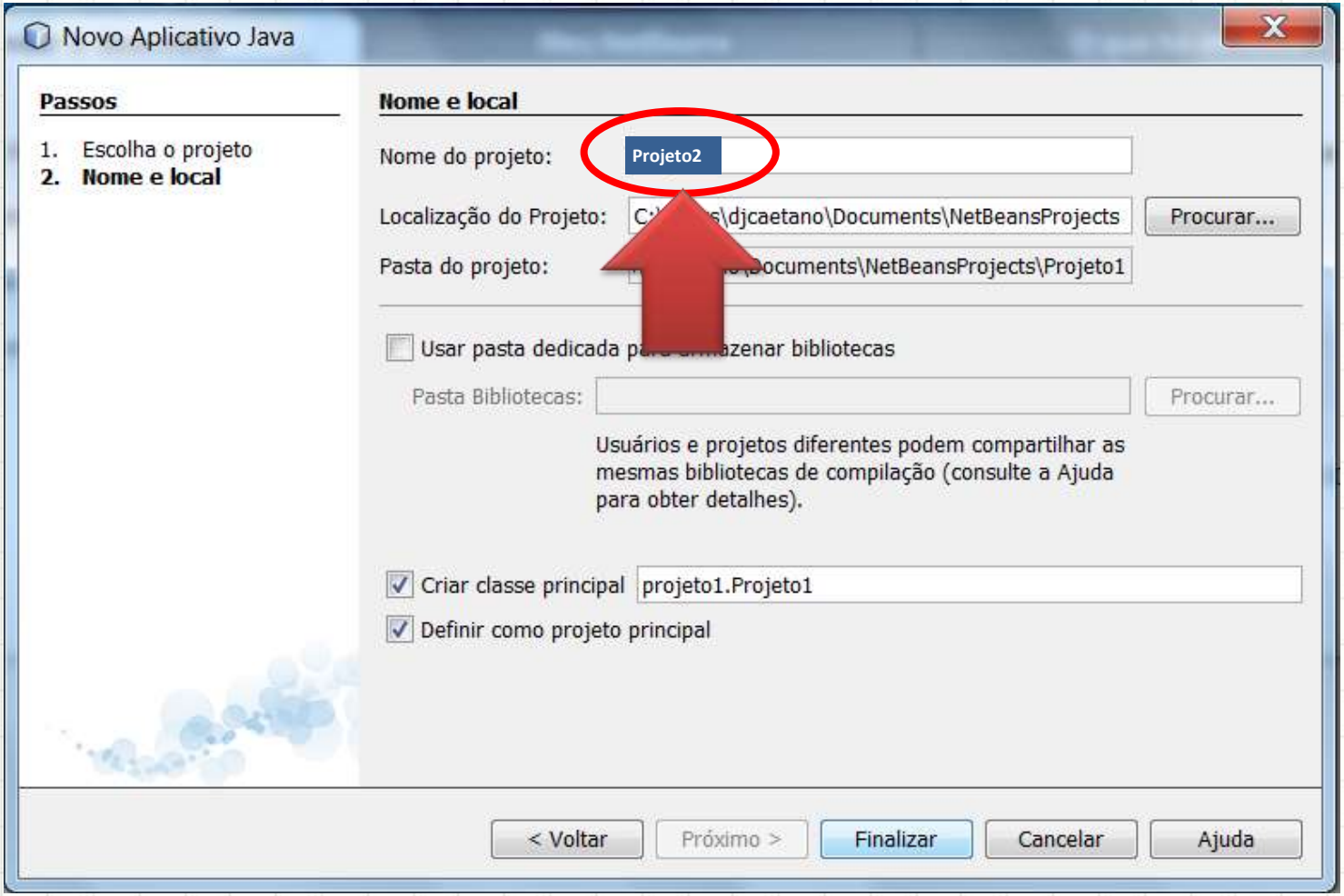
Definindo uma classe

- Escolha o tipo: **Java** e **Aplicativo Java**



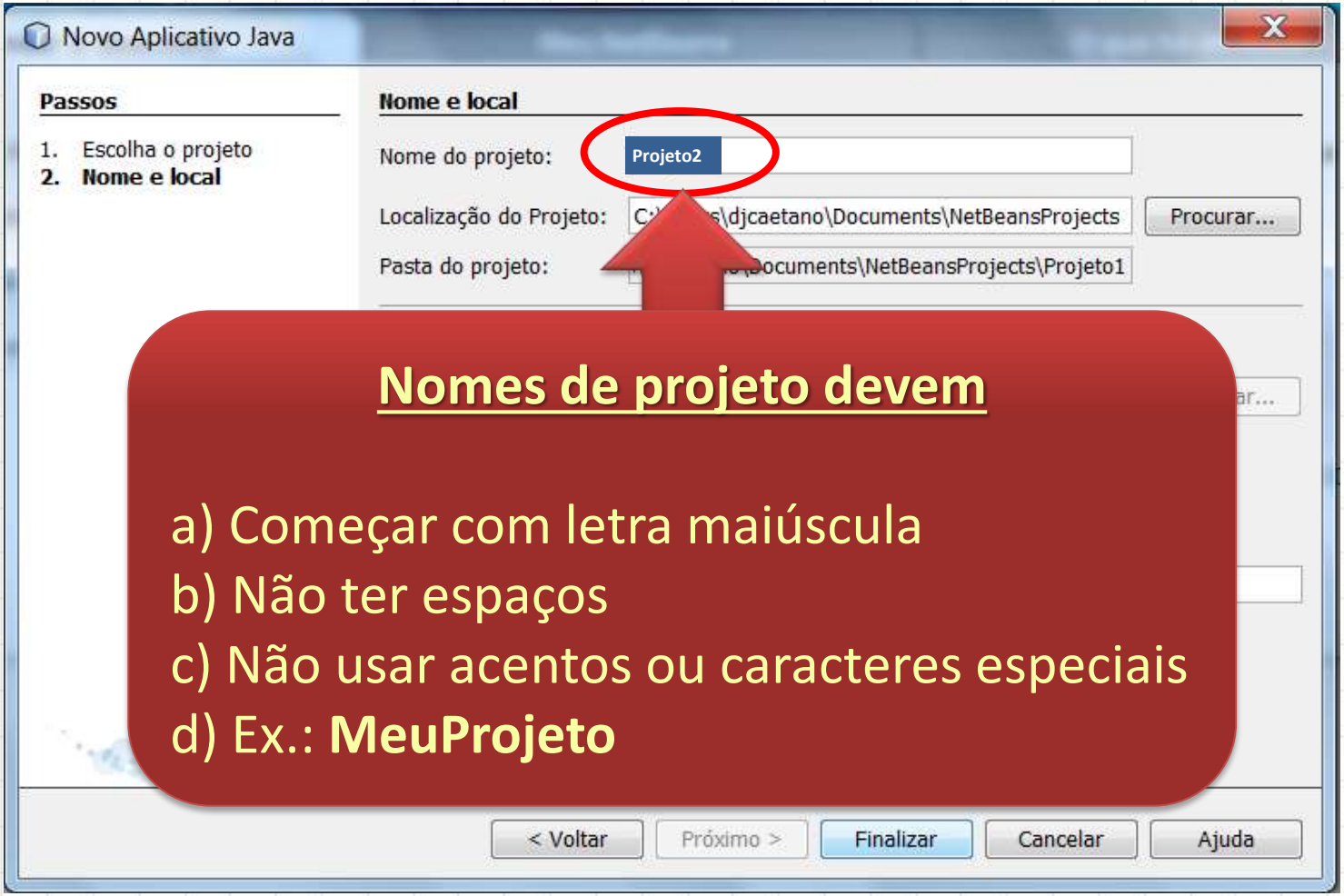
Definindo uma classe

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



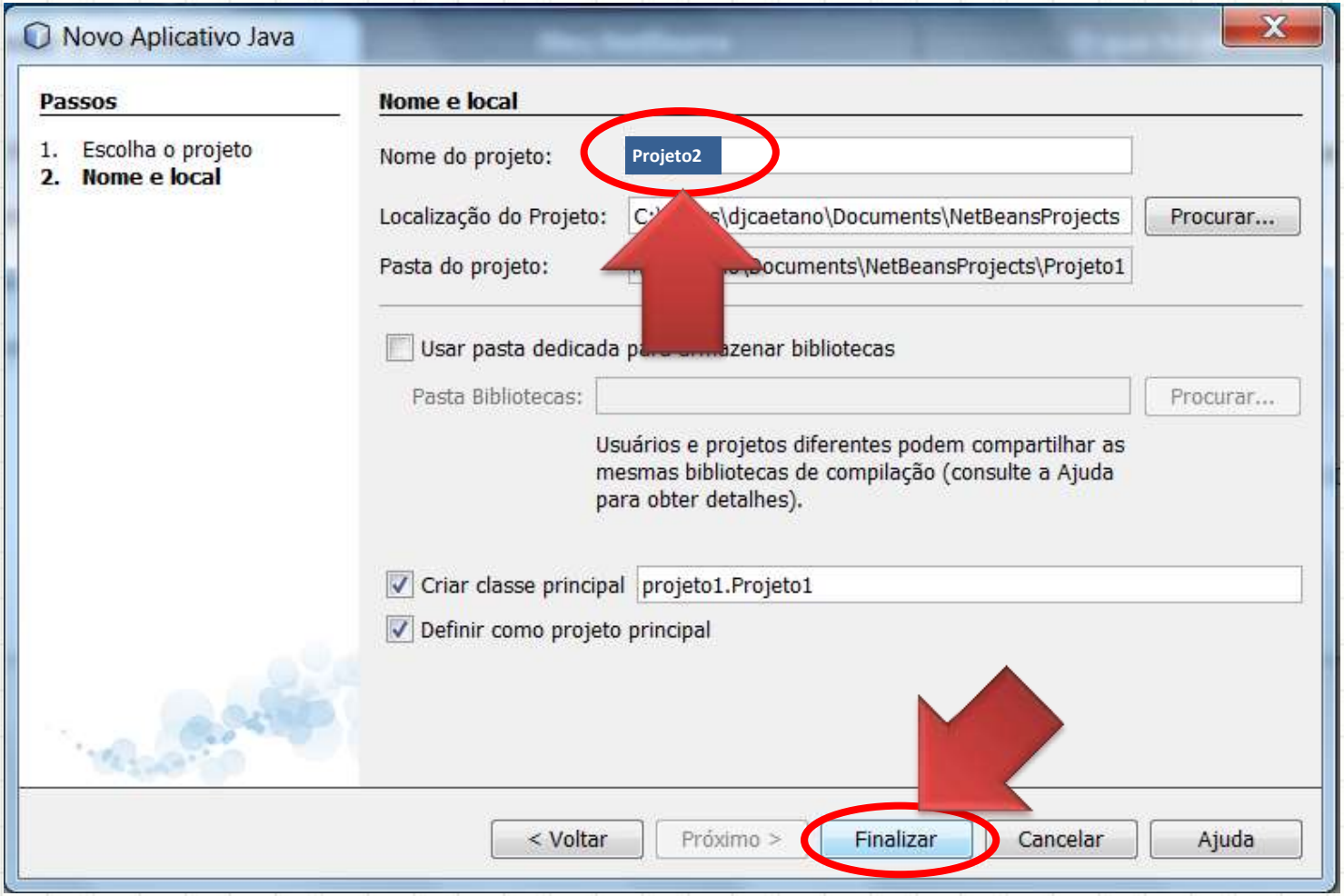
Definindo uma classe

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



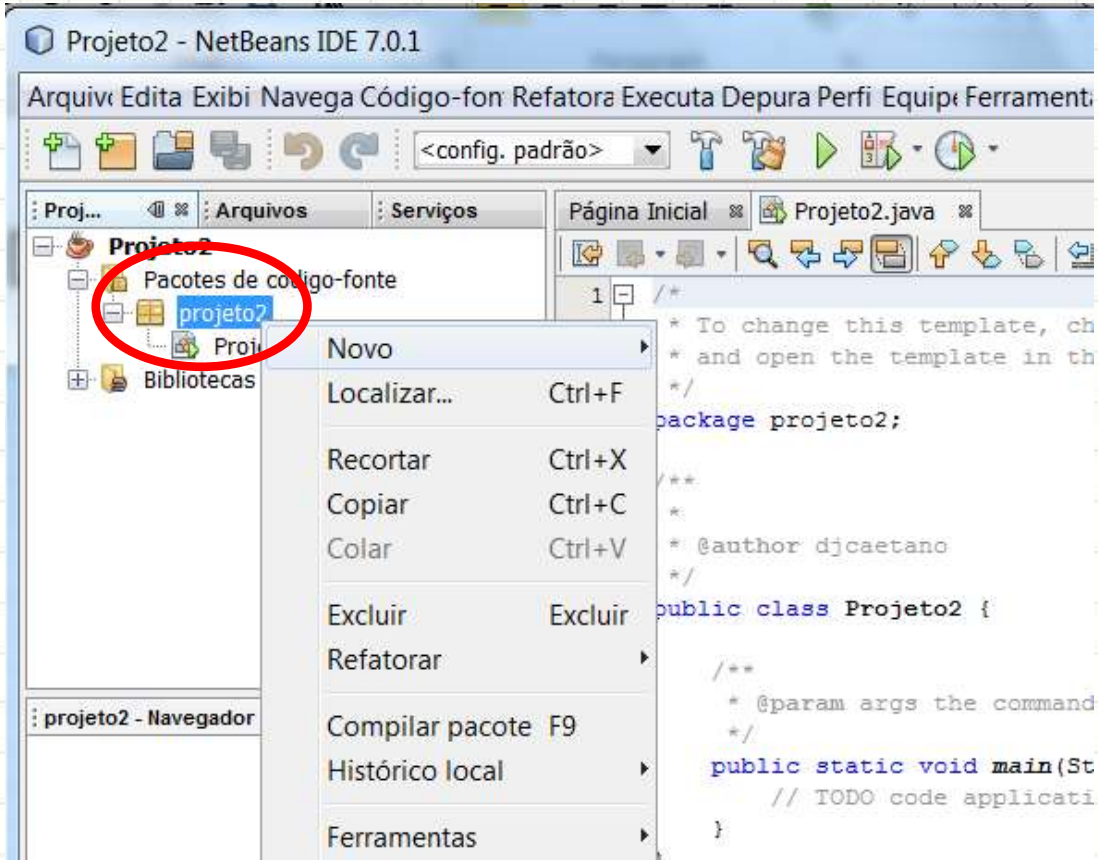
Definindo uma classe

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



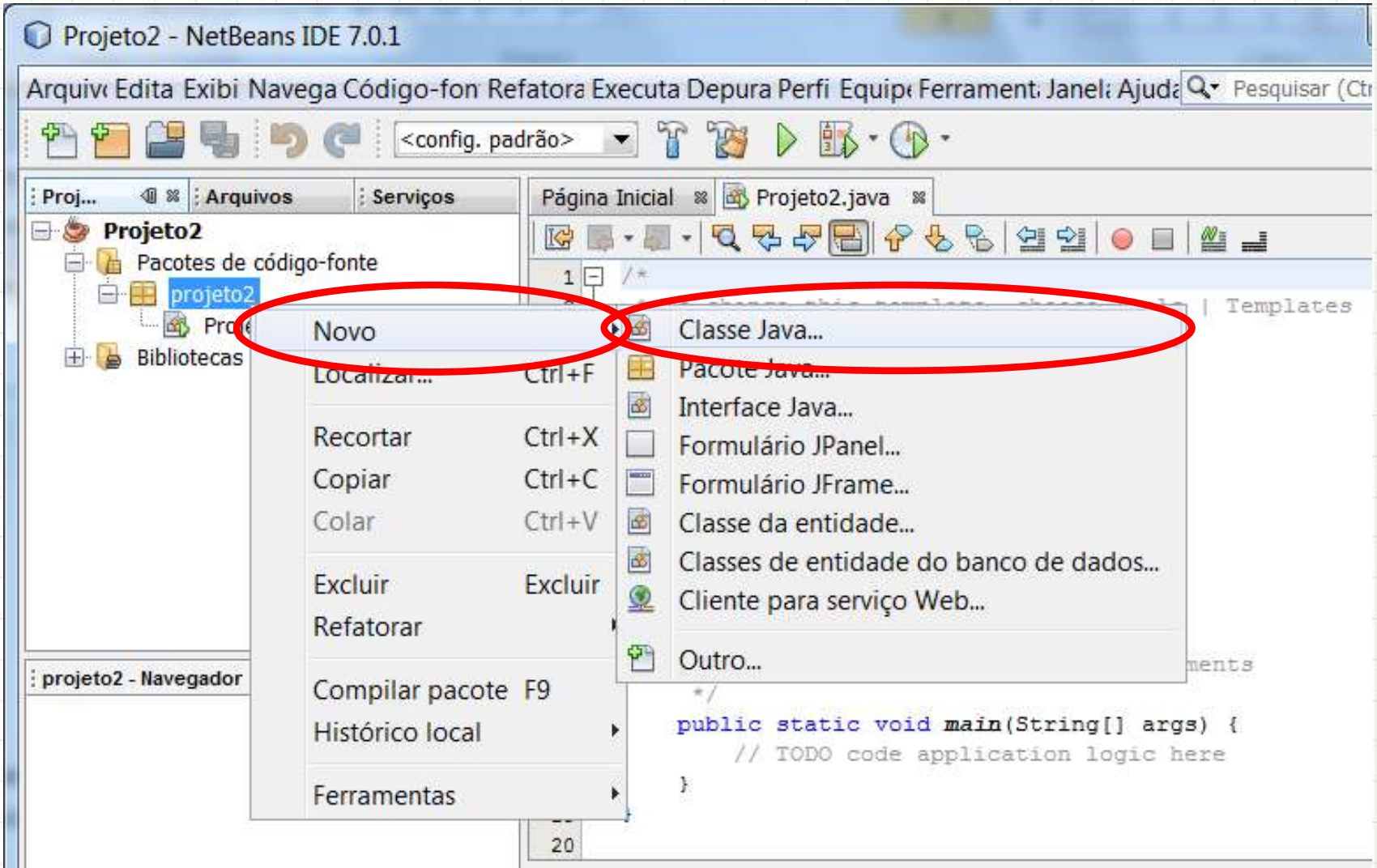
Definindo uma Classe

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



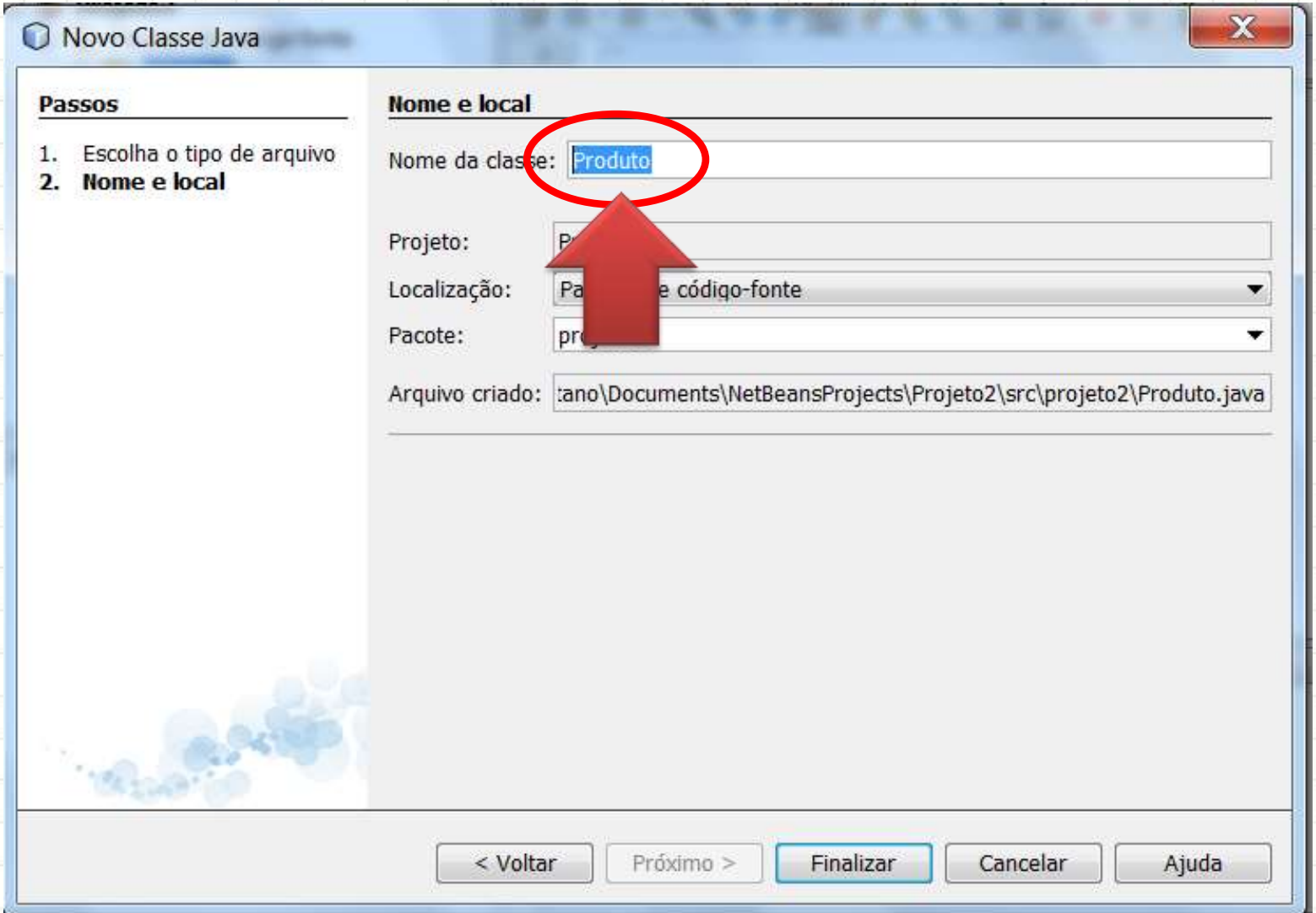
Definindo uma Classe

