Introdução a Vetores

Conteudista: Prof. Esp. Alexander Gobbato Albuquerque
Revisão Textual: Prof.ª M.ª Magnólia Gonçalves Mangolini

Objetivos da Unidade:

- Enriquecer o conhecimento do aluno com novos conceitos;
- Aprender o conceito de vetores e como utilizá-los juntamente com os objetos e também passagem de parâmetros.

📄 Contextualização

Já estudamos, nos capítulos anteriores, o encapsulamento de informações e a reutilização de códigos. Vimos também como criar os métodos, usar encapsulamento e chamar os métodos criados através de objetos.

Nessa Unidade veremos os conceitos de vetores, e com o conceito aprendido poderemos criar estruturas complexas para atender a demanda de desenvolvimento.

Vamos aprender mais sobre vetores!

Material Teórico

Introdução a Vetores

Antes de começar a falar de *arrays*, devemos lembrar o que é uma variável e para que ela serve.

Nas Unidades anteriores, vimos que variável é a representação simbólica dos elementos, e que ficam armazenadas em memória. Cada variável corresponde a uma posição, cujo conteúdo pode se alterado ao longo do tempo durante a execução de um programa. No entanto, a variável só pode ser armazena por um único tipo.

Também vimos que as variáveis podem ser de três tipos: numéricas, alfanuméricas e lógicas, e que para declarar uma variável precisamos definir o seu nome e que tipo de dados será armazenado nela.

Antes de definir vetor, imaginemos a seguinte situação:

Precisamos criar um programa que armazene as seguintes notas de um aluno: 8.0, 10.0, 9.0, 10.0, 8.5, 10.0 e que calcule a média final.

A solução é bem simples. Definiremos uma variável, alocamos um espaço na memória do computador para armazenar uma e somente uma constante por vez, seja ela literal, numérica ou lógica. Quando atribuímos um valor à variável sobrescrevemos seu conteúdo.

Em algumas rotinas é necessário a manipulação de várias variáveis ao mesmo tempo, por exemplo, imagine que o programa deverá controlar o nome de 100 pessoas, ao invés de criar 100 variáveis é possível à criação de apenas uma, que é definida como *array*.

Mas como criar um *array*? É muito simples: especificamos o nome do *array* e o número de posições da memória que queremos alocar. Cada posição de memória pode armazenar um valor.

Os *arrays* são criados para que possam armazenar várias informações do mesmo tipo ou da mesma classe; com um vetor também é possível o armazenamento de vários objetos. Essas diversas informações que foram armazenadas em *array* são de fácil manipulação. As informações ficam disponíveis em forma de tabelas e podem ser acessadas por meio do índices.

Nas Unidades anteriores vimos que para se chamar um programa em *Java*, devemos criar um programa principal com o método *main*, e como parâmetro do método devemos criar um *string* args[ ], isso nada mais é do que um *array* de *Strings*.

Os *arrays* estão presentes em todas as linguagens de programação, em *Java*, os *arrays* são objetos que permitem armazenar uma lista de itens relacionados.

Arrays Unidimensionais

Os *arrays* de uma dimensão são aqueles que possuem um único índice para acessar os elementos. Eles podem ser declarados da seguinte forma:

TIPO-DE-DADO NOME-DO-ARRAY[ ] = NEW TIPO-DO-DADO[QTDE].

Em que:

Tipo-de-dado: Qualquer tipo de variável ou de classe;
Nome-do-array: Qualquer nome válido vale o mesmo conceito para a criação dos nomes de variáveis.

