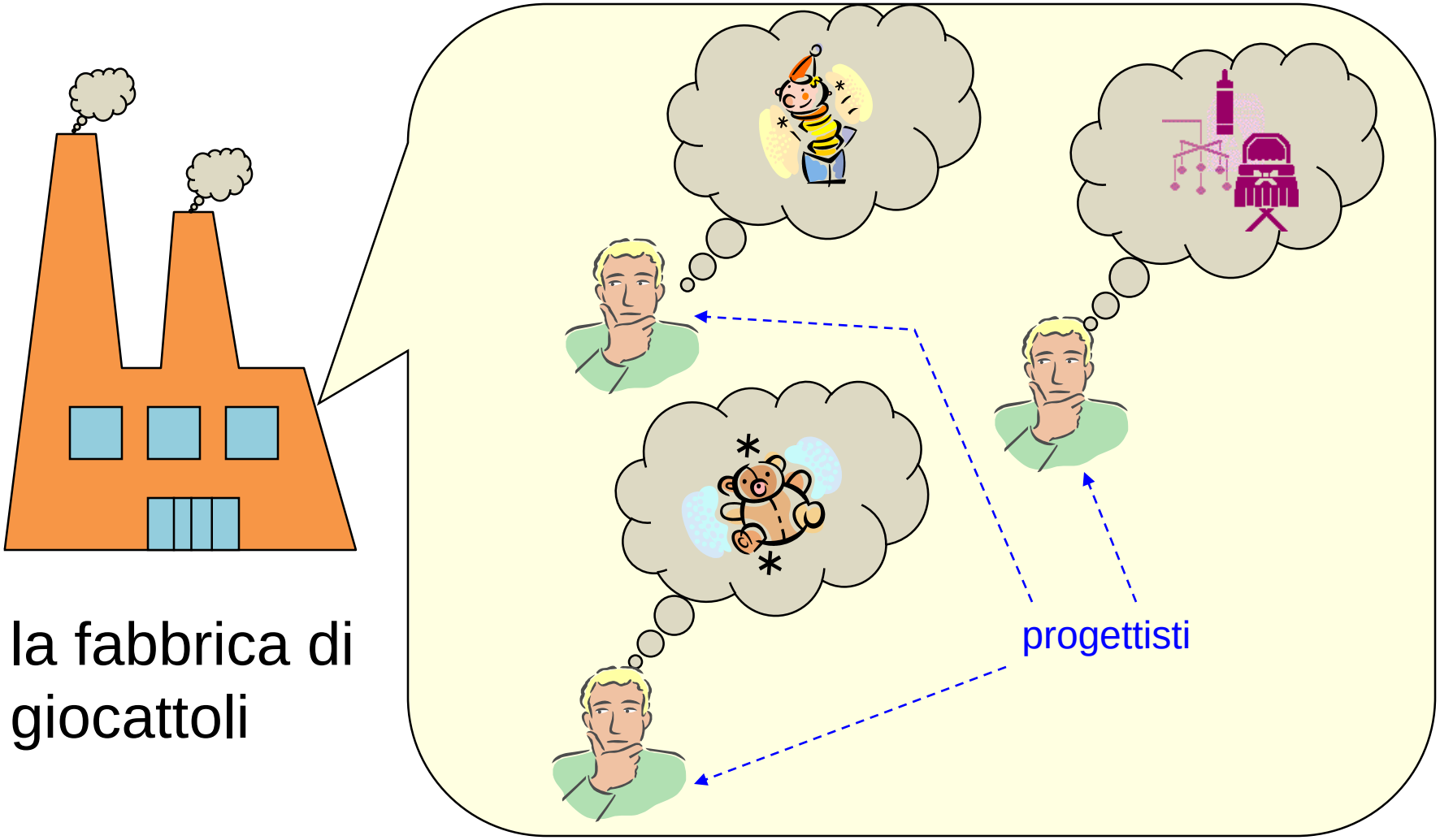
Programmazione Orientata agli Oggetti

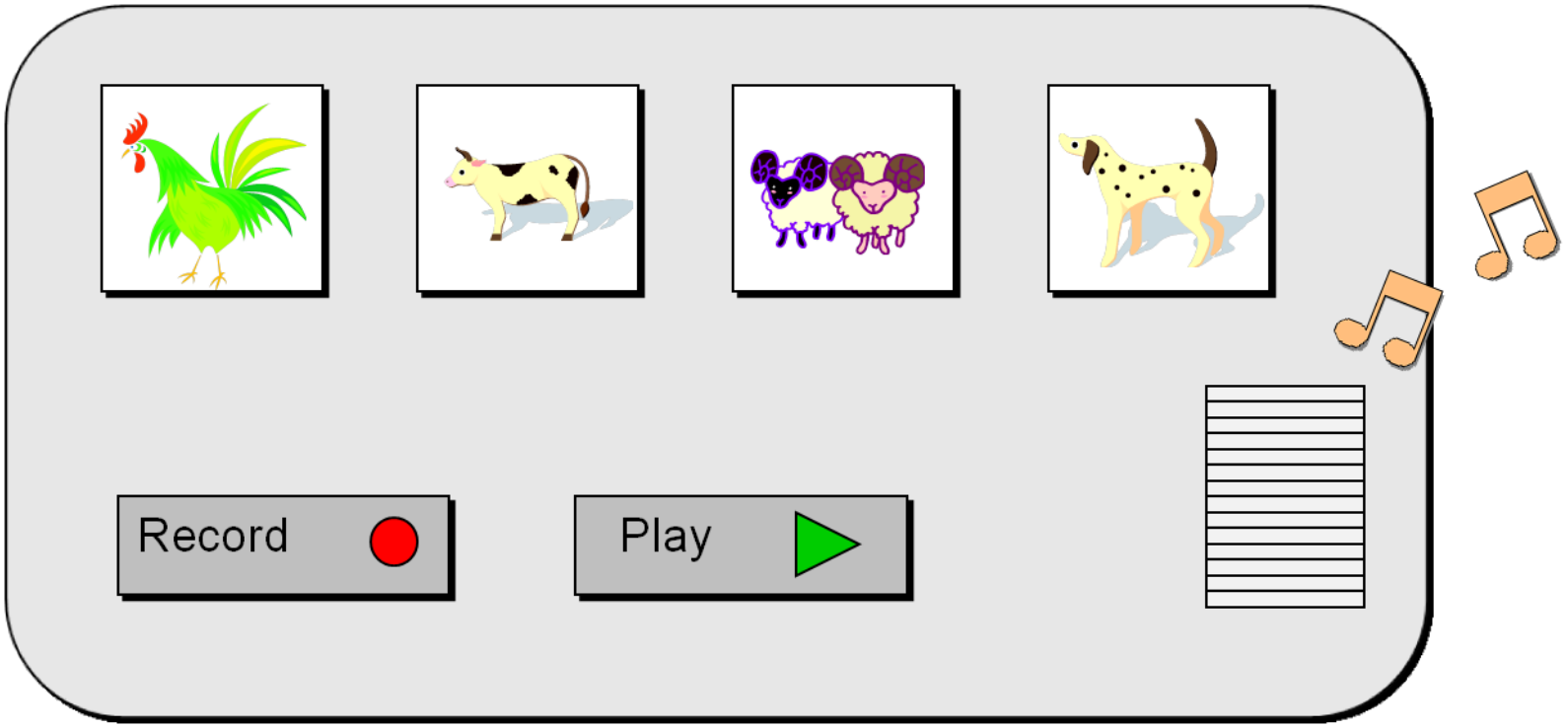
Emilio Di Giacomo e Walter Didimo

Una metafora dal mondo reale



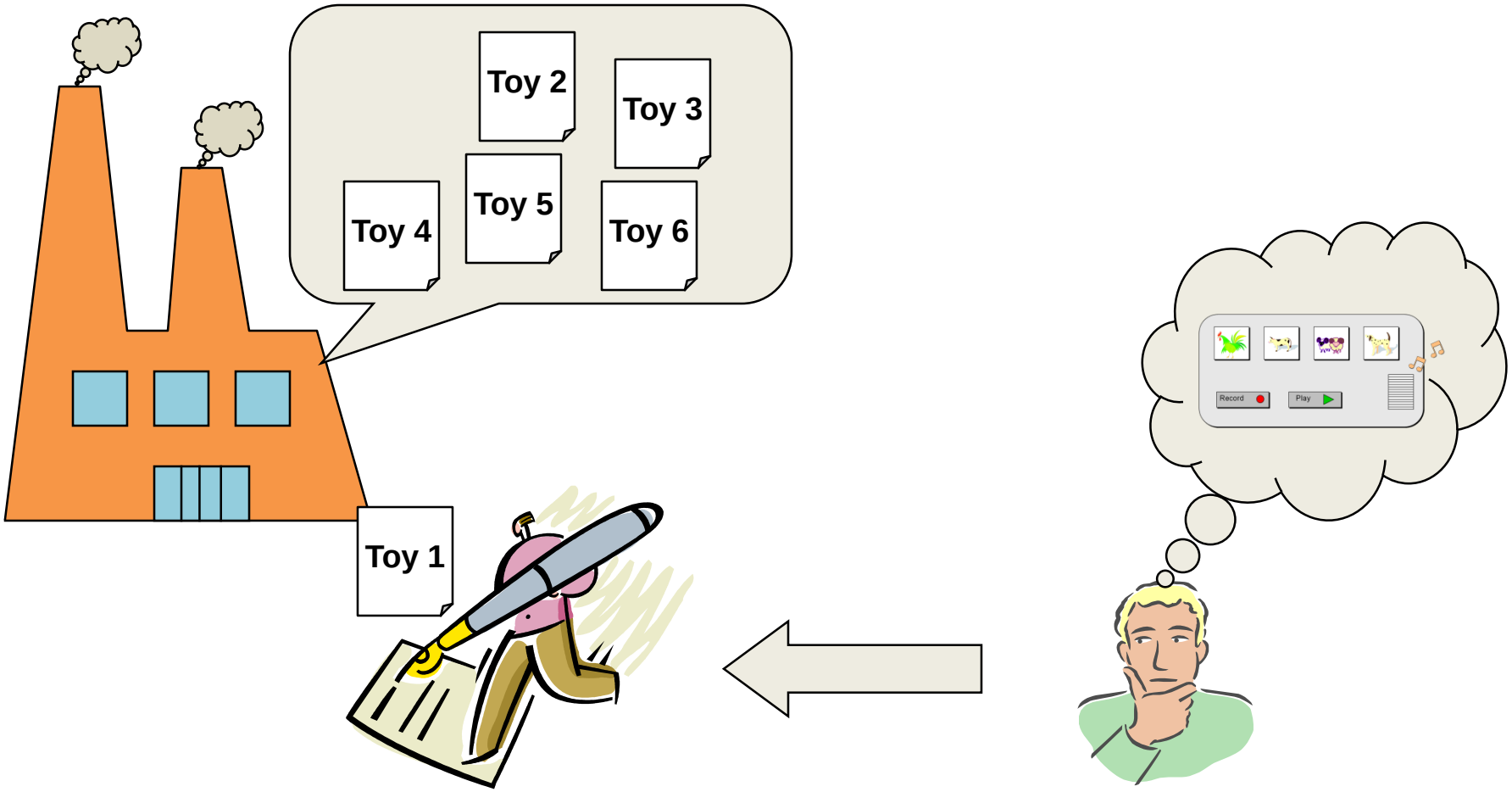
la fabbrica di giocattoli

Un semplice giocattolo

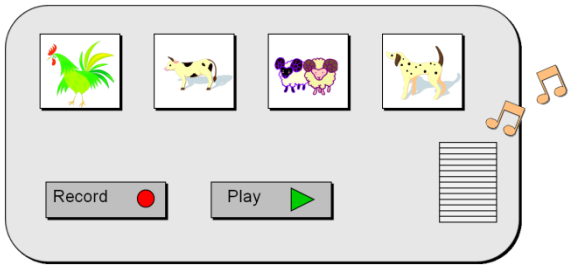


Impara i suoni

Dall’idea al progetto



Il progetto del giocattolo



