



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

- Uma classe é um molde para objetos.
- Um objeto é uma instância de uma classe.
- Uma classe é uma *abstração* das características *relevantes* de um grupo de coisas do mundo real.

Lembrar....

Classes: estruturas que contém os dados que devem ser representados e as operações que devem ser efetuadas com esses dados para determinado modelo.

Objetos: é como chamamos a **materialização da classe**, que assim poderá ser usada para representar dados e executar operações.

Obs.: podemos ter vários objetos de uma única classe.

Atributos: são as propriedades (ou dados) da classe.
(“variáveis”)

Métodos: os métodos expressam as funcionalidades da classe
(“funções”)

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;
    }
}
```

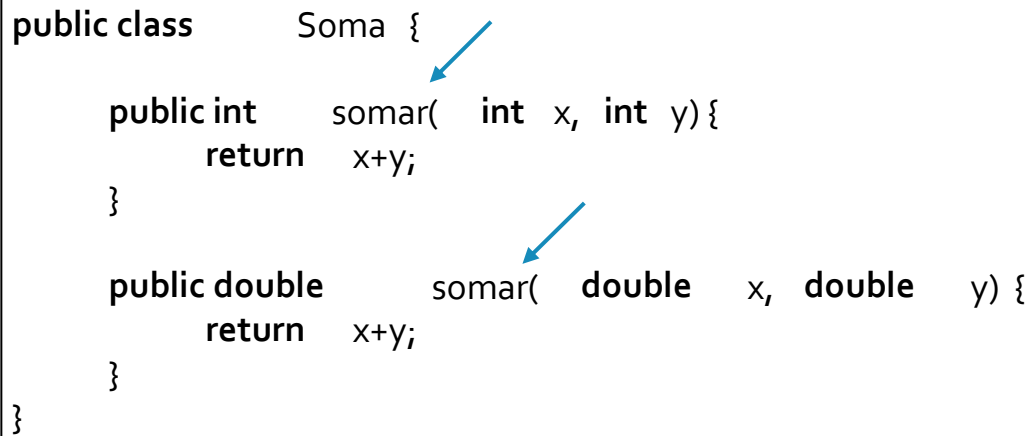
```
    int numero1, numero2;
    //solicitar que o usuário digite 2 numeros
    System.out.println("\n Digite o primeiro numero: ");
    numero1 = teclado.nextInt();
    System.out.println("\n Digite o segundo numero: ");
    numero2 = teclado.nextInt();
```

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

Sobrecarga (overload)

- Permite a um método **dentro da mesma classe** ter **várias implementações** as quais são diferenciadas com base na **quantidade e tipo de parâmetros**

```
public class Soma {  
    public int somar( int x, int y) {  
        return x+y;  
    }  
    public double somar( double x, double y) {  
        return x+y;  
    }  
}
```



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 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

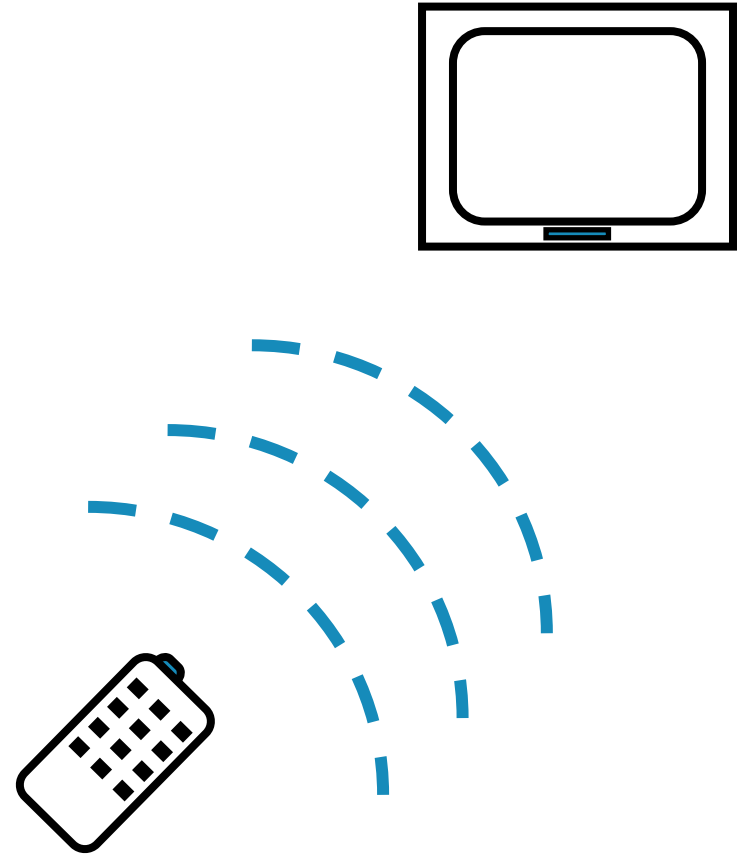
```
public class Calculadora {  
  
