

INSTITUTO MAUÁ DE TECNOLOGIA



Linguagens I

Array

Profº. Tiago Sanches da Silva

Arrays

Arrays

Dentro de um bloco, podemos declarar diversas variáveis e usá-las:

```
int idade1;  
int idade2;  
int idade3;  
int idade4;
```

Isso pode se tornar um problema quando precisamos mudar a quantidade de variáveis a serem declaradas de acordo com um parâmetro.

Para facilitar esse tipo de caso podemos declarar um **vetor (array)** de inteiros:

```
int[] idades;
```

Arrays

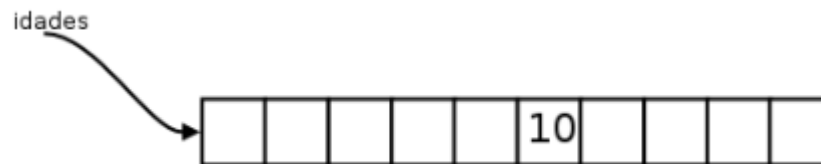
O `int[]` é um tipo. Um **array** é sempre um objeto, portanto, a variável `idades` é uma referência. Vamos precisar criar um objeto para poder usar a array. Como criamos o objeto-array?

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

O que fizemos foi criar um array de `int` de 10 posições e atribuir o endereço no qual ela foi criada. Podemos ainda acessar as posições do array:

```
idades[5] = 10;
```

Resultando em:



Arrays

Trecho completo:

```
int[] idades;  
  
idades = new int[10];  
  
idades[5] = 10;
```

O código acima altera a sexta posição do array. No Java, os índices do array vão de **0** a **n-1**, onde **n** é o tamanho dado no momento em que você criou o array. Se você tentar acessar uma posição fora desse alcance, um erro ocorrerá durante a execução.

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
```

Array de objetos

Array de objetos

Mesmo que seja comum ouvirmos a expressão “**array de objetos**”, temos que ter em mente que na verdade possuímos um **array** de referências para os objetos, e portanto eles precisam ser criados individualmente.

```
Conta[] minhasContas;  
minhasContas = new Conta[10];
```

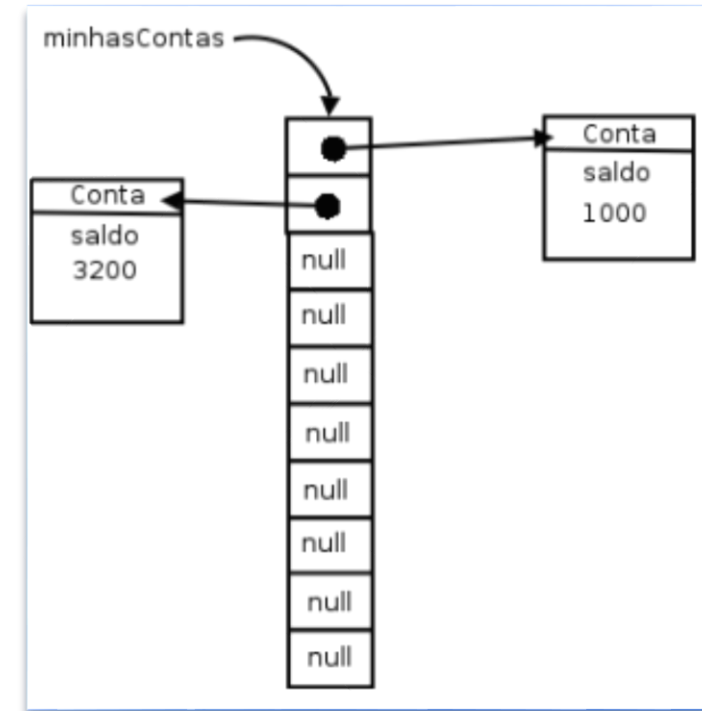
Quantas objetos contas foram criadas aqui?

Na verdade, nenhuma. Foram criados 10 espaços que você pode utilizar para guardar uma referência a uma Conta. Por enquanto, eles se referenciam para lugar nenhum (null).

Array de objetos

Você deve popular seu array antes.

