Fundamentos de programação

Array - unidimensional

Edson Moreno

edson.moreno@pucrs.br
http://www.inf.pucrs.br/~emoreno

Array

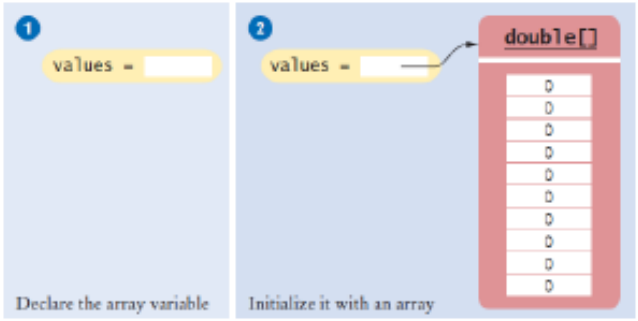
- Um programa de computador frequentemente necessita armazenar uma lista de valores para então processá-los
  - Por exemplo:
    - Cálculo de variância ou desvio padrão
    - Divisão das despesas de uma festa entre um grupo de amigos
    - etc.
- Se você tiver uma lista de valores (por exemplo, 32, 54, 67,5, 29, 35, 80, 115, 44,5,100, 65), quantas variáveis seriam necessárias?
  - double input1, input2, input3, ...
- Arrays resolvem este problema
- Um array armazena sequências de valores do mesmo tipo

Declaração de array

- Declarar um array envolve 2 etapas
  - 1 Declarar a variável array

    double[] values;
  - 2 Inicializar o array

    values = **new double**[10];



- O array não pode ser utilizado até que o compilador saiba qual o tamanho do array na etapa 2

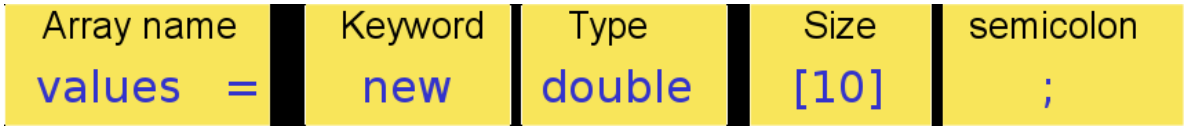
Declaração de array

- As seguintes partes devem ser especificadas

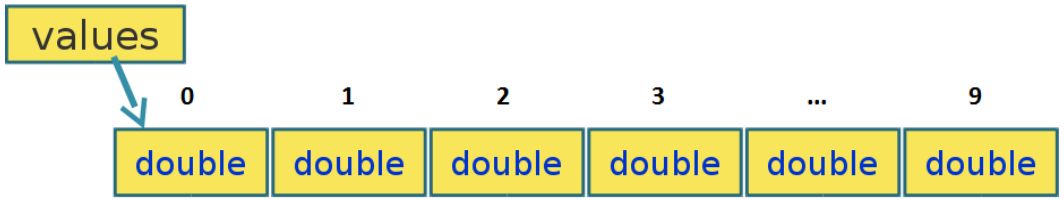
- Esta declaração especfica que
  - Há um array chamado *values*
  - Que os seus elementos são do tipo *double*
  - E que (AINDA) não foi definido quantos elementos ele poderá armazenar
- Outras considerações
  - Arrays podem ser declarados em qualquer lugar onde também seja possível declarar uma variável
  - Não use palavras-reservadas ou nomes que já estejam em uso

Declaração de array

- Reserva-se memória para todos os elementos



- Agora o compilador sabe que o array chamado values necessita de [10]
- elementos do tamanho do tipo double
- O array também está sendo inicializado: cada elemento do array recebe o valor 0
- Não se pode alterar o tamanho do array depois da sua declaração!



