

Capítulo 05:

Classes, métodos e parâmetros

Leonardo Moura Leitão

Marcelo Gonçalves Pinheiro Quirino

Agenda

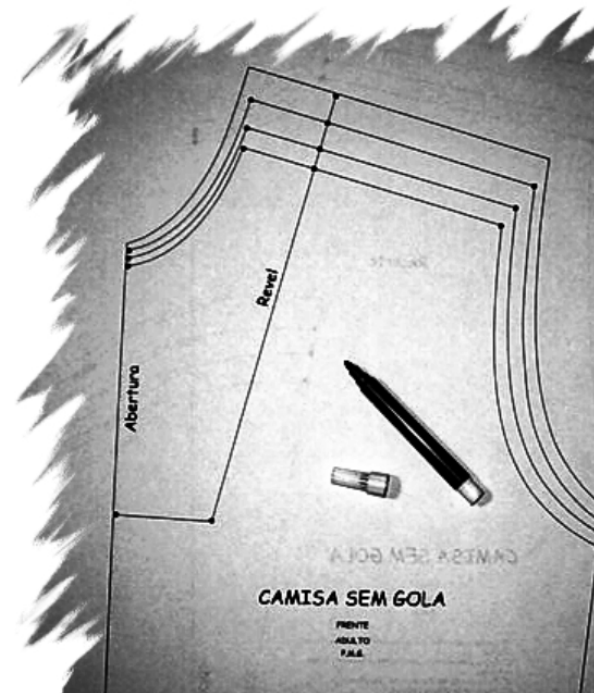
- Classe
- Objeto
- Construtor
- Acesso aos membros
- Valor x Referência
- Variáveis de instância
- Variáveis de classe

Agenda

- Método
- Passagem de parâmetros
- Variáveis locais
- Valores padrões

Classe

- A **essência** de Java é a classe, visto que **TODAS** as atividades de um programa Java ocorrem **dentro de uma classe**.
- A classe é um **molde** que define a forma do objeto.
- Objetos são **instâncias** de uma classe.



Anatomia da Classe

Variáveis de instância

Construtores

Métodos

Outras classes

Classe

- Uma classe é criada usando a palavra-chave **class**.

```
class NomeDaClasse
{
    // Variáveis de instâncias
    tipo var1;
    tipo var2;

    // Método
    tipo metodo1 (tipo nomeDoParametro)
    {
        // Corpo do método
    }
}
```

Objeto

- A classe **define um tipo**, mas não cria um objeto. Então, como os objetos são criados?
- Existe uma categoria especial de métodos, chamado **construtores**, que são responsáveis pela criação dos objetos.

```
Calculadora calc = new Calculadora();
```

Usado p/ criar
objetos

Construtor

- O método construtor determina quais ações devem ser executadas durante a criação de um objeto.
- Em Java, o construtor é definido como um método cujo nome deve ter o **mesmo nome da classe** e sem indicação do tipo de retorno - nem mesmo void.

Construtor

```
class Data  
{
```

```
    int dia;  
    int mes;  
    int ano;
```

Construtor padrão
(Sem parâmetros)

```
    Data () {}
```

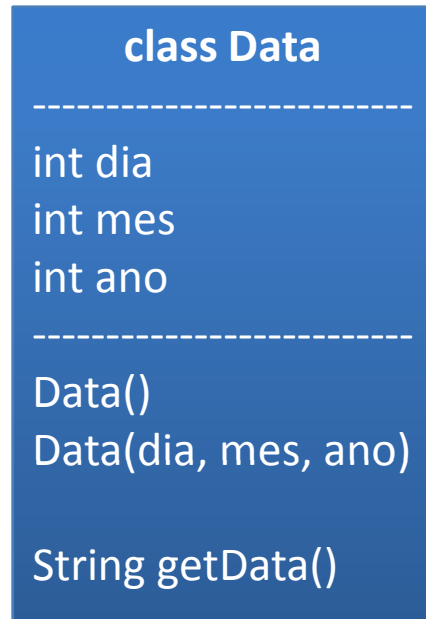
Construtor com parâmetros

```
    Data ( int dia, int mes, int ano )  
    {  
        this.dia = dia;  
        this.mes = mes;  
        this.ano = ano;  
    }
```

```
    ...
```