    int num1;  
    int num2;  
  
    Calculadora()  
  
    Calculadora(int n1, int n2)  
  
    Calculadora (int n)  
  
    public int Somar()  
    {  
        return num1 + num2;  
    }  
  
    public int Somar(int n1, int n2)  
    {  
        num1 = n1; 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

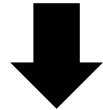
Encapsulamento

VISIBILIDADE da classe

Public

Private

Modificadores de Acesso



Quem pode “ver” e acessar o conteúdo de cada atributo e método?
Somente a classe ou o programa principal?

Encapsulamento

Tipos de Visibilidade

■ Público **Public**

Os Atributos e Métodos que PÚBLICOS, são **visíveis** para o **programa** que está utilizando a classe

■ Privado **Private**

Os Atributos e Métodos que PRIVATE, são **visíveis** para apenas **DENTRO da classe**

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.


```

1 public class RegistroAcademico {
2
3     //Atributos COM ENCAPSULAMENTO
4     private String nomeDoAluno; //nome do aluno
5     private int numeroDeMatricula; //número de matrícula
6     private int códigoDoCurso; //código do curso (1 .. 4)
7     private double percentualDeCobranca; //percentual em relação ao preço cheio, de 0 a 100%
8
9     //construtor
10    RegistroAcademico(String n, int m, int c, double p){
11        nomeDoAluno = n;
12        numeroDeMatricula = m;
13        códigoDoCurso = c;
14        percentualDeCobranca = p;
15    }
16    //Metodo
17    public double calculaMensalidade() {
18        double mensalidade = 0;
19        //determinação do valor cheio da mensalidade, dependendo do curso.
20
21        if (códigoDoCurso == 1) // Redes
22            mensalidade = 450.00;
23        if (códigoDoCurso == 2) // Ciência da Computação
24            mensalidade = 500.00;
25        if (códigoDoCurso == 3) // Engenharia da Computação
26            mensalidade = 550.00;
27        if (códigoDoCurso == 4) // Sistemas de Informacao
28            mensalidade = 380.00;
29
30        //calcula o desconto. Se o percentual for zero, a mensalidade também o será.
31        if (percentualDeCobranca == 1)
32            mensalidade = 450.00;
33        else mensalidade = mensalidade * 100.0 / percentualDeCobranca;
34
35        return mensalidade;
36    }
37 }

```

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;
```

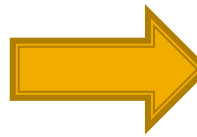
```
    Calculadora()  
    {  
        num1 = 0;  
        num2 = 0;  
    }
```