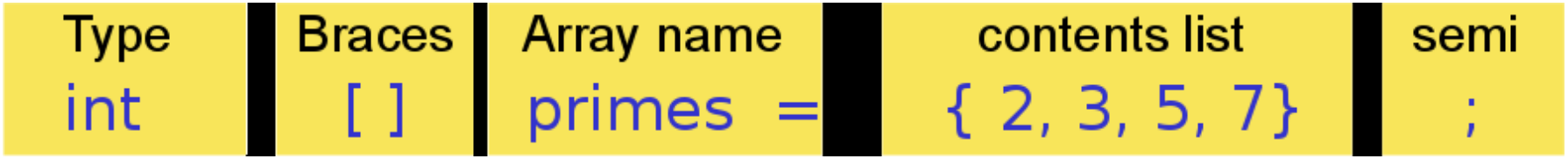
Declaração de array

- Declaração e criação em 1 linha

- Está sendo feita a declaração de um array chamado values para armazenar
- elementos do tipo double
- Está sendo reservada memória para armazenamento de [10] elementos do tipo
- double
- Os elementos estão sendo inicializados com 0

Declaração e inicialização

- Pode-se declarar e definir os conteúdos iniciais de todos os elementos de um array usando:

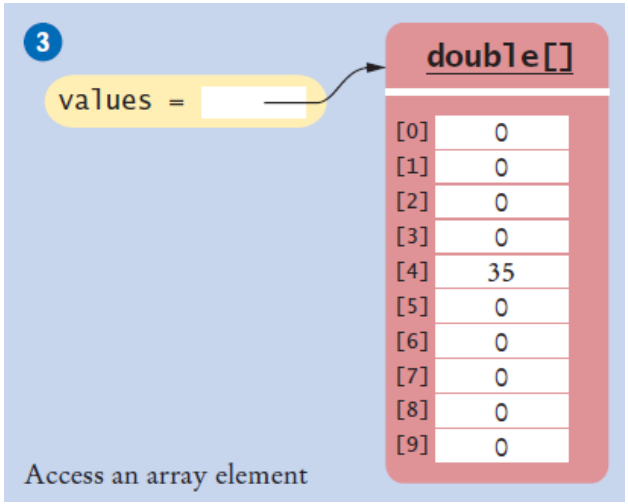


- Está sendo declarado que:
  - O array primes conterá elementos do tipo int
  - Haverá espaço para 4 elementos (automaticamente contados pelo compilador) que conterão os seguintes valores iniciais: 2, 3, 5 e 7
  - O par de chaves determina uma lista de valores iniciais para o array

Acesso ao array

- Cada elemento de um array é numerado:
  - Este número é chamado de índice
  - Acessa-se um elemento especificando o nome do array e o índice numérico
- Elementos no array values são acessados por um índice inteiro i, usando a notação values[i]

```
public static void main(String[] args) {
  double[] values;
  values = new double[10];
  values[4] = 35;
}
```



Sintaxe

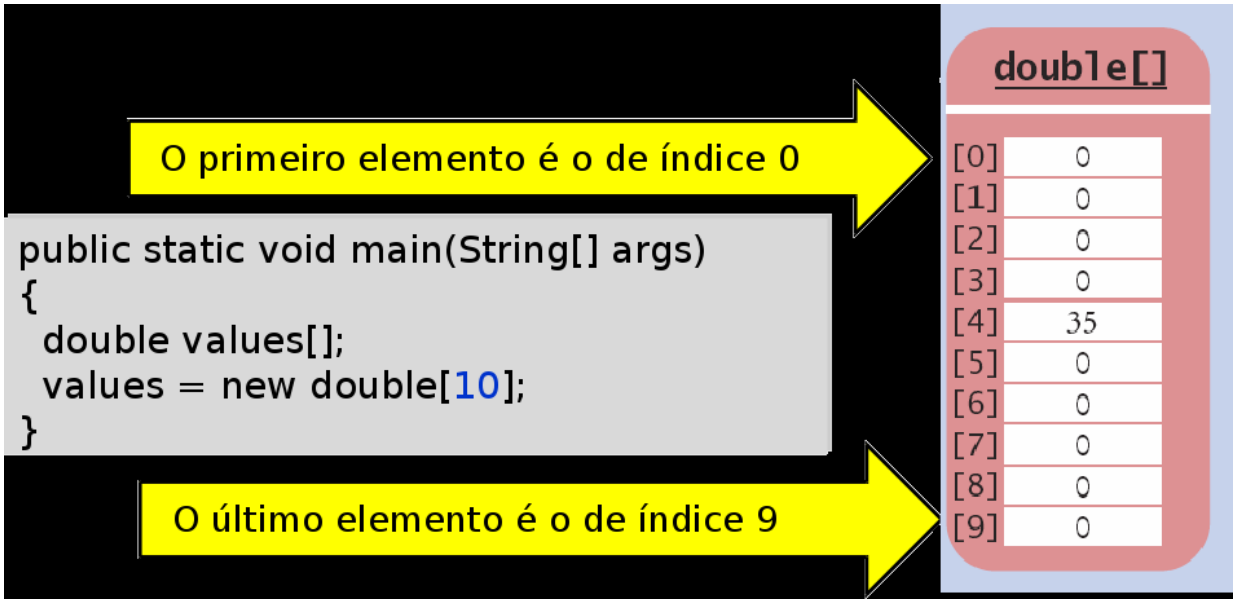
- Para declarar um array, especifique
  - O nome da variável array
  - O tipo de seus elementos
  - O tamanho (número de elementos)

```
public static void main(String[] args) {
  double[] values = new double[10];
  double[] moreValues = {32, 54, 67, ..., 35};
                                  // lista dos valores de inicialização

  values[i] = 27;
     // I deve estar entre 0 e length do array -1
}
```

