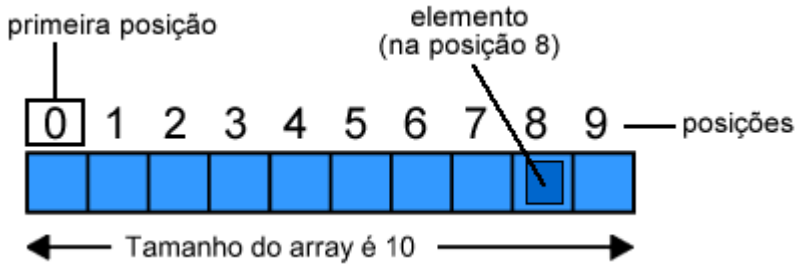
AULA 6 - ARRAYS

Arrays são objetos que armazenam diversas variáveis do mesmo tipo. Eles podem conter variáveis de referência primitivas ou de objeto, mas o array propriamente dito sempre será um objeto da pilha, mesmo se for declarado para armazenar elementos primitivos. Em outras palavras, não existe algo como um array primitivo, mas você pode criar um array de elementos primitivos.



Array de 10 elementos

Cada item em um array é chamado de elemento, e cada elemento é acessado pela posição numérica. Como na ilustração acima as posições são numerdas a partir do 0. O 9th elemento, por exemplo, é acessado na posição 8.

O programa a seguir cria um array de inteiros, atribui algumas posições a ele, e imprime cada valor à saída padrão.

```
class ArrayDemo {

    public static void main(String[] args) {

        int[] meuArray;             // declara um array de inteiros

        meuArray = new int[10];     // aloca a memória para 10 inteiros

        meuArray[0] = 100; // inicializa o primeiro elemento
        meuArray[1] = 200; // inicializa o segundo elemento
        meuArray[2] = 300; // etc.
        meuArray[3] = 400;
        meuArray[4] = 500;
        meuArray[5] = 600;
        meuArray[6] = 700;
        meuArray[7] = 800;
        meuArray[8] = 900;
        meuArray[9] = 1000;

        System.out.println("Elemento na posicao 0: " + meuArray[0]);
        System.out.println("Elemento na posicao 1: " + meuArray[1]);
        System.out.println("Elemento na posicao 2: " + meuArray[2]);
        System.out.println("Elemento na posicao 3: " + meuArray[3]);
        System.out.println("Elemento na posicao 4: " + meuArray[4]);
        System.out.println("Elemento na posicao 5: " + meuArray[5]);
        System.out.println("Elemento na posicao 6: " + meuArray[6]);
        System.out.println("Elemento na posicao 7: " + meuArray[7]);
        System.out.println("Elemento na posicao 8: " + meuArray[8]);
        System.out.println("Elemento na posicao 9: " + meuArray[9]);
    }
}
```

A saída desse programa é:

```
Elemento na posicao 0: 100
Elemento na posicao 1: 200
Elemento na posicao 2: 300
Elemento na posicao 3: 400
Elemento na posicao 4: 500
Elemento na posicao 5: 600
Elemento na posicao 6: 700
Elemento na posicao 7: 800
Elemento na posicao 8: 900
Elemento na posicao 9: 1000
```

Em uma situação de programação real, você provavelmente utilizaria umas das construções fazendo loopings (laços) para preencher o seu array, pois seria melhor do que escrever cada linha individualmente como mostrado acima. Entretanto, este exemplo ilustra claramente a sintaxe da disposição.

Declarando um array

O programa acima declara meuArray com a seguinte linha de código:

```
int[] meuArray;          // declara um array de inteiros
```

Como declarações para variáveis de outros tipos, uma declaração de array tem dos componentes: o tipo do array e o nome do array. O tipo de um array é escrito como tipo[], onde tipo é o tipo de dados dos elementos contidos no array; os colchetes são caracteres especiais que indicam que essa variável na verdade é um array.

Como com as variáveis de outros tipos, a declaração não cria realmente o array - diz simplesmente ao compilador que esta variável terá um array do tipo especificado.

Do mesmo modo você pode declarar arrays de outros tipos:

```
byte[] meuArrayDeBytes;
short[] meuArrayDeShorts;
long[] meuArrayDeLongs;
float[] meuArrayDeFloats;
double[] meuArrayDeDoubles;
boolean[] meuArrayDeBooleans;
char[] meuArrayDeChars;
String[] meuArrayDeStrings;
```

Exemplos:

1. boolean resultado[] = { true, false, true, false };
2. double [] grades = {100, 90, 80, 75};
3. String diassemana[] = {"Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado", "Domingo"};

Você também pode colocar os colchetes após o nome do array:

```
float meuArrayDeFloats[]; // forma menos utilizada
```

Essa forma é menos utilizada, pois com os colchetes antes do nome do array fica mais fácil entender de que se trata de um array.

Criando, inicializando, e acessando um Array

Uma das formas de se criar um array é utilizando o operador new. A declaração seguinte no programa ArrayDemo, aloca um array com memória para dez elementos do tipo inteiro e atribui o array à variável meuArray.

```
meuArray  = new int[10];  // cria um array de inteiros
```

Se esta declaração faltasse, o compilador imprimiria um erro como o seguinte, e a compilação falharia:

```
ArrayDemo.java:9: variable meuArray might not have been initialized
```

As próximas linhas atribuem valores para cada elemento do array:

```
meuArray[0] = 100; // inicializa o primeiro elemento
meuArray[1] = 200; // inicializa o segundo elemento
meuArray[2] = 300; // etc.
```

Cada elemento do array é acessado por seu índice numérico:

```
System.out.println("Elemento na posicao 0: " + meuArray[0]);
System.out.println("Elemento na posicao 1: " + meuArray[1]);
System.out.println("Elemento na posicao 2: " + meuArray[2]);
```

Alternativamente, você pode usar atalhos de sintaxe para criar e inicializar um array:

```
int[] meuArray = {100, 200, 300, 400, 500, 600, 700, 800, 900, 1000};
```

Aqui o tamanho do array é determinado pelo número de valores informados entre { e }.

