VOLTAR

Vetores e coleções de objetos

Capacitar o aluno a trabalhar com vetores e coleções de objetos como formas de armazenamento de dados temporários, que permite manipular volumes maiores de informações do que as variáveis simples. Entender como isso funciona e, quando se faz necessário, sua utilização.

NESTE TÓPICO

Vetores

O vetor, também chamado de Array, é uma estrutura simples que consegue armazenar um determinado número de informações do mesmo tipo.

Vetores unidimensionais

Como o nome diz, são vetores de uma dimensão. Para criar uma variável do tipo vetor, necessitamos apenas adicionar colchetes [] antes ou depois do nome da variável.

Exemplo:

```
1.  int meuVetor[];
2.  meuVetor = new int[] {1, 3, 5, 7, 9};
```

ou

```
1.  int meuVetor[] = {1, 3, 5, 7, 9};
```

O primeiro exemplo criou o vetor do tipo inteiro e depois o instanciou determinando seus valores. Já o segundo criou o vetor e já o instanciou, determinando seus valores diretamente.

Para obter algum valor do vetor, basta acessá-lo por meio de seu índice. Vamos imprimir o segundo elemento do vetor:

```
1.  out.println(meuVetor[1]);
```

Lembre-se que o primeiro elemento de um vetor é sempre zero no JSP (Java) e na maioria das linguagens. No entanto, há linguagens como o Visual Basic da Microsoft, por exemplo, que o vetor pode começar em um. (MSDN, 2013)

Para ilustrar, vejamos como ficaria o código para exibir todos os elementos de um vetor, utilizando o comando **for**:

```
1.  int meuVetor[] = {1, 3, 5, 7, 9};
2.  for (int i = 0; i < 5; i++) {
3.     out.println(meuVetor[i]);
4.  }
```

Note que o **for** foi determinado para ser executado de zero a quatro, já que a condição é **i < 5**. No entanto, o vetor é tratado pela linguagem como um objeto. Dessa forma, podemos utilizar length para acessar a quantidade de elementos de um vetor. Exemplo:

```
1.  int meuVetor[] = {1, 3, 5, 7, 9};
2.  for (int i = 0; i < meuVetor.length; i++) {
3.  out.println(meuVetor[i]);
4.  }
```

Agora, analise o programa a seguir. Veja que nele estão os exemplos mostrados anteriormente e também um novo exemplo que cria uma função para retornar o dia da semana por extenso.

A função recebe um número inteiro, que indica o número do dia da semana. Busca o dia por extenso dentro do vetor, utilizando o número como índice. Observe, no entanto, que há a necessidade de se colocar -1, porque o vetor começa sempre do zero, como já mencionado.

```
<%!
    private String DiaDaSemana(int nDiaDaSemana) {
        String diasDaSemana[] = {"Domingo", "Segunda", "Terca",
            "Quarta", "Quinta", "Sexta", "Sábado"};
        return diasDaSemana[nDiaDaSemana - 1];
    }
%>
<!DOCTYPE html>
<html>
    <head>
        <title>Exemplo de Vetores</title>
    </head>
    <body>
        <hr>
        <h4>Exemplo de Vetores</h4>
        <hr>Exibindo elementos de um vetor simples:
        <%
            int meuVetor[] = {1, 3, 5, 7, 9};
            for (int i = 0; i < meuVetor.length; i++) {
                out.print(meuVetor[i]);
            }
        %>

        <hr>Exibindo o primeiro dia da semana
        (através de função que armazena os dias da semana em um vetor):
        <%
            out.print(DiaDaSemana(1));
        %>

        <hr>Exibindo todos os dias da semana:
        <%
            for (int i = 1; i <= 7; i++) {
                out.print(DiaDaSemana(i));
                switch (i) {
                    case 6:
                        out.print(" e ");
                        break;
                    case 7:
                        out.print(".");
                        break;
                    default:
                        out.print(", ");
                }
            }
        %>
        <hr>
    </body>
</html>
```

Com esse programa devemos obter o seguinte resultado:

Exemplo de Vetores

Exibindo elementos de um vetor simples: 13579

Exibindo o primeiro dia da semana (através de função que armazena os dias da semana em um vetor): Domingo

Exibindo todos os dias da semana: Domingo, Segunda, Terca, Quarta, Quinta, Sexta e Sábado.

Vetores bidimensionais ou matrizes

No vetor unidimensional podemos ter a seguinte visão dos dados em relação ao armazenamento na memória, em que cada informação é guardada em uma coluna de uma única linha:

Índice	0	1	2

No caso da necessidade de armazenar dados para mais de uma linha, poderíamos criar um vetor bidimensional, analogicamente.

Índice	0	1	2
0	Nome	Telefone	Idade
1	João	7777-9999	32
2	Aline	2211-3324	17
3	Mirian	6666-8888	23

Para declarar um vetor bidimensional, basta acrescentar mais um colchetes []. Vejamos o exemplo seguinte que armazena o nome, telefone e idade de três pessoas. Na primeira linha, armazena um cabeçalho para as demais, assim, a primeira linha é de cabeçalho e as outras de dados:

```
1.  <% out.println("Exemplo de Vetor (Array) bidimensional<hr>");
2.  String planilha[][] = {
3.      {"Nome   ", "Telefone  ", "Idade"},
4.      {"Joao   ", "7777-9999 ", "32    "},
5.      {"Aline  ", "2211-3324 ", "17    "},
6.      {"Mirian ", "6666-8888 ", "23    "}
7.  };
8.  for (int lin = 0; lin < planilha.length; lin++) {
9.      for (int col = 0; col < planilha[lin].length; col++) {
10.         out.print(planilha[lin][col]);
11.     }
12.     out.print("<br>");
13. }
14. %>
```

Resultado obtido no browser com a execução do código anterior:

```
Exemplo de Vetor (Array) bidimensional

Nome   Telefone  Idade
Joao   7777-9999 32
Aline  2211-3324 17
Mirian 6666-8888 23
```

Coleções de objetos

Uma maneira muito simples de trabalhar com coleção de objetos em JSP ou Java é a utilização da interface List, que fornece métodos específicos para trabalhar com coleção de objetos, como inserir, remover, comparar, interagir etc. (DOCS, 2012a).

