Dicas da Turma do psvm

Manipulando Arrays

Fernando Anselmo
(fernando.anselmo@bsb.politec.com.br) atua na área de Desenvolvimento de Sistemas desde 1987. Autor de oito livros e diversos artigos em revistas especializadas. Atualmente, ocupa o cargo de Líder de Célula na Fábrica de Software na Politec Regional Brasília, Instrutor da X25 Informática e Coordenador do Brasília Java Users Group (DFJUG).

O que é "psvm"? É simplesmente a abreviatura do método mais famoso do Java, o public static void main (String args[]), as pessoas que têm grande experiência em programação normalmente são chamados de "cobras", resolvi apelidar os novatos de turma do psvm, ou seja, o iniciante, aquele que está começando agora e no máximo o que aprendeu a fazer foi:

```
public class Hello
{
    public static void main(String [] args)
    {
        System.out.println("Hello World");
    }
}
```

Se você está nessa turma, não tenha vergonha, saiba que fazer isso já foi um grande passo, principalmente sozinho. Brinco sempre com meus alunos dizendo que Java também se aprende com dicas acumuladas durante o tempo, então sempre irei substituir isso que você conquistou por um determinado trecho de código. Sendo assim, efetue as trocas substituindo exatamente o tão suado:

```
System.out.println("Hello World");
```

ou seja, use sempre o mesmo programa Hello como base e aprenda como funcionam os arrays em Java. Eles são um grande mistério e raramente alguém se preocupa em mostrar como funcionam, por causa de alguns detalhes, que são extremamente simples, exigem uma determinada dose de experiência e principalmente lógica e é isto que será apresentado neste artigo. Antes mesmo de começarmos, saiba que o Java versão 5.0 será usado, assim não causaremos nenhum problema de incompatibilidade.

Primeira Dica

Um array em Java é tratado como um objeto, afinal ele advém da classe Array, e como todo objeto é preciso:

Passo 1 – Declará-lo: int[] meuArray;
Passo 2 – Construí-lo: meuArray = new int[2];
Passo 3 – Inicializá-lo: meuArray[0] = 2; meuArray[1] = 4.

Aqui criamos um array com uma dimensão e duas posições do tipo inteiro, note os colchetes ("[]"), cada conjunto indica uma dimensão, imagine isso em termos de fichas dentro de uma gaveta, separada em pastas. Cada pasta seria uma posição do array, sendo que a primeira pasta é a posição zero, a segunda a posição um, e assim sucessivamente até atingirmos o total de pastas menos um. Poderíamos, por exemplo, navegar neste array da seguinte forma:

```
for (int pos = 0; pos < meuArray.length; pos++)
    System.out.println(meuArray[pos]);
```

Um laço de repetição for é realizado com os seguintes argumentos:

1º) valor de um atributo inicial (int pos = 0);
2º) condição lógica de repetição, enquanto esta for verdadeira repita o passo 3 (pos < meuArray.length);
3º) comando de incremento ou mesmo decremento (pos++).

Quanto às dimensões. Uma dimensão seria uma gaveta com as pastas, e duas dimensões? Armário completo. Localizamos em qual gaveta do armário a pasta está,

Segunda Dica

Um array, uma vez criado, não permite um aumento ou diminuição do seu número de pastas (elementos) sem a sua total destruição, ou seja, realizar estas tarefas NÃO é possível:

```
int [] meuArray = new int[5]; // criado um array com 5 elementos
...
meuArray.length = 8;        // aumentamos o número de elementos para 8
```

Na linha onde se tenta aumentar o tamanho do array, acusará um erro indicando que o atributo length não pode ser modificado, a única maneira de realizarmos o processo é a total reconstrução do array, do seguinte modo:

```
int [] meuArray = new int[5]; // criado um array com 5 elementos
...
meuArray = new int[8];      // reconstrução do array com 8 elementos
```

