



- **Orientação a Objetos**

- **Prof. Juliana**

- **27/8/2025**

Roteiro:

- Motivação;
- Paradigma de Orientação a Objetos;
- Conceitos de Orientação a Objetos:
 - ✓ Classes;
 - ✓ Objetos;
 - ✓ Atributos;
 - ✓ Métodos;
 - ✓ Encapsulamento;
 - ✓ Abstrações.

Todos sabem como se faz ovos de páscoa?

Como damos o formato de ovo para o chocolate?



Basta usar uma forma!

E como fazemos para criar outros formatos?



O segredo está nas formas

- Com as formas podemos criar diversas vezes o chocolate que "precisamos". Assim não há necessidade de esculpir o ovo artesanalmente todas as vezes;
- As formas são nossas **Classes**;
- Os ovos de chocolate são nossos **Objetos**;
- Mas o que isso tem a ver com software?

Nem sempre é tão simples...



Estamos aqui para projetar softwares complexos



Paradigma da Orientação a Objetos

8

R = É uma forma de abordar um problema.

- Formalização estabelecida;
- Linguagem própria: sintaxes, códigos, significados que permitam comunicar idéias.

Paradigma da Orientação a Objetos

9

- **Alan Kay** (um dos pais da orientação a objetos) estabeleceu os princípios da O.O.:
 - ✓ Qualquer coisa é um objeto.
 - ✓ Objetos realizam tarefas através da requisição de serviços a outros objetos.
 - ✓ Cada objeto pertence a uma determinada classe.
 - ✓ Uma classe agrupa objetos similares.
 - ✓ A classe é um repositório de todo comportamento associado ao objeto.

Paradigma da Orientação a Objetos

10

- Um programa de computador implementa uma solução para um problema do mundo real.
- O paradigma de O.O. assume que o mundo é composto de objetos.
- Todo objeto em O.O. é composto por:
 - ✓ Estado interno
 - ✓ Comportamento

Paradigma da Orientação a Objetos

11

- **Estado Interno de um Objeto:**
 - ✓ É definido pelos dados (atributos);
 - ✓ Registra e permite ao objeto se lembrar do efeito de sua operação.

Paradigma da Orientação a Objetos

12

- **Comportamento de um Objeto:**
 - ✓ É definido pelo seu conjunto de operações (métodos);
 - ✓ As operações de um objeto são utilizadas para responder mensagens internas ou externas;
 - ✓ O resultado de uma operação depende da mensagem recebida e do estado interno do objeto

Conceitos de Orientação a Objetos

Classes

- ✓ Classe: é um modelo de objeto. Consiste de descrições de estado e métodos.
- ✓ Os **métodos** são descrições do comportamento do objeto. Definem o comportamento dos objetos de uma classe.
- ✓ Os **atributos** definem as características de um objeto. O conjunto de atributos de um objeto é chamado de estado do objeto.

✓

Classe

15

- ✓ **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**.
- ✓ Exemplos: Conta, Atendimento e Pessoa