ImparaISuoni

nome

- ROM con il verso del gallo
- ROM con il verso della mucca

…

- RAM per registrare suoni
- microfono
- altoparlante

…

attributi

- **riproduciVersoDelGallo** {
  - leggi la ROM con il verso del gallo
  - riproduci suono tramite altoparlante

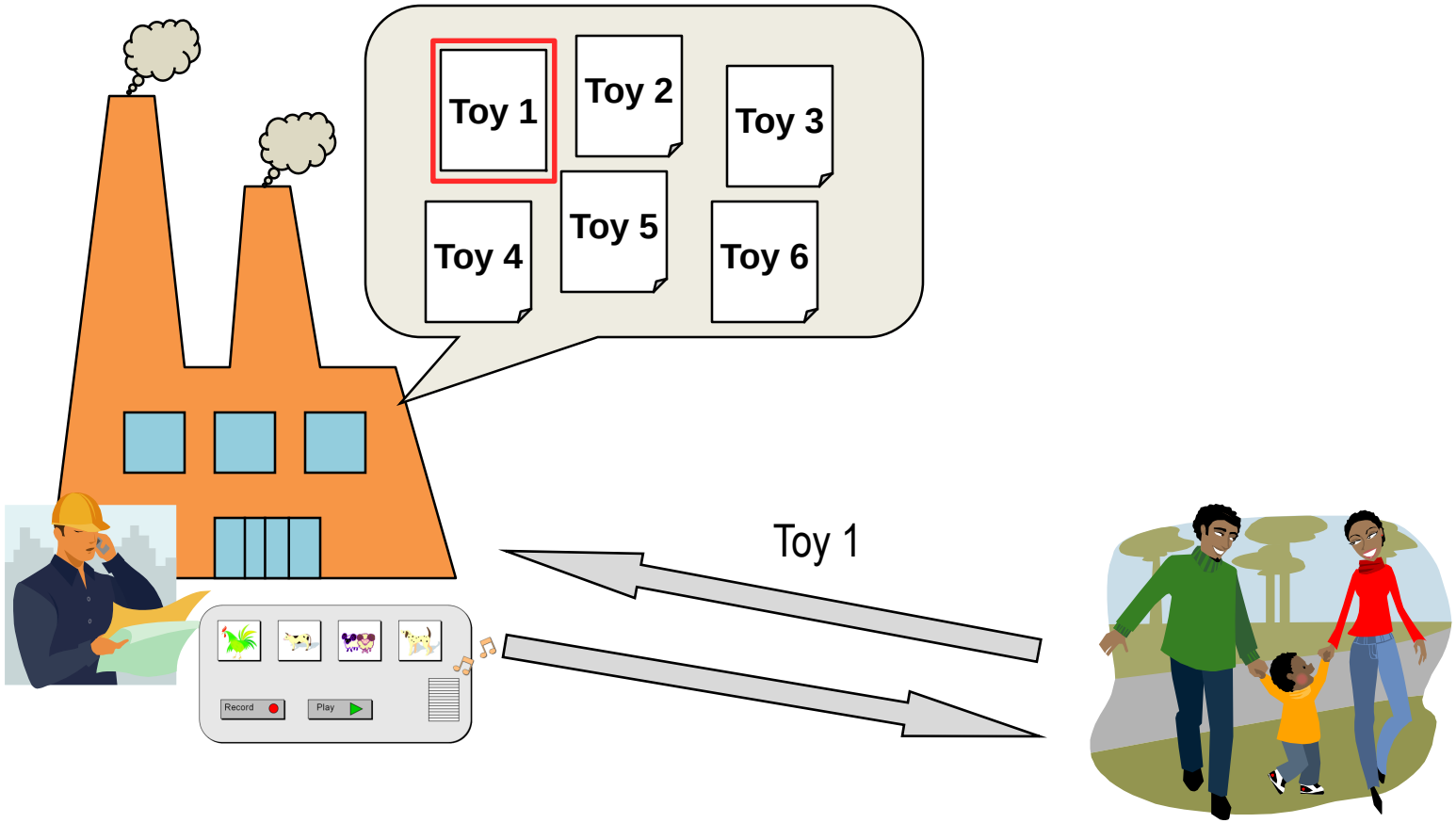
  }
- **registraSuono** {
  - attiva microfono
  - acquisisci e memorizza suono in RAM

