



Programação Orientada a Objetos I

Prof^a. Angela Abreu Rosa de Sá, Dr^a.

Contato: angelaabreu@gmail.com

Material Didático

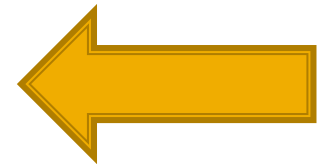
Programação Orientada a Objetos

Sumário

Unidade 1 Fundamentos da orientação a objetos	7
Seção 1.1 - Histórico e introdução à orientação a objetos	9
Seção 1.2 - Conceitos básicos de orientação a objetos	22
Seção 1.3 - Construtores e sobrecarga	37
Unidade 2 Estruturas de programação orientadas a objetos	59
Seção 2.1 - Estruturas de decisão e controle em Java	61
Seção 2.2 - Estruturas de repetição em Java	76
Seção 2.3 - Reutilização de classes em Java	93
Unidade 3 Exceções, classes abstratas e interfaces	111
Seção 3.1 - Definição e tratamento de exceções	113
Seção 3.2 - Definição e uso de classes abstratas	126
Seção 3.3 - Definição e uso de interfaces	141
Unidade 4 Aplicações orientadas a objetos	155
Seção 4.1 - Arrays em Java	157
Seção 4.2 - Strings em Java	173
Seção 4.3 - Coleções e arquivos	188

Conceitos Fundamentais

- **Classe / Objeto**
- **Construtor**
- **Atributos**
- **Métodos**
- **Sobrecarga**
- **Encapsulamento**
- Herança/Generalização/Especialização
- Polimorfismo



Construtores

Construtores são **métodos especiais** que são chamados automaticamente quando instâncias são criadas por meio do **new** – o que causa a execução automática do construtor

Eles garantem que o código contido neles **será executado antes de qualquer outro código em outros métodos**, já que uma instância de uma classe (objeto)

```
public class Calculadora {
```

```
    //atributos
    int numero1;
    int numero2;
```

```
    //construtor
    Calculadora(int n1, int n2)
    {
        numero1 = n1;
        numero2 = n2;
    }
}
```

```
    int numero1, numero2;
    //solicitar que o usuário digite 2 numeros
    System.out.println("\n Digite o primeiro numero: ");
    numero1 = teclado.nextInt();
    System.out.println("\n Digite o segundo numero: ");
    numero2 = teclado.nextInt();
```

```
    Calculadora objetoCalculadora = new Calculadora(numero1, numero2);
```

Sobrecarga - Construtor

```
public class Calculadora {  
  
    int num1;  
    int num2;  
  
    Calculadora()  
    {  
        num1 = 0;  
        num2 = 0;  
    }  
  
    Calculadora(int n1, int n2)  
    {  
        num1 = n1;  
        num2 = n2;  
    }  
  
    Calculadora (int n)  
    {  
        num1 = n;  
        num2 = n;  
    }  
  
}
```

Sobrecarga - Construtor

```
public class Principal01 {  
  
    public static void main(String[] args) {  
  
        Calculadora obj1 = new Calculadora();  
  
        Calculadora obj2 = new Calculadora(2,4);  
  
        Calculadora obj3 = new Calculadora(5);  
  
    }  
  
}
```

Sobrecarga - Métodos

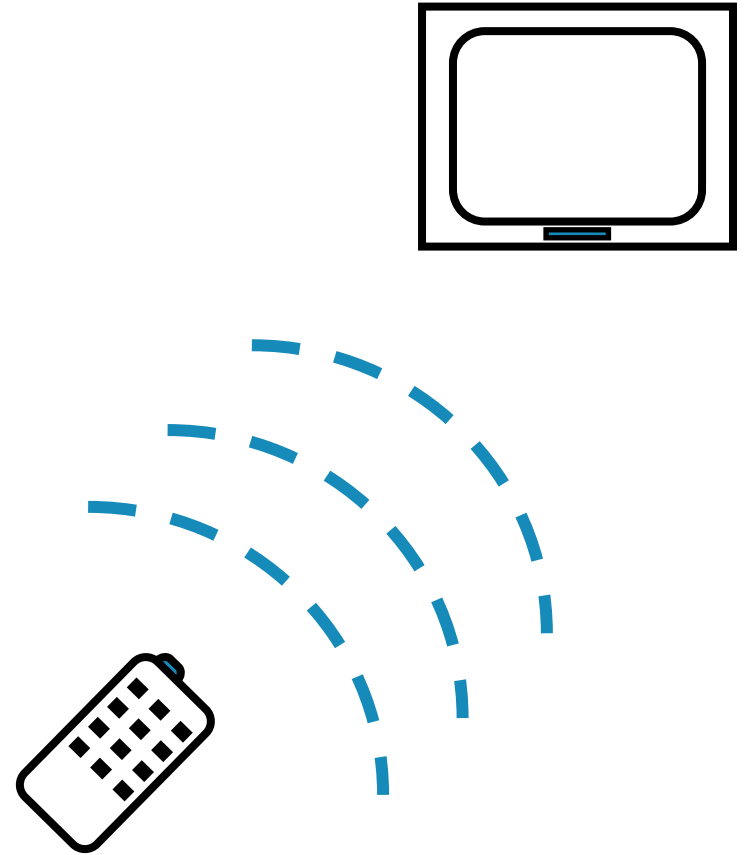
```
public class Calculadora {  
  
    int num1;  
    int num2;  
  
    Calculadora()  
  
    Calculadora(int n1, int n2)  
  
    Calculadora (int n)  
  
    public int Somar()  
    {  
        return num1 + num2;  
    }  
  
    public int Somar(int n1, int n2)  
    {  
        num1 = n1;  
        num2 = n2;  
  
        return num1 + num2;  
    }  
  
}
```

Sobrecarga - Métodos

```
public class Principal01 {  
    public static void main(String[] args) {  
        Calculadora obj1 = new Calculadora();  
        Calculadora obj2 = new Calculadora(2,4);  
        Calculadora obj3 = new Calculadora(5);  
  
        int somatorio = obj2.Somar();  
        int somatorio2 = obj2.Somar(3, 4);  
  
    }  
}
```


Encapsulamento

- Objetos devem “**esconder**” a sua complexidade.
- Legibilidade
- Clareza
- Reuso



Deixar **PÚBLICO** somente o que é necessário para enviar mensagens para o objeto.

