

Vetores

Arrays

- ❑ **O problema:** 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

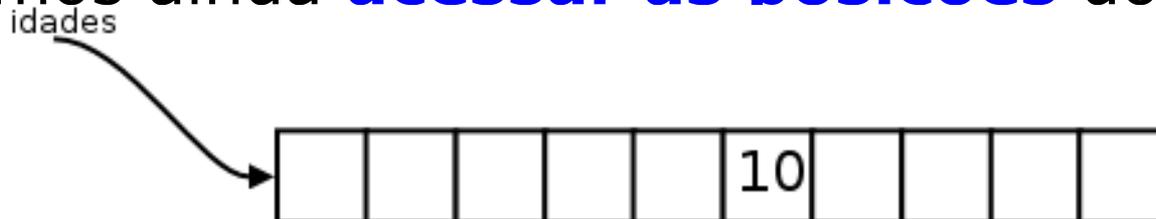
- O **int[]** é um **tipo**. Uma array é **sempre um objeto**, portanto, a variável `idades` é uma **referência**;

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

- O que fizemos foi criar uma array de `int` de 10 posições e **atribuir o endereço** no qual ela foi criada.

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

- Podemos ainda **acessar as posições** do array:



Arrays

- ❑ 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: 10
    at ArrayIndexOutOfBoundsExceptionTeste.main(ArrayIndexOutOfBoundsExceptionTeste.java:5)
```

- ❑ E se usarmos uma **variável** para definir o tamanho do array?

```
int numerosDoBilhete[] = new int[n];
```

Arrays de Referências

- ❑ Quando criamos uma array de alguma **classe**, ela **possui referências**.
- ❑ O objeto está na **memória principal** e, no array, só ficam guardadas as **referências** (endereços).

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

- ❑ Quantas contas foram criadas aqui?
- ❑ **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)**.

Arrays de Referências

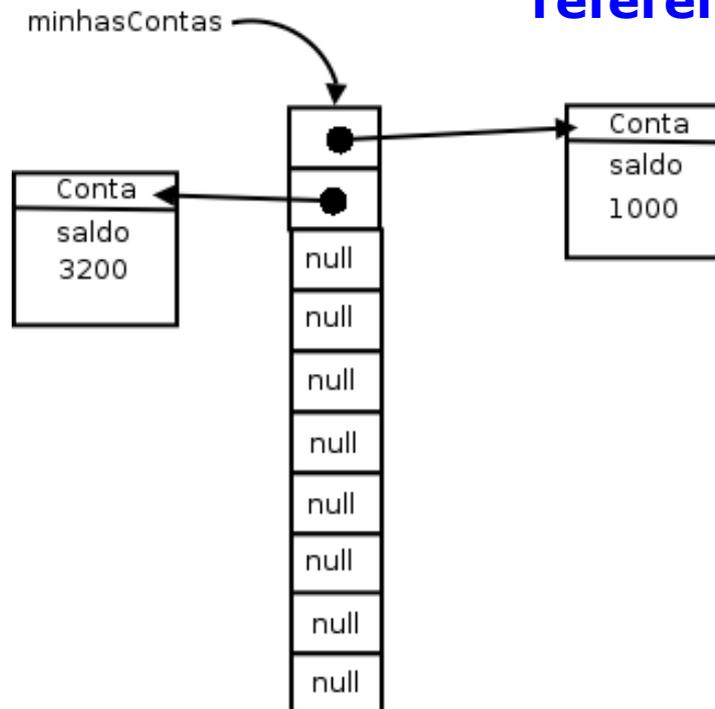
□ Populando o array;

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

□ Fazendo diretamente:

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

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



Percorrendo um Array

□ Percorrer um array é simples:

```
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]);  
    }  
}
```

□ Mas, se ele for **argumento** de um **método**?

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

□ Até onde o **for** deve ir?

Percorrendo um Array

- Toda **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]);  
    }  
}
```

Percorrendo Array no Java 5.0

- **A partir do Java 5.0** uma nova **sintaxe** para percorremos arrays foi **disponibilizada**;

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

        // imprimindo toda a array
        for (int x : idades) {
            System.out.println(x);
        }
    }
}
```

- No caso de você não ter necessidade de manter uma **variável** com o **índice** que indica a posição do **elemento no vetor** (que é uma grande parte dos casos), podemos usar o **enhanced-for**.

Percorrendo Array no Java 5.0

- ❑ Não precisamos mais do **length** para percorrer matrizes cujo **tamanho não** conhecemos:

```
class AlgumaClasse {  
    void imprimeArray(int[] array) {  
        for (int x : array) {  
            System.out.println(x);  
        }  
    }  
}
```

Arrays Multidimensionais

```
1
2 public class InitArray {
3
4     public static void main( String args[]){
5
6         int[][] array1 = {{1, 2, 3}, { 4, 5, 6} };
7         int[][] array2 = {{1, 2}, {3}, { 4, 5, 6} };
8
9         System.out.println(" Valores no array1 por linha são");
10    }
11
12    public static void saidaArrays( int[][] array){
13
14        for ( int linha = 0; linha < array.length; linha++){
15
16            for ( int coluna = 0; coluna < array[linha].length; linha++){
17
18                System.out.printf( "%d ", array[linha][coluna] );
19            }
20
21            System.out.println();
22        }
23    }
24 }
25
```

Exercício