



## Oggetti software

Capitolo 2  
ottobre 2011

### Contenuti

- ◆ Programmi e programmazione
- ◆ Introduzione agli oggetti software
- ◆ Esempi di oggetti software
  - l'oggetto software **System.out**
  - l'oggetto software **Math**
  - un oggetto software per la lettura dalla tastiera
  - robot e labirinti
- ◆ Oggetti software
- ◆ Classi e costruzione di oggetti software
  - costruzione di oggetti software
  - componenti di una classe
- ◆ Esempi di classi
  - la classe **Math**
  - le classi **Robot** e **Labirinto**
- ◆ Il linguaggio UML

## Oggetti software

Questo capitolo introduce la programmazione e descrive le principali nozioni del paradigma di programmazione orientata agli oggetti

- che cos'è la programmazione
- oggetti software e loro uso – anche mediante esempi
- progettazione di oggetti software (cenni)
- trattazione indipendente dai linguaggi di programmazione che possono essere utilizzati in pratica

## Programmi e programmazione

Un **programma** (o **applicazione**) è usato da un utente

- per gestire un insieme di informazioni
- per far eseguire un insieme di operazioni a un calcolatore
  - l'esecuzione di un'operazione corrisponde allo svolgimento di una sequenza di azioni da parte del calcolatore

Un programma mostra ad un utente la rappresentazione, nel calcolatore, di una porzione di mondo reale o virtuale

- la porzione di mondo rappresentata da un programma è la **realtà di interesse** del programma
- un programma rappresenta nel calcolatore i dati e le operazioni di una certa realtà di interesse

La nozione di programma percepita da chi realizza programmi è complementare a quella percepita dall'utente

## Il punto di vista del programmatore

Un **programmatore** è uno che progetta e realizza programmi

Per un programmatore, un **programma** è un insieme di frasi che descrivono una certa realtà di interesse

- un **linguaggio di programmazione** è un linguaggio specializzato, comprensibile da parte di un calcolatore
- un programma è un insieme di frasi in un linguaggio di programmazione

Da quali frasi è composto un programma?

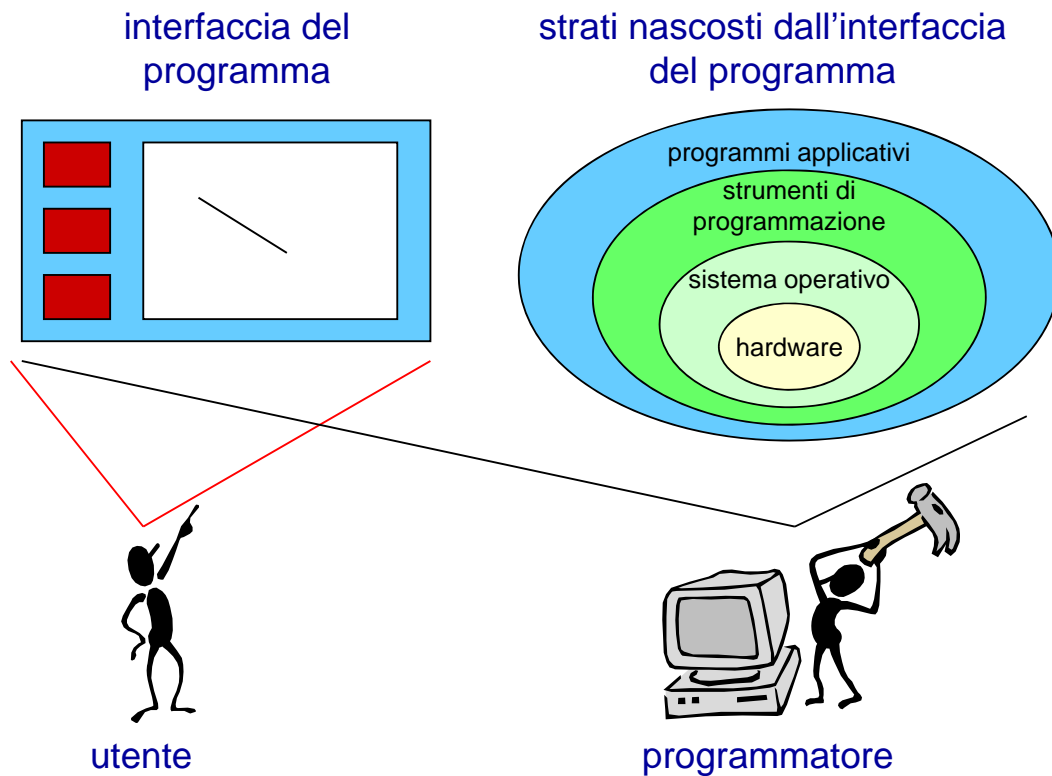
- alcune frasi servono a descrivere come vanno rappresentati i dati della realtà di interesse
- altre frasi servono a descrivere quali azioni vanno svolte (dal calcolatore) quando un utente chiede l'esecuzione di una certa operazione

