



Programação 2

Arrays e ArrayLists

Prof. Domingo Santos

domingos.santos@upe.br

- Arrays
- ArrayList
- try - catch

Arrays

- Um array é um grupo de variáveis (chamados elementos ou componentes) que contém valores todos do mesmo tipo
- São objetos; portanto, são considerados tipos por referência
- Os elementos de um array podem ser tipos primitivos ou tipos por referência
- Índice:
 - posição do elemento
 - número positivo
- Utilização dos valores:
 - `int sum = c[0] + c[1] + c[2]`

Nome do array (c) →	c[0]	-45
	c[1]	6
	c[2]	0
	c[3]	72
	c[4]	1543
	c[5]	-89
	c[6]	0
	c[7]	62
	c[8]	-3
	c[9]	1
	c[10]	6453
Índice (ou subscrito) do elemento no array c →	c[11]	78

- arrays são criados com a palavra-chave `new`
- É preciso definir o tipo dos elementos do array e, o número de elementos
- Exemplos:

```
int[] c = new int[12];  
String[] b = new String[100];
```

- Quando um array é criado, cada um de seus elementos recebe um valor padrão:
 - zero para os elementos de tipo primitivo numéricos
 - `false` para elementos boolean
 - `null` para referências.
-

- Exemplo

```
public class InitArray
{
    public static void main(String[] args)
    {
        // declara array variável e o inicializa com um objeto array
        int[] array = new int[10]; // cria o objeto array

        System.out.printf("%s%8s%n", "Index", "Value"); // títulos de coluna

        // gera saída do valor de cada elemento do array
        for (int counter = 0; counter < array.length; counter++)
            System.out.printf("%5d%8d%n", counter, array[counter]);
    }
} // fim da classe InitArray
```

- Exemplo

Definição de uma variável final:

- devem ser inicializadas antes de serem utilizadas e não podem ser modificadas depois

```
package com.example.helloworld;
public class InitArray
{
    public static void main(String[] args) {

        final int ARRAY_LENGTH = 10; // declara constante

        int[] array = new int[ARRAY_LENGTH]; // cria o objeto array

        System.out.printf("%s%8s%n", "Index", "Value"); // títulos de coluna

        // gera saída do valor de cada elemento do array
        for (int counter = 0; counter < array.length; counter++) {
            System.out.printf("%5d%8d%n", counter, array[counter]);
            array[counter] = counter;
        }
        System.out.println();
        for (int counter = 0; counter < array.length; counter++) {
            System.out.printf("%5d%8d%n", counter, array[counter]);
        }
    }
}
```

- Exemplo utilizando um inicializador de array

```
public class InitArray
{
    public static void main(String[] args)
    {
        // A lista de inicializador especifica o valor inicial de cada elemento
        int[] array = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };

        System.out.printf("%s%8s%n", "Index", "Value"); // títulos de coluna
        // gera saída do valor de cada elemento do array
        for (int counter = 0; counter < array.length; counter++) {
            System.out.printf("%5d%8d%n", counter, array[counter]);
        }

        } // fim da classe InitArray
}
```


- Exemplo somando os elementos de um array

```
public class InitArray
{
    public static void main(String[] args)
    {
        int[] array = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
        int total = 0;

        // adiciona o valor de cada elemento ao total
        for (int counter = 0; counter < array.length; counter++)
            total += array[counter];

        System.out.printf("Total of array elements: %d\n", total);
    }
} // fim da classe SumArray
```

A instrução for aprimorada

- Iterar pelos elementos de um array sem usar um contador

- Sintaxe:

```
for (parâmetro : nomeDoArray)  
    instrução
```

- Estratégia para simplificação

```
public class EnhancedForTest { // FORMA APRIMORADA  
    public static void main(String[] args) {  
        int[] array = { 87, 68, 94, 100, 83, 78,  
                        85, 91, 76, 87 };  
  
        int total = 0;  
        // adiciona o valor de cada elemento ao total  
        for (int number : array) {  
            total += number;  
        }  
  
        System.out.printf("Total of array elements: %d%n",  
                           total);  
    }  
}
```

Passagem por valor versus passagem por referência

Passagem por valor

uma cópia do valor do argumento é passada para o método chamado

As alterações na cópia do método chamado não afetam o valor na original

Suportado pelo Java

Passagem por referência

O método chamado pode acessar o valor do argumento no chamador diretamente e modificar esses dados, se necessário

Desempenho eliminando a necessidade de copiar dados

Não suportado pelo Java

```
int var_a = 10;  
int var_b = var_a;  
var_b = var_b * 2;
```

