Técnico em Desenvolvimento de Sistemas

Lógica de Programação II

Arrays

Alex Helder Cordeiro do Rosário de Oliveira

Instituto Federal de Brasília - *Campus* Brasília

Exemplo Motivador

- Imaginemos um programa para o IBGE, por exemplo, que tenha de processar a renda de todas as famílias de uma cidade.
- A renda de cada família terá de ser armazenada em uma variável.
- Se tivermos de declarar cada variável explicitamente, teremos um programa extremamente extenso.

Exemplo Motivador

```
public static void main(String[] args) {
    float renda0001;
    float renda0002;
    ....
    float renda8793;

    String rendaTexto0001 = JOptionPane.showInputDialog("Entre a renda:");
    renda0001 = Float.parseFloat(rendaTexto0001);
    String rendaTexto0002 = JOptionPane.showInputDialog("Entre a renda:");
    renda0002 = Float.parseFloat(rendaTexto0002);
    ....
    String rendaTexto8793 = JOptionPane.showInputDialog("Entre a renda:");
    renda8793 = Float.parseFloat(rendaTexto8793);

    float media = (renda0001+renda0002+renda0003+....+renda8793)/8793;
}
```

Solução

- Usar uma estrutura que permita:
  - Alocar de uma só vez, espaço suficiente para diversas variáveis.
  - Agregar operações repetidas nas diferentes variáveis em algum tipo de laço.

Arrays

- É um conjunto de variáveis (elementos) de um mesmo tipo;
- Os arrays podem ser de qualquer tipo de variáveis (tanto primitivos quanto objetos);
- Não pode ser armazenado, em um mesmo array, variáveis de tipos diferentes;
- Os elementos são identificados através do nome do array e um índice.

Arrays - Declaração

- Declara-se semelhante a variáveis comuns, usando o tipo seguido do nome da variável;
- A declaração distingue-se apenas pela presença de colchetes.
- Existem duas formas de declaração aceitas pelo java*.

```
int arrayDeInteiros[];
int[] arrayDeInteiros;
```

- Apesar disso, a convenção de código indica que deve ser usada, necessariamente a segunda forma para clareza de código.
  - A primeira forma provavelmente é mantida devido à semelhança com a programação em C;
  - Na segunda forma, a indicação que a variável é do *tipo* array está na declaração do *tipo*.

*Considere "aceitas" como sendo "compila e executa".

Arrays - Inicialização

- Semelhante a outros objetos, a construção de um *novo* array se faz com a palavra chave *new**.
- Usamos `new` antes do tipo da variável (que obrigatoriamente é o mesmo tipo da declaração) seguido da quantidade de elementos entre colchetes.

```
int[] arrayDeInteiros;
arrayDeInteiros = new int[156];
int[] arrayDeInteiros = new int[156];
```

- A quantidade de elementos pode ser definida em tempo de execução, isto é, não precisa ser constante em tempo de compilação†.
- Uma vez construído o objeto do array, não podemos alterar o seu tamanho.

*Significa novo em inglês.

†Pode ser definida pelo usuário no decorrer da execução do programa.

Arrays - Inicialização

- Uma vez construído o array, as variáveis que constituem este array são automaticamente inicializados com valor padrão.

boolean	`false`
byte, short, int, long	`0`
float, double	`0.0`
char	`'\u0000'`
Objetos	`null`

Array - Acesso

- Cada elemento será acessado pelo nome do array seguido do seu indice entre colchetes;

```
arrayDeInteiros[5] = 32;

int variavelAuxiliar = arrayDeInteiros[13];
```

- O índice é um número inteiro que varia de 0 ao tamanho do array - 1*.
- Tentar acessar elementos com índice negativo, maior ou igual ao tamanho ou com valor não inteiro resulta em erro.
- Para obter o tamanho de um array recebido de uma fonte externa, podemos usar o atributo `lenght`.

```
int tamanhoDoArray = arrayDeInteiros.lenght;
```