- Agora selecione **novo** > **classe java**



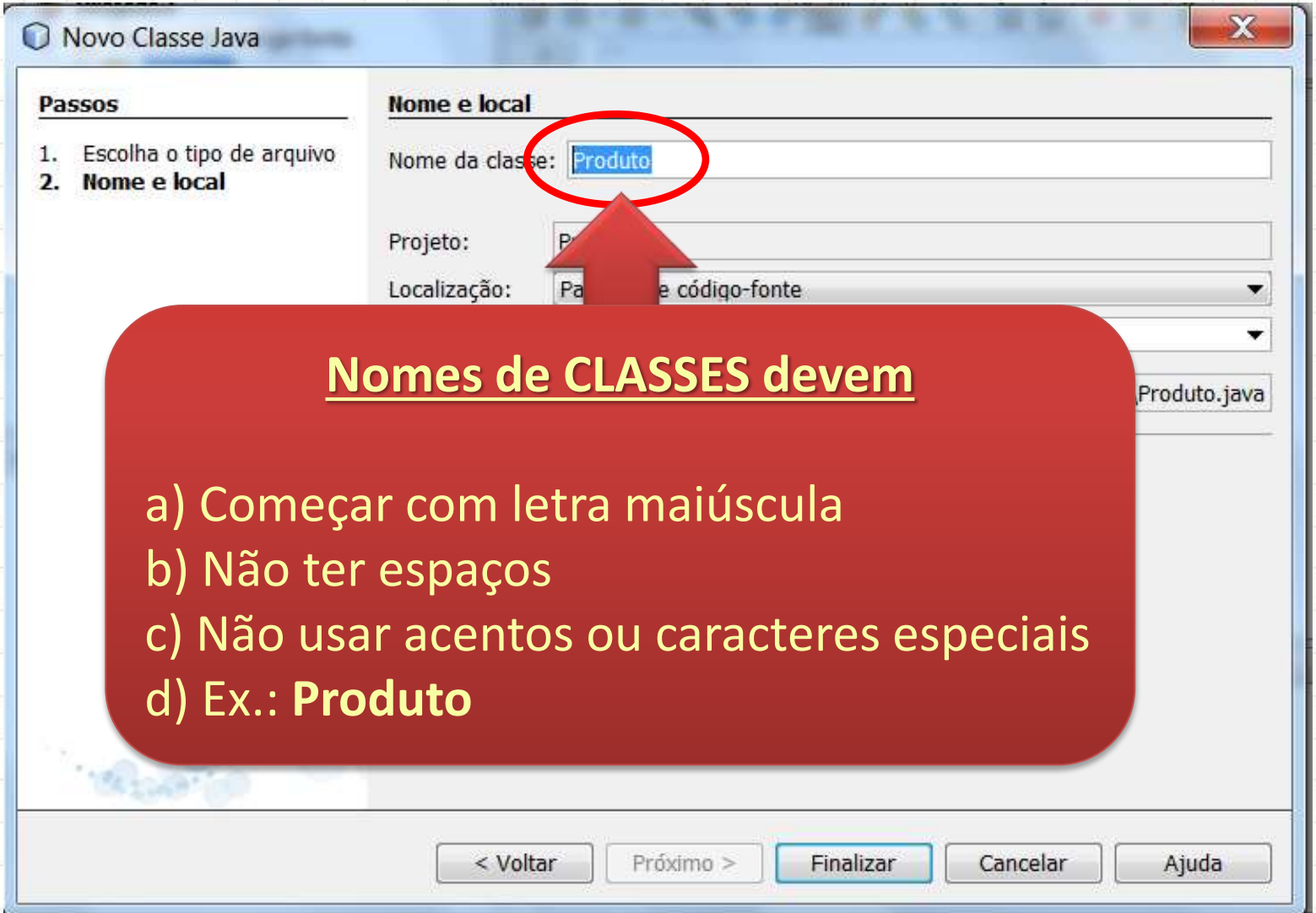
Definindo uma Classe

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



Definindo uma Classe

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

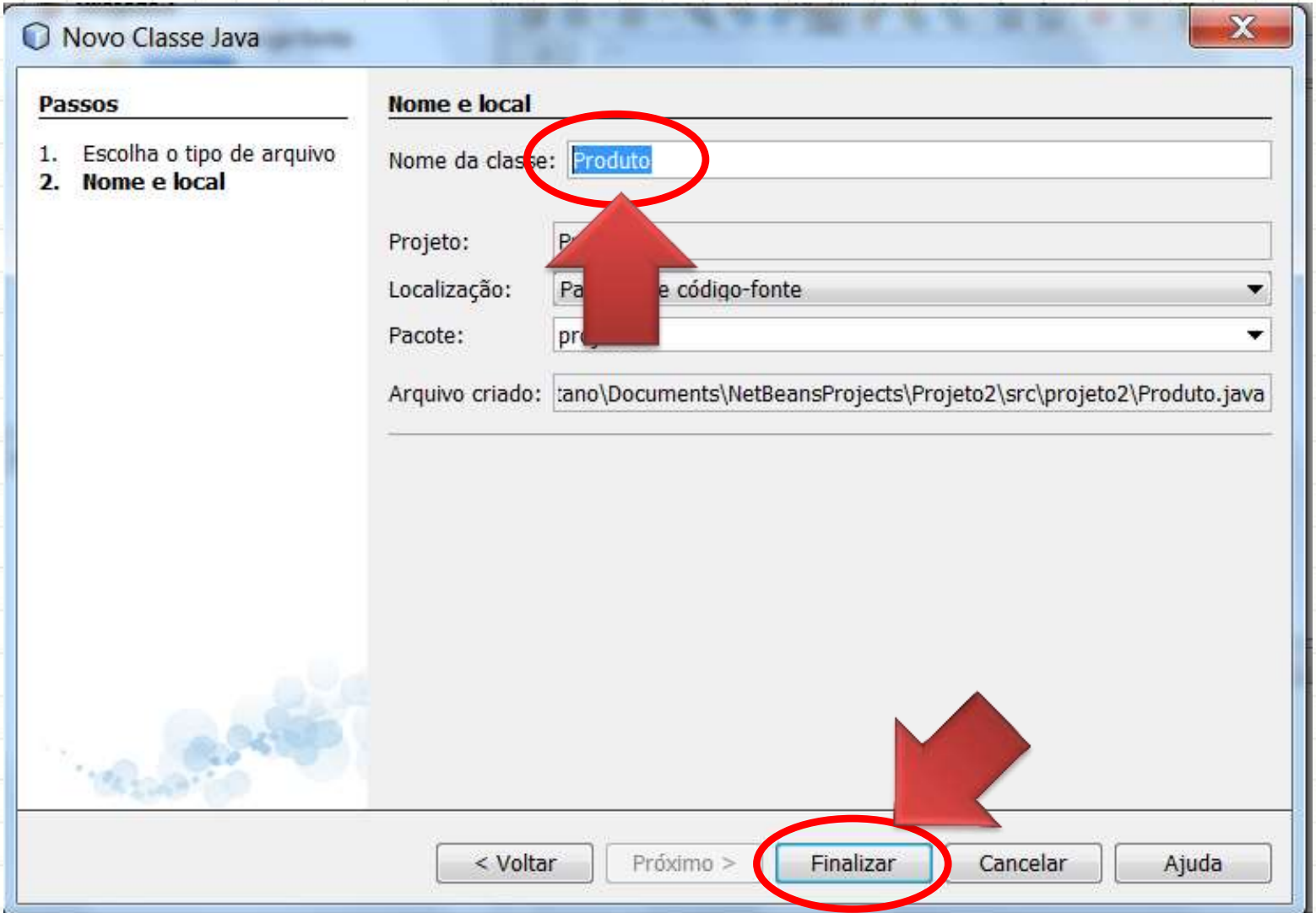


Nomes de CLASSES devem

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

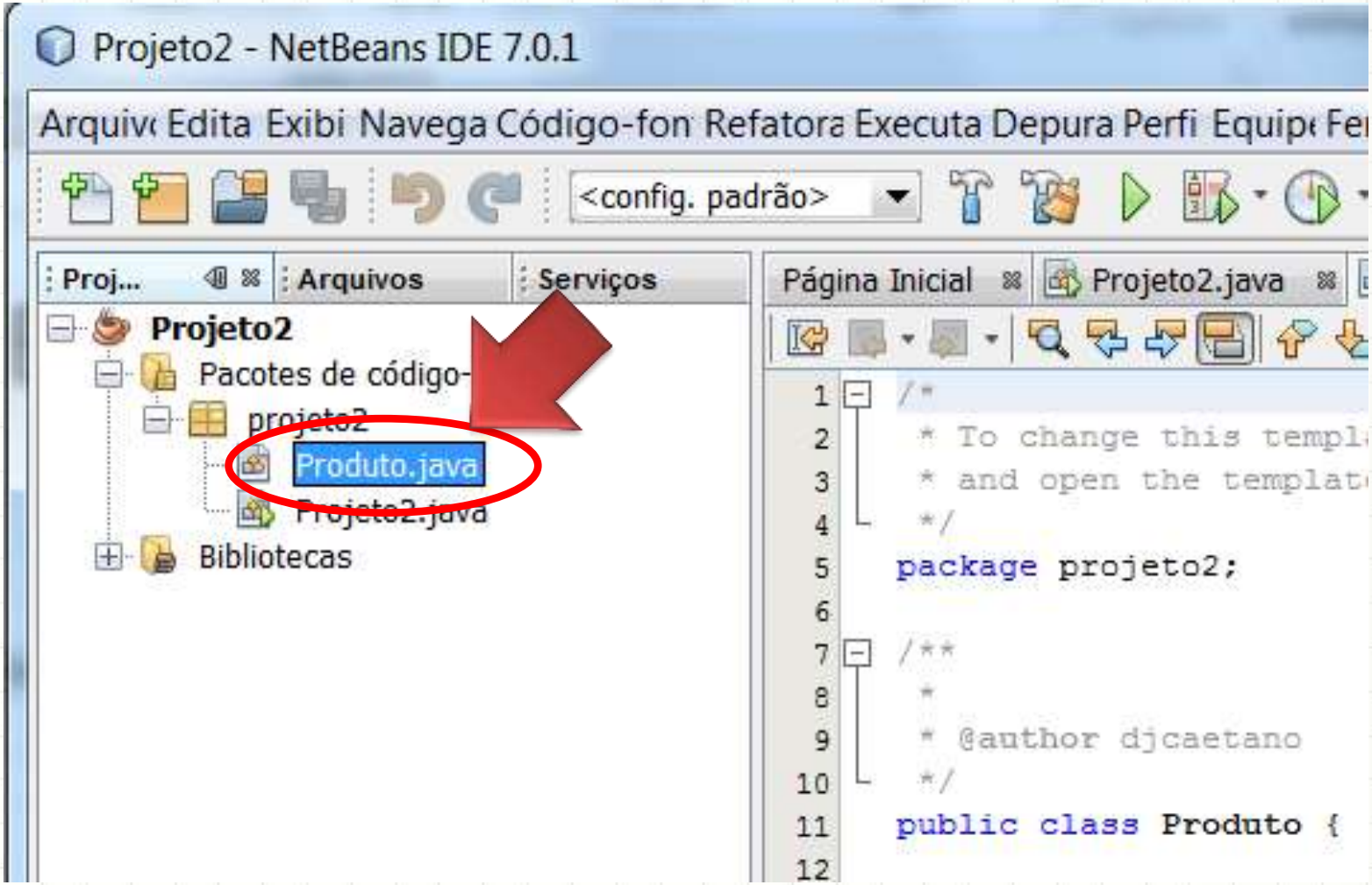
Definindo uma Classe

- E clique em finalizar



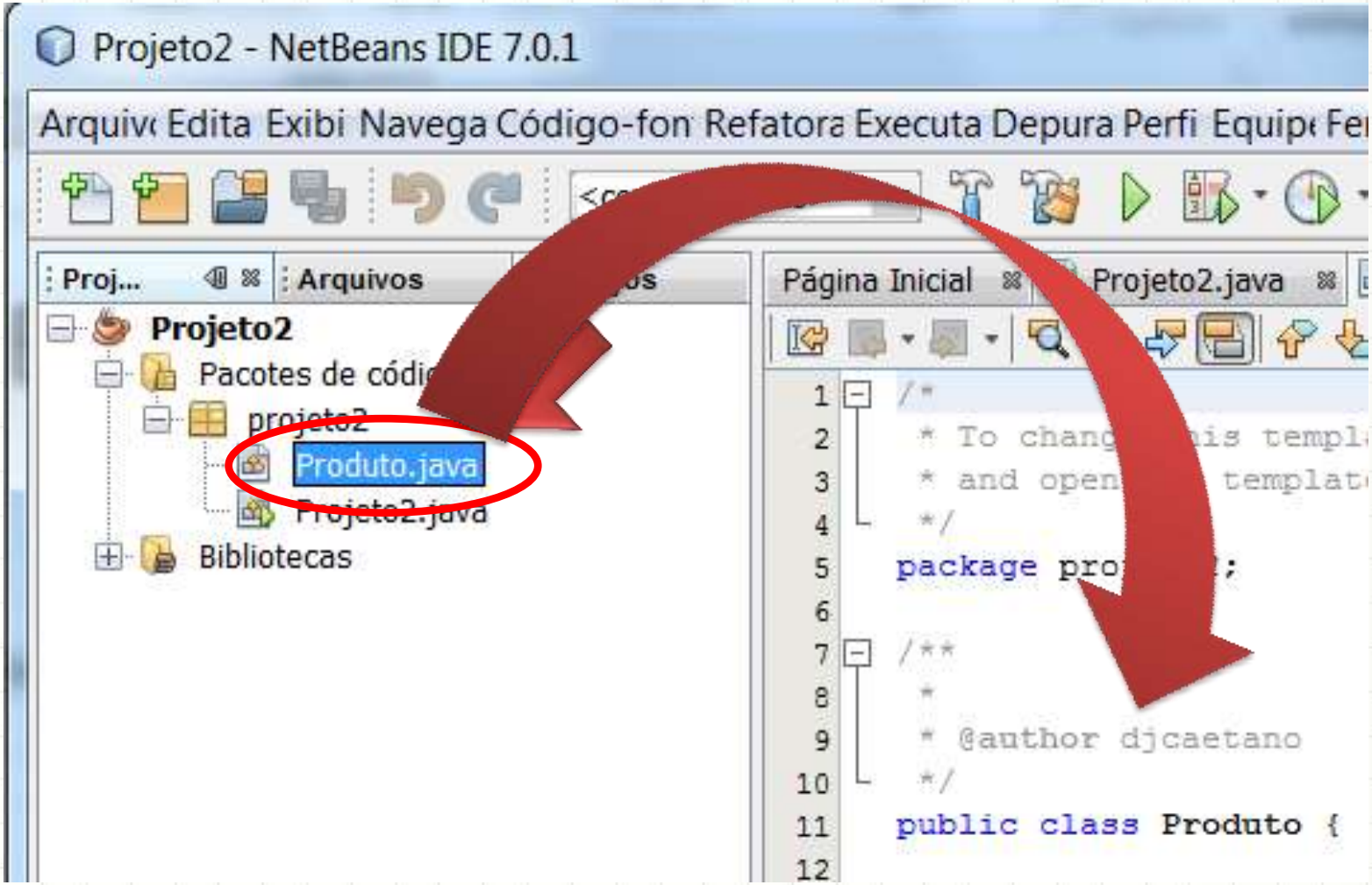
Definindo uma Classe

- Observe a classe na área de projeto



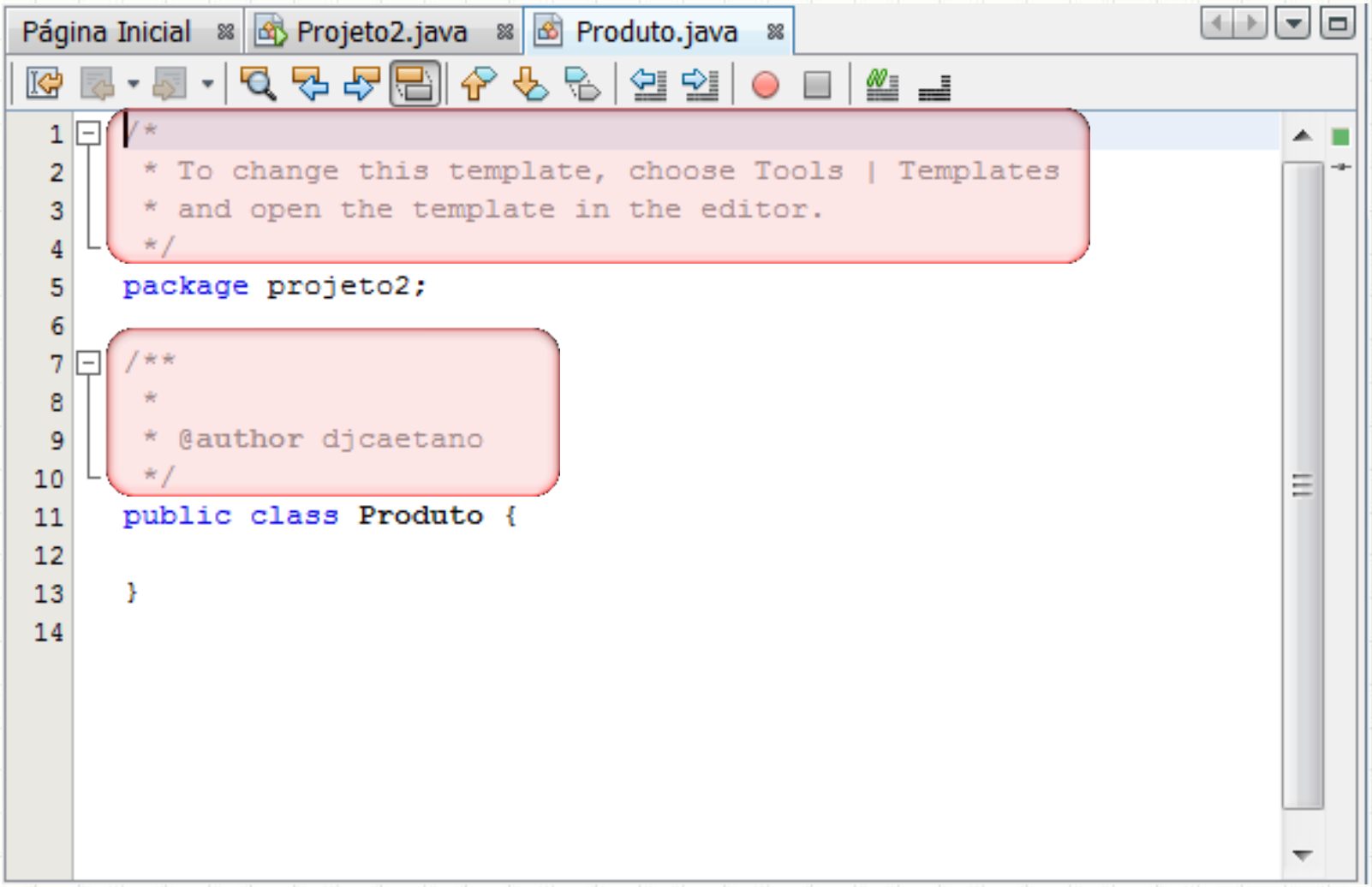
Definindo uma Classe

- Clique 2 vezes nesse ícone para ver o código



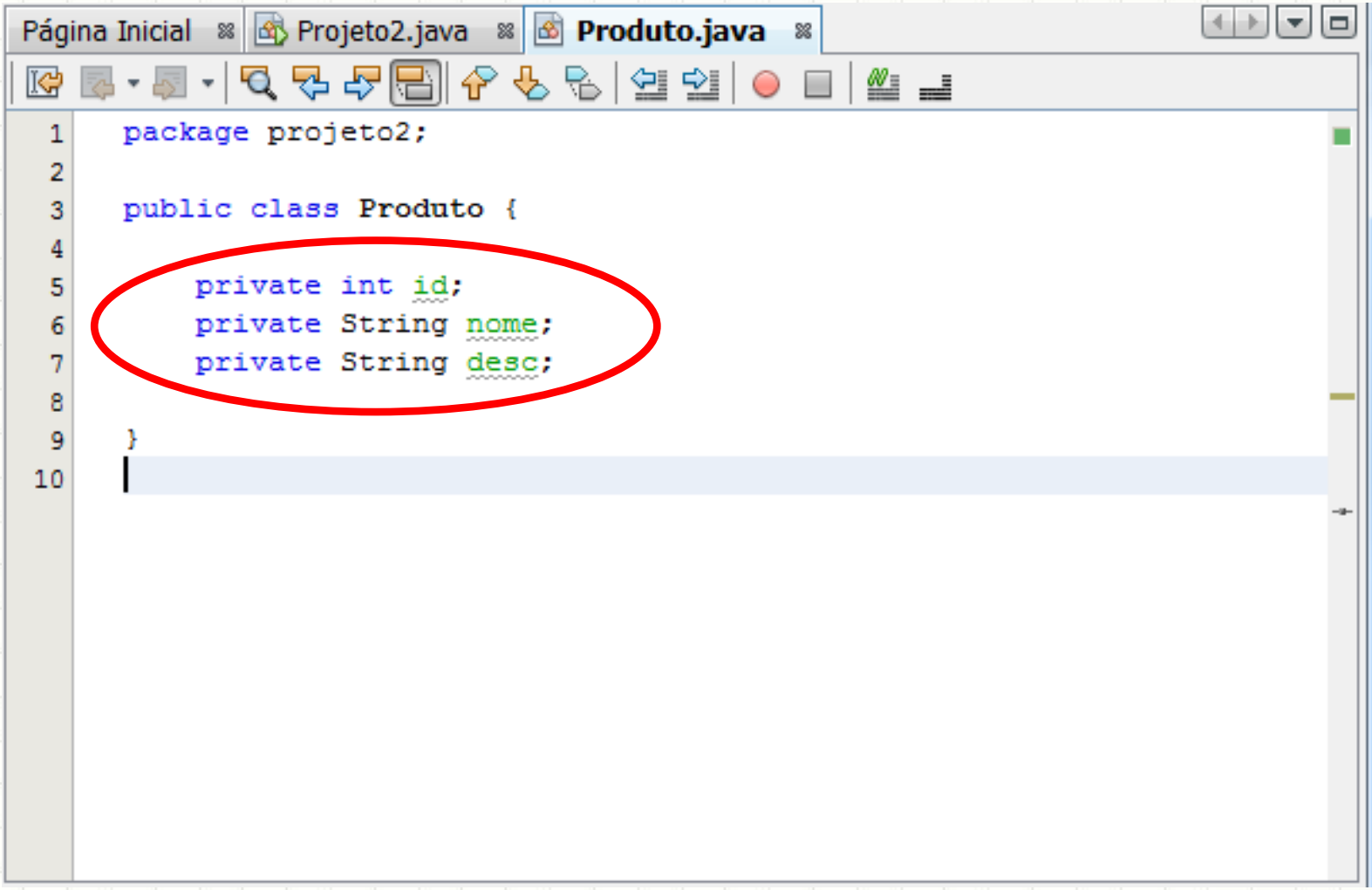