Resultado

var_a = 10
var_b = 20

Resultado

var_a = 20
var_b = 20

Passagem por valor versus passagem por referência

- Exemplo arrays + métodos e cópias de valores

```
// multiplica cada elemento de um array por 2
public static void modifyArray(int[] array2)
{
    for (int counter = 0;
        counter < array2.length;
        counter++)

        array2[counter] *= 2;
}

// multiplica argumento por 2
public static void modifyElement(int element)
{
    element *= 2;
    System.out.printf("Value of element in modifyElement: %d\n",
        element);
}
```

Passagem por valor versus passagem por referência

- Exemplo arrays + métodos e cópias de valores

```
public class PassArray {  
    public static void main(String[] args) {  
        int[] array = { 1, 2, 3, 4, 5 };  
  
        System.out.printf(  
            "Effects of passing reference to entire array:%n"+  
            "The values of the original array are:%n"  
        );  
  
        // gera saída de elementos do array original  
        for (int value : array) {  
            System.out.printf("%d", value);  
        }  
  
        modifyArray(array); // passa a referência do array  
        System.out.printf("%n\nThe values of the modified array"  
            + "are:%n");  
  
        // gera saída de elementos do array modificado  
        for (int value : array) {  
            System.out.printf("%d", value);  
        }  
  
        System.out.printf(  
            "%n\nEffects of passing array element value:%n"+  
            "array[3] before modifyElement: %d\n", array[3]);  
  
        modifyElement(array[3]); // tenta modificar o array[3]  
        System.out.printf("array[3] after modifyElement: %d\n", array[3]);  
    }  
}
```

```
Effects of passing reference to entire array:  
The values of the original array are:  
12345
```

```
The values of the modified array are:  
246810
```

```
Effects of passing array element value:  
array[3] before modifyElement: 8  
Value of element in modifyElement: 16  
array[3] after modifyElement: 8
```

Passagem por valor versus passagem por referência

- Por qual motivo o Java modificou os valores da variável do array?
- Particularidade de objetos no java:
 - Ambos objetos principal e cópia, referenciam o mesmo objeto na memória



Passagem por valor versus passagem por referência

- Exemplo arrays + métodos e cópias de valores

```
public class PassArray {  
    public static void main(String[] args) {  
        . . .  
  
        array[3] = modifyElement(array[3]); // tenta modificar o array[3]  
        System.out.printf("array[3] after modifyElement: %d\n" , array[3]);  
  
    }  
    . . .  
    // multiplica argumento por 2  
    public static int modifyElement(int element)  
    {  
        element *= 2;  
        System.out.printf("Value of element in modifyElement: %d\n" , element);  
        return element;  
    }  
}
```

```
Effects of passing reference to entire array:  
The values of the original array are:  
12345
```

```
The values of the modified array are:  
246810
```

```
Effects of passing array element value:  
array[3] before modifyElement: 8  
Value of element in modifyElement: 16  
array[3] after modifyElement: 16
```

- Representação de tabelas de valores com os dados organizados em linhas e colunas por meio de arrays com duas dimensões

	Coluna 0	Coluna 1	Coluna 2	Coluna 3
Linha 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Linha 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Linha 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Índice de coluna
Índice de linha
Nome do array

- Exemplo 1:

```
int[][] b = {{1, 2}, {3, 4}};
```

- Quais são os valores nas posições `b[0][1]` e `b[1][0]`?

- Exemplo 2:

```
int[][] b = {{1, 2}, {3, 4, 5}};
```

- Não é necessário ter mesma quantidade de linhas

- Exemplo 3:

```
int[][] b = new int[3][4];
```

- Inicialização no elementos nulos

- Exemplo 4:

```
int[][] b = new int[2][]; // cria 2 linhas  
b[0] = new int[5]; // cria 5 colunas para a linha 0  
b[1] = new int[3]; // cria 3 colunas para a linha 1
```

