Vetores e Arrays

INSTITUTO FEDERAL
São Paulo

Prof. Dr. João Paulo Lemos Escola

Tópicos da aula

- Vetores e matrizes;
- Classe Arrays;
- ArrayList;
- HashMap, HashSet e Iterator.

Vetores

- Um vetor é uma variável que armazena diversos valores indexados;
- Normalmente utilizado quando há necessidade de armazenar um conjunto de valores, o que acarretaria utilização de muitas variáveis;
- Sintaxe:

```
Tipo-de-dado[] nome-do-vetor = new tipo-de-dado[quantidade]
```

Vetores (cont.)

- Criando um vetor:

```
int idades = new int[10];
String nomes = new String[50];
Pessoa clientes = new Pessoa[100];
```

- Atribuindo valor a um índice do vetor:

```
idades[0] = 31;
nomes[49] = “João”;
clientes[5] = new Pessoa(“José da Silva”, “255.522.344-05”);
```

- Apresentando valor de um índice do vetor:

```
System.out.println(“Idade: “+idades[0]);
System.out.println(“Nome: “+nomes[3]);
System.out.println(“Cliente: “+clientes[5].getNome());
```

Exercício A05ex01

- Crie um programa com três vetores: int (tamanho: 3), double (tamanho: 5) e String (tamanho: 8); Atribua valores iniciais em todos os índices e mostre o primeiro e último valor de cada vetor;
- Não utilizar laços.

Apresentando dados do Vetor

- Podemos utilizar o laço de repetição FOR para mostrar todos os dados de um vetor:

```
for (int i : idades){
    System.out.println(“Idade: “+i);
}

for (Pessoa c : clientes){
    System.out.println(“Nome: “+c.getNome());
    System.out.println(“Nome: “+c.getCPF());
}
```

Exercício A05ex02

- Com o código do programa anterior, mostre o conteúdo completo dos vetores utilizando laços de repetição;

Matriz

- Também chamado de array bidimensional;
- Representa uma tabela de dados que podem ser acessados do modo linha/coluna;
- Sintaxe:

```
Tipo-de-dado[][] nome-do-array = new tipo-de-dado[linhas][colunas]
```

- Exemplos de declaração:

```
int[][] tabela = new int[10][2];
double[][] dados = new double[1][20];
String[][] convidados = new String[100][5];
```

Matriz (cont.)

- Atribuindo valor na matriz:

```
tabela[0][0] = 1; // primeiro valor da matriz
tabela[9][1] = 0; // último valor da matriz
dados[0][19] = Math.sqrt(Math.PI);
convidados[0][2] = “José da Silva”;
```

- Apresentando valor de um índice (linha/coluna) do array:

```
System.out.println(“Valor: “+tabela[0][0]);
System.out.println(“Dado: “+dados[0][9]);
System.out.println(“Primeiro convidado: “+convidados[0][0]);
```

Apresentando dados do Array

- Podemos utilizar o laço de repetição FOR para mostrar todos os dados de uma matriz:

```
for (int linha=0; linha<=tabela.length-1; linha++){
    for (int coluna=0; coluna<=tabela.length-1; coluna++){
        System.out.println(“Valor: “+tabela[linha][coluna]);
    }
}


for (int[] linha : tabela){
    for (int celula : linha){
        System.out.println(“Valor: “+celula);
    }
}
```

Exercício A05ex03

- Crie um programa que contenha uma matriz do tipo String para armazenar os nomes dos colegas da sala (coluna 1) e o time que cada um torce (coluna 2);
- Adicione um laço de repetição para mostrar os nomes e times de cada colega de sala;

Declarando arrays com dados

- É possível declarar arrays e inicializá-los na mesma linha de comando:

```
String[] nomes = {“José”, “João”, “Maria”, “Luiz”};
int valores[] = {1, 44, 22, 567, 1000, 0, 0};
double custos[] = {1.22, 9.1, 0.77};
int idades[][] = {{0,0,0}, {0,4,1}, {1000,222,333}};
```

Classe Arrays

- Permite a manipulação de Arrays de forma elegante:
  - sort(array): ordena o vetor;
  - equals(array1, array2): retorna true se os vetores forem iguais ou false caso contrário;
  - copyOf(vetor, novoTamanho): retorna um subvetor a partir do vetor passado como parâmetro;
  - copyOfRange(vetor, inicio, fim): retorna um subvetor de acordo com o índice inicial e final;
  - fill(vetor, valor): preenche o vetor com o valor especificado;