Encapsulamento

Ocultação de detalhes internos



Tornar a classe uma **CÁPSULA**, isto é, uma Caixa Preta



Deixar visível somente o **NECESSÁRIO** para a manipulação da classe.

Public
Private

Encapsulamento

```
public class Calculadora {  
  
    public int num1;  
    public int num2;  
  
    Calculadora()  
    {  
        num1 = 0;  
        num2 = 0;  
    }  
  
    Calculadora(int n1,int n2)..  
  
    Calculadora (int n)..  
  
    public int Somar()  
    {  
        return num1 + num2;  
    }  
  
    public int Somar(int n1, int n2)  
    {  
        num1 = n1;  
        num2 = n2;  
  
        return num1 + num2;  
    }  
}
```



```
public class Principal01 {  
  
    public static void main(String[] args) {  
  
        Calculadora obj1 = new Calculadora();  
  
        obj1.num1 = 2;  
        obj1.num2 = 4;  
  
        int somatorio = obj1.Somar();  
  
    }  
}
```

Encapsulamento

```
public class Calculadora {  
  
    private int num1;  
    private int num2;  
  
    Calculadora()  
    {  
        num1 = 0;  
        num2 = 0;  
    }  
  
    Calculadora(int n1, int n2) {  
  
    }  
  
    Calculadora (int n) {  
  
    }  
  
    public int Somar()  
    {  
        return num1 + num2;  
    }  
  
    public int Somar(int n1, int n2)  
    {  
        num1 = n1;  
        num2 = n2;  
  
        return num1 + num2;  
    }  
}
```



```
public class Principal01 {  
  
    public static void main(String[] args) {  
  
        Calculadora obj1 = new Calculadora();  
  
        obj1.num1 = 2;  
        obj1.num2 = 4;  
  
        int somatorio = obj1.Somar();  
    }  
}
```

Problems × @ Javadoc Declaration

Errors, 1 warning, 0 others

Description

Resource

Errors (2 items)

The field Calculadora.num1 is not visible

Principal01.java

The field Calculadora.num2 is not visible

Principal01.java

Warnings (1 item)

Build path specifies execution environment JavaSE- Ex01

Atributos **não VISÍVEIS** para
outras classes. **Não podem ser**
acessados diretamente!

Encapsulamento

```
public class Calculadora {
```

```
    private int num1;  
    private int num2;
```

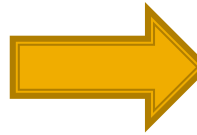
```
    Calculadora()  
    {  
        num1 = 0;  
        num2 = 0;  
    }
```

```
    Calculadora(int n1, int n2) {
```

```
    Calculadora (int n) {
```

```
    public int Somar()  
    {  
        return num1 + num2;  
    }
```

```
    public int Somar(int n1, int n2)  
    {  
        num1 = n1;  
        num2 = n2;  
  
        return num1 + num2;  
    }
```



```
public class Principal01 {
```

```
    public static void main(String[] args) {
```

```
        Calculadora obj1 = new Calculadora(2,4);
```

```
        int somatorio = obj1.Somar();
```

```
        //outra opcao
```

```
        Calculadora obj2 = new Calculadora();
```

```
        int somatorio2 = obj2.Somar(2, 4);
```

1

Atributos **PRIVATE** só serão
acessados e alterados do
métodos **PUBLIC**

Encapsulamento

```
public class Calculadora {  
  
    private int num1;  
    private int num2;  
  
    Calculadora(int n1, int n2)  
    {  
        num1 = n1;  
        num2 = n2;  
    }  
  
    private int Somar()  
    {  
        return num1 + num2;  
    }  
  
    public double Media()  
    {  
        int soma = Somar();  
  
        double media = soma/2;  
  
        return soma;  
    }  
}
```

Os atributos são **PRIVADOS**...

E se precisar alterar o conteúdo deles depois da criação do objeto?

E se precisar saber qual é o conteúdo destes atributos?

```
public class Calculadora {
```

```
    private int num1;  
    private int num2;
```

```
    Calculadora(int n1,int n2)  
    {  
        num1 = n1;  
        num2 = n2;  
    }
```

```
    private int Somar()  
    {  
        return num1 + num2;  
    }
```

```
    public double Media()  
    {  
        int soma = Somar();  
  
        double media = soma/2;  
  
        return soma;  
    }
```

```
    public void AlterarNumeros(int n1, int n2)  
    {  
        num1 = n1;  
        num2 = n2;  
        return;  
    }
```

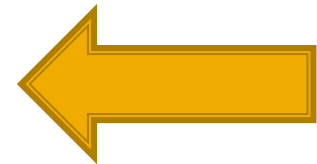
```
    public int RetornaNumero1()  
    {  
        return num1;  
    }
```

```
    public int RetornaNumero2()  
    {  
        return num2;  
    }
```

Métodos **PÚBLICOS** do tipo “**GET**
e SET” para acessar e retornar o
conteúdo de atributos que são
PRIVATE.

Conceitos Fundamentais

- **Classe / Objeto**
- **Construtor**
- **Atributos**
- **Métodos**
- **Sobrecarga**
- **Encapsulamento**
- Herança/Generalização/Especialização
- Polimorfismo

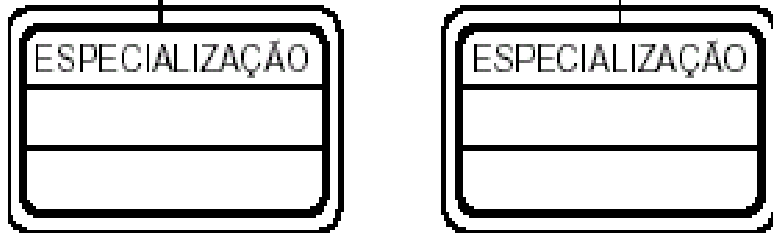


Herança

“ É um mecanismo para derivar novas classes a partir de classes existentes”

