

Programação Orientada a Objetos

Fausto Maranhão Ayres

2

A Linguagem Java

Sintaxe Java

- Baseada em C
- Fortemente tipada
- Fortemente orientada a objetos
- Alocação/liberação automática de memória
- Tratamento de exceção

Tutorial Java (Oracle)

<https://docs.oracle.com/javase/tutorial/java/index.html>

Os tutoriais Java TM



«Anterior • Trilha • Próxima»

[Pagina inicial](#)


Os Tutoriais Java foram escritos para JDK 8. Os exemplos e práticas descritos nesta página não tiram proveito das melhorias introduzidas em versões posteriores e podem usar tecnologia não mais disponível.

Consulte [Alterações na linguagem Java](#) para obter um resumo dos recursos de linguagem atualizados no Java SE 9 e versões subsequentes.

Consulte as [Notas de versão do JDK](#) para obter informações sobre novos recursos, aprimoramentos e opções removidas ou reprovadas para todas as versões do JDK.

Trilha: Aprendendo a linguagem Java

Esta trilha cobre os fundamentos da programação na linguagem de programação Java.

 **Conceitos de programação orientada a objetos** ensina os conceitos básicos por trás da programação orientada a objetos: objetos, mensagens, classes e herança. Esta lição termina mostrando como esses conceitos se traduzem em código. Sinta-se à vontade para pular esta lição se você já estiver familiarizado com a programação orientada a objetos.

fausto.ayres@ifpb.edu.br

3

Livros

1. Use a Cabeça! Java. SIERRA & BATES.
2. Java: Como Programar. DEITEL & DEITEL.
3. Core Java. HORSTMANN, CAY S.

fausto.ayres@ifpb.edu.br

4

Dica

- É importante aprender com os erros.
 - insira alguns erros no seu programa e veja as mensagens
- Erros mais comuns são:
 - Esquecimento de “}” e “;”
 - Nome de classe com letra minúscula:
 - Ex: system, string

Lembre-se que CTRL Z desfaz as alterações no editor do texto

Comentários

- 3 tipos:
 - `// comentário de linha`
 - `/*
 comentário de bloco
*/`
 - `/**
 * Comentário de bloco HTML a ser inserido no
 * arquivo <i>index.html<\i> pelo javadoc
 *
 * @author Fausto Ayres

 * @date 1/1/2020
 */`

Palavras reservadas do Java

abstract	do	if	package	synchronized
boolean	double	implements	private	this
break	else	import	protected	throw
byte	extends	instanceof	public	throws
case	false	int	return	transient
catch	final	interface	short	true
char	finally	long	static	try
class	float	native	strictfp	void
const	for	new	super	volatile
continue	goto	null	switch	while
default	assert			

fausto.ayres@ifpb.edu.br

7

Saída padrão (console)

- Métodos **println()** e **print()** exibem na console qualquer expressão (na forma de String)

```
System.out.println(2 + 3);           //"5"  
System.out.println("sala" + 21);    //"sala21"  
System.out.println();               //avança linha  
System.out.println("\n");           //avança linha
```

```
System.out.print("a");               //a  
System.out.print("b");               //ab
```

fausto.ayres@ifpb.edu.br

8

Tipos de dados

■ Tipo primitivo:

Tipo primitivo	Valores
boolean	true, false
char	caractere UNICODE
int	número inteiro
long	número inteiro grande
double	número fracionário

■ Tipo objeto:

- Todas as classes e interfaces

9

Declaração de variável e atribuição

■ Em Java, todas as variáveis são tipadas

tipo identificador = valor ou objeto;

letras e dígitos (case sensitive)

atribuição

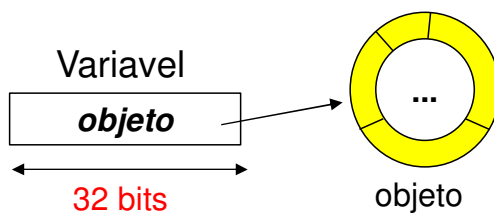
```
int a=5;           //valor
double x=4.7;      //valor
boolean p=true;    //valor
String nome="joao"; //objeto
```

Tamanho da variável

■ Tamanho depende do tipo



tipo primitivo	tamanho
boolean	8 bits
char	16 bits
int	32 bits
long	64 bits
double	64 bits



11

Inicialização obrigatória

- O compilador acusa erro na falta de inicialização de qualquer variável.

```
int i;  
System.out.println(i);    // erro de compilação
```

Escopo da variável

- O uso da variável depende do bloco onde é declarada (regra da ling. C).

```
public static void main(...) {  
    int a=2;                                //a é global  
  
    if (a > 0) {                             Bloco interno  
        int b=3;                             //b é local  
        a=4;  
        System.out.println(a+b);           //7  
    }  
  
    System.out.println(a);                 //4  
    System.out.println(b);                 //inacessível  
}
```

fausto.ayres@ifpb.edu.br

13

Declaração de Constante

- Adiciona-se a palavra reservada **final**
 - Permite apenas uma única atribuição

Ex:

```
final int N = 10 ;  
N = 20;           //erro
```

fausto.ayres@ifpb.edu.br

14

Operadores aritméticos

```
int i;  
i=5;  
i++;           // 6  
i++;           // 7  
  
i=2+5*3;       // 17  
i=(2+5)*3;     // 21
```

As expressões são executadas:

1. Segundo a prioridade dos operadores
 Maior: * /
 Menor: +-
2. Da esquerda para a direita
3. Parênteses alteram a ordem

Conversão double para int

```
i = (int) 2.75;           // 2
```

Conversão int para double

```
double x = (double) i;    //2.0
```

fausto.ayres@ifpb.edu.br

15

Divisão

Divisão parcial

```
i = 5 / 2;           // 2  
i = 5 % 2;           // 1
```

Divisão completa

```
x = (double)5 / 2    //2.5 como resultado de 5.0/2
```

OBS:

A divisão completa requer pelo menos um double.

fausto.ayres@ifpb.edu.br

16

Operadores relacionais

- Valores são comparados através dos operadores

==

!=

>

>=

<

<=

Obs:

Objetos, incluindo strings, são comparados através dos métodos equals() e compareTo()

Ex: `nome.equals("joao")`

fausto.ayres@ifpb.edu.br

17

Operadores Lógicos:

E:

&&

OU:

||

NÃO:

!

fausto.ayres@ifpb.edu.br

18

Tabela com todos operadores

Nível	Operadores	maior
1	. (seletor) [] ()	↓
2	++ -- ~ instanceof new clone - (unário)	
→ 3	* / %	
→ 4	+ -	
5	<< >> >>>	
6	< > <= >=	
7	== !=	
8	&	
9	^	
10		
11	&&	
12		
13	?:	
14	= op=	
15	,	menor

fausto.ayres@ifpb.edu.br

19

Classe utilitária Math

- Oferece métodos matemáticos:
- Não cria objeto.

```
double x;
```

```
long lo;
```

```
//inteiro longo
```

```
x = Math.abs(-3);
```

```
// 3 (modulo)
```

```
x = Math.pow(2,3);
```

```
// 8 (potencia)
```

```
x = Math.sqrt(16);
```

```
// 4 (raiz)
```

```
lo= Math.round(2.75);
```

```
// 3 (arredonda)
```

```
x = Math.floor(2.4);
```

```
// 2.0 (piso)
```

```
x = Math.ceil(2.4);
```

```
// 3.0 (teto)
```

```
x = Math.sin(3.1415);
```

```
// 0
```

```
x = Math.PI;
```

```
// valor de pi
```

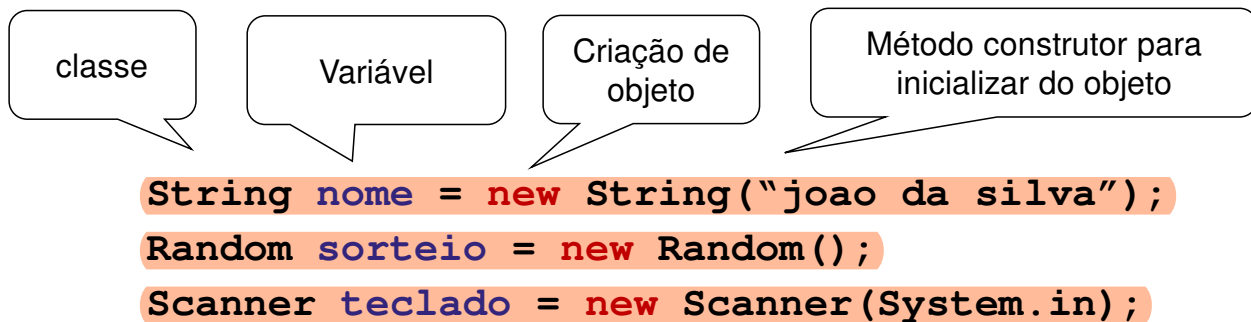
fausto.ayres@ifpb.edu.br

20

Variável de referência

- Armazena um objeto criado pelo operador **new** e inicializado pelo **construtor** da classe