## Che cosa è la programmazione

La programmazione è

- **scrittura di programmi** – un **programmatore** è una persona che scrive programmi
- **controllo** – un calcolatore fa quello che gli viene detto di fare
- **insegnamento** – un calcolatore impara a eseguire nuove operazioni solo se gli viene detto come vanno fatte
- **modellazione** – un programma rappresenta nel calcolatore una certa realtà di interesse – una porzione di mondo
- **astrazione** – il programmatore deve identificare le caratteristiche essenziali della realtà di interesse da modellare, evitando di descrivere dettagli inutili
- **concretezza** – il calcolatore per eseguire un compito ha bisogno di istruzioni dettagliate
- **risoluzione di problemi** – un programma permette normalmente di fare cose utili, ovvero di “risolvere problemi”
- **creatività** – è più facile descrivere un problema che trovare una sua soluzione

## I punti di vista dell'utente e del programmatore



7

Oggetti software

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Paradigmi di programmazione

Esistono diversi approcci alla programmazione, chiamati paradigmi di programmazione

- **programmazione imperativa**
- **programmazione funzionale**
- **programmazione logica**

Il paradigma di **programmazione orientato agli oggetti**

- **trasversale ai paradigmi imperativo, funzionale e logico**
- **basato sul concetto di oggetto software**
- **in questo corso, la programmazione a oggetti viene presentata come un'estensione della programmazione imperativa**

8

Oggetti software

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

# Oggetti software

Il paradigma di programmazione orientato agli oggetti è basato sul seguente punto di vista

- il mondo reale è fatto di **oggetti**
- dato che un programma è la rappresentazione nel calcolatore di una certa realtà di interesse, allora anche il programma è composto da oggetti, chiamati **oggetti software**
- l'esecuzione di un programma consiste nella cooperazione di un insieme di componenti software chiamati oggetti software
- un programma è la descrizione delle diverse tipologie di oggetti software necessari in un programma

Gli oggetti software

- sono componenti software
- rappresentano elementi della realtà di interesse
- un programma modella una realtà di interesse come una collezione di oggetti software che cooperano
- sono modulari – componibili e riusabili

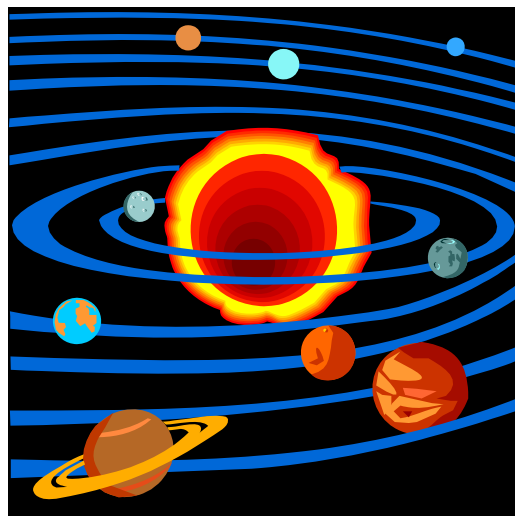
9

Oggetti software

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Un'analogia: oggetti meccanici

Come costruire un modello meccanico del sistema solare?