```
    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;  
    }
```

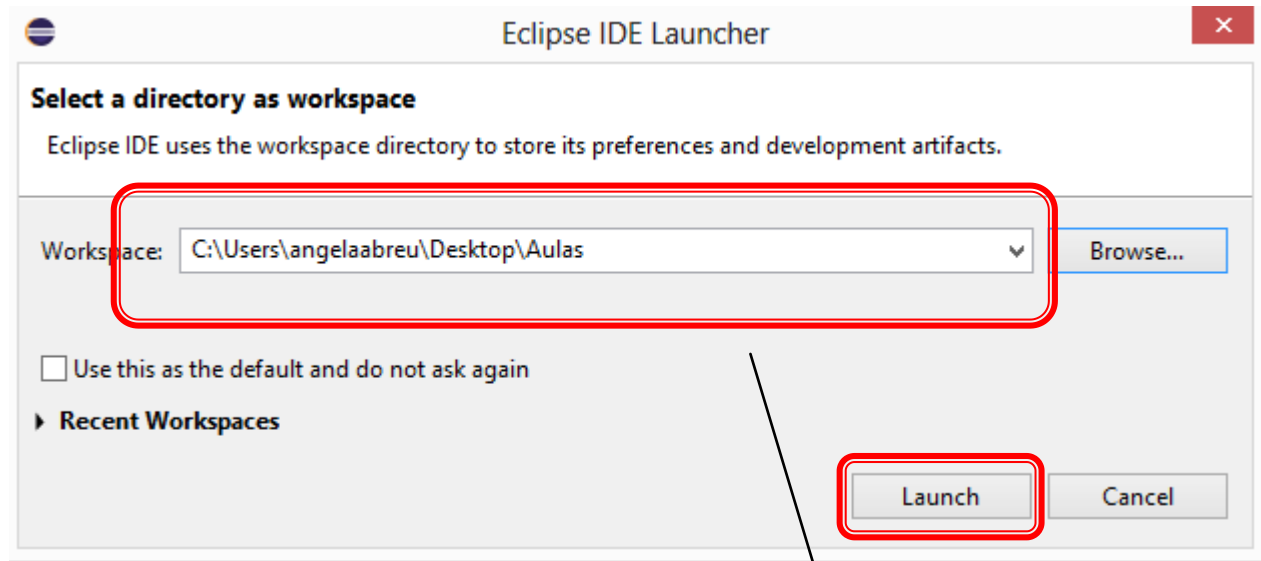
```
    public int RetornaNumero2()  
    {  
        return num2;  
    }
```

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

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.



Welcome to the Eclipse IDE for Java Developers

**Review IDE configuration settings**

Review the IDE's most fiercely contested preferences

**Create a Hello World application**

A guided walkthrough to create the famous Hello World in Eclipse

**Create a new Java project**

Create a new Java Eclipse project

**Checkout projects from Git**

Checkout Eclipse projects hosted in a Git repository

**Import existing projects**

Import existing Eclipse projects from the filesystem or archive

**Launch the Eclipse Marketplace**

Enhance your IDE with additional plugins and install your Marketplace favorites

**Overview**

Get an overview of the features

**Tutorials**

Go through tutorials

**Samples**

Try out the samples

**What's New**

Find out what is new

File Edit Source Refactor Navigate Search Project Run Window Help

New **Alt+Shift+N** ▸

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

Java Project

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 × @ Javadoc Declaration

0 items

Description	Resource	Path	Location	Type

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location

Location:

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE 'jdk-18' and workspace compiler preferences

[Configure...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files

[Configure d...](#)

Working sets

☐ Add project to working sets

Working sets:





Module


☒ Create module-info.java file




< Back

Next >

New module-info.java

Create module-info.java

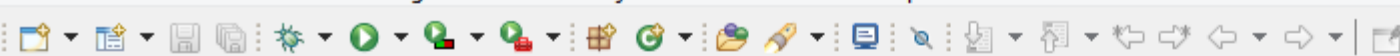
 Discouraged module name. By convention, module names usually start with a lowercase letter

Module name:

☐ Generate comments (configure templates and default value [here](#))

Create

Don't Create



Package Explorer

Package Explorer

New

Go Into

Open in New Window

Open Type Hierarchy F4

Show In Alt+Shift+W

Copy Ctrl+C

Copy Qualified Name

Paste Ctrl+V

Delete Delete

Build Path

Source Alt+Shift+S

Refactor Alt+Shift+T

Import...

Export...

Refresh F5

Close Project

Assign Working Sets...

Coverage As

Run As

Debug As

Restore from Local History...

Team

Compare With

Configure

Java Project

Project...

Package

Class Create a Java class

Interface

Enum

Record

Annotation

Source Folder

Java Working Set

Folder

File

Untitled Text File

JUnit Test Case

Other... Ctrl+N

Problems

@ Javadoc

Declaration

Errors, 1 warning, 0 others

Description

Resource

Path

Location

T

Warnings (1 item)

