Um pouco de arrays:

Declarar , instanciar , popular e percorrer arrays

Katyusco F. Santos

Applied Software Engineering Group
IFPB Campina Grande

Simples matriz(array) inteiros:


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

int [ ] idades; => por enquanto temos apenas uma referencia, vamos agora criar o objeto:

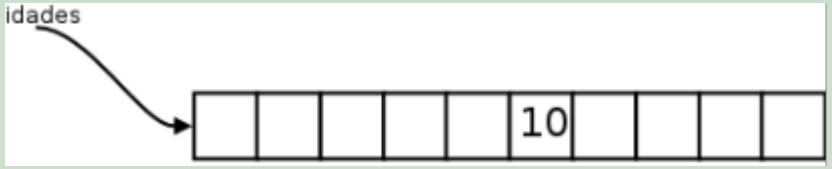
```
idades = new int [10];
```

Muita calma

- Idades = **new int** [10];

Na linha acima, criamos um array de 10 posições, essas posições se estendem de 0 a 9, e podemos começar a fazer as atribuições da seguinte forma:

Idades[5] = 10; => esse código altera o sexto campo do array.



Sempre Lembrar

- 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(ArrayIndexOutOfBoundsExcepti
onTeste.java:5)
```

Arrays de objetos

- Class Conta

```
{
    int numConta;
    int cpf;
    double saldo;
    double limite;
}
```

Então fazemos o seguinte:

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

PERGUNTA: Quantas contas foram criadas aqui?

Vamos popular o array

- Continuando :

System.out.println(minhasContas[0].saldo);

Um erro ocorrerá, pois na posição zero não fazemos nenhuma referencia a algum objeto , por enquanto todos referenciam para o null.

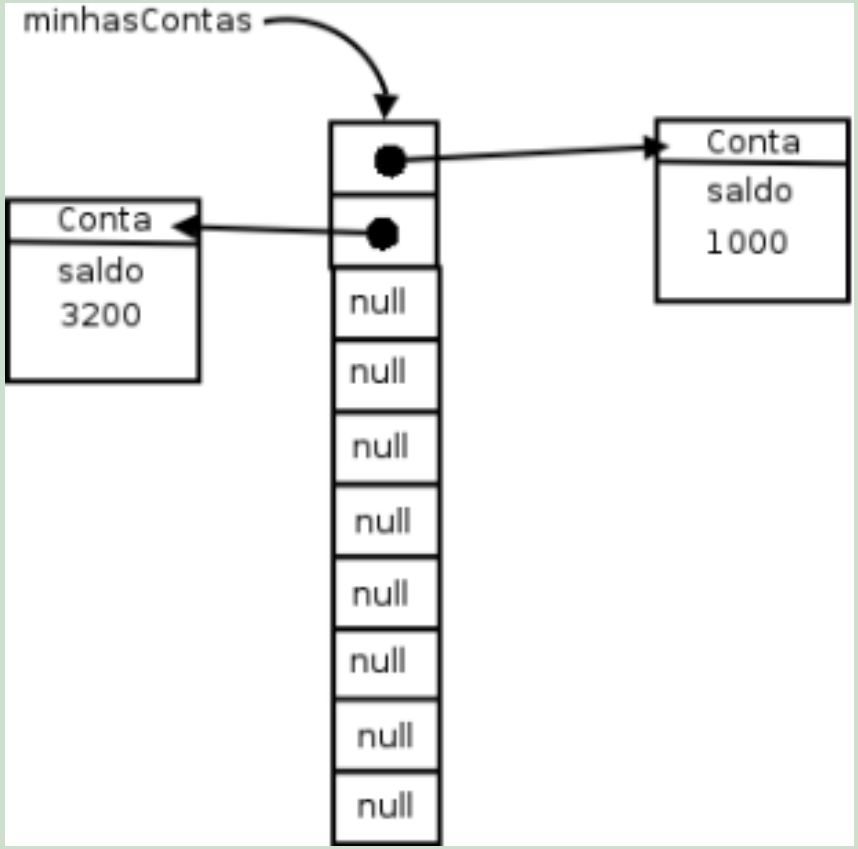
```
Conta contaNova = new Conta ( ); //criando o objeto
contaNova.saldo = 1000.0; // instanciando o objeto
minhasContas[0] = contaNova; // referenciando
```

Popular de forma direta

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

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

- **OBS:** "*Uma array de tipos primitivos guarda valores, uma array de objetos guarda referências*".



E por fim: vamos percorrer


```
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 ] );
        }
}
```

Recebendo array como argumento em um método

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

- **OBS:** “A partir do momento que uma array foi criada, ela não pode mudar de tamanho. Se você precisar de mais espaço, será necessário criar uma nova array, e antes de se referenciar para ela, copie os elementos da array velha”.

Percorrer um array no java 5.0

- O Java 5.0 traz uma nova sintaxe para percorremos arrays (e coleções, que veremos mais a frente).

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**.

O que mudou:

Por valor

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

Por referência

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