

Arrays Unidimensionais

Arrays
Multidimensionais

Lista de argumentos de
tamanho variável

Classe Arrays

Coleção Genérica
ArrayList<T>

Referências

Arrays e ArrayLists

prof. Fábio Luiz Usberti

MC322 - Programação Orientada a Objetos

Instituto de Computação - UNICAMP - 2014



Arrays Unidimensionais

Arrays
Multidimensionais

Lista de argumentos de
tamanho variável

Classe Arrays

Coleção Genérica
ArrayList<T>

Referências

- 1 Arrays Unidimensionais
- 2 Arrays Multidimensionais
- 3 Lista de argumentos de tamanho variável
- 4 Classe Arrays
- 5 Coleção Genérica ArrayList<T>
- 6 Referências

Arrays Unidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Declaração

- Um array (**vetor**) consiste em um grupo de elementos todos **de mesmo tipo** armazenados de forma **sequencial**.
- Em Java, arrays são objetos, portanto são considerados tipos referenciados.
- Os elementos de um array podem ser **tipos primitivos ou referenciados** (incluindo outros arrays).
- Para obter um certo elemento de um array, devem ser especificados o **nome da variável** que referencia o array e a **posição relativa** do elemento no array (índice).

Arrays Unidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Array de tamanho 12

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

c[11]

78

Índice (ou subscripto) do
elemento no array c

Arrays Unidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Declaração

- Um objeto array possui como **atributo público** seu próprio tamanho (`length`).
- O tamanho de um array não pode ser modificado (`length` é um atributo `final`).
- Assim como qualquer outro objeto, um array pode ser instanciado com a palavra chave `new`.
- Também devem ser especificados em uma instanciação o tipo e o número de posições do array.

```
int [] b = new int[ 12 ]; // array de elementos primitivos
String [] c = new String[ 100 ]; // array de elementos referenciados
```

Arrays Unidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Declaração

- Quando um array é criado, cada elemento recebe um valor default, de acordo com o tipo do elemento.
- É possível colocar os colchetes ([]) tanto antes quanto depois do nome da variável.

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

- Quando os colchetes são colocados logo após o tipo, todas as variáveis da declaração passam a ser arrays.

```
int[] a, b, c; // declarando três arrays de inteiros
```

Arrays Unidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Exemplo

```
// GradeBook.java
// Livro de notas contendo um array para armazenar as notas dos alunos.

public class GradeBook {
    private String courseName; // nome do curso
    private int grades[]; // array de notas dos alunos

    // construtor
    public GradeBook(String name, int gradesArray[]) {
        courseName = name;
        grades = gradesArray;
    } // fim construtor

    // método para armazenar o nome do curso
    public void setCourseName(String name) {
        courseName = name;
    } // fim método setCourseName

    // método para obter o nome do curso
    public String getCourseName() {
        return courseName;
    } // fim método getCourseName

    // imprime mensagem de boas-vindas para o usuário
    public void displayMessage() {
        // getCourseName recebe o nome do curso
        System.out.printf("Bem-vindo ao livro de notas para '%s'\n\n",
            getCourseName());
    } // fim método displayMessage

    /* continua na próxima página... */
}
```

Arrays Unidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Exemplo

```
/* ... continua da página anterior */

// procura pela menor nota
public int getMinimum() {
    int lowGrade = grades[0];

    // varre o vetor
    for (int grade : grades) {

        if (grade < lowGrade)
            lowGrade = grade;
    } // fim for

    return lowGrade; // retorna a menor nota
} // fim método getMinimum

// procura pela maior nota
public int getMaximum() {
    int highGrade = grades[0];

    // varre o vetor
    for (int grade : grades) {

        if (grade > highGrade)
            highGrade = grade;
    } // fim for

    return highGrade; // retorna a maior nota
} // fim método getMaximum

/* continua na próxima página... */
```


