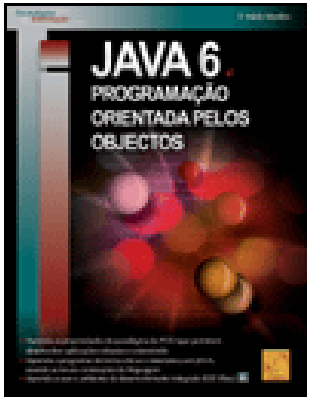
Programação Orientada a Objectos - P. Prata, P. Fazendeiro

"JAVA6 e Programação Orientada Pelos Objectos",
F. Mário Martins, FCA, Julho de 2009.



"Thinking in Java", 4 th Edition, Bruce Eckel.

Objectos:

Um objecto corresponde à representação de uma entidade sob a forma de:

- um identificador único
- um conjunto de atributos privados (estado do objecto)

- um conjunto de operações

(únicas a aceder ao estado do objecto, constituindo o comportamento do objecto)

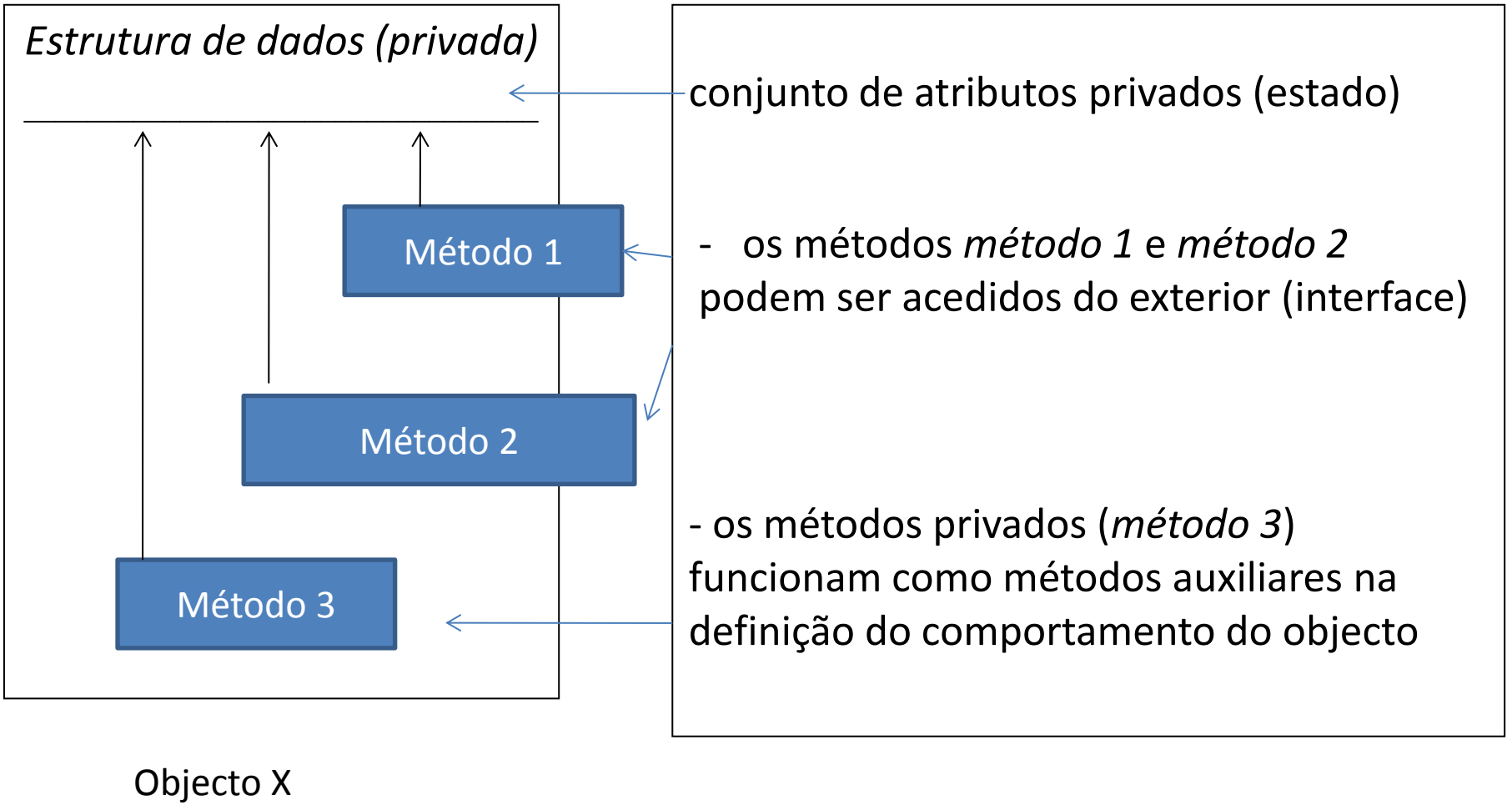
Desse conjunto de operações,

- umas são invocáveis a partir do exterior (operações públicas)
- outras são apenas acessíveis a partir de outras internas ao objecto (operações privadas)