Exercício A05ex04

- Crie um programa com um vetor “a” inicializado com nomes 5 nomes de cidades da região;
- Mostre os valores de “a” utilizando laço de repetição;
- Crie uma cópia do vetor “a”, chamada “b” e mostre se os dados são iguais ao primeiro vetor;
- Organize os nomes de “a” utilizando a classe Arrays e mostre o resultado;
- Crie um subvetor chamado “sub” que contenha todos os valores do vetor “a”, menos o primeiro;

ArrayList

- Um ArrayList é uma boa prática de POO para armazenar conjuntos de variáveis (ou objetos);
- Não é necessário conhecer o tamanho final do conjunto;
- Podemos utilizar um laço For (each) de forma facilitada:

```
ArrayList<String> dados = new ArrayList<>();
dados.add("banana");
dados.add("abacaxi");
dados.add("morango");

for(String s: dados)
    System.out.println(s);
```

ArrayList: manipulação

- add()
  - Adiciona um item no ArrayList;
- size()
  - Retorna a quantidade de itens do ArrayList;
- get()
  - Retorna o valor do índice especificado;
- isEmpty()
  - Retorna true caso o ArrayList esteja vazio;
- clear()
  - Exclui os itens do ArrayList;
- contains()
  - Retorna true caso o ArrayList possua o conteúdo passado por parâmetro.

Exercício A05ex05

- Protótipo de software de papelaria:
  - Crie um ArrayList com 10 nomes de produtos;
  - Crie um ArrayList com os preços dos produtos, na mesma ordem;
  - Solicite o código do produto e a quantidade;
  - Mostre o nome do produto, quantidade, preço unitário e preço total;
  - Ao final, mostre o valor total e quantidade total de itens do pedido.

Classe Collections

- Contém diversos métodos para manipulação de ArrayLists, por exemplo:
  - sort: ordenação;
  - binarySearch (busca binária): procura o valor e retorna o índice correspondente;
  - fill: preenche o ArrayList com determinado valor igualmente em todas as posições;
  - copy: copia o conteúdo de um ArrayList para outro ArrayList de mesmo comprimento;

HashMap

- Trata-se de um tipo de vetor que armazena pares de itens no formato chave/valor;
- Permitindo índices de outros tipos, além do padrão inteiro;

```
HashMap<String, String> estados = new HashMap<>();
estados.put("SP", "São Paulo");
estados.put("MG", "Minas Gerais");
estados.put("RJ", "Rio de Janeiro");
estados.put("RS", "Rio Grande do Sul");
System.out.println(estados);
```

```
HashMap<String, Integer> estados = new HashMap<>();
estados.put("SP", 44);
estados.put("MG", 20);
estados.put("RJ", 16);
estados.put("RS", 11);
System.out.println(estados);
```

HashMap (cont.)

- Métodos da classe HashMap:
  - .get(): retorna o item especificado;
  - .remove(): permite remover um dos itens;
  - .clear(): limpa o vetor;
  - .size(): retorna o tamanho do vetor;
  - .keySet(): retorna as chaves de índices;
    - Pode ser utilizado para iteração em laço FOR;
  - .values(): retorna os valores do vetor;
    - Pode ser utilizado para iteração em laço FOR;

HashSet

- Permite somente dados não-repetidos;
- Valores redundantes são ignorados;
  - .contains(): verifica se o dado existe;
  - .remove(): exclui o dado;
  - .clear(): limpa o vetor;
  - .isEmpty(): verifica se o vetor está vazio.

```
HashSet<String> estados = new HashSet<>();
estados.add("MG");
estados.add("MG");
estados.add("RJ");
estados.add("RS");
System.out.println(estados);
```

Iterator

- Permite percorrer o vetor:
  - .next(): move para o próximo item;
  - .hasNext(): retorna true caso existam itens a percorrer;
  - .remove(): exclui o item atual;

```
ArrayList<String> estados = new ArrayList<>();
estados.add("SP");
estados.add("RS");
estados.add("MG");
estados.add("RJ");
Iterator<String> it = estados.iterator();
do{
    System.out.println(it.next());
}while(it.hasNext());
```

O que aprendemos?

- Vetores e matrizes;
- Classe Arrays;
- ArrayList;
- HashMap, HashSet e Iterator.

Na próxima aula...

- Métodos;
- Parâmetros e argumentos;
- Sobrecarga.