Arrays Unidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Exemplo

```
/* ... continua da página anterior */

// determina a nota média
public double getAverage() {
    int total = 0;

    // soma a nota dos estudantes
    for (int grade : grades)
        total += grade;

    // retorna a nota média da turma
    return (double) total / grades.length;
} // fim método getAverage

// realiza diversas operações sobre os dados de entrada (notas)
public void processGrades() {
    // imprime as notas
    outputGrades();

    // chama método getAverage para determinar a nota média
    System.out.printf("\nA nota média é %.2f\n", getAverage());

    // chama método getMinimum e getMaximum
    System.out.printf("A menor nota é %d\nA maior nota é %d\n\n",
        getMinimum(), getMaximum());

    // chama método outputBarChart para imprimir o gráfico de distribuição de notas
    outputBarChart();
} // fim método processGrades

/* continua na próxima página... */
```

Arrays Unidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Exemplo

```
/* ... continua da página anterior */

// imprime gráfico de barras com a distribuição de notas
public void outputBarChart() {
    System.out.println("Distribuição de notas:");

    // armazena a frequência das notas em cada intervalo de 10
    int frequency[] = new int[11];

    // para cada grade, incrementa a frequência apropriada
    for (int grade : grades)
        ++frequency[grade / 10];

    // para cada grade de frequência, imprime um gráfico de barra
    for (int count = 0; count < frequency.length; count++) {
        // imprime barra de rótulos ( "00-09: ", ..., "90-99: ", "100: " )
        if (count == 10)
            System.out.printf("%5d: ", 100);
        else
            System.out.printf("%02d-%02d: ", count * 10, count * 10 + 9);

        // imprime barra de asteriscos
        for (int stars = 0; stars < frequency[count]; stars++)
            System.out.print("*");

        System.out.println();
    } // fim for
} // fim método outputBarChart

/* continua na próxima página... */
```

Arrays Unidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Exemplo

```
/* ... continua da página anterior */

// imprime o conteúdo da array de notas
public void outputGrades() {
    System.out.println("As notas são:\n");

    // imprime a nota de cada aluno
    for (int student = 0; student < grades.length; student++)
        System.out.printf("Aluno %2d: %3d\n", student + 1,
                           grades[student]);
} // fim método outputGrades

public static void main(String args[]) {
    // array que armazena a nota dos alunos
    int gradesArray[] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };

    GradeBook myGradeBook = new GradeBook(
        "MC302 Programação Orientada a Objetos", gradesArray);
    myGradeBook.displayMessage();
    myGradeBook.processGrades();
} // fim main
} // fim classe GradeBook
```

Arrays Unidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Exemplo: impressão na tela

Bem — vindo ao livro de notas para
MC302 Programação Orientada a Objetos!

As notas são:

Aluno 1: 87

Aluno 2: 68

Aluno 3: 94

Aluno 4: 100

Aluno 5: 83

Aluno 6: 78

Aluno 7: 85

Aluno 8: 91

Aluno 9: 76

Aluno 10: 87

A nota média é 84.90

A menor nota é 68

A maior nota é 100

Distribuição de notas:

00-09:

10-19:

20-29:

30-39:

40-49:

50-59:

60-69: *

70-79: **

80-89: ****

90-99: **

100: *

Arrays Multidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Declaração sem inicialização

- Um array multidimensional é um array onde **cada elemento é uma referência para outro array**.
- Trata-se de uma estrutura de dados com pelo menos duas dimensões.
- Um array bidimensional (**matriz**) pode ser utilizado para representar valores armazenados em uma tabela organizada em linhas e colunas.
- Um array bidimensional pode ser declarado da seguinte forma:

```
<tipo> <[]> <nome> = new <tipo>[<#linhas>][<#colunas>];
```

- Exemplo:

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

Arrays Multidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Matriz (array bi-dimensional)

	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

Arrays Multidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Declaração com inicialização

- Um array bidimensional pode ser declarado e inicializado com valores definidos pelo programador.

$$b = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

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

- Caso análogo de um array tridimensional:

$$c[0] = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad c[1] = \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix}$$

```
int [][][] c = { b, { { 5, 6 }, { 7, 8 } } };
```

Arrays Multidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Matrizes com linhas de tamanho variável

- As linhas de uma mesma matriz não precisam ter o mesmo tamanho.