Números de índice de Array

- Números de índices de arrays iniciam em 0 e os demais são números inteiros positivos
- Um array com 10 elementos tem índices de 0 até 9: **NÃO há elemento 10**



Verificação de limites de array

- Um array sabe quantos elementos ele pode armazenar
  - values.length é o tamanho de um array chamado values
  - Trata-se de um valor inteiro que corresponde ao índice do último elemento + 1
- Usa-se isto para verificar e prevenir erros de acesso fora dos limites
- Strings e arrays usam sintaxes diferentes para encontrar os seus tamanhos
  - Strings: name.length()
  - Arrays: values.length

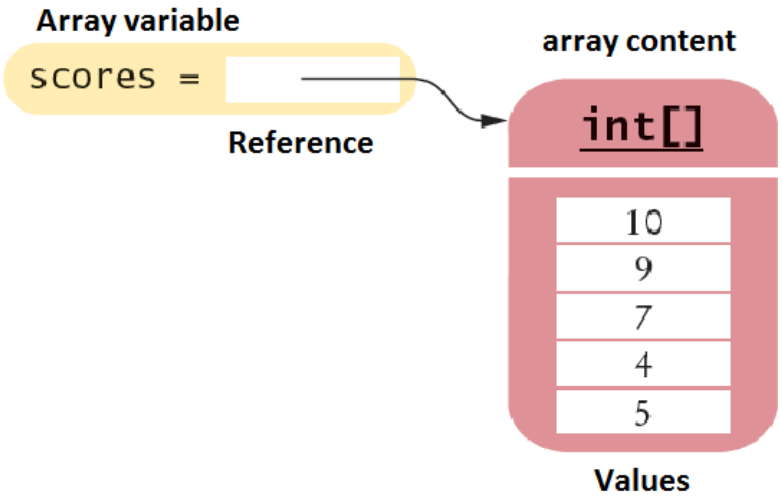
```
public static void main(String[] args) {
  int i = 10, value = 34;
  double values[];
  values = new double[10];
  if (0 <= i && i < values.length) { // length is 10
    values[i] = value;
  }
}
```

Exemplo de declaração de array

Declaração	Explicação
`int[] numeros = new int[10];`	Um *array* de 10 inteiros. Todos os elementos inicializados com zero.
`final int TAMANHO = 10;` `int[] numeros = new int[TAMANHO];`	É uma boa ideia usar uma constante, em vez de um número constante.
`int tamanho = in.nextInt();` `double[] dados = new double[tamanho];`	O tamanho não precisa ser uma constante.
`int[] quadrados = { 0, 1, 4, 9, 16 };`	Um *array* de 5 inteiros, com valores iniciais.
`String[] amigos = { "Joao", "Maria", "Paulo" };`	Um *array* com 3 *strings*.
`double[] data = new int[10]; // ERRO`	**ERRO!** Não se pode inicializar uma variável `double[]` com um *array* do tipo `int[]`.

