# **Programação orientada a objetos**

Java:

* Apresentada pela Sun Microsystems em agosto de 1995.
* Nasceu em ambientes UNIX.
* 2009: Oracle compra a Sun.

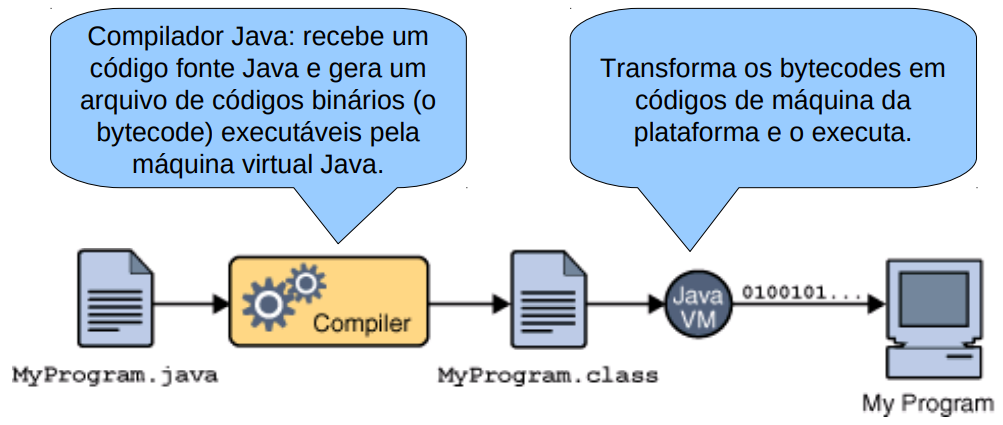
A tecnologia Java pode ser entendida como:

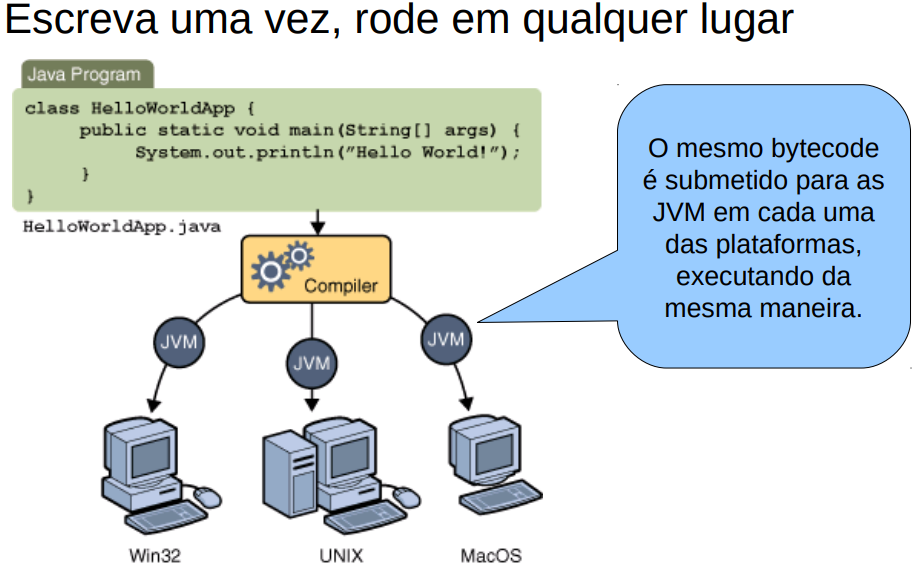
* Uma linguagem de programação;
* Um ambiente de execução de aplicações (JRE);
* Um ambiente de desenvolvimento (SDK)‏;
* Orientada a Objetos;
* Similar ao C++, mas simplificada
  + Não implementa herança múltipla;
  + Abstrai o conceito de ponteiro.
* Enorme biblioteca de classes/funções;
* Voltada para a criação de aplicações distribuídas (biblioteca TCP/IP)‏;
* Permite executar algoritmos em concomitância ou simultaneamente;
* Execução dinâmica e sob demanda;
* Reflexiva (RTTI – Runtime Type Identification);
* Maior segurança
  + Não permite manipulação direta de ponteiros;
  + Coleta automática de lixo (garbage collector).
* Portabilidade;
* Compilador Java gera código para um processador virtual (máquina Java)‏, o chamado Bytecode;
* Cada plataforma onde o aplicativo Java roda, deve ter um ambiente de execução (JRE - Java Runtime Environment)‏ que interpreta cada bytecode e gera instruções da máquina em questão;
* Existem otimizadores para evitar a perda de performance pela interpretação, os chamados compiladores Just-In-Time.