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

- Os elementos da matriz `d` fazem referência a arrays de tamanhos distintos.
- `d[0]` referencia um array de tamanho `2` e `d[1]` referencia um array de tamanho `3`.
- O exemplo abaixo declara uma matriz `e` sem inicialização explícita, onde a primeira linha possui `5` elementos e a segunda linha possui `3` elementos.

```
int [][] e = new int[ 2 ][ ]; // cria duas colunas 2  
e[ 0 ] = new int[ 5 ]; // primeira linha possui 5 colunas  
e[ 1 ] = new int[ 3 ]; // segunda linha possui 3 colunas
```


Arrays Multidimensionais

Arrays Unidimensionais

Arrays
MultidimensionaisLista de argumentos de
tamanho variável

Classe Arrays

Coleção Genérica
ArrayList<T>

Referências

Exemplo

```
// InitArray .java
// Inicializa arrays bi-dimensionais.
public class InitArray {

    public static void main(String args[]) {
        int array1 [][] = new int [2][];
        array1[0] = new int[2];
        array1[1] = new int[4];
        int array2 [][] = { { 1, 2 }, { 3 }, { 4, 5, 6 } };

        System.out.println("Valores por linha da array1 ");
        outputArrayFor(array1);

        System.out.println("\nValores por linha da array2 ");
        outputArrayEnhancedFor(array2);
    } // fim main

    // imprime os elementos de um array bi-dimensional
    public static void outputArrayFor(int array [][]) {
        // laço para iterar entre as linhas
        for (int row = 0; row < array.length; row++) {
            // laço para iterar entre as colunas
            for (int column = 0; column < array[row].length; column++)
                System.out.printf("%d ", array[row][column]);

            System.out.println();
        } // fim for
    } // fim método outputArray

    /* continua na próxima página... */
```

Arrays Multidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Exemplo

```
/* ... continua da página anterior */

// imprime os elementos de um array bi-dimensional com for aprimorado
public static void outputArrayEnhancedFor(int array[][]) {
    // laço para iterar entre as linhas
    for (int[] row : array) {
        // laço para iterar entre os elementos
        for (int element : row)
            System.out.printf("%d ", element);

        System.out.println();
    } // fim for
} // fim método outputArray

} // fim classe InitArray
```

Arrays Multidimensionais

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Exemplo: impressão na tela

Valores por linha da array1

```
0 0  
0 0 0 0
```

Valores por linha da array2

```
1 2  
3  
4 5 6
```

Lista de argumentos de tamanho variável

Declaração

- Com uma lista de argumentos de tamanho variável é possível criar métodos que recebem um número de argumentos **definidos em tempo de execução**.
- A lista de argumentos deve conter o tipo do argumento seguido de reticências (`...`) e a identificação da lista de parâmetros.

```
public PrintStream printf(String format, Object... args)
```

- Um método pode conter **no máximo uma lista** variável de parâmetros.
- As reticências devem estar na última posição da lista de parâmetros.

Lista de argumentos com tamanho variável

Arrays Unidimensionais

Arrays
MultidimensionaisLista de argumentos de
tamanho variável

Classe Arrays

Coleção Genérica
ArrayList<T>

Referências

Exemplo

```
// VarargsTest.java
// Usando lista de argumentos de tamanho variável.

public class VarargsTest {
    // determina a média aritmética
    public static double average(double... numbers) {
        double total = 0.0;

        // somando o valor total com o laço for aperfeiçoado
        for (double d : numbers)
            total += d;

        return total / numbers.length;
    } // fim método average

    public static void main(String args[]) {
        double d1 = 5.0;
        double d2 = 10.0;
        double d3 = 20.0;

        System.out.printf("d1 = %.1f d2 = %.1f d3 = %.1f\n", d1, d2, d3);

        System.out.printf("Média de d1 e d2 é %.1f\n", average(d1, d2));
        System.out.printf("Média de d1, d2 e d3 é %.1f\n", average(d1, d2, d3));
    } // fim main
} // fim classe VarargsTest
```