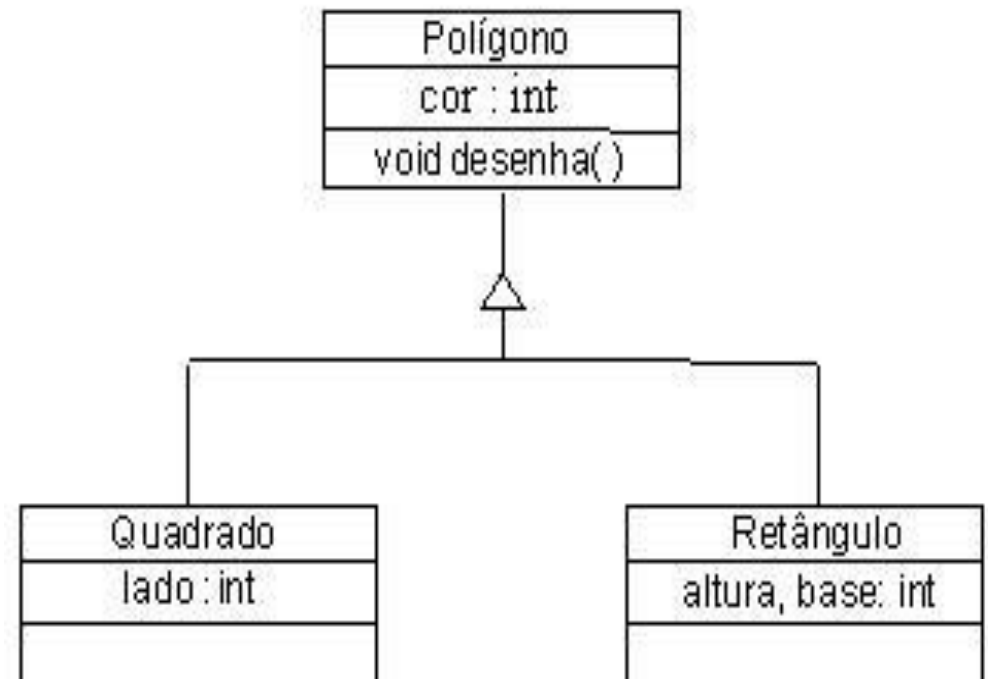
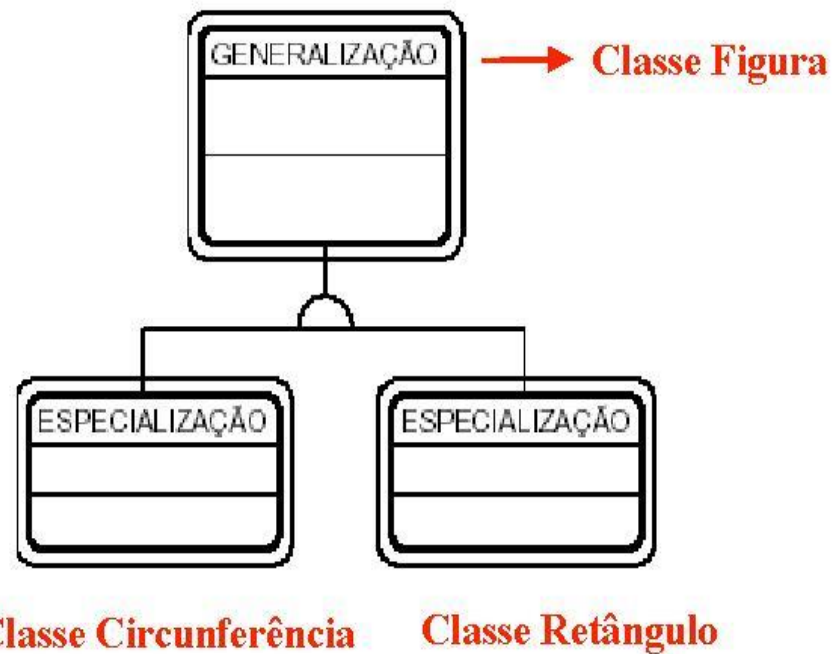


Superclasse ou classe pai



Subclasse ou classe filha

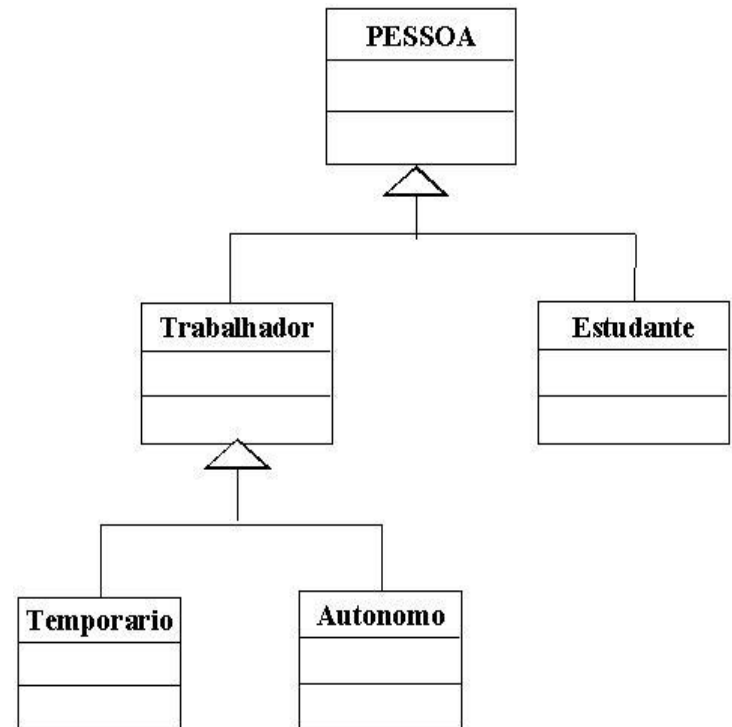
Herança



Herança

- Redução no custo de manutenção;
- Aumento na reutilização de código ;
- Modularização;

Vantagens



SEM Herança

Funcionário

Nome, cargo, data
de admissão

```
public class Funcionario {  
  
    String nome;  
    String cpf;  
    double salario;  
  
}
```

Gerente

Nome, cargo, data
de admissão, Quantidade
de funcionários que
gerencia, senha de acesso.

```
public class Gerente {  
  
    String nome;  
    String cpf;  
    double salario;  
    int senha;  
    int numeroDeFuncionariosGerenciados;  
  
}
```

Herança

Funcionário

Nome, cargo, data
de admissão

```
public class Funcionario {  
  
    String nome;  
    String cpf;  
    double salario;  
  
}
```

Gerente

Nome, cargo, data
de admissão, Quantidade
de funcionários que
gerencia, senha de acesso.