  }

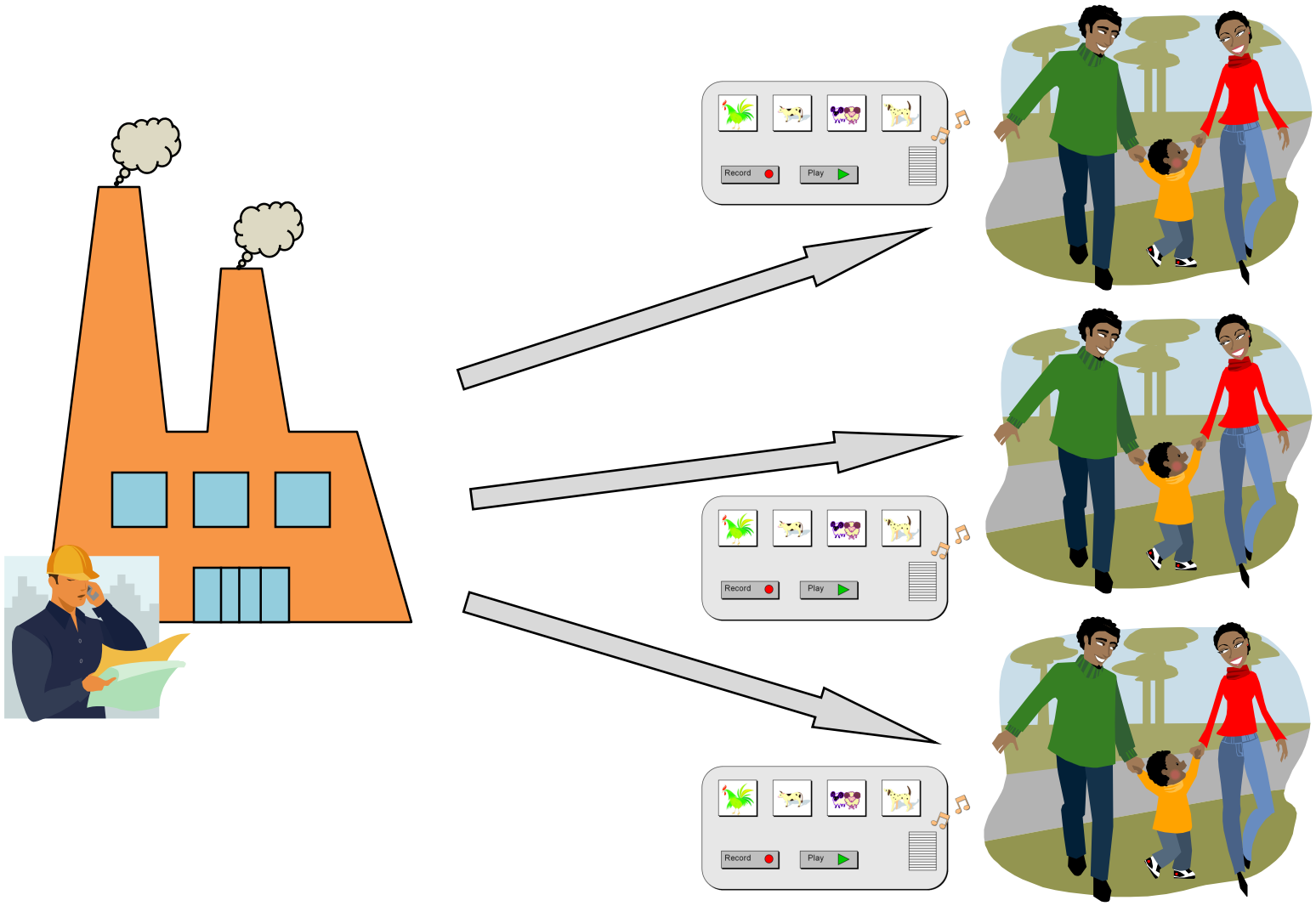
…

comportamento

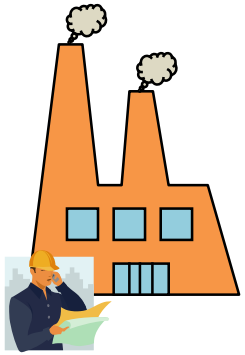
Richiesta e creazione di giocattoli



Indipendenza dei giocattoli (stato)

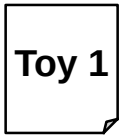


Il parallelo con la programmazione



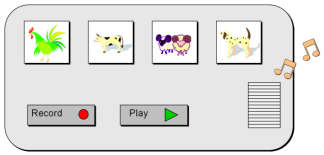
calcolatore con ambiente di esecuzione

programmatore



classe

parti di un programma



oggetto

Classi e oggetti: primo sguardo

- Un programma è la definizione di un certo numero di classi
- Ogni classe ha un *nome* e descrive le caratteristiche di una tipologia di oggetti, cioè:
  - *come* gli oggetti sono fatti: attributi (o proprietà o campi)
  - *cosa* gli oggetti sanno fare (quali servizi offrono a chi li vuole usare): metodi (l'insieme dei metodi di un oggetto costituisce il suo comportamento)
- Un oggetto è sempre l'istanza di un classe

Stato di un oggetto

- Due oggetti (istanze) di una stessa classe sono:
  - *simili*, perché hanno gli stessi attributi e lo stesso comportamento;
  - *diversi*, perché hanno cicli di vita indipendenti, cioè si possono usare in modo diverso e i loro attributi assumono in generale valori diversi nel tempo
- Lo stato di un oggetto è l'insieme dei valori assunti dai suoi attributi in un determinato istante di tempo (fotografia istantanea)

Struttura di una classe

nel corso useremo una rappresentazione grafica delle classi simile a quella definita da UML (Unified Modeling Language) - standard introdotto nel 1996 per la documentazione del software

Nome di una classe: regole

- I programmi Java sono scritti utilizzando un alfabeto di caratteri chiamato Unicode
  - *caratteri speciali:* simboli di punteggiatura, spazi bianchi, operatori aritmetici, parentesi, …
  - ai caratteri speciali viene riservata una particolare interpretazione;
- Il nome di una classe non può contenere caratteri speciali
- Il primo carattere del nome di una classe non può essere una cifra {0, 1, 2, …, 9}

Nome di una classe: convenzioni

- Oltre alle regole Java, esistono convenzioni diffuse per definire il nome di una classe:
  - se il nome è composto da una sola parola, il primo carattere è maiuscolo e gli altri sono in minuscolo
  - se il nome è composto da più parole, il primo carattere di ogni parola è maiuscolo e gli altri sono in minuscolo; le parole sono concatenate tra loro

Esempi

NOMI VALIDI		NOMI NON VALIDI
CONVENZIONE RISP.	CONVENZIONE NON RISP.	
`Rettangolo`	`rettangolo`	`Rettang.`
`AutoDaNoleggio`	`AutodaNoleggio`	`Auto Da Noleggio`
`Classe1`	`CLASSE1`	`1Classe`

Attributi

- Un attributo è una variabile:
  - come tutte le variabile (che abbiamo già visto in C), l'attributo può assumere un solo valore alla volta, ma tale valore può variare molte volte nel tempo, cioè durante il ciclo di vita di un oggetto