- sole e pianeti
- orbite
- moto dei pianeti

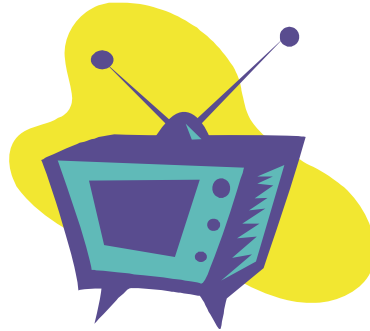
10

Oggetti software

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Un'altra analogia: il televisore

### Il televisore



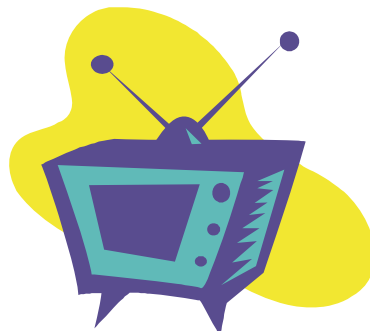
Ogni oggetto software ha un **nome** – chiamato **riferimento**

- in questo caso, lo chiameremo “il televisore”
- il nome è univoco, e consente di referenziare (far riferimento a) l’oggetto

## Operazioni e comportamento

Il televisore sa fare delle cose – sa eseguire delle operazioni

- sa accendersi
- sa sintonizzarsi su un canale
- sa variare il volume
- sa spegnersi

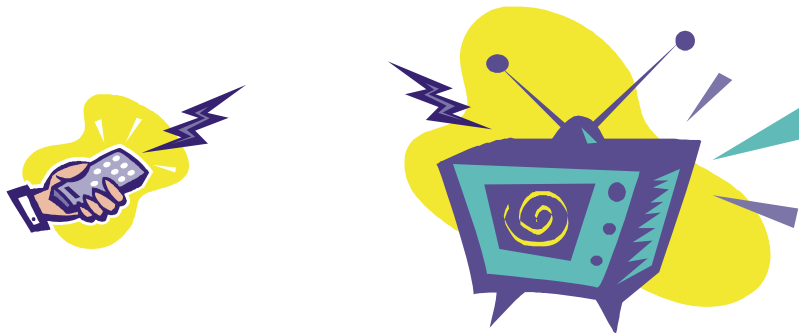


Ogni oggetto software sa eseguire delle **operazioni**

- queste operazioni caratterizzano il **comportamento** dell’oggetto

## Operazioni e messaggi

Come si può chiedere al televisore di eseguire un'operazione?



A un oggetto software è possibile richiedere l'esecuzione di un'operazione mediante l'**invio di un messaggio**

- quando all'oggetto viene chiesto di eseguire un'operazione, allora l'oggetto esegue l'operazione, in modo reattivo

## Interfaccia e uso

Come faccio a conoscere i modi in cui posso usare un oggetto?

L'**interfaccia** di un oggetto software è la descrizione dell'insieme delle operazioni che l'oggetto software sa eseguire

- descrizione di un'operazione
  - formato del messaggio da inviare all'oggetto software per fargli eseguire l'operazione
  - significato dell'operazione

Esempio: un'operazione che il televisore sa eseguire

- spegnimento del televisore
  - messaggio: premere il tasto «**spegnimento**»
  - significato: il televisore si spegne

L'interfaccia di un oggetto software è il suo "manuale d'uso"

- va conosciuta per poter usare un oggetto software
- ma non serve sapere come l'oggetto software è fatto dentro

## Proprietà e stato

In ogni istante di tempo, un oggetto software è in un certo stato

- lo **stato** di un oggetto software è descritto da un insieme di **proprietà**

Esempio: proprietà del televisore

- *accensione*
- *canale*
- *volume*

Ciascuna proprietà è caratterizzata da

- **nome**
- **valore corrente**
- **insieme dei valori ammessi**

## Comportamento e stato

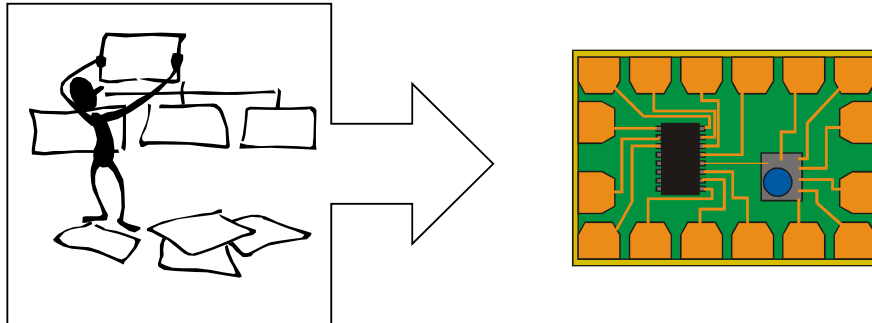
L'effetto di un'operazione può consistere nel cambiamento dello stato (cioè, del valore delle proprietà) dell'oggetto software che la esegue