```
public class Aluno {  
  
    private String nome;  
    private float notal;  
    private float nota2;  
    private float nota3;  
    private float media;  
  
    public Aluno(String n, float n1, float n2, float n3)  
    {  
        nome = n;  
        notal = n1;  
        nota2 = n2;  
        nota3 = n3;  
        media = 0;  
    }  
  
    private float Somar()  
    {  
        float soma = notal+nota2+nota3;  
        return soma;  
    }  
  
    private float CalculaMedia()  
    {  
        media = Somar()/3;  
        return media;  
    }  
  
    public String ResultadoFinal()  
    {  
        String resultado = "";  
        float media = CalculaMedia();  
        if( media >= 6)  
            resultado = "Aprovado";  
        else if ((media >= 4) && (media < 6))  
            resultado = "Recuperação";  
        else resultado = "Reprovado";  
  
        return resultado;  
    }  
}
```

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

Java Class

⚠ The use of the default package is discouraged.

Source folder: Aula01/src

Browse...

Package:

(default)

Browse...

☐ Enclosing type: Triangulo

Browse...

Name:

Principal

Modifiers:

☒ public ☐ package ☐ private ☐ protected☐ abstract ☐ final ☐ static☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass:

java.lang.Object

Browse...

Interfaces:

Add...

Remove

Which method stubs would you like to create?

☒ public static void main(String[] args)☐ Constructors from superclass☒ Inherited abstract methodsDo you want to add comments? (Configure templates and default value [here](#))☐ Generate comments

Finish

Cancel

```

import java.util.Scanner;

public class Principal01 {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String nome;
        int nota1, nota2, nota3;

        System.out.println("***** Dados do Aluno ***** ");
        System.out.print("Digite o nome do aluno: ");
        nome = scan.next();
        System.out.print("Digite a nota 1: ");
        nota1 = scan.nextInt();
        System.out.print("Digite a nota 2: ");
        nota2 = scan.nextInt();
        System.out.print("Digite a nota 3: ");
        nota3 = scan.nextInt();

        Aluno objetoAluno; //declaração do objeto
        objetoAluno = new Aluno(nome, nota1,nota2,nota3);

        System.out.println("***** Boletim final ***** ");
        System.out.println("Aluno " + objetoAluno.RetornaNome() + " está " + objetoAluno.ResultadoFinal());

    }
}

```

<terminated> Principal01 [Java Application] C:\Program Files\

***** Dados do Aluno *****

Digite o nome do aluno: XXXX

Digite a nota 1: 8

Digite a nota 2: 8

Digite a nota 3: 8

***** Boletim final *****

Aluno XXXX está Aprovado


```

public class Principal01 {
    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);
        Aluno []vetorObjetosAluno; //VETOR DE OBJETOS ALUNO

        System.out.print("Digite a quantidade de alunos: ");
        int qtdeAlunos = scan.nextInt();

        vetorObjetosAluno = new Aluno[qtdeAlunos]; //CRIAR O VETOR DE OBJETOS

        String nome;
        int nota1, nota2, nota3;

        for (int i = 0; i < qtdeAlunos; i++)
        {
            System.out.println("***** Dados do Aluno ***** ");
            System.out.print("Digite o nome do aluno: ");
            nome = scan.next();
            System.out.print("Digite a nota 1: ");
            nota1 = scan.nextInt();
            System.out.print("Digite a nota 2: ");
            nota2 = scan.nextInt();
            System.out.print("Digite a nota 3: ");
            nota3 = scan.nextInt();

            //INSTANCIAR O OBJETO PARA CADA POSIÇÃO DO VETOR
            vetorObjetosAluno[i] = new Aluno(nome, nota1,nota2,nota3);
        }

        System.out.println("***** Boletim final ***** ");
        for (int i = 0; i < qtdeAlunos; i++)
        {
            System.out.println("Aluno " + vetorObjetosAluno[i].RetornaNome() + " está " + vetorObjetosAluno[i].ResultadoFinal());
        }
    }
}

```

```

***** Dados do Aluno *****
Digite o nome do aluno: AA
Digite a nota 1: 10
Digite a nota 2: 10
Digite a nota 3: 10
***** Dados do Aluno *****
Digite o nome do aluno: BBBB
Digite a nota 1: 2
Digite a nota 2: 2
Digite a nota 3: 2
***** Boletim final *****
Aluno AA está Aprovado
Aluno BBBB está Reprovado

```

