PROGRAMAÇÃO ORIENTADA A OBJETOS

Sobrecarga de métodos, arrays e varargs

Prof. Emanoel Barreiros

Sobrecarga (*overloading*) de métodos

- O *overloading* ocorre quando você escreve métodos com o mesmo nome dentro da mesma classe
- Para o Java, o nome do método é apenas uma das características do método
- A assinatura completa do método é composta por seu nome e, a quantidade, ordem e tipos de parâmetros
- Ex:
  - System.out.println(double)
  - System.out.println(int)
  - System.out.println(String)
  - Etc.

Sobrecarga (*overloading*) de métodos

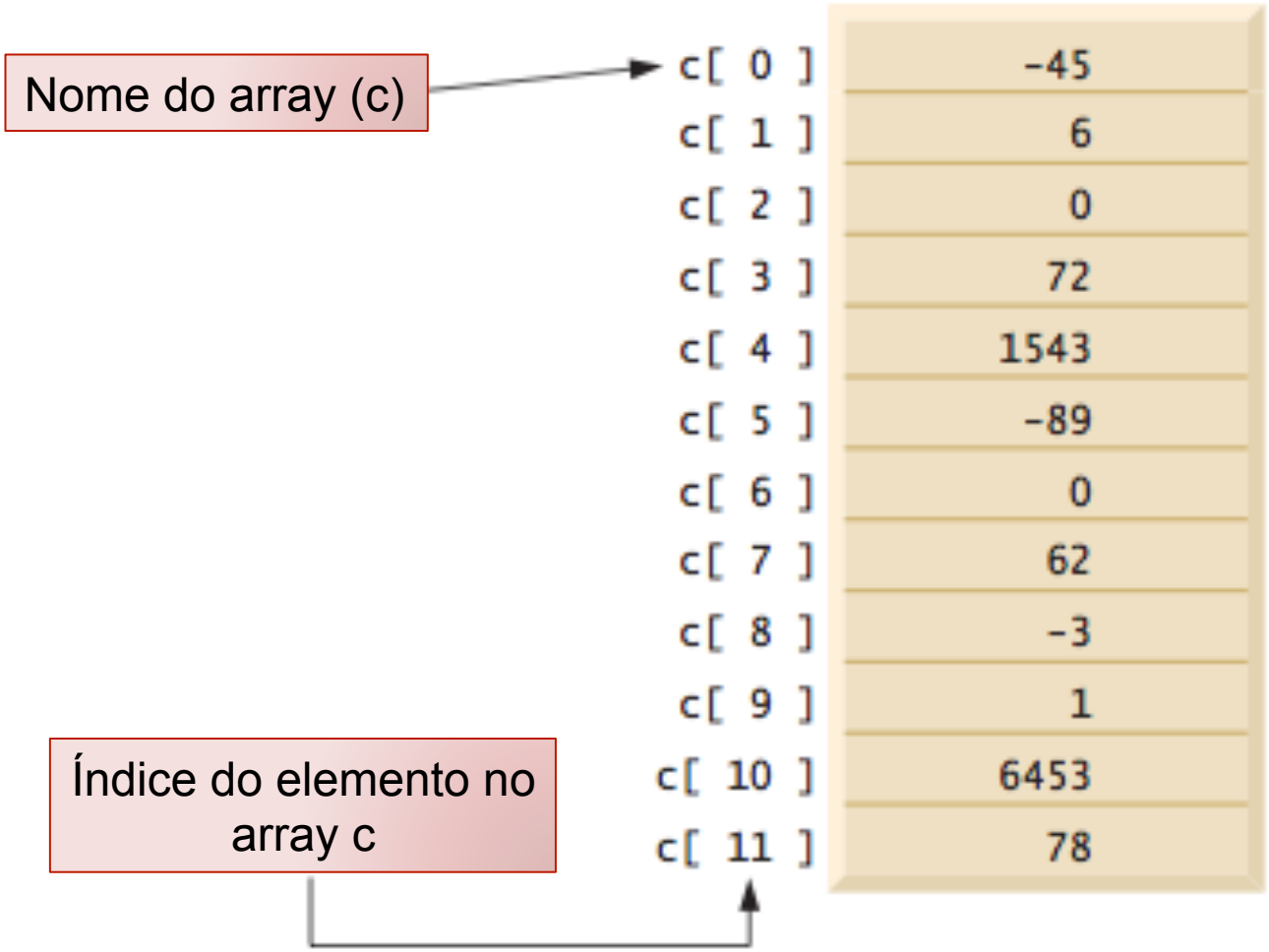
- Na maioria dos casos, o compilador decide qual método deverá ser chamado em tempo de compilação, em outros casos, a decisão é feita em tempo de execução
- Para declarar métodos com sobrecarga, você precisa apenas declarar os métodos com o mesmo nome na mesma classe
- ATENÇÃO: O compilador não gosta de métodos com assinaturas iguais. Métodos com a mesma assinatura na mesma classe geram erros de compilação