- **ad esempio**
  - **cambio il canale scegliendo il canale 7**
  - **cambio il volume usando il pulsante +**

## Progettazione

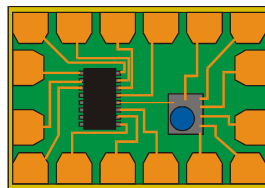
La progettazione del televisore viene effettuata da un progettista

- prima definisce l'interfaccia del televisore
  - progetto delle modalità d'uso
- poi realizza il progetto del televisore
  - progetto del funzionamento interno



La **progettazione** di un oggetto software è la definizione delle caratteristiche dell'oggetto software e dei dettagli per la sua realizzazione (**implementazione**)

## Progettazione e riuso



La progettazione del televisore consiste nell'assemblaggio di

- oggetti (componenti elettronici) standard

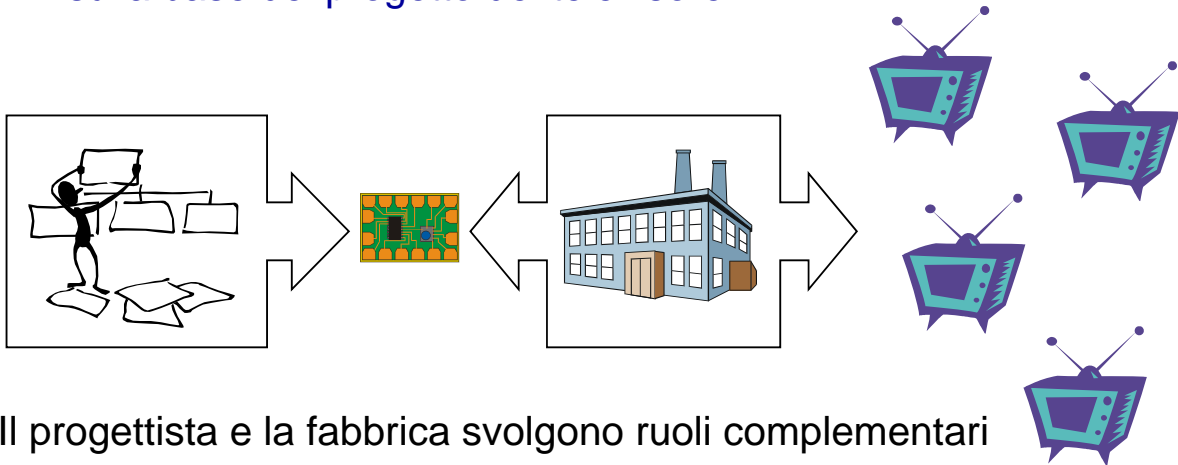


- componenti elettronici progettati in precedenza
- componenti elettronici progettati appositamente

## Costruzione

La costruzione del televisore viene fatta da una fabbrica

- sulla base del progetto del televisore



Il progettista e la fabbrica svolgono ruoli complementari

- necessità di un linguaggio comune

Per gli oggetti software

- il progetto è specificato in un linguaggio di programmazione
  - un progetto per ciascuna “tipologia di oggetto”
- la costruzione è fatta nel/dal calcolatore

## Oggetti software

Un **oggetto software**

- ha un nome – che permette di referenziarlo
- ha un comportamento – sa eseguire delle operazioni
  - si può richiedere a un oggetto software di eseguire un’operazione mediante l’invio di un messaggio
  - l’uso di un oggetto software è descritto dalla sua interfaccia
- ha uno stato – è caratterizzato da un insieme di proprietà

## Progettazione e costruzione di oggetti software

### Un oggetto software

- è progettato e realizzato da un programmatore
  - la progettazione e realizzazione di componenti software è chiamata **implementazione**
  - una **classe** è il progetto di un oggetto software
  - i linguaggi per la definizione classi sono i linguaggi di programmazione (orientati agli oggetti)
- è costruito da un calcolatore
  - gli oggetti software esistono (in modo virtuale) solo nei calcolatori
  - il calcolatore viene usato come una macchina virtuale che sa gestire degli oggetti software, mediante l'esecuzione di istruzioni di appositi linguaggi di programmazione orientati agli oggetti