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

Classes

16

- **Classes:**

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

Bicicleta

Pessoa

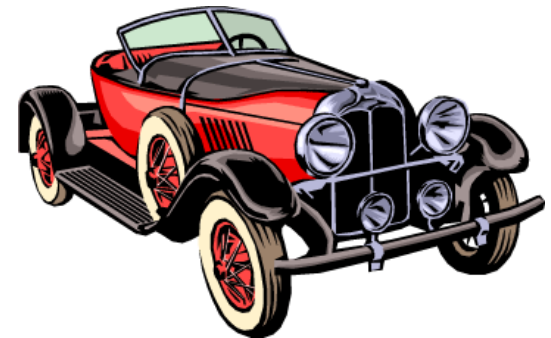
Objetos

- ✓ **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



Atributos

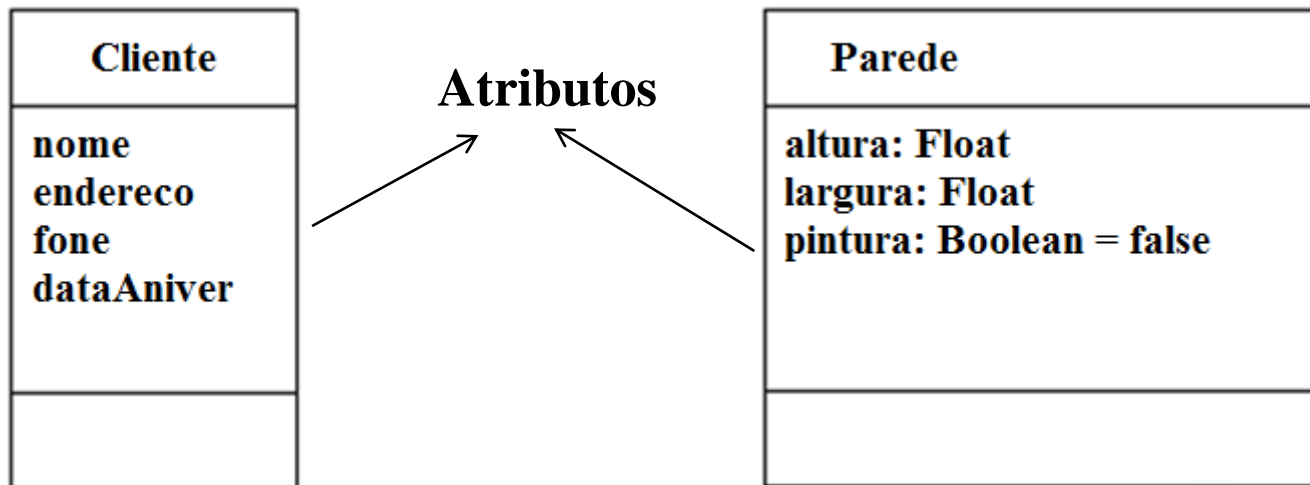
- **Atributo:** propriedade de uma classe que definem as características de um objeto.
- O conjunto de atributos de um objeto é chamado de estado do objeto.
- **Exemplos:**
 - ▣ Cachorro → Cor, idade, nome.
 - ▣ Carro → Cor, marca, ano.
 - ▣ Pessoa → Data Nascimento, nome, RG.



Atributos

19

- Exemplos de Atributos:



Atributos

20

- Os atributos podem ser de instância ou de classe :
 - ✓ **Atributos de classe** são constantes com valores pré-definidos e que não sofrem alteração ao longo da execução;
 - ✓ **Atributos de instância** determinam o estado de cada objeto e sofrem alteração ao longo da execução do programa (variáveis).