Limpando a área

- Vamos apagar os comentários do NetBeans



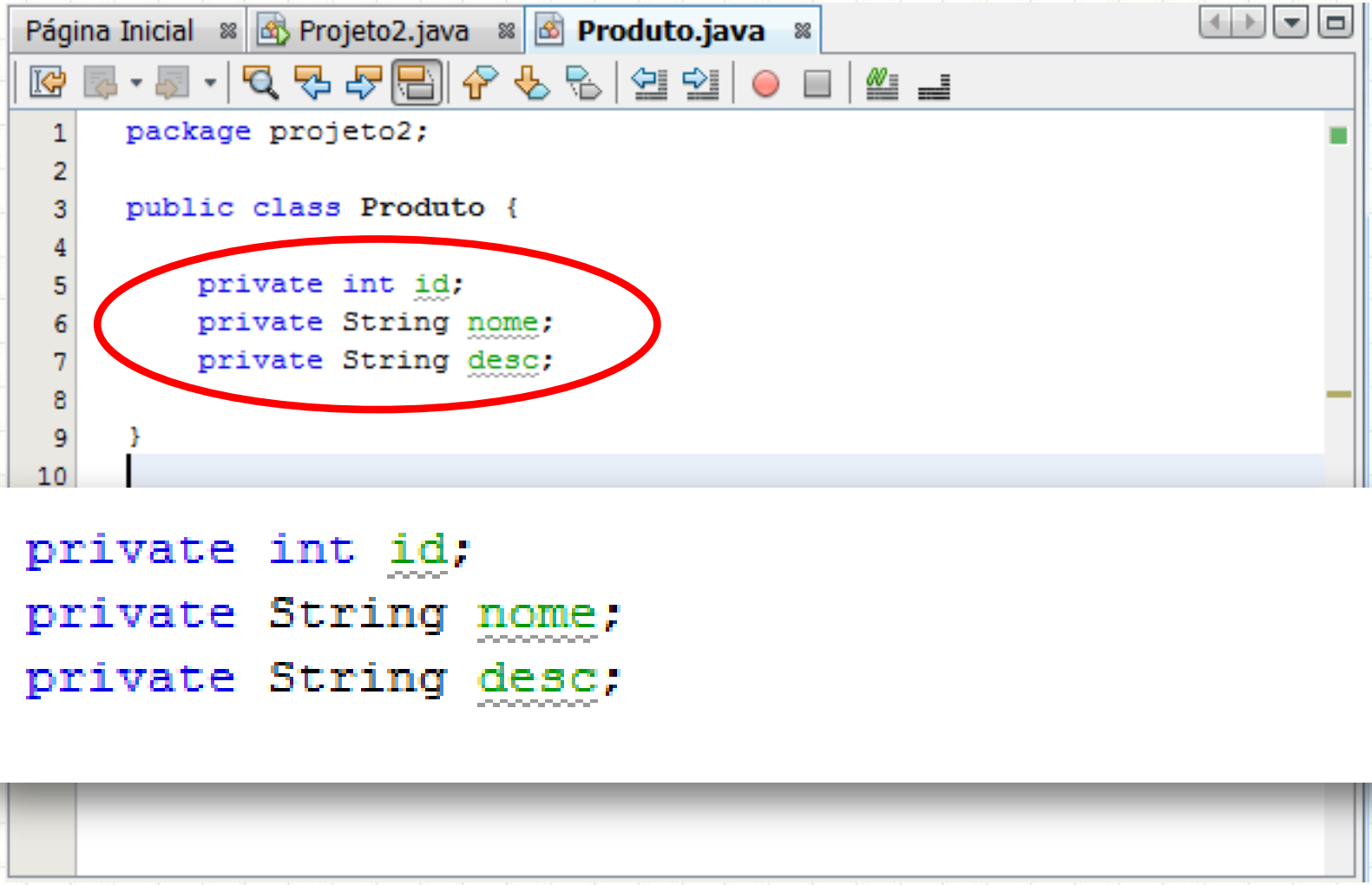
Adicionando Atributos

- Vamos adicionar **atributos** no produto



Adicionando Atributos

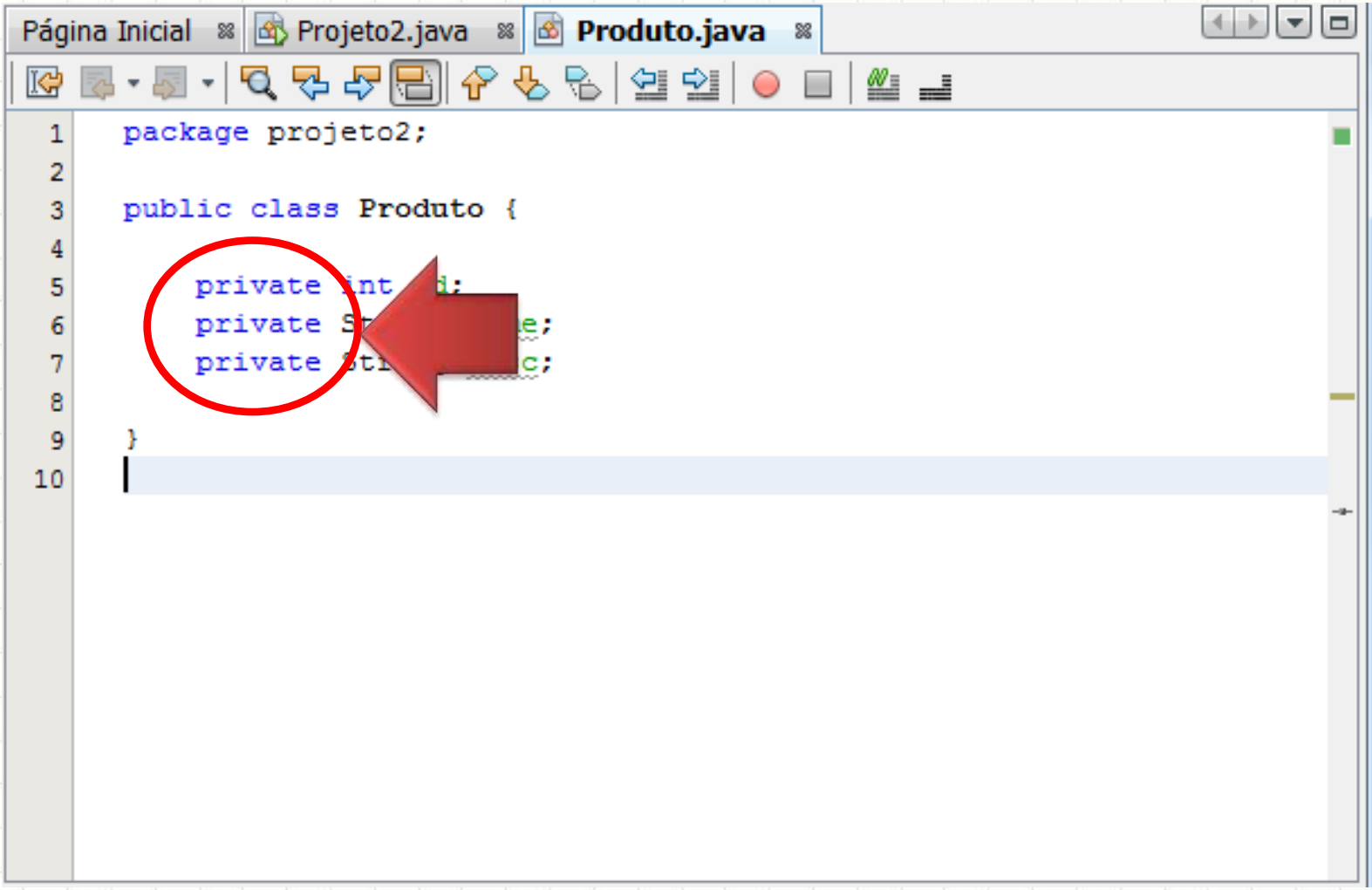
- Vamos adicionar **atributos** no produto




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

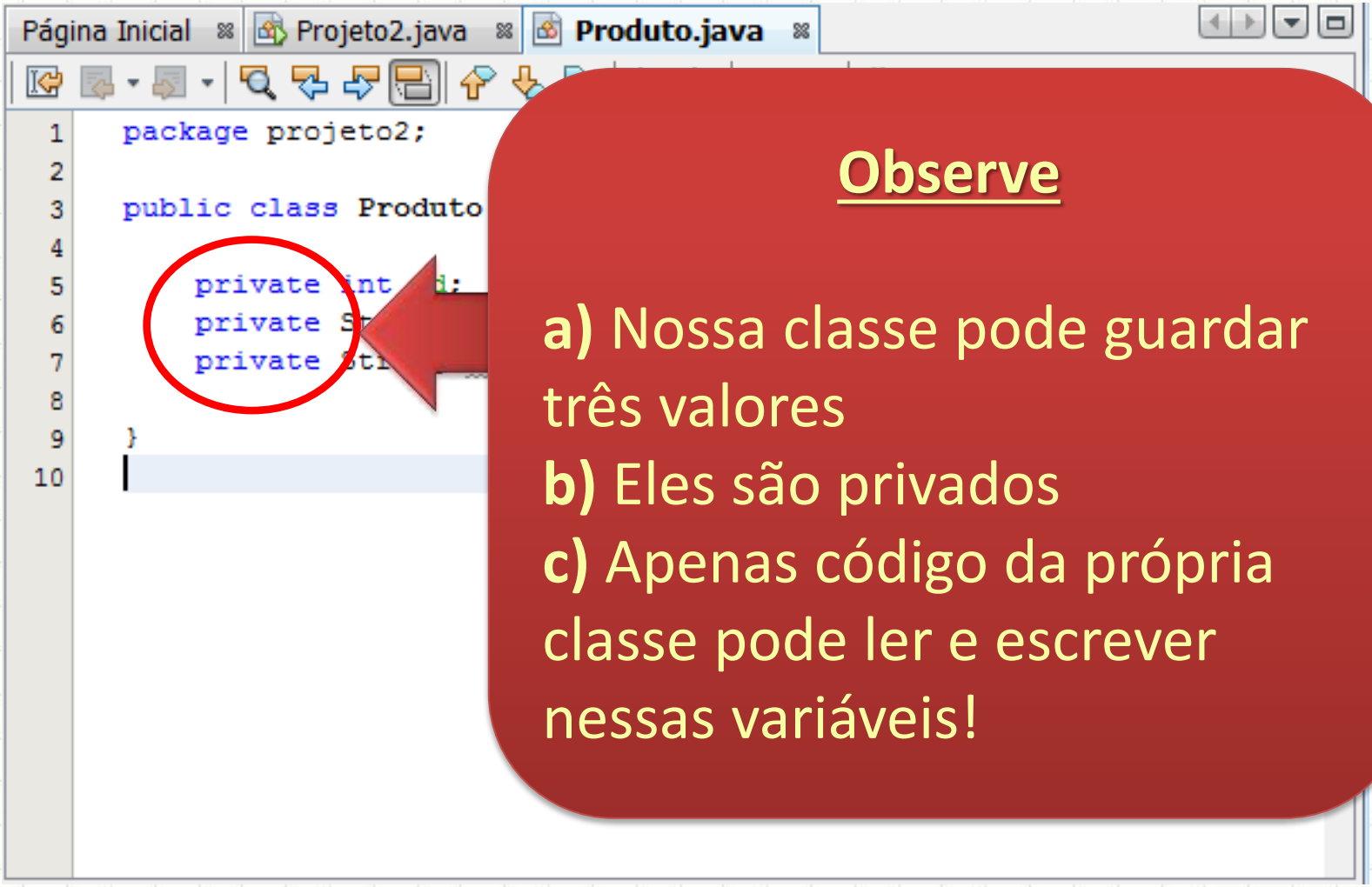
Adicionando Atributos

- Vamos adicionar **atributos** no produto



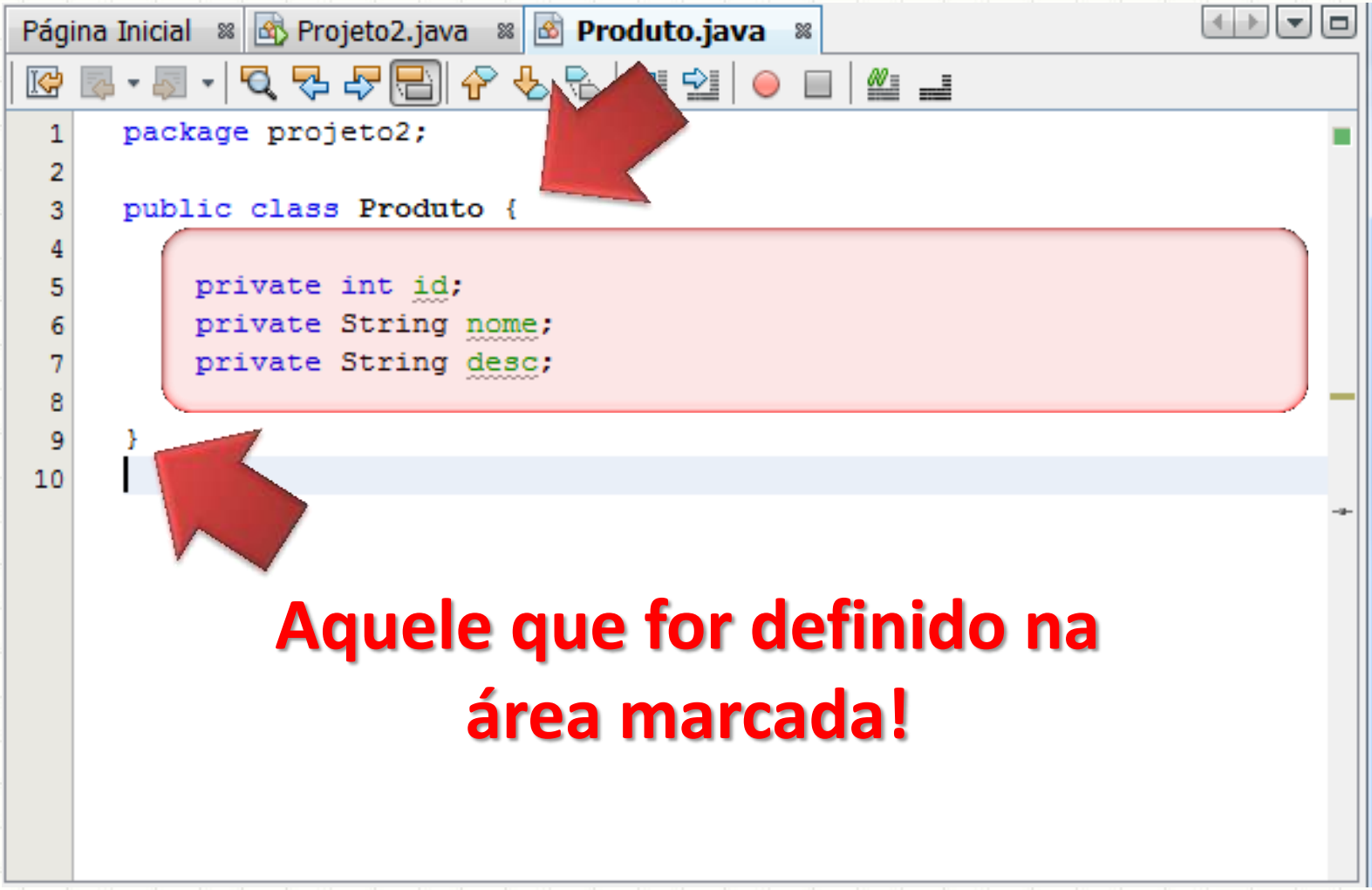
Adicionando Atributos

- Vamos adicionar **atributos** no produto



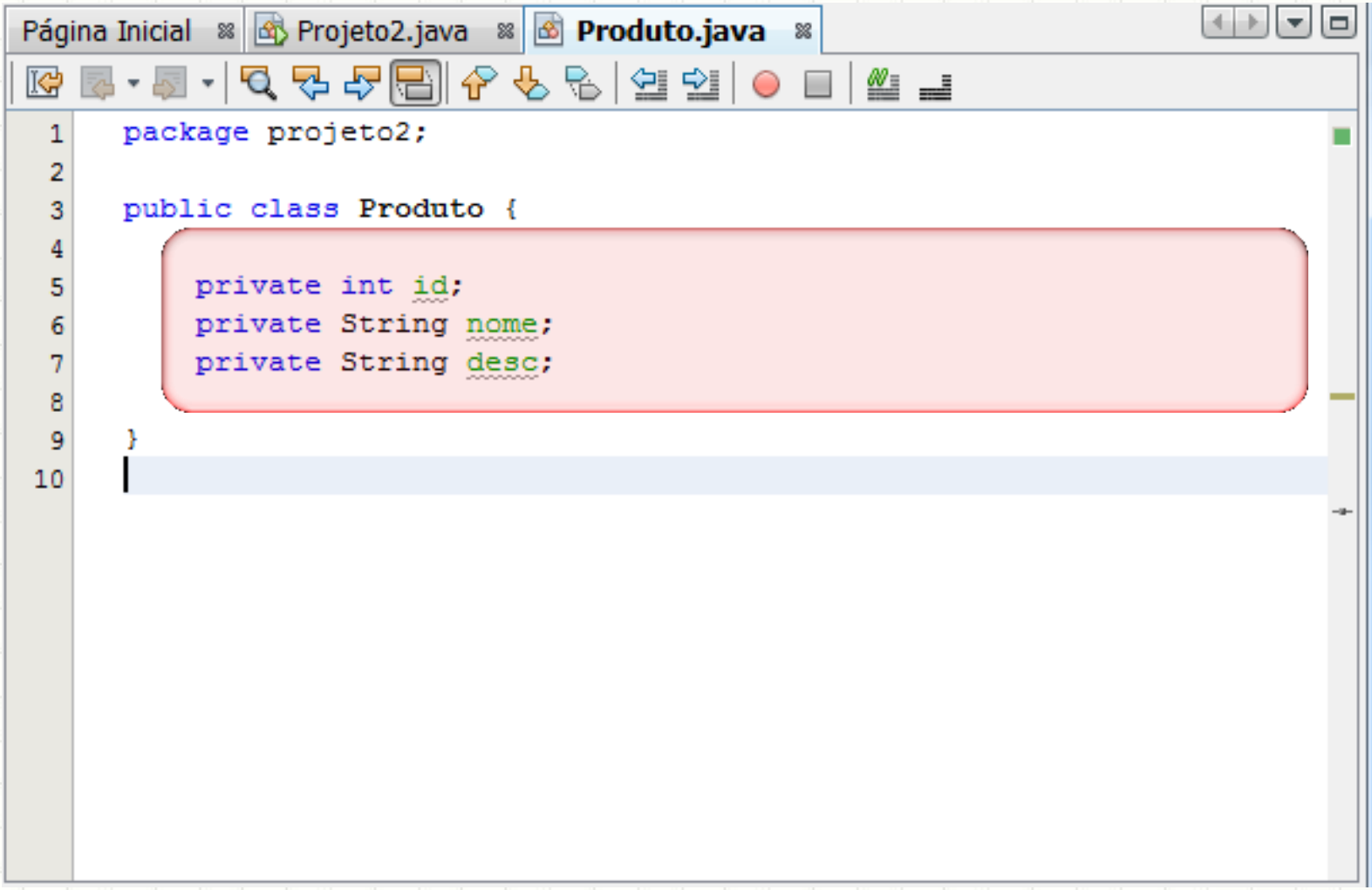
Escopo e Visibilidade

- O que é “código da própria classe”?



