

INTRODUÇÃO A CLASSES

Ramon Lummertz

RELEMBRANDO QUE...

- Em java os dados primitivos são
 - inteiros
 - int
 - short
 - byte
 - long
 - ponto flutuante
 - double
 - float
 - boolean
 - char

RELEMBRANDO QUE...

Java é fortemente tipado

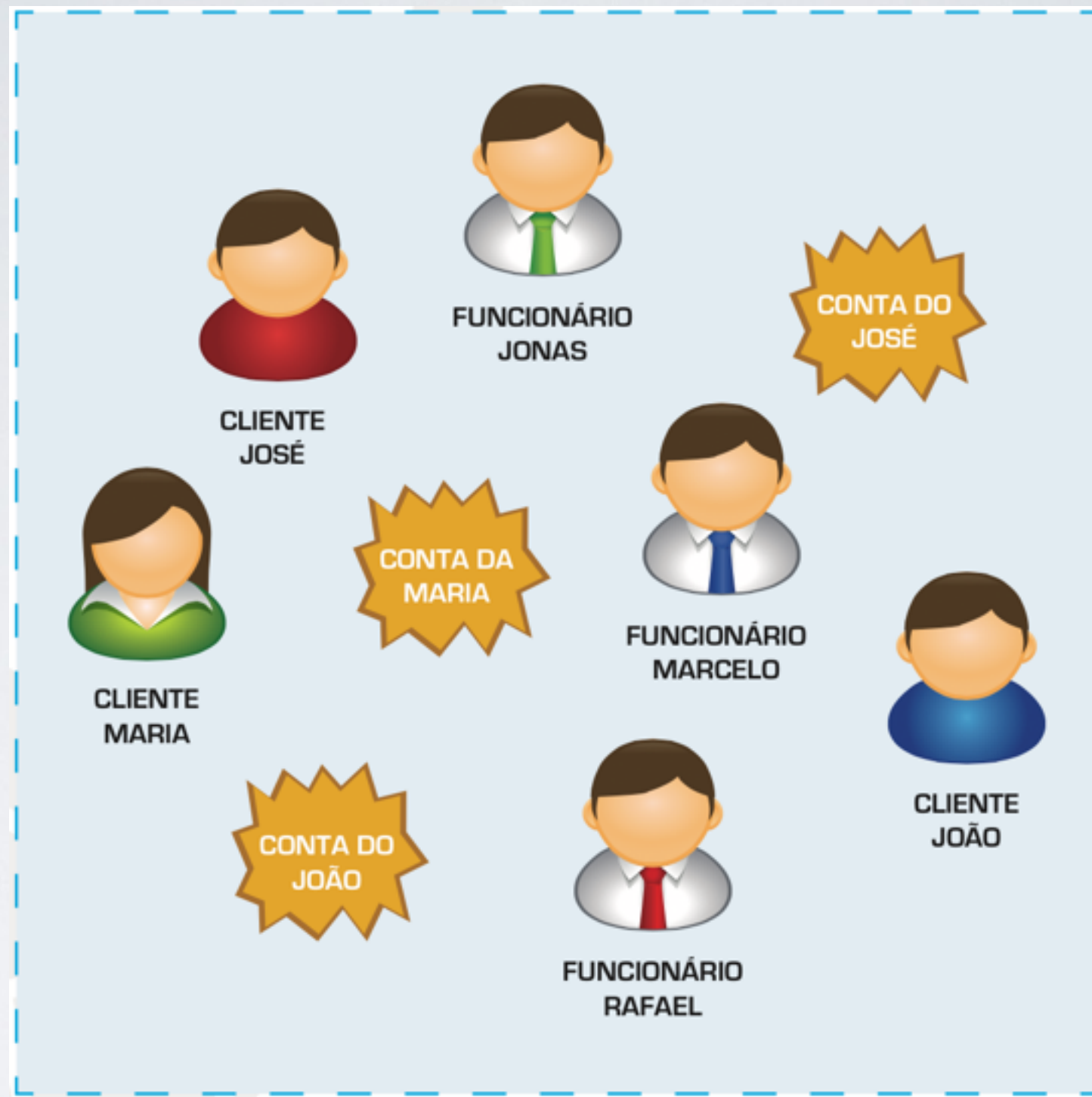
Java é caseSensitive

ORIENTAÇÃO A OBJETOS

DOMÍNIO E APLICAÇÃO

Um **domínio** é composto pelas entidades, informações e processos relacionados a um determinado contexto.