Você também pode declarar um array de arrays (também conhecido como array multidimensional), utilizando dois ou mais conjuntos de colchetes [][]. Cada elemento, deve conseqüentemente ser alcançado por um número correspondente de valores de índice.

Na linguagem de programação de Java, um array multidimensional é simplesmente um array cujos os elementos são outros arrays.

Um exemplo de array multidimensional seria:

```
class ArrayMultiDemo {

   public static void main(String[] args) {

      String[][] nomes = {{"Mr. ", "Mrs. ", "Ms. "},{"Smith", "Jones"}};

      System.out.println(nomes[0][0] + nomes[1][0]); //Mr. Smith
      System.out.println(nomes[0][2] + nomes[1][1]); //Ms. Jones

   }
}
```

A saída desse programa é:
Mr. Smith
Ms. Jones

Por fim, você pode utilizar a propriedade **length** para determinar o tamanho de um array. O código

```
System.out.println (meuArray.length);
```

imprimirá o tamanho do array meuArray.

Classe array

Existe uma classe que auxilia a utilização de array, que está localizada no pacote java.util, tem o nome de Arrays, e possui os seguintes métodos:

- **binarySearch** – permite uma pesquisa nos elementos de um determinado array ordenado, retornando um atributo inteiro com a posição deste elemento;
- **equal** – permite a comparação entre dois arrays, retornando um booleao verdadeiro(true), caso os array 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.

Suponha que desejamos descobrir no objeto meuArray a posição que contem o elemento 4552. O método binarySearch da classe Arrays é o mais indicado para isto, porém, no array meuArray não está ordenado, sendo uma condição para utilização do método binarySearch . O primeiro passo então seria ordenar o meuArray utilizando o método sort:

```
int[] meuArray = {22,2,242,4552,36};

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

Agora podemos utilizar o método binarySearch desta maneira:

```
   java.util.Arrays.binarySearch(meuArray,4552);
```

Alterando o código exemplo temos:

```
public class MeuArray {
  public static void main( String args[]){
    int[] meuArray;
    meuArray = new int[10];
    meuArray[0] = 22;
    meuArray[1] = 2;
    meuArray[2] = 242;
    meuArray[3] = 4552;
    meuArray[4] = 36;

   java.util.Arrays.sort(meuArray);

   for (int i = 0; i < meuArray.length; i++){
      System.out.println("posição " + i + " = " + meuArray[i]);
    }

  System.out.println("posição do elemento 4552 :" + java.util.Arrays.binarySearch(meuArray,4552));
 }
}
```

Vamos agora construir um objeto array idêntico ao meuArray da seguinte forma:

```
      int [] espelho =  meuArray;
```

O método equals da classe Arrays, permite comparar dois arrays retornando “true” ou “false”, como mostra o código a seguir::

```
   if (java.util.Arrays.equals(meuArray,espelho)){
         System.out.println("são identicos...");
   }else{
         System.out.println("são diferentes...");
   }
```

O método fill da classe Arrays, preenche todos os elementos de um array com um dado valor, da seguinte forma:

```
    java.util.Arrays.fill(espelho,5522);
```

Percorrendo todos os elementos deste array verificamos que os valores são os mesmos para todas as posições do array espelho:

posição 0 = 5522
posição 1 = 5522
posição 2 = 5522
posição 3 = 5522
posição 4 = 5522
posição 5 = 5522
posição 6 = 5522
posição 7 = 5522
posição 8 = 5522
posição 9 = 5522

Para finalizar, o método toString() da classe Arrays permite exibir todos elementos de um objeto array, como mostra o código a seguir:

System.out.println("Conteúdo do objeto array : " + java.util.Arrays.toString(meuArray));

```
public class MeuArray {

  public static void main( String args[]){
    int[] meuArray;
    meuArray = new int[10];
    meuArray[0] = 22;
    meuArray[1] = 2;
    meuArray[2] = 242;
    meuArray[3] = 4552;
    meuArray[4] = 36;
    java.util.Arrays.sort(meuArray);
    for (int i = 0; i < meuArray.length; i++){
       System.out.println("posiÃ§Ã£o " + i + " = " + meuArray[i]);
    }
    System.out.println("Conteúdo do objeto array : " +
java.util.Arrays.toString(meuArray));

    System.out.println("posição do elemento 4552 :" +
java.util.Arrays.binarySearch(meuArray,4552));

    int [] espelho = meuArray;
    if (java.util.Arrays.equals(meuArray,espelho)){
       System.out.println("são identicos...");
    else{
       System.out.println("são diferentes...");
    }

    java.util.Arrays.fill(espelho,5522);

    for (int j = 0; j < meuArray.length; j++){
      System.out.println("posição " + j + " = " + meuArray[j]);
    }

    System.out.println("Conteúdo do objeto array : " +
java.util.Arrays.toString(meuArray));

 }

}
```

A saída produzida no terminal é a que segue:

Conteúdo do objeto array : [0, 0, 0, 0, 0, 2, 22, 36, 242, 4552]