- Inicialização com elementos nulos, e com quantidades diferentes de colunas

Exemplo exibindo na tela

```
public class InitArray
{
    // cria e gera saída de arrays bidimensionais
    public static void main(String[] args)
    {
        int[][] array1 = {{1, 2, 3}, {4, 5, 6}};
        int[][] array2 = {{1, 2}, {3}, {4, 5, 6}};

        System.out.println("Values in array1 by row are");
        outputArray(array1); // exibe array1 por linha
        System.out.printf("%nValues in array2 by row are%n");
        outputArray(array2); // exibe array2 por linha
    }

    public static void outputArray(int[][] array)
    {
        // faz um loop pelas linhas do array
        for (int row = 0; row < array.length; row++)
        {
            // faz um loop pelas colunas da linha atual
            for (int column = 0; column < array[row].length; column++)
                System.out.printf("%d ", array[row][column]);

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

- Métodos estáticos prontos
- Sem a necessidade de “reinventar a roda”
- `import java.util.Arrays`

Exemplos:

sort , ordena um array em ordem ascendente	<pre>int[] numbers = {5, 2, 8, 1, 6}; Arrays.sort(numbers);</pre>
binarySearch : Realiza uma busca binária em um array ordenado e retorna o índice do elemento encontrado	<pre>int[] numbers = {1, 2, 3, 4, 5}; int index = Arrays.binarySearch(numbers, 3);</pre>
toString : Converte um array em uma string para exibição	<pre>int[] numbers = {1, 2, 3, 4, 5}; String arrayString = Arrays.toString(numbers);</pre>
fill : Preenche um array com um valor específico.	<pre>int[] numbers = new int[5]; Arrays.fill(numbers, 10);</pre>
copyOf : Copia parte de um array para outro array	<pre>int[] source = {1, 2, 3, 4, 5}; int[] destination = Arrays.copyOf(source, 3);</pre>
Arrays.stream(source).average().orElse(0) : cálculo da média, retorno do tipo Double	<pre>int[] source = {1, 2, 3, 4, 5}; double result = Arrays.stream(source).average().orElse(0); //mesmo para max e min</pre>

- Exercício de sala:
 - O usuário deve entrar com 5 números, e deve-se;
 - Exibir na tela a média;
 - Exibir na tela do menor para o maior;
-

ArrayList

- Classe que fornece métodos eficientes que organizam, armazenam e recuperam seus dados
- Diferentemente dos Arrays que não mudam automaticamente o tamanho em tempo de execução
- Alterar dinamicamente seu tamanho para acomodar mais elementos
- Exemplos:

```
ArrayList<String> list;
```

```
ArrayList<Integer> integers;
```

alguns métodos comuns da classe ArrayList

Método	Descrição
add	Adiciona um elemento ao <i>final</i> do ArrayList.
clear	Remove todos os elementos do ArrayList.
contains	Retorna true se o ArrayList contém o elemento especificado; caso contrário, retorna false.
get	Retorna o elemento no índice especificado.
indexOf	Retorna o índice da primeira ocorrência do elemento especificado no ArrayList.
remove	Sobrecarregado. Remove a primeira ocorrência do valor especificado ou o elemento no índice especificado.
Size	Retorna o número de elementos armazenados em ArrayList.

Listando itens de um ArrayList

```
// exibe elementos do ArrayList no console
public static void display(ArrayList<String> items, String header)
{
    System.out.printf(header); // exibe o cabeçalho
    // exibe cada elemento em itens
    for (String item : items) {
        System.out.printf(" %s", item);
    }
    System.out.println();
}
```



Exemplo ArrayList - parte 1

```
//Demonstração da coleção ArrayList<T> genérica.
import java.util.ArrayList;
public class ArrayListCollection {
    public static void main(String[] args) {

        // cria um novo ArrayList de strings com uma capacidade inicial de 10
        ArrayList<String> items = new ArrayList<String>();

        items.add("red"); // anexa um item à lista
        items.add(0, "yellow"); // insere "yellow" no índice 0

        // cabeçalho
        System.out.print("Display list contents with counter-controlled loop:");
        // exibe as cores na lista
        for (int i = 0; i < items.size(); i++)
            System.out.printf(" %s", items.get(i));
        . . .
    }
}
```



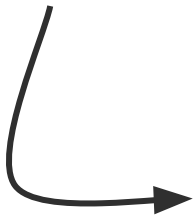
Display list contents with counter-controlled loop: yellow red

Exemplo ArrayList - parte 2

```
//Demonstração da coleção ArrayList<T> genérica.
import java.util.ArrayList;
public class ArrayListCollection {
    public static void main(String[] args) {
        . . .

        // exhibe as cores usando for aprimorada no método display
        display(items, "%nDisplay list contents with enhanced for statement:");

        items.add("green"); // adiciona "green" ao fim da lista
        items.add("yellow"); // adiciona "yellow" ao fim da lista
        display(items, "List with two new elements:");
        . . .
    }
}
```



```
Display list contents with enhanced for statement: yellow red

List with two new elements: yellow red green yellow
```

Exemplo ArrayList - parte 3

```
//Demonstração da coleção ArrayList<T> genérica.
import java.util.ArrayList;
public class ArrayListCollection {
    public static void main(String[] args) {
        . . .