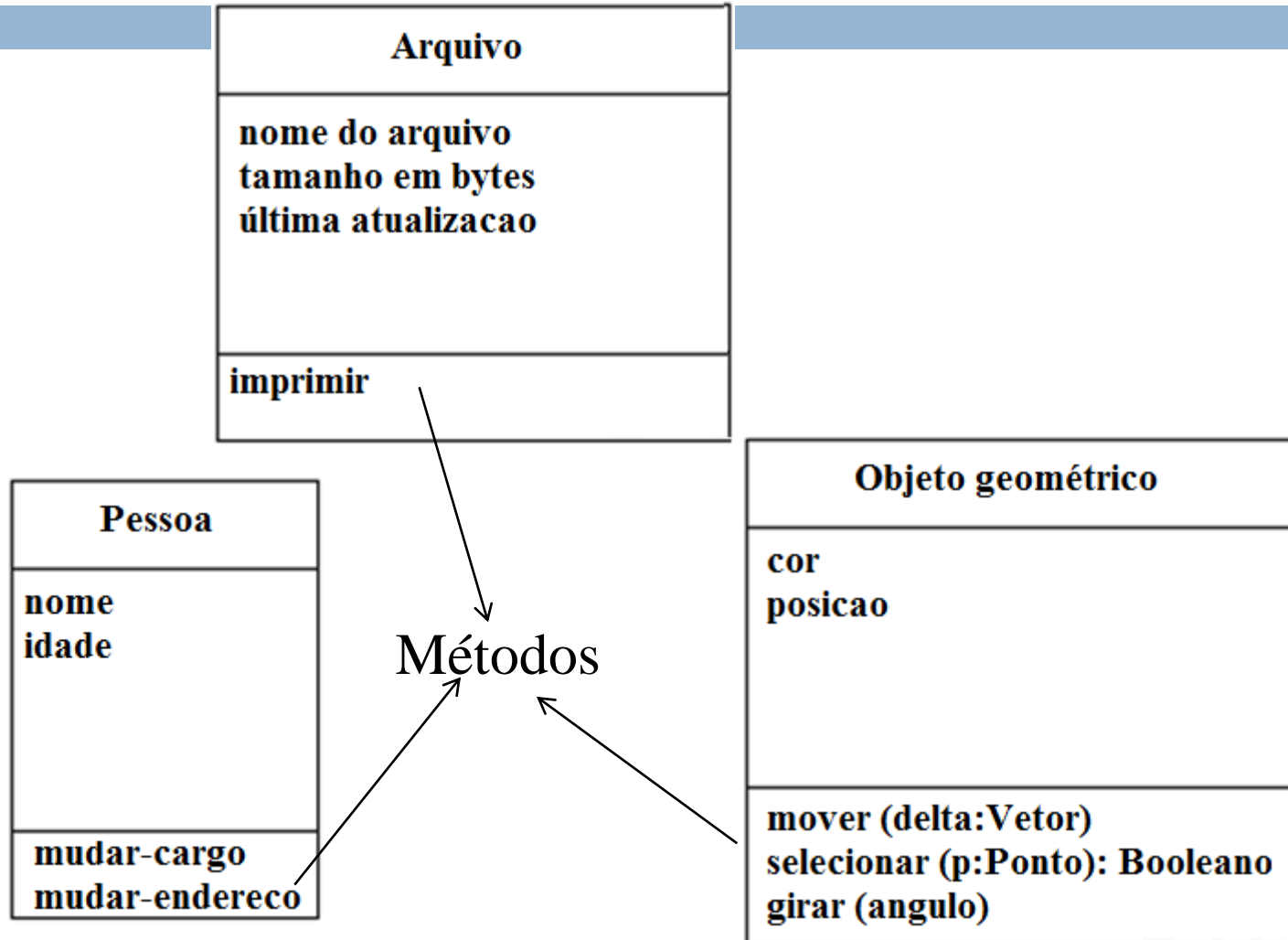
Métodos

- ✓ **Método:** é uma descrição da operação de um objeto. Define o comportamento de um objeto.
- ✓ **Exemplos:**
 - ✓ Carro → Ligar, desligar, correr
 - ✓ Cachorro → Latir, comer, dormir



Métodos

22



Comportamento dos Objetos

23

- O **estado** afeta o comportamento, o **comportamento** afeta o estado.



*Variáveis de
Instâncias*



Métodos

Usam as
variáveis de
instâncias

- Cada instância de uma classe (**cada objeto de um tipo específico**) pode ter seus próprios valores exclusivos para suas variáveis de instância.



