



ED Estrutura de Dados e Armazenamento

Arrays (vetores)

Profa. Célia Taniwaki

Arrays

- Estruturas de dados consistindo em itens de dados relacionados do mesmo tipo
- Grupo de variáveis (elementos) contendo valores do mesmo tipo.
- Arrays são objetos.
- Elementos podem ser tipos primitivos ou objetos
- Nome do elemento
 - nome do array, seguido de colchetes, com o índice dentro dos colchetes.
 Ex: c[3]
- Índice referencia um determinado elemento em um array
 - Primeiro elemento tem índice zero
 - Último elemento tem índice = nro de elementos 1

Arrays - Exemplo

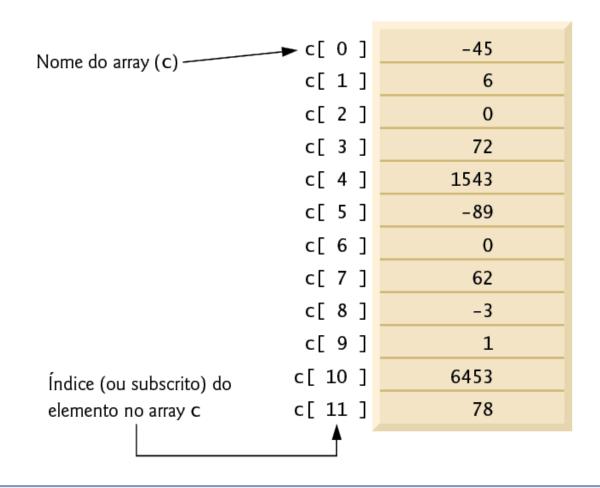


Figura 7.1 | Um array de 12 elementos.

Arrays

- Índice:
 - Deve ser um inteiro n\u00e3o negativo
 - Pode ser uma expressão (variável ou cálculo envolvendo número e variável)
- Tamanho
 - Fornecido pela variável de instância length
 - length não pode ser alterado pois é uma variável final

Declaração de arrays

- Objetos array
 - Criados com palavra-chave new.
 - Especifica o tipo de elemento e o número de elementos em uma expressão de criação de array.
 - Retorna uma referência que pode ser armazenada em uma variável de array.
- Declaração e expressão de criação de arrays de 12 elementos int

```
int[] c = new int[ 12 ];
```

Pode ser realizado em duas etapas como segue:

```
int[] c; // declara a variável de array
c = new int[ 12 ]; // cria o array
```

Inicializador de array

Inicializador de array.

- Uma lista de expressões separadas por vírgulas (chamadas lista de inicializadores) entre chaves.
- Utilizado para criar um array e inicializar seus elementos.
- O comprimento, ou tamanho, do array é determinado pelo número de elementos na lista inicializadora.

```
int[] n = \{ 10, 20, 30, 40, 50 \};
```

Cria um array de cinco elementos com valores de índice 0–4.

 O compilador conta o número de inicializadores na lista para determinar o tamanho do array.

Configura a operação new apropriada "nos bastidores".

7

Exemplo – Uso de array

```
//Cria um vetor de 10 elementos, atribui os
//números 0, 10, 20, ..., 90 a cada elemento
//do vetor, e imprime esses números
public class ExemploArray1 {
  public static void main(String args[]) {
   int[] vetor = new int[10];
   for (int i = 0; i < 10; i++) {
      vetor[i] = i * 10;
   for (int i = 0; i < 10; i++) {
      System.out.println(vetor[i]);
```

Outro exemplo

```
//Lê 10 números, armazena-os em um vetor
//Depois imprime esses números
import java.util.Scanner;
public class ExemploArray2 {
  public static void main(String args[]) {
   Scanner input= new Scanner(System.in);
   int[] v = new int[10];
   for (int i = 0; i < v.length; i++) {
      System.out.println("Digite um número: ");
      v[i] = input.nextInt();
   for (int i = 0; i < v.length; i++) {
      System.out.println(v[i]);
```

Instrução for aprimorado

- Assim como no ArrayList, é possível utilizar a instrução for aprimorada (for enhanced) com arrays
- Sintaxe:

```
for ( parâmetro : nomeDoArray )
  instrução
```

- onde *parâmetro* tem um tipo e um identificador
- e nomeDoArray é o array pelo qual iterar.
- O tipo do parâmetro deve ser consistente com o tipo de elemento no array.

Exemplo de uso de for aprimorado

```
public class ForEnhanced {
  public static void main (String args[]) {
    int[] vet = {87,65,70,12,3,90,45,12,30,100};
    int soma = 0;
                                                Variável num assume cada
    //soma os elementos de vet
                                                elemento de vet a cada
                                                iteração do for
    for (int num : vet)
       soma += num;
    System.out.println("A soma dos elementos do vetor é " +
                         soma);
```

Instrução for aprimorado

- A instrução for aprimorada somente pode ser utilizada para obter elementos do array
- Não pode ser utilizada para modificar elementos em um array
- Para modificar elementos, utilize a instrução for tradicional.
- A instrução for aprimorada percorre o vetor inteiro, e portanto, não é indicada quando o vetor não está totalmente preenchido.

Semelhanças e diferenças entre array e ArrayList

array

- Armazena elementos de mesmo tipo
- Uma vez definido seu tamanho, não pode ter seu tamanho alterado dinamicamente (em tempo de execução), a não ser que se crie uma cópia do array, com espaço maior ou menor
- Aceita elementos de tipo primitivos ou não

ArrayList

- Armazena elementos de mesmo tipo
- Seu tamanho pode ser alterado em tempo de execução
- Não aceita elementos de tipo primitivo (int, char, double, float, boolean). No lugar desses, usa-se, respectivamente, os objetos wrapper Integer, Character, Double, Float, Boolean.

Agradeço a sua atenção!

Célia Taniwaki

celia.taniwaki@sptech.school



SÃO PAULO TECH SCHOOL