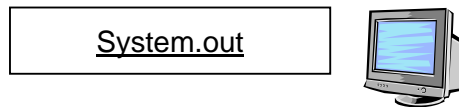
## Esempi di oggetti software

Un programma modella una realtà di interesse come una collezione di oggetti software che cooperano

- molti oggetti software modellano oggetti concreti della realtà di interesse
- altri oggetti software modellano entità concettuali della realtà di interesse
- altri oggetti software vengono introdotti per esigenze realizzative
  - in particolare, i programmi

## L'oggetto software System.out

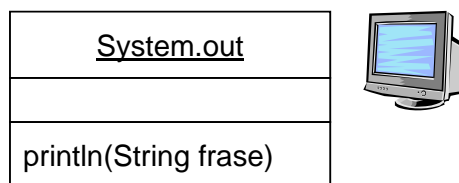
**System.out** è un oggetto software che modella lo schermo del calcolatore



Ciascun oggetto software è caratterizzato da un **nome**, che permette di referenziarlo

- **identificatore** o **riferimento**
- **nomi univoci**

## System.out – operazioni



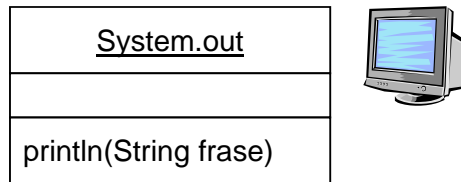
**System.out** sa eseguire un'operazione che visualizza una frase sullo schermo (su se stesso)

- questa operazione si chiama **println**

Ciascun oggetto software ha un **comportamento**

- ovvero, sa eseguire delle **operazioni**

## System.out – operazioni



Le operazioni sono di solito parametriche – sono definite rispetto a un certo numero di **parametri** (o **argomenti**)

- l'operazione **println** ha come parametro la frase da visualizzare
- questo parametro deve essere una **stringa** (una sequenza di caratteri)

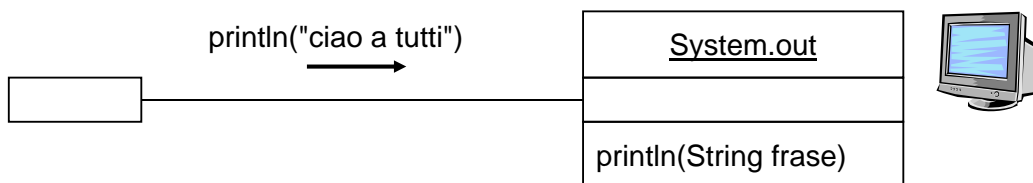
L'uso di un oggetto software è descritto dalla sua **interfaccia**

- ciascuna operazione è descritta
  - dal suo **prototipo** – che comprende un nome ed un elenco di parametri
  - da una descrizione (spesso informale) del suo effetto

## System.out – invio di un messaggio

Per far visualizzare a **System.out** una frase, ad esempio *ciao a tutti*, è necessario inviargli il messaggio

```
println("ciao a tutti")
```

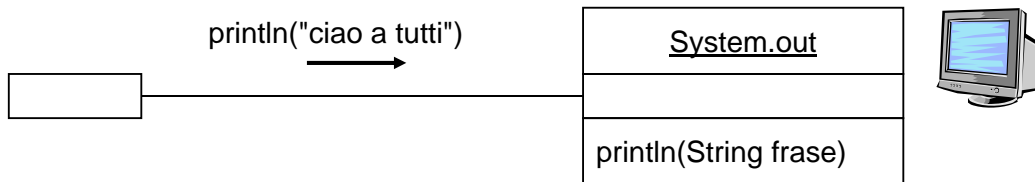


Si può richiedere a un oggetto software di eseguire un'operazione mediante l'invio di un **messaggio**

## System.out – invio di un messaggio

Per far visualizzare a **System.out** una frase, ad esempio *ciao a tutti*, è necessario inviargli il messaggio

```
println("ciao a tutti")
```

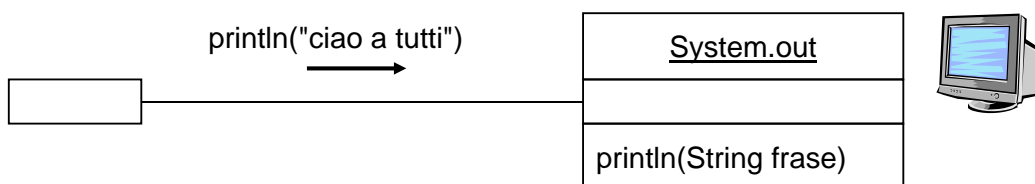


Il messaggio `println("ciao a tutti")` è formato sulla base del prototipo dell'operazione da eseguire, e comprende