O problema é que ao fazermos isso todos os elementos do array serão perdidos. Uma maneira de conseguirmos sem perdemos os elementos é criarmos um array "espelho" e após a reconstrução do original, copiarmos os elementos. Considere o seguinte array:

para depois localizá-la dentro da gaveta. Podemos então, declará-la e navegá-la do seguinte modo:

```
String [][] dado = new String[3][2];
// Ou seja, um armário de 3 Gavetas e 2 Pastas em cada gaveta
for (int gaveta = 0; gaveta < dado.length; gaveta ++)
    for (int pasta = 0; pasta < dado[gaveta].length; pasta++)
        dado[gaveta][pasta] = "Gaveta " + gaveta + " pasta " + pasta;
```

Desta vez, criamos diretamente um array de objetos do tipo String que é composto de três gavetas e duas pastas (esta é uma segunda forma de criação dos arrays), feitas a declaração e construção em uma única linha.

Em seguida, navegamos por sobre as gavetas. Dentro desse laço, inserimos mais um para navegar por sobre as pastas que cada gaveta tiver. Em cada pasta encontrada em uma determinada gaveta, colocamos um valor. Para visualizarmos os dados, basta repetir a mesma estrutura:

```
for (int gaveta = 0; gaveta < dado.length; gaveta ++)
    for (int pasta = 0; pasta < dado[gaveta].length; pasta++)
        System.out.println(dado[gaveta][pasta]);
```

Poderíamos continuar nesta idéia indo para três dimensões (uma sala, cada uma com vários armários), quatro dimensões (um andar, cada um com várias salas), cinco, seis e assim por diante, porém vamos parar por aqui, pois acredito que você já entendeu a idéia.

Para quem ainda ficou com dúvida, eis aqui o programa final desta dica:

```
public class Hello
{
    public static void main(String [] args)
    {
        int[] meuArray;
        meuArray = new int[2];
        meuArray[0] = 2; meuArray[1] = 4;
        for (int pos = 0; pos < meuArray.length; pos++)
            System.out.println(meuArray[pos]);

        String [][] dado = new String[3][2];
        for (int gaveta = 0; gaveta < dado.length; gaveta ++)
            for (int pasta = 0; pasta < dado[gaveta].length; pasta++)
                dado[gaveta][pasta] = "Gaveta " + gaveta + " pasta " + pasta;

        for (int gaveta = 0; gaveta < dado.length; gaveta ++)
            for (int pasta = 0; pasta < dado[gaveta].length; pasta++)
                System.out.println(dado[gaveta][pasta]);
    }
}
```

```
int [] meuArray = {6, 7, 4};
System.out.println("Original: " + meuArray[0] + "," + meuArray[1] + "," + meuArray[2]);
```

Essa é uma terceira maneira de declarar, construir e inicializar o array. Em uma única instrução, criamos um array de inteiros com três elementos, sendo que o elemento zero (meuArray[0]) tem o valor seis, o elemento um (meuArray[1]) o valor sete e o elemento dois (meuArray[2]) o valor quatro.

Desejamos aumentar mais dois elementos neste array, para tanto é preciso fazer o seguinte:

```
int [] espelho = meuArray;
meuArray = new int[meuArray.length+2];

for (int pos = 0; pos < espelho.length; pos++)
    meuArray[pos] = espelho[pos];
System.out.println("Aumentado: " + meuArray[0] + "," + meuArray[1] + "," +
    meuArray[2] + "," + meuArray[3] + "," + meuArray[4]);
```

Inicialmente é criado um novo array "espelho" com os mesmos dados do array original. Em seguida, o array original é ampliado em duas posições e por fim cada elemento é copiado do array "espelho". Note que as novas posições são inicializadas com o valor-padrão 0.

Assim, como não é possível adicionar pastas a sua gaveta, também não é possível removê-las, resultando que o mesmo procedimento deverá ser tomado nesta situação. Suponha que desejamos retirar a pasta um que possui o valor sete:

```
espelho = meuArray;
meuArray = new int[meuArray.length-1];
int pos1 = 0;
for (int pos2 = 0; pos2 < espelho.length; pos2++)
    if (pos2 != 1)
        meuArray[pos1++] = espelho[pos2];
System.out.println("Diminuído: " + meuArray[0] + "," + meuArray[1] + "," +
    meuArray[2] + "," + meuArray[3]);
```

