



- **Classes**

- **Prof. Juliana**

- **10/09/2020**

Roteiro:

2

- Definição de Classe.
- Definição de Objetos.
- Encapsulamento

Classe

3

- ✓ **Classe:** é um modelo de objeto. Consiste de descrições de estado e métodos.
- **Atributo:** definem as características de um objeto. O conjunto de atributos de um objeto é chamado de **estado do objeto**.
- ✓ **Método:** é uma descrição da operação de um objeto. Define o **comportamento de um objeto**.

Conta
- numero : int - saldo : double - limite : double
+ sacar(quantia : double) : boolean + depositar(quantia : double) : void + consultarSaldo() : double

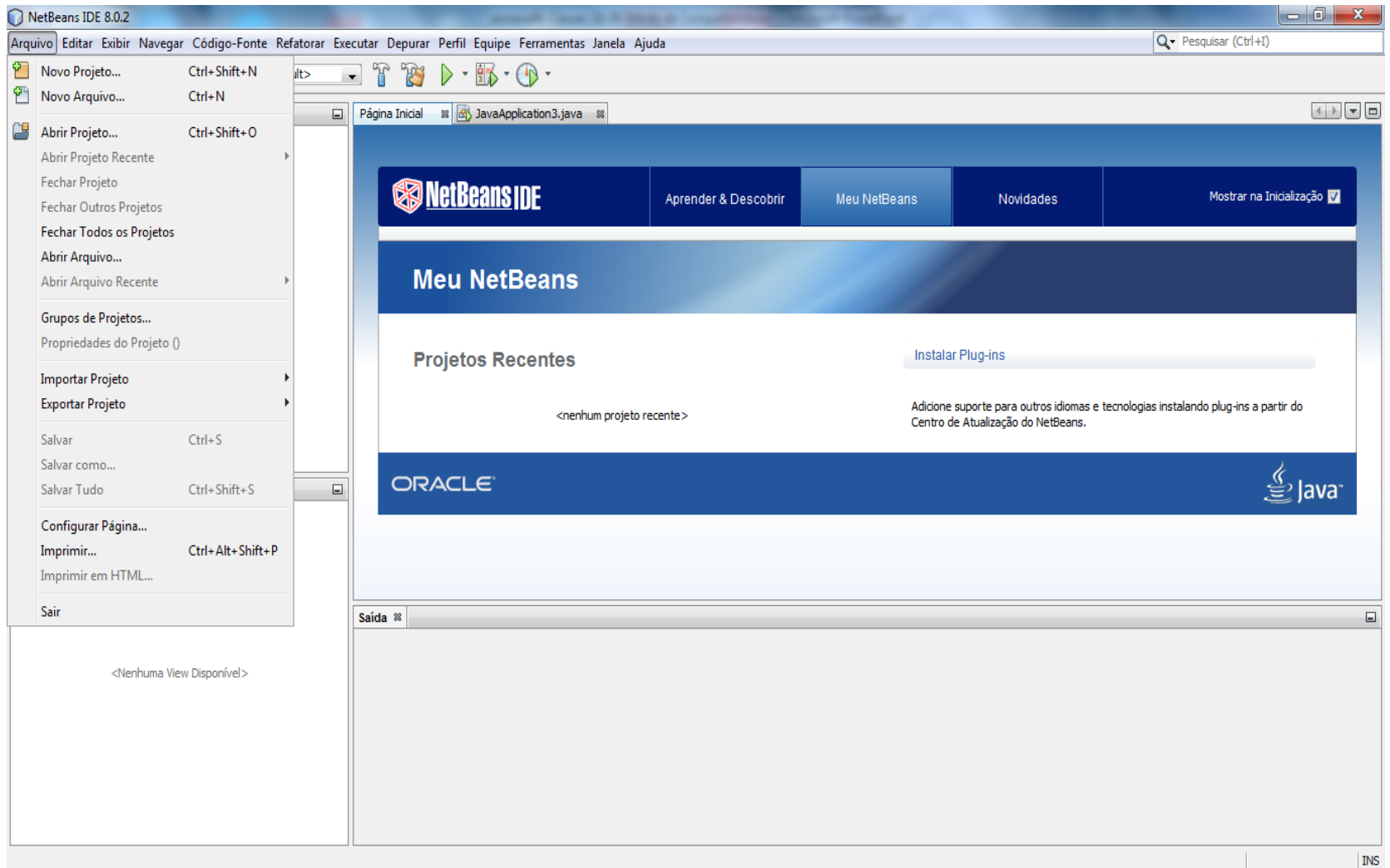
Objetos

- ✓ **Objeto:** é uma instância de uma classe.
- ✓ Um objeto é capaz de:
 - ✓ Armazenar seu estado através dos atributos.
 - ✓ Responder mensagens recebidas através de seus métodos.
 - ✓ Enviar mensagens a outros objetos.
- ✓ **Exemplos:**
 - ✓ Classe Cachorro → Objetos Nana e Cindy
 - ✓ Classe Pessoa → Objetos José e Pedro