Setters e *Getters*

- Como modificar estes valores?



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()**.

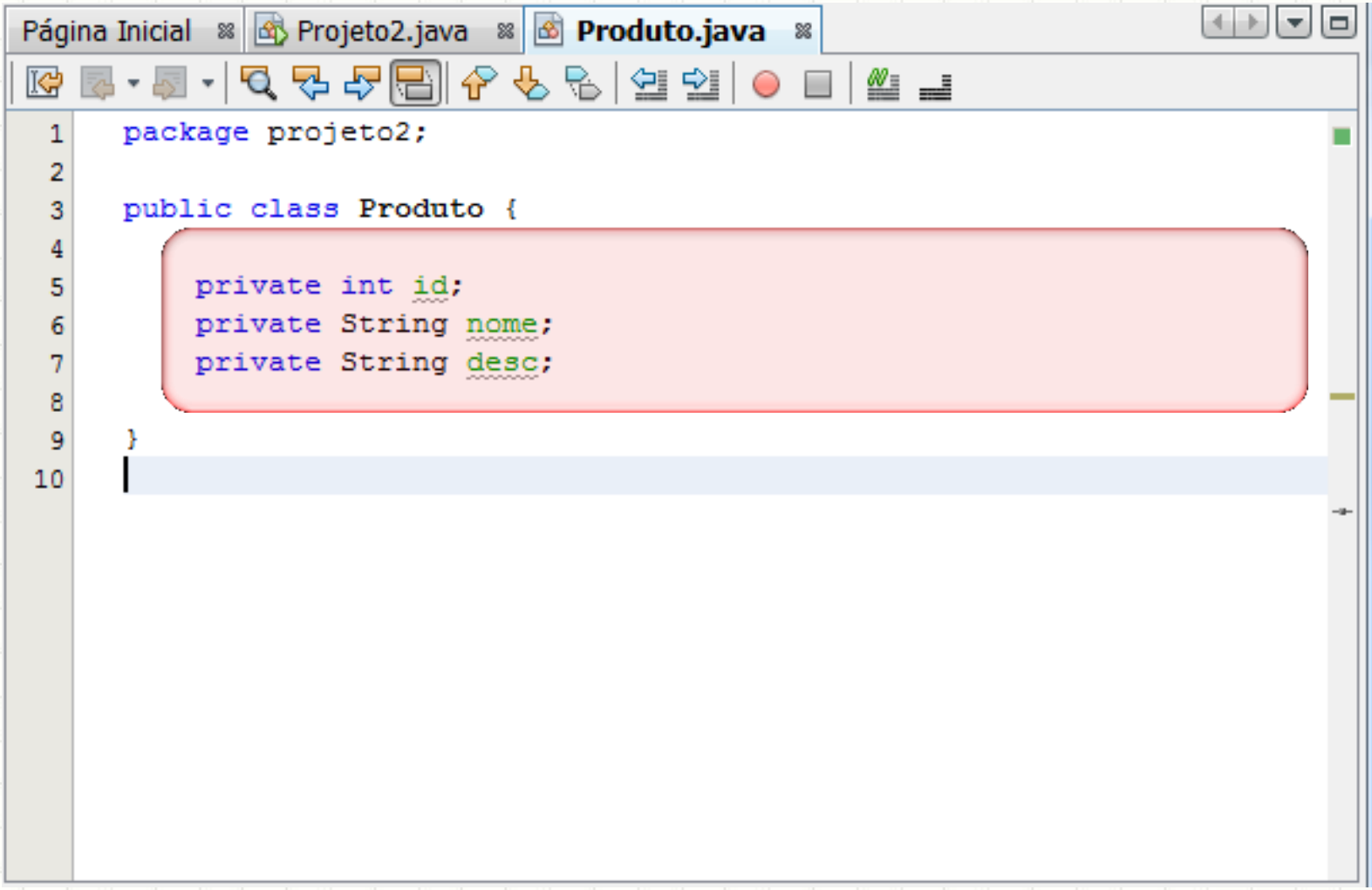
Setters e *Getters*

- 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

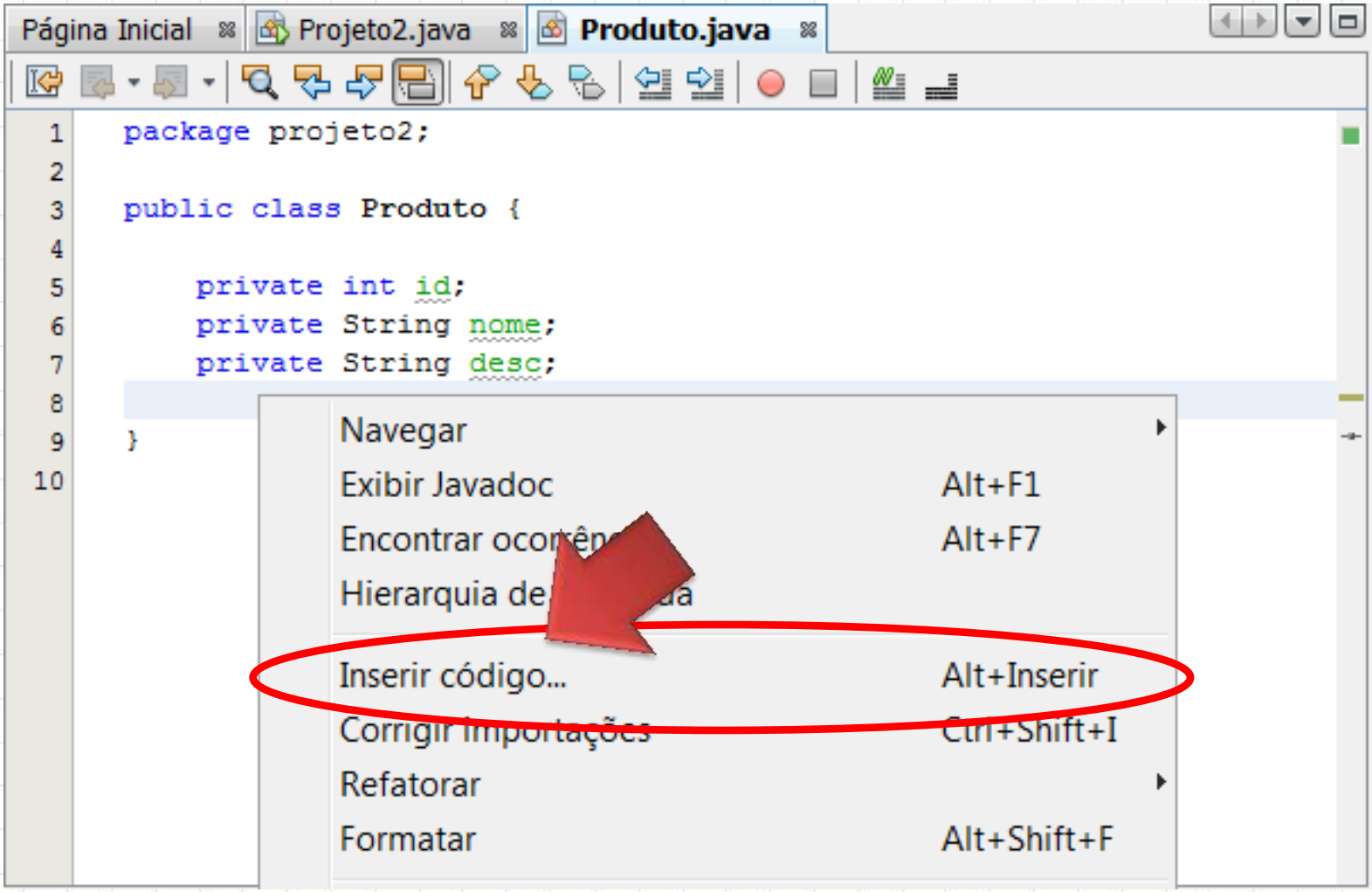
Setters e *Getters* – Criando *Setters*

- Clique com o botão direito na área da classe



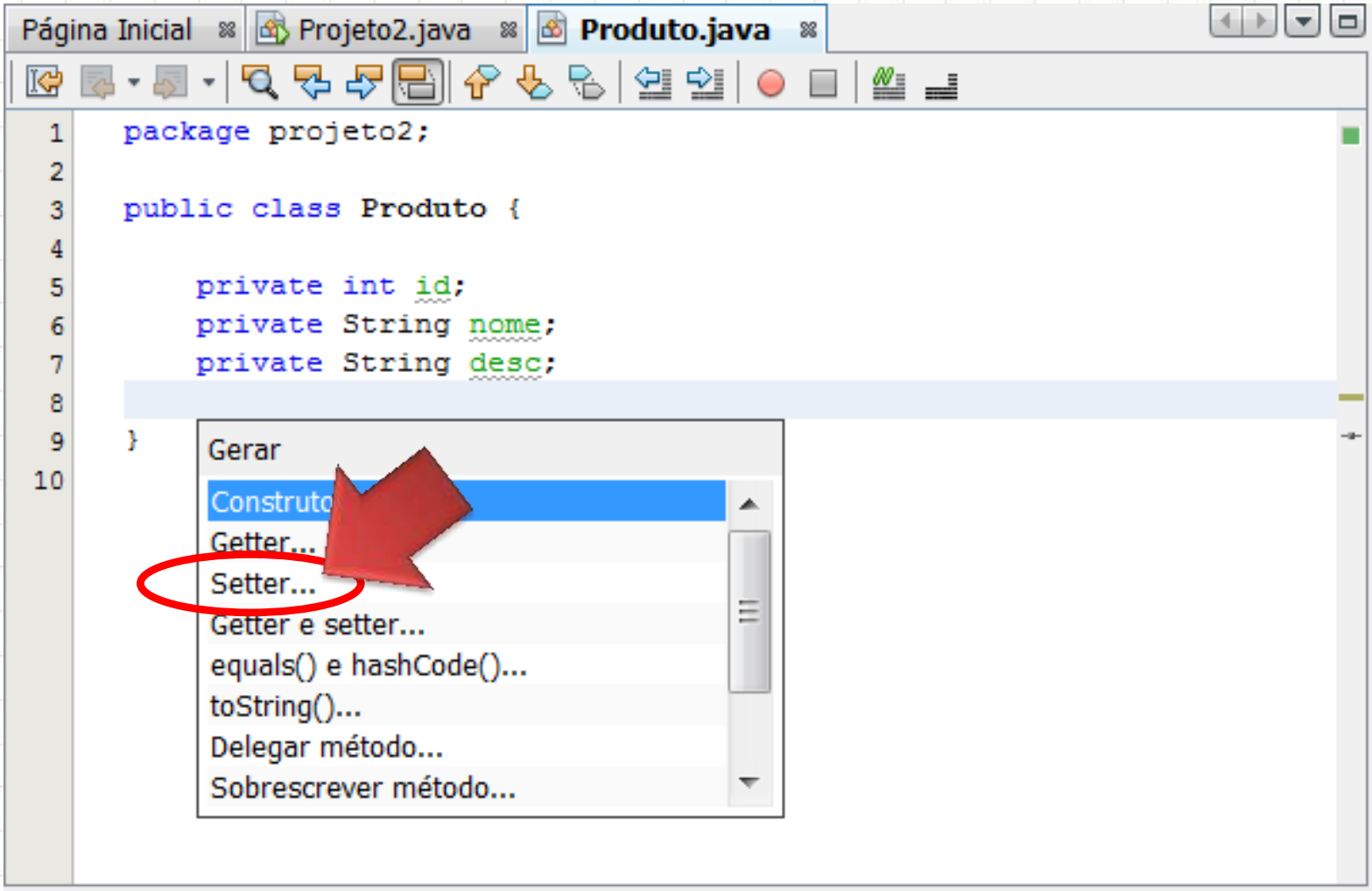
Setters e *Getters* – Criando *Setters*

- Selecione a opção **inserir código**



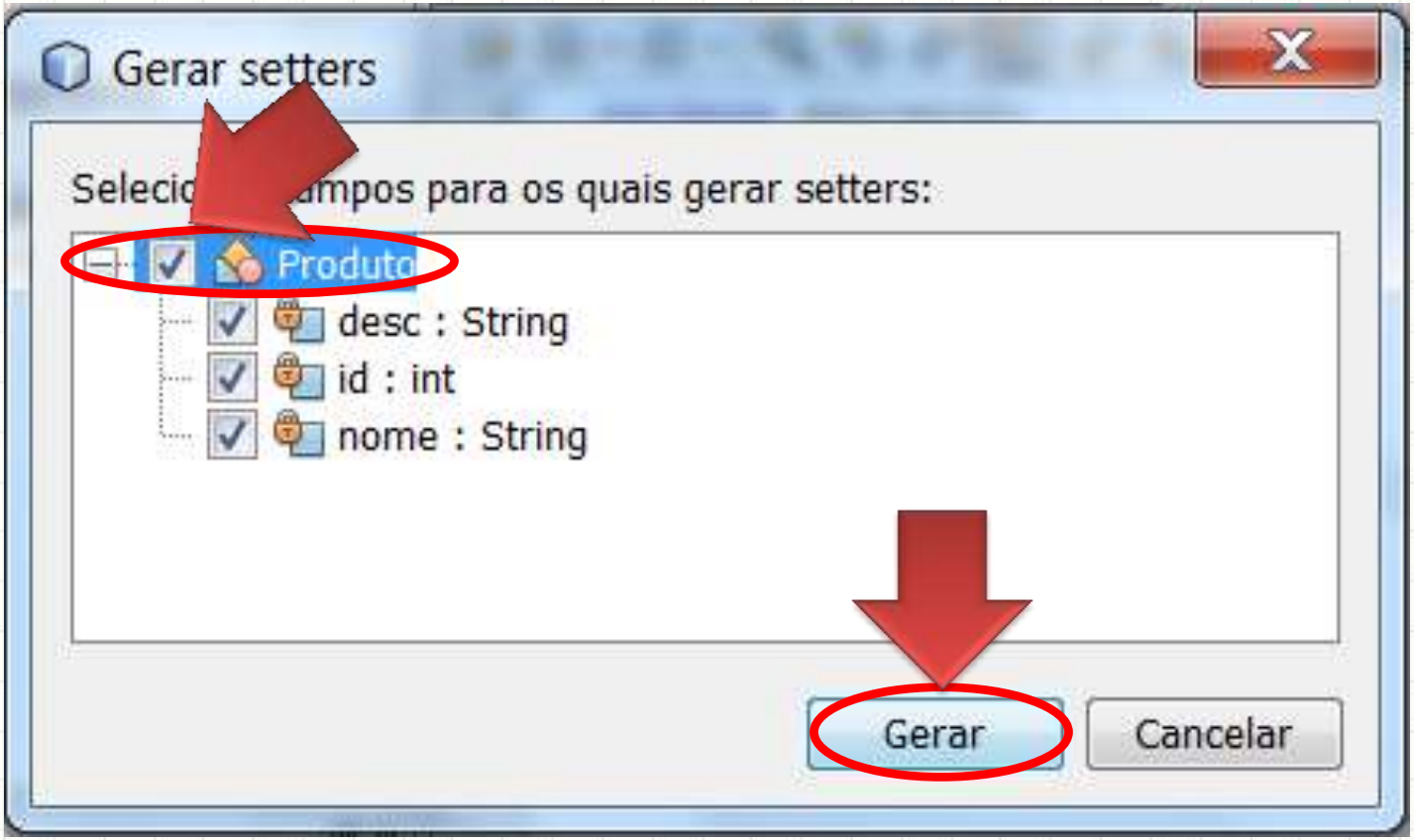
Setters e *Getters* – Criando *Setters*

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



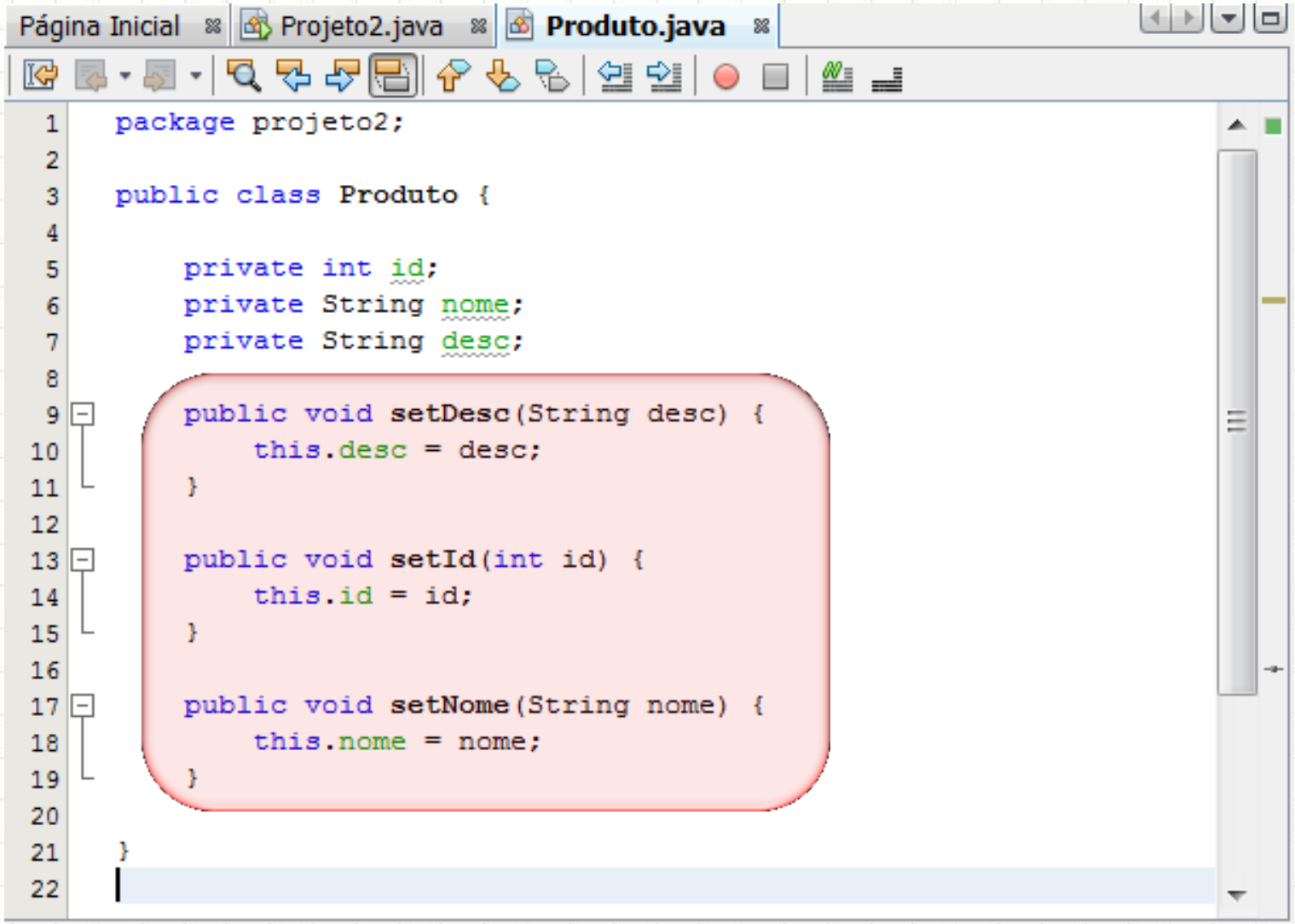
Setters e *Getters* – Criando *Setters*

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



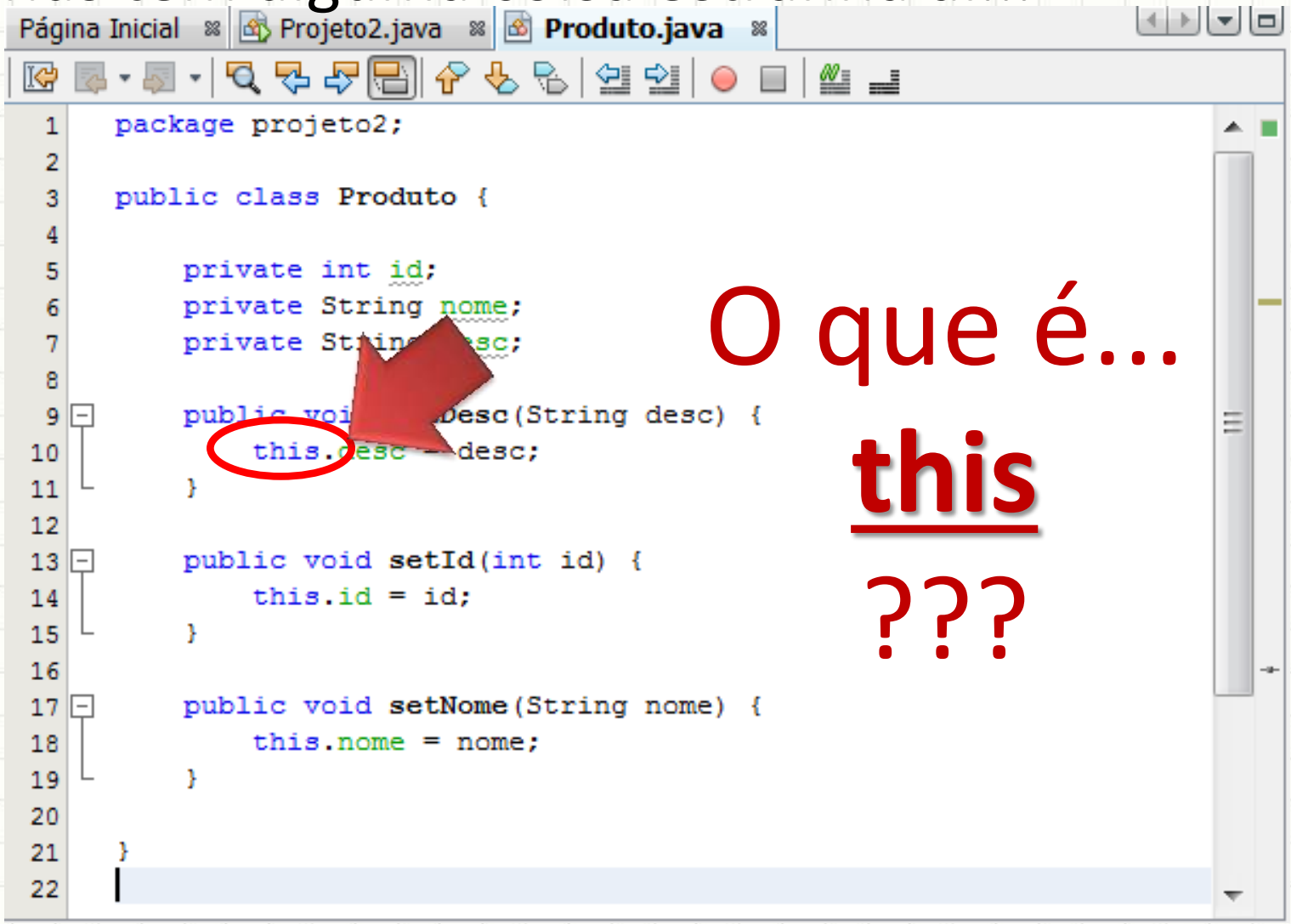
Setters e *Getters* – Criando *Setters*

- Pronto! O NetBeans criou tudo sozinho!



