Hortolândia, 25 de fevereiro de 2014.

Aulas 13 e 14 – Estruturas de dados homogêneas em Java (Matrizes e vetores)

Introdução

Estruturas de dados homogêneas são a base de muitas soluções de programação. Através delas podemos manipular grandes quantidades de dados de forma prática eficaz.

A linguagem Java implementa este tipo de estruturas de maneira muito interessante, misturando conceitos comuns a outras linguagens e adicionando utilidades próprias da linguagem. Nesta aula aprenderemos como são implementadas e manipuladas estas estruturas na linguagem Java.

1. Vetores e matrizes em Java

A linguagem Java utiliza uma sintaxe de declaração de vetores e matrizes um pouco diferente de linguagens anteriores, como C e C++. Isto se deve ao fato de que todas variáveis deste tipo em Java serem criadas de forma dinâmica, ou seja, através de alocação de memória no momento da declaração.

Assim, a sintaxe para declarar vetores e matrizes em Java é a seguinte:

```
//declaração vetor
tipo nome[] = new tipo[quantidadeElementos];
Ex:
float notasAlunos[] = new float[50];//armazena até 50 notas
//declaração matriz 2d
tipo nome[][] = new tipo[quantidade1][quantidade2];
Ex:
char jogoVelha[][] = new char[3][3];
//declaração matriz 3d
tipo nome[][][] = new tipo[qtde1][qtde2][qtde3];
...
```

O termo **new** serve para inicializar as variáveis e definir quantidades de elementos para cada dimensão. Também é possível inicializar vetores e matrizes já inserindo valores nas mesmas:

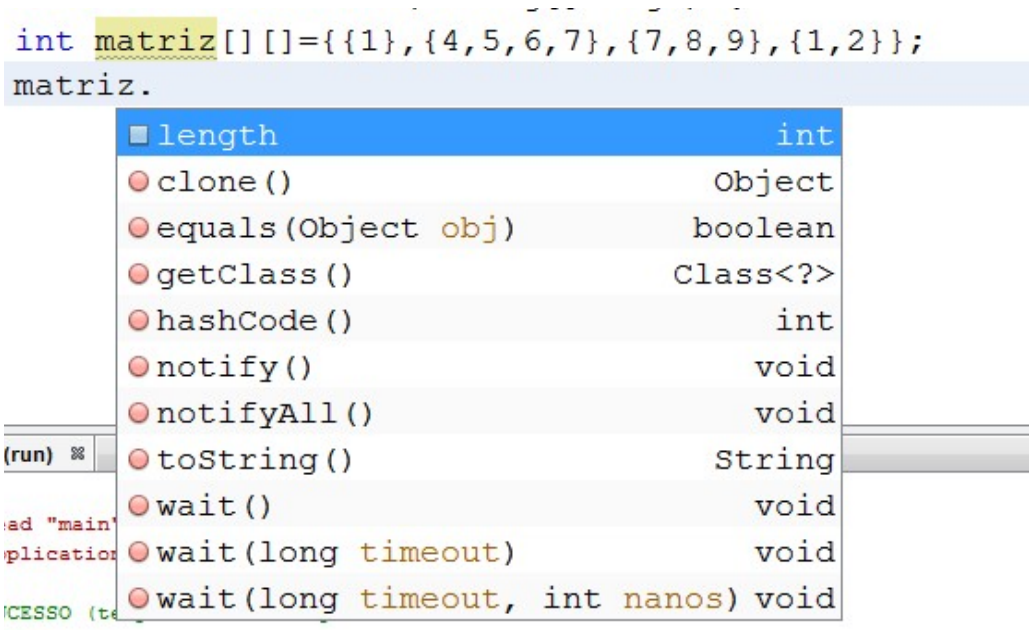
```
//inicialização com elementos vetor
tipo nome[] = {elemento1,elemento2,elemento3,...};
Ex:
String diaSemana[] = {"D","S","T","Q","Q","S","S"};
//inicialização matriz 2d com valores
tipo nome[][] = {{e1,e2,...}{e1,e2,...}{e1,e2,...}}
Ex:
```

```
int matriz[][]={{1,2,3},
                {4,5,6},
                {7,8,9}};//matriz de 3 x 3 elementos
...
```