Exemplos:

Criar um *array* de nome Cidades contendo 50 elementos do tipo *String* e seu índice varia de 0 a 49.

```
String Cidades[] new String[50]
```

Criar um *array* de nome mes contendo 12 elementos do tipo int e seu índice varia de 0 a 11.

```
int mes [] new int [12]
```

Para se atribuir valores a um vetor (*array*), devemos colocar o índice desejado entre os colchetes, como demonstrado abaixo:

```
Ciedades[0] = "São Paulo";
Ciedades[10] = "Rio de Janeiro";
```

```
mes[] = 1;
mes[11] = 12;
```

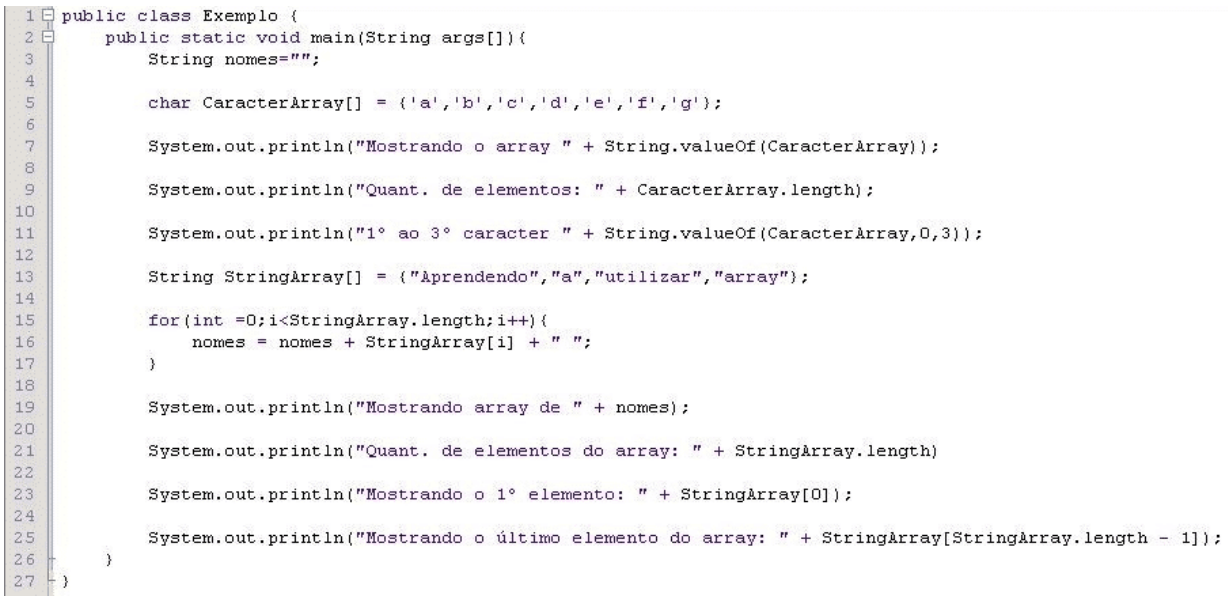
O exemplo que será demonstrado a seguir utiliza um *array* para armazenar um conjunto de argumentos do tipo inteiro, informado pelo usuário na linha de execução:

```
class Exemplo {
 public static void main (String args[]) {
  int i, total = 0;
  int N[] = new int [10];

   if (args.length > 0) {
    try {
     for (i = 0; i < args.length; i++) {
      N[i] = Integer.parseInt(args [i]) :
      total = total + N[i];
     }

     System.out.printIn ("0 Total de números digitados na ordem inverse é:'

     for (i = args.length; i <= 0; i--) {
      System.out.printIn (N[i] + " ");
      }
    } catch (NumberFormatException E){
        System. out . printin ("Os argumentos devem ser numeros inteiros")
        }
    } else {
       System . out . printIn ("Digite pelo menos um número");
     }
    }
}
```

Os vetores (*arrays*) podem ser criados e inicializados ao mesmo tempo, para isso, deve-se utilizar o operador *new* para instanciar um novo objeto. Os *arrays* criados entre chaves e separados por vírgula devem ter o mesmo tipo que a variável. Veja a sintaxe abaixo:

TIPO-DE-DADO NOME-DO-ARRAY[ ] = {valores separados por vírgula}

O exemplo abaixo mostra como usar, e explica também a função String.valueOf( ) para manipulação do conteúdo de um *array* de caracteres.



```
1  public class Exemplo {
2      public static void main(String args[]){
3          String nomes="";
4
5          char CaracterArray[] = {'a','b','c','d','e','f','g'};
6
7          System.out.println("Mostrando o array " + String.valueOf(CaracterArray));
8
9          System.out.println("Quant. de elementos: " + CaracterArray.length);
10
11         System.out.println("1° ao 3° caracter " + String.valueOf(CaracterArray,0,3));
12
13         String StringArray[] = {"Aprendendo","a","utilizar","array"};
14
15         for(int =0;i<StringArray.length;i++){
16             nomes = nomes + StringArray[i] + " ";
17         }
18
19         System.out.println("Mostrando array de " + nomes);
20
21         System.out.println("Quant. de elementos do array: " + StringArray.length)
22
23         System.out.println("Mostrando o 1° elemento: " + StringArray[0]);
24
25         System.out.println("Mostrando o último elemento do array: " + StringArray[StringArray.length - 1]);
26     }
27 }
```

Figura 1

Fonte: Reprodução

#ParaTodosVerem: a imagem é composta por um fundo branco, que contem descrição de um trecho de código, pode haver partes em que é necessário utilizar a tecla "TAB" para realizar a indentação correta. A tecla "TAB" será representada como [TAB] na descrição do código a seguir:
linha 1: public class Exemplo {
linha 2: [TAB]public static void main(String args[ ]){
linha 3: [TAB][TAB]String nomes="";
linha 4: Esta linha não contém código
linha 5: [TAB][TAB]char CaracterArray[ ] = {'a','b','c','d','e','f','g'};
linha 6: Esta linha não contém código
linha 7: [TAB][TAB]System.out.println("Mostrando o array "+
String.valueOf(CaracterArray));

```
linha 8: Esta linha não contém código
linha 9: [TAB][TAB]System.out.println("Quant. de elementos: "+
CaracterArray.length);
linha 10: Esta linha não contém código
linha 11: [TAB][TAB]System.out.println("1º ao 3º caracter " +
String.valueOf(CaracterArray,0,3));
linha 12: Esta linha não contém código
linha 13: [TAB][TAB]String StringArray[ ] =
{"Aprendendo","a","utilizar","array"};
linha 14: Esta linha não contém código
linha 15: [TAB][TAB]for(int =0;i<StringArray.length;i++){
linha 16: [TAB][TAB][TAB]nomes = nomes + StringArray[i] + " ";
linha 17: [TAB][TAB]}
linha 18: Esta linha não contém código
linha 19: [TAB][TAB]System.out.println("Mostrando array de " + nomes);
linha 20: Esta linha não contém código
linha 21: [TAB][TAB]System.out.println("Quant. de elementos do array: " +
StringArray.length);
linha 22: Esta linha não contém código
linha 23: [TAB][TAB]System.out.println("Mostrando o 1º elemento: " +
StringArray[0]);
linha 24: Esta linha não contém código
linha 25: [TAB][TAB]System.out.println("Mostrando o último elemento do
array: " + StringArray[StringArry.lengtg - 1]);
linha 26: [TAB]}
linha 27: }
Fim do código.
Fim da descrição.
```

- **Linha 5:** criando *array* de caracteres;
- **Linha 9:** mostrando a quantidade de elementos do *array*;
- **Linha 15:** armazenando informações no *array*.

Importante!

Alguns pontos devem ser levados em consideração quando se trabalhar com vetores (*arrays*) de caracteres e *strings*: nos *array* de caracteres são utilizados ‘ (apóstrofos) e já nos *arrays* de *string* são utilizados “ (aspas).