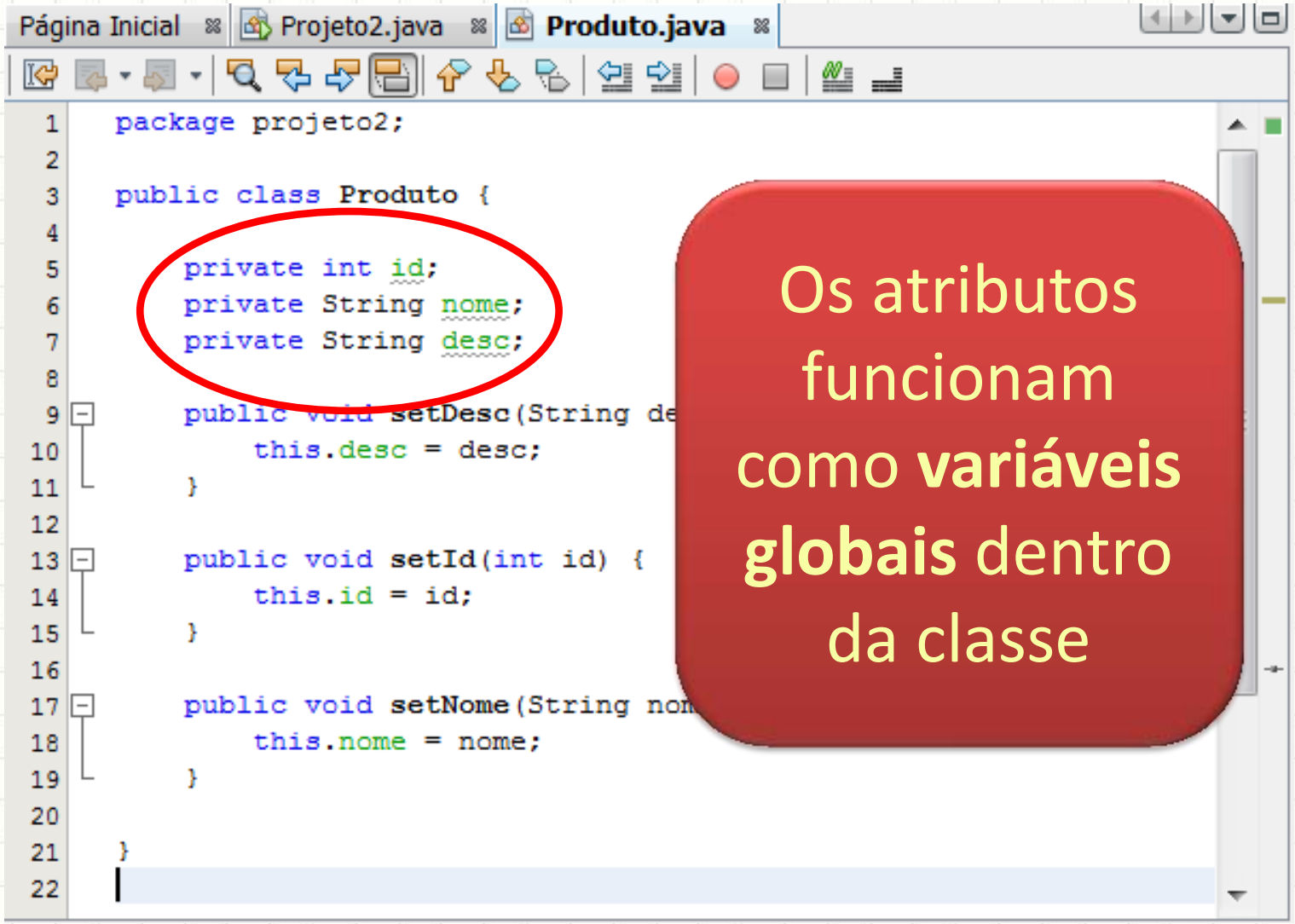
Setters e *Getters* – Criando *Setters*

- Mas tem alguma coisa estranha aí...



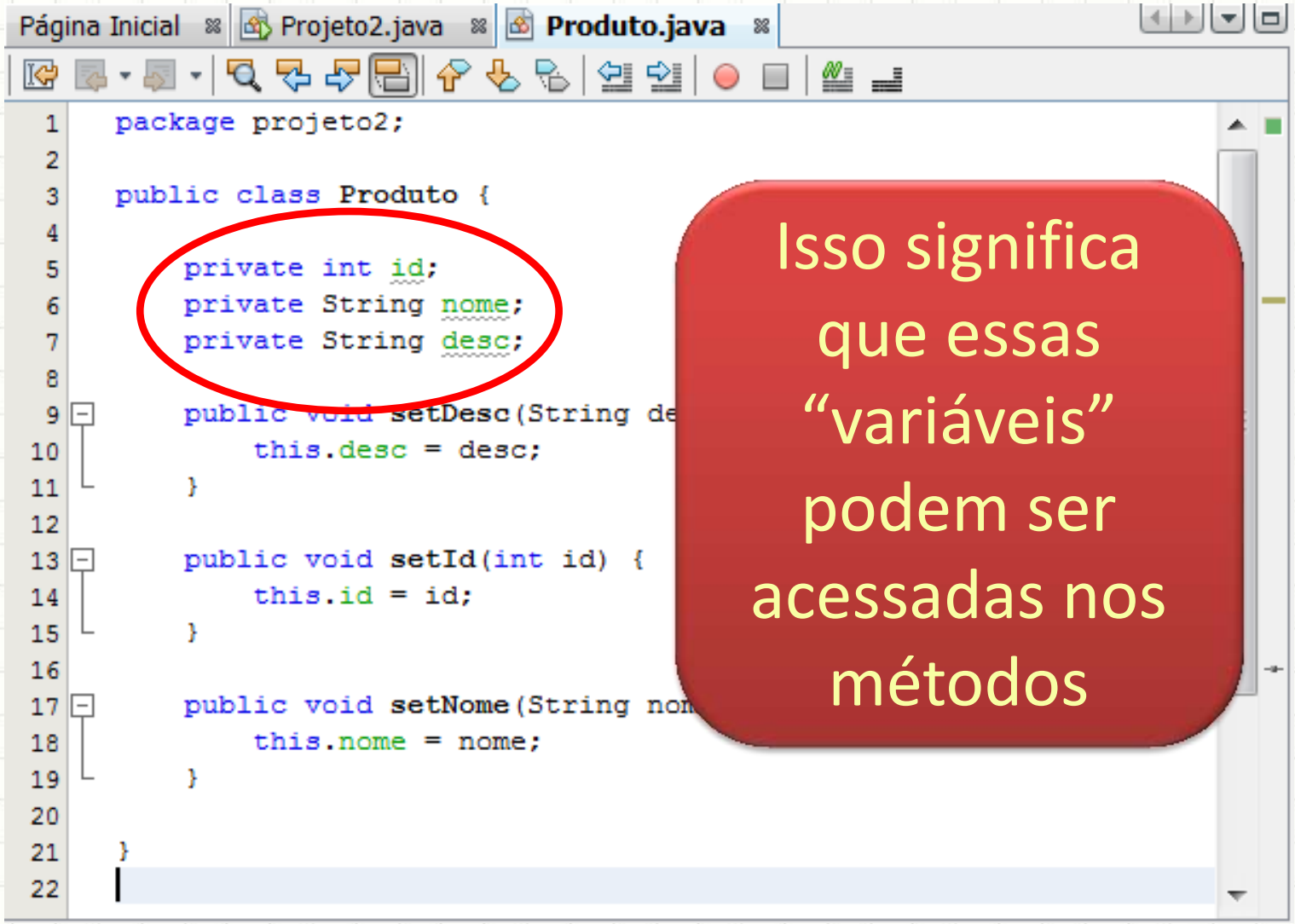
Setters e *Getters* – Criando *Setters*

- Primeiro vamos entender uma coisa...



Setters e *Getters* – Criando *Setters*

- Primeiro vamos entender uma coisa...



Setters e *Gett*

- Primeiro vamos er

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
```

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

Setters e *Gett*

- Primeiro vamos er

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 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.

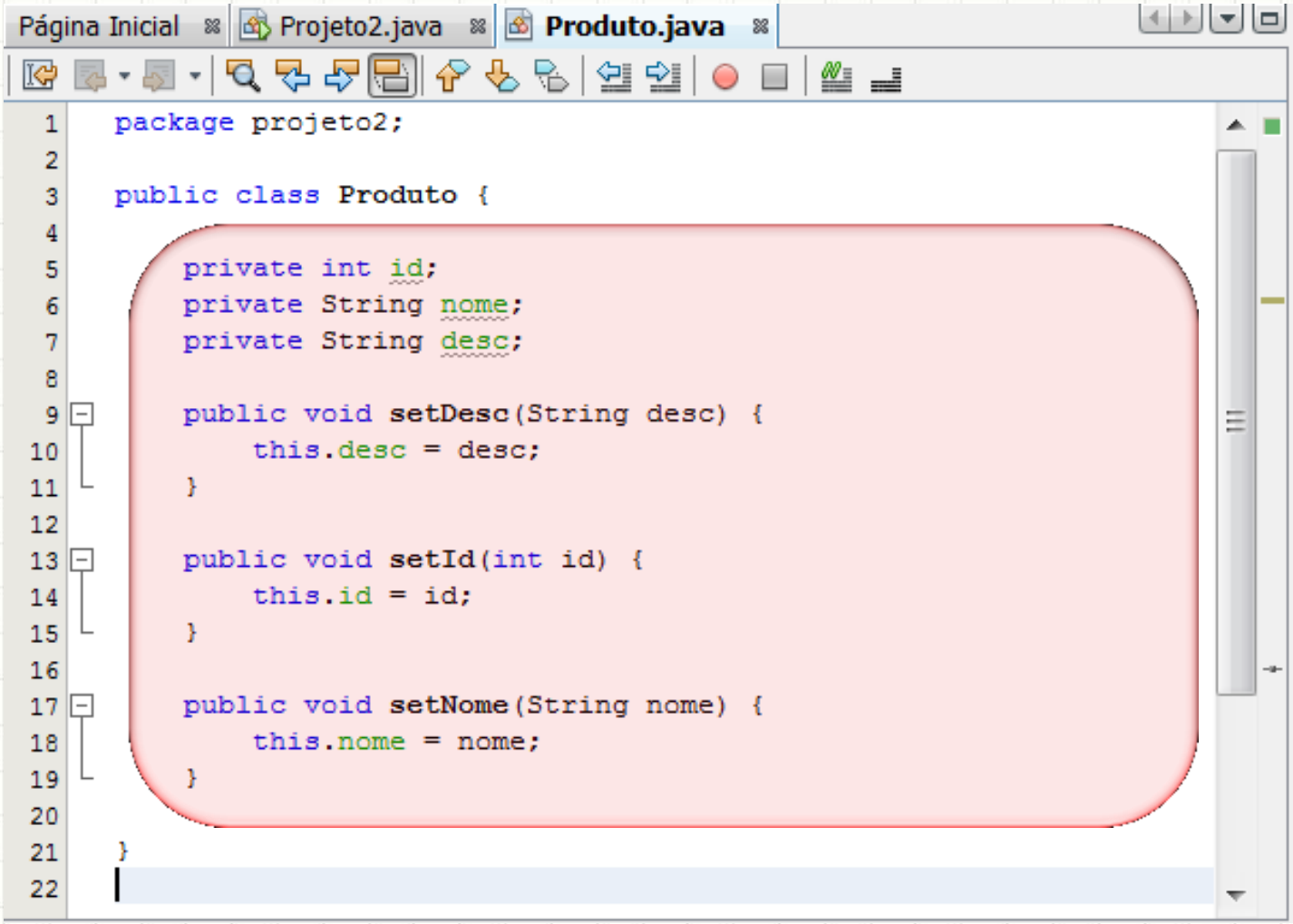
```
4
5       private int id;
6       private String nome;
7       private String desc;
8
9       public void setDesc(String desc) {
10          this.desc = desc;
11      }
12
13      public void setId(int id) {
14          this.id = id;
15      }
16
17      public void setNome(String nome) {
18    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