ARRAYS E ARRAYLISTS

Arrays

- Arrays (ou vetores) são coleções de valores
- Armazenam itens relacionados de mesmo tipo
- Arrays são objetos, logo, são tipos por referência
- Para referenciar um item em particular em um array, utilizamos o seu índice no array entre colchetes ([ ])
- O primeiro índice do array tem índice 0
- A variável de instância length do array indica o seu tamanho

Um array na memória



Acessando elementos de arrays

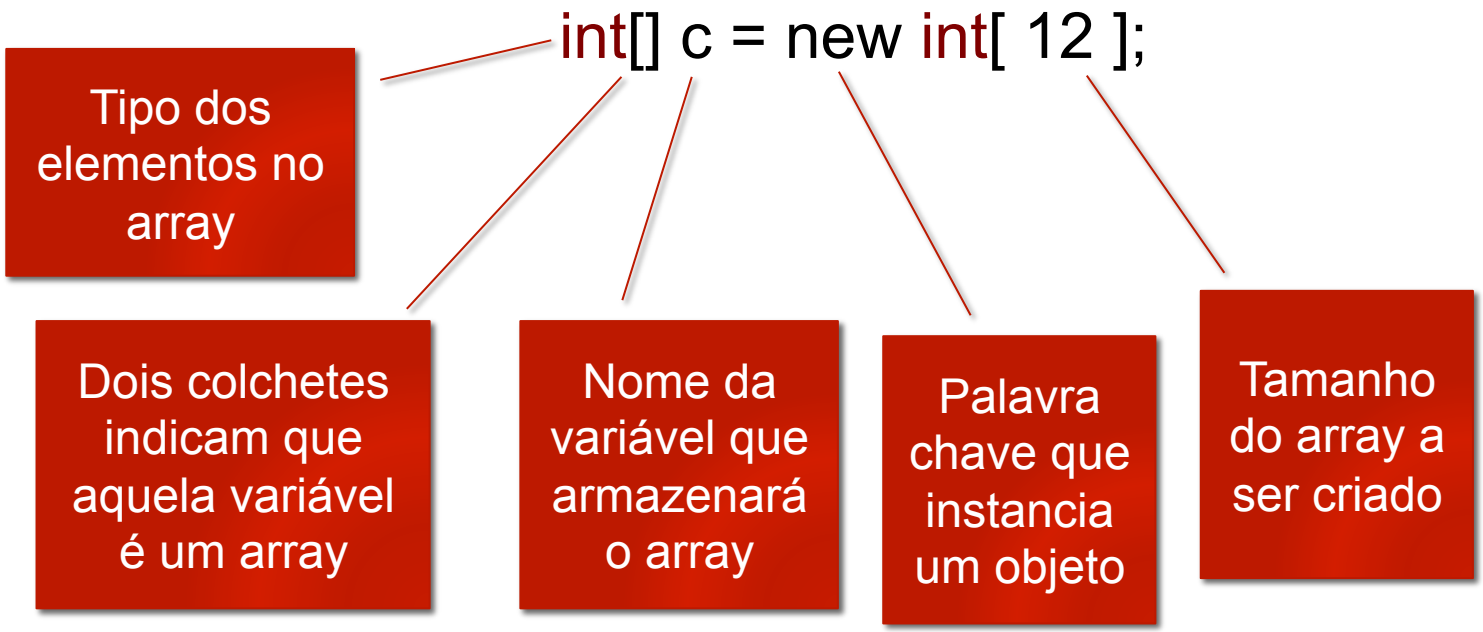
- Elementos do array são acessados pelos seus índices
- Um índice precisa ser um inteiro não negativo, logo, o índice pode ser representado por uma expressão
- OBS: O índice de um vetor deve ser um int ou algum outro valor que possa ser promovido para int (byte, short ou char)
- Ex:
  - Se a = 1 e b = 3, o comando c [a+b] = 55, coloca o valor 55 na posição 4 do array