- Come ogni variabile, un attributo è descritto da:
  - il tipo di valori che può assumere, detto tipo (o dominio) dell'attributo
  - un nome che lo identifica

Tipi di attributi

- Esistono due principali categorie di tipi di attributi in Java:
  - primitivi: numeri interi, numeri reali, caratteri e un tipo chiamato *boolean*, con due soli possibili valori {*true, false*}
  - riferimento: per ogni classe di nome *C* esiste un tipo di attributo di nome *C*; un attributo di tipo *C* può memorizzare solo riferimenti ad oggetti di tipo *C*

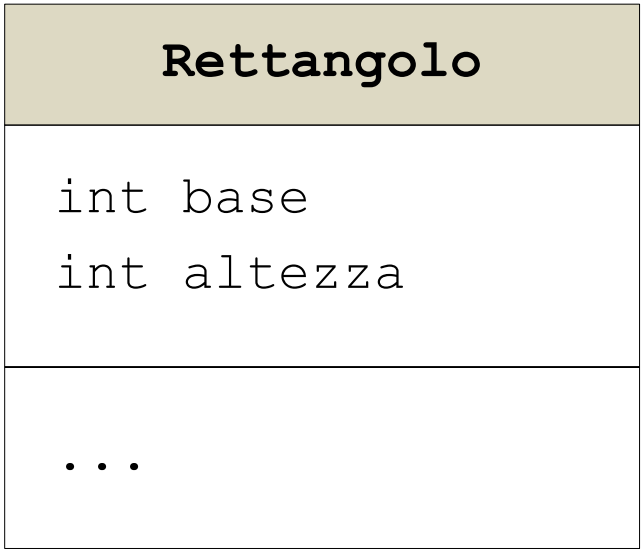
Tipi di attributi: esempi

- Ad esempio in Java:
  - il tipo primitivo *int* indica numeri interi nell'intervallo $[-2^{31}, 2^{31}-1]$ (codificabili con 32 bit); un attributo di tipo *int* può assumere (solo) valori in questo intervallo
  - Il tipo primitivo *double* indica numeri reali in virgola mobile codificati con 64 bit
  - la classe *String* modella qualunque stringa (sequenza finita di caratteri); un attributo di tipo *String* può memorizzare (solo) riferimenti ad oggetti della classe *String*
- Una trattazione sistematica dei tipi di attributo, con riferimento a Java, sarà svolta più avanti nel corso

Nome di un attributo

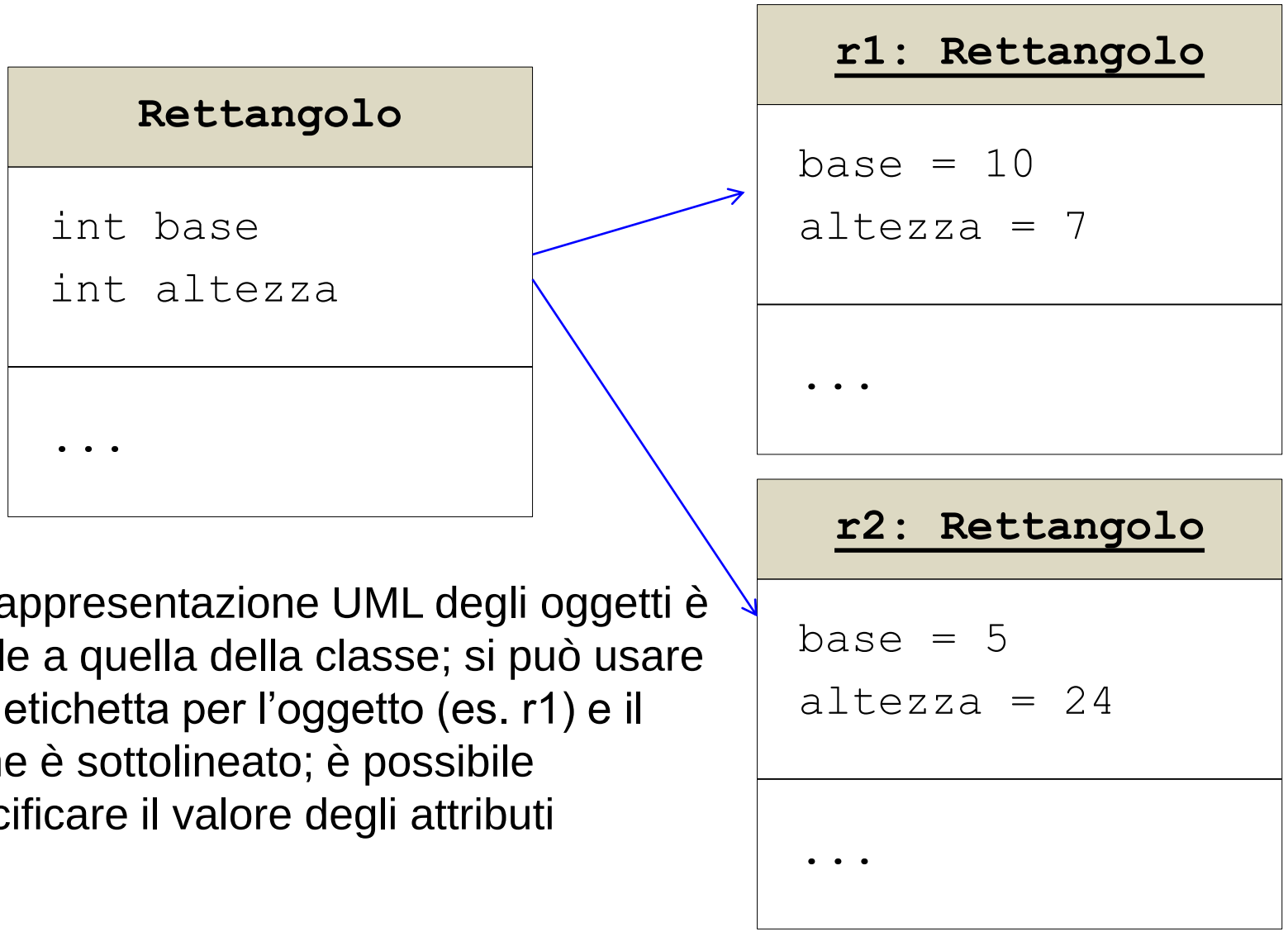
- Il nome di un attributo può essere definito con le stesse regole del nome di una classe
- anche le convenzioni sono analoghe, ma il primo carattere è in minuscolo anziché in maiuscolo

Esempio di classe e attributi



Esempio di classe che modella rettangoli nel piano; ogni oggetto *Rettangolo* ha due attributi di tipo *int*, che memorizzano i valori della base e dell'altezza

Esempio di istanze



La rappresentazione UML degli oggetti è simile a quella della classe; si può usare una etichetta per l'oggetto (es. r1) e il nome è sottolineato; è possibile specificare il valore degli attributi

Metodi

- Un metodo definisce un servizio che un oggetto può offrire, cioè una procedura (funzione) che l'oggetto può svolgere su richiesta
- Ogni metodo è definito da:
  - una intestazione: specifica il nome del metodo, cosa il metodo sa fare, in che modo si può chiedere ad un oggetto di eseguirlo, ed eventuali restrizioni su chi può effettuare queste richieste.
  - un corpo: descrive la sequenza di istruzioni che l'oggetto deve svolgere per eseguire il metodo; il corpo di un metodo è anche detto implementazione del metodo

Metodi: intestazione

- L'intestazione di un metodo contiene obbligatoriamente le seguenti informazioni:
  - nome del metodo: definito con le stesse regole e convenzioni degli attributi
  - parametri formali: variabili usate per passare dati di input al metodo; similmente agli attributi, ogni parametro ha un tipo e un nome
  - tipo di ritorno: il tipo di valore eventualmente restituito dal metodo al termine della sua esecuzione; se un metodo non restituisce valori, il suo tipo di ritorno si indica con *void*