Uma **aplicação** pode ser desenvolvida para automatizar ou tornar factível as tarefas de um domínio. Portanto, uma aplicação é basicamente o “reflexo” de um domínio.



DOMINIO BANCARIO

OBJETOS ATRIBUTOS E MÉTODOS

As entidades identificadas no domínio devem ser representadas de alguma forma dentro da aplicação correspondente.

Nas aplicações orientadas a objetos, as entidades são representadas por **objetos**.

Uma aplicação orientada a objetos é composta por objetos. Em geral, um objeto representa uma entidade do domínio.

Um objeto é composto por atributos e métodos.

OBJETOS ATRIBUTOS E MÉTODOS

Atributos

- Um atributo é uma variável que pertence a um objeto.
- Os dados de um objeto são armazenados nos seus atributos.

OBJETOS ATRIBUTOS E MÉTODOS

Métodos

○ próprio objeto deve realizar operações de consulta ou alteração dos valores de seus atributos. Essas operações são definidas nos **métodos** do objeto. Os métodos também são utilizados para possibilitar interações entre os objetos de uma aplicação.



FUNCIONÁRIO
JONAS



FUNCIONÁRIO
MARCELO



FUNCIONÁRIO
RAFAEL



CONTA DA
MARIA



CONTA DO
JOSÉ



CONTA DO
JOÃO



CLIENTE
MARIA



CLIENTE
JOSÉ



CLIENTE
JOÃO

Funcionario
nome = Rafael Cosentino codigo = 1 salario = 1000
tiraFerias aumentaSalario mudaHorario

Funcionario
nome = Marcelo Rezende codigo = 2 salario = 2000
tiraFerias aumentaSalario mudaHorario

Funcionario
nome = Jonas Hirata codigo = 3 salario = 3000
tiraFerias aumentaSalario mudaHorario

Conta
saldo = 2000 numero = 1
deposita saca geraExtrato

Conta
saldo = 4000 numero = 2
deposita saca geraExtrato

Conta
saldo = 8000 numero = 3
deposita saca geraExtrato

Cliente
nome = João do Pulo codigo = 1
mudaEndereco fazEmprestimo

Cliente
nome = José do Caixão codigo = 2 mudaEndereco fazEmprestimo

Cliente
nome = Maria do Bairro codigo = 3 mudaEndereco fazEmprestimo

CLASSES

Antes de um objeto ser criado, devemos definir quais serão os seus atributos e métodos. Essa definição é realizada através de uma **classe** elaborada por um programador. A partir de uma classe, podemos construir objetos na memória do computador que executa a nossa aplicação

CLASSE



CLASSE



Figura 3.7: Diversas casas com características diferentes

Assim como duas casas construídas a partir da mesma planta podem possuir características diferentes.

CLASSE EM JAVA

```
1 class Conta {  
2     double saldo;  
3     double limite;  
4     int numero;  
5 }
```