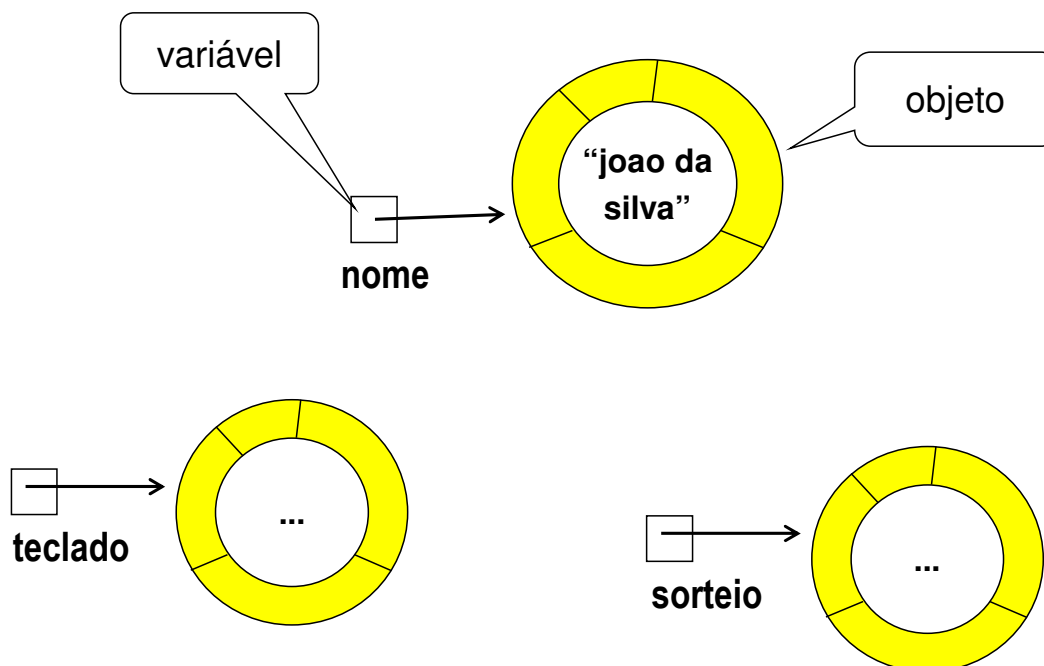
Exemplos



fausto.ayres@ifpb.edu.br

21

Variável de referência



fausto.ayres@ifpb.edu.br

22

Sintaxe para chamada de método

`variavel.método(...)`

A chamada de um método pode (ou não) retornar um resultado, dependendo do tipo do método

```
int i = nome.length();
```

```
//método do tipo int
```

```
lista.clear();
```

```
//método do tipo void
```

Principais classes do Java

Classe String

- Cria objeto para armazenar uma cadeia de caracteres Unicode entre "" e possui tamanho *ilimitado*

```
String s;  
s = ""; //um objeto com 0 caractere  
s = "a"; //um objeto com 1 caractere  
s = "sala" + " 1002" // "sala 1002"  
System.out.println(s.length()); //9
```

Igualdade de strings

- métodos **equals()** ou **equalsIgnoreCase()**

```
String a = "ana";  
String b = "joao";  
  
if (a.equals(b))  
    System.out.println("igual");  
else  
    System.out.println("diferente");  
  
if (!a.equals(b))  
    System.out.println("diferente");  
else  
    System.out.println("igual");
```

Conversão de string

■ Conversão de string para tipos primitivos

```
int i = Integer.parseInt("2");  
double d = Double.parseDouble("2.5");  
boolean b = Boolean.parseBoolean("true");
```

■ Conversão de tipos primitivos para String

```
String s;  
s = Integer.toString(2);  
s = Double.toString(2.5);  
s = Boolean.toString(true);
```

ou simplesmente concatenar com ""

```
S = 2+""; // "2"
```

Classe Random

■ Cria objeto para gerar números aleatórios entre 0 e n-1

```
Random sorteio = new Random();  
int a = sorteio.nextInt(100); // 0 a 99  
int b = sorteio.nextInt(60) + 30; // 30 a 89
```

Classe Scanner

- Um objeto para acessar o teclado para ler números e caracteres

```
Scanner teclado = new Scanner(System.in);
```

```
int i = teclado.nextInt();
```

```
double d = teclado.nextDouble();
```

```
String s = teclado.nextLine();
```

```
teclado.close();
```

Exemplo

- Programa para ler dois nomes e exibir uma mensagem de amizade pra eles

```
import java.util.Scanner;
```

CTRL+SHIFT+O

```
public class Amigo {
```

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

```
        Scanner teclado = new Scanner(System.in);
```

```
        System.out.println("Qual é o seu nome?");
```

```
        String nome1 = teclado.nextLine();
```

```
        System.out.println(nome1+" , de quem vc é amigo?");
```

```
        String nome2 = teclado.nextLine();
```

```
        System.out.println(nome1 +" é amigo de "+ nome2);
```

```
        teclado.close();
```

```
    }
```

```
}
```

Run

Qual é o seu nome?

maria

maria, de quem vc é amigo?

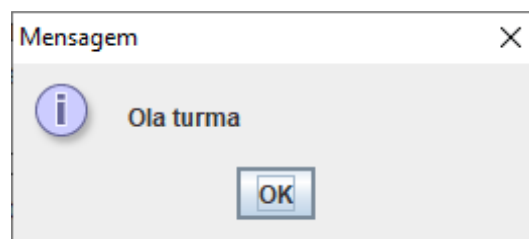
joao

maria é amigo de joao

Classe JOptionPane

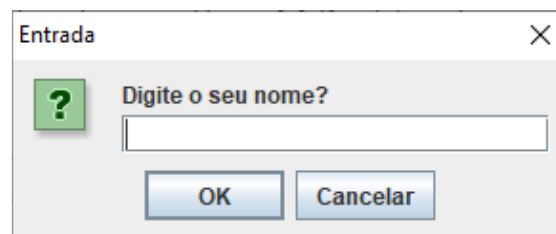
- Possui métodos que exibem *Janelas de Diálogos*

```
JOptionPane.showMessageDialog(null, "Ola turma");
```



```
String nome =
```

```
JOptionPane.showInputDialog("Digite o seu nome?");
```