Detalhes dos arrays

- Arrays em java são imutáveis
- Uma vez criados, não pode sofrer alterações (não podem aumentar ou diminuir de tamanho nem mudar de tipo)
- Se você precisar de um array maior, terá que criar um novo array, copiar todos os elementos do array antigo para o array novo e desprezar o antigo para ser coletado pelo Garbage Collector

Declarando e inicializando arrays

- A declaração é bem simples:



Utilizando inicializadores de arrays

- Você pode criar seus arrays e inicializá-los no mesmo comando, caso já saiba de antemão os valores que eles deverão armazenar:
- Ex:
  - int[] n = { 10, 20, 30, 40, 50 };

- Observe que neste caso, não é necessário usar a palavra chave new

O Comando *for* avançado

- O *for* melhorado consegue iterar sobre os elementos de uma coleção sem a necessidade de um contador:

```
int[] notas = {4, 5, 7, 10, 2};
for (int nota : notas) {
        //código
}
```

- OBS: O *for* melhorado só pode iterar sobre elementos do array, não pode alterá-los. Se seu código precisa alterar valores de elementos do array, utilize o *for* tradicional.

Passagem por valor e passagem por referência

- Em Java, todas as passagens de valores são feitas por valor, i.e., métodos recebem uma cópia dos valores das variáveis
  - Alterações nos valores dessas variáveis serão visíveis apenas dentro do escopo do método
- Para os tipos por referência, o que é passado é uma cópia da referência do objeto
  - O método tem acesso ao mesmo objeto externo através dessa cópia e pode alterar valores de seus atributos que também serão visíveis externamente
  - Mas como recebe apenas uma cópia da referência, não pode fazer o objeto externo apontar para outro lugar na memória

Passando arrays como parâmetros para métodos

- A passagem de um array como parâmetro de um método funciona da mesma maneira que qualquer outro tipo de variável:

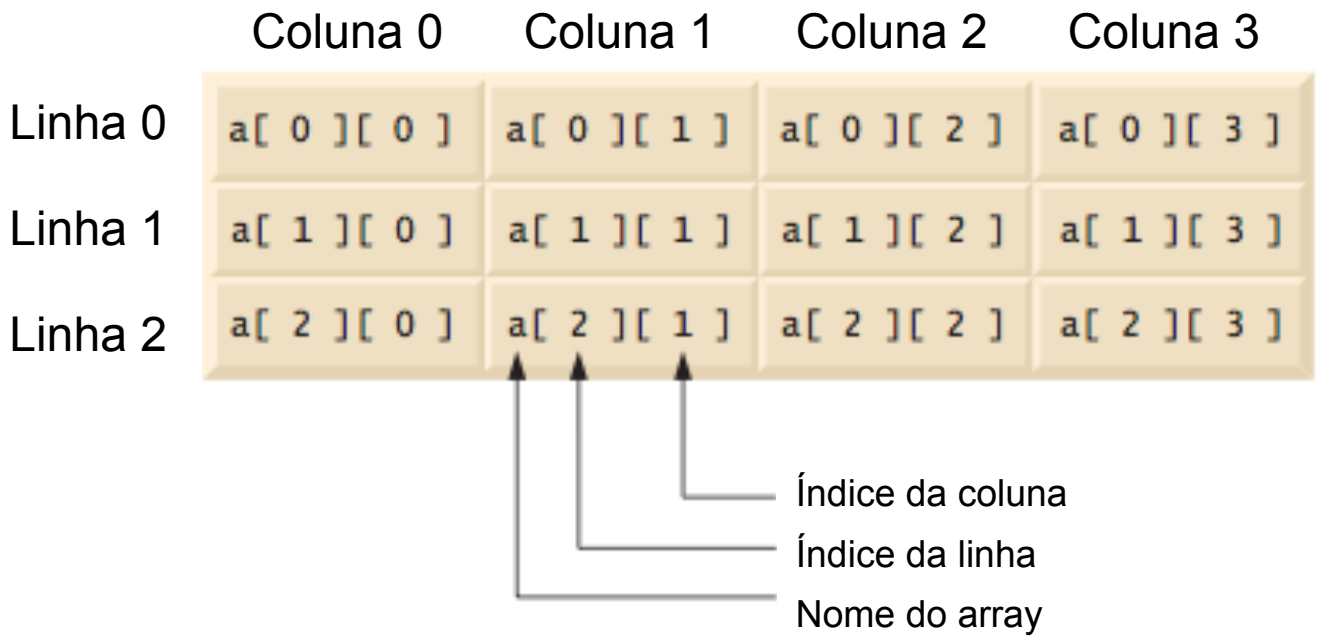
```
int[] notas = {3, 5, 7, 9};
int media = calcularMedia(notas);
```