Nome: Nina
Peso= 1 Kilo



Nome: Bravo
Peso= 10 Kilo

latir()

Exemplo Classe Dog

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

variáveis de instância
(estado)

Métodos
(comportamento)

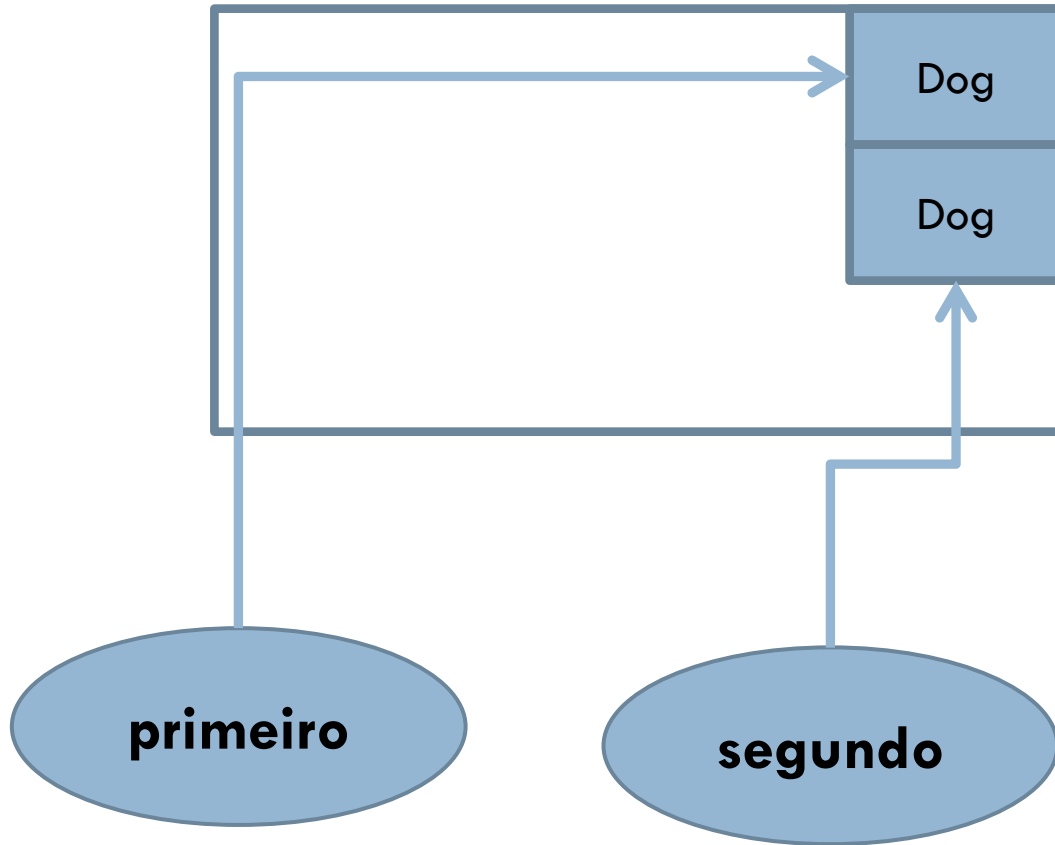
Tenho uma **referência primeiro** a um **objeto** do tipo **Cachorro**.

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

```
Class Teste{  
    public static 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();  
    }  
}
```