Referências a arrays

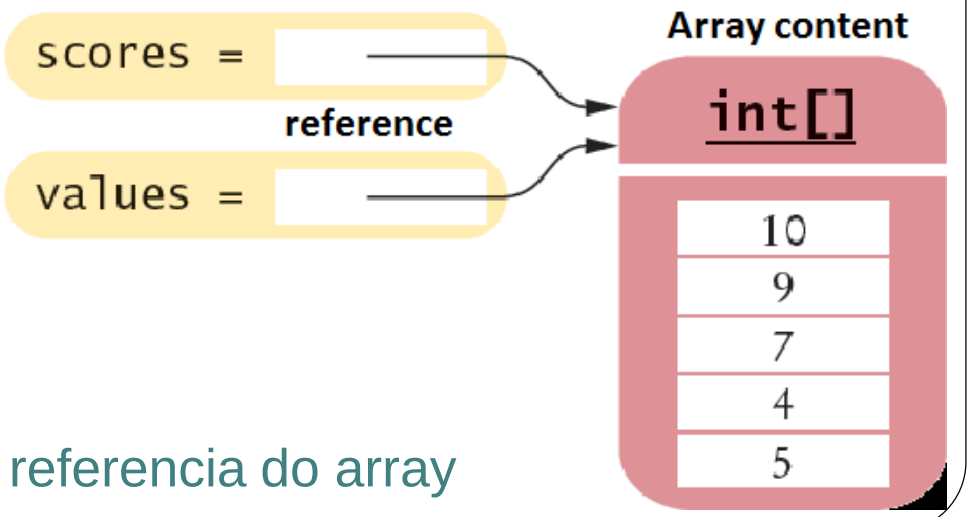
- É preciso perceber que há uma diferença entre:
  - Variável array: chamada de “manipulador” do array
  - Conteúdos do array: memória onde os valores estão armazenados
- Uma variável array contém uma referência aos conteúdos do array
- A referência é a localização dos conteúdos do array (na memória)




```
int [] scores = { 10, 9, 7, 4, 5 };
```

Referências a array

- Pode-se fazer uma variável array referenciar os mesmos conteúdos de outra variável array
- Uma variável array especifica a localização de um array
- Copiar uma referência corresponde a se ter uma segunda referência para o mesmo conteúdo




```
int [] resultados = { 10, 9, 7, 4, 5 };
int [] valores = resultados; // Copia da referencia do array
```

Array parcialmente preenchido

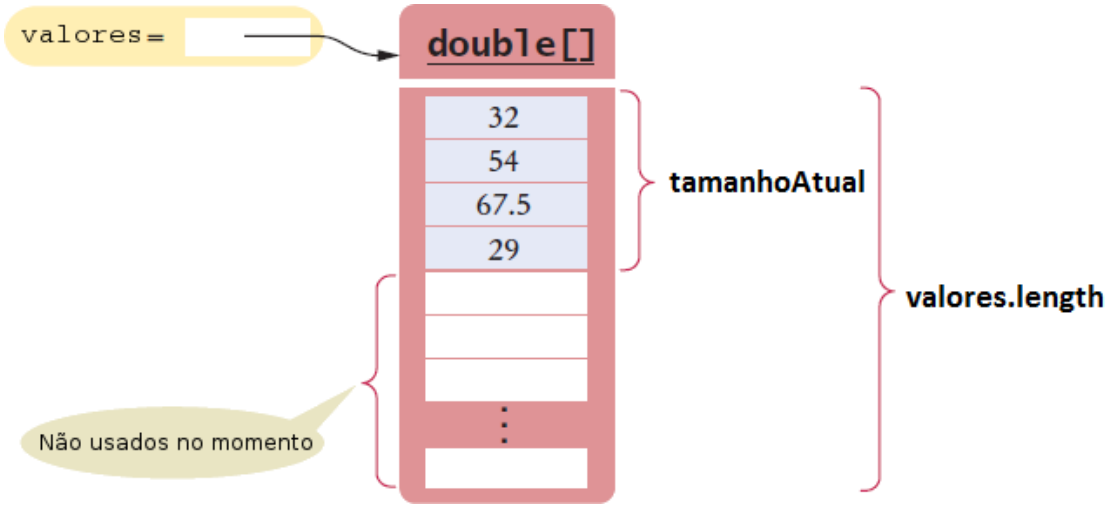
- Um array não pode ter o seu tamanho alterado durante a execução
  - O programador pode necessitar obter valores até o número máximo de elementos necessário
  - Usa-se uma variável (tamanhoAtual no exemplo a seguir) para controlar quantos elementos já foram obtidos

```
final int TAMANHO = 100;
 double[] valores = new double[TAMANHO];
 int tamanhoAtual = 0;
 Scanner in = new Scanner(System.in);
 while (in.hasNextDouble()) {
  if (tamanhoAtual < valores.length) {
   valores[tamanhoAtual] = in.nextDouble();
   tamanhoAtual++;
  }
 }
```