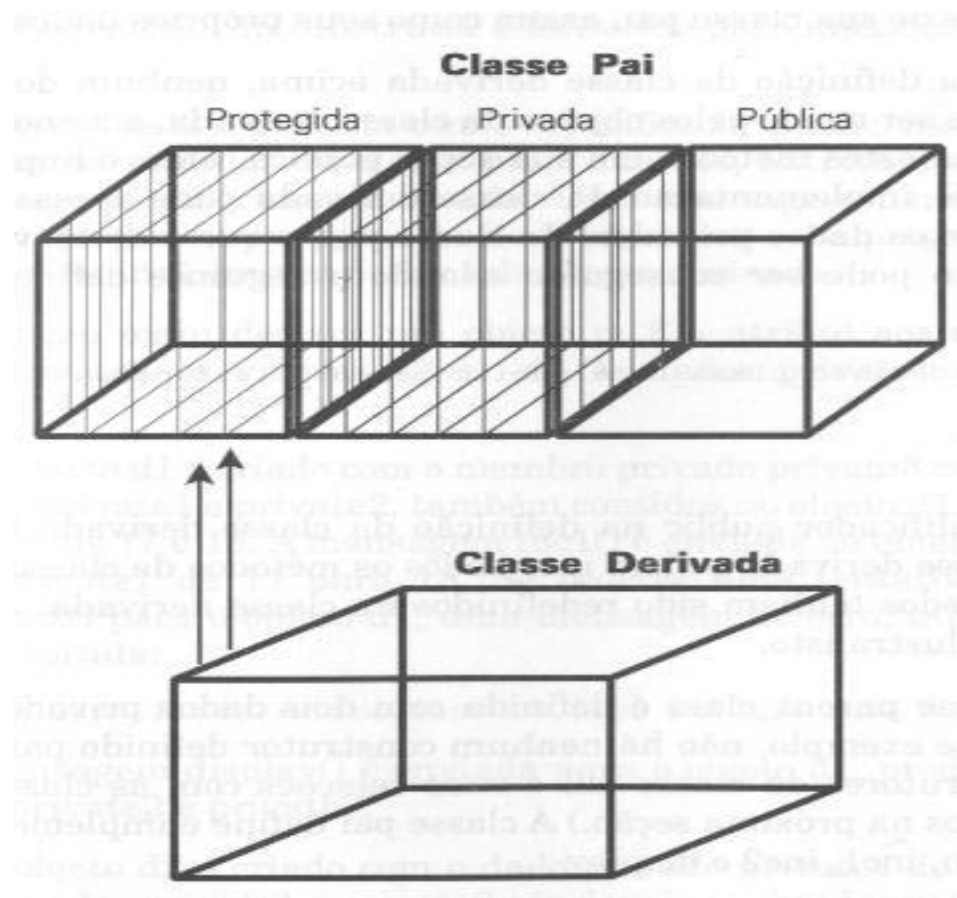
```
public class Gerente extends Funcionario {  
  
    int senha;  
    int numeroDeFuncionariosGerenciados;  
  
}
```

Herança

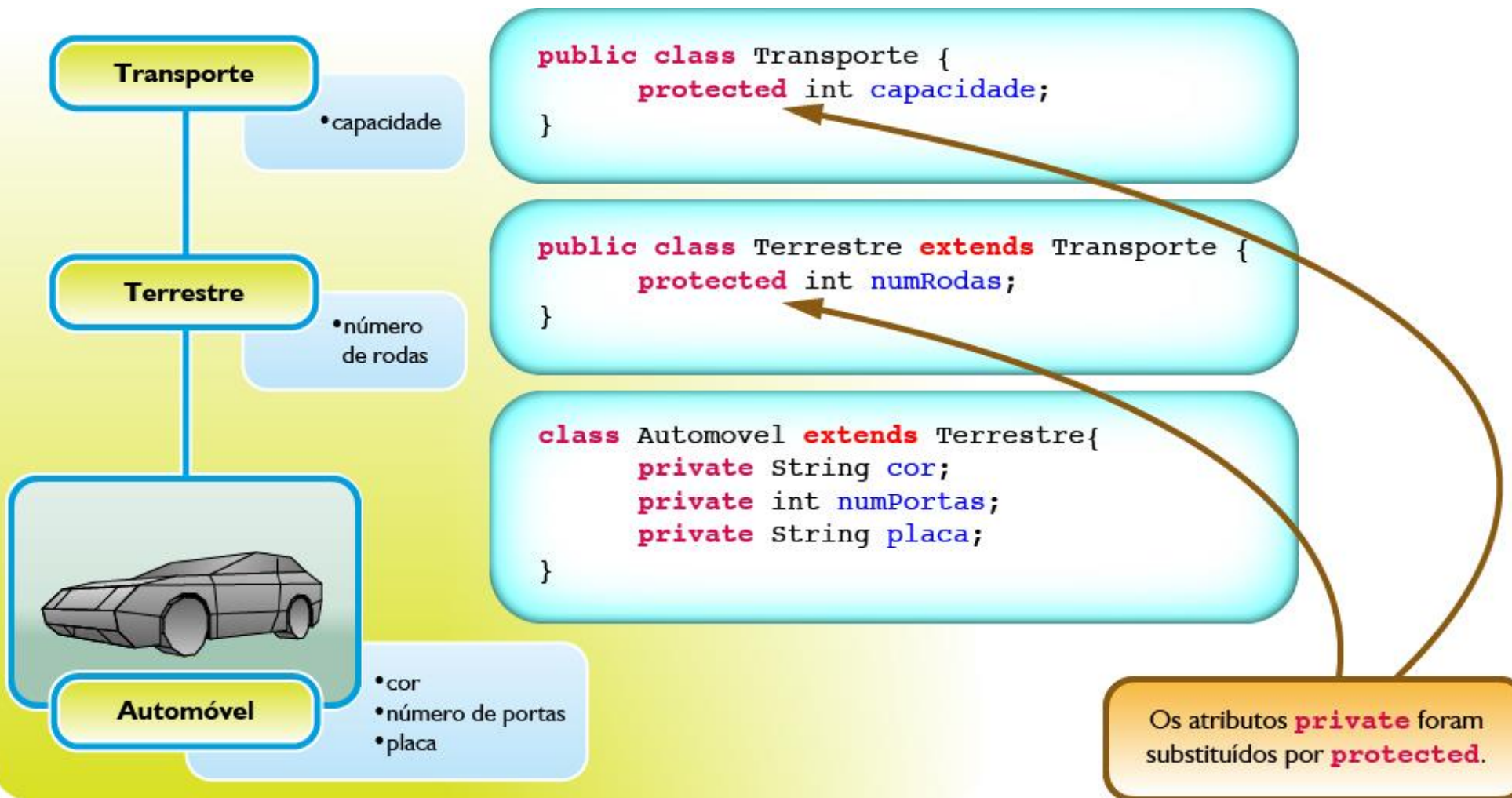
Tipo de acesso	Descrição
Public	Estes atributos e métodos são sempre acessíveis em todos os métodos de todas as classes. Este é o nível menos rígido de encapsulamento, que equivale a não encapsular.
Private	Esses atributos e métodos são acessíveis somente nos métodos da própria classe. Este é o nível mais rígido de encapsulamento.
<u>Protected</u>	Esses atributos e métodos são acessíveis nos métodos da própria classe e suas subclasses.