Referência

25

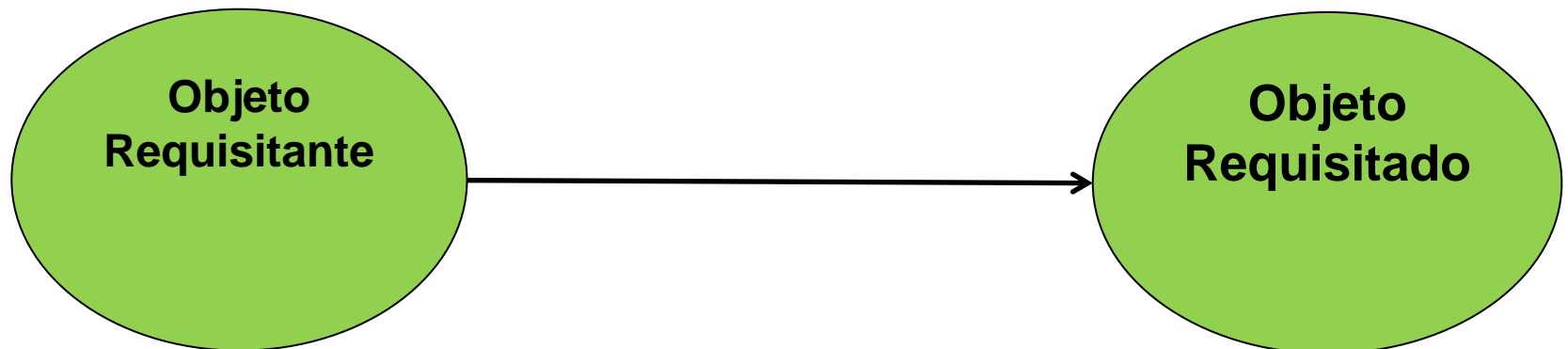


Encapsulamento

- ❑ **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

28

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

29

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.