Exemplo

- Programa para ler dois nomes e exibir uma mensagem de amizade pra eles

```
import javax.swing.JOptionPane;

public class Amigo2 {
    public static void main(String[] args){
        String nome1 = JOptionPane.showInputDialog(
            "Qual é o seu nome?");

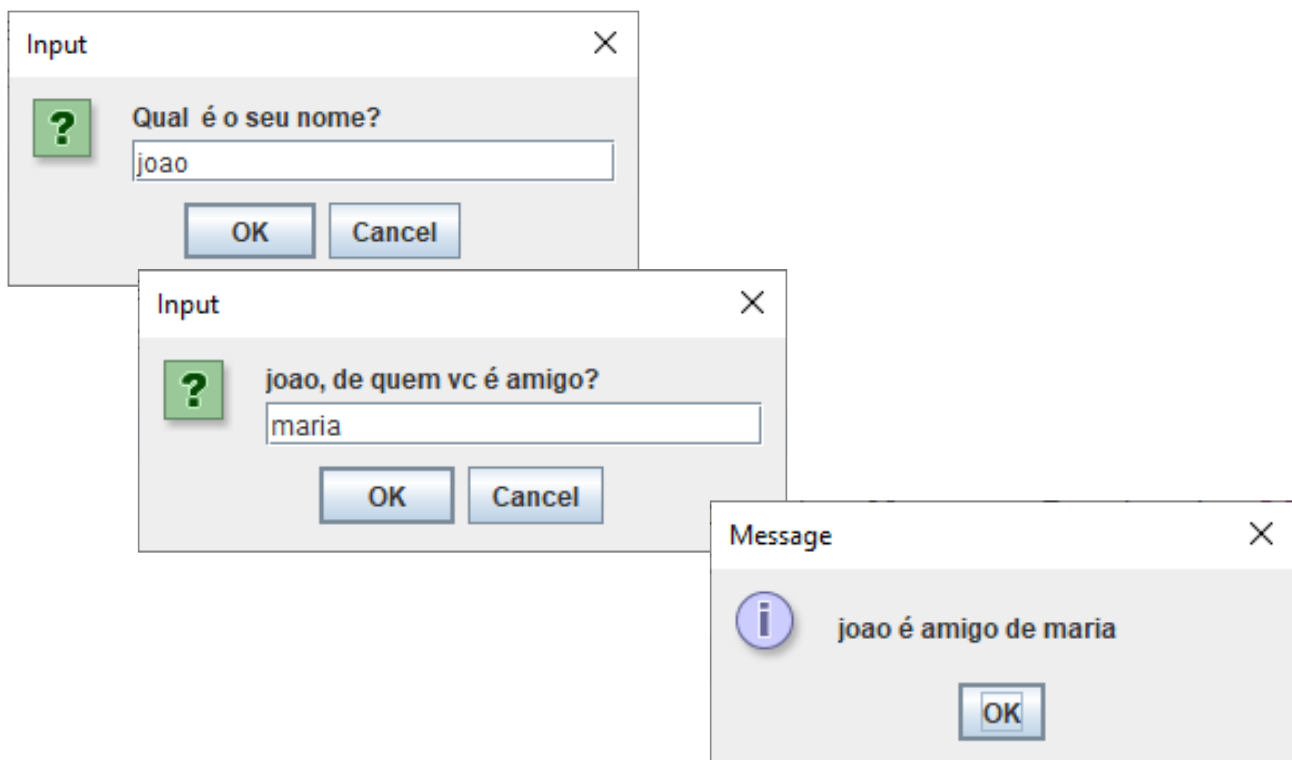
        String nome2 = JOptionPane.showInputDialog(
            nome1 + ", de quem vc é amigo?");

        JOptionPane.showMessageDialog(null,
            nome1 + " é amigo de " + nome2);
    }
}
```

fausto.ayres@ifpb.edu.br

33

Run



fausto.ayres@ifpb.edu.br

34

Estruturas de Controle

Estruturas de controle

■ Bloco de comandos:

```
{  
    comando1;  
    comando2;  
    comandoN;  
}
```

■ Seleção

```
if (condição)  
    comando/bloco  
[ else  
    comando/bloco ]
```

```
if(a > b)  
    n = 0;
```

O else é opcional

```
if(c == 0)  
    n++;  
  
else {  
    c = n;  
    n = 1;  
}
```

Não precisa de bloco, pois só tem um comando

Bloco com dois comandos

Exemplo

Índice de Masa Corporal

$$\text{IMC} = \frac{\text{Peso (Kg)}}{\text{Altura (m)}^2}$$

CLASSIFICAÇÃO	IMC
Abaixo do Peso	Abaixo 18,5
Peso Normal	18,5 - 24,9
Sobrepeso	25 - 29,9
Obesidade Grau I	30 - 34,9
Obesidade Grau II	35 - 39,9
Obesidade Grau III ou Mórbida	Maior ou Igual 40

IMC

```
public class IMC {  
    public static void main(String[] args) {  
        double peso = 150;  
        double altura = 1.70;  
        double imc = peso / Math.pow(altura, 2);  
        if (imc < 18.5)  
            System.out.println("abaixo do normal");  
        else  
            if (imc < 25)  
                System.out.println("normal");  
            else  
                if (imc < 30)  
                    System.out.println("acima do normal");  
                else  
                    if (imc < 35)  
                        System.out.println("obesidade 1");  
                    else  
                        if (imc < 40)  
                            System.out.println("obesidade 2");  
                        else  
                            System.out.println("Morbidade");  
    }  
}
```

Lendo peso e altura do teclado

```
public class IMC {  
    public static void main(String[] args) {  
        Scanner teclado = new Scanner(System.in);  
  
        double peso=0 ;  
        double altura=0 ;  
        try{  
            // ler peso e altura como string e converter  
            peso = Double.parseDouble(teclado.nextLine());  
            altura = Double.parseDouble(teclado.nextLine());  
        }  
        catch(Exception e){  
            System.out.println("formato do numero incorreto");  
            System.exit(0);    //termina programa  
        }  
  
        double imc = peso / Math.pow(altura,2);  
        ...  
    }  
}
```