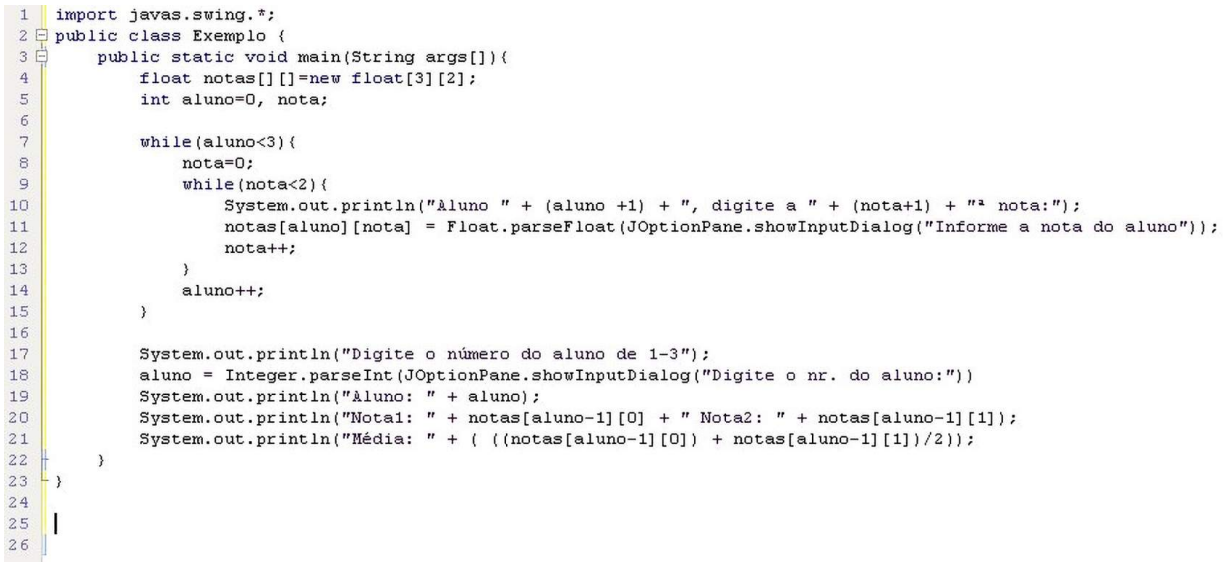
A função String.valueOf() pode ser usada para apresentar os elementos do *array* de caracteres ou um trecho dele. A função String.valueOf() não funciona para um conjunto de *arrays* do tipo *string*, apenas do tipo *char*(conjunto de caracteres).

Arrays Bidimensionais

Os *arrays* bidimensionais ou *arrays* multidimensionais permite a criação de vetores com mais de um índice. Essa característica possibilita que os valores sejam armazenados na forma de matrizes de qualquer dimensão.

TIPO-DO-DADO NOME-DO-ARRAY[ ][ ] = NEW TIPO-DO-DADO[<índice>][<índice>].

O exemplo abaixo demonstra o uso de *arrays* bidimensionais para armazenar 2 (duas) notas de 3 (três) alunos (total de 6 notas). Uma vez armazenadas, o programa solicita ao usuário o número do aluno para exibir suas notas e também a média.



```
import javas.swing.*;
public class Exemplo {
    public static void main(String args[]){
        float notas[][]=new float[3][2];
        int aluno=0, nota;

        while(aluno<3){
            nota=0;
            while(nota<2){
                System.out.println("Aluno " + (aluno +1) + ", digite a " + (nota+1) + "ª nota:");
                notas[aluno][nota] = Float.parseFloat(JOptionPane.showInputDialog("Informe a nota do aluno"));
                nota++;
            }
            aluno++;
        }

        System.out.println("Digite o número do aluno de 1-3");
        aluno = Integer.parseInt(JOptionPane.showInputDialog("Digite o nr. do aluno:"))
        System.out.println("Aluno: " + aluno);
        System.out.println("Nota1: " + notas[aluno-1][0] + " Nota2: " + notas[aluno-1][1]);
        System.out.println("Média: " + ( ((notas[aluno-1][0]) + notas[aluno-1][1])/2));
    }
}
```