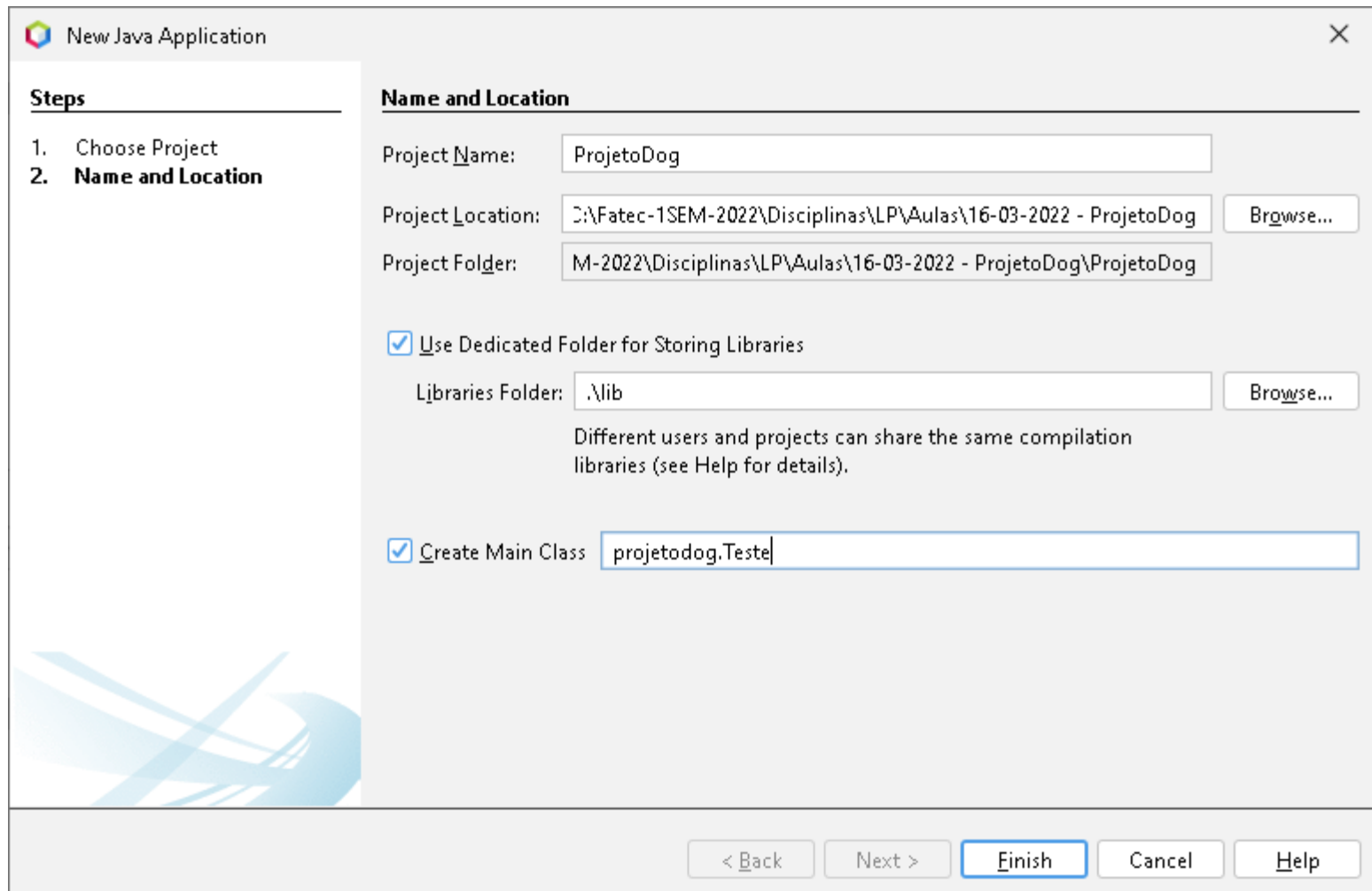
Prática

30

- Criar um novo Projeto Dog.
- Criar uma classe Teste (main)
- Criar uma classe Dog.

Criando o ProjetoDog

31



New Java Application

Steps

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

Name and Location

Project Name:

Project Location:

Project Folder:

☒ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

< Back Next > **Finish** Cancel Help

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

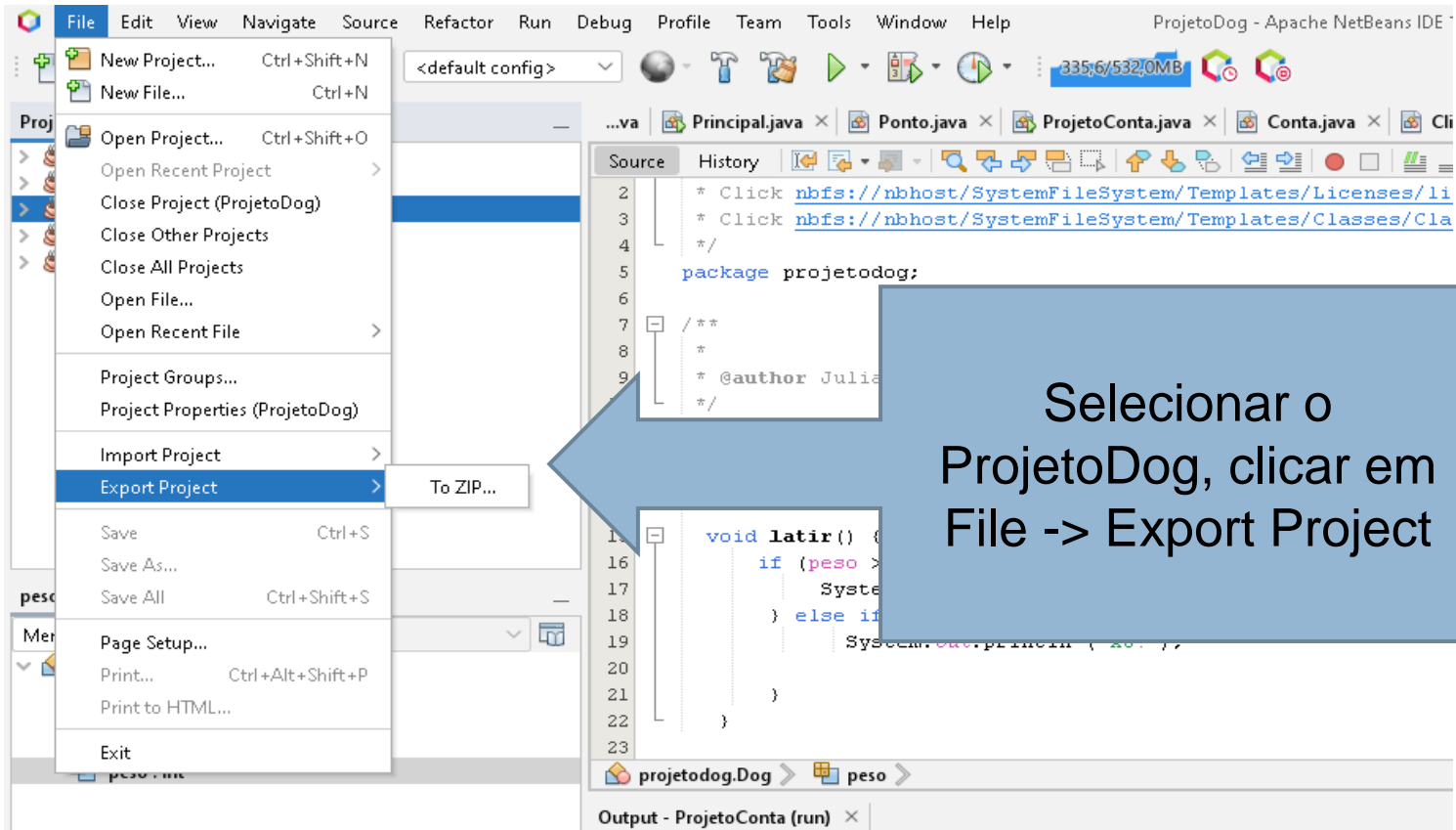
métodos de captura e
configuração públicos