Criando um projeto no NetBeans

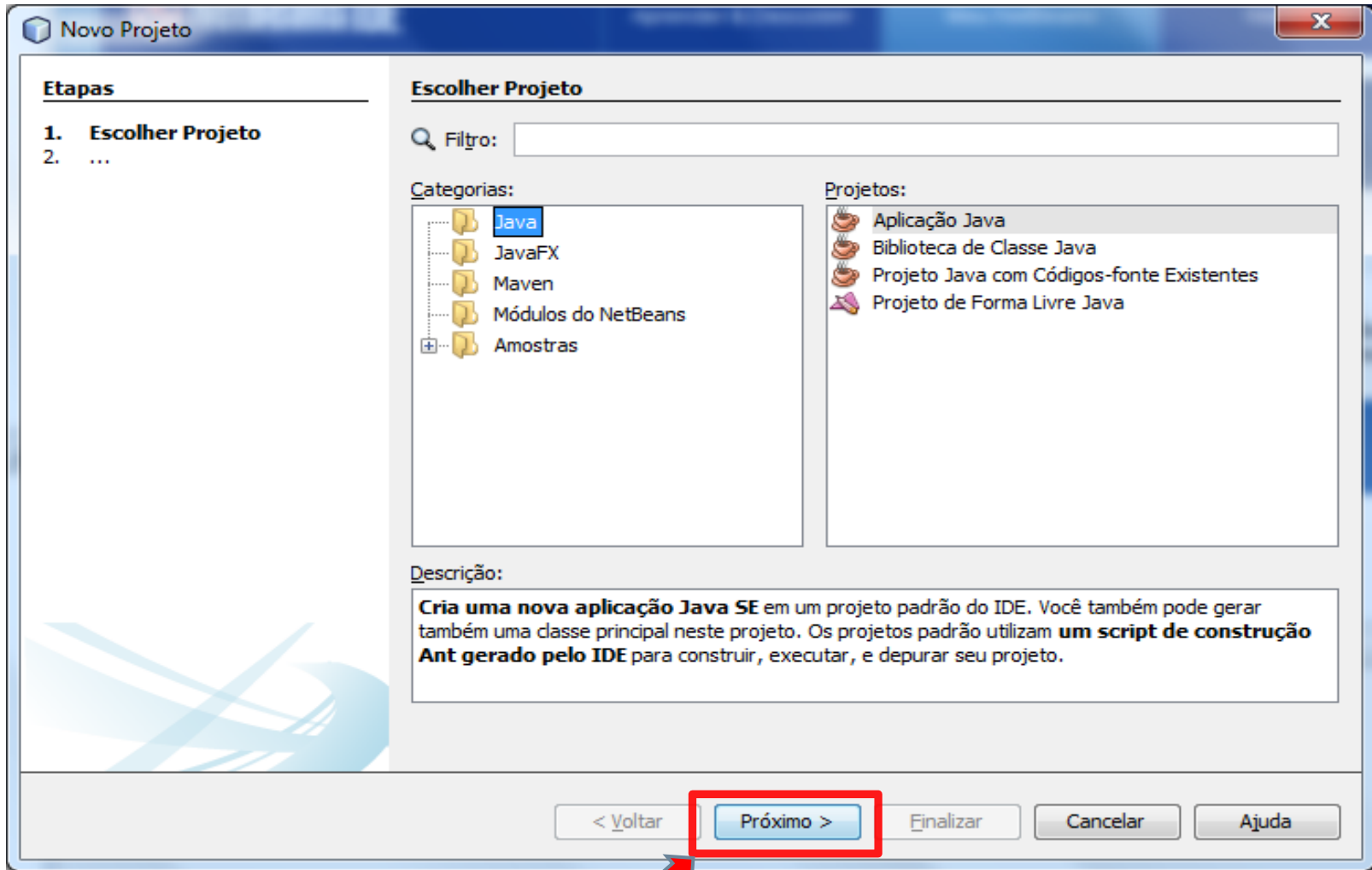
Arquivo -> Novo Projeto

6



Java -> Aplicação Java

7



Nome do Projeto: ProjetoConta

Criar Classe Principal: semana06.Teste

8

Novo Aplicação Java

Etapas

1. Escolher Projeto
2. **Nome e Localização**

Nome e Localização

Nome do Projeto:

Localização do Projeto:

Pasta do Projeto:

☒ Usar Pasta Dedicada para Armazenar Bibliotecas

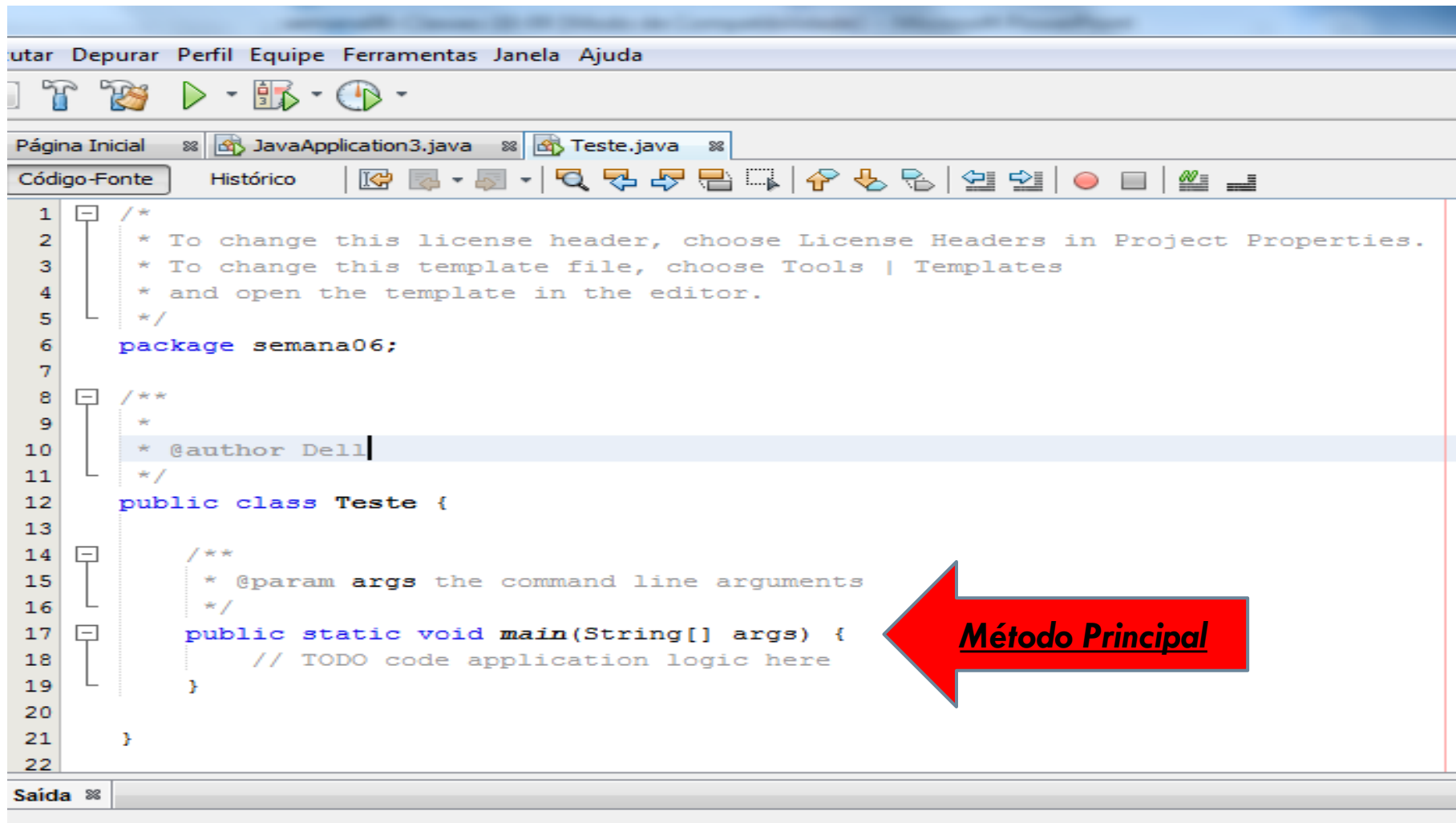
Pasta Bibliotecas:

Usuários e projetos diferentes podem compartilhar as mesmas bibliotecas de compilação (consulte a Ajuda para obter detalhes).