CRIANDO OS OBJETOS

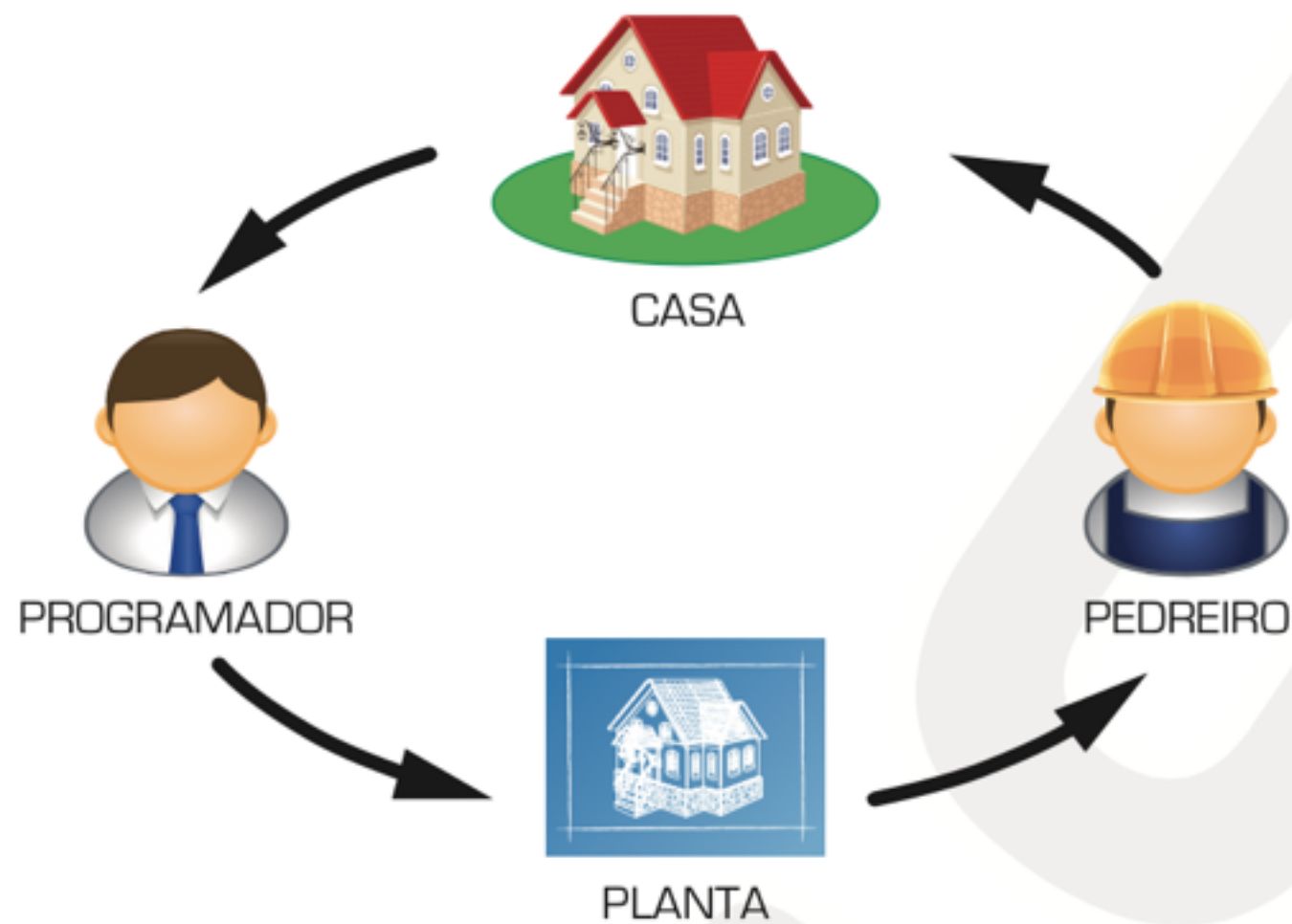


Figura 3.8: Construindo casas

ALOCANDO UM OBJETO

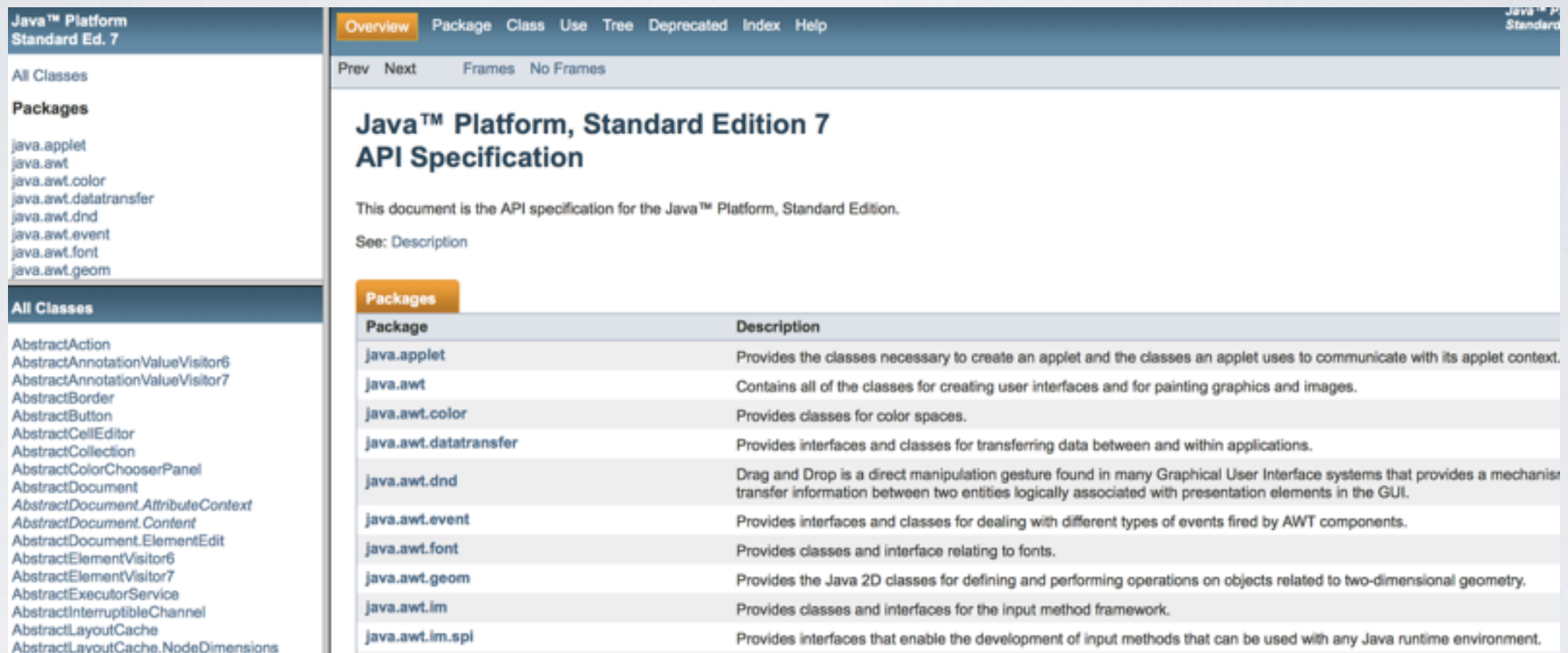
Ao utilizar o comando **new**, um objeto é alocado em algum lugar da memória. Para que possamos acessar esse objeto, precisamos de sua referência. O comando **new** devolve a referência do objeto que foi criado.

```
1 Conta referencia = new Conta();
```

MANIPULANDO ATRIBUTOS

```
1 Conta referecia = new Conta();  
2  
3 referecia.saldo = 1000.0;  
4 referecia.limite = 500.0;  
5 referecia.numero = 1;  
6  
7 System.out.println(referecia.saldo);  
8 System.out.println(referecia.limite);  
9 System.out.println(referecia.numero);
```


USANDO CLASSE QUE JA EXISTE



Java™ Platform
Standard Ed. 7

Overview Package Class Use Tree Deprecated Index Help

Prev Next Frames No Frames

Java™ Platform, Standard Edition 7 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.

All Classes

- AbstractAction
- AbstractAnnotationValueVisitor6
- AbstractAnnotationValueVisitor7
- AbstractBorder
- AbstractButton
- AbstractCellEditor
- AbstractCollection
- AbstractColorChooserPanel
- AbstractDocument
- AbstractDocument.AttributeContext
- AbstractDocument.Content
- AbstractDocument.ElementEdit
- AbstractElementVisitor6
- AbstractElementVisitor7
- AbstractExecutorService
- AbstractInterruptibleChannel
- AbstractLayoutCache
- AbstractLayoutCache.NodeDimensions