39

Seleção múltipla

switch (char/int/String)

```
{  
    case valor1:    comando1;    break;  
    case valor2:    comando2;    break;  
    ...  
    default:        comandoN;    break;  
}
```

```
switch (dia){  
    case 1:         msg = "segunda";         break;  
    case 2:         msg = "terca";           break;  
    ...  
    case 7:         msg = "domingo";         break;  
    default:        msg = "inexistente";  
}
```

Estrutura de Repetição FOR

```
for(inicialização; teste; avanço)
    bloco ou comando;
```

repete o bloco n vezes

Exemplo: sortear 6 números

```
Random sorteio = new Random();
int numero;
for(int i=1; i<=6; i++) {
    numero = sorteio.nextInt(60);
    System.out.println(numero);
}
```

Boa pratica:
declarar variável antes do loop

fausto.ayres@ifpb.edu.br

41

Repetição (while)

```
while (condição)
    bloco ou comando;
```

Repete o bloco enquanto a condição for verdadeira

Exemplo: Adivinhar uma senha

```
Scanner teclado = new Scanner(System.in);
System.out.println("digite a senha");
String senha = teclado.nextLine();
while(!senha.equals("ifpb")) { //equalsIgnoreCase()
    System.out.println("tente outra vez");
    senha = teclado.nextLine();
}
teclado.close();
System.out.println("ok, vc acertou");
```

42

Repetição (do-while)

```
do{  
    ...  
}while (condição)
```

Repete o bloco enquanto a condição for verdadeira

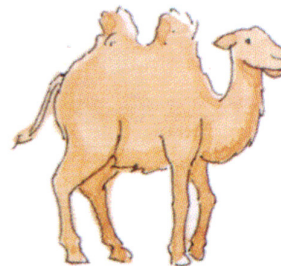
Exemplo:

```
do{  
    numero = sorteio.nextInt(60);  
    System.out.println(numero);  
}  
while(numero != 51) ;
```

43

Convenção de Nomes no Java

Notação **CamelCase**



Nomes de classes (1ª letra maiúscula)

Alo**M**undo, **S**ystem, **S**canner, **S**tring, **D**ouble **M**ath,
JOption**P**ane, ...

Nomes de métodos (1ª letra minúscula)

next**L**ine(), next**I**nt(), show**M**essage**D**ialog()
parse**I**nt(), equals**I**gnore**C**ase(), ...

Array

Conceito de array

- Um array é um **objeto** para armazenar **N elementos de mesmo tipo** em posições da memória de 0 até N-1
- Limitação: **tamanho fixo** (uma vez criado não pode ser alterado)

Exemplo: array de inteiros

```
public class TesteArray{  
  
    public static void main(String args[]){  
  
        int[] numeros = new int[4] ;  
        numeros[0] = 8;  
        numeros[1] = 2;  
        numeros[2] = 9;  
        numeros[3] = 2;  
  
        //int[] numeros = {8,2,9,2};  
  
        System.out.println(numeros.length);  
        System.out.println(numeros[0]);  
        System.out.println(numeros[3]);  
        System.out.println(numeros[2]);  
    }  
}
```

numeros	
[0]	8
[1]	2
[2]	9
[3]	2

4
8
2
9

length é um atributo do objeto

Varredura

■ *for tradicional* (acesso pelo índice)

```
for(int i=0; i < numeros.length; i++)  
    System.out.println(numeros[i]);
```

■ *for-each* (acesso pelo elemento)

Tipo do elemento Variável de iteração array

```
for(int n : numeros)  
    System.out.println(n);
```


Cont.

■ dobrar os números do array

```
int[] numeros = {8, 5, 30};  
for(int i=0; i<numeros.length; i++)  
    numeros[i] = 2*numeros[i];  
  
for(int n : numeros)  
    System.out.println(n);
```

16
10
60

	numeros
[0]	8 16
[1]	5 10
[2]	30 60

Classe utilitária *Arrays*

```
//obter uma String dos elementos  
Arrays.toString(nomes)
```

```
//ordenar elementos  
Arrays.sort(nomes)
```

```
//inicializar elementos  
Arrays.fill(nomes, "-")
```

ordenação

```
System.out.println(Arrays.toString(numeros));
```

```
Arrays.sort(numeros);
```

```
System.out.println(Arrays.toString(numeros));
```

```
[16, 10, 60]
```

```
[10, 16, 60]
```

array bidimensional (matriz)

```
//criação e inicialização  
int[][] matriz = new int[3][3];  
matriz[0][0] = 1;  
matriz[0][1] = 2;  
matriz[0][2] = 3;  
matriz[1][0] = 4;  
matriz[1][1] = 5;  
matriz[1][2] = 6;  
matriz[2][0] = 7;  
matriz[2][1] = 8;  
matriz[2][2] = 9;
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

```
//criação e inicialização
```

```
int[][] matriz = {{1,2,3},{4,5,6},{7,8,9}};
```

Cont.