Arrays Unidimensionais

Arrays
MultidimensionaisLista de argumentos de
tamanho variável

Classe Arrays

Coleção Genérica
ArrayList<T>

Referências

Lista de argumentos com tamanho
variável

Exemplo: impressão na tela

```
d1 = 5.0
```

```
d2 = 10.0
```

```
d3 = 20.0
```

```
Média de d1 e d2 é 7.5
```

```
Média de d1, d2 e d3 é 11.7
```

Arrays Unidimensionais

Arrays
MultidimensionaisLista de argumentos de
tamanho variável

Classe Arrays

Coleção Genérica
ArrayList<T>

Referências

Classe Arrays

API Java

- A classe `Arrays` encontra-se no pacote `java.util` e possui muitos métodos estáticos para manipulações típicas de arrays.
- Esses métodos foram sobrecarregados para arrays de tipos primitivos e de objetos.
- Exemplos de métodos pertencentes a essa API são **ordenação, busca binária, preenchimento, comparação e cópia**.
- Passar uma referência `null` para algum método dessa classe resulta em um **erro em tempo de execução**.

Arrays Unidimensionais

Arrays
MultidimensionaisLista de argumentos de
tamanho variável

Classe Arrays

Coleção Genérica
ArrayList<T>

Referências

Classe Arrays

Exemplo

```
// ArrayManipulations.java
// Exemplos de uso dos métodos da classe Arrays
import java.util .Arrays;

public class ArrayManipulations {

    // imprime vetor
    public static void displayArray(double[] array, String description) {
        System.out.println(description + ": " + Arrays.toString(array));
    } // fim método displayArray

    /* continua na próxima página... */
```


Arrays Unidimensionais

Arrays

Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica

ArrayList<T>

Referências

Classe Arrays

```

/* ... continua da página anterior */
public static void main(String[] args) {
    double[] doubleArray = { 8.4, 9.3, 0.2, 7.9, 3.4 };

    // ordena o vetor em ordem crescente
    Arrays.sort(doubleArray);
    displayArray(doubleArray, "doubleArray");

    // preenche um vetor com 10 elementos de números 7s
    double[] filledDoubleArray = new double[10];
    Arrays.fill (filledDoubleArray, 7.0);
    displayArray(filledDoubleArray, "filledIntArray ");

    // copia conteúdo de um vetor para outro
    double[] doubleArrayCopy = Arrays.copyOf(doubleArray, doubleArray.length * 2);
    displayArray(doubleArrayCopy, "doubleArrayCopy");

    // compara doubleArray com doubleArrayCopy
    boolean b = Arrays.equals(doubleArray, doubleArrayCopy);
    System.out.printf("\ndoubleArray %s doubleArrayCopy\n", (b ? "==" : "!="));

    // compara doubleArray com filledDoubleArray
    b = Arrays.equals(doubleArray, filledDoubleArray);
    System.out.printf("doubleArray %s filledDoubleArray\n", (b ? "==" : "!="));

    // procura em doubleArray pelo valor 3.4
    int location = Arrays.binarySearch(doubleArray, 3.4);
    if (location >= 0)
        System.out.printf("Achei valor 3.4 na posição %d do vetor doubleArray\n", location);
    else
        System.out.println("O valor 3.4 não foi encontrado no vetor doubleArray");
} // fim main
} // fim classe ArrayManipulations

```

Arrays Unidimensionais

Arrays
MultidimensionaisLista de argumentos de
tamanho variável

Classe Arrays

Coleção Genérica
ArrayList<T>

Referências

Classe Arrays

Exemplo: impressão na tela

```
doubleArray: [0.2, 3.4, 7.9, 8.4, 9.3]  
filledIntArray : [7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0]  
doubleArrayCopy: [0.2, 3.4, 7.9, 8.4, 9.3, 0.0, 0.0, 0.0, 0.0, 0.0]
```

```
doubleArray != doubleArrayCopy  
doubleArray != filledDoubleArray  
Achei valor 3.4 na posição 1 no vetor doubleArray
```