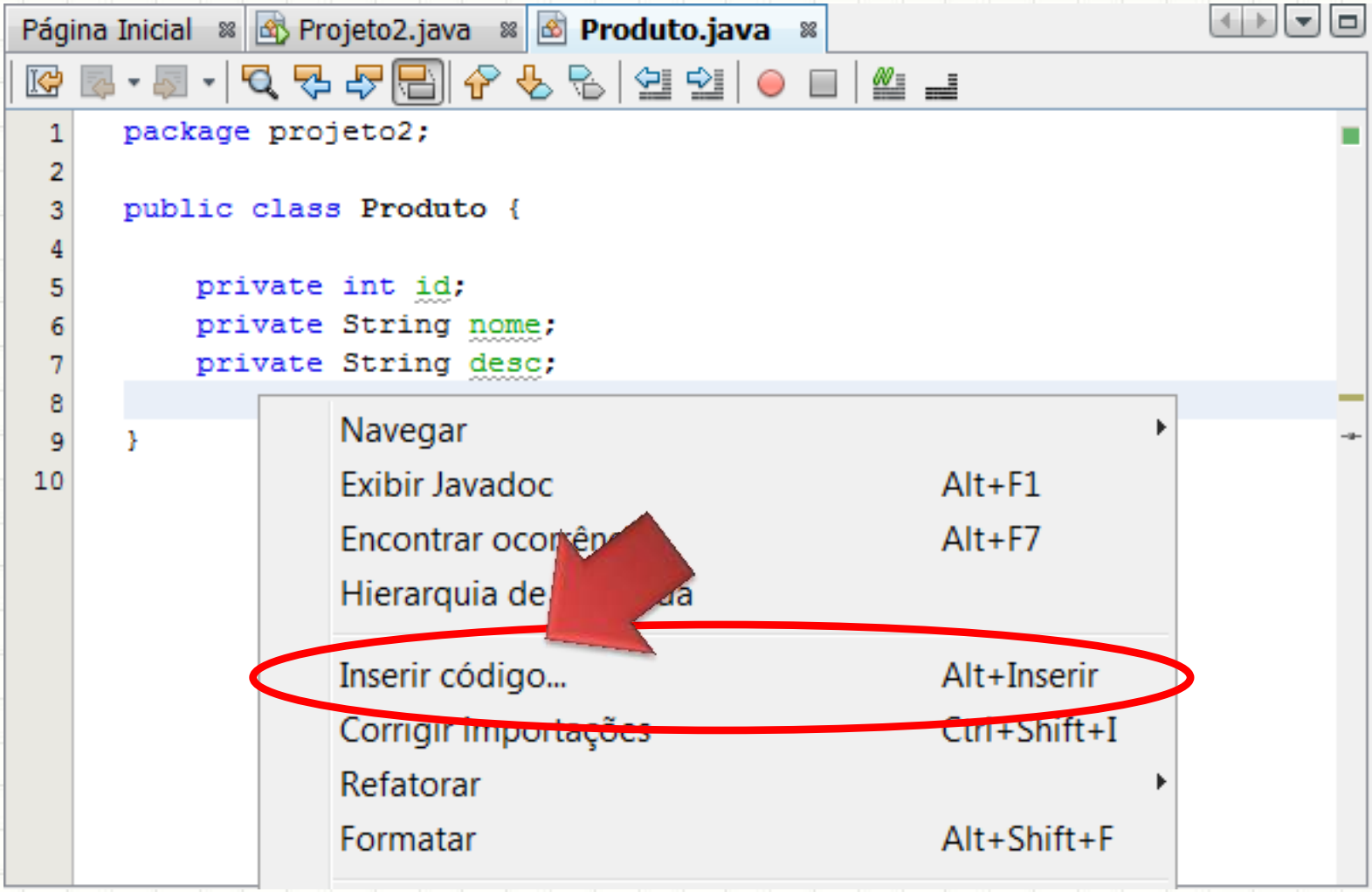
Setters e *Getters* – Criando *Getters*

- Clique com o botão direito na área da classe



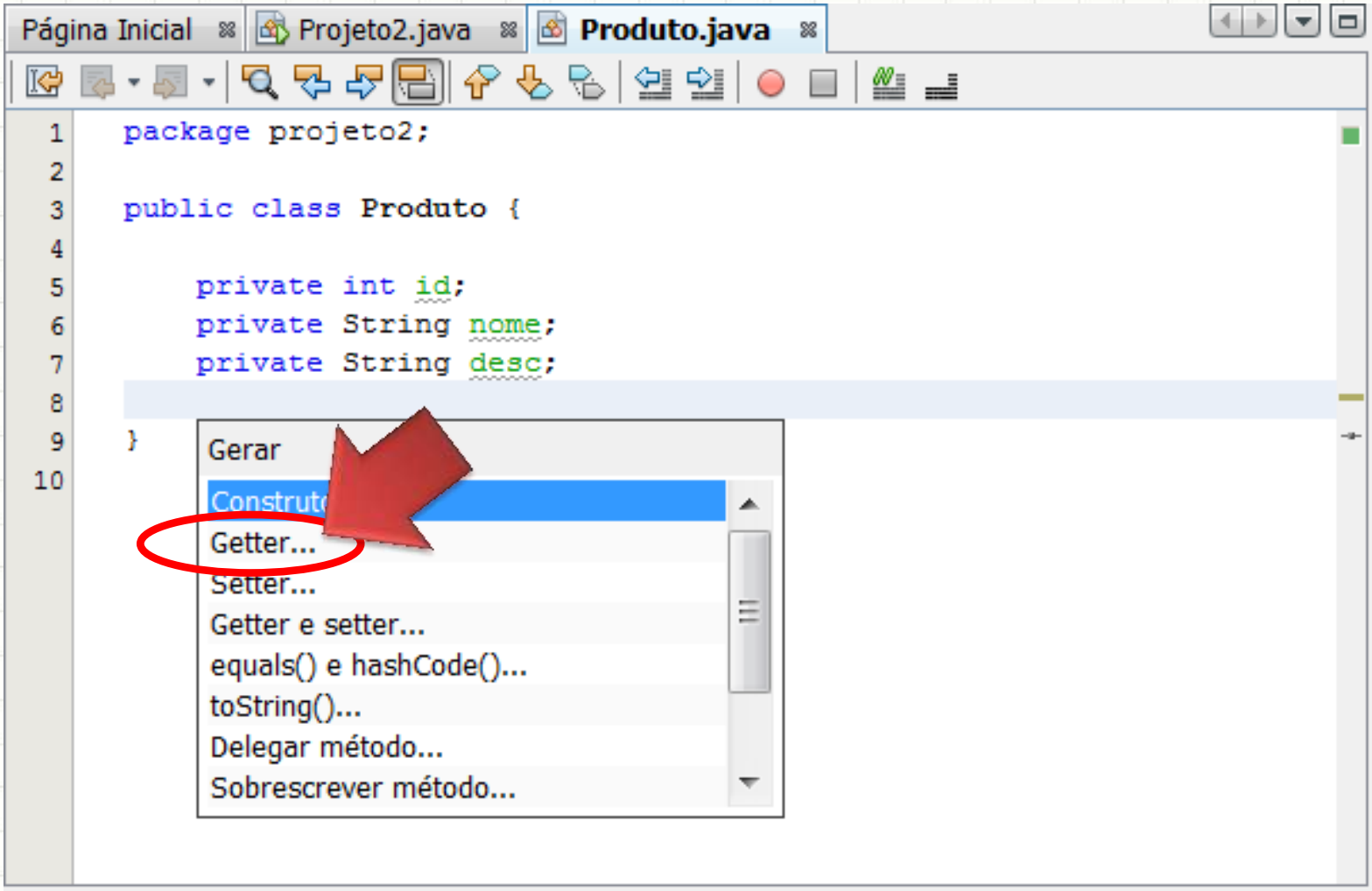
Setters e *Getters* – Criando *Getters*

- Selecione a opção **inserir código**



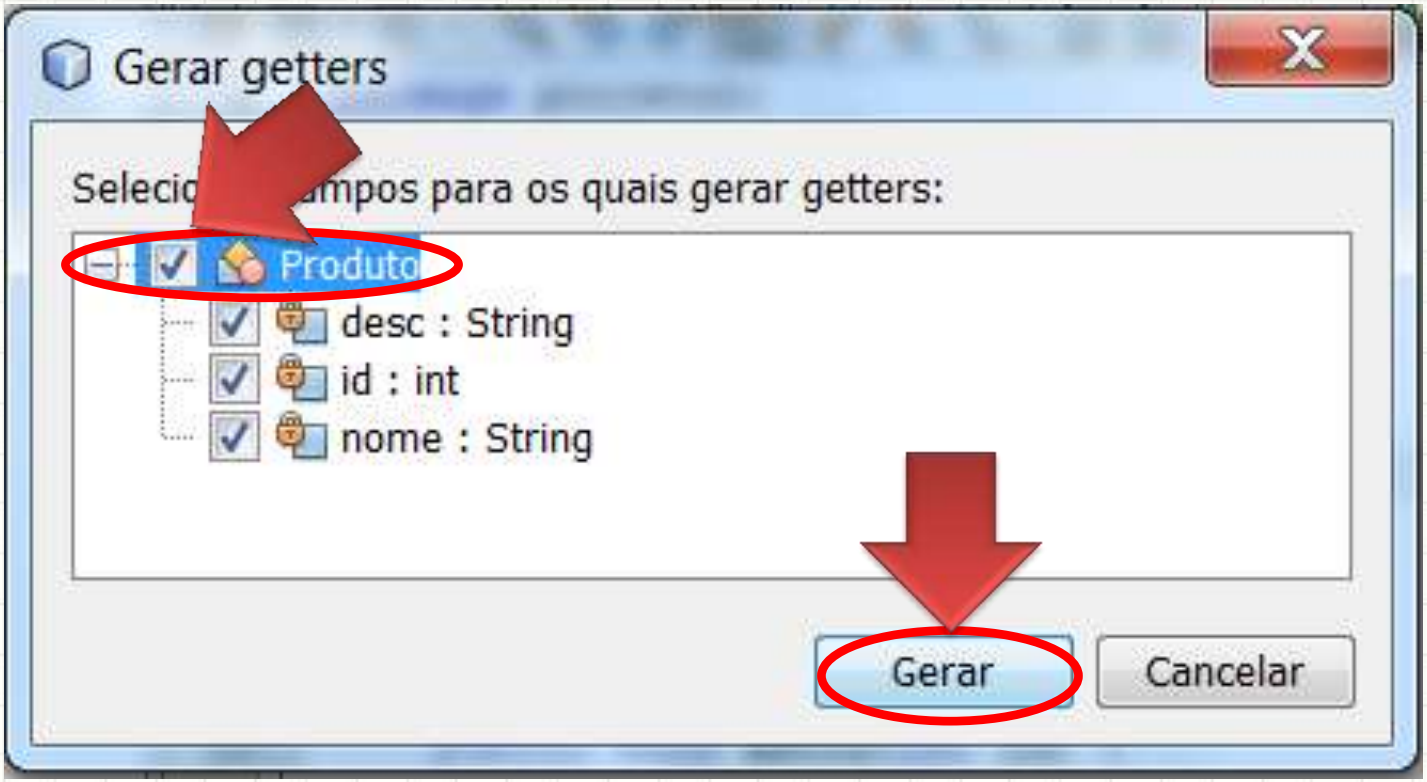
Setters e *Getters* – Criando *Getters*

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



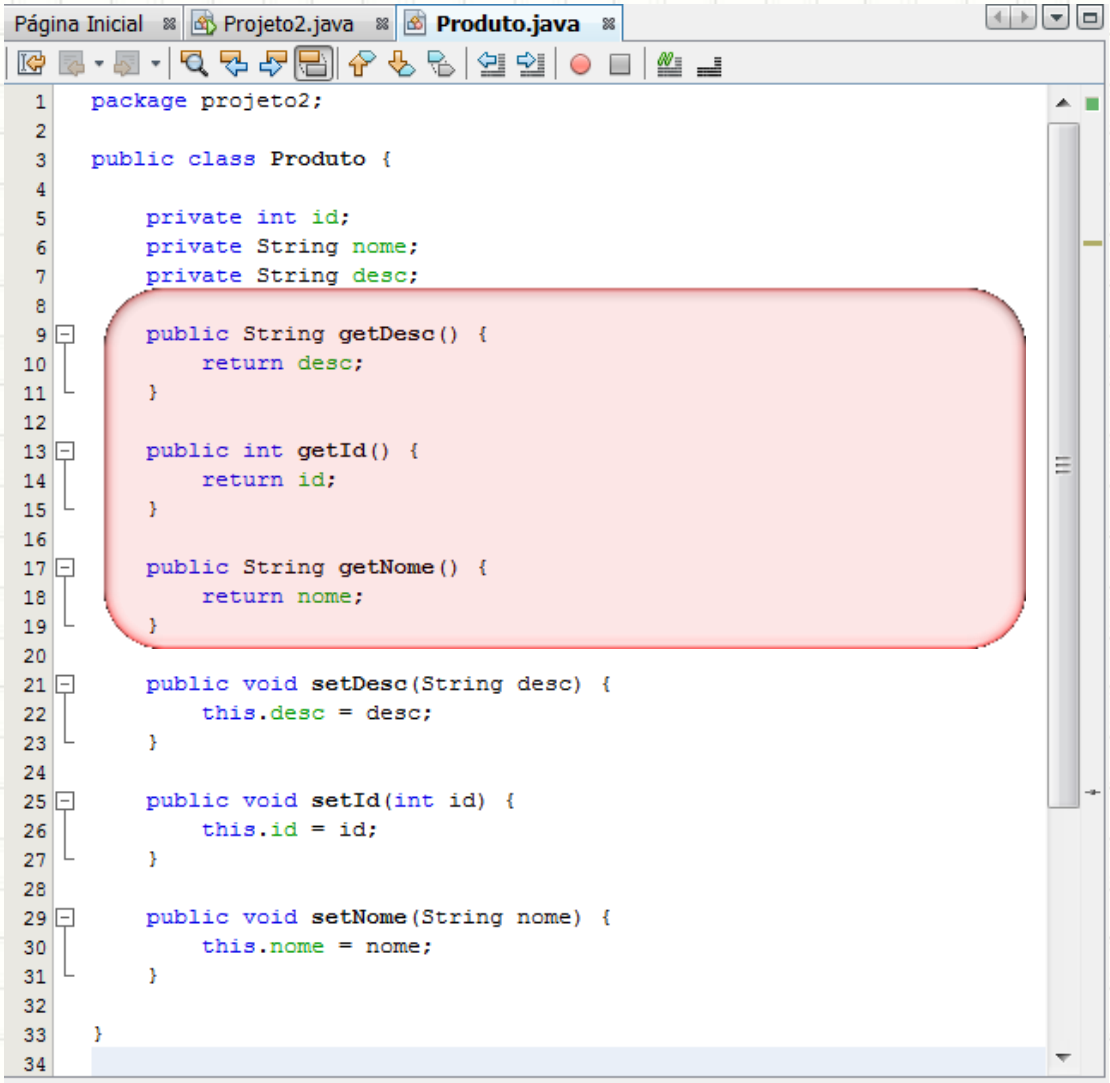
Setters e *Getters* – Criando *Getters*

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



Setters e *Getters* – Criando *Getters*

- Pronto! O NetBeans criou tudo sozinho!



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. [illegible] ido por [illegible]ariável
  3. [illegible]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

Encapsulamento

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

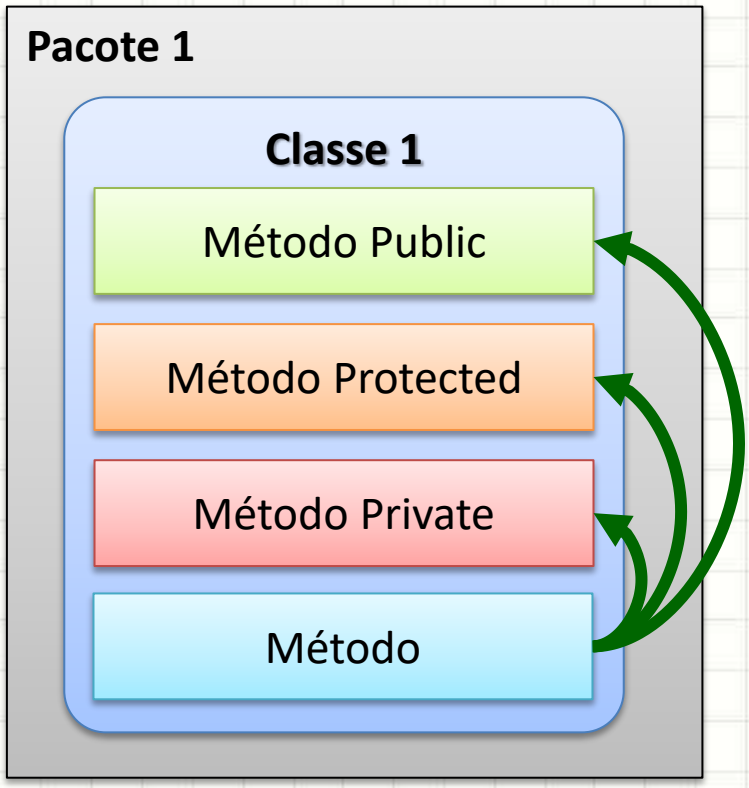
Especificadores de Acesso

- 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!

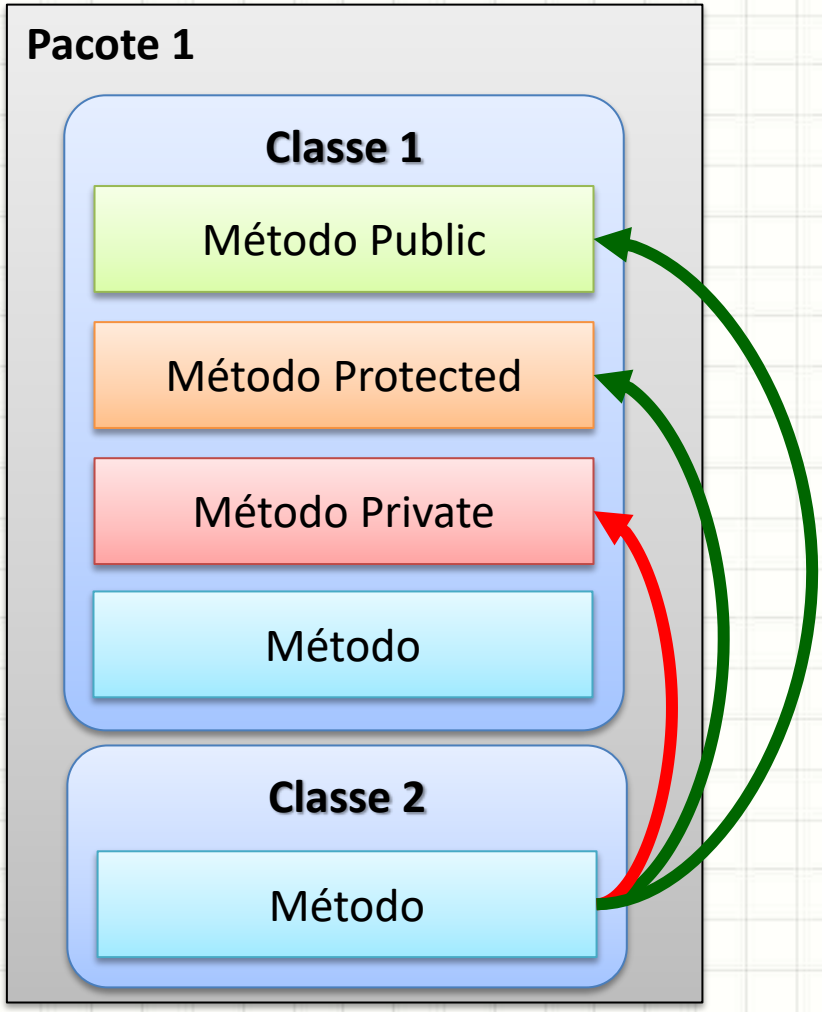
Especificadores de Acesso

- Compreenda os acessos



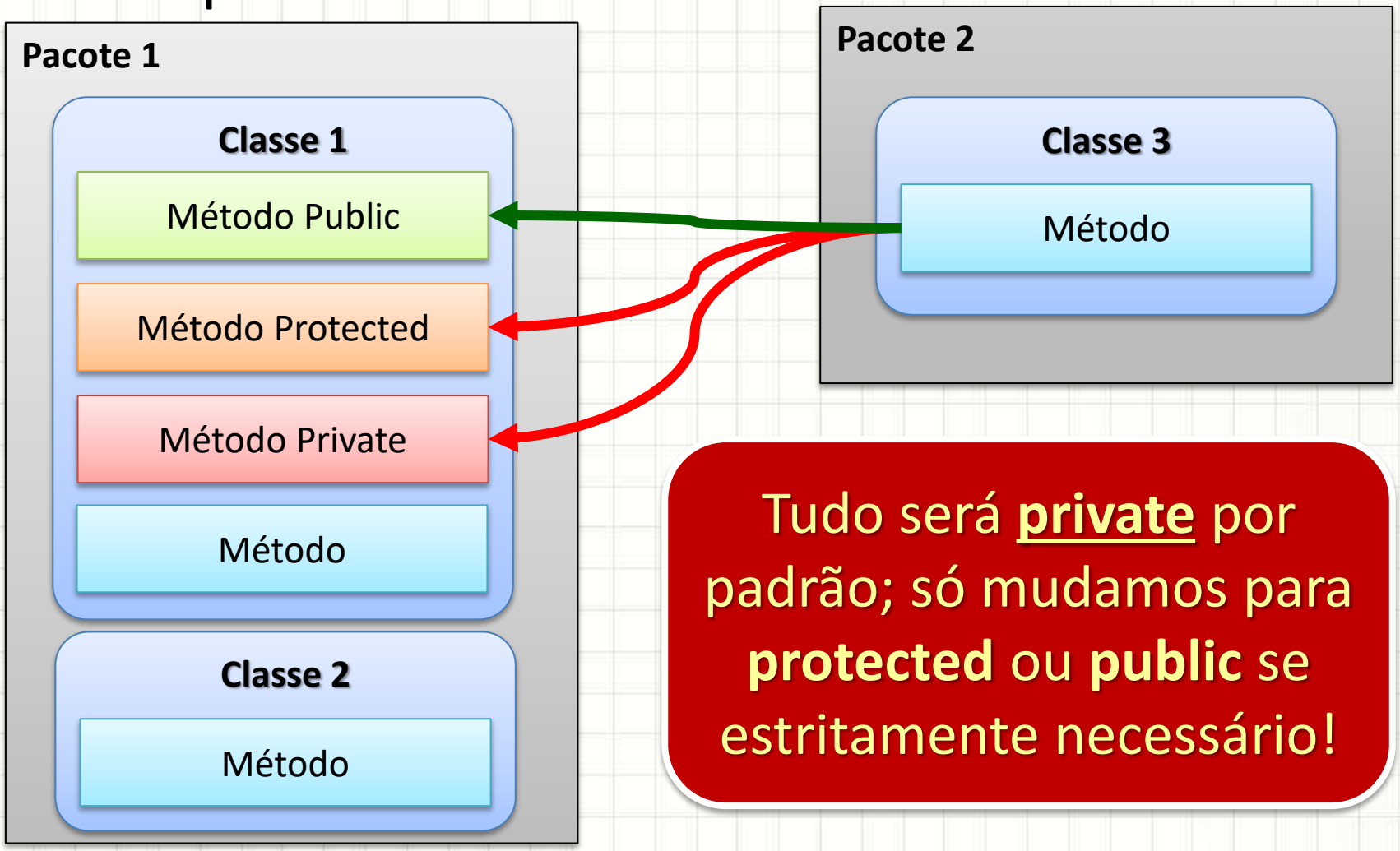
Especificadores de Acesso

- Compreenda os acessos



Especificadores de Acesso

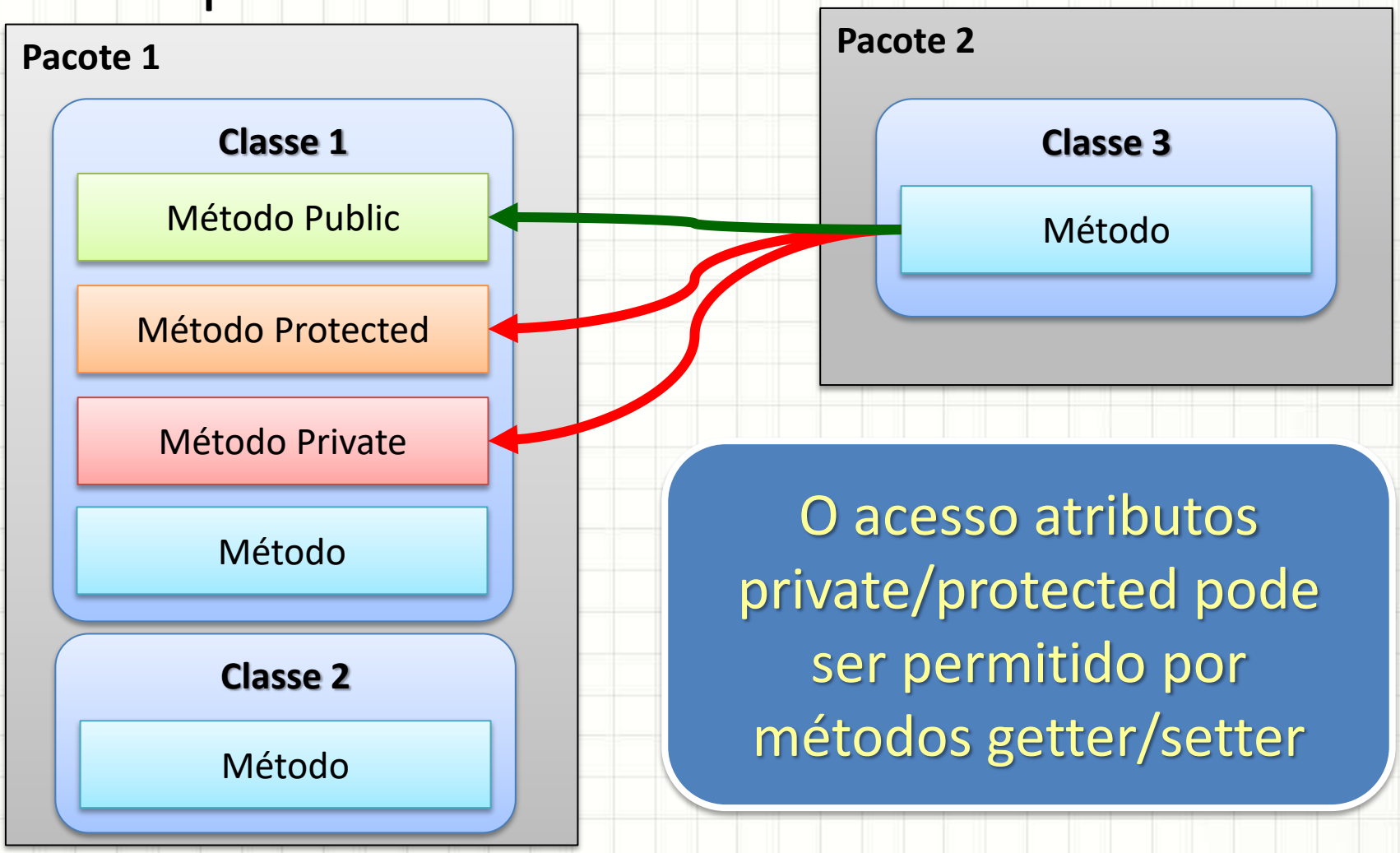
- Compreenda os acessos



Tudo será **private** por padrão; só mudamos para **protected** ou **public** se estritamente necessário!

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 anteiror, 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 o dá
  p

É preciso construir

O que são Objetos?

- U
  u o dá
  p

Transformar a planta

O que são Objetos?

- U
  u o dá
  p

Em um **OBJETO**

Como criar objetos?