- O conjunto de operações públicas do objecto define o conjunto de serviços que o objecto é capaz de realizar e constitui a interface do objecto também designada por API (Application Programmer's Interface) do objecto

- Aos identificadores que guardam os valores dos atributos chamam-se variáveis (de instância)

- Às operações que representam o comportamento do objecto, chamam-se métodos (de instância)


Estrutura de dados (privada)
Método 1
Método 2
Método 3
conjunto de atributos privados (estado)
- os métodos *método 1* e *método 2* podem ser acedidos do exterior (interface)
- os métodos privados (*método 3*) funcionam como métodos auxiliares na definição do comportamento do objecto
Objecto X

Mensagens:

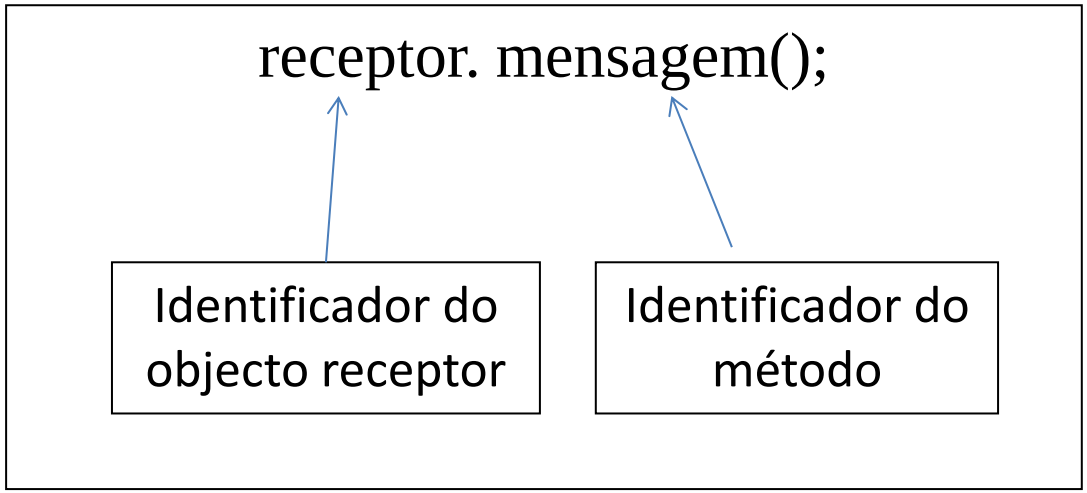
Os objectos vão interactuar entre si através de um mecanismo de envio de mensagens.

Quando um objecto pretende invocar um método de um outro objecto, envia uma mensagem para que tal método seja executado.

Em cada computação, isto é, em cada alteração do estado do programa existe um objecto que é emissor de uma mensagem e um outro que é o receptor dessa mensagem.

Em geral a sintaxe para o envio de uma mensagem a um objecto tem uma das seguintes formas:

a)



- envio de uma mensagem sem argumentos a um objecto, sem retorno de resultado pelo método correspondente

b) receptor. mensagem(arg1, arg2, … arg*n*);

- envio de uma mensagem com argumentos, sem retorno de resultado

c) resultado = receptor. mensagem();

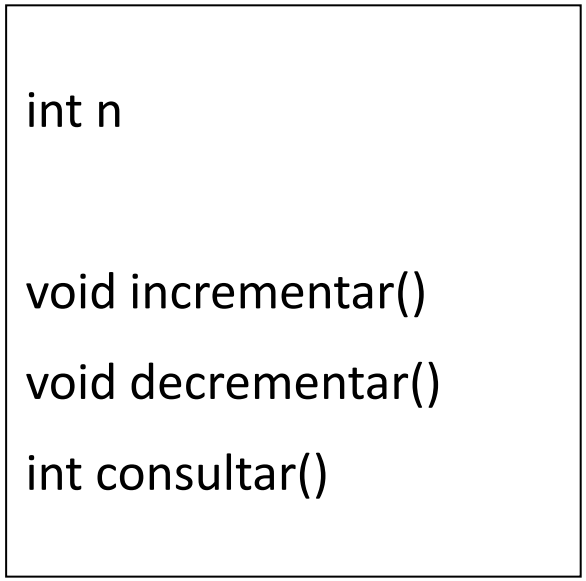
- envio de uma mensagem sem argumentos, com retorno de resultado