Metodi: prototipo

- Le informazioni obbligatorie nell'intestazione di un metodo (nome + parametri formali + tipo di ritorno) si chiamano prototipo del metodo
- La sintassi è la seguente:

<tipo ritorno> *<nome metodo>* **(***<elenco parametri formali>***)**

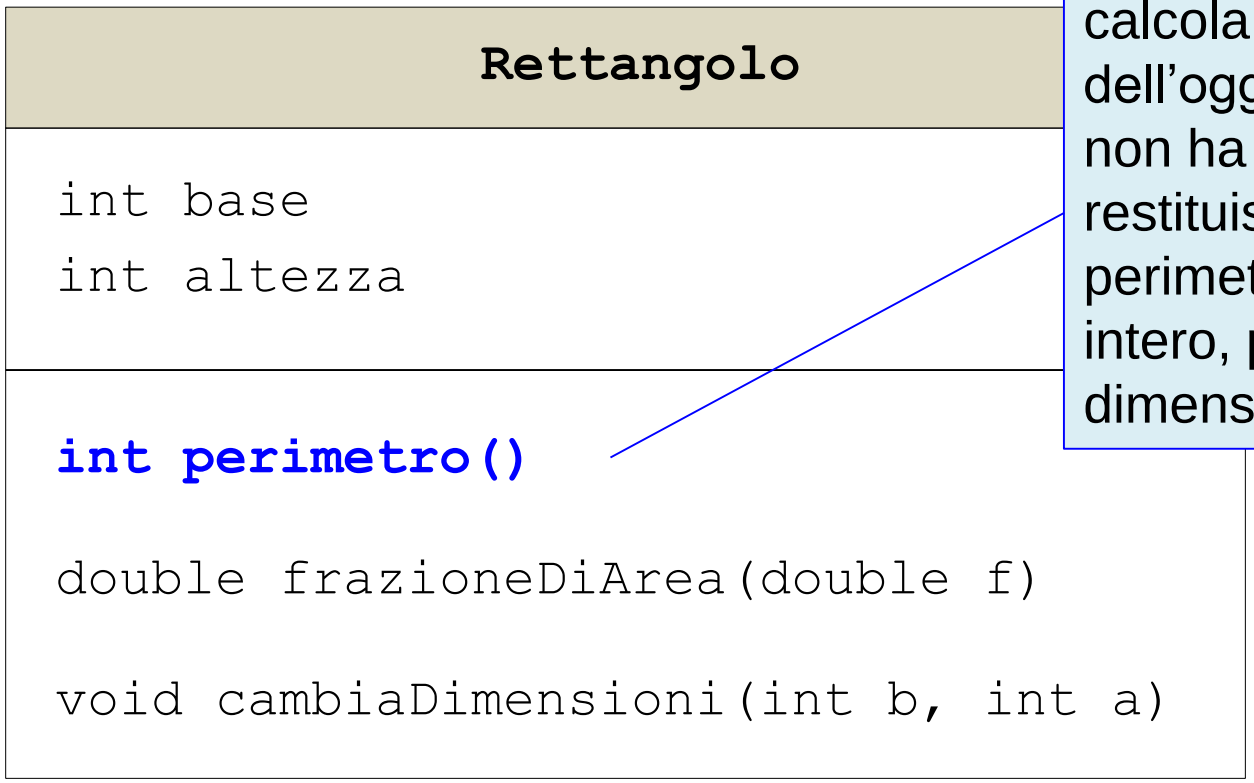
nell'elenco, i parametri formali vanno separati da una virgola; ogni parametro è descritto dal tipo e dal nome (in questo ordine); se il metodo non prevede parametri, l'elenco va lasciato vuoto

