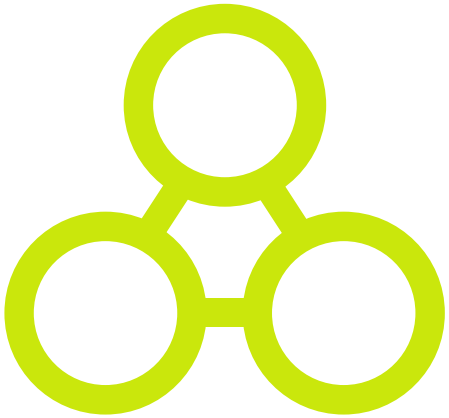
Desenvolvimento OO com Java

Membros estáticos



nemo

ontology & conceptual modeling research group

João Paulo A. Almeida

Adaptado de

Vítor E. Silva Souza

(vitorsouza@inf.ufes.br)

http://www.inf.ufes.br/~vitorsouza

Departamento de Informática

Centro Tecnológico

Universidade Federal do Espírito Santo

Atributos e métodos de classe

- Vimos até agora que atributos e métodos pertencem aos objetos:
  - Não se faz nada sem antes criar um objeto (new)!
- No entanto, há situações que você quer usá-los sem ter que criar objetos:
  - Deseja-se um atributo associado a uma classe como um todo (todos os objetos compartilham a mesma variável, similar a uma "variável global");
  - Deseja-se chamar um método mesmo que não haja objetos daquela classe criados.
    - Típico da abordagem procedural

Atributos e métodos `static`

- Usando a palavra-chave `static` você define um atributo ou método de classe ("estático"):
  - Atributo/método pertence à classe como um todo;
  - Pode-se acessá-los mesmo sem ter criado um objeto;
  - Objetos podem acessá-los como se fosse um membro de objeto, só que compartilhado;
  - O contrário não é possível: métodos `static` não podem acessar atributos/métodos não-`static` diretamente (precisa criar um objeto).

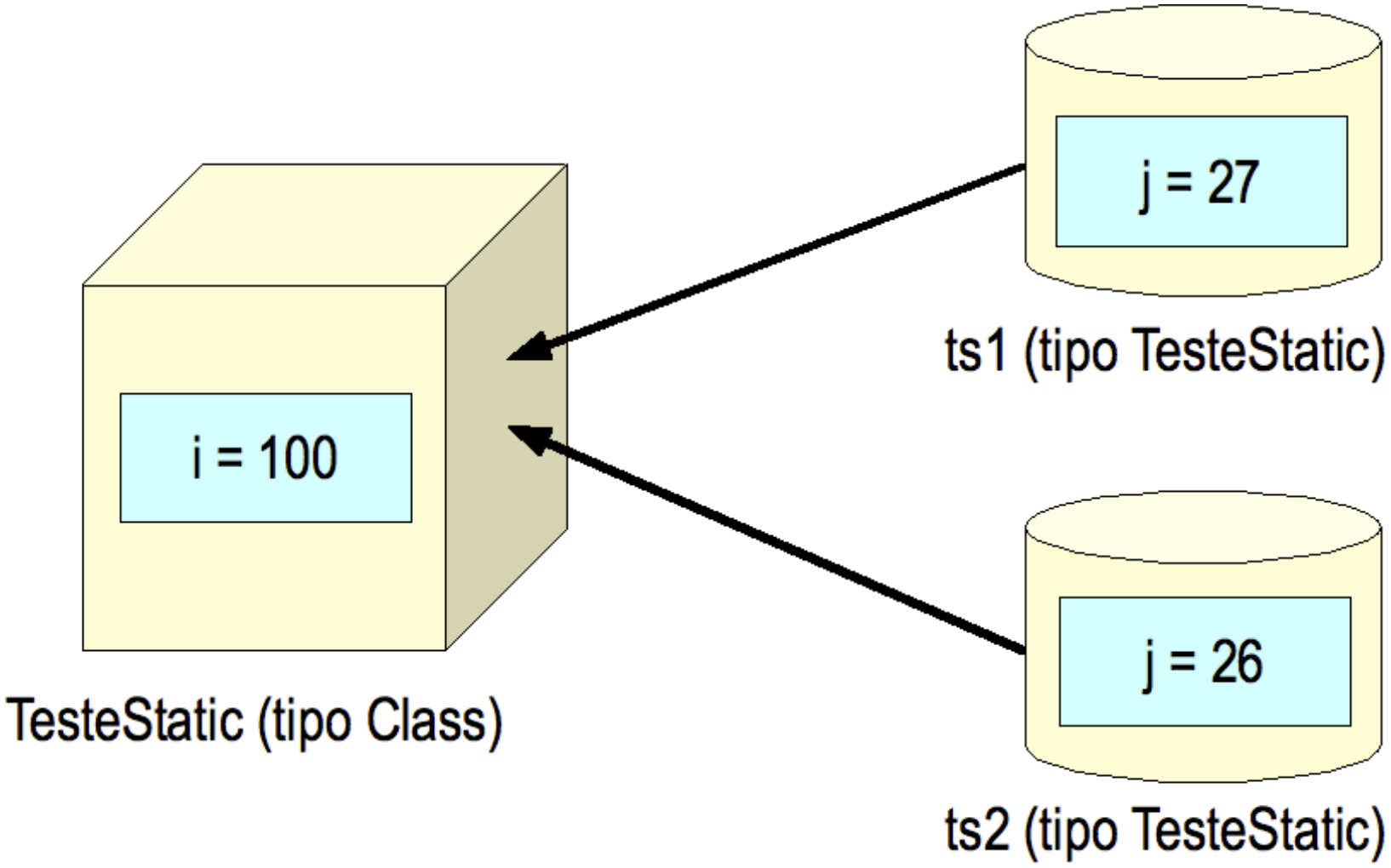
Atributos de classe (“estáticos”)

```
public class TesteStatic {
  static int i = 47;

  int j = 26;

  public static void main(String[] args) {
    TesteStatic ts1 = new TesteStatic();
    TesteStatic ts2 = new TesteStatic();

    // 47 26
    System.out.println(ts1.i + " " + ts1.j);

    // 47 26
    System.out.println(ts2.i + " " + ts2.j);

    /* Continua... */
```