Herança

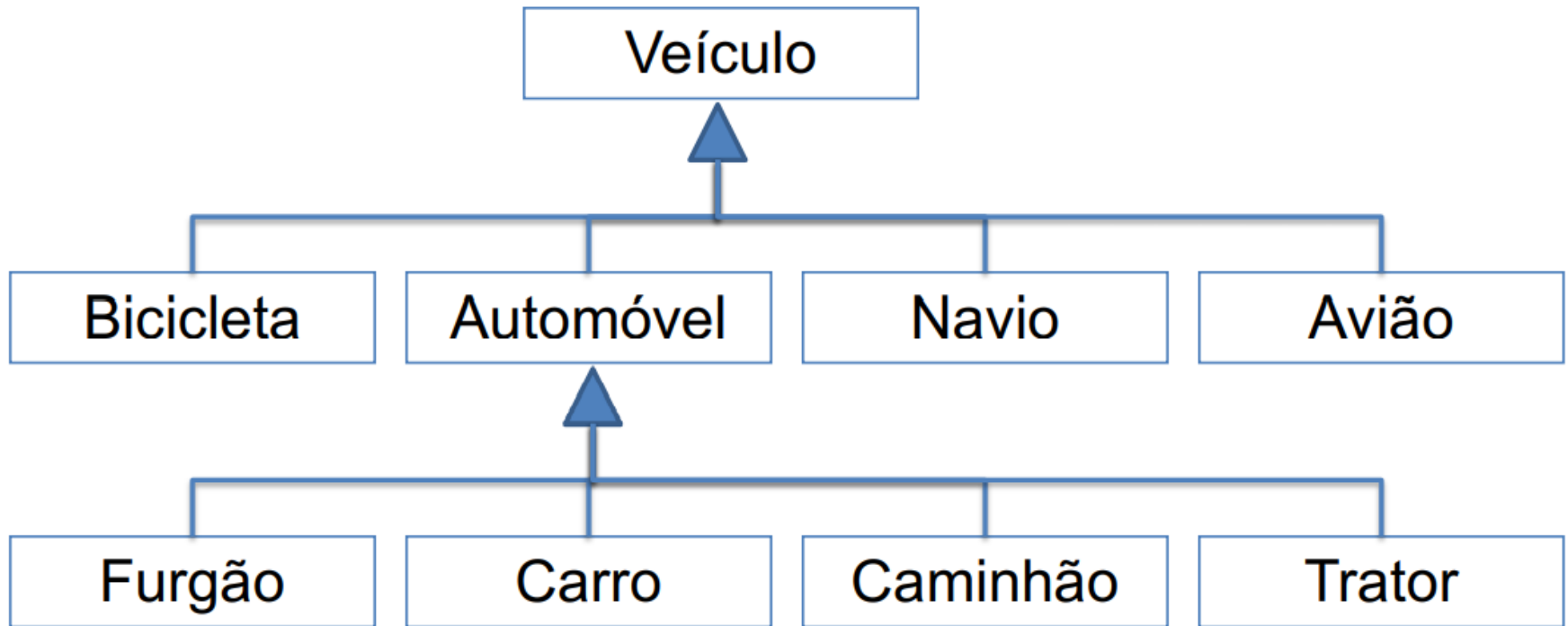
Tipos de acesso



Herança



Herança



```
public class Funcionario {  
  
    protected String nome;  
    protected String cpf;  
    protected double salario;  
  
    public Funcionario(String n, String numcpf, double sal)  
    { //inicializar os atributos  
        nome = n;  
        cpf = numcpf;  
        salario = sal;  
    }  
}
```

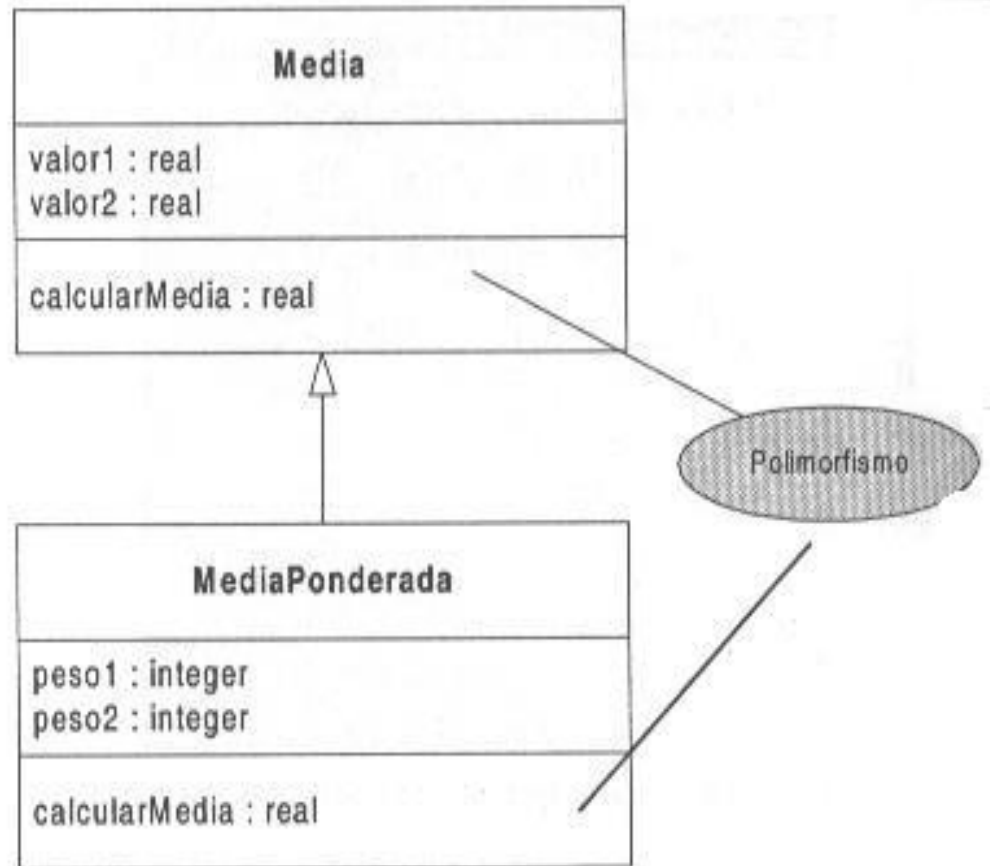
```
public class Gerente extends Funcionario {  
  
    private int senha;  
    private int numeroDeFuncionariosGerenciados;  
  
    public Gerente (String n, String numcpf, double sal, int s, int numgerenciado)  
    {  
        super(n,numcpf,sal); //chamando o construtor da super classe (classe pai)  
        senha = s;  
        numeroDeFuncionariosGerenciados = numgerenciado;  
    }  
}
```