```
Conta contaNova = new Conta();  
contaNova.saldo = 1000.0;  
minhasContas[0] = contaNova;
```

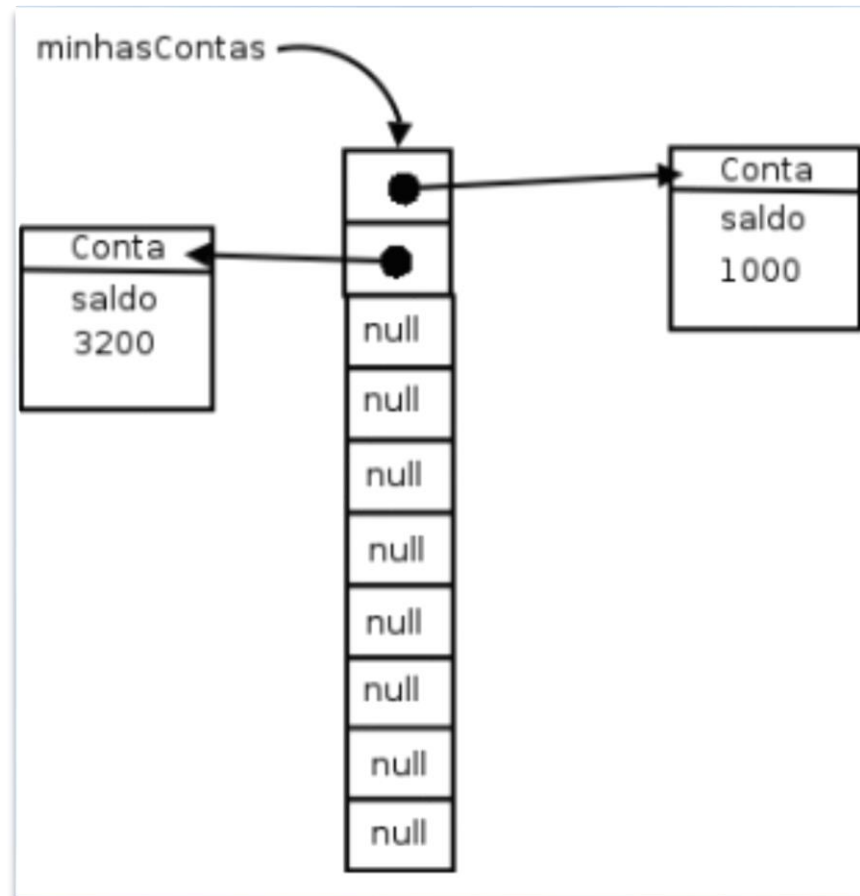


Ou você pode fazer assim:

```
minhasContas[1] = new Conta();  
minhasContas[1].saldo = 3200.0;
```


Array de objetos

Uma **array** de tipos primitivos guarda valores, uma array de objetos guarda **referências**.



Percorrendo um array

Percorrer um **array** é muito simples, basta utilizar alguma estrutura de repetição. Sugestões?

```
public static void main(String args[]) {  
    int[] idades = new int[10];  
    for (int i = 0; i < 10; i++) {  
        idades[i] = i * 10;  
    }  
    for (int i = 0; i < 10; i++) {  
        System.out.println(idades[i]);  
    }  
}
```

Percorrendo um array

Não é incomum recebermos um array como argumento em um método, neste caso como saberemos o tamanho dele?