Usado p/ referenciar a instância
corrente da classe

Acesso aos membros



`Data data1 = new Data();`



`Data data2 = new Data(1, 8, 2009);`



objeto.membro



`data1.mes = 5`

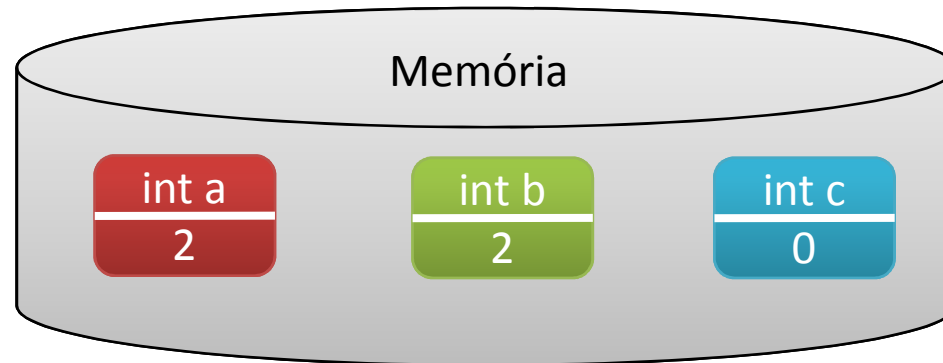
`data2.getData()`

1/8/2009

Valor x Referência

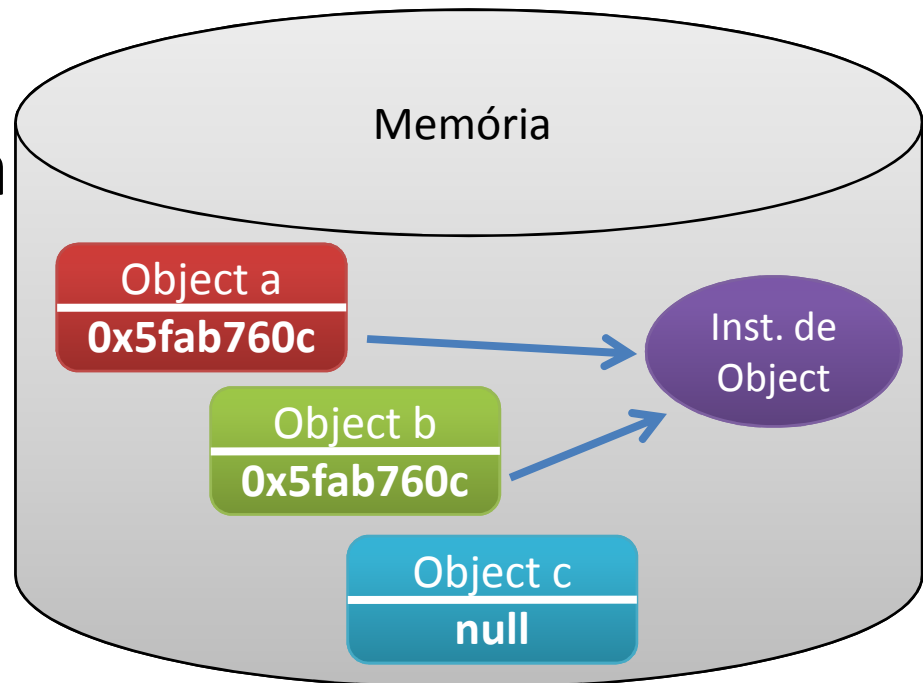
- Tipos primitivos

```
int a = 2;  
int b = a;  
int c;
```



- Variáveis de referência

```
Object a = new Object();  
Object b = a;  
Object c;
```



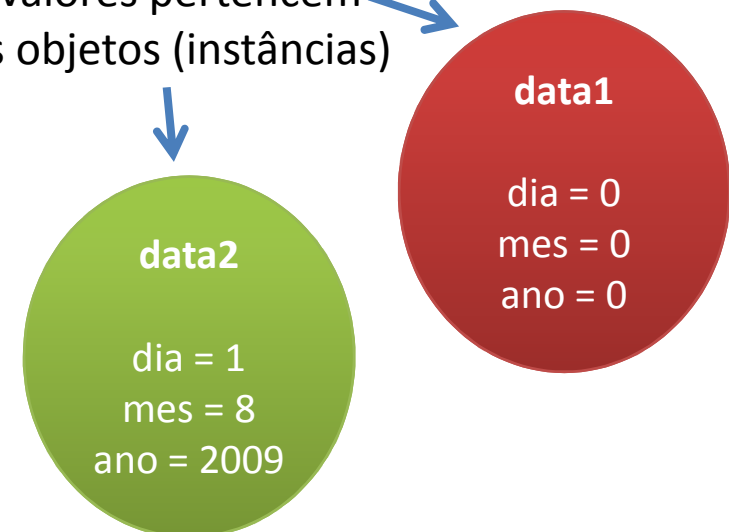
Variáveis de instância

- Uma variável que pertence a uma instância de uma classe.
- As variáveis de instância são definidas no corpo da classe, mas fora dos métodos.

```
class Data
{
    int dia;
    int mes;
    int ano;

    String getData ()...
```