Esempio di classe e metodi

Rettangolo
`int base` `int altezza`

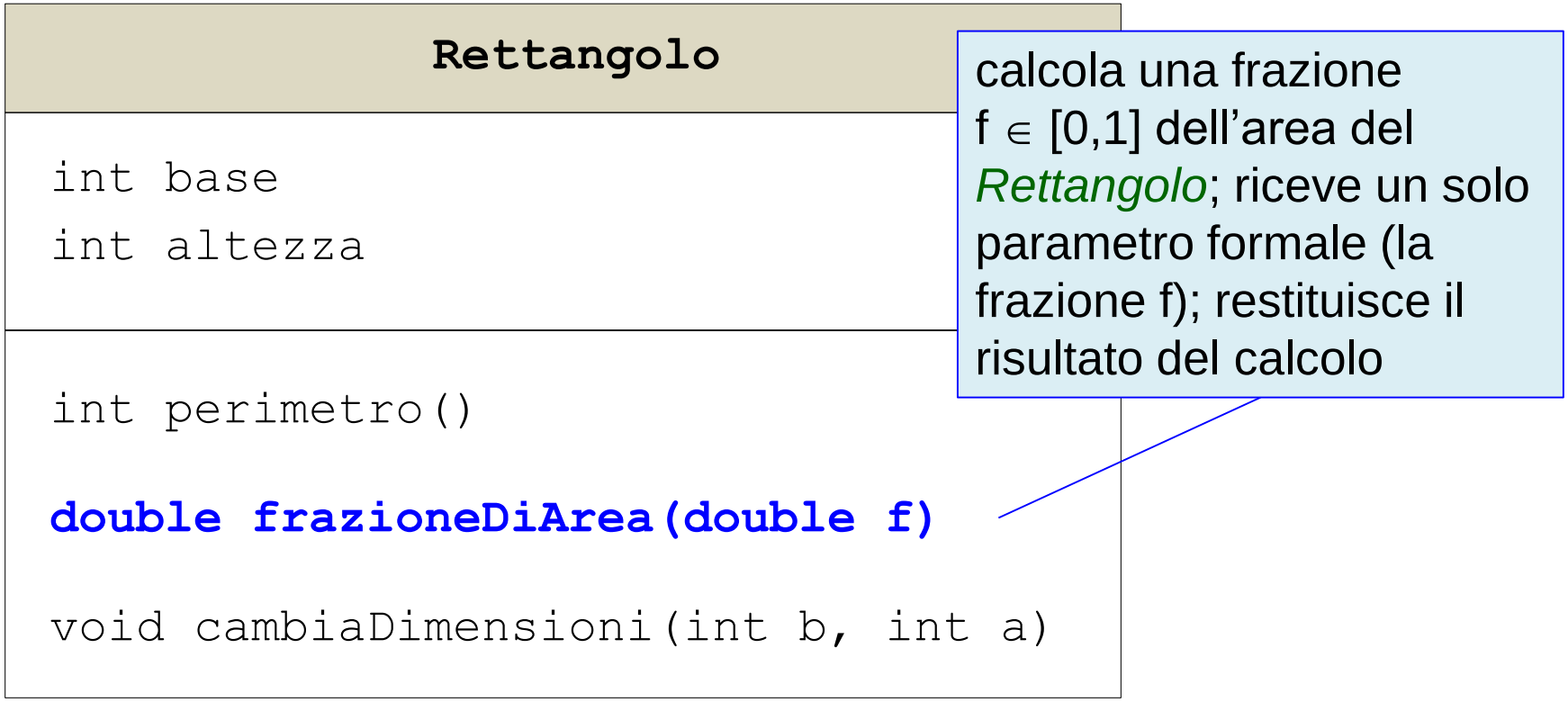
`int perimetro()`
 	
`double frazioneDiArea(double f)`
 	
`void cambiaDimensioni(int b, int a)`

Esempio di classe e metodi

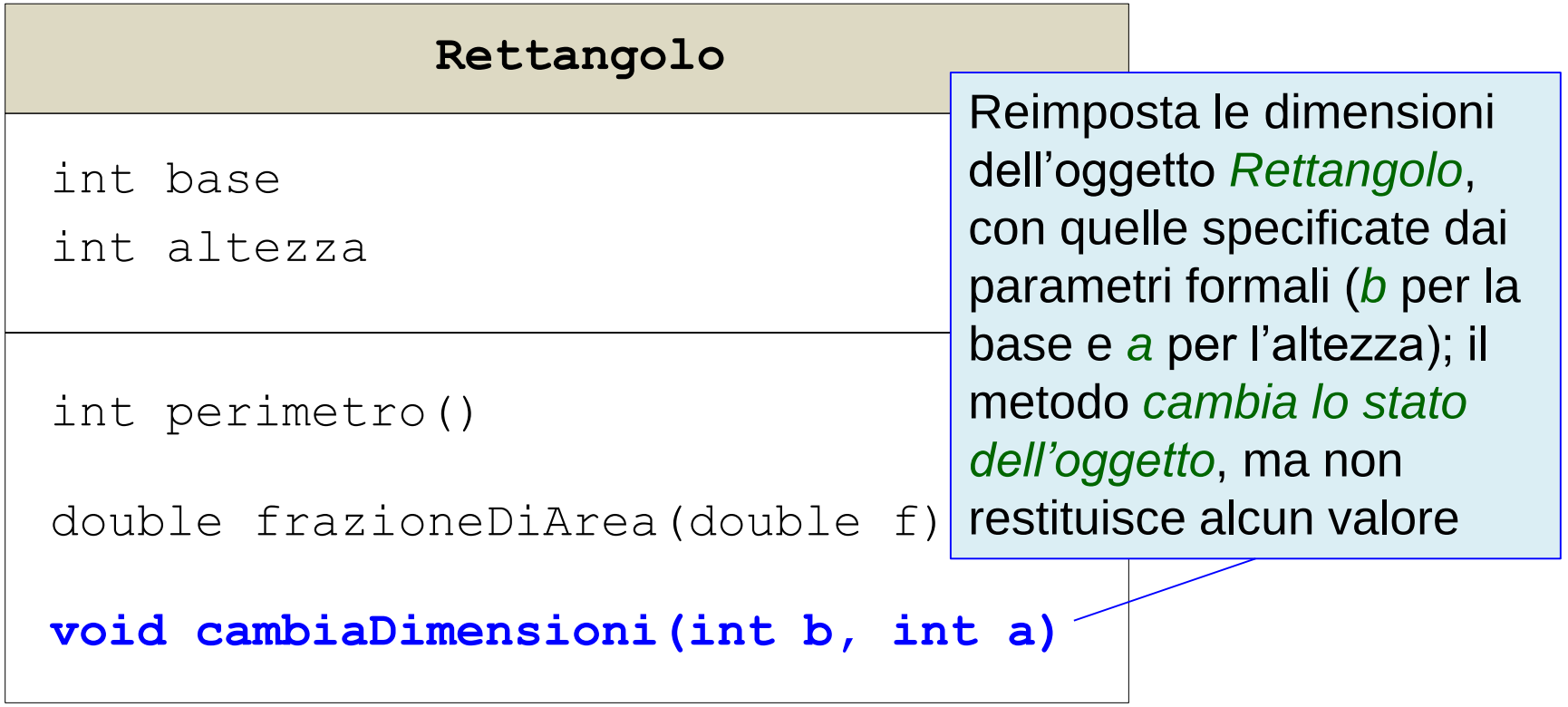


calcola il perimetro dell'oggetto *Rettangolo*; non ha parametri formali e restituisce il valore del perimetro (un valore intero, poiché le dimensioni sono intere)

Esempio di classe e metodi



Esempio di classe e metodi

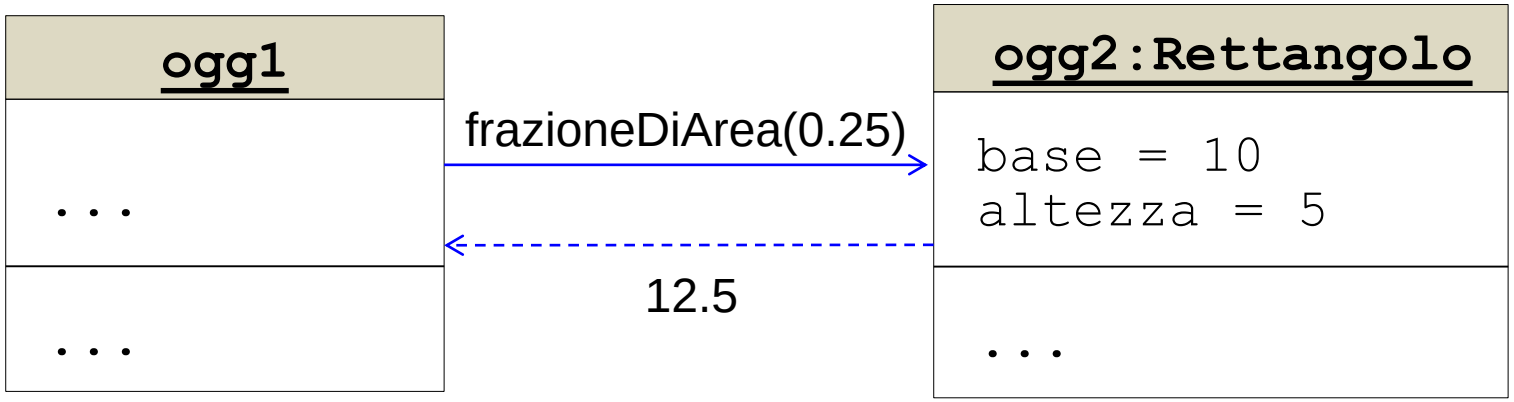


Metodi: invocazione

- Chiedere ad un oggetto di eseguire un suo metodo equivale ad *inviargli un messaggio*; si dice anche che viene invocato un metodo sull'oggetto
  - per invocare un metodo serve solo conoscere la sua intestazione (non la sua implementazione)
  - così come per usare correttamente le funzioni del giocattolo *ImparaISuoni* serve solo sapere cosa esse fanno e quali tasti premere per richiederle (non serve sapere come sono eseguite dal giocattolo)

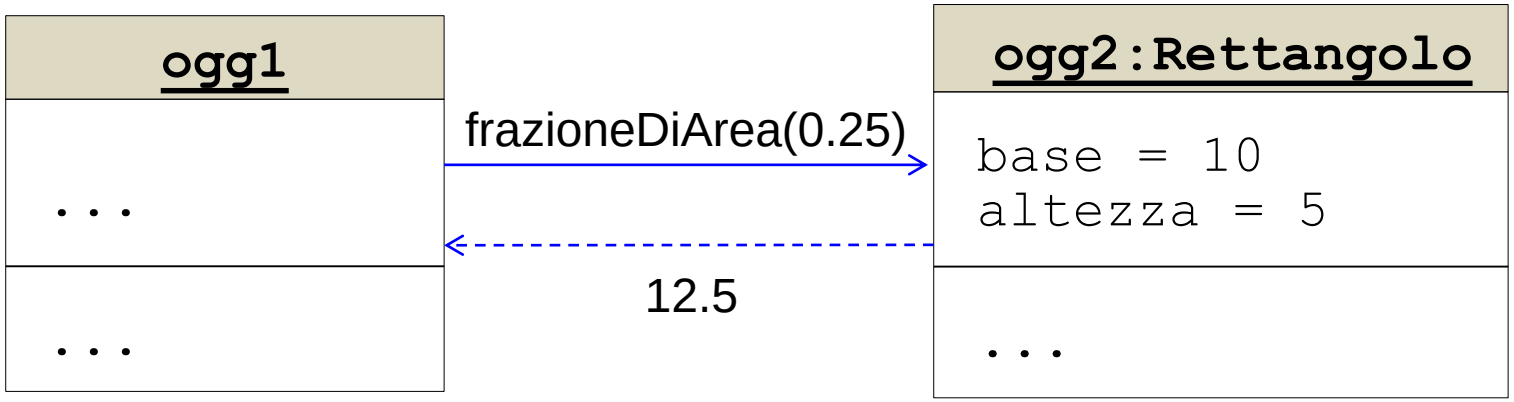
Oggetti chiamanti e riceventi

- Chi invoca i metodi sugli oggetti?
- Nel corpo di un suo metodo, un oggetto *ogg1* può invocare un altro metodo su un oggetto *ogg2*, attraverso una apposita istruzione di invocazione
  - *ogg1* è detto oggetto chiamante
  - *ogg2* è detto oggetto ricevente