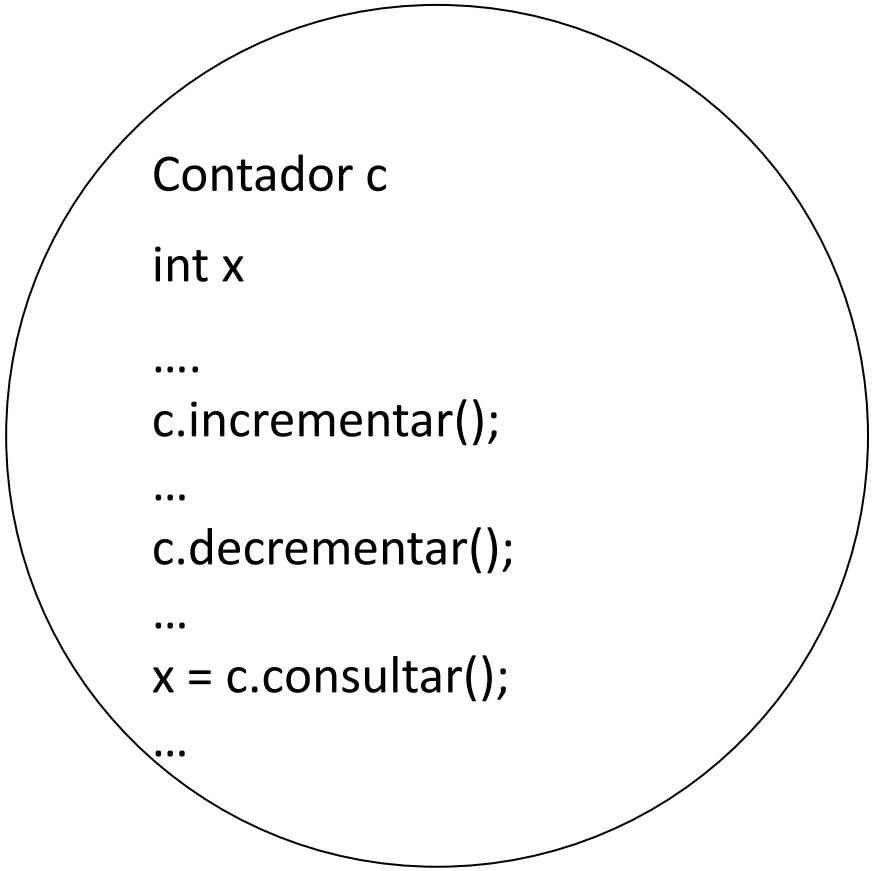
d) resultado = receptor. mensagem(arg1, arg2, … arg*n*);

- envio de uma mensagem com argumentos e com retorno de resultado

Exemplo:

Objecto do tipo contador





-Em linguagens com verificação de tipos (por ex.lo, C++, Java) parâmetros e resultados têm um dado tipo.

- Os argumentos e o resultado de uma mensagem têm que ser compatíveis com os tipos dos parâmetros e do valor devolvido pelo método correspondente.

Instâncias Vs. Classes:

O mecanismo de classificação permite, na generalidade das linguagens, criar vários objectos do mesmo tipo, isto é, objectos que possuam exactamente a mesma estrutura e o mesmo comportamento.

"Uma classe é um modelo, a partir do qual podem ser criados objectos. Contém a definição dos atributos e dos métodos do objecto"

Uma classe é um objecto "especial" que serve para:

- conter a descrição da estrutura e do comportamento de objectos do mesmo tipo,

- criar objectos particulares possuindo tal estrutura e comportamento.

Definida uma classe, os seus objectos são criados através de um mecanismo de instanciação, pelo qual o objecto é inicializado, passando a existir e podendo receber mensagens de outros objectos.

Cada objecto vai ter as suas próprias instâncias das variáveis de estado, enquanto que o código que implementa os métodos permanece armazenado na classe.

Apesar de todas as instâncias da classe exibirem um comportamento comum, uma vez que partilham as mesmas operações, elas não são iguais. Cada objecto possui o seu próprio estado que pode variar ao longo do tempo.

Quando um objecto é criado (instanciado) a inicialização do seu estado é geralmente feita por invocação automática de um método de inicialização (o construtor da classe).

Se, num dado programa, necessitarmos de vários objectos do tipo Contador, definimos a classe Contador e a partir dela criamos os objectos desse tipo, isto é criamos instâncias da classe Contador:

Em Java:

```
Contador   conta_1 = new  Contador();
Contador   conta_2 = new Contador();
```

Os objectos conta_1 e conta_2 possuem ambos uma variável de instância, n, que apenas poderá conter valores inteiros. Estes dois objectos, tal como qualquer outra instância de Contador, serão capazes de responder às mensagens incrementar(), decrementar() e consultar():

(com x e y variáveis inteiras)

```
conta_1.incrementar();
conta_2.decrementar();
x = conta_1.consultar();
y = conta_2.consultar();
```

- supondo que cada contador é inicializado com o valor 0, qual será o valor de x e y?

Numa linguagem de programação orientada a objectos "pura" todas as entidades deveriam ser objectos.
A linguagem Smalltalk é a que mais se aproxima desse modelo (classes, instâncias e mensagens são objectos).

Em Java existem tipos de dados que não são objectos (tipos primitivos e arrays).

Existem linguagens que permitem programar com classes e objectos, mas que não possuem um mecanismo de Herança *(que estudaremos mais tarde)*.

Estas linguagens não são consideradas orientadas a objectos.

Em resumo:

O modelo mais simples de linguagem Orientada a Objectos é:

"Uma linguagem é Orientada a Objectos, se suporta Objectos, se esses objectos pertencem a Classes e se hierarquias de classes podem ser definidas, incrementalmente, por um mecanismo de Herança."

O Objecto constitui a unidade fundamental de encapsulamento, enquanto Classes e Herança realizam os princípios de partilha de comportamento e evolução.