Coleção Genérica ArrayList<T>

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

API Java

- A classe `ArrayList` (`java.util`) consiste em uma estrutura de dados alternativa para armazenar dados de modo sequencial, como um array.
- A diferença do `ArrayList` está no fato de que essa estrutura **modifica seu tamanho automaticamente** para comportar novos dados ou após a remoção de dados.
- `ArrayList<T>` é uma das **coleções genéricas** disponíveis para um aplicativo Java.
- É uma **coleção** pois foi implementada para armazenar grupos de objetos relacionados (de mesmo tipo).
- É **genérica** pois foi implementada para armazenar um tipo genérico `T` (não primitivo), definido pelo programador na declaração de uma variável e instanciação de um objeto `ArrayList<T>`.

```
ArrayList< Integer > list = new ArrayList<Integer>();
```

Coleção Genérica ArrayList<T>

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

```
// ArrayListCollection.java
// Demonstração da coleção genérica ArrayList<T>.
import java.util . ArrayList;

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

        // cria uma nova ArrayList que armazena Strings. Capacidade inicial igual a 10 (default) .
        ArrayList<String> items = new ArrayList<String>();
        items.add("vermelho"); // anexa um novo elemento à lista
        items.add(0, "amarelo"); // insere um elemento na posição 0
        System.out.println("Imprime conteúdo da lista: " + items.toString());

        items.add("verde"); // anexa um elemento à lista
        items.add("amarelo"); // anexa um elemento à lista
        System.out.println("Imprime a lista com os dois novos elementos: " + items.toString());

        items.remove("amarelo"); // remove a primeira ocorrência "amarelo"
        System.out.println("Remove a primeira ocorrência de amarelo: " + items.toString());

        items.remove(1); // remove item na posição 1 (segundo elemento da lista)
        System.out.println("Remove o segundo elemento da lista (verde): " + items.toString());

        // verifica se um certo elemento se encontra na lista
        System.out.printf("\n"vermelho" %sestá na lista\n",
            items.contains("vermelho") ? "" : "não ");

        // imprime o número de elementos contidos na lista
        System.out.printf("Tamanho da lista: %s\n", items.size());

    } // fim main
} // fim classe ArrayListCollection
```

Coleção Genérica ArrayList<T>

Arrays Unidimensionais

Arrays Multidimensionais

Lista de argumentos de tamanho variável

Classe Arrays

Coleção Genérica ArrayList<T>

Referências

Exemplo: impressão na tela

```
Imprime conteúdo da lista: [amarelo, vermelho]
Imprime a lista com os dois novos elementos: [amarelo, vermelho, verde, amarelo]
Remove a primeira ocorrência de amarelo: [vermelho, verde, amarelo]
Remove o segundo elemento da lista (verde): [vermelho, amarelo]
"vermelho" está na lista
Tamanho da lista: 2
```

Arrays Unidimensionais

Arrays
MultidimensionaisLista de argumentos de
tamanho variável

Classe Arrays

Coleção Genérica
ArrayList<T>

Referências

Referências

- 1 Java: Como Programar, Paul Deitel & Heivey Deitel; Pearson; 7a. Ed. (no. chamada IMECC – 05.133 D368j)
- 2 Data Structures and Algorithms with Object Oriented Design Patterns in Java, Bruno Preiss;
(<http://www.brpreiss.com/books/opus6/>)
- 3 The Java Tutorials (Oracle)
(<http://docs.oracle.com/javase/tutorial/>)
- 4 Guia do Usuário UML, Grady Booch et. al.; Campus(1999)
- 5 Java Pocket Guide - Robert Liguori & Patricia Liguori; O'Reilley, 2008.