Para que não fique nenhuma dúvida, eis aqui o programa final desta dica:

```
public class Hello
{
    public static void main(String [] args)
    {
        int [] meuArray = {6, 7, 4, 3, 2};
        System.out.println("Original: " + meuArray[0] + "," + meuArray[1] +
            "," + meuArray[2]);
        int [] espelho = meuArray;
        meuArray = new int[meuArray.length+3];
        for (int pos = 0; pos < espelho.length; pos++)
            meuArray[pos] = espelho[pos];
        System.out.println("Aumentado: " + meuArray[0] + "," + meuArray[1]
            + "," + meuArray[2] + "," + meuArray[3] + "," + meuArray[4]);
        espelho = meuArray;
        meuArray = new int[meuArray.length-1];
        int pos1 = 0;
        for (int pos2 = 0; pos2 < espelho.length; pos2++)
            if (pos2 != 1)
                meuArray[pos1++] = espelho[pos2];
        System.out.println("Diminuído: " + meuArray[0] + "," + meuArray[1]
            + "," + meuArray[2] + "," + meuArray[3]);
    }
}
```

Terceira Dica

Por fim, existe uma classe que auxilia a utilização de array, ela está no pacote java.util e chama-se arrays (com "s" no final), composta basicamente de cinco métodos auxiliares, sendo eles:

- binarySearch – permite uma pesquisa nos elementos de um determinado array ordenado, retornando um atributo inteiro com a posição deste elemento;
- equals – permite a comparação entre dois arrays, retornando um booleano verdadeiro caso os arrays sejam iguais;
- fill – realiza o preenchimento de todos os elementos de um determinado array;
- sort – faz uma ordenação nos elementos de um determinado array;
- toString – mostra os elementos de um determinado array.

Vamos começar criando um array qualquer:

```
int [] meuArray = {6, 7, 4, 2, 3};
```

Se tentarmos usar o método binarySearch do seguinte modo:

```
System.out.println(java.util.Arrays.binarySearch(meuArray, 3));
```

obtemos como resposta o valor -1, ao invés de quatro que seria a posição correta do elemento, este valor negativo significa que o elemento não foi encontrado. Entretanto, se antes procedermos uma ordenação do tipo:

```
java.util.Arrays.sort(meuArray);
```

e fizermos a pesquisa:

```
System.out.println(java.util.Arrays.binarySearch(meuArray, 3));
```

obtemos como resposta o valor um, que significa que o elemento foi encontrado na posição um do array. Note que isso só foi realizado porque primeiramente ordenamos o array.

Podemos mostrar o array completo do seguinte modo:

```
System.out.println(java.util.Arrays.toString(meuArray));
```

e teremos como resposta [2, 3, 4, 6, 7], ou seja, realmente o valor três está na posição um (lembrando que todo o array começa com posição zero).

Poderíamos criar um "espelho" deste array e verificarmos se ambos são iguais do seguinte modo:

```
int [] espelho = meuArray;
if (java.util.Arrays.equals(meuArray, espelho))
   System.out.println("São iguais");
else
   System.out.println("São diferentes");
```

Por fim, preenchemos todos os elementos do array original com o valor dez e o mostramos da seguinte maneira:

```
java.util.Arrays.fill(meuArray, 10);
System.out.println(java.util.Arrays.toString(meuArray));
```

Note que o uso desta classe auxilia a manipulação dos arrays que agora você já domina. E para que não fique nenhuma dúvida, eis aqui o programa final desta dica:

```
public class Hello
{
    public static void main(String [] args)
    {
        int [] meuArray = {6, 7, 4, 2, 3};
        java.util.Arrays.sort(meuArray);
        System.out.println(java.util.Arrays.binarySearch(meuArray, 3));
        System.out.println(java.util.Arrays.toString(meuArray));
        int [] espelho = meuArray;
        if (java.util.Arrays.equals(meuArray, espelho))
            System.out.println("São iguais");
        else
            System.out.println("São diferentes");
        java.util.Arrays.fill(meuArray, 10);
        System.out.println(java.util.Arrays.toString(meuArray));
    }
}
```