Figura 2

Fonte: Reprodução

#ParaTodosVerem: a imagem é composta por um fundo branco, que contem descrição de um trecho de código, pode haver partes em que é necessário utilizar a tecla "TAB" para realizar a indentação correta. A tecla "TAB" será representada como [TAB] na descrição do código a seguir:
linha 1: import javas.swing.*;
linha 2: public class Exemplo {
linha 3: [TAB]public static void main(String args[]) {
linha 4: [TAB][TAB]float notas[][]=new float[3][2];
linha 5: [TAB][TAB]int aluno = 0, nota;
linha 6: Esta linha não contém código
linha 7: [TAB][TAB]while(aluno<3){
linha 8: [TAB][TAB][TAB]nota=0;
linha 9: [TAB][TAB][TAB]while(nota<2){
linha 10: [TAB][TAB][TAB][TAB]System.out.println("Aluno " + (aluno +1) + ", digite a " + (nota+1) + "ª nota:");
linha 11: [TAB][TAB][TAB][TAB]notas[aluno][nota] = Float.parseFloat(JOptionPane.showInputDialog("Informe a nota do aluno"));
linha 12: [TAB][TAB][TAB][TAB]nota++;
linha 13: [TAB][TAB][TAB]}
linha 14: [TAB][TAB][TAB]aluno++;
linha 15: [TAB][TAB]}
linha 16: Esta linha não contém código
linha 17: [TAB][TAB]System.out.println("Digite o número do aluno de 1-3");
linha 18: [TAB][TAB]aluno = Interger.parseInt(JOptionPane.showInputDialog("Digite o nr. do aluno:"))
linha 19: [TAB][TAB]System.out.println("Aluno: " + aluno);
linha 20: [TAB][TAB]System.out.println("Nota1: " + notas[aluno-1][0] + " Nota2: " + notas[aluno-1][1]);

linha 21: [TAB][TAB]System.out.println("Média: " + (((notas[aluno-1][0]) + notas[aluno-1][1])/2);
linha 22: [TAB]}
linha 23: }
Fim do código.
Fim da descrição.

Arrays de Objetos

Do mesmo jeito que criamos *array* de dados primitivos (*int*, *float*, *Double*, *string* e *char*), é possível criar um *array* para armazenamento de objetos. Isso é muito útil, pois permite as mesmas operações com diversos objetos do mesmo tipo.

O exemplo abaixo utiliza um *array* de objetos, aproveitando das funcionalidades da classe Veículo.

```
public class Veiculo{
    private int velocidade;

    public void setVelocidade (int vVelocidade) {
        velocidade = vVelocidade;
    }

    public int getVelocidade(){
        return velocidade;
    }
}

public class Exemplo {
    public static void main (String args[]){
        Veiculo veic[] = new Veiculco[300] ;

        for (int indice = 0; indice < 300; indice++) {
            veic [indice] = new Veiculo () ;
        }

        veic[0].setVelocidade(10);
        vein[10].set.Velocidade(200):
```

```
            veic[250].setVelocidade(150);

            for (int i = 0; i < 300; i++){
                veic[i].setVelocidade(0) ;
            }
        }
    }
```

Passagem de *Arrays* em Métodos

Os métodos já foram vistos nos capítulos anteriores. Vimos que é possível a criação de métodos que recebem valores, manipulam esses valores e retornam um resultado.

A passagem de *arrays* em métodos é basicamente o mesmo das variáveis. Quando um método é invocado, um vetor qualquer é informado; esse vetor pode ser manipulado internamente pelo método e também retornar o resultado.

Public static TIPO_DE-ARRAY[ ] nome-do-método (tipo-do-array nome-do-array[ ])

O exemplo a seguir mostra o método que recebe um *array* do tipo inteiro, organiza seus elementos e retorna o *array*.

```
import javas.swing.*;
public class Exemplo {
    public static void main(String args[]){
        int numeros[] = new int[10];
        for(int i=0; i < 10; i++){
            JOptionPane.showMessageDialog(null);
            numeros[i] = Integer.parseInt(JOptionPane.showInputDialog("Digite o numero " + (i+1)));
        }

        numeros = ordenaArray(numeros);
        mostraArray(numeros);
    }

    public static int[] ordenaArray(int arr[]){
        int x,y,aux;
        for(x=0;x<arr.length;x++){
            for(y=0;y<arr.length;y++){
                if(arr[x]<arr[y]){
                    aux=arr[y];
                    arr[y]=arr[x];
                    arr[x]=aux;
                }
            }
        }
        return arr;
    }

    public static void mostraArray(int arra[]){
        for(int i=0;i<arr.length;i++){
            System.out.println(arr[i] + " ");
        }
    }
}
```