- Operador **new**

Classe

new

Objeto

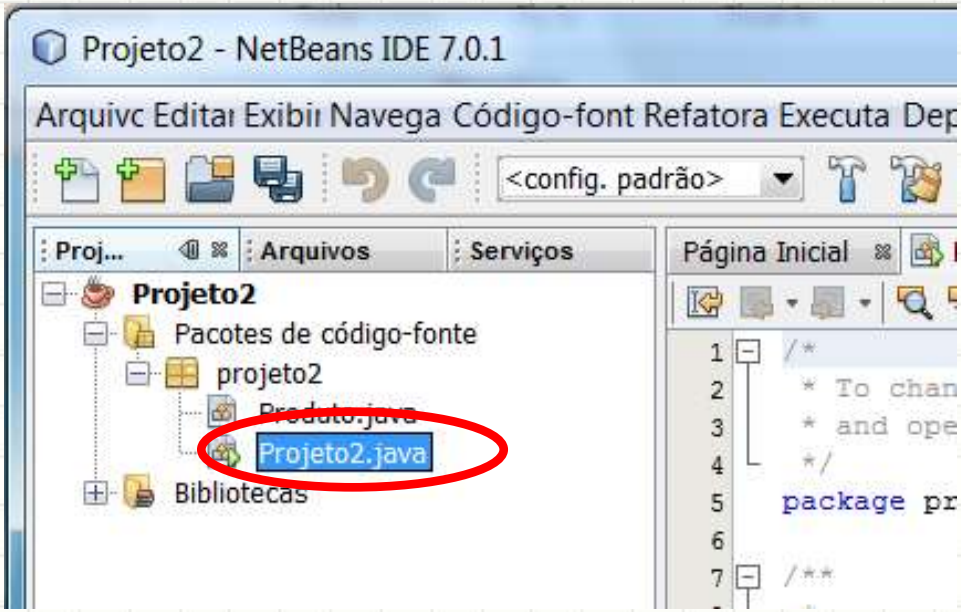
- Ex.: classe Cliente

```
Cliente oCliente = new Cliente();
```

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

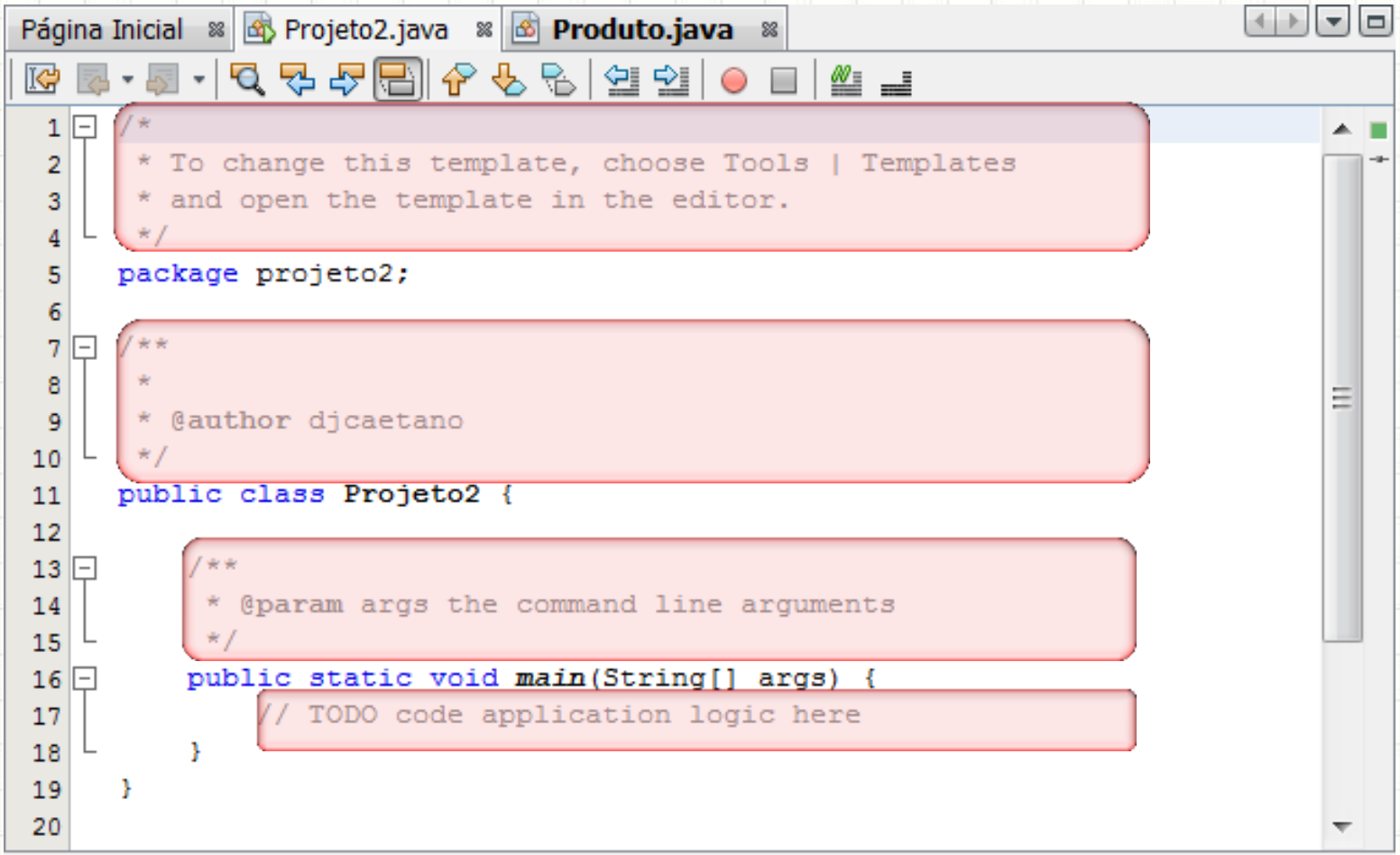
Como criar objetos?

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



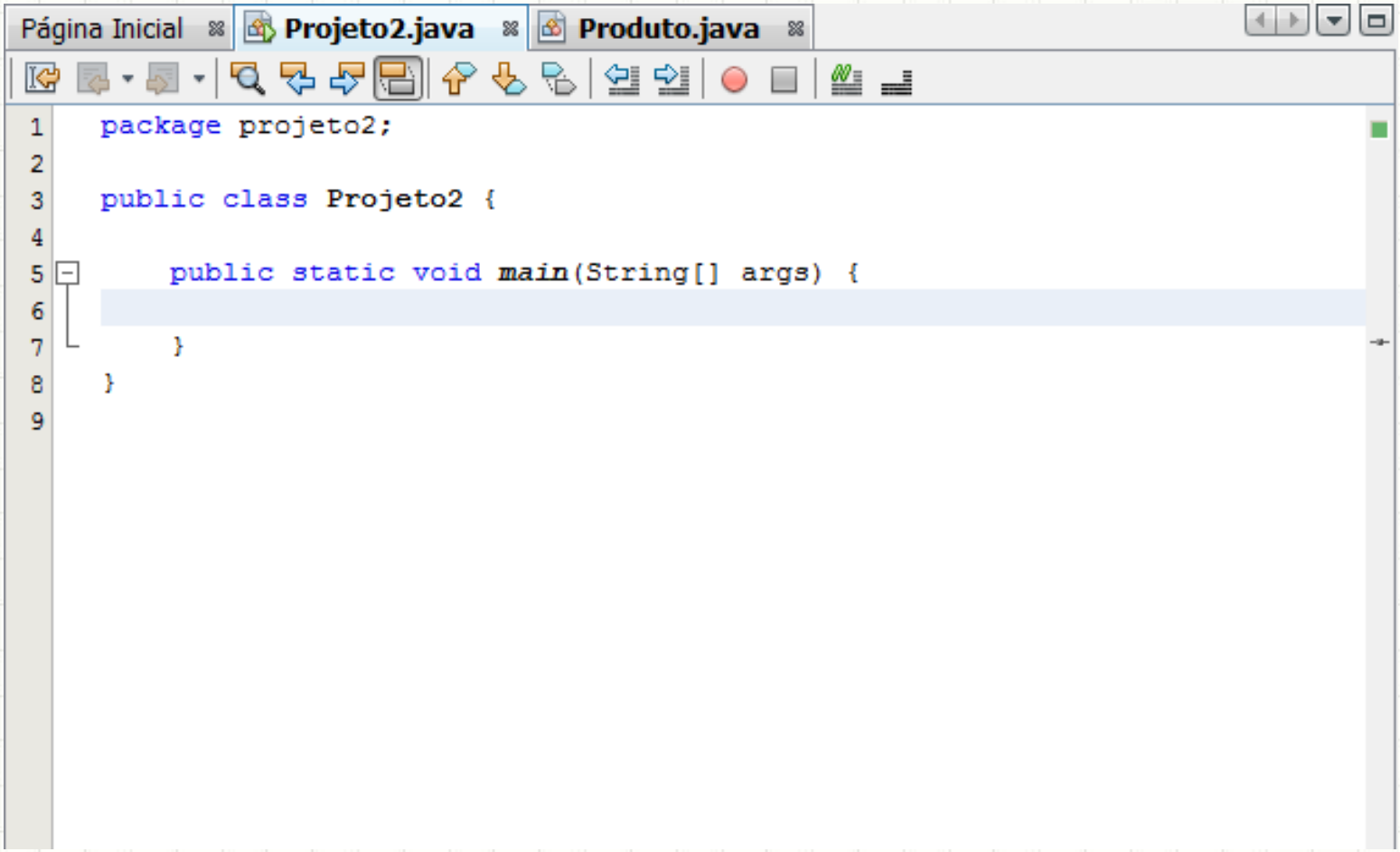
Como criar objetos?

- Apague os comentários do NetBeans...



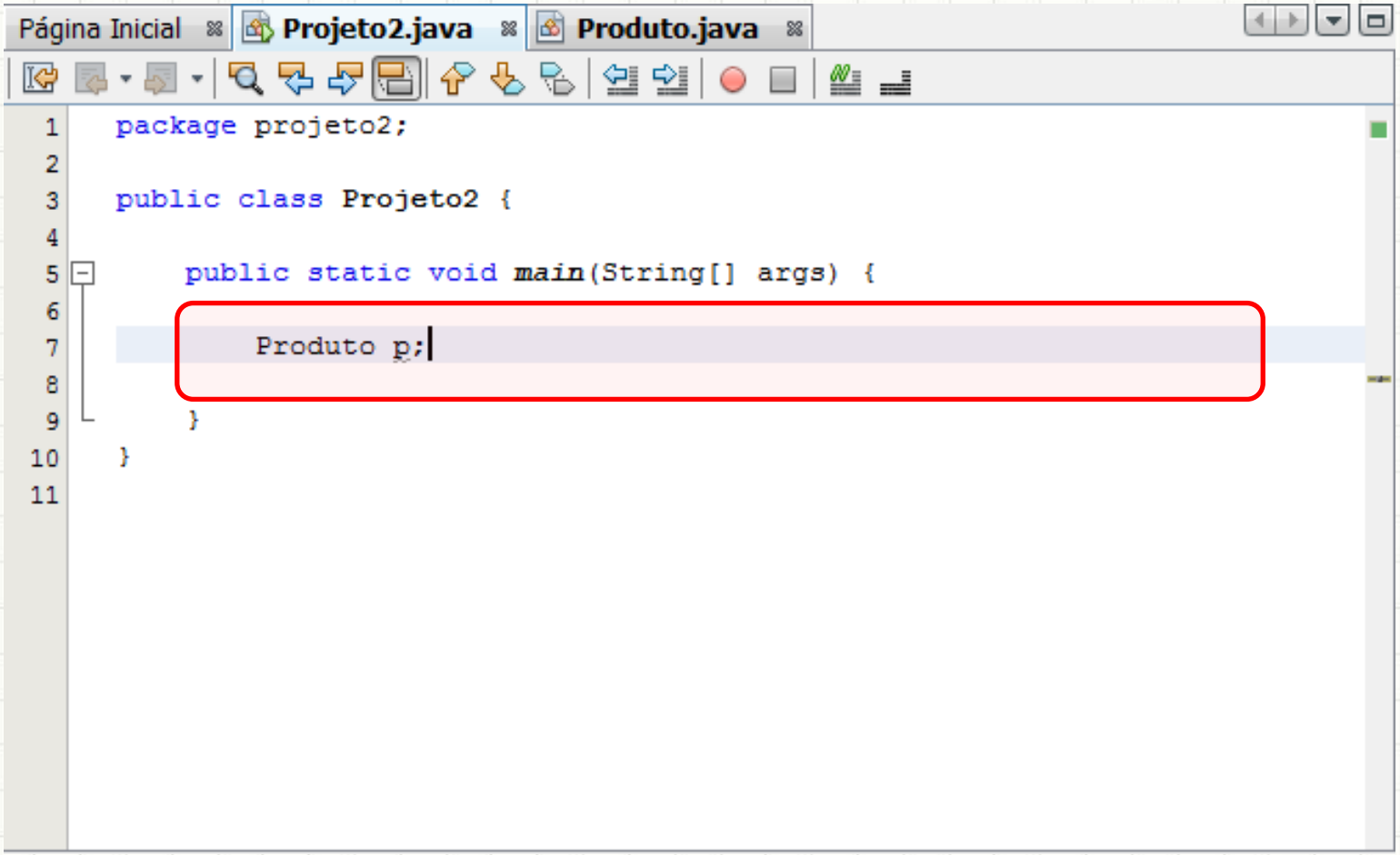
Como criar objetos?

- Apague os comentários do NetBeans...



Como criar objetos?

- Vamos declarar uma variável para o produto



Como criar objetos?