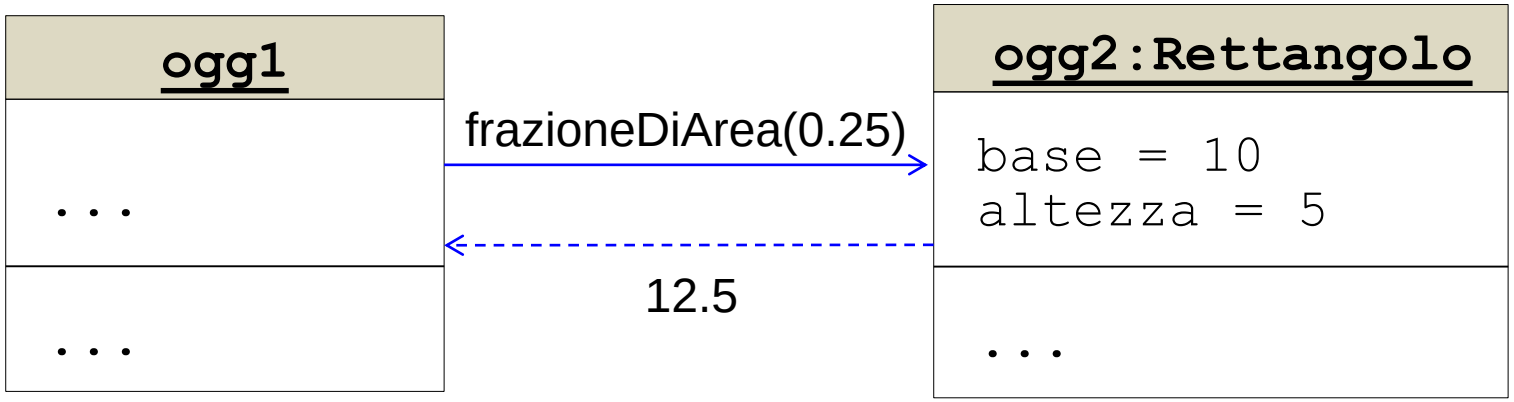
Signature

- Un'istruzione di invocazione di un metodo su un oggetto ricevente deve specificare:
  - il nome del metodo che esso deve eseguire
  - il valore dei parametri formali del metodo
- Il nome del metodo e la lista dei suoi parametri si chiama anche signature (firma) del metodo

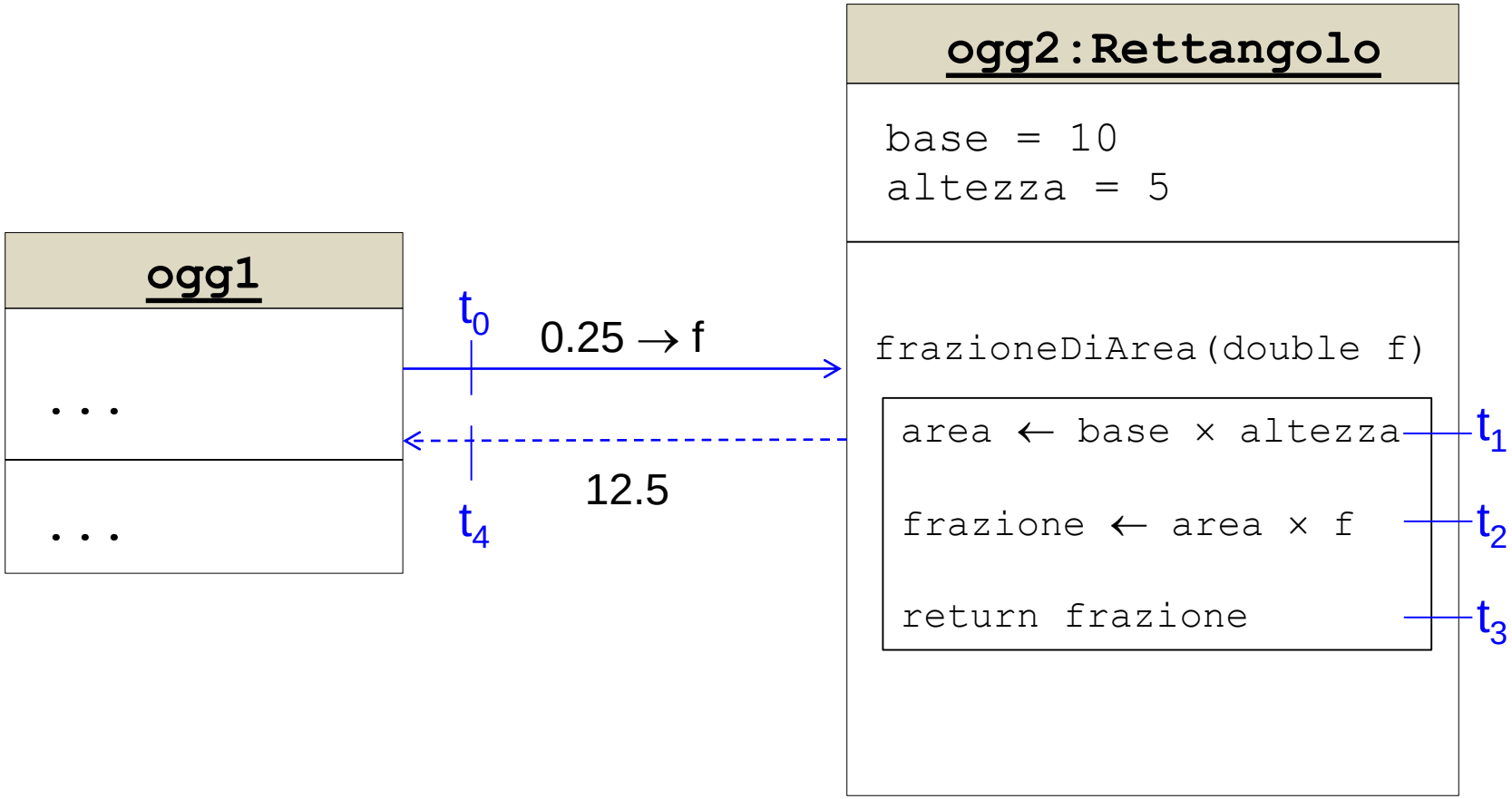


Unicità della signature

- La signature è dunque una parte del prototipo (che esclude il tipo di ritorno)
- In una classe non possono esserci due metodi con la stessa signature, poiché si genererebbe ambiguità all'atto della chiamata



Dettaglio invocazione metodi



All'atto dell'invocazione, il chiamante passa al ricevente i valori effettivi da sostituire ai parametri formali; tali valori si chiamano parametri attuali

Invocazione metodi e espressioni

- L'invocazione di un metodo che ha un tipo di ritorno diverso da *void* costituisce una espressione:
  - il valore dell'espressione è il valore restituito dal metodo al termine della sua esecuzione
  - si dice anche che all'invocazione del metodo rimane associato il valore che il metodo restituisce