<http://docs.oracle.com/javase/7/docs/api/>

CLASSE SCANNER

java.util

Class Scanner

java.lang.Object
java.util.Scanner

All Implemented Interfaces:

Closeable, AutoCloseable, Iterator<String>

```
public final class Scanner
extends Object
implements Iterator<String>, Closeable
```

A simple text scanner which can parse primitive types and strings using regular expressions.

A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various next methods.

For example, this code allows a user to read a number from `System.in`:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

MÃOS A OBRA!

EXERCÍCIOS

- 1- Implemente uma classe chamada Aluno para definir os objetos que representarão os alunos de uma escola. Essa classe deve declarar três atributos: o primeiro para o nome, o segundo para o CPF e o terceiro para a data de nascimento dos alunos.
- 2- Faça uma classe chamada TestaAluno e crie dois objetos da classe Aluno atribuindo valores a eles. A classe também deve mostrar na tela as informações desses objetos.
- 3 - Em uma escola, além dos alunos temos os funcionários, que também precisam ser representados em nossa aplicação. Então implemente outra classe chamada Funcionário que contenha dois atributos: o primeiro para o nome e o segundo para o salário dos funcionários.
- 4 -Faça uma classe chamada TestaFuncionario e crie dois objetos da classe Funcionário atribuindo valores a eles. Mostre na tela as informações desses objetos