- Vamos declarar uma variável para o produto

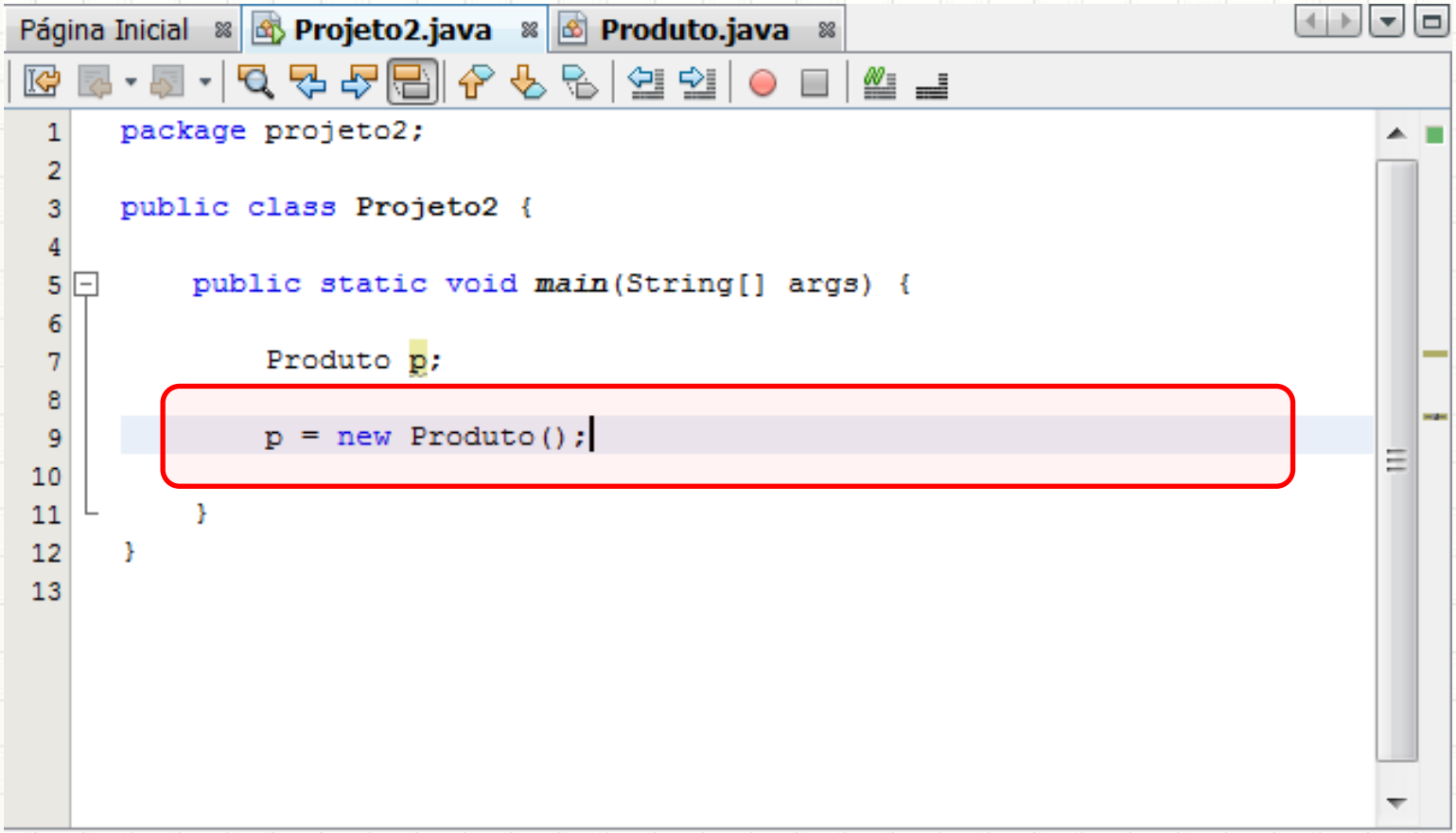
Página Inicial | Projeto2.java | Produto.java

```
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?

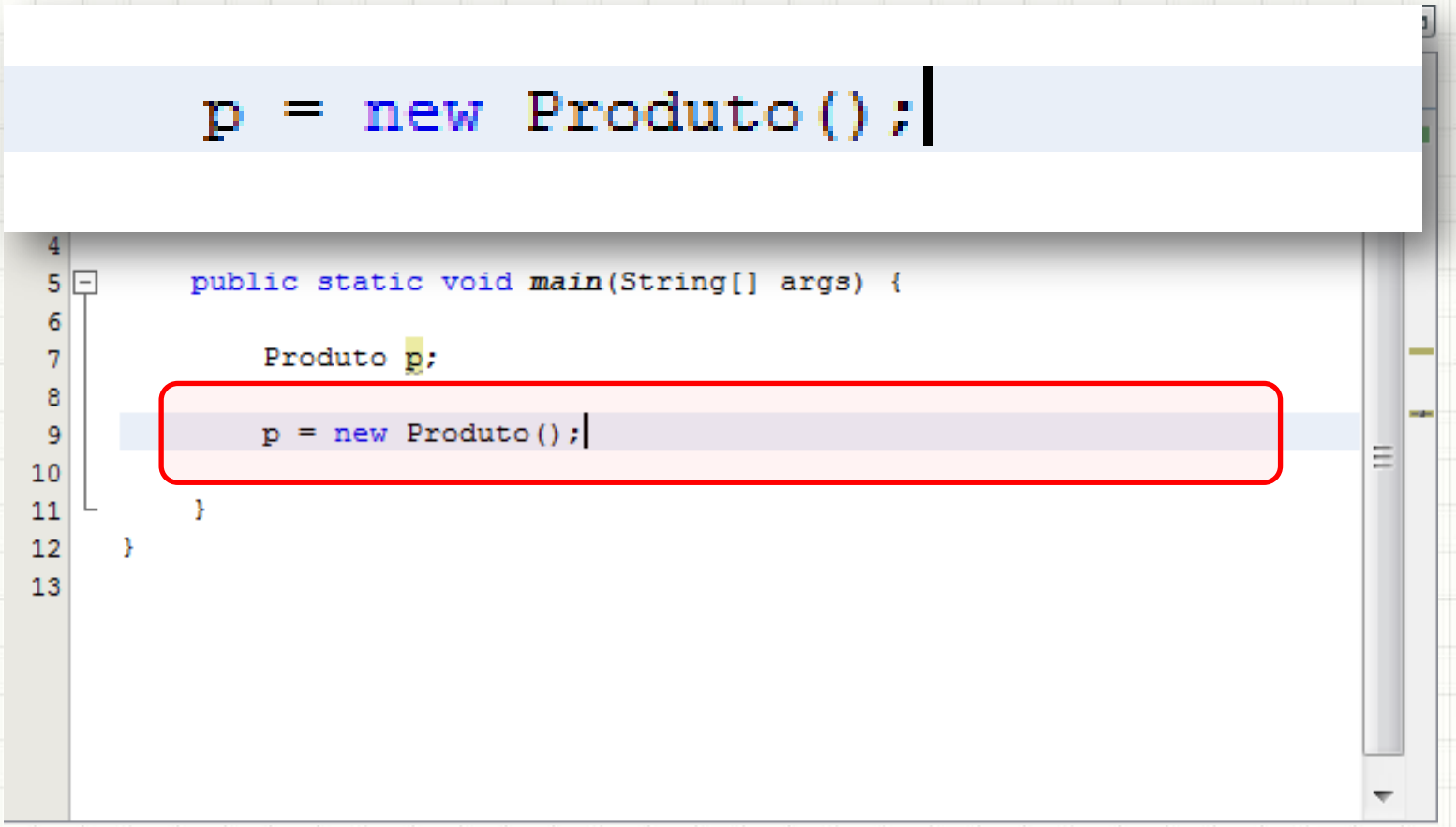
- Agora vamos **criar o objeto**



Como criar objetos?

- Agora vamos **criar o objeto**

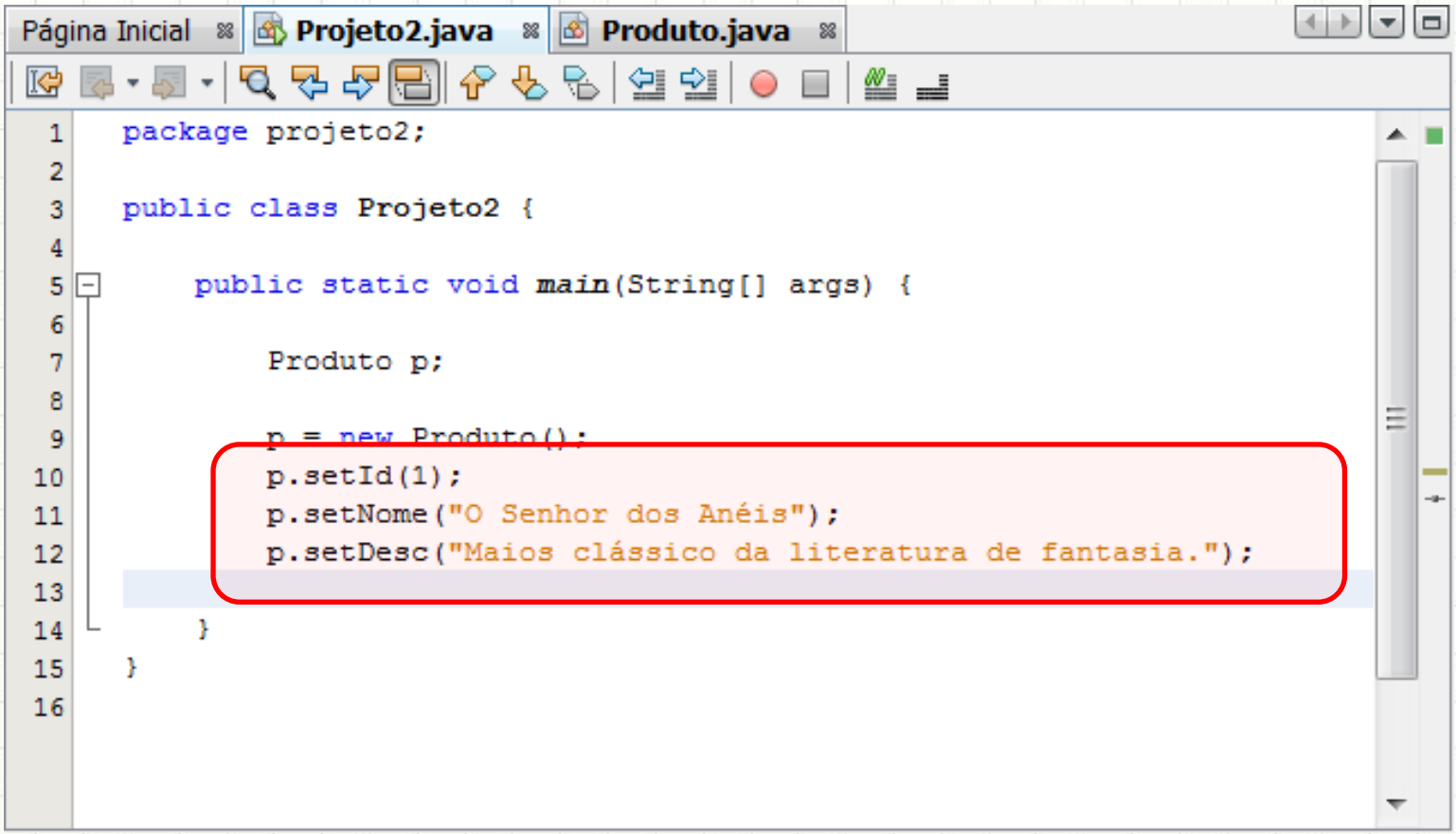
```
p = new Produto();
```



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

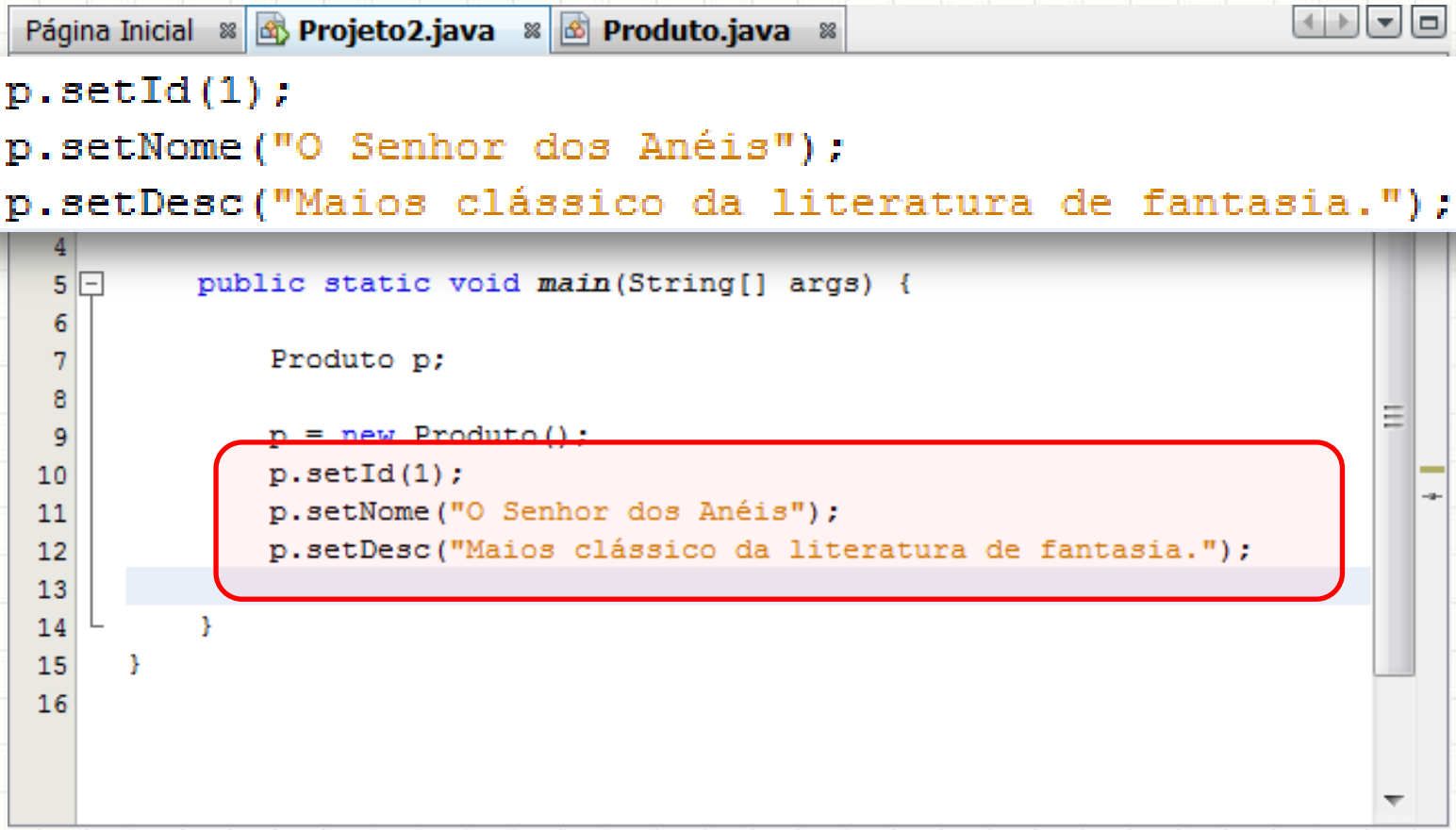
Como criar objetos?

- E **armazenar algumas informações**



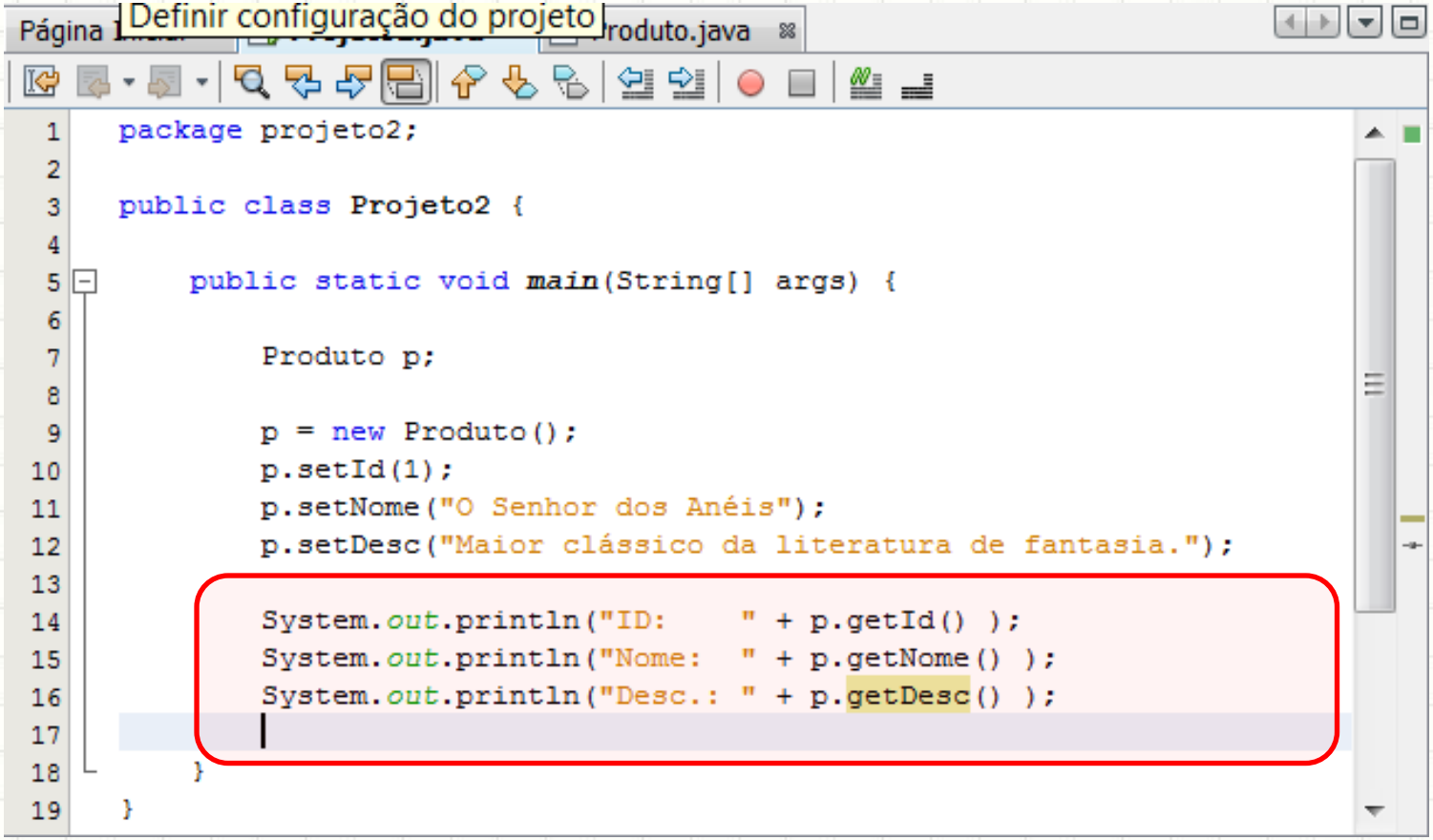
Como criar objetos?

- E **armazenar algumas informações**



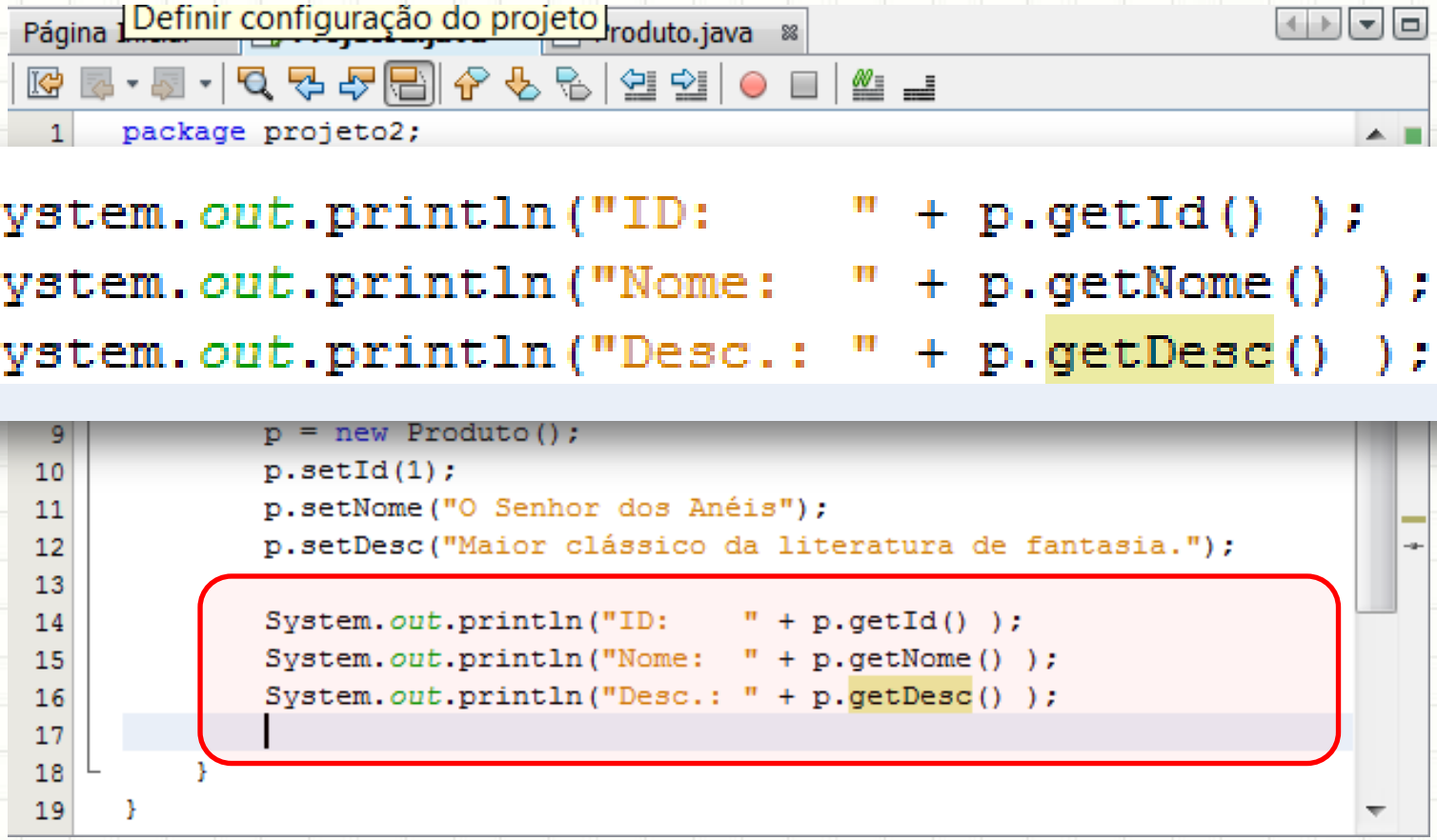
Como criar objetos?

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



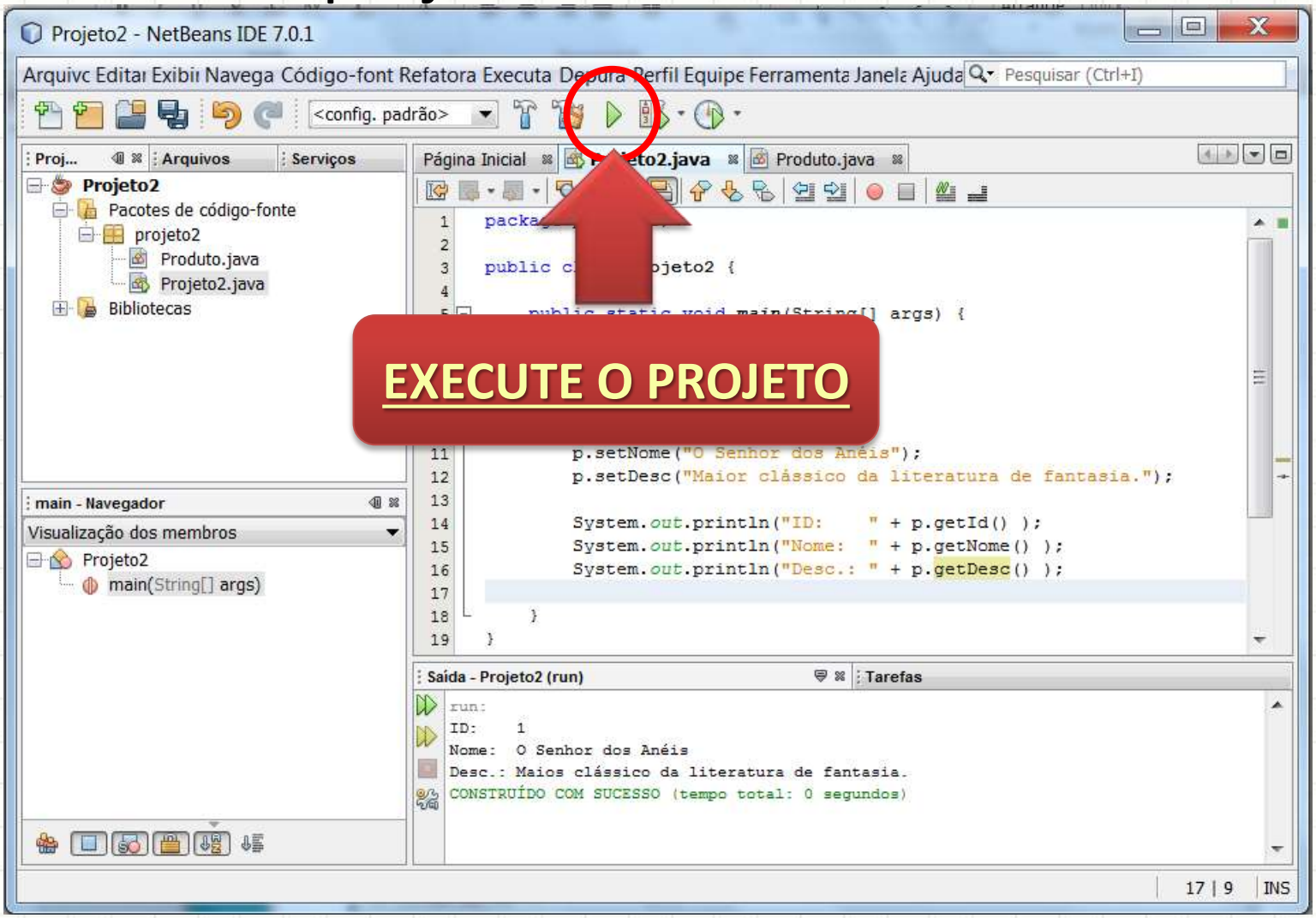
Como criar objetos?

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



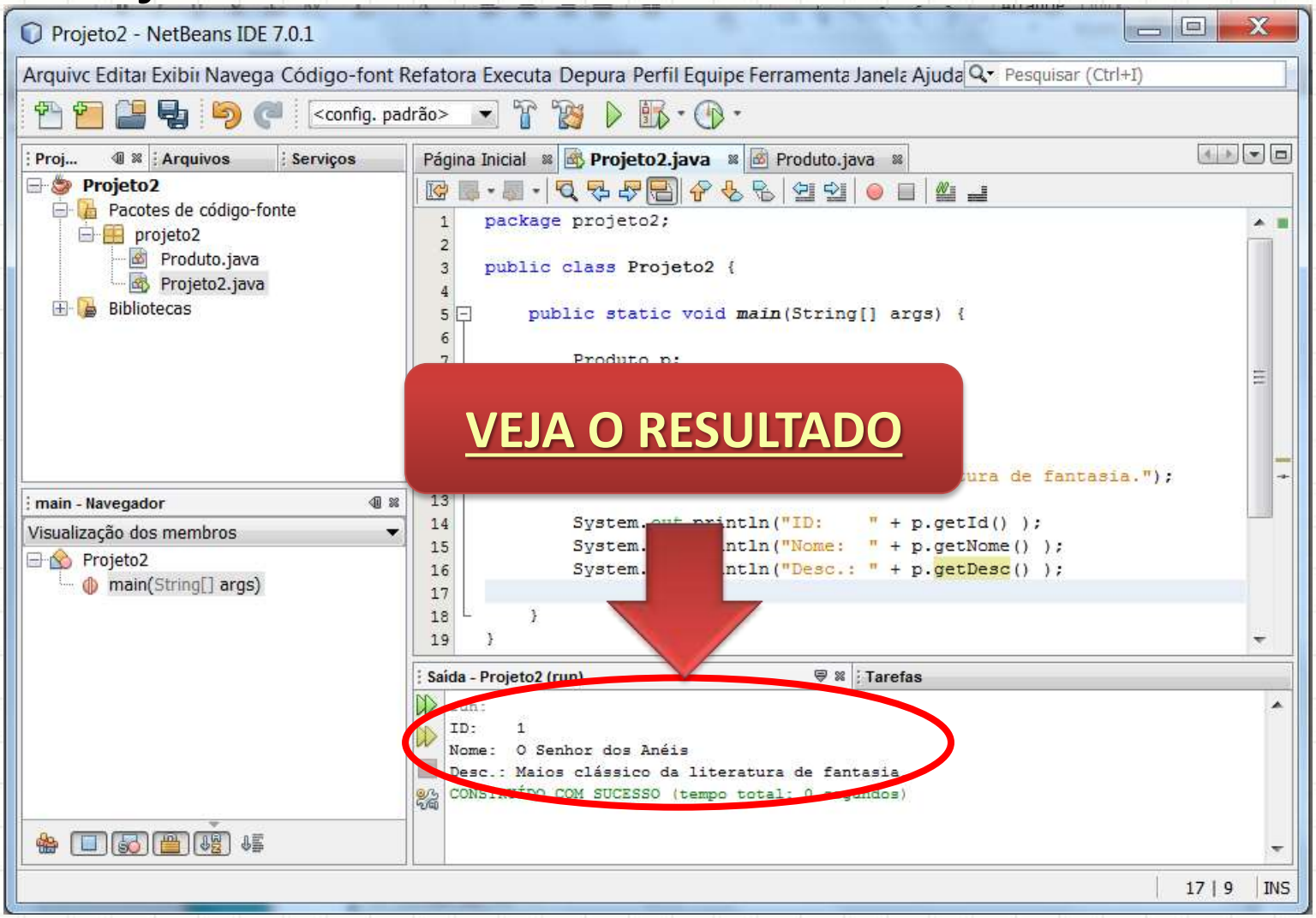
Como criar objetos?

- Execute o projeto...



Como criar objetos?

- E veja o resultado...



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