Os valores pertencem
aos objetos (instâncias)



Variáveis de classe

- Estas variáveis são declaradas com o modificador **static**. Isso informa ao compilador que há exatamente **uma única cópia** desta variável, independente do número de instâncias da classe.

```
class Data
{
    int dia;
    int mes;
    static int ano;

    String getData ()...
```

class Data

int ano = 2008
...

Cópia
Única!

data1

dia = 0
mes = 0

data2

dia = 1
mes = 8

Método

- Métodos são sub-rotinas que **manipulam os dados** (variáveis de instância) definidos numa classe.
- Em muitos casos, os métodos (**públicos**) provêem **acesso** aos dados (**privados**).
- Boa prática: cada método executa **somente uma tarefa**.

Anatomia do Método

retorno

Assinatura do método

nomeDoMetodo(parametros)

Corpo do método

Método

```
class Data
```

```
{
```

```
    int dia;
```

```
    int mes;
```

```
    int ano;
```

Retorno do
Método

Nome do Método

```
String getData ()
```

Lista de parâmetros
(pode estar vazia)

```
{
```

```
    return dia + "/" + mes + "/" + ano;
```

```
}
```

Não possui
retorno!

Usado p/ retornar o
valor do método

```
void exibirDia ()
```

```
{
```

```
    System.out.println( dia );
```

```
}
```

```
}
```


Método main

- Porta de entrada de um programa Java

```
class MeuPrograma
{
    public static void main ( String[] args )
    {
        System.out.println( "Param1: " + args[0] );
        System.out.println( "Param2: " + args[1] );
        System.out.println( "Param3: " + args[2] );
    }
    ...
}
```

C:\WINDOWS\system32\cmd.exe

```
C:\>java MeuPrograma 123 parametro2 EsseEOParametro3
Param1: 123
Param2: parametro2
Param3: EsseEOParametro3
C:\>
```

Passagem de parâmetros

- O número de parâmetros passados deve ser igual ao da definição do método.
- Cada parâmetro, individualmente, deve ter tipo compatível com o da definição do método.

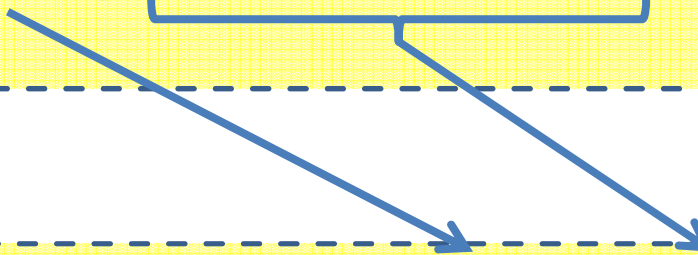
Passagem de parâmetros

- Para **tipos primitivos**, a **passagem é por valor**, ou seja, o valor (e não o endereço) da variável é passado.
- Para **objetos**, uma **cópia da referência** ao objeto é passada, e não uma cópia do objeto.
- Uma expressão pode ser usada como argumento.

Passagem de parâmetros

- Chamada de um método

```
int valor1 = 1;  
int valor2 = 2;  
int valor3 = 3;  
  
somar( valor1, valor2 + valor3 );  
...
```



```
public static int somar(int a, int b)  
{  
    return a + b;  
}
```

Variáveis locais

- São usadas para armazenar o estado temporário de um método e somente são **acessíveis dentro desse**, sendo abandonadas, automaticamente, na sua saída.

```
class Data {  
    ...  
    String getData () {  
        String retorno = null;  
        retorno = dia + "/" + mes + "/" + ano;  
        return retorno;  
    } ...  
}
```

Valores padrões

- Variáveis globais **são inicializadas** com os valores padrões (esses valores dependem do tipo).

Tipo	Valor (Padrão)	Tipo	Valor (Padrão)
byte	0	short	0
int	0	long	0
float	0.0	double	0.0
char	0 ou '\u0000'	boolean	False
Objeto	null		

- Variáveis locais **não são inicializadas**, por isso, o programador deve iniciá-las antes de usar!