```
//varredura
for(int linha=0; linha<=2; linha++) {
    for(int coluna=0; coluna<=2; coluna++)
        System.out.print(matriz[linha][coluna]+" ");
    System.out.println();
}
```

Lista

Conceito de lista

- Uma lista é um **objeto** para armazenar uma quantidade **ilimitada** de **objetos** de **mesmo tipo**
- Possui métodos para inclusão, remoção e acesso aos objetos armazenados

fausto.ayres@ifpb.edu.br

55

Principais classes

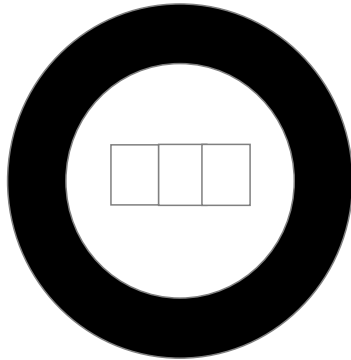
- **ArrayList<T> e LinkedList<T>**
onde T é tipo dos objetos da lista
- T não pode ser tipo primitivo, tem que ser classe
 - No lugar de tipo primitivo, usa-se a classe encapsuladora correspondente:
 - Integer
 - Double
 - Boolean
 - O java converte automaticamente **valor primitivo** em **objeto encapsulado** e vice-versa

56

ArrayList x LinkedList

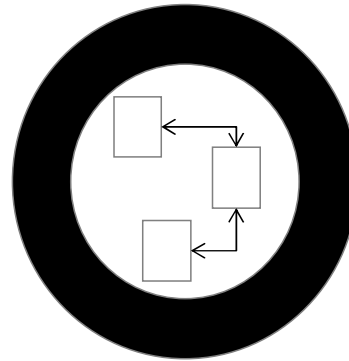
- Compartilham os mesmos métodos, mas com implementações diferentes

ArrayList<T>



Lista contígua

LinkedList<T>



Lista encadeada

fausto.ayres@ifpb.edu.br

57

Principais métodos

```
boolean add(Object o)  
boolean remove(Object o)  
Object remove(int index)  
Object get(int index)  
int size()  
boolean isEmpty()  
void clear()  
boolean addAll(List lista2)  
boolean removeAll(List lista2)
```

fausto.ayres@ifpb.edu.br

58

Exemplo: lista de inteiros

```
public class TesteArrayList{

    public static void main(String args[]){

        ArrayList<Integer> lista = new ArrayList<>();
        lista.add(8);
        lista.add(2);
        lista.add(9);
        lista.add(2);
        System.out.println(lista.size());
        System.out.println(lista.get(0));
        System.out.println(lista.get(3));
        System.out.println(lista.get(2));
        System.out.println(lista);

        lista.remove(0);
        lista.set(1, 999);
        System.out.println(lista);
    }
}
```

	lista
[0]	8
[1]	2
[2]	9
[3]	2

```
4
8
2
9
[8, 2, 9, 2]

[2, 999, 2]
```

fausto.ayres@ifpb.edu.br

59

Varredura

■ *for tradicional* (acesso pelo índice)

```
for(int i=0; i < lista.size(); i++)
    System.out.println(números.get(i));
```

■ *for-each* (acesso pelo elemento)

Tipo do elemento

Variável de iteração

ArrayList

```
for(int n : lista)
    System.out.println(n);
```

fausto.ayres@ifpb.edu.br

60

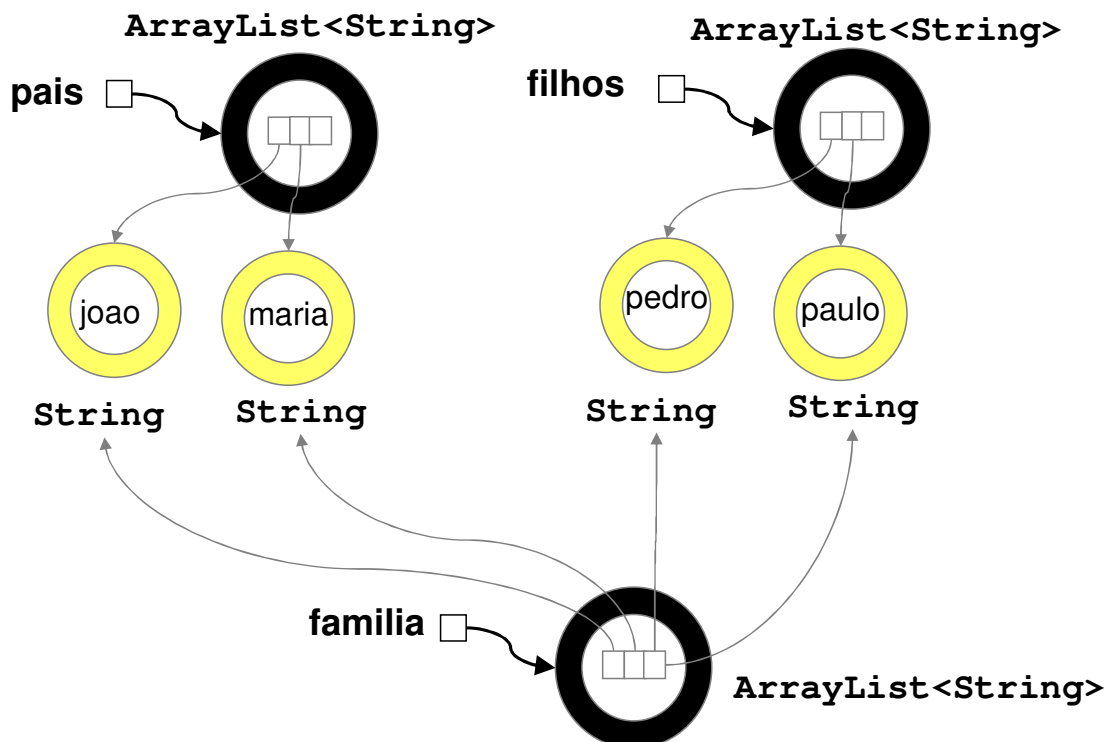
Unindo listas