- Como são objetos, métodos que recebem um array como parâmetro, receberão a cópia da referência desses objetos
  - Alterações nos elementos do array serão visíveis externamente

ARRAYS MULTIDIMENSIONAIS

Arrays multidimensionais

- Podemos ter arrays com quantas dimensões desejarmos



Arrays multidimensionais

- Arrays multidimencionais podem ser encarados como arrays de arrays. Ex:
  - int[ ][ ] b = { { 1, 2 }, { 3, 4 } };

- Podemos também criar arrays multidimensionais da mesma forma que criamos os arrays unidimensionais usando o new. Ex:
  - int[ ][ ] b = new int[3][4];

MÉTODOS COM LISTA DE PARÂMETROS VARIÁVEIS

Varargs

- Com listas de argumentos variáveis, você pode declarar métodos que recebem um número indefinido de parâmetros
- A declaração é bem simples e pode ser feita assim:

```
public double media(double… notas) {
    double total = 0;

    for (double valores : notas) {
        total += valores;
    }
    return total / notas.length;
}
```

Varargs

- Variáveis recebidas como varargs são tratadas como um vetor de elementos do tipo do varargs.
- No exemplo anterior, a variável notas é do tipo double[]
- Será que nós já usamos algum método do próprio Java que usa o conceito de lista de parâmetros variável?
  - O printf é um exemplo clássico do uso deste conceito:
    - System.out.printf("Olá %s", nomeUsuario);
    - System.out.printf("Nota 1: %d, Nota 2: %d, Média: %d", nota1, nota2, media);

Exercício

1. Crie um método que calcule o produto de vários números inteiros recebidos através de parâmetro definido com varargs. OBS: na sua implementação, utilize o *for* melhorado.

Projetos

- Crie um pequeno jogo da velha usando um array de duas dimensões. O seu jogo deve alternar jogadas entre o jogador da bola (O) e o jogador do X. A cada iteração, o jogo pergunta ao jogador da vez onde ele deseja colocar sua marcação, solicitando uma linha e uma coluna. Após cada jogada, o jogo deve exibir o estado atual do tabuleiro e checar se existe um ganhador. Caso exista um ganhador, o jogo deve informar quem é o ganhador e finalizar; caso não exista ganhador, permitir que mais uma jogada seja realizada; caso não existam mais jogadas possíveis e nenhum jogador tenha vencido, informar que houve um empate.

Projetos

- Crie um software que gerencie um estacionamento. A interface deve ser de texto. O Software deve saudar o usuário com uma mensagem assim que ele iniciar. A partir deste momento ele vai funcionar até que o usuário dê um comando para finalizar. Ao dar entrada num veículo, o usuário deve informar a placa e a hora da entrada, no formato hh:mm. O sistema deve armazenar esses dados, pois ao dar saída, ele deve calcular o tempo que o carro ficou no estacionamento e informar o preço a ser pago pelo cliente.