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();  
    }  
}
```

Exportando o ProjetoDog

34



Abstrações

35

- É um modelo que ignora detalhes menos importantes e inclui aspectos importantes (relevantes) e essenciais de alguma coisa;
 - ✓ Abstração permite concentrar a atenção nas características importantes de um objeto;
 - ✓ Tudo depende do contexto: o que é importante em um contexto pode não ser importante em outro contexto.

Quais informações um aluno possui?

Quais informações do aluno são importantes para um sistema de software de uma biblioteca?

Abstrações

36

- **Abstrações:**

- ✓ Abstração adequada permite que o mesmo modelo seja utilizado para:

- ❖ Análise;
 - ❖ Projeto;
 - ❖ Programação;
 - ❖ Projeto de Banco de Dados;
 - ❖ Projeto de Interface;
 - ❖ Documentação.

Exemplo - Conta

37

Conta
+numero: int +saldo: double +limite: double +nome: String
+saca(valor: double): boolean +deposita(valor: double)

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;  
    }  
}
```

```
public class Programa {  
    public static void main(String[] args) {  
        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);  
    }  
}
```

 Problems  @ Javadoc  Declaration  Console 

<terminated> Programa [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (02/04/2013 23:41:16)

Saldo atual: 1000.0

1300.0

```
public class Programa {  
    public static void main(String[] args) {  
        Conta minhaConta=new Conta();  
        minhaConta.nome = "Ana";  
        minhaConta.saldo = 1000;  
        minhaConta.numero=1234;  
        System.out.println("Saldo atual: " + minhaConta.saldo);  
        boolean sacar_ok = minhaConta.saca(2000);  
        if (sacar_ok) {  
            System.out.println("O saque efetuado");  
        } else {  
            System.out.println("O saque não foi efetuado");  
        }  
  
        minhaConta.deposita(500);  
        System.out.println(minhaConta.saldo);  
    }  
}
```

 Problems  @ Javadoc  Declaration  Console 

<terminated> Programa [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (02/04/2013 23:47:07)

Saldo atual: 1000.0

O saque não foi efetuado

1500.0

Exemplo – Classe Conta

41

```
public class teste_referencia {  
  
    public static void main(String args[]) {  
        Conta c1 = new Conta();  
        c1.deposita(300);  
  
        Conta c2 = c1; // linha importante!  
        c2.deposita(400);  
  
        System.out.println(c1.saldo);  
        System.out.println(c2.saldo);  
    }  
}
```

 Problems @ Javadoc  Declaration  Console 

<terminated> teste_referencia [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (03/04/2013 00:07:58)

700.0

700.0

Referência

42