Polimorfismo

poli = muitas,
morphos = formas

POLIMORFISMO

MUITAS FORMAS



```
public class Calculadora {
```

```
    private int num1;  
    private int num2;
```

```
    Calculadora(int n1,int n2)  
    {  
        num1 = n1;  
        num2 = n2;  
    }
```

```
    private int Somar()  
    {  
        return num1 + num2;  
    }
```

```
    public double Media()  
    {  
        int soma = Somar();  
  
        double media = soma/2;  
  
        return soma;  
    }
```

```
    public void AlterarNumeros(int n1, int n2)  
    {  
        num1 = n1;  
        num2 = n2;  
        return;  
    }
```

```
    public int RetornaNumero1()  
    {  
        return num1;  
    }
```

```
    public int RetornaNumero2()  
    {  
        return num2;  
    }
```

Métodos **PÚBLICOS** do tipo “**GET**
e SET” para acessar e retornar o
conteúdo de atributos que são
PRIVATE OU PROTECTED.

```
public class Funcionario {
```

```
    protected String nome;
```

```
}
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

Toggle Comment Ctrl+7

Remove Block Comment Ctrl+Shift+\

Generate Element Comment Alt+Shift+J

Correct Indentation Ctrl+I

Format Ctrl+Shift+F

Format Element

Add Import Ctrl+Shift+M

Organize Imports Ctrl+Shift+O

Sort Members...

Clean Up...

Override/Implement Methods...

Generate Getters and Setters...

Generate Delegate Methods...

Generate hashCode() and equals()...

Generate toString()...

Generate Constructor using Fields...

Generate Constructors from Superclass...

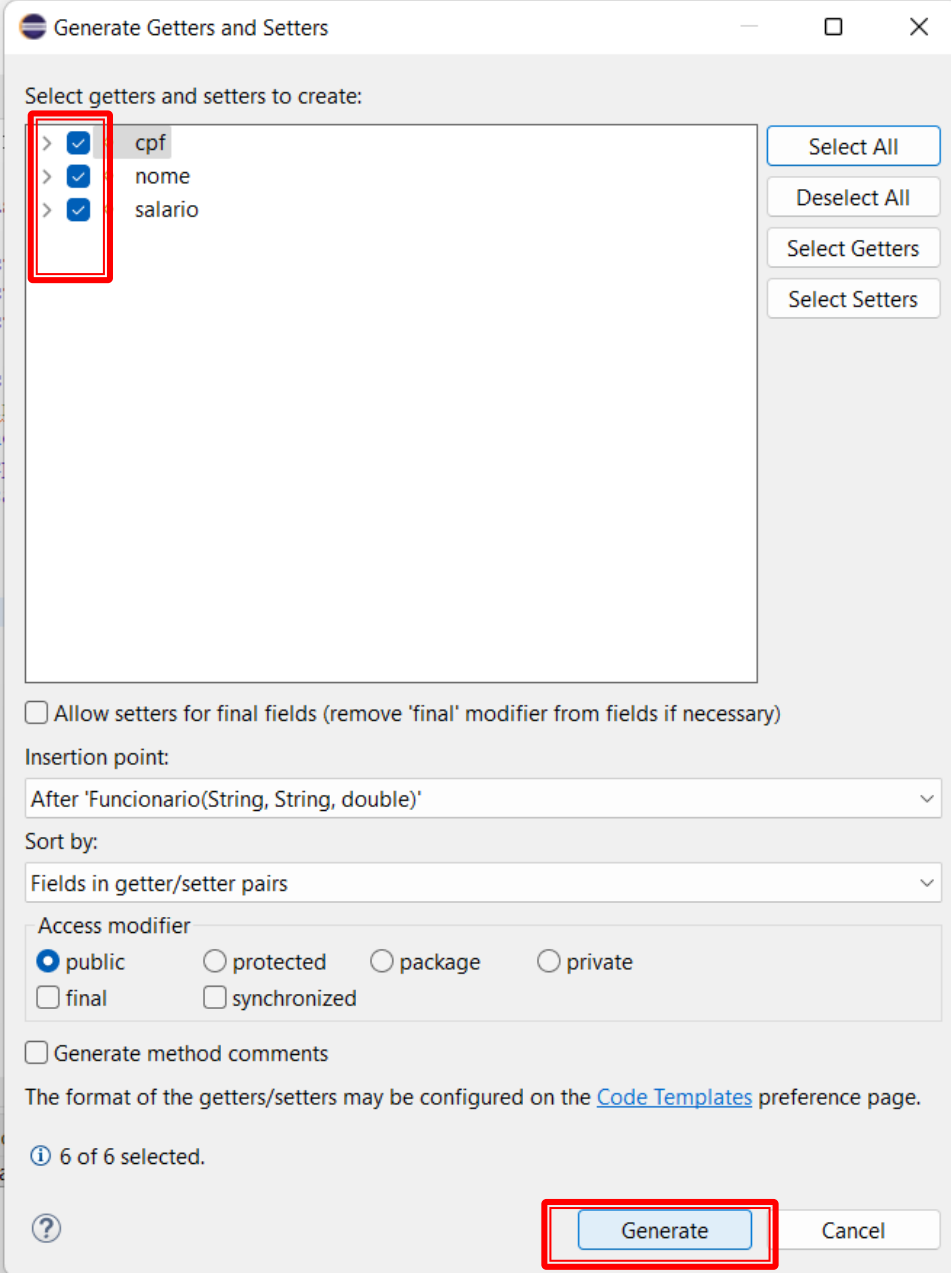
Externalize Strings...

jdk-18.0.1.1\bin\javaw.exe (24 de ago. de 2022 11:50:33 – 11:50:35) [pid: 19364]

Writable

Smart Insert

16 : 5 : 270