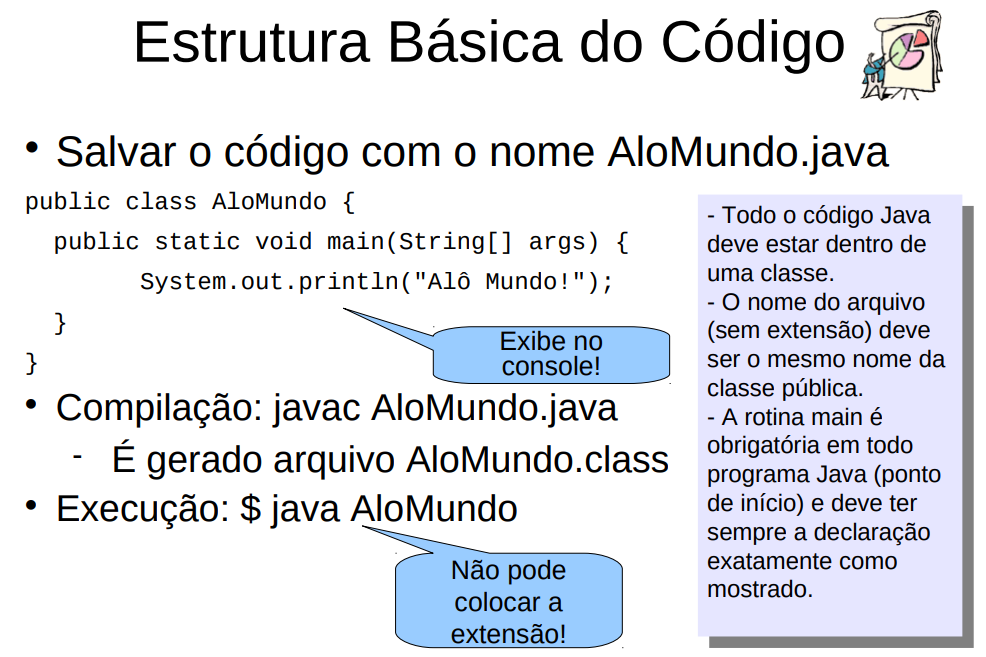
Ambiente Desenvolvimento Java

* JVM (Java Virtual Machine)‏
  + Carrega, verifica e executa bytecode Java numa plataforma (hardware + sistema operacional)‏;
  + Existem JVMs para várias plataformas.
* Coletor de Lixo (Garbage collection)
  + Encarrega-se de liberar automaticamente memória alocada pelo programa que não é mais necessária.

Processo de Compilação e Execução







Identificadores e Palavras Reservadas

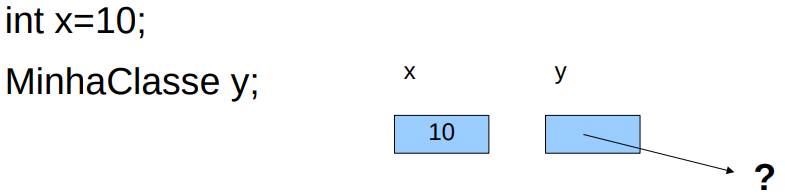
* Identificador: qualquer nome dado pelo programador dentro de um programa.
  + Exemplos: variável, classe, funções, etc.
* Java é sensível à caixa (case sensitive);
* 1°caractere deve ser letra, \_ ou $;
* Não pode ter espaço em branco;
* Não pode ser igual a palavra reservada:
  + abstract, boolean, break, byte, case, catch, char, class, public, continue, default, do, double, else, enum, extends, false, final, finally, float, for, private, if, implements, import, instanceof, int, interface, long, native, new, null, package, private, protected, public, return, short, static, const, super, switch, strictfp, synchronized, this, throw, throws, transient, true, try, void, volatile, while.

Comentários em Código

* Comentário de uma linha com // comentário;
  + Todo resto da linha é ignorado.
* Comentário de bloco com /\* comentário \*/;
* JavaDoc: permite gerar documentação a partir dos comentários do código fonte.

Tipos de Dados

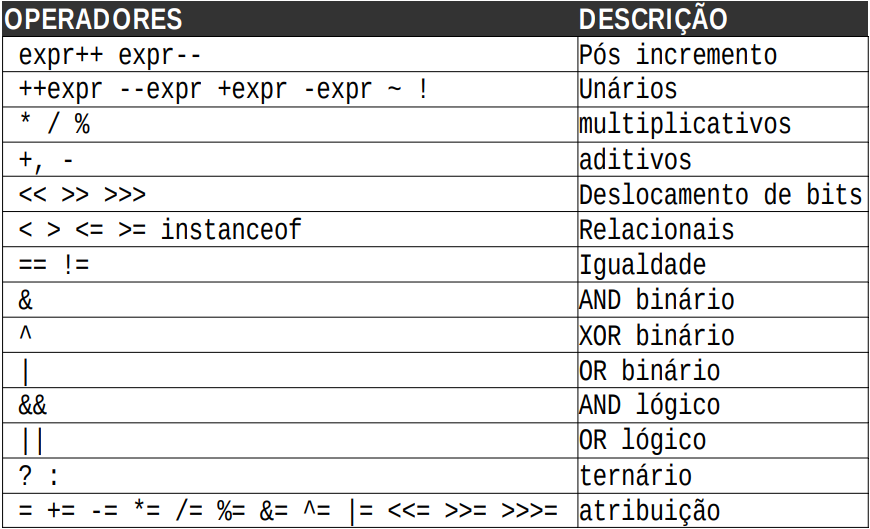
* Duas categorias: primitivos e de referência;
* Tipos primitivos: dados simples ou escalares;
* Tipos de referência: guardam, ao invés do dado propriamente dito, um “ponteiro” para onde o objeto está supostamente instanciado (heap)‏;



Tipos Primitivos:

* Compatíveis entre plataformas (diferente do C)‏;
* **Boolean**: valor lógico true ou false;
* Char: caractere em notação Unicode (16 bits)‏. Pode ser usado com inteiro entre 0 e 65535;
* **Byte**: inteiro de 8 bits. Assume valores entre -128 e 127;
* **Short:** inteiro de 16 bits. Assume valores entre -32768 e 32767;
* **Int**: inteiro de 32 bits. Valores entre -2.147.483.648 e 2.147.483.647.
* **Long**: inteiro 64 bits. Valores entre -263 e 263-1.
* **Float**: representa números em 32 bits na notação de ponto flutuante.
* **Double**: representa números em 64 bits na notação de ponto flutuante. Da mesma forma que o float, não deve ser usado para representar valores precisos (usar BigDecimal)‏.
* Para todo tipo primitivo existe uma classe (Wrapper)‏ correspondente
  + Classe Integer para o tipo int, Character para o tipo char, Double para double, etc.
* Permite o programador Java trabalhar apenas com variáveis do tipo referência;
* Boxing e Unboxing permitem uso combinado
  + − int y = 4; Integer x = 5 + y;
* Tipos primitivos:
  + Mais rápidos, pois podem ser manipulados diretamente por instruções elementares de hardware.