```
public class Jogador {  
  
    private String nome;  
    private int idade;  
    private float altura;  
  
    public Jogador(String n, int i, float alt)  
    {  
        nome = n;  
        idade = i;  
        altura = alt;  
    }  
  
    public String RetornaNome()  
    {  
        return nome;  
    }  
  
    public float RetornaAltura()  
    {  
        return altura;  
    }  
  
    public float RetornaIdade()  
    {  
        return idade;  
    }  
}
```

```
private Jogador []VetorJogadores;
private int totaljogadores ;

public TimeVolei()
{
    VetorJogadores = new Jogador[6];
    totaljogadores = 0;
}

public void InsereJogador(String n, int i, float d)
{
    if(totaljogadores < 6)
    {
        VetorJogadores[totaljogadores] = new Jogador(n,i,d);
        totaljogadores++;
    }
    return;
}

public String NomeJogadorMenorAltura()
{
    float MenorAltura = 999999;
    String NomeJogador = "";

    for (int i = 0; i < totaljogadores; i++)
    {
        if(VetorJogadores[i].RetornaAltura() < MenorAltura)
        {
            MenorAltura = VetorJogadores[i].RetornaAltura();
            NomeJogador = VetorJogadores[i].RetornaNome();
        }
    }
    return NomeJogador;
}
```

```

public class Principal01 {
    public static void main(String[] args) {

        //Declarar e Instanciar o objeto
        TimeVolei objetoTime = new TimeVolei();

        //Inserir os jogadores
        objetoTime.InsereJogador("Anderson", 34, (float) 1.98);
        objetoTime.InsereJogador("Flavio", 28, (float) 2.08);
        objetoTime.InsereJogador("Robson", 20, (float) 1.99);
        objetoTime.InsereJogador("Roberto", 31, (float) 1.91);
        objetoTime.InsereJogador("Andre", 36, (float) 1.97);
        objetoTime.InsereJogador("Leandro", 33, (float) 2.03);

        System.out.println("O jogador com a menor altura eh " + objetoTime.NomeJogadorMenorAltura());
    }
}

```

```

<terminated> Principal01 [Java Application] C:\Program Files\Java\jdk
O jogador com a menor altura eh Roberto

```

Exercícios 😊

Exercício 1

```
public class Principal01 {  
    public static void main(String[] args) {  
  
        Scanner scan = new Scanner(System.in);  
        Aluno []vetorObjetosAluno; //VETOR DE OBJETOS ALUNO  
  
        System.out.print("Digite a quantidade de alunos: ");  
        int qtdeAlunos = scan.nextInt();  
  
        vetorObjetosAluno = new Aluno[qtdeAlunos]; //CRIAR O VETOR DE OBJETOS  
  
        String nome;  
        int nota1, nota2, nota3;  
  
        for (int i = 0; i < qtdeAlunos; i++)  
        {  
            System.out.println("***** Dados do Aluno ***** ");  
            System.out.print("Digite o nome do aluno: ");  
            nome = scan.next();  
            System.out.print("Digite a nota 1: ");  
            nota1 = scan.nextInt();  
            System.out.print("Digite a nota 2: ");  
            nota2 = scan.nextInt();  
            System.out.print("Digite a nota 3: ");  
            nota3 = scan.nextInt();  
  
            //INSTANCIAR O OBJETO PARA CADA POSIÇÃO DO VETOR  
            vetorObjetosAluno[i] = new Aluno(nome, nota1,nota2,nota3);  
        }  
  
        System.out.println("***** Boletim final ***** ");  
        for (int i = 0; i < qtdeAlunos; i++)  
        {  
            System.out.println("Aluno " + vetorObjetosAluno[i].RetornaNome() + " está " + vetorObjetosAluno[i].ResultadoFinal());  
        }  
    }  
}
```

<terminated> Principal01.java Application C:\Program Files\jav

```
Digite a quantidade de alunos: 2  
***** Dados do Aluno *****  
Digite o nome do aluno: AA  
Digite a nota 1: 10  
Digite a nota 2: 10  
Digite a nota 3: 10  
***** Dados do Aluno *****  
Digite o nome do aluno: BBBB  
Digite a nota 1: 2  
Digite a nota 2: 2  
Digite a nota 3: 2  
***** Boletim final *****  
Aluno AA está Aprovado  
Aluno BBBB está Reprovado
```

Exercício 2