```
ArrayList<String> pais = new ArrayList<>();  
pais.add("joao");  
pais.add("maria");  
  
ArrayList<String> filhos = new ArrayList<>();  
filhos.add("pedro");  
filhos.add("paulo");  
  
ArrayList<String> familia = new ArrayList<>();  
familia.addAll(pais); //copia as referências  
familia.addAll(filhos); //copia as referências  
  
System.out.println(familia);  
  
["joao", "maria", "pedro", "paulo"]
```

fausto.ayres@ifpb.edu.br

61

Unindo listas



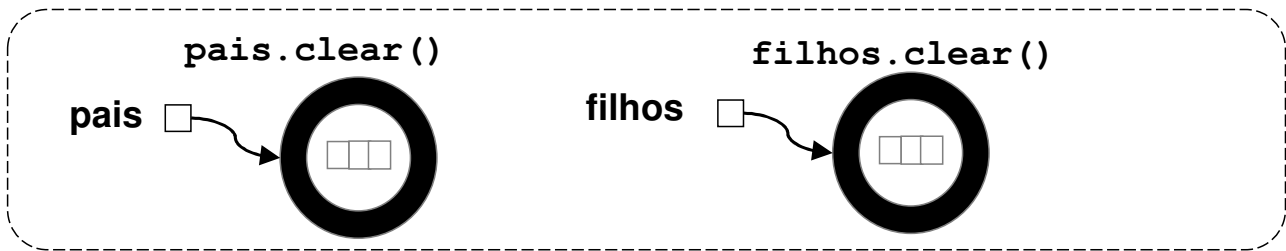
Um objeto pode pertencer a
várias listas

62

Esvaziando lista

■ Esvaziar as listas **pais** e **filhos**

```
pais.clear();           // size() é 0  
filhos.clear();         // size() é 0
```



Cuidado com a indexação

- Assim como no array, o uso indevido de índice aborta o programa com a mensagem “**Array Index Out Of Bounds Exception**”

```
System.out.println(família.get(5));           //aborta  
família.remove(5);                           //aborta
```


Classe utilitária **Collections**

Collections.shuffle(lista)

// embaralha

Collections.sort(lista)

// ordena

```
System.out.println(lista);
```

```
Collections.sort(lista);
```

```
System.out.println(lista);
```

[8, 2, 9, 2]

[2, 2, 8, 9]

String

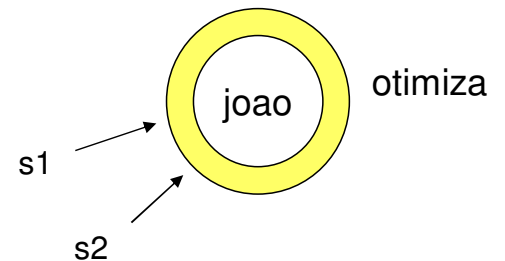
O que é uma string?

- Uma string é um **objeto** que armazena uma sequência ilimitada de caracteres Unicode

- Duas formas de criação:

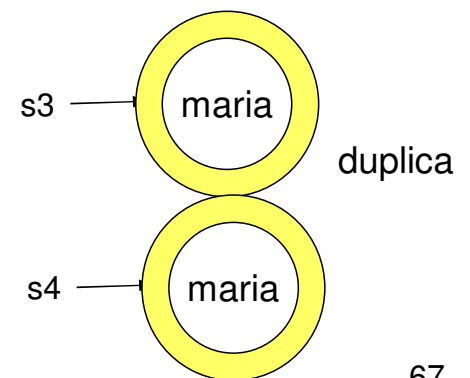
```
String s1 = "joao";
```

```
String s2 = "joao";
```



```
String s3 = new String("maria");
```

```
String s4 = new String("maria");
```



fausto.ayres@ifpb.edu.br

67

Bloco de string

- Mantém a estrutura original do texto, incluindo as quebras de linha

```
String texto = """  
poo poo poo  
  poo  
""";
```

```
System.out.println(texto);
```

```
poo poo poo  
  poo
```

fausto.ayres@ifpb.edu.br

68

Comparação

- Strings são comparadas como objetos

```
public boolean equals(String s)
```

```
public boolean equalsIgnoreCase(String s)
```

```
public int compareTo(String s)
```

```
public int compareToIgnoreCase(String s)
```

comparação	resultado
if ("maria".compareTo("ana") > 0)	true
if ("joao".compareTo("paulo") < 0)	true
if ("ze".compareToIgnoreCase("ZE") == 0)	true
if ("ze".equals("ZE"))	false

Principais métodos da classe String

<code>int length()</code>	<code>//tamanho</code>
<code>String toUpperCase()</code>	<code>//maiúsculas</code>
<code>String toLowerCase()</code>	<code>//minúsculas</code>
<code>boolean contains(String)</code>	<code>//busca</code>
<code>boolean startsWith(String)</code>	<code>//prefixo</code>
<code>boolean endsWith(String)</code>	<code>//sufixo</code>
<code>String replace(String, String)</code>	<code>//substitui</code>
<code>String trim()</code>	<code>//descarta branco</code>
<code>String substring(int, int)</code>	<code>//cópia</code>
<code>String substring(int)</code>	<code>//cópia</code>
<code>String[] split(String)</code>	<code>//separação</code>

Os métodos da classe String não modificam a string, mas retornam a cópia modificada do objeto

Exemplos