☒ Criar Classe Principal

Criação da Classe Teste

9



The screenshot shows an IDE window with the following components:

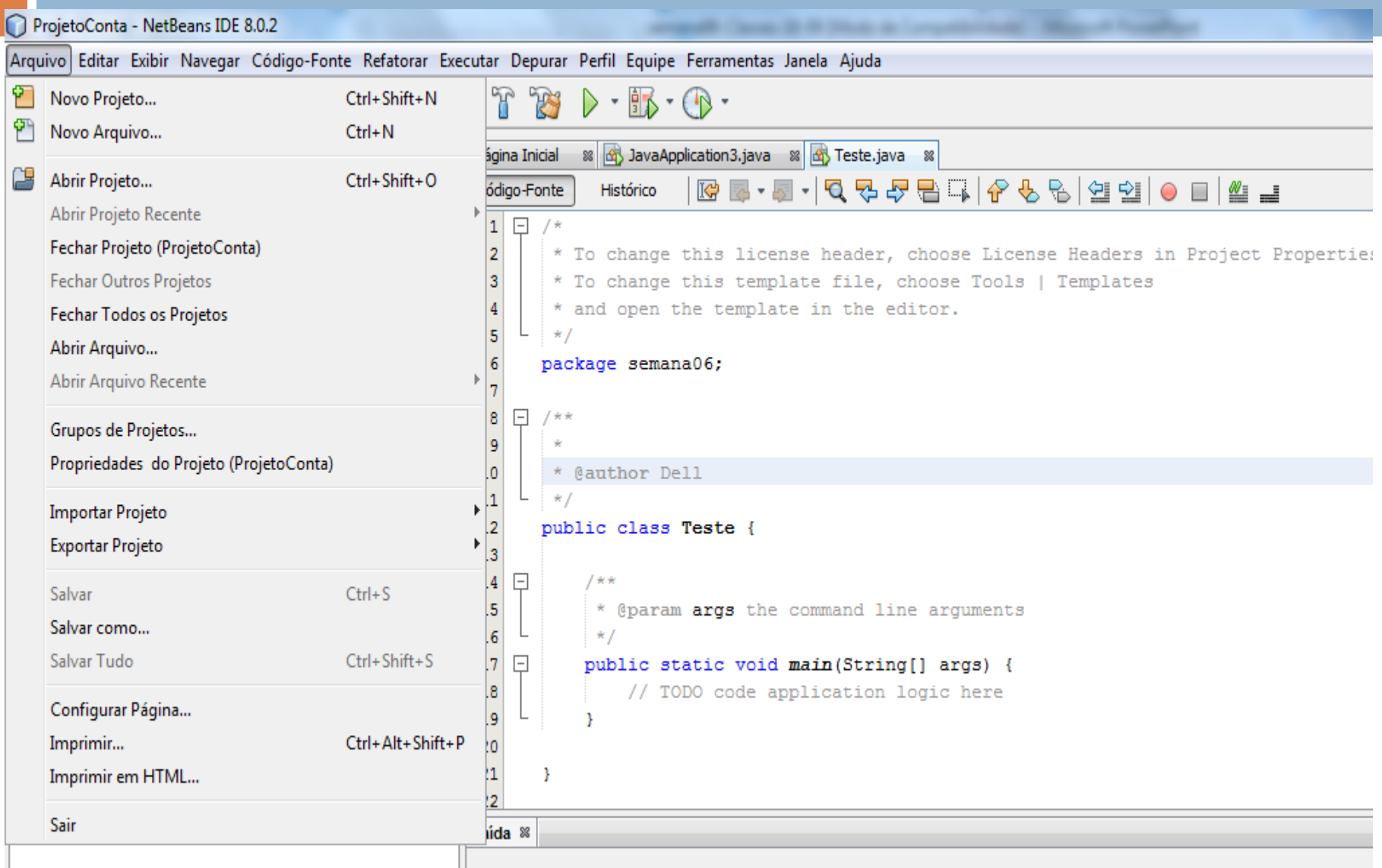
- Menu Bar:** Arquivo, Editar, Depurar, Perfil, Equipe, Ferramentas, Janela, Ajuda.
- Toolbar:** Icons for running, debugging, and other development actions.
- Tab Bar:** Shows two open files: 'JavaApplication3.java' and 'Teste.java'.
- Code Editor:** Displays the source code for 'Teste.java'. The code includes a package declaration, a license header, and a public class 'Teste' with a 'main' method. A red arrow points to the 'main' method with the text 'Método Principal'.
- Output Console:** Labeled 'Saída', it is currently empty.

```
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 semana06;
7
8  /**
9   *
10  * @author Dell
11  */
12  public class Teste {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19      }
20
21  }
22
```