> Em conjunto com *ArrayList*, torna-se muito útil na programação orientada a objetos, pois permite inclusive aumentar a quantidade de elementos automaticamente. (DOCS, 2012b).

Vamos analisar atentamente o próximo exemplo. Para colocá-lo em prática, primeiramente se faz necessário a criação de uma classe chamada Contato.

```
1.  package suporte;
2.
3.  public class Contato {
4.
5.      String nome;
6.      String telefone;
7.      int idade;
8.
9.      public Contato(String nome, String tel, int idade) {
10.         this.nome = nome;
11.         this.telefone=tel;
12.         this.idade=idade;
13.     }
14.
15.     public String getNome() {
16.         return nome;
17.     }
18.
19.     public void setNome(String nome) {
20.         this.nome = nome;
21.     }
22.
23.     public String getTelefone() {
24.         return telefone;
25.     }
26.
27.     public void setTelefone(String telefone) {
28.         this.telefone = telefone;
29.     }
30.
31.     public int getIdade() {
32.         return idade;
33.     }
34.
35.     public void setIdade(int idade) {
36.         this.idade = idade;
37.     }
38. }
```

Agora o JSP para teste da interface List com a Classe ArrayList:

```
1.  <%@page import="java.util.List"%>
2.  <%@page import="java.util.ArrayList"%>
3.  <%@page import="suporte.Contato"%>
4.  <%!
5.      private List<Contato> getContatos()  {
6.          // Declara colecao contatos para classe Contato.
7.          List<Contato> contatos = new ArrayList<Contato>();
8.
9.          // Adicionar instancias do objeto Contato na coleção:
10.         contatos.add(new Contato("Joao", "7777-9999", 32));
11.         contatos.add(new Contato("Aline", "2211-3324", 17));
12.         contatos.add(new Contato("Mirian", "6666-8888", 23));
13.         // Retornar coleção:
14.         return contatos;
15.     }
16. %>
17.
18. <!DOCTYPE html>
19. <html>
20.     <head>
21.         <title>Exemplo de Coleções de Objetos</title>
22.     </head>
23.     <body>
24.         <hr>
25.         <h4>Exemplo de Coleções de Objetos</h4>
26.
27.         <pre>
28.             <% out.println("<hr>Exibindo elementos de um vetor simples:");
29.                 // Obter a lista e armazenar em ctts:
30.                 List<Contato> ctts = getContatos();
31.
32.                 // Laço para exibir a lista:
33.                 for (Contato ct : ctts)  {
34.                     out.println("---------------------------");
35.                     out.println("Nome____: " + ct.getNome());
36.                     out.println("Telefone: " + ct.getTelefone());
37.                     out.println("Idade___: " + ct.getIdade());
38.                 }
39.                 out.println("-----------------------------");
40.             %>
41.         </pre>
42.         <hr>
43.     </body>
44. </html>
```

Nas linhas dois e três o programa está fazendo o import das classes necessárias para a utilização do List e do ArrayList.

Na linha quatro, o import da classe Contato é criado para esse exemplo.

Na linha seis é feita a definição do método getContatos, que irá retornar um objeto contendo uma coleção de objetos Contato.

Na linha oito o objeto de coleção (contatos) é criado.

Nas linhas 11, 12 e 13 adicionam dados a objetos Contato que, por sua vez, é acrescentado na coleção (contatos).

A linha 15 retorna à coleção de objetos.

Na linha 31 é criada a variável de coleção ctts, que irá receber a coleção de objetos retornada pelo método getContatos().

Na linha 34 é feito um laço para obter os dados de cada Contato no objeto ct, da coleção ctts.

Das linhas 35 a 38 é realizada a exibição de cada atributo da classe Contato, por meio de seus métodos get. (getNome(), getTelefone() e getIdade()).

Pronto! Com esse exemplo conseguimos demonstrar o básico sobre coleção de objetos em JSP (ou Java). Veja a seguir o resultado em tela.

```
Exibindo elementos de um vetor simples:
-------------------------------
Nome____: Joao
Telefone: 7777-9999
Idade___: 32
-------------------------------
Nome____: Aline
Telefone: 2211-3324
Idade___: 17
-------------------------------
Nome____: Mirian
Telefone: 6666-8888
Idade___: 23
-------------------------------
```

Referências

DOCS, Oracle Java. *Lista de interface*. Disponível em <docs.oracle.com> (http://docs.oracle.com/javase/6/docs/api/java/util/List.html). Acesso em: 03 nov. 2012.

DOCS, Oracle Java. *Classe ArrayList*. Disponível em <docs.oracle.com> (http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html). Acesso em: 03 nov. 2012.

MSDN. *Visual Basic for Applications Reference*. Disponível em: <msdn.microsoft.com (http://msdn.microsoft.com/en-us/library/aa266179(v=vs.60).aspx). Acesso em: 08 jan. 2013.

Avalie este tópico

ANTERIOR
Estruturas de desvio condicional e de repetição em JSP

Índice

PRÓXIMO
Diretivas JSP e objetos implícitos

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site

Ajuda? (https://ava.un idCurso=)