Operadores em Java:

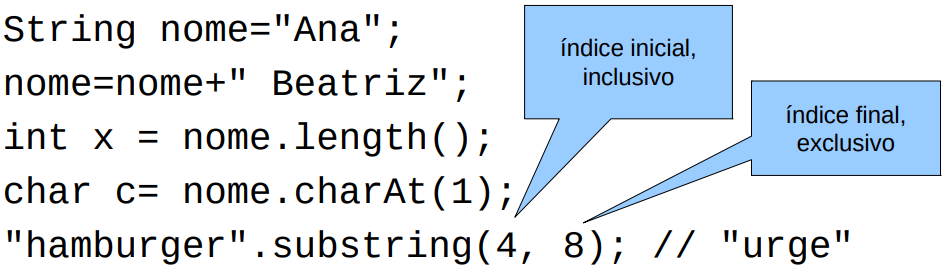


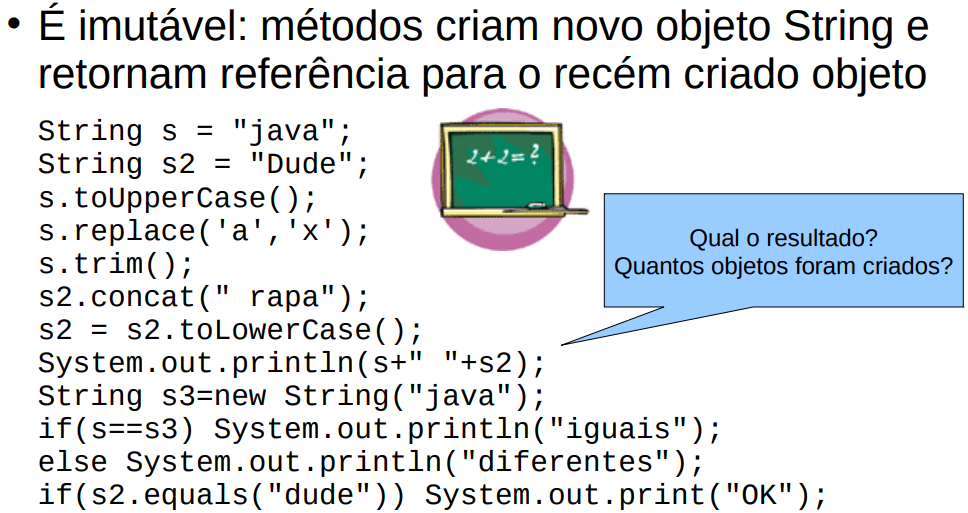
Variáveis

* Tipo caractere:
  + Armazena um caractere Unicode de 16 bits;
  + De 0 a 255 obedece a tabela ASCII estendida;
  + Constante representada entre apóstrofes ou pelo valor numérico.
  + Exemplos:
    - char c='a';
    - char b='C'; char quebra='\n'; // escape newline
    - char a=65; // letra A maiúscula Unicode

String em Java

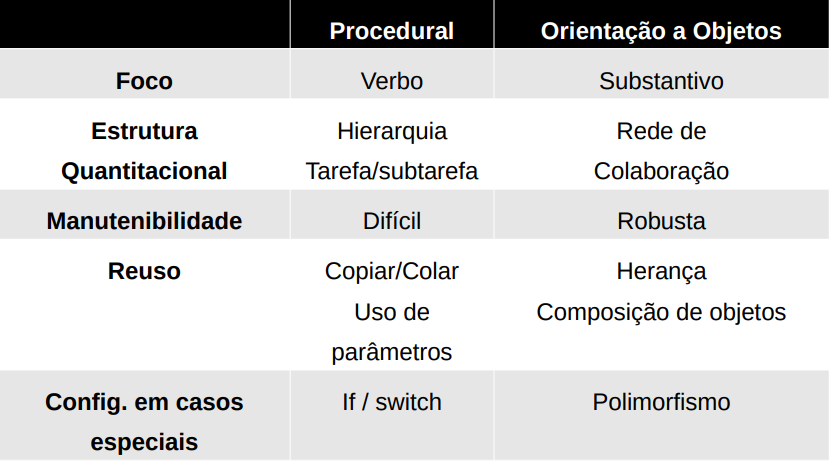
* String é uma classe, não um tipo primitivo;
* Mesmo sendo uma classe, tem um tratamento especial dentro do código Java;
* Possui operações de concatenação, extração de substring, procura de trecho, etc.



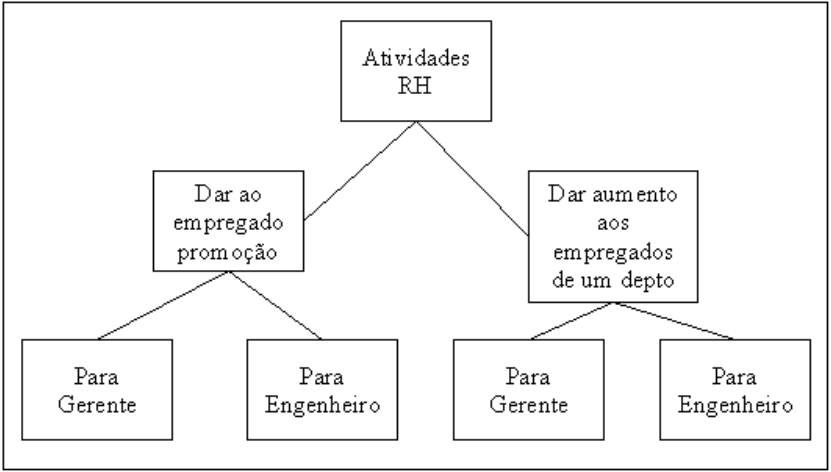


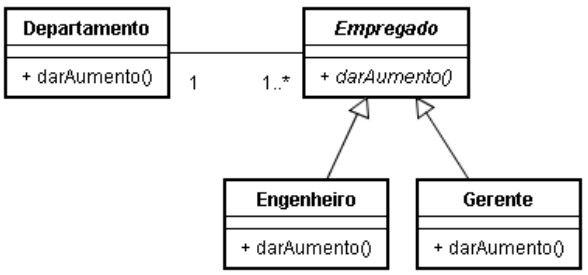
**\*Um vetor é declarado em Java do seguinte modo: int [] vetor = nem int [10];**

**Paradigma Procedural x OO**

****

**Exemplo Paradigma Procedural x OO**

****

****

**Princípios OO**

**O que é objeto?**

* Entre as várias definições, para o fim de POO podemos destacar: “Algo perceptível, conhecido, pensado ou representado, por qualquer um dos nossos sentidos, em caráter estável”;
* Pode ser algo físico (um carro, uma caneta, um sapo, etc.) ou algo puramente mental (uma conta bancária, uma equação, uma sessão de aula, etc.).