Método Principal

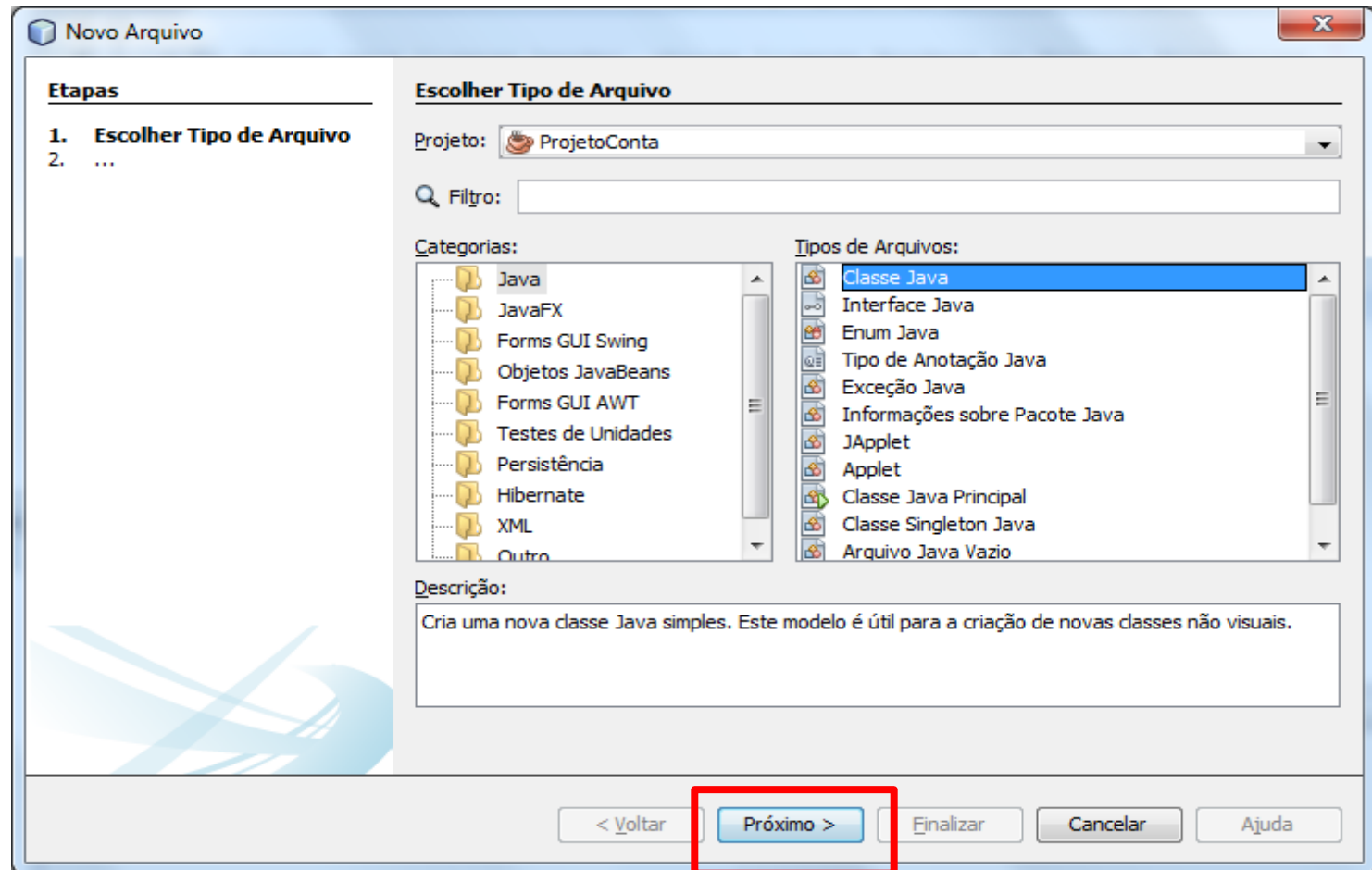
Arquivo -> Novo Arquivo

10



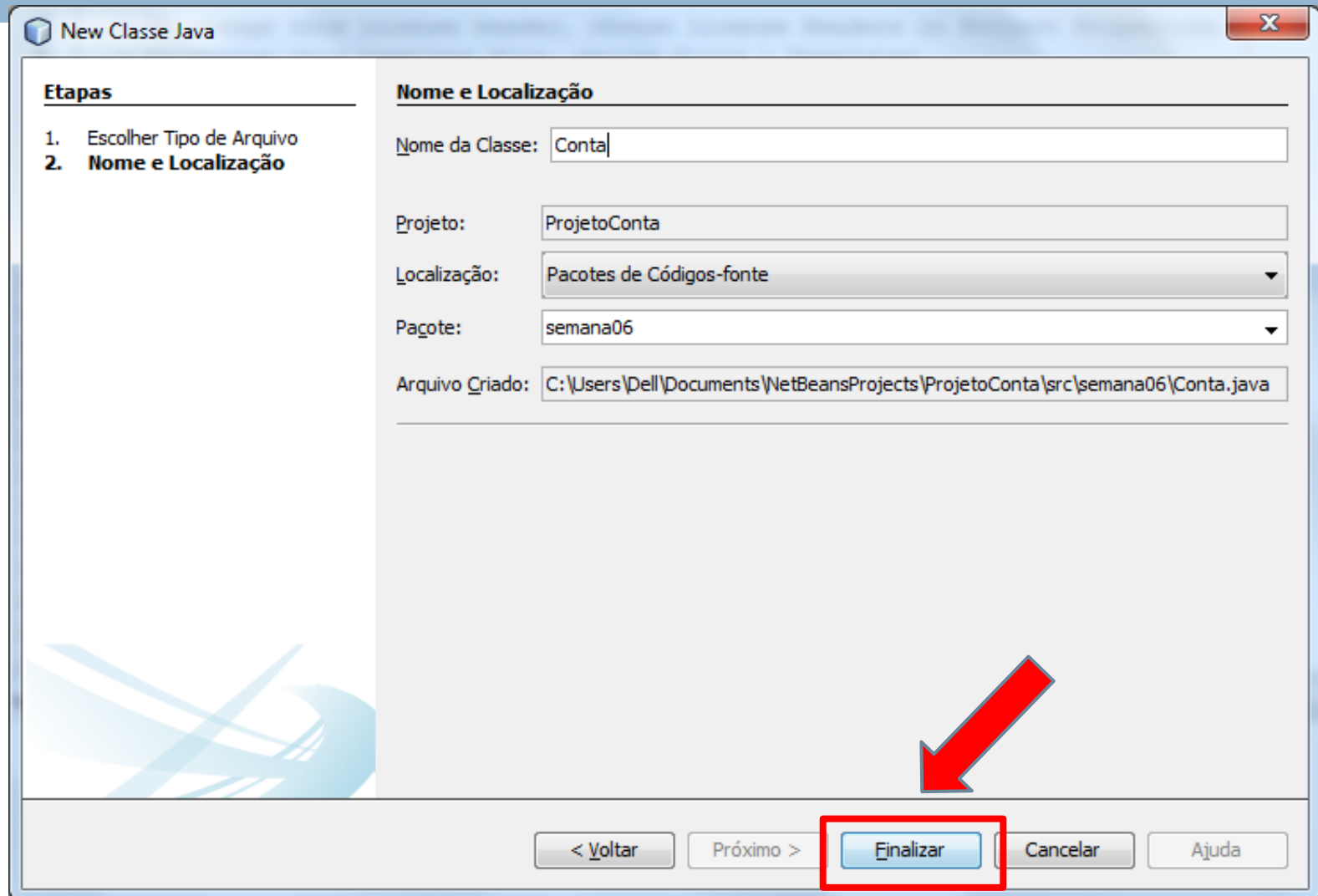
Java -> Classe Java

11



Nome da Classe: Conta

12



New Class Java

Etapas

1. Escolher Tipo de Arquivo
2. **Nome e Localização**

Nome e Localização

Nome da Classe:

Projeto:

Localização:

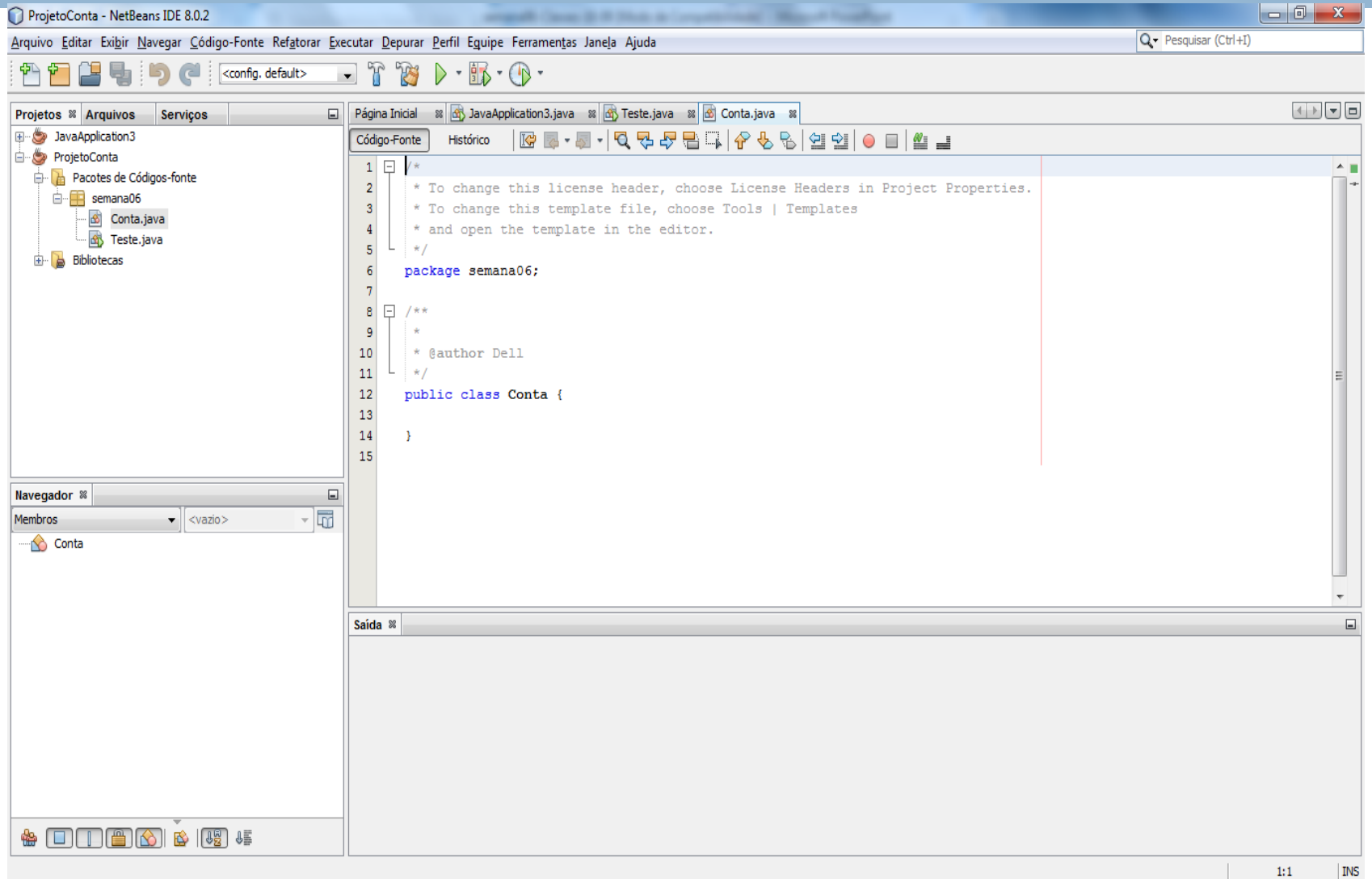
Paquete:

Arquivo Criado:

Finalizar

Classe Conta criada...

13

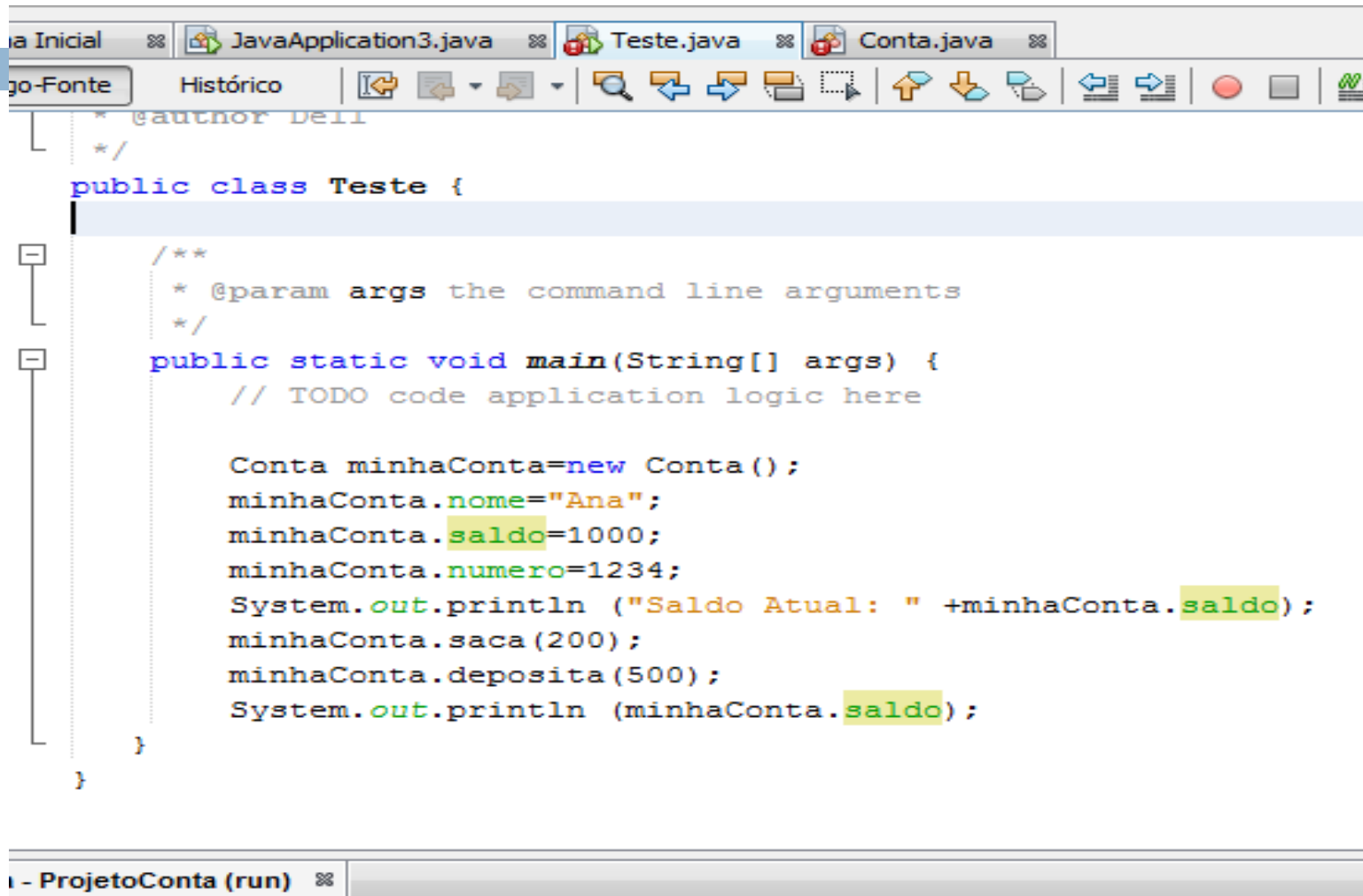


Exemplo – Classe Conta

```
public class Conta {  
    int numero;  
    String nome;  
    double saldo;  
    double limite;  
  
    boolean saca(double valor) {  
        if (this.saldo < valor) {  
            return false;  
        }  
        else {  
            this.saldo = this.saldo - valor;  
            return true;  
        }  
    }  
  
    void deposita(double quantidade) {  
        this.saldo += quantidade;  
    }  
}
```