Percorrendo array

- No exemplo, usa-se tamanhoAtual (e não valores.length) para determinar qual o último elemento
- Um laço for é uma escolha natural para percorrer um array

```
for (int i = 0; i < tamanhoAtual; i++) {
  System.out.println(valores[i]);
}
```



Erros comuns

- **Erro nos limites de arrays**
  - Índices de arrays iniciam em 0 e terminam em tamanho - 1
  - Acessar um elemento que não existe é um erro bastante comum
  - Resultado: exceção lançada em tempo de execução

(java.lang.ArrayIndexOutOfBoundsException)

```
public class OutOfBounds {
  public static void main(String[] args) {
    double values[];
    values = new double[10];
    values[10] = 100; // ERRO=EXCECAO
  }
}
```

Erros comuns

- **Arrays não inicializados**
  - Não se esqueça de inicializar variáveis array
  - O compilador vai gerar um erro como “variable values might not have been initialized”

```
double[] values;
...
values[0] = 29.95; // ERRO!
```

```
double[] values;
values = new double[10];
values[0] = 29.95; // SEM ERRO!
```

Laço de iteração

- Usar laços for para percorrer um array é bastante comum:
  - O for abreviado simplifica este processo
  - Este laço também é chamado de for each (“para cada ”)
  - Este código pode ser lido como: “Para cada elemento do array”
- À medida que o laço avança, ele:
  - Acessará cada elemento sequencialmente (de 0 até o tamanho - 1)
  - Copiará o seu valor para a variável de induçao
  - Executará o corpo do laço
- Não é possível para:
  - Alterar elementos
  - Obter erros de limite

```
double[] values = ...;
double total = 0;
for (double element : values) {
  total = total + element;
}
```

Algoritmos comuns para array

- Preencher um array
- Soma e média de valores
- Encontrar máximo e mínimo
- Saída de elementos com separadores
- Busca linear
- Remoção de um elemento
- Inserção de um elemento
- Troca de elementos
- Cópia de arrays
- Aumento de tamanho de arrays
- Leitura da entrada

Preenchimento de array

- Inicializar um array com um conjunto de valores calculados
- Por exemplo: preencher um array com os quadrados de 0 até 10

```
int[] quadrados = new int[11];
for (int i = 0; i < quadrados.length; i++) {
  quadrados[i] = i * i;
}
```

Soma e média de valores

- Use o laço for abreviado e não se esqueça de evitar a divisão por zero

```
double total = 0, media = 0;
if (valores.length > 0) {
  for (double elemento : valores) {
    total = total + elemento;
  }
  media = total / valores.length;
}
```

Encontrar max e min

- Defina que o maior (ou menor) é o primeiro elemento
- Teste os outros elementos usando for ou for abreviado

```
// Laco tipico para encontrar o maximo
double maior = valores[0];
for (int i = 1; i < valores.length; i++) {
  if (valores[i] > maior) {
    maior = valores[i];
  }
}

// Laco abreviado para encontrar o maximo
double maior = valores[0];
for (double elemento : valores) {
  if (elemento > maior) {
    maior = elemento;
  }
}
```

Saída de elementos com separadores

- Imprime-se o separador antes de todos os elementos, com exceção do primeiro

```
double[] valores = {32, 54, 67.5, 29, 35};
for (int i = 0; i < valores.length; i++) {
  if (i > 0) {
    System.out.print(" | ");
  }
  System.out.print(valores[i]);
}
```

  - Resultado: 32 | 54 | 67.5 | 29 | 35

- Ou usa-se o método Arrays.toString() (muito útil para depuração)
  - Teste o resultado

```
import java.util.*;
// ...
double[] valores = {32, 54, 67.5, 29, 35};
System.out.println(Arrays.toString(valores));
```

