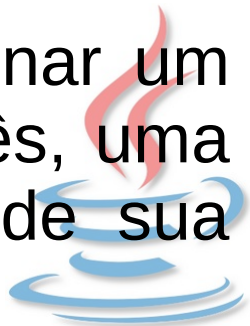
Tipos de dados e operadores

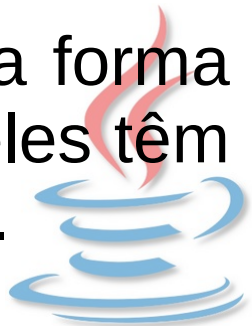
Arrays

- Um array é um conjunto de variáveis do mesmo tipo, referenciadas por um nome comum.
- Em Java, os arrays podem ter uma ou mais dimensões, embora o array unidimensional seja o mais popular. Os arrays são usados para vários fins, porque oferecem um meio conveniente de agrupar variáveis relacionadas.
- Por exemplo, você pode usar um array para armazenar um registro da temperatura máxima diária durante um mês, uma lista de médias de preços de ações ou uma lista de sua coleção de livros de programação.



Arrays

- A principal vantagem de um array é que ele organiza os dados de tal forma que é fácil tratá-los. Por exemplo, se você tiver um array contendo as rendas de um determinado grupo de famílias, será fácil calcular a renda média percorrendo-o.
- Os arrays também organizam os dados de forma que eles possam ser facilmente classificados.
- Embora os arrays Java possam ser usados da mesma forma que os arrays de outras linguagens de programação, eles têm um atributo especial: são implementados como objetos.



Arrays

- Esse fato é uma das razões para uma discussão dos arrays ter sido adiada até os objetos serem introduzidos.
- Na implementação de arrays como objetos, muitas vantagens importantes são obtidas e uma delas, que não é mesmo importante, é que os arrays não usados podem ser alvo da coleta de lixo.

Arrays unidimensionais

- Um arrays unidimensional é uma lista de variáveis relacionadas. Essas listas comuns em programação.
- Por exemplo, você pode usar um array unidimensional para armazenar os números de conta dos usuários ativos em uma rede.
- Outro array poderia ser usado para armazenar as médias de rebatidas atuais de um time de baseball.

Arrays unidimensionais

- Já que os arrays são implementados como objetos, a criação de um array é um processo de duas etapas. Primeiro, você declara uma variável de referência de array. Depois, aloca memória para o array, atribuindo uma referência dessa memória à variável de array.
- Portanto, os arrays Java são alocados dinamicamente com o uso do operador **new**.
- Um elemento individual de um array é acessado com o uso de um índice. Um índice descreve a posições de um elemento dentro de um array. Em Java todos os arrays têm zero como o índice de seu primeiro elemento.

Arrays multidimensionais

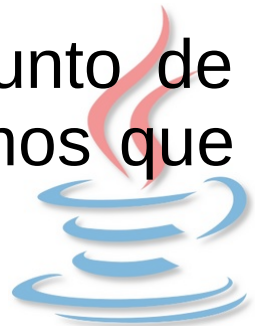
- Embora o array unidimensional seja o mais usado em programação, os arrays multidimensionais(arrays de duas ou mais dimensões) certamente não são raros.
- Em Java, o array multidimensional é um array composto por arrays.
- A forma mais simples de array multidimensional é o array bidimensional.

Arrays multidimensionais

- Um array bidimensional é , na verdade, uma lista de array unidimensionais.
- Para declarar um array bidimensional de tipo inteiro e tamanho 10, 20 chamdo **table**, você escreveria:

```
Int table[][] = new int[10][20];
```

- Java insere cada dimensão em seu próprio conjunto de colchetes. Para acessar o ponto 3,5 de **table**, temos que usar **table[3][5]**.



Arrays irregulares

- Quando alocamos memória para um array multidimensional, só temos que especificar a memória da primeira dimensão(a da extrema esquerda).
- As outras dimensões podem ser alocadas separadamente.
- Embora não haja vantagens em alocar individualmente os conjuntos da segunda dimensão, pode haver em outras.
- Por exemplo, quando alocamos as dimensões separadamente, não precisamos alocar o mesmo número de elementos para cada índice.

Arrays de três ou mais dimensões

- Java permite arrays com mais de duas dimensões.
- Aqui está a forma geral de declaração de um array multidimensional:

Tipo nome[ ][ ]...[ ] = new *tipo[tam1][tam2]...[tamN];*

- Por exemplo, a declaração a seguir cria um array tridimensional inteiro de 4 x 10 x 3

```
Int multidim[ ][ ][ ] = new int [4][10][3];
```

Inicializando arrays multidimensionais

- Um array multidimensional pode ser inicializado com a inserção da lista de inicializadores de cada dimensão dentro de seu próprio conjunto de chaves.
- Por exemplo, a forma geral de inicialização de um array bidimensional é mostrado abaixo:

```
especificador-tipo-nome_array[ ][ ] = {
   {val,val,val,...val},
   {val,val,val,...val},
    .....
   {val,val,val,...val},
};
```