Classe Teste

15



The screenshot shows an IDE with three tabs: 'JavaApplication3.java', 'Teste.java', and 'Conta.java'. The 'Teste.java' tab is active, displaying the following Java code:

```
/**
 * @author Dell
 */
public class Teste {

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

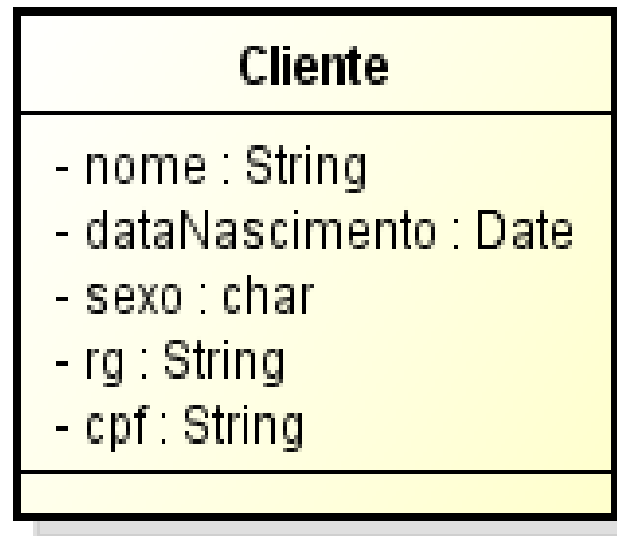
        Conta minhaConta=new Conta();
        minhaConta.nome="Ana";
        minhaConta.saldo=1000;
        minhaConta.numero=1234;
        System.out.println ("Saldo Atual: " +minhaConta.saldo);
        minhaConta.saca(200);
        minhaConta.deposita(500);
        System.out.println (minhaConta.saldo);
    }
}
```