```
public class Funcionario {  
  
    protected String nome;  
    protected String cpf;  
    protected double salario;  
  
    public Funcionario(String n, String numcpf, double sal)  
    { //inicializar os atributos  
        nome = n;  
        cpf = numcpf;  
        salario = sal;  
    }  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    public String getCpf() {  
        return cpf;  
    }  
  
    public void setCpf(String cpf) {  
        this.cpf = cpf;  
    }  
  
    public double getSalario() {  
        return salario;  
    }  
  
    public void setSalario(double salario) {  
        this.salario = salario;  
    }  
}
```

this

Orientação a objetos

Encapsulamento

Herança

Composição

Polimorfismo

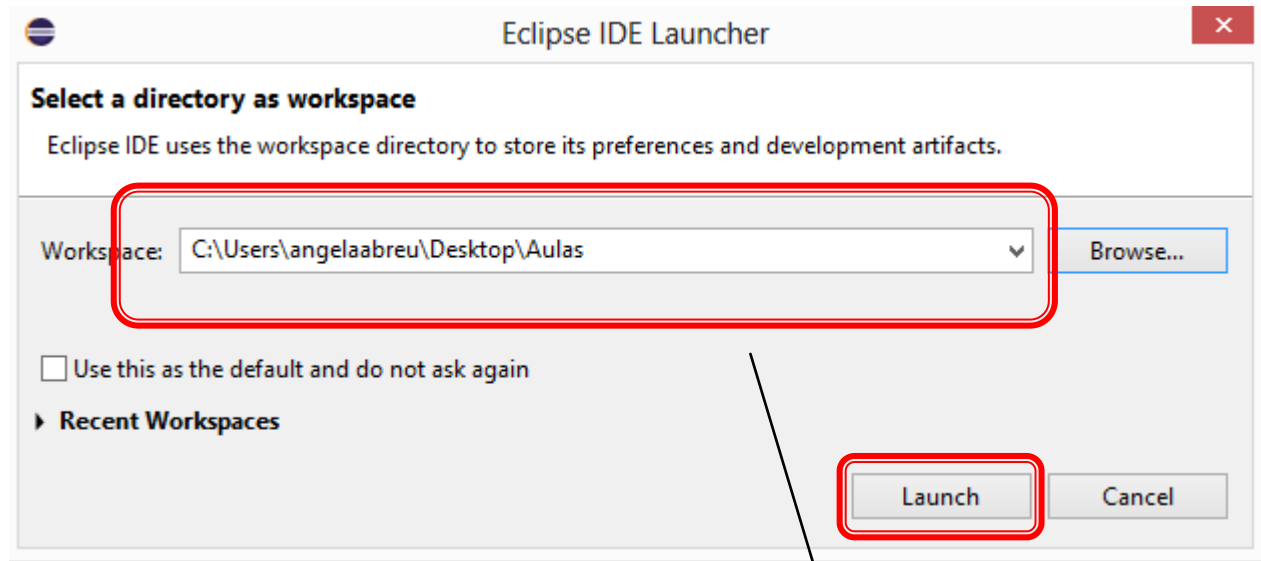
Princípio da abstração

Exercícios 😊

eclipse IDE for Java Developers





Orientação a Objetos



Diretório que irá armazenar o seu projeto.

Para cada novo programa, **CRIEM uma nova pasta** para armazenar o projeto.

File Edit Source Refactor Navigate Search Project Run Window Help

New **Alt+Shift+N**  **Java Project** 

Open File...

Open Projects from File System...

Recent Files

Close Editor **Ctrl+W**

Close All Editors **Ctrl+Shift+W**

Save **Ctrl+S**

Save As...

Save All **Ctrl+Shift+S**

Revert

Move...

Rename... **F2**

Refresh **F5**

Convert Line Delimiters To

Print... **Ctrl+P**

Import...

Export...

Properties **Alt+Enter**

Switch Workspace

Restart

Exit

Create a Java project

Package

Class

Interface

Enum

Record

Annotation

Source Folder

Java Working Set

Folder

File

Untitled Text File

JUnit Test Case

Other... **Ctrl+N**

Problems x @ Javadoc Declaration

0 items

Description	Resource	Path	Location	Type

Exercício 1

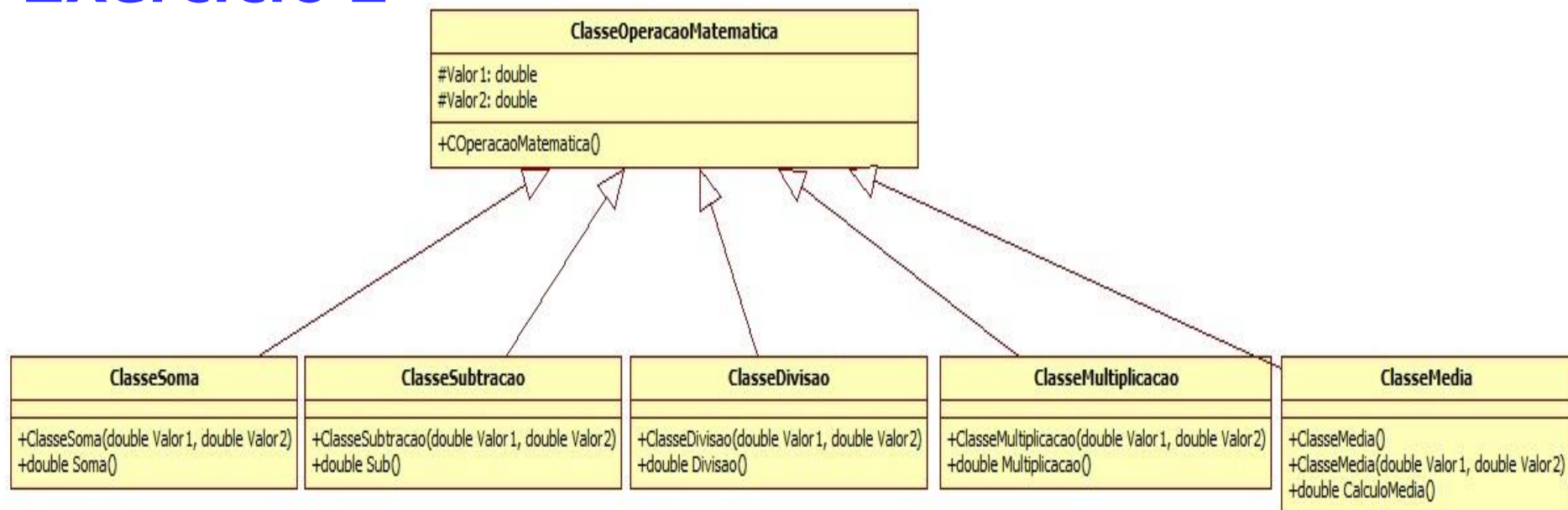
```
public class Pessoa {  
    //atributos  
    protected String nome;  
  