Busca linear

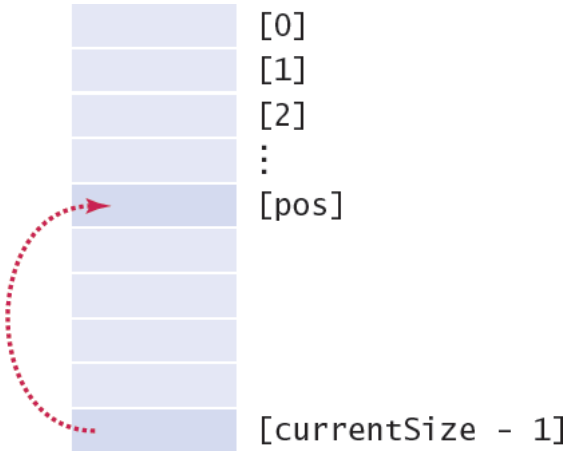
- Busca-se um valor específico em um array
  - Inicia-se pelo primeiro elemento e para-se quando/se o valor for encontrado
  - Usa-se uma variável booleanda achou para controlar o final do laço

```
int valorBuscado = 100; int pos = 0;
boolean achou = false;
while (pos < valores.length && !achou) {
  if (valores[pos] == valorBuscado) {
  achou = true;
  }
  else {
    pos++;
  }
}

if (achou)
  System.out.println("Encontrado na posicao: " + pos);
else
  System.out.println("Nao encontrado");
```

Remoção de elementos

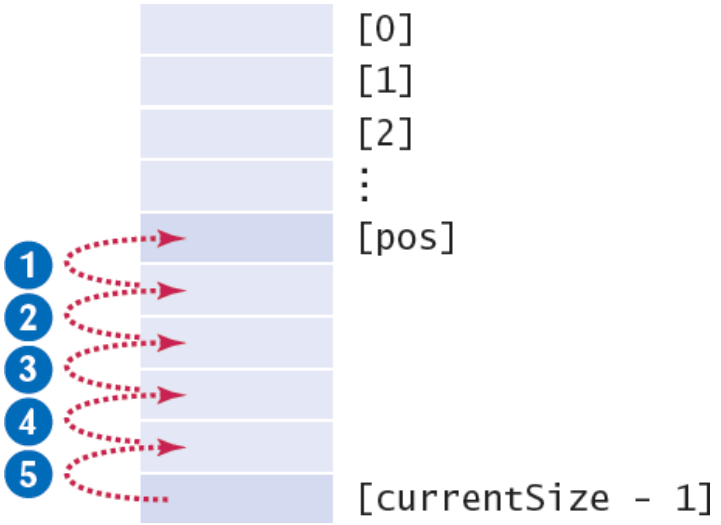
- Exige que se mantenha uma variável com o tamanho atual (número de elementos válidos)
- Não se pode deixar um “buraco” no array
- Se NÃO é preciso manter o array ordenado: copie o último elemento sobre o elemento atual e atualize o tamanho atual




```
if ( pos>=0 && pos<=valores.length-1 ) {
  if ( tamanhoAtual > 1 ) {
    valores[pos] = valores[tamanhoAtual - 1];
  }
  tamanhoAtual--;
}
```

Remoção de elementos

- Se é preciso manter o array **ordenado**: mova todos os elementos que estão após pos uma posição (em direção ao início do array) e atualize o tamanho atual



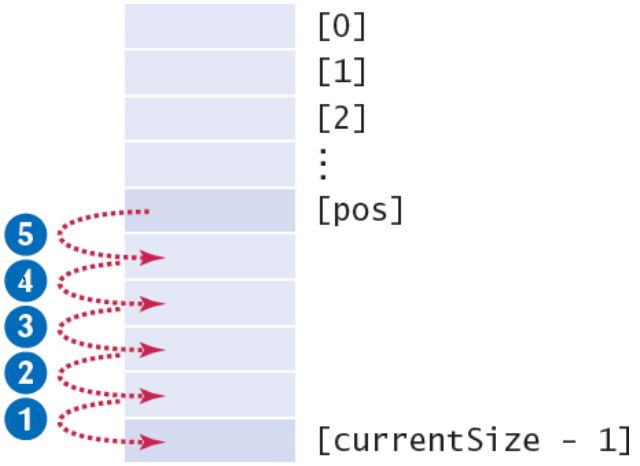

```
if ( pos>=0 && pos<=valores.length-1 ) {
  for (int i = pos; i < tamanhoAtual - 1; i++) {
  valores[i] = valores[i + 1];
  }
  tamanhoAtual--;
}
```