```
void imprimeArray(int[] array) {  
  
    for (int i = 0; i < ???; i++) {  
        System.out.println(array[i]);  
    }  
}
```

Percorrendo um array

Como dito anteriormente, todo **array** é um objeto, portanto pode possuir atributos e métodos.

Todo **array** em Java tem um atributo que se chama **length**, e você pode acessá-lo para saber o tamanho do **array** ao qual você está se referenciando naquele momento:

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

Array não mudam de tamanho!

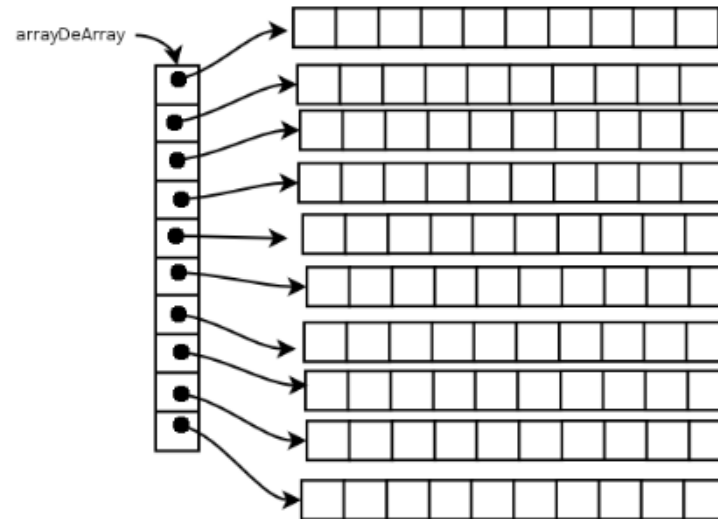
A partir do momento que um **array** foi criado, não pode mudar de tamanho.

Se você precisar de mais espaço, será necessário criar um novo **array** e, antes de se referir a ele, copie os elementos do **array** antigo.

Caso seja necessário a mudança de tamanho dinamicamente, devemos usar listas e coleções do Java. (Veremos posteriormente)

Arrays

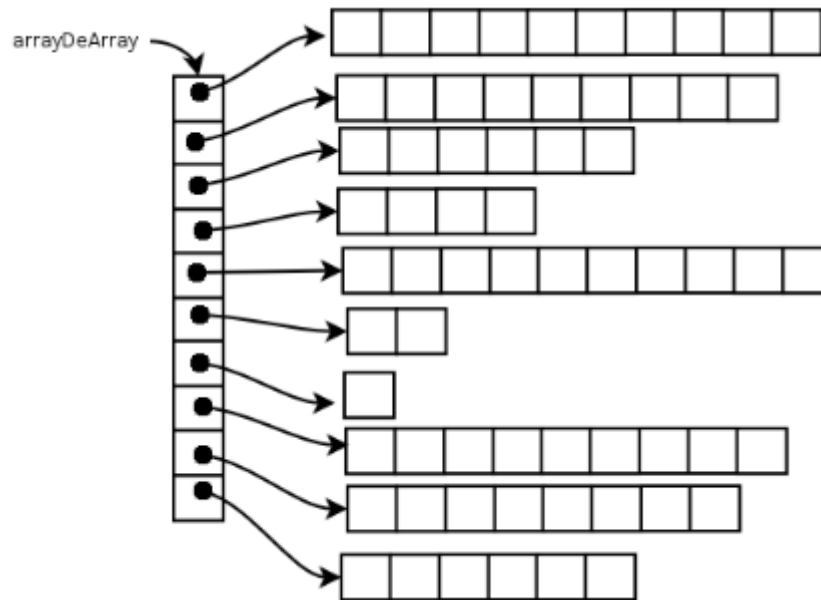
Arrays podem ter mais de uma dimensão. Isto é, em vez de termos um **array** de 10 contas, podemos ter um **array** de 10 por 10 contas e você pode acessar a conta na posição da coluna x e linha y. Na verdade, um **array** bidimensional em Java é um **array** de **arrays**.



```
int[][] matriz = new int[3][2];
```

Arrays

Um **array** bidimensional não precisa ser retangular, isto é, cada linha pode ter um número diferente de colunas.



Como seria isso? Como é a implementação? Pesquisem!

Input

Um dos pontos fortes do Java é o enorme bibliotecas de código disponível para você.

Tudo o que você precisa fazer é referenciar qual biblioteca você deseja usar e, em seguida, chamar um método. Uma classe realmente útil que manipula a entrada de um usuário é chamada de classe **Scanner**. A classe **Scanner** pode ser encontrada na biblioteca **java.util**.

```
import java.util.Scanner;
```

Input

Primeiramente criamos um objeto que seja do tipo `Scanner` e no construtor da classe enviamos o parâmetro `System.in` que é um `InputStream` (não se preocupe, veremos mais tarde).

Após a criação basta utilizar as função **next** para obter e converter a entrada digitada pelo usuário. Ex.:

```
import java.util.Scanner;
Scanner scan = new Scanner(System.in);
String s = scan.next();
int i = scan.nextInt();
```

Input

A função `next` irá capturar tudo que foi digitado até ser pressionado a tecla **Enter**. E então irá parsear e tentar transformar a entrada no tipo de variável de retorno da função, ou seja, `nextInt()` tentará transformar o que foi digitado pelo usuário em um valor inteiro.

```
import java.util.Scanner;
Scanner scan = new Scanner(System.in);
String s = scan.next();
int i = scan.nextInt();
```

Verifique quais são os tipos de entradas que podemos ter.

Concessionária

Como podemos melhorar nosso programa com o que aprendemos hoje?

Perguntas?