**Pensar**

* Substantivo → características;
* Verbos → comportamento.

**Orientação a Objetos**

* Procura imitar na estrutura do programa a forma como enxergamos os elementos do contexto-problema observado;
* O mundo é percebido por nós como tendo "coisas", coisas essas que têm características e comportamentos → objetos, que interagem entre si.

**Classe**

* Abstração do objeto do mundo real, como um modelo simplificado para ser representado no computador;
* Definem características e comportamentos afins a todos os objetos dessa classe
* A partir de uma classe, podemos criar 1 ou mais objetos (instanciação).

**Objeto**

* Instância de uma classe.

**Propriedade**

* Característica comum a todos os objetos daquela classe.

**Operação**

* Ação que objetos de uma classe podem executar. Um método público é uma implementação de uma operação.

**Mensagem**

* Ocorre quando um objeto (emissor), durante a execução de um método, invoca o método de outro objeto (receptor)

**Encapsulamento e Acoplamento**

* Objetos precisam colaborar entre si
* Interface do objeto: conjunto de mensagens que ele pode receber

**Encapsulamento:**

* Capacidade de esconder detalhes de implementação.

**Acoplamento:**

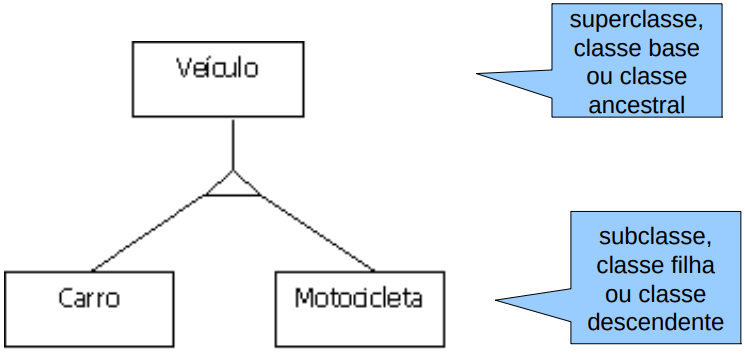
* Quanto menos um objeto-usuário souber dos detalhes da implementação de um objeto receptor da mensagem, menos o objeto-usuário será afetado por eventuais alterações no receptor

**Coesão**

* Medida que determina o quão objetivo é o propósito de uma classe;
* Quanto mais coesa, menor o nível de acoplamento com outras classes;
* Princípio da Responsabilidade Singular
  + Single Responsability Principle.
  + Uma classe deve ter somente um motivo para mudar.

**Herança**

* Criação de classes baseadas em classes já existentes para conseguir uma reutilização de código segura;
* É a capacidade de criarmos novos tipos de objetos (classes) a partir de tipos já existentes;
* A ideia é especializar as características herdadas de uma classe mais genérica. A classe filha herda todas as características e comportamentos da classe pai. Isso cria o que se chama de árvore de hierarquia.

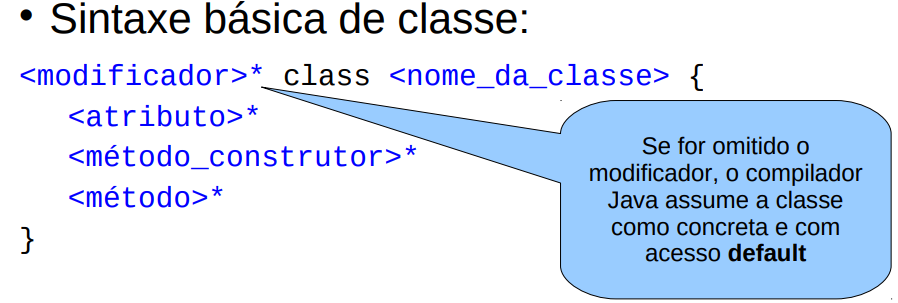
****

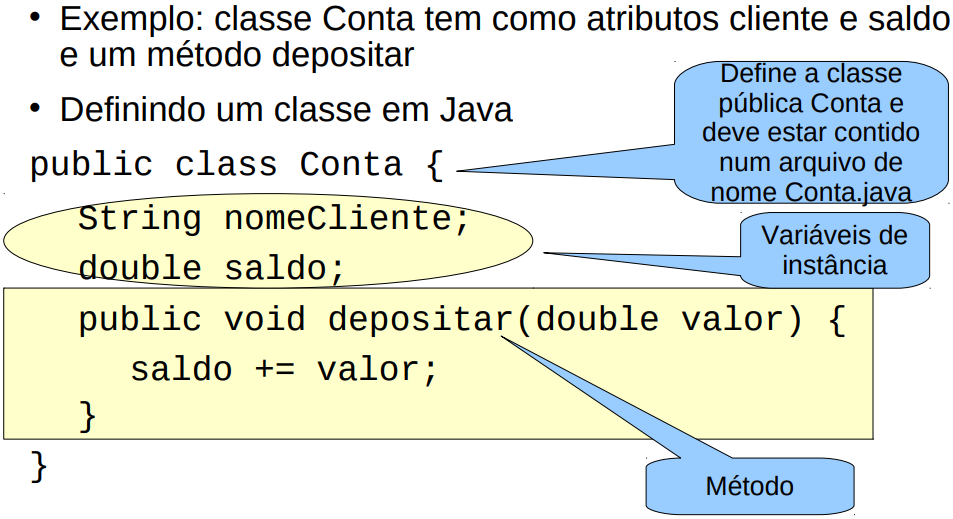
**Polimorfismo**

* Origem grega: significa "muitas formas";
* Nasce da separação da interface de objeto das implementações dessa interface;
* Uma mesma mensagem que pode ser enviada para vários objetos, comporta-se de maneira diferente dependendo da implementação definida na classe daquele objeto.
* **Princípios da POO**
* Tudo é um objeto;
* Programa: grupos de objetos que se relacionam através de mensagens (chamadas às funções;
* Um objeto é a composição de outros objetos;
* Todo objeto tem um tipo (classe ou interface);
* Todos objetos de um mesmo tipo podem receber as mesmas mensagens;
* É fundamental uma boa modelagem que mapeie em objetos os elementos do domínio.