Below the code editor, the 'ProjetoConta (run)' console window shows the execution output:

```
run:
Saldo Atual: 1000.0
1300.0
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

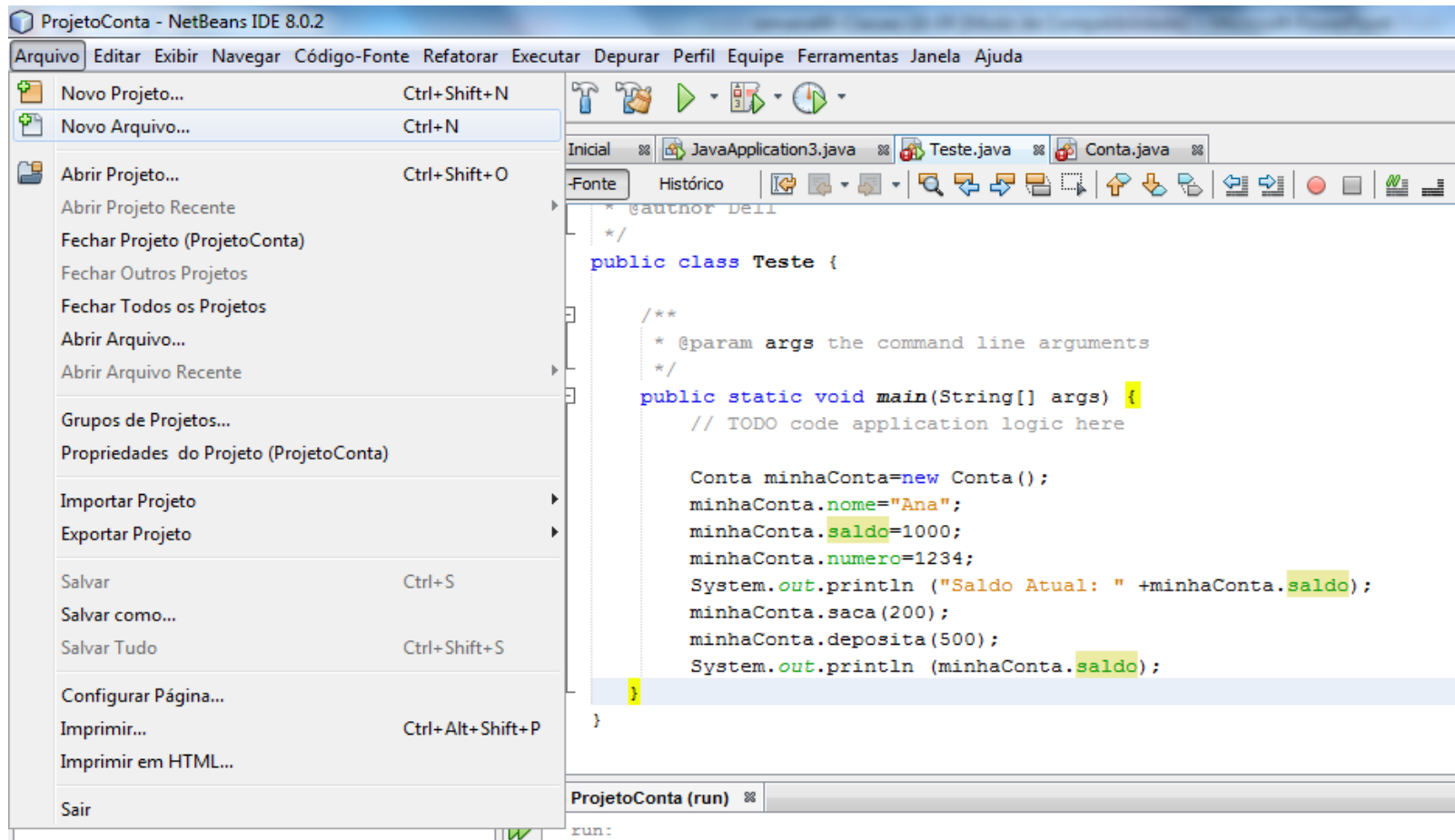
Implementar a Classe Cliente

16



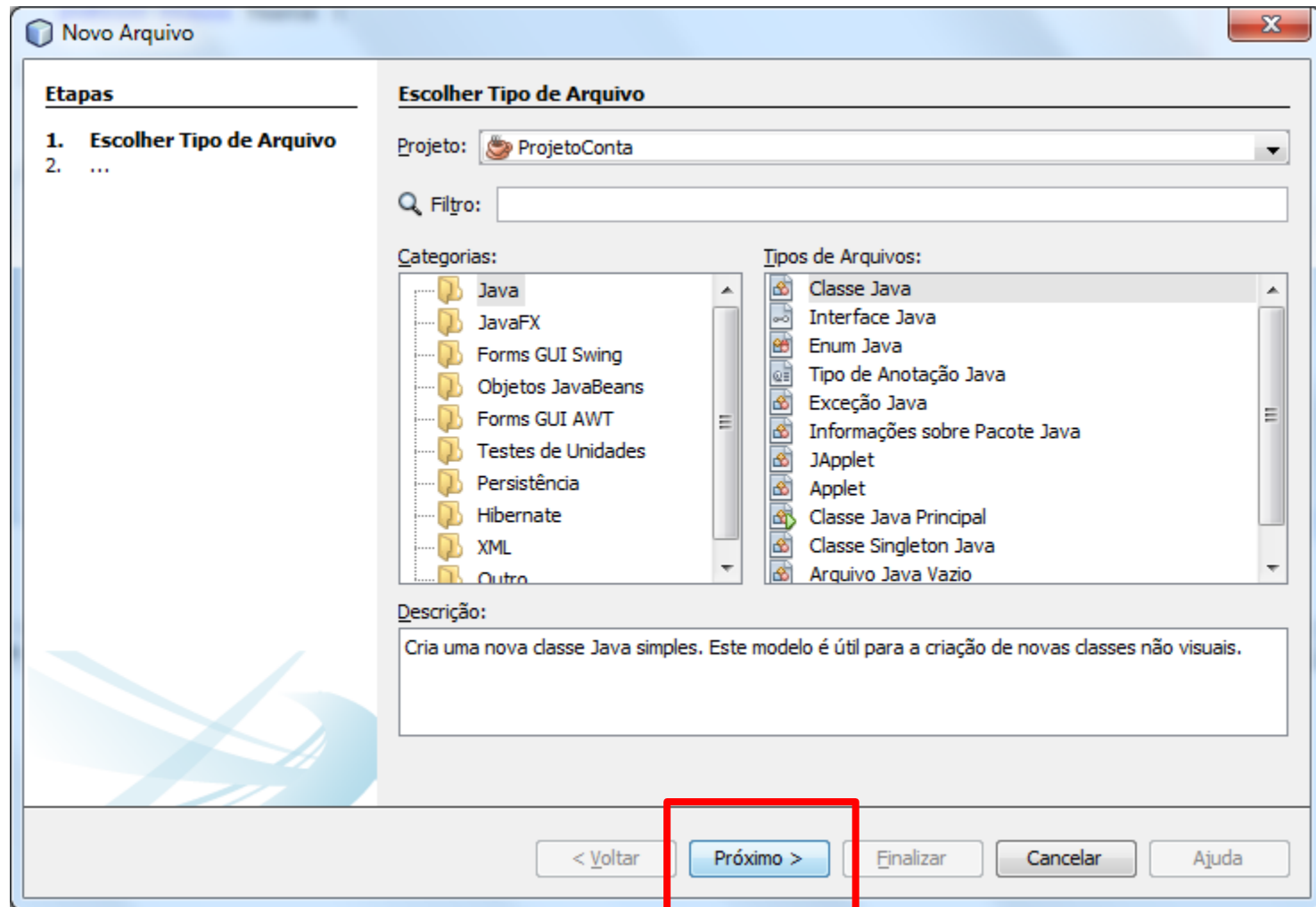
Arquivo -> Novo Arquivo

17



Java-> Classe Java

18



Nome da Classe: Cliente

19

New Classe Java

Etapas

1. Escolher Tipo de Arquivo
2. **Nome e Localização**

Nome e Localização

Nome da Classe:

Projeto:

Localização:

Pacote:

Arquivo Criado:

< Voltar Próximo > **Finalizar** Cancelar Ajuda

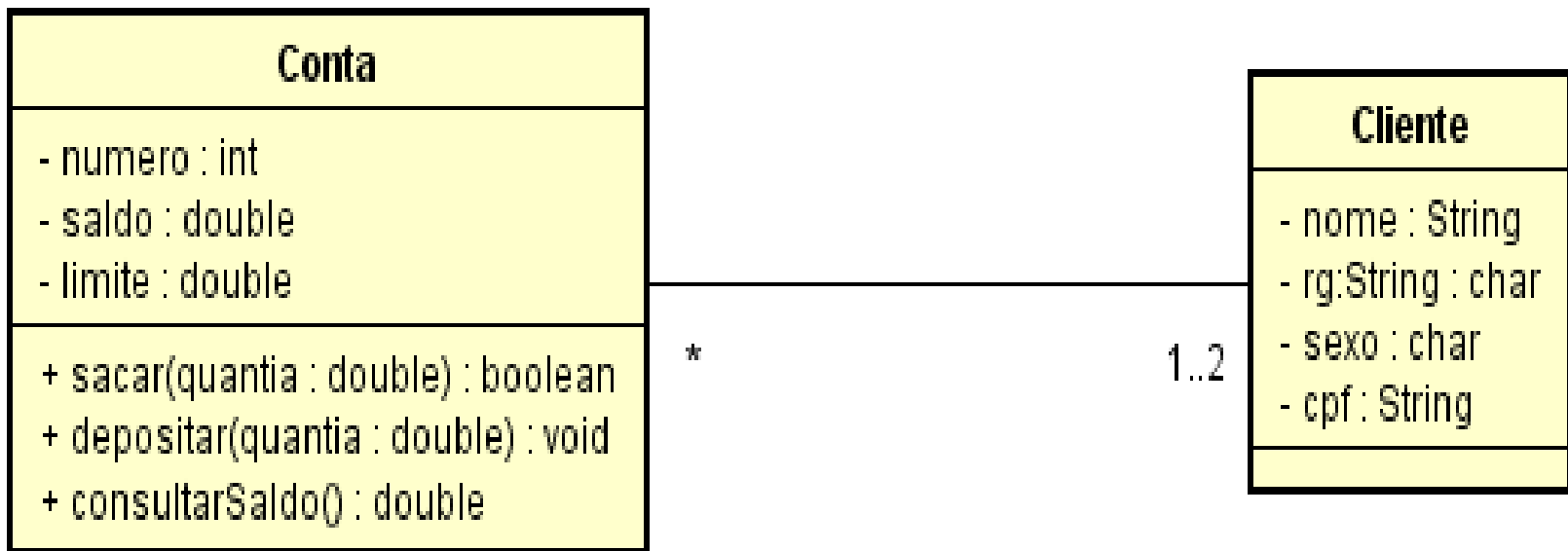
Classe Cliente

20