Figura 3

Fonte: Reprodução

#ParaTodosVerem: A imagem é composta por um fundo branco, que contem descrição de um trecho de código, pode haver partes em que é necessário utilizar a tecla "TAB" para realizar a indentação correta. A tecla "TAB" será representada como [TAB] na descrição do código a seguir:
linha 1: import javas.swing.*;
linha 2: public class Exemplo {
linha 3: [TAB]public static void main(String args[]) {
linha 4: [TAB][TAB]int numeros[] = new int [10];
linha 5: [TAB][TAB]for(int i=0; i< 10; i++){
linha 6: [TAB][TAB][TAB]JOptionPane.showMessageDialog(null,);
linha 7: [TAB][TAB][TAB]numeros[i] =
Interger.parseInt(JOptionPane.showInputDialog("Digite o numero " + (i+1)));
linha 8: [TAB][TAB]}
linha 9: Esta linha não contém código
linha 10: [TAB][TAB]numeros = ordenaArray(numeros);
linha 11: [TAB][TAB]mostraArray(numeros);
linha 12: [TAB]}
linha 13: Esta linha não contém código
linha 14: [TAB]public static int[] ordenaArray(int arr[]){
linha 15: [TAB][TAB]int x,y,aux;
linha 16: [TAB][TAB]for(x=0; x<arr.length; x++){

linha 17: [TAB][TAB][TAB]for(y=0;y<arr.length;y++){
linha 18: [TAB][TAB][TAB][TAB]if(arr[x] <arr[y]){
linha 19: [TAB][TAB][TAB][TAB][TAB]aux=arr[y];
linha 20: [TAB][TAB][TAB][TAB][TAB]arr[y]=arr[x];
linha 21: [TAB][TAB][TAB][TAB][TAB]arr[x]=aux;
linha 22: [TAB][TAB][TAB][TAB]}
linha 23: Esta linha não contém código
linha 24: [TAB][TAB][TAB]}
linha 25: [TAB][TAB]}
linha 26: [TAB][TAB]return arr;
linha 27: [TAB]}
linha 28: Esta linha não contém código
linha 29: [TAB]public static void mostraArray(int arra[]){
linha 30: [TAB][TAB]for(int i=0; i<arr.length; i++){
linha 31: [TAB][TAB][TAB]System.out.println(arr[i] + " ");
linha 32: [TAB][TAB]}
linha 33: [TAB]}
linha 34: }
Fim do código.
Fim da descrição.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Livros

Aprenda Programação Orientada a Objetos em 21 Dias

SINTES, T. **Aprenda Programação Orientada a Objetos em 21 dias**. São Paulo: *Pearson Education* do Brasil, 2002, v. 1.

Java: como Programar

DEITEL, P.; DEITEL, H. ***Java*: Como Programar**. 8. ed. São Paulo: *Pearson Education* do Brasil, 2010.

Core Java

HORSTMANN, C. S.; CORNELL, G. ***Core Java***. 8. ed. São Paulo: *Pearson Education* do Brasil, 2010, v. 1.

Leitura

Orientação a Objetos – Simples Assim!

Clique no botão para conferir o conteúdo.

ACESSE

Referências

DEITEL, P.; DEITEL, H. **Java como Programar**. 8. ed. São Paulo: *Pearson Education* do Brasil, 2010.

FURGERI, S. ***JAVA 2:*** Ensino Didático; Desenvolvendo e Implementando Aplicações. Érica, 2002.

HORSTMANN, C. S.; CORNELL, G. ***Core Java***. 8. ed. São Paulo: Pearson Education do Brasil, 2010, v. 1.

SINTES, T. **Aprenda Programação Orientada a Objetos em 21 Dias**. São Paulo: Pearson Education do Brasil, 2002, v. 1.