Atributos de classe (“estáticos”)

```
    /* Continuação... */

    ts1.i++;
    ts1.j++;

    // 48 27
    System.out.println(ts1.i + " " + ts1.j);

    // 48 26
    System.out.println(ts2.i + " " + ts2.j);

    TesteStatic.i = 100;
    System.out.println(ts1.i);      // 100
    System.out.println(ts2.i);      // 100
  }
}
```

Armazenamento em memória



Métodos de classe (“estáticos”)

```
public class TesteStatic {
  static int i = 47;
  int j = 26;

  static void imprime(String s) {
    System.out.println(s);
  }

  static void incrementaI() { i++; }

  void incrementaJ() { j++; }

  public static void main(String[] args) {
    TesteStatic ts1 = new TesteStatic();

    /* Continua... */
```

Métodos de classe (“estáticos”)

```
    /* Continuação... */

    incrementaI();                      // OK
    TesteStatic.incrementaI();          // OK
    ts1.incrementaI();                  // OK

    // incrementaJ(); causa erro!
    // TesteStatic.incrementaJ() também!

    ts1.incrementaJ();                  // OK

    // 50 27
    imprime(ts1.i + " " + ts1.j);
  }
}
```

Armazenamento em memória

- Todos os métodos, estáticos ou não, são armazenados na área de código da classe;
- A única diferença é que métodos estáticos podem ser chamados independente de objetos criados;
- Isso é essencial para o método `main()`!

Inicialização de atributos `static`

- Atributos estáticos são inicializados somente quando a classe é usada pela primeira vez;
  - Se a classe não for usada, não são inicializados.
- São inicializados antes dos atributos não-estáticos daquela classe;
- Seguem o mesmo processo usado para atributos não-estáticos:
  1. São zerados (inicializados com seus valores *default*: 0, `false` ou `null`);
  2. Recebem os seus valores iniciais (se especificados), na ordem em que foram definidos na classe.

Blocos de inicialização estática

- No exemplo da classe `Aleatorio`, inicializamos uma variável no construtor porque não conseguíamos fazê-lo em uma só linha;
- E se esta variável for `static`?

```
class Aleatorio {
  int numero;
  Aleatorio(int max) {
     Random rand = new Random();
     numero = rand.nextInt(max);
  }
}
```

Blocos de inicialização estática

- Resolvemos a questão com blocos de inicialização estática;
- Os blocos estáticos de uma classe são executados quando a classe é usada pela 1ª vez.

```
class Aleatorio {
  static int numero;

  static {
     Random rand = new Random();
     numero = rand.nextInt(20);
  }
}
```

Blocos de inicialização não estática

- Também podemos fazer blocos de inicialização não-estática;
- Executados antes dos construtores: chamados em cada criação de objeto.

```
class Aleatorio {
  int numero;

  {
     Random rand = new Random();
     numero = rand.nextInt(20);
  }
}
```