```
5  L  ... */
6      package semana06;
7
8  [- import java.util.Date;
9
10 [- /**
11     *
12     * @author Dell
13     */
14     public class Cliente {
15         String nome;
16         Date dataNascimento;
17         char sexo;
18         String rg;
19         String cpf;
20     }
21
```

Relacionando a classe Cliente com a classe Conta:

21



Para refletir

- Qual a razão de os métodos sacar, depositar e consultar saldo ficarem na classe conta?
- Estes métodos não deveriam ficar na classe cliente já que é ele quem executa estas operações?

Classe Dog

Dog
Nome
peso
setNome()
setPeso()
latir()

variáveis de instância
(estado)

Métodos
(comportamento)

Tenho uma **referência primeiro** a um **objeto** do tipo **Cachorro**.

```
class Dog {  
    String nome;  
    int peso;  
  
    void latir() {  
        if (peso >= 9) {  
            System.out.println ("AUUUUU!");  
        } else if (peso < 9) {  
            System.out.println ("AU!");  
        }  
    }  
}
```

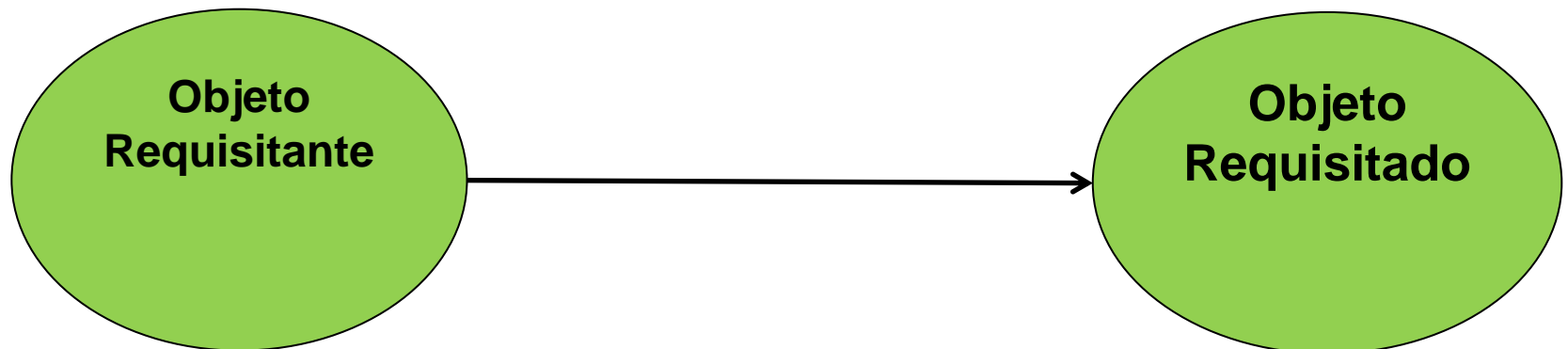
```
Class Teste{  
    public void main (String[] args) {  
  
        Dog primeiro=new Dog();  
        primeiro.nome="Nina";  
        primeiro.peso=1;  
  
        Dog segundo=new Dog();  
        segundo.nome="Bravo";  
        segundo.peso=10;  
  
        primeiro.latir();  
        segundo.latir();  
    }  
}
```

Encapsulamento

- ❑ **Encapsulamento:** “Esconder” o comportamento e os atributos de um objeto.
- ❑ Usuário de uma classe não precisa conhecer detalhes da implementação, precisa apenas conhecer sua interface.
- ❑ Vantagem 1: podemos alterar a implementação sem afetar os usuários da classe.
- ❑ Vantagem 2: proteger o acesso direto aos atributos da classe, manter a consistência dos valores.

Encapsulamento

- ❑ Criamos métodos para acessar os atributos da classe.
- ❑ Métodos chamados de getters e setters.
- ❑ Aplicação da abstração: esconder os detalhes do funcionamento interno de uma classe.



Encapsulamento

26

Acessando atributos encapsulados:

- **Getters**
 - ✓ **Objetivo:** permitir que o atributo encapsulado seja lido por outra classe;
 - ✓ **Nomenclatura:** utiliza-se o prefixo get seguido do nome da variável;
 - ✓ **Acesso:** não-privado (mais utilizado como public);
 - ✓ **Tipo de Retorno:** o mesmo tipo do atributo encapsulado;
 - ✓ **Parâmetro:** nenhum ()

Encapsulamento

27

Acessando atributos encapsulados:

- **Setters**
 - ✓ **Objetivo:** permitir que o atributo encapsulado possa ter seu valor modificado por outra classe;
 - ✓ **Nomenclatura:** utiliza-se o prefixo set seguido do nome da variável;
 - ✓ **Acesso:** não-privado (mais utilizado como public);
 - ✓ **Tipo de Retorno:** não retorna valor - *void*;
 - ✓ **Parâmetro:** apenas um, de mesmo tipo e mesmo nome do atributo encapsulado.

Prática

28

- Criar um novo Projeto DOG.
- Criar uma classe Dog

Encapsulamento a classe Dog

```
public class Dog {
```

```
    private String nome;  
    private int peso;
```

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

```
    public int getPeso(){  
        return peso;  
    }
```

```
    public void setNome(String n) {  
        nome=n;  
    }
```

```
    public void setPeso (int p) {  
        peso=p;  
    }
```

```
    void latir() {  
        if (peso >= 9) {  
            System.out.println ("AUUUUU!");  
        } else if (peso < 9) {  
            System.out.println ("AU!");  
        }  
    }  
}
```

torna a variável de
instancia privada

métodos de captura e
configuração públicos

Encapsulamento a classe Dog

```
public class Teste {  
    public static void main (String[] args) {  
        DOG primeiro=new DOG();  
        primeiro.setNome("Nina");  
        primeiro.setPeso(1);  
  
        DOG segundo=new DOG();  
        segundo.setNome("Bravo");  
        segundo.setPeso(10);  
  
        System.out.println ("Dog Primeiro:" + primeiro.getNome());  
        System.out.println ("Peso Primeiro" + primeiro.getPeso());  
        primeiro.latir();  
  
        System.out.println ("Dog Segundo" + segundo.getNome());  
        System.out.println ("Peso Segundo" + segundo.getPeso());  
        segundo.latir();  
    }  
}
```