        items.remove("yellow"); // remove o primeiro "yellow"
        display(items, "Remove first instance of yellow:");
        items.remove(1); // remove o item no índice 1
        display(items, "Remove second list element (green):");
        // verifica se um valor está na List
        System.out.printf("\n\"red\" is %s in the list%n", items.contains("red") );

        // exibe o número de elementos na List
        System.out.printf("Size: %s%n", items.size());
    }
}
```

Remove first instance of yellow: red green yellow

Remove second list element (green): red yellow

"red" is true in the list

Size: 2

Calcular média de valores. Esses valores possuem tamanhos variados

Parte 1

```
public static double calculateAverage(ArrayList<Integer> list) {  
    int sum = 0;  
    for (int num : list) {  
        sum += num;  
    }  
    if (list.size() == 0) {  
        return 0; // Retorna 0 se a lista estiver vazia  
    }  
    return (double) sum / list.size();  
}
```

Calcular média de valores. Esses valores possuem tamanhos variados

Parte 1

```
import java.util.Scanner;
import java.util.ArrayList;
public class ArrayListMean
{
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();

        Scanner input = new Scanner(System.in);
        System.out.print("Forma de uso, valores negativos saem do programa, "+
            "\n apenas positivos são considerados\n");
        System.out.print("Digite o 1º: ");

        int number = input.nextInt();
        int count = 2;

        while (number>-1) {
            System.out.printf("Digite o %d: ", count);
            number = input.nextInt();
            numbers.add(number);
            count ++;
        }
        double average = calculateAverage(numbers);
        System.out.println("A média é: " + average);
    }
}
```

Tratamento de exceções: processando a resposta incorreta

- Indica um problema que ocorre quando um programa é executado.
- Representa exceção à regra
- Programas tolerantes a falhas que podem resolver (ou tratar) exceções
- Permite que um programa continue a executar como se nenhum problema fosse encontrado
- Sintaxe:

```
try
{
    // Código que pode lançar uma exceção
} catch (TipoDeExcecao nomeDaExcecao) {
    // Código para lidar com a exceção
}
```

TipoDeExcecao	Descrição
NullPointerException	Lançada quando você tenta acessar ou operar em um objeto que não foi inicializado ou está definido como null.
ArrayIndexOutOfBoundsException	Ocorre quando você tenta acessar um índice fora dos limites de um array.
FileNotFoundException	Lançada ao tentar acessar um arquivo que não existe ou que não pode ser encontrado no sistema de arquivos.
IOException	Representa uma exceção de entrada/saída genérica. É uma superclasse para várias exceções relacionadas a operações de entrada/saída.
NumberFormatException	Gerada quando uma operação de conversão de string para um tipo numérico falha devido a um formato inválido da string.
Exception	Classe de exception genérica

- Exemplos

```
try {  
    int resultado = 10 / 0; // Tentativa de divisão por zero  
    System.out.println(resultado); // Esta linha não será alcançada  
} catch (ArithmeticException e) {  
    System.out.println("Erro: divisão por zero"); // Tratamento da exceção  
}
```

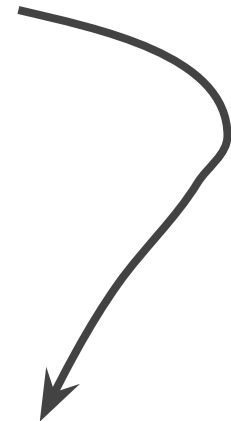
```
try {  
    int numero = Integer.parseInt(numeroString); // Tentativa de conversão de string para inteiro  
    System.out.println("Número: " + numero); // Esta linha não será alcançada  
} catch (NumberFormatException e) {  
    System.out.println("Erro: formato de número inválido");  
}
```


- Exemplos

```
public class InitArray
{
    public static void main(String[] args)
    {
        int[] array = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
        int total = 0;

        // adiciona o valor de cada elemento ao total
        for (int counter = 0; counter < 11; counter++) {
            total += array[counter];
        }

        System.out.printf("Total of array elements: %d\n", total);
    }
} // fim da classe SumArray
```



```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 10
at com.example.helloworld.InitArray.main(InitArray.java:12)
```

- Exemplos

```
public class InitArray
{
    public static void main(String[] args)
    {
        int[] array = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
        int total = 0;

        // adiciona o valor de cada elemento ao total
        for (int counter = 0; counter < 11; counter++) {
            try {

                total += array[counter];

            } catch (ArrayIndexOutOfBoundsException e) {

                System.out.println("Erro: " + e.getMessage()); // Imprime a mensagem da exceção
            }
        }
        System.out.printf("Total of array elements: %d\n", total);
    }
} // fim da classe SumArray
```

```
Erro: Index 10 out of bounds for length 10
Total of array elements: 849
```