Um dado interessante é que a linguagem Java permite matrizes terem diferentes quantidades de elementos nas diversas dimensões, apesar de não ser muito comum a utilização dessa funcionalidade. Veja um exemplo de inicialização de matriz 2d com diversas quantidades de elementos em cada linha:

```
//inicialização de matriz2d com quantidades
//variadas de elementos em cada linha
int matriz[][]={{1},
                {4,5,6,7},
                {7,8,9},
                {1,2}};
```

Após a inicialização, além de armazenar dados, vetores e matrizes em Java possuem diversos métodos auxiliares para lidar com esse tipo de informação, como por exemplo saber o tamanho de um vetor/matriz, cloná-lo, etc. Eles podem ser acessados colocando o nome da variável seguido de um sinal de ponto (.). Veja:



Matrizes e vetores em Java não são apenas variáveis comuns. Elas podem ser passadas para outras matrizes e vetores via operações de atribuição. No entanto, quando uma matriz recebe outra, ao invés de fazer uma cópia dos dados, ela na verdade recebe a referência em memória da outra matriz/vetor, algo semelhante à utilização de ponteiros em linguagem C/C++. Assim, a instrução em negrito:

```
int vetor1[] = {1,2,3,4,5};
```

```
int vetor2[] = vetor1;
```

faz com que a variável vetor2 compartilhe o conteúdo da variável vetor1. Assim, caso seja alterado o valor de qualquer uma delas, essa alteração refletirá instantaneamente na outra. Na prática, as duas armazenam dados no mesmo local da memória. Se quisermos realmente copiar de forma independente um vetor sem criar essa ligação, podemos usar o método clone( ) para fazer isso. Veja:

```
int vetor1[] = {1,2,3,4,5};

int vetor2[] = vetor1.clone();
```

Nesse caso, as duas variáveis conterão os mesmos valores, mas cada uma terá seu espaço separado na memória. Assim, se alterarmos um elemento de vetor1, o vetor2 não será afetado. É muito importante prestar atenção a esse detalhe, caso contrário podemos perder nossos dados e corrompê-los em operações diversas sobre os mesmos.

2. Percurso de vetores e matrizes em Java

Para percorrer vetores e matrizes em Java, o mais comum é a utilização de um laço, geralmente do tipo for. Além do for clássico, a linguagem Java possui um for expandido que permite o percurso de todos elementos de um vetor/matriz de forma simplificada. A seguir é mostrado um percurso sobre um vetor colocando o valor 5 em todas as suas posições, usando um laço for comum e depois o mesmo procedimento com um for expandido:

```
int vetor1[] = new int[10];

//for comum

for(int i=0;i<10;i++){

    vetor1[i]=5;//coloca valor 5 na posicao atual do vetor

}

//for expandido

for(int i:vetor1){

    i=5;

}
```

Dessa maneira, pode-se percorrer os vetores e matrizes tanto para leitura quanto gravação de dados. Outros laços podem ser usados também, apesar de não terem geralmente a praticidade do **for** para esse tipo de tarefa.

3. Matrizes e vetores heterogêneos em Java

Uma curiosidade que será depois explicada pelos conceitos de orientação a objetos, é que em Java um vetor ou uma matriz do tipo especial Object pode receber qualquer tipo de informação em cada um de seus elementos. Veja:

```
Object misturado[] = {1,2.5,"abc",'d'};
```

5. Conclusões

A presente aula tratou de diversos aspectos da **linguagem de programação Java**. Dentre eles, podem-se destacar: a declaração e a inicialização de estruturas de dados homogêneas e sua manipulação, peculiaridades da linguagem e a sintaxes de cada uma delas.

6. Referências Bibliográficas

[1] DEITEL, H.M., DEITEL, P.J. **Java – Como programar**. Terceira edição. Porto Alegre: Bookman Editora, 2001.

[2] SANTOS, Rafael. **Introdução à Programação orientada a objetos usando Java**. 8ª Reimpressão. Rio de Janeiro: Campus - Elsevier, 2003.