**Classe:**

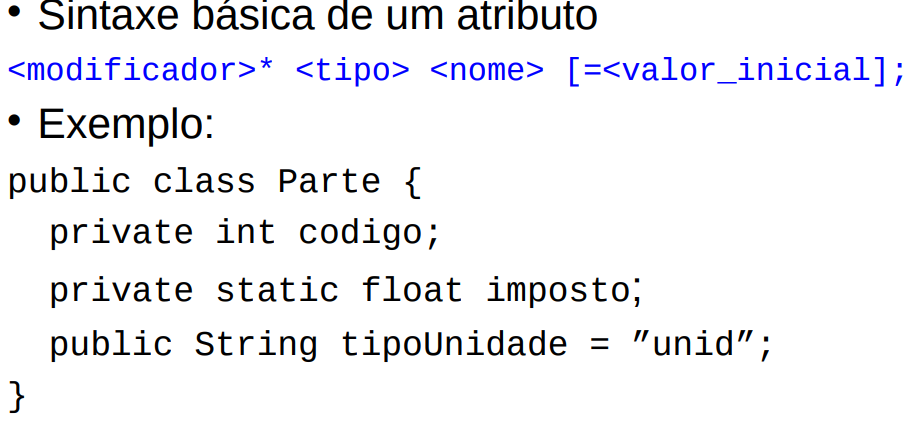
* Define uma mesma forma e comportamento;
* A partir de uma classe não abstrata, pode-se criar um ou mais objetos (instanciação);
* Cada objeto pode ter um estado, que é o conjunto de valores dados a cada característica definida pela classe a qual o objeto pertence;
* Objetos podem ter estados diferentes entre si, mesmo sendo da mesma classe;
* A definição de uma classe é baseada na declaração dos atributos e métodos.

****

****

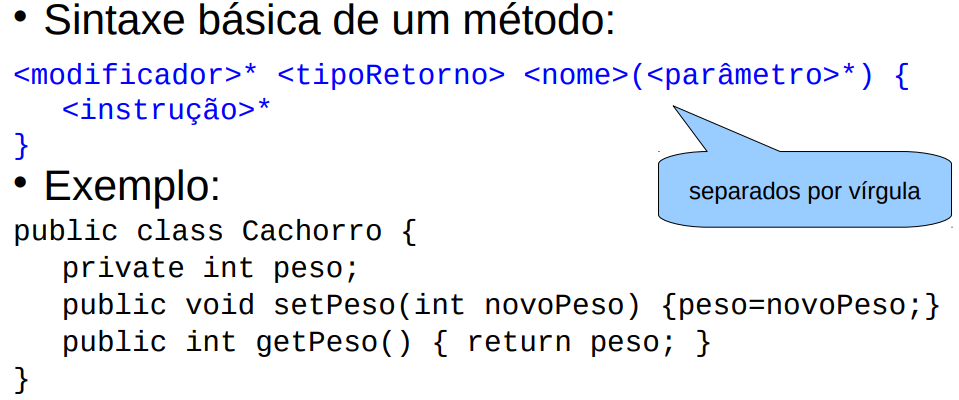
**Atributos**

* Os atributos podem ser de Instância ou de Classe
* Atributo de Instância
  + Define a característica de um objeto em particular;
  + Cada objeto definirá o seu valor para o atributo de instância;
  + O valor de um atributo de instância pode ser definido no momento da criação do objeto (instanciação)‏ ou durante a vida do objeto;
  + Caso um valor não seja definido explicitamente, o Java atribui um valor padrão.
* Variável de Classe
  + Define uma característica da classe, não de uma instância específica;
  + O valor da variável é único para todos os objetos dessa classe;
  + Para definir uma variável de classe, devemos usar o modificador static na definição;
  + Cenário: imagine uma classe Funcionario, onde todos os funcionários não podem ter salários maiores que um teto. O atributo teto poderia ser definido como um atributo de classe de Funcionario.

****

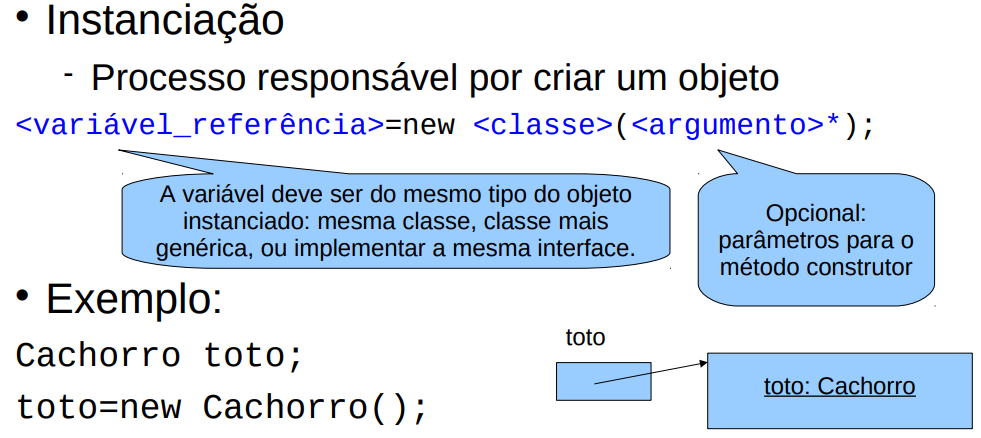
**Métodos**

* Definem o comportamento de uma classe;