- il nome dell'operazione – **println**
- (tra parentesi) il parametro **"ciao a tutti"**
  - le virgolette delimitano la stringa ma non ne fanno parte

## Ricezione di messaggi ed esecuzione di operazioni

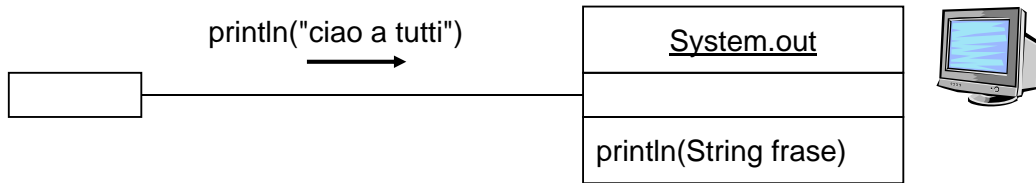
Quando un oggetto software riceve un messaggio, allora esegue l'operazione che gli viene richiesta, in modo parametrico rispetto ai dati specificati



Un oggetto software esegue un'operazione solo quando questa esecuzione gli viene richiesta mediante un messaggio

## Ricezione di messaggi ed esecuzione di operazioni

Quando un oggetto software riceve un messaggio, allora esegue l'operazione che gli viene richiesta, in modo parametrico rispetto ai dati specificati



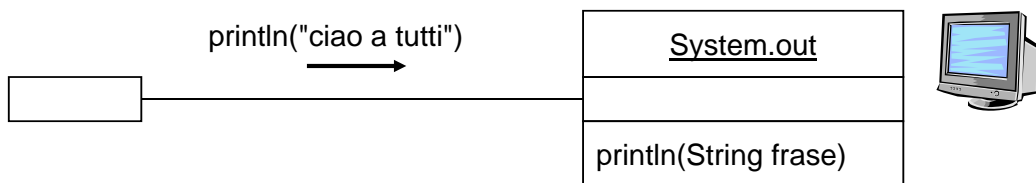
Quando **System.out** riceve il messaggio

```
println("ciao a tutti")
```

allora **System.out** visualizza sullo schermo (su se stesso) la frase

**ciao a tutti**

## Invio di un messaggio



Aspetti dell'invio di un messaggio

- **messaggio**
- **destinatario** (o **ricevitore**) del messaggio
- **mittente** del messaggio

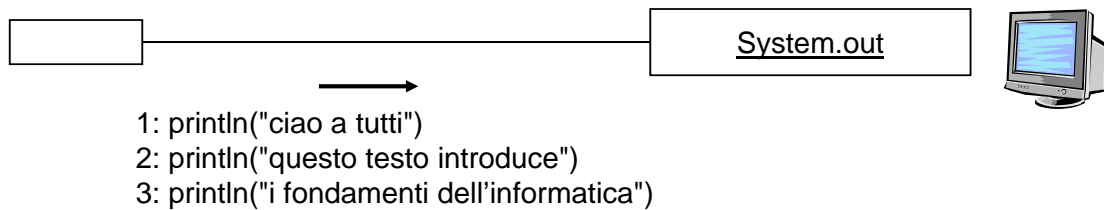
Il rettangolo vuoto rappresenta un altro oggetto software

## System.out – uso

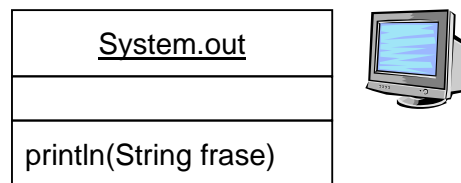
Durante l'esecuzione di un programma vengono di solito scambiati più messaggi

Si supponga di voler visualizzare sullo schermo tre frasi – anziché una sola

- per ciascuna frase da visualizzare sullo schermo è necessario inviare a **System.out** un diverso messaggio
- i messaggi vanno inviati in sequenza



## Modellazione e rappresentazione



**System.out** modella lo schermo del calcolatore

- modellare vuol dire rappresentare alcune caratteristiche
- la modellazione è un'attività di rappresentazione, spesso parziale

Si dice che un oggetto software modella un oggetto