codecademy

Matrizes bidimensionais

Instruções de iteração aninhadas

Em Java, instruções de iteração aninhadas são instruções de iteração que aparecem no corpo de outra instrução de iteração. Quando um loop é aninhado dentro de outro loop, o loop interno deve completar todas as suas iterações antes que o loop externo possa continuar.

```
for(int outer = 0; outer < 3; outer++){
    System.out.println("The outer index
is: " + outer);
    for(int inner = 0; inner < 4;
inner++){
        System.out.println("\tThe inner
index is: " + inner);
    }
}
```

Declarando matrizes 2D

Em Java, arrays 2D são armazenados como arrays de arrays. Portanto, a maneira como as matrizes 2D são declaradas é semelhante a objetos de matriz 1D. Arrays 2D são declarados definindo um tipo de dados seguido por dois conjuntos de colchetes.

```
int[][] twoDIntArray;
String[][] twoDStringArray;
double[][] twoDDoubleArray;
```

Acessando elementos de matriz 2D

Em Java, ao acessar o elemento de um array 2D usando `arr[first][second]`, o `first` índice pode ser pensado como a linha desejada e o `second` índice é usado para a coluna desejada. Assim como as matrizes 1D, as matrizes 2D são indexadas a partir de `0`.

```
//Given a 2d array called `arr` which
stores `int` values
int[][] arr = {{1,2,3},
               {4,5,6}};

//We can get the value `4` by using
int retrieved = arr[1][0];
```

Listas de inicializadores

Em Java, as listas de inicializadores podem ser usadas para fornecer rapidamente valores iniciais para arrays 2D. Isso pode ser feito de duas maneiras diferentes.

1. Se o array ainda não foi declarado, um novo array pode ser declarado e inicializado na mesma etapa usando colchetes.
2. If the array has already been declared, the `new` keyword along with the data type must be used in order to use an initializer list

```
// Method one: declaring and
intitializing at the same time
double[][] doubleValues = {{1.5, 2.6,
3.7}, {7.5, 6.4, 5.3}, {9.8,  8.7, 7.6},
{3.6, 5.7, 7.8}};

// Method two: declaring and initializing
separately:
String[][] stringValues;
stringValues = new String[][]
{{"working", "with"}, {"2D", "arrays"},
{"is", "fun"}};
```

Modify 2D Array Elements

In Java, elements in a 2D array can be modified in a similar fashion to modifying elements in a 1D array. Setting `arr[i][j]` equal to a new value will modify the element in row `i` column `j` of the array `arr`.

```
double[][] doubleValues = {{1.5, 2.6,
3.7}, {7.5, 6.4, 5.3}, {9.8,  8.7, 7.6},
{3.6, 5.7, 7.8}};

doubleValues[2][2] = 100.5;
// This will change the value 7.6 to
100.5
```

Row-Major Order

"Row-major order" refers to an ordering of 2D array elements where traversal occurs across each row - from the top left corner to the bottom right. In Java, row major ordering can be implemented by having nested loops where the outer loop variable iterates through the rows and the inner loop variable iterates through the columns. Note that inside these loops, when accessing elements, the variable used in the outer loop will be used as the first index, and the inner loop variable will be used as the second index.

```
for(int i = 0; i < matrix.length; i++) {
    for(int j = 0; j < matrix[i].length;
j++) {
        System.out.println(matrix[i][j]);
    }
}
```

Column-Major Order

"Column-major order" refers to an ordering of 2D array elements where traversal occurs down each column - from the top left corner to the bottom right. In Java, column major ordering can be implemented by having nested loops where the outer loop variable iterates through the columns and the inner loop variable iterates through the rows. Note that inside these loops, when accessing elements, the variable used in the outer loop will be used as the second index, and the inner loop variable will be used as the first index.

```
for(int i = 0; i < matrix[0].length; i++)
{
    for(int j = 0; j < matrix.length;
j++) {
        System.out.println(matrix[j][i]);
    }
}
```

Traversing With Enhanced For Loops

In Java, enhanced for loops can be used to traverse 2D arrays. Because enhanced for loops have no index variable, they are better used in situations where you only care about the values of the 2D array - not the location of those values

```
for(String[] rowOfStrings
: twoDStringArray) {
    for(String s : rowOfStrings) {
        System.out.println(s);
    }
}
```