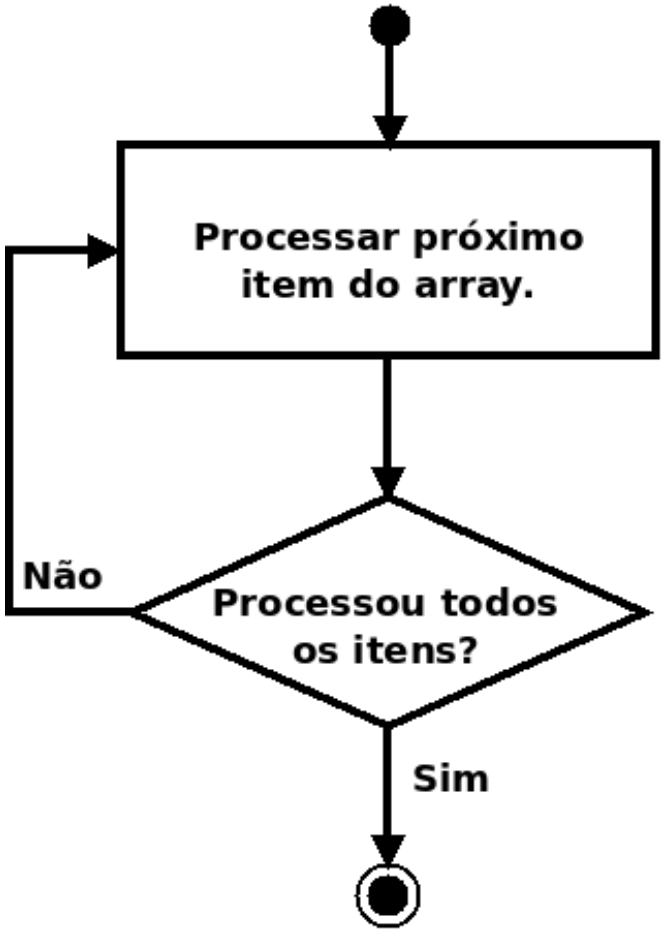
*O "- 1" se dá ao fato de que incluímos o "0".

Array - Manipulação dentro de `for`

- Os arrays permitem a realização de operações repetidas dentro de um laço, utilizando o índice para identificar cada item.
- Uma vez que sabemos a quantidade de vezes que queremos repetir a operação (quantidade de itens do array), podemos usar o laço `for`.
- Exemplo de atribuição de valores aos elementos do array através de uma fórmula (de números pares):

```
int[] array = new int[20];
for(int i = 0; i < 20; i++) {
    array[i] = 2*i;
}
```

Array - Manipulação dentro de `for`



Array - Manipulação dentro de `for`

- Exemplo de como obter do usuário os valores para os elementos do array:

```
int[] array = new int[20];
for(int i = 0; i < 20; i++) {
    String texto = JOptionPane.showInputDialog("Insira um
    array[i] = Integer.parseInt(texto);
}
```

- Exemplo de como apresentar o array ao usuário:

```
    System.out.println("Elementos do Array:");
    for(int i = 0; i < 20; i++) {
        System.out.print(array[i]+"  ");
    }
```

*Exemplo encontrado no arquivo `Arrays.java`.

Sua vez:

11. Escreva um programa que receba 10 números e armazene-os em um array. Depois apresente **quais** destes números são menores que 10.

Sua vez:

12. Escreva um programa que receba 10 números e armazene-os em um array. Depois apresente **quantos** destes números são menores que 10.

Array - Inicialização de objetos

```
JButton[] botoes = new JButton[10];
```

- Quando inicializamos um array de objetos, cada elemento deste array ainda terá valor `null`;
- Em outras palavras, os objetos que fazem parte do array não foram inicializados;
- Para poder utilizar cada elemento, deve-se fazer a inicialização de cada um (dentro de um for, por exemplo)*.

```
JButton[] botoes = new JButton[10];
for(int i = 0; i < 10; i++) {
    botoes[i] = new JButton();
}
```

*Se não for inicializado, pode lançar um NullPointerException.