    //construtor  
    public Pessoa(String nome)  
    {  
        this.nome = nome;  
    }  
    //métodos  
    public String getNome() {  
        return nome;  
    }  
}
```

```
public class PessoaFisica extends Pessoa {  
    //atributos  
    private long cpf;  
  
    //construtor  
    public PessoaFisica(String nome, long cpf) {  
  
        super(nome); //chamando o construtor da classe pai  
        this.cpf = cpf;  
    }  
  
    //métodos  
    public String getNome() {  
        return "Pessoa Fisica: " + super.getNome() + " - CPF: " + cpf;  
    }  
}
```

```
public class PessoaJuridica extends Pessoa {  
    //atributos  
    private long cnpj;  
    //construtor  
    public PessoaJuridica(String nome, long cnpj) {  
  
        super(nome); //chamando o construtor da classe pai  
        this.cnpj = cnpj;  
    }  
  
    //métodos  
    public String getNome() {  
        return "Pessoa Juridica: " + super.getNome() + " - CNPJ: " + cnpj;  
    }  
}
```

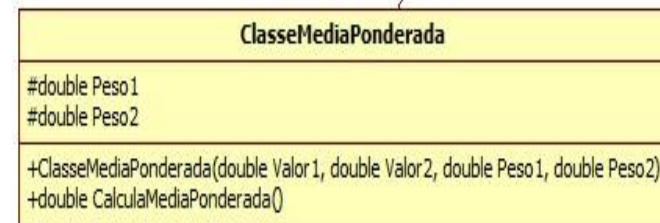
```
public class Principal {  
  
    public static void main(String[] args) {  
  
        //instanciar os objetos  
        PessoaFisica objFisica = new PessoaFisica("Ana Santos", 01452263562);  
        PessoaJuridica objJuridica = new PessoaJuridica("Robson Silva", 888888);  
  
        System.out.println("**** " + objFisica.getNome());  
        System.out.println("**** " + objJuridica.getNome());  
  
    }  
  
}
```


Exercício 2



No programa principal, o usuário deve informar qual é a operação matemática que será executada:

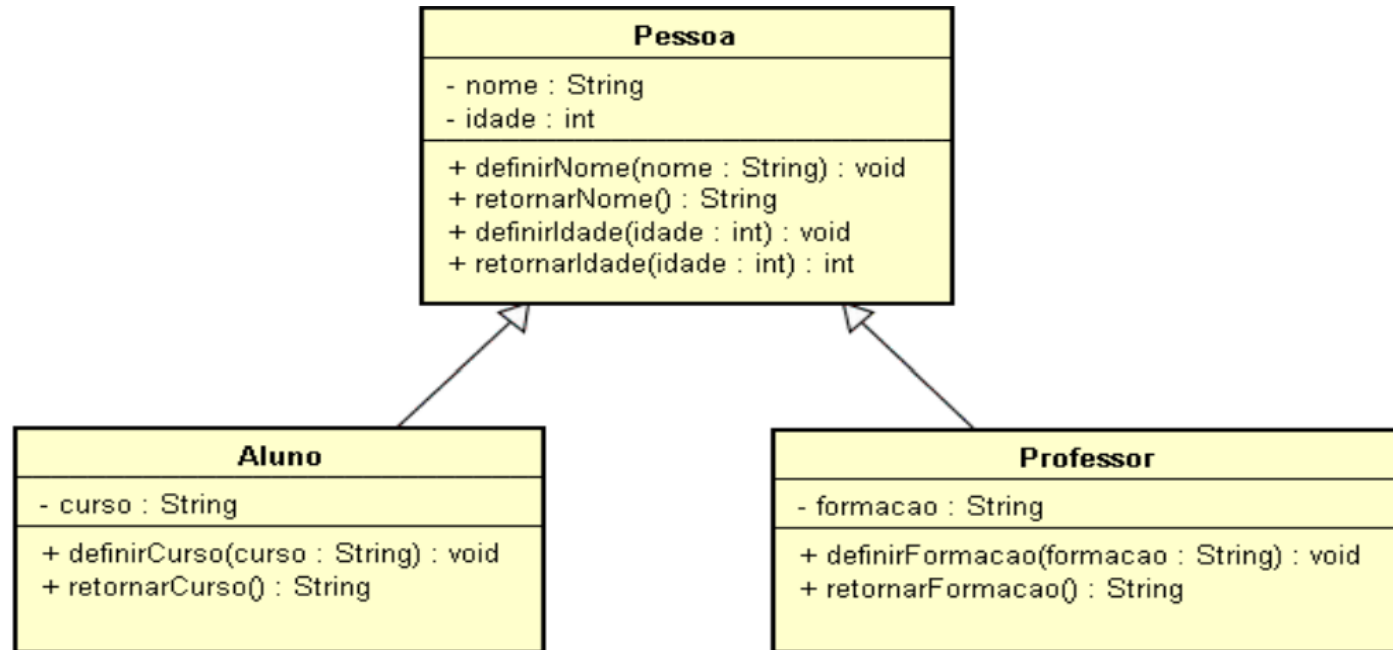
- 1 – Soma
- 2 – Subtração
- 3 – Divisão
- 4 – Multiplicação
- 5 – Média
- 6 – Média ponderada



Em seguida, deve informar 2 números para executar a operação.

Obs. Se for selecionado a Média ponderada, deverá informar o peso de cada número.

Exercício 3



No programa principal, o usuário deve informar os dados dos alunos e professores que serão cadastrados. Em seguida, o programa deve mostrar a formação dos professores e o curso dos Alunos.

Sugestão: utilizar vetor de objetos no programa principal



Muito Obrigada!

Prof^a. Angela Abreu Rosa de Sá, Dr^a.

Contato: angelaabreu@gmail.com