Declaração de array alternativo

- A forma comum de declaração de um array, como já vimos, é:

tipo nome [ ] = new int[3];

- Há uma segunda forma que pode ser usada na declaração de um array:

tipo[ ] nome = new int[3];

- *Em Java são permitidas as duas representações. Ambas são reconhecidas pela JVM e executadas.*

Usando Lenght

- Já que os arrays são implementados como objetos, cada array tem uma variável de instância **lenght** associada que contém o número de elementos que ele pode conter.
- Em outras palavras, **lenght** contém o tamanho do array propriamente dito.
- Exercício da pg 147, demonstra a utilização do **lenght.**

O laço for-each

- No trabalho com arrays, é comum encontramos situações em que um array deve ser examinado do início ao fim, elemento a elemento.
- Por exemplo, para calcularmos a soma dos valores contidos em uma array, cada elemento do array deve ser examinado. A mesma situação ocorre no cálculo de uma média, na busca de um valor, na cópia de um array e assim por diante.
- Já que essas operações do tipo “início ao fim” são tão comuns. Java define uma segunda forma do laço **for** que otimiza a operação.

O laço for-each

- A segunda forma de **for** implementa um laço de estilo "*for-each*".
- Um laço for-each percorre um conjunto de objetos, como um array, de maneira rigorosamente sequencial, do início ao fim.
- Nos últimos anos, os laços de estilos for-each ganharam popularidade tanto entre projetistas quanto entre programadores de linguagens de computador.

O laço for-each

- Originalmente Java não oferecia um laço de estilo for-each.
- No entanto, com o lançamento de JDK 5, o laço **for** foi melhorado para fornecer essa opção.
- O estilo for-each de **for** também é chamado de laço **for** melhorado.
- A forma geral do laço **for** de estilo for-each é:

for(tipo var-iter: conjunto) bloco de instruções

- Exercicio da pg.155 para entender melhor o funcionamento do laço for-each.

O laço for-each

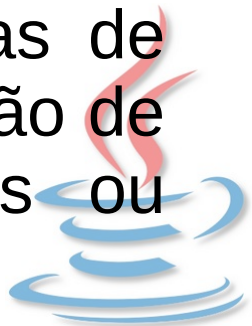
- O laço **for** melhorado também funciona em arrays multidimensionais.
- Lembre-se, no entanto, de que, em Java, os arrays multidimensionais são arrays de arrays.
- Esse é um detalhe importante na iteração por um array multidimensional, porque cada iteração obtém o array seguinte e não um elemento individual.
- Para entender as implicações desse fato, considere o programa a seguir. Ele usa laços **for** aninhados para obter os elementos de um array bidimensional por ordem de linha, da primeira à última.
- Exercício da pg. 157

Aplicando o laço for melhorado

- Já que o laço **for** de estilo for-each só pode percorrer o array sequencialmente, do início ao fim, você deve estar achando que seu uso é limitado.
- No entanto, isso não é verdade.
- Vários algoritmos precisam exatamente desse mecanismo.
- Um dos mais comuns é a busca.
- Exercício da pg. 158, usa um laço **for** para procurar um valor em um array não classificado.

Operadores bitwise

- Além de oferecer os operadores aritméticos, relacionais e lógicos, Java também fornece operadores adicionais que expandem o conjunto de problema ao qual Java pode ser aplicado.
- As operações bitwise pode ser usadas em valores do tipo **long,int,short,char** ou byte, mas não podem ser usadas com tipos: **boolean, float, double ou tipos de classe**.
- Essas operações são importantes em várias tarefas de programação de nível de sistema em que a informação de status de um dispositivo devem ser consultadas ou construídas.



O operador ternário(?)

- Um dos operadores mais fascinantes de Java é o operador ?.
- É usado para substituir instruções **if-else** que têm esta forma geral:

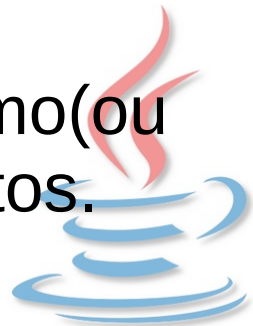
```
if(condição)
  var= expression1;
else
  var= expression2;
```

O operador ternário(?)

- Aqui, o valor atribuído a *var* depende do resultado da condição que controla **if**.
- O operador ? é chamado de operador ternário porque requer três operandos. Ele tem a forma geral:

Exp1 ? Exp2 : Exp3

- Exp1 é uma expressão booleana e Exp2 e Exp3 são expressões de qualquer tipo menos void.
- No entanto, o tipo de Exp2 e Exp3 deve ser o mesmo(ou compatível). Observe o uso e a posição dos dois pontos.
- Exercício da pg. 177



O operador ternário(?)

- Você não precisa atribuir o valor produzido pelo operador ? a uma variável.
- Poderia usar o valor como argumento em uma chamada a um método.
- Ou, se as expressões forem todas de tipo boolean, o operados ? pode ser usado como a expressão condicional em um laço ou instrução if.
- Por exemplo, este é o programa anterior reescrito de maneira pouco mais eficiente é demonstrado no exercício da pg. 178