Attributi e metodi di classe

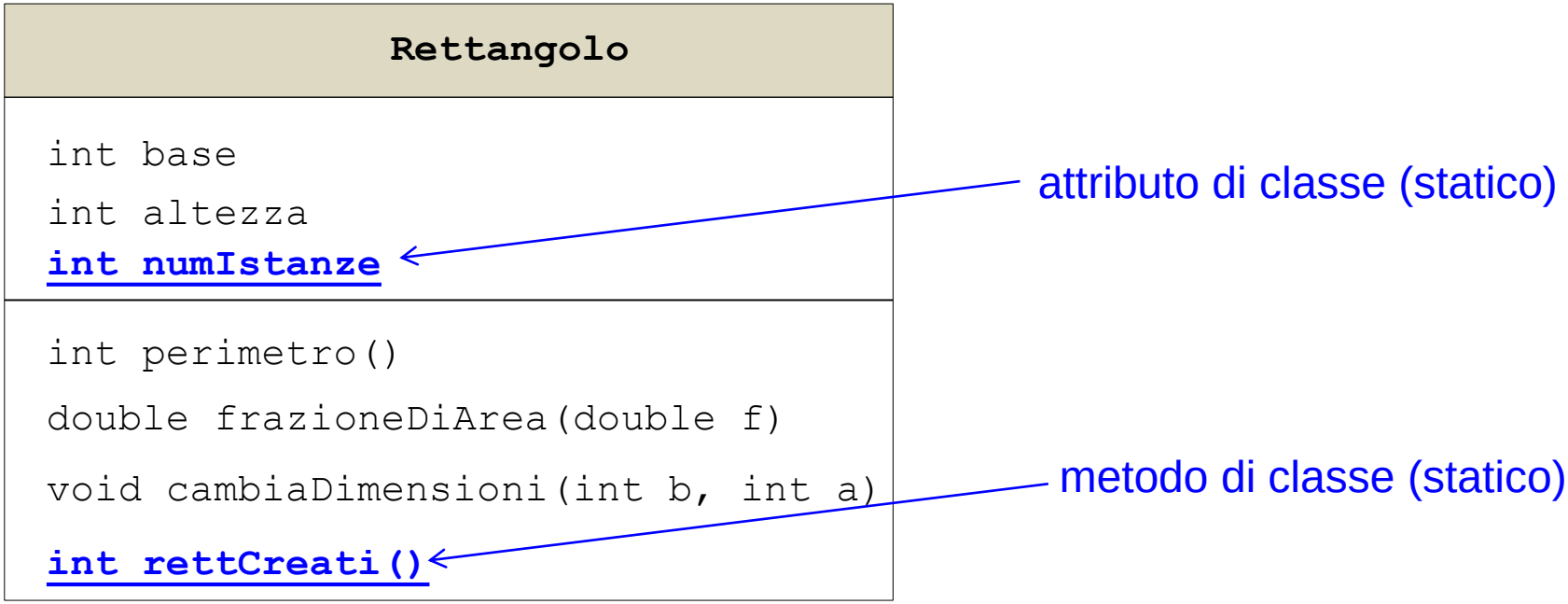
- Gli attributi e i metodi di un oggetto si chiamano anche attributi e metodi di istanza
- In una classe è tuttavia possibile definire anche attributi e metodi di classe (o statici)
  - un attributo di classe è riferito allo stato della classe e non dei suoi oggetti – *esiste unicamente all'interno di una classe*, cioè non viene creata una copia per ogni oggetto!!
  - un metodo di classe viene eseguito dalla classe, e va dunque invocato sulla classe (non su una sua istanza)

Attributi e metodi di classe

- Per distinguere attributi e metodi di classe, ogni linguaggio prevede una parola chiave da specificare all'atto della definizione
  - come vedremo più avanti, Java richiede la parola chiave *static*
- Nella notazione grafica UML, gli attributi e i metodi di classe vengono sottolineati

Esempio di attributi e metodi di classe

- Potremmo ad esempio definire un attributo statico nella classe *Rettangolo*, che memorizzi il numero di istanze via via create durante l'esecuzione di un programma
- Potremmo anche definire un metodo statico che restituisca il valore di tale attributo



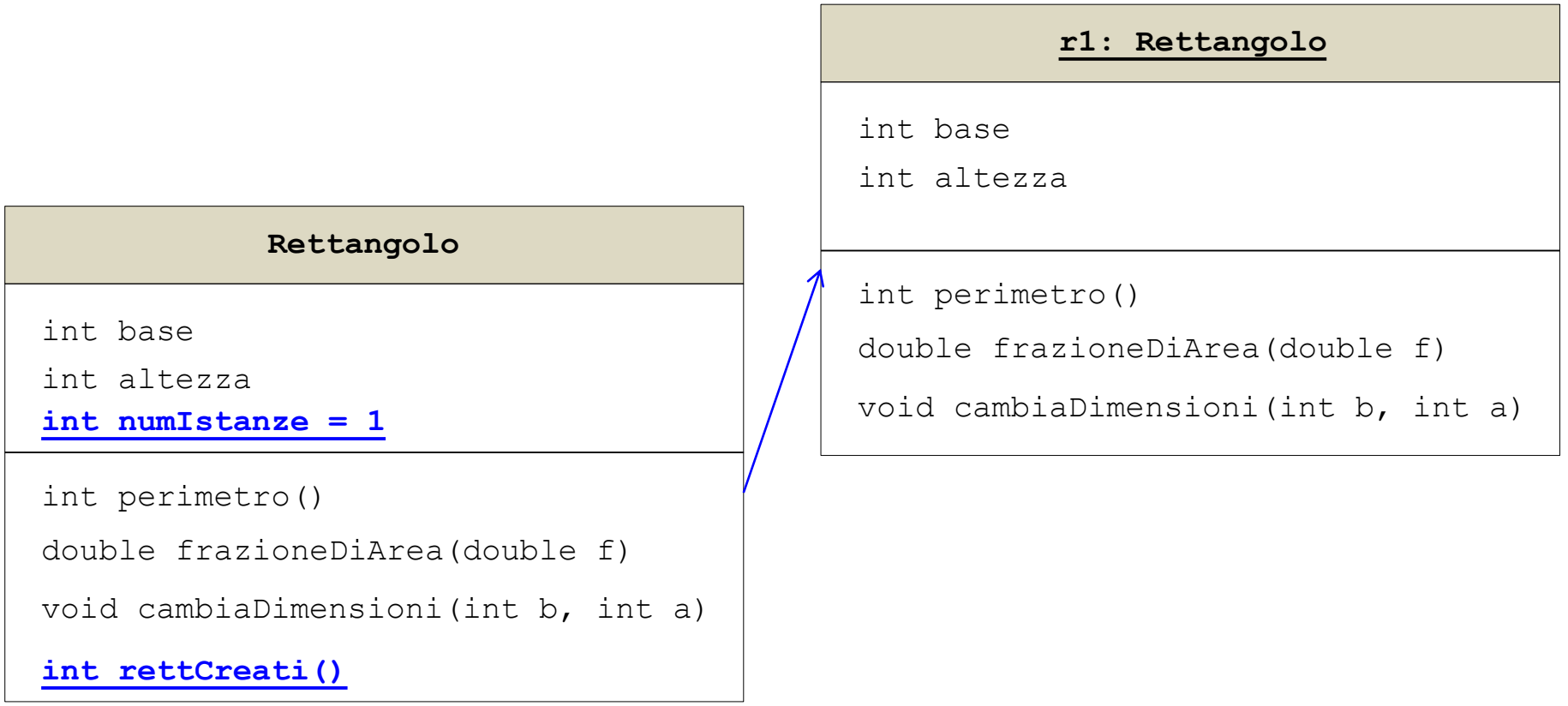
Esempio di attributi e metodi di classe

Rettangolo

`int base`
`int altezza`
`int numIstanze = 0`

`int perimetro()`
`double frazioneDiArea(double f)`
`void cambiaDimensioni(int b, int a)`
`int rettCreati()`

Esempio di attributi e metodi di classe



Esempio di attributi e metodi di classe