```
String nome = "joao da silva";
```

```
System.out.println(nome.contains("jo"));           //true
System.out.println(nome.contains("silva"));         //true
System.out.println(nome.startsWith("joao"));        //true
System.out.println(nome.endsWith("va"));            //true
int i = nome.indexOf("da");                          //5
```

```
System.out.println(nome.replace("joao", "joana"));
//joana da silva
```

```
String frase = "  muitos    brancos  ";
System.out.println(frase.trim());    //"muitos    brancos"
```

Concatenação

■ Sempre resulta em String

```
String nome = "joao" + "cruz";           //"joao cruz"
String termo = "2" + "2";                 //"22"
String local = "sala" + 100                //"sala100"
```

Separar

■ **split()** separa string em um array

```
String texto;
```

```
texto = "11-01-2021";
```

```
String[] partes = texto.split("-"); // ["11", "01", "2021"]
```

```
texto = "joao";
```

```
String[] letras = texto.split(""); // ["j", "o", "a", "o"]
```

Juntar

■ **join()** junta um array em uma string

```
texto = String.join("/", partes); // "11/01/2021"
```

```
texto = String.join("", letras); // "joao"
```

Método substring()

- substring(inicio, fim+1)

```
String palavra = "linguagem";
```

```
String s1 = palavra.substring(0, 3); // "lin"
```

```
String s2 = palavra.substring(4, 5); // "u"
```

```
String s3 = palavra.substring(5); // "agem"
```

Exercício

- Uma prova objetiva tem 10 questões cujas respostas podem ser "a", "b", "c", "d" ou "e".
- Faça um programa para ler as respostas do gabarito e as respostas da prova e calcular o número de acertos

Ex

Digite o gabarito com 10 caracteres:

aaee**cbdbcd**

Digite a resposta com 10 caracteres

aaee**cccaad**

Número de acertos: 6

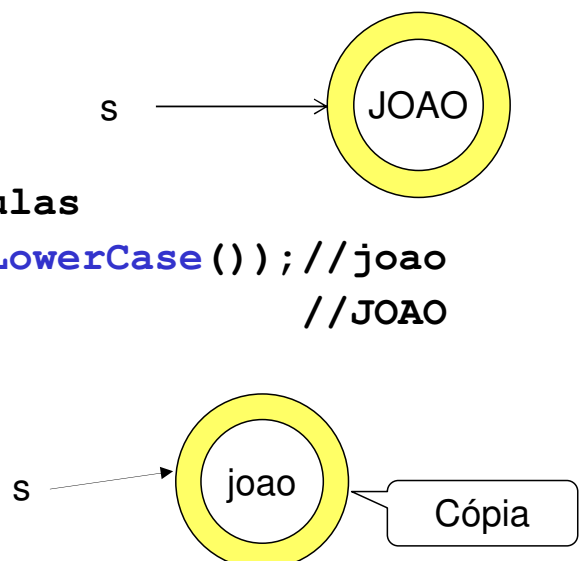
tópicos adicionais sobre string

OBS: objeto String é imutável

- Um objeto String **não pode ser** alterado.
- Os métodos da classe String retornam cópias modificadas

```
String s = "JOAO";  
//obter copia em minusculas  
System.out.println(s.toLowerCase()); //joao  
System.out.println(s);                //JOAO
```

```
//guardar a copia  
s = s.toLowerCase();
```



classes mutáveis

■ Classes que permitem alteração interna da string

- **StringBuffer**
- **StringBuilder**

```
StringBuffer buffer = new StringBuffer("");  
buffer.append("joao");           // "joao"  
buffer.append("silva");          // "joaosilva"  
buffer.insert(4, " da ");        // "joao da silva"  
buffer.replace(8,12,"sousa");    // "joao da sousa"  
  
String s = buffer.toString();    // "joao da sousa"
```

Formatação de número

String.format("formato", numero)

```
System.out.println( String.format("%10.0f", 2.75 ));  
System.out.println( String.format("%10.1f", 2.75 ));  
System.out.println( String.format("%10.2f", 2.75 ));  
System.out.println( String.format("%10.3f", 2.75 ));
```

Resultados:

```
_____3  
_____2,8      alinhamento à direita  
_____2,75  
_____2,750
```


Expressões regulares

Pattern.matches(regex, string)

Validar nome

```
String nome = "joao da silva";  
if (Pattern.matches("^[[ ]|\\p{L}]*+$", nome)) //true  
...
```

Validar email

```
String email = "joao.silva@gmail.com";  
if (Pattern.matches("^\\w*(\\.\\w*)?@\\w*\\. [a-z]+(\\. [a-z]+)?$",  
    email))  
...
```

Construção de regex

Simbolo	significado
*	Qualquer cadeia de caracter
?	opcional
+	Uma ou mais vezes
[]	Um dos elementos do conjunto
^	Exclusão
-	Intervalo
.	Qualquer caractere
\	Classe de caracteres pré-definidos: \d : Um dígito, \D : Algo que não seja um dígito, \s : Um espaço em branco, \S : Algo que não seja um espaço em branco, \w : Qualquer letra, dígito ou underscore (_) \W : Algo que não seja letra, dígito ou underscore