```
public class Principal01 {  
    public static void main(String[] args) {  
  
        //Declarar e Instanciar o objeto  
        TimeVolei objetoTime = new TimeVolei();  
  
        //Inserir os jogadores  
        objetoTime.InsereJogador("Anderson", 34, (float) 1.98);  
        objetoTime.InsereJogador("Flavio", 28, (float) 2.08);  
        objetoTime.InsereJogador("Robson", 20, (float) 1.99);  
        objetoTime.InsereJogador("Roberto", 31, (float) 1.91);  
        objetoTime.InsereJogador("Andre", 36, (float) 1.97);  
        objetoTime.InsereJogador("Leandro", 33, (float) 2.03);  
  
        System.out.println("O jogador com a menor altura eh " + objetoTime.NomeJogadorMenorAltura());  
    }  
}
```

<terminated> Principal01 [Java Application] C:\Program Files\Java\jdk
O jogador com a menor altura eh Roberto

Exercício 3

Solicitar que o usuário digite os dados dos jogadores;

Inserir novos métodos na Classe TimeVolei:

- Saber o nome do jogador mais alto;
- Saber a maior altura do jogador que têm idade acima de um determinado valor (INFORMADO PELO USUÁRIO);
- Saber a quantidade de jogadores que têm uma determinada idade (INFORMADO PELO USUÁRIO);
- Saber a quantidade de jogadores que possui a altura maior do que um determinado valor (INFORMADO PELO USUÁRIO);

SOBRECARGA

- Saber a média de idade de todos os jogadores;
- Saber a média de idade dos jogadores que têm acima de uma determinada altura (INFORMADA PELO USUÁRIO);

Exercício 4

Construa a classe Livros em Java, que obedeça à descrição abaixo:

Livro
- titulo: String - qtdPaginas: Integer - paginasLidas: Integer
+ Livro() + Livro(nome: String, qtdPaginas: Integer) + verificarProgresso(): void

- Crie os métodos get e set para cada um dos atributos.
- Crie ainda o método *verificarProgresso* que deverá calcular a porcentagem de leitura do livro até o momento. Para isso, deverá usar os valores dos atributos *qtdPaginas* e *paginasLidas*, através da formula: **porcentagem = paginasLidas * 100 / qtdPaginas**. O valor da porcentagem deverá ser mostrado na tela conforme a mensagem “Você já leu X por cento do livro”, onde o valor de X é o valor calculado pela fórmula apresentada anteriormente.

Crie um programa principal que instancie um objeto da Classe Livro. Em seguida, mostre os dados do livro cadastrado e o progresso de leitura. Altere o título e a quantidade de páginas lidas. Apresente novamente os dados do livro e o progresso de leitura.

Exercício 5

Implemente um Programa Orientado a Objetos, para armazenar as notas finais dos alunos de uma disciplina. No programa principal, o usuário deverá :

- **Informar a quantidade de alunos que serão cadastrados**
- **Inserir as notas dos alunos (uma por uma).**
- **Saber a média de todas as notas inseridas;**
- **Saber a média das notas que estão acima de um determinado valor.**
- **Saber quantos alunos foram reprovador (nota final < 4)**

- **Construtor**

- Informar a quantidade de alunos que serão cadastrados (parâmetro do construtor)
- Alocar memória para o vetor que irá armazenar as notas dos alunos;
- Inicializar os atributos necessários;

- **Métodos** (públicos):

- InserirNotas(double Nota) ;
- Media();
- Media (double valor);

- **Atributos**

- TODOS os atributos deverão ser private.

SUGESTÃO!!!

Exercício 6

Implemente uma classe `Empregado` , com atributos `nome`, `cpf`, e `salario` – todos devem ser encapsulados. Escrever um programa principal que:

- Instancie um vetor de 5 objetos da classe `empregado`. Obs.: o usuário deverá digitar as informações;
- Apresente na tela um relatório de todos os empregados cadastrados.



Muito Obrigada!

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

Contato: angelaabreu@gmail.com