****

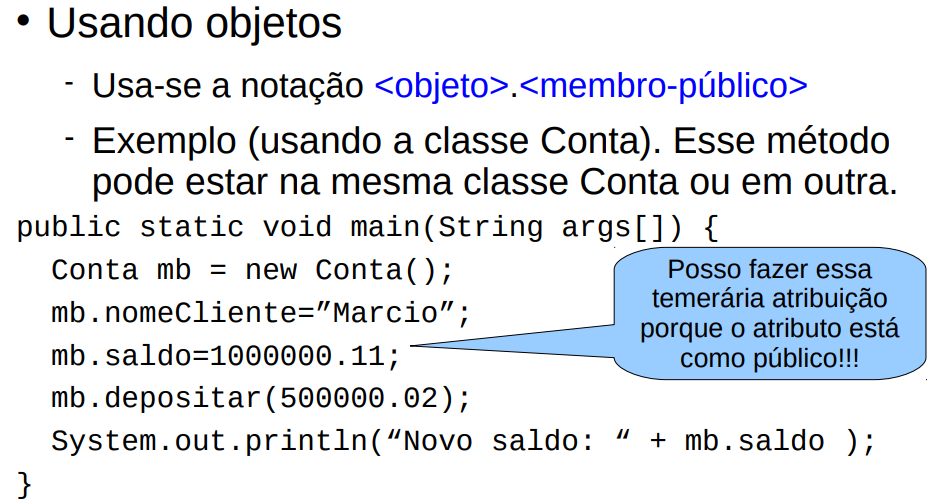
**Objetos**

* Definição: abstração de um elemento concreto ou abstrato, que pertence a uma classe e que pode ser identificado de maneira única;
* Estado: valores de seus atributos (variáveis)‏ e o relacionamento com outros objetos, que pode mudar várias vezes durante seu ciclo de vida;
* Comportamento: conjunto de métodos (classe);
* Identidade: cada objeto tem identificação exclusiva, mesmo tendo os objetos o mesmo estado.

****

**Processo de instanciação:**

* Uma classe é requisitada para criar um representante dela (objeto);
* Esse recém criado objeto terá todas as variáveis e métodos definidos pela classe;
* Cada classe tem um ou mais métodos especiais chamados construtores, responsáveis por criar o objeto, inicializando o estado do objeto;
* Caso uma classe não tenha um método construtor explícito, o Java coloca um construtor padrão.

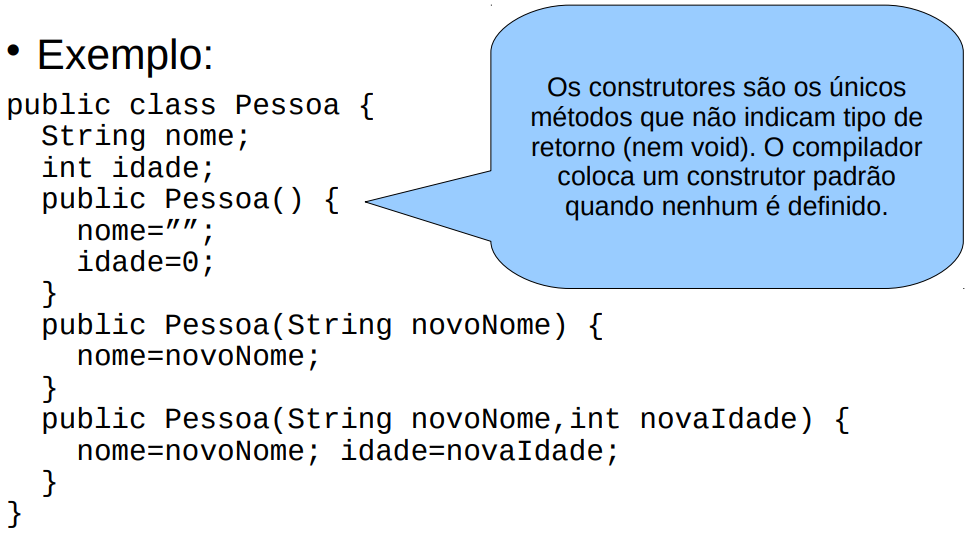
****

**Sobrecarga de métodos**

* Ocorre quando dois ou mais métodos têm o mesmo nome, mas com assinaturas diferentes;
* Assinatura do método: consiste do nome mais os tipos de parâmetros que ele recebe.

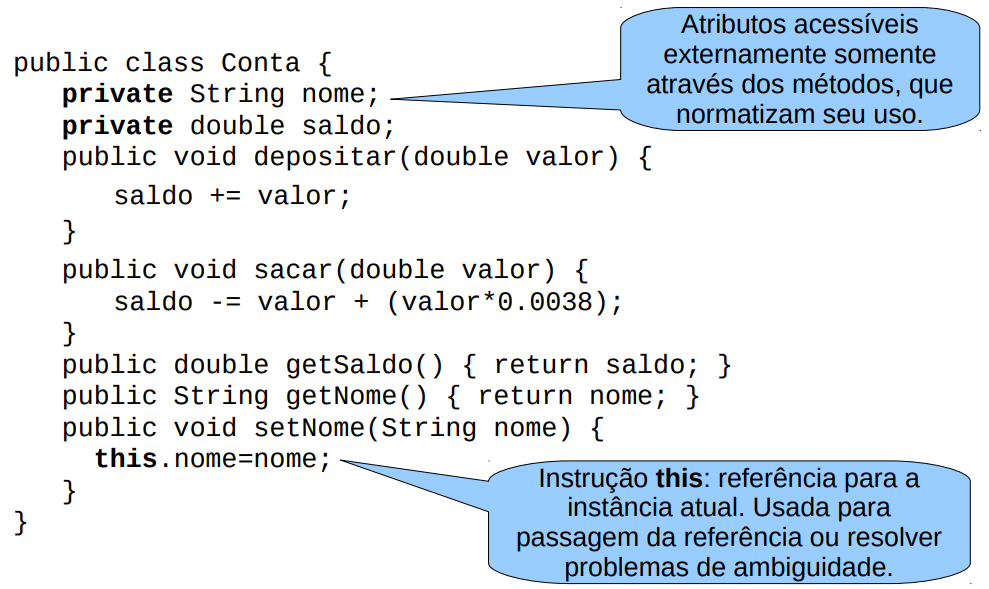
**Construtores**

* Método especial contido na classe, usado na instanciação de um novo objeto para preparar seu estado inicial de forma apropriada;
* O desenvolvedor pode prover um ou mais métodos construtores personalizados;
* Caso não seja definido um construtor, Java provê implicitamente um construtor padrão;
* O nome dos métodos construtores deve ser o mesmo do nome da classe;
* Sintaxe similar ao método, sem tipo de retorno.