Inserção de elementos

- Se não é preciso manter a ordenação, apenas adiciona-se o novo valor no final e atualiza-se o tamanho

```
if ( tamanhoAtual < valores.length ) {
  valores[tamanhoAtual] = novoValor;
  tamanhoAtual++;
}
```

- Se é preciso manter a ordem, localiza-se a posição correta para o novo elemento, move-se todos os elementos válidos uma posição (em direção ao final do array) e atualiza-se o tamanho

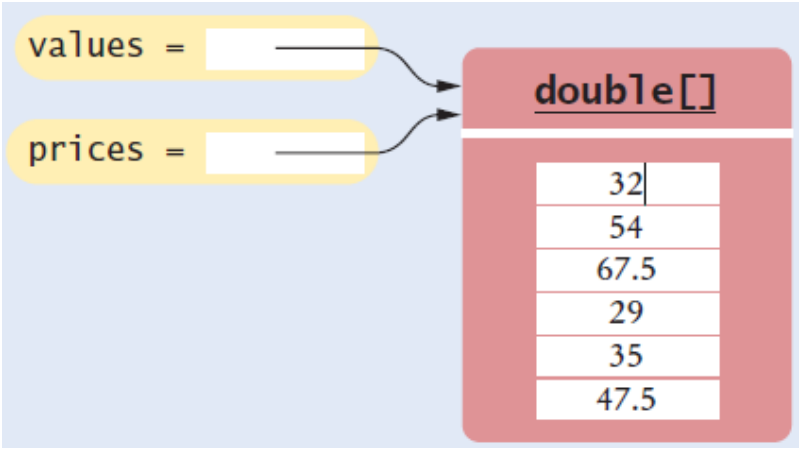



```
if (tamanhoAtual < valores.length) {
  tamanhoAtual++;
  for (int i = tamanhoAtual - 1; i > pos; i--) {
    valores[i] = valores[i - 1]; // move p/ inicio
  }
  valores[pos] = novoValor; // preenche buraco
}
```

Cópia de array

- Copiar arrays não é a mesma coisa que copiar apenas a referência
  - A cópia de arrays cria 2 conjuntos de conteúdos
  - Exemplo de cópia de referência:

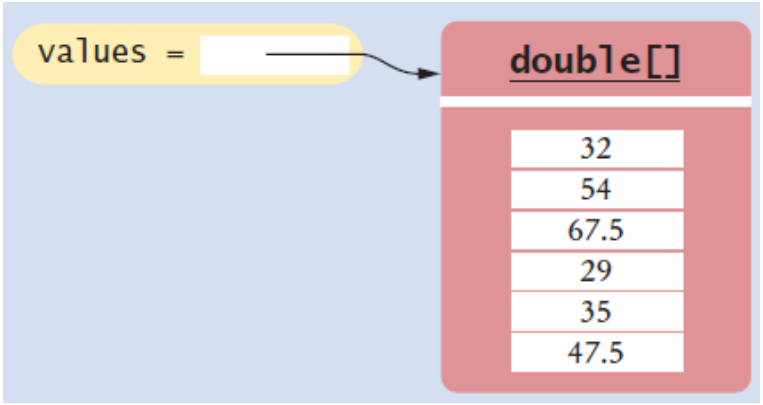
```
double[] values = { 32, 54, 67.5, 29, 35, 47.5 };
// Copia de referencia
double[] prices = values;
```