****

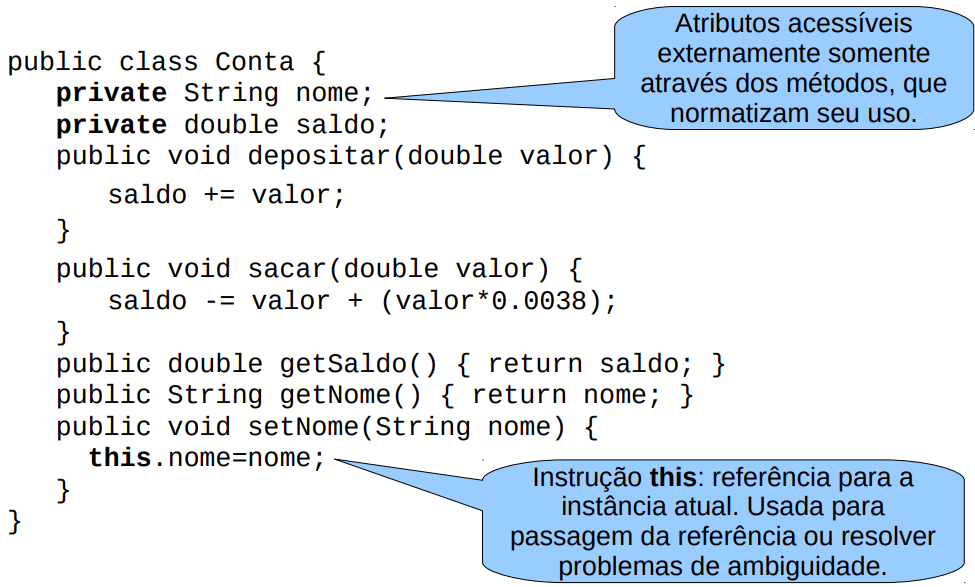
**Encapsulamento**

* Tornar inacessíveis aspectos internos do objeto;
* Diminuição do nível de acoplamento;
* Conceito de caixa-preta: só conheço a interface do objeto, não detalhes de implementação;
* Para encapsular o objeto, devo deixar como públicos estritamente os métodos que devem ser acessados por outros códigos;
* Uso dos modificadores private e protected para atributos e métodos escondidos na classe.

****

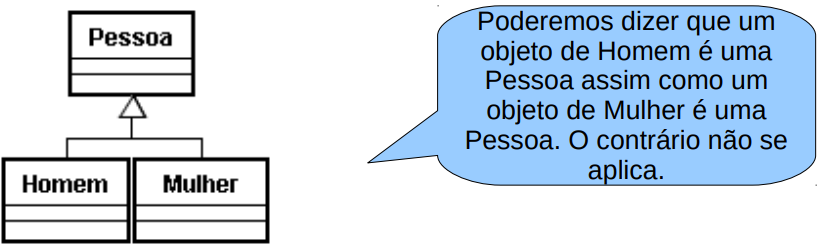
**Encapsulamento**

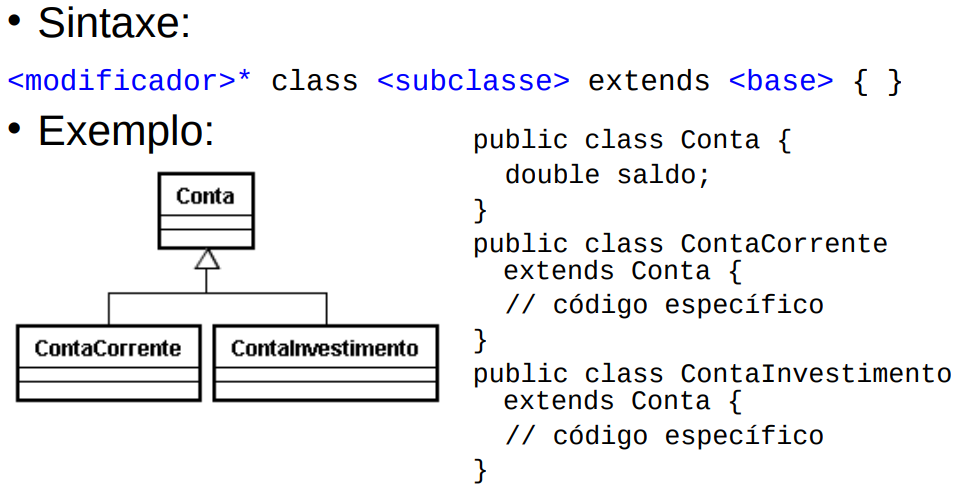
* Tornar inacessíveis aspectos internos do objeto;
* Diminuição do nível de acoplamento;
* Conceito de caixa-preta: só conheço a interface do objeto, não detalhes de implementação;
* Para encapsular o objeto, devo deixar como públicos estritamente os métodos que devem ser acessados por outros códigos;
* Uso dos modificadores private e protected para atributos e métodos escondidos na classe.