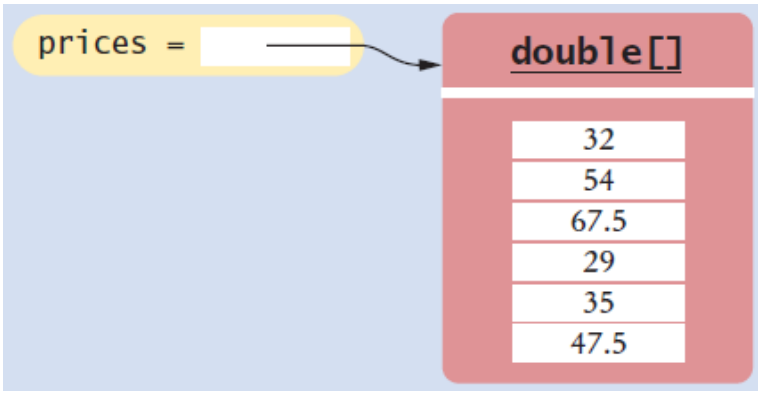


Cópia de array

- Pode-se usar o método Arrays.copyOf (Java 6):

```
double[] values = { 32, 54, 67.5, 29, 35, 47.5 };
// copyOf cria uma nova copia, retornando a referencia
double[] prices = Arrays.copyOf(values, values.length);
```

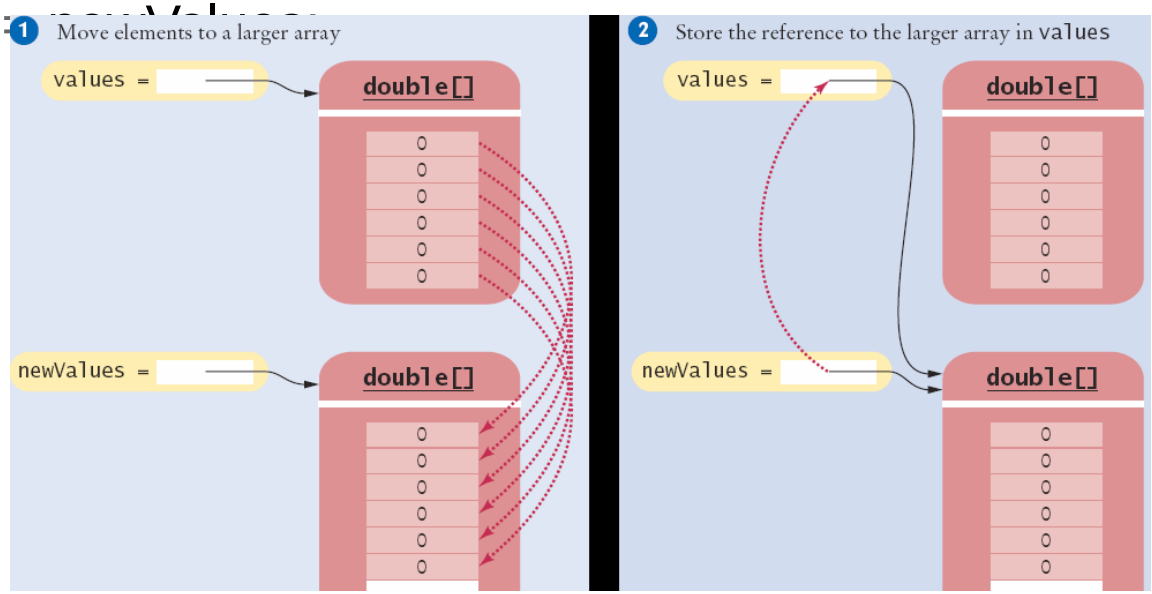




Aumento do tamanho do array

- Não é possível mudar o atributo correspondente ao tamanho de um array, mas é possível criar outra área de memória com copyOf e fazer a variável array apontar para ela
- Por exemplo, para duplicar o tamanho de um array existente:

```
double[] values = { 32, 54, 67.5, 29, 35, 47.5 };
double[] newValues = Arrays.copyOf(values, 2 *
values.length);
values = newValues;
```



Exercício

- Crie um programa com o tamanho do array sendo solicitado quando da execução do programa. Preencha as várias posições com valores aleatórios. A seguir, crie as opções de listagem de valores do vetor, eliminação de elementos e inclusão de elementos no vetor.

Exercício

- Considere um array parcialmente preenchido

  double[] valores = new double[100];

- com valores até

  int tamAtual;

- E escreva um trecho de programa em Java para eliminar todos os valores zero deste array, sem criar um novo array.

- Considere um array de inteiros chamado valores e escreva um programa em Java para inverter este array (ou seja, troca o primeiro elemento pelo último, o segundo pelo penúltimo,e assim sucessivamente).