****

**Herança**

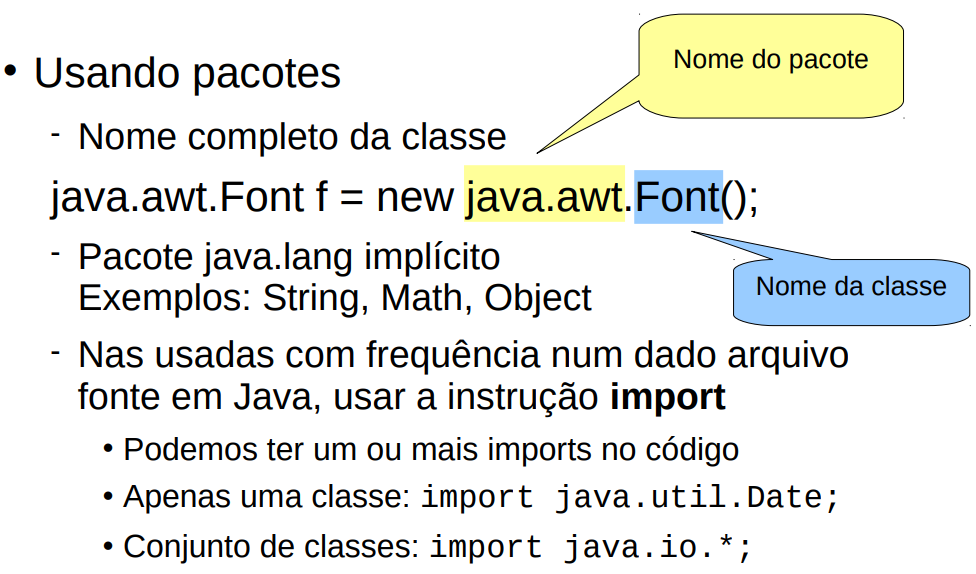
* Consiste em construir sub classes, ou seja, criar uma nova classe herdando as características e comportamento de uma já existente;
* A subclasse também é chamada de filha, derivada ou especialização;
* A classe da qual a subclasse foi criada é chamada de pai, classe base ou superclasse;
* Em Java, todas as classes herdam de Object;
* Uma subclasse é criada com o intuito de introduzir especializações nas características e comportamentos herdados da superclasse;
* É um relacionamento forte entre classes

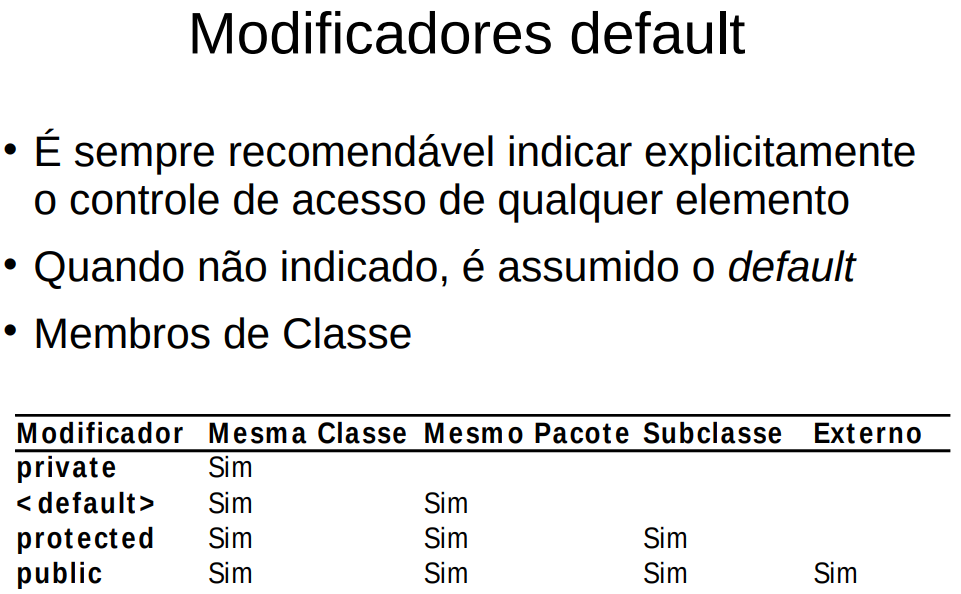
****

****

**Pacote**

* Modo de agrupar classes e interfaces afins;
* Permitem um acesso seletivo no código usuário a somente ao conjunto de classes de interesse;
* Eliminam ambiguidades: conflito de nomes;
* Permite organizar as classes em “pastas”;
* Permite conferir uma identidade para um conjunto de classes (organização, pessoa, etc.);
* Um pacote pode ter classes e interfaces, ou outros pacotes, formando uma estrutura hierárquica (java.io, java.net, java.util, etc.).

****

****

**Observações:**

**Características do Java:**

* Retro compatibilidade através da máquina virtual (Java Vitrual Machine - JVM).
  + Carrega, verifica e executa Bytecode java numa plataforma (hardware + S.O.);
  + Existem JVMs para várias plataformas.
* Garbage collection:
  + Encarrega-se de liberar automaticamente a memória alocada pelo programa que não é mais necessária.
* Como funciona a compilação e Execução do Java?
  + . O compilador Java recebe um código fonte Java e gera um arquivo de códigos binários (Bytecode) executáveis pela JVM.
  + . Transforma os Bytecodes em código de máquina da plataforma e o executa.
  + .class = arquivo de Bytecode
  + .jar = conjunto de Bytecodes
* O java é case sensitive.
* \*Javac é o compilador primário da linguagem Java, incluído no Java Development Kit da Oracle Corporation.
* Todo objeto tem um tipo.
* Toda classe sofre herança de object.
* Toda classe concreta não tem modificador abstract. Posso instanciar objetos a partir dela.
* final - classe - a classe não pode ser herdada.
* final - variável - uma vez atribuído um valor, ele não poderá mais ser alterado.
* Método de instância é executado sobre um objeto.
* Tipos Primitivos x Tipos Referência
* Primitivos: são aqueles que são declarados e imediatamente já alocam o espaço do valor que está contido.
  + short, byte, int, long, double e float;
  + char;
  + boolean.
* . Referência: Servem para qualquer objeto. A classe gera um objeto. 'Endereço' onde se encontra o objeto.
* classe.referência.objeto
* Classe = conjunto de variáveis e funções
* Objeto = algo perceptível, conhecido, pensado ou representado, por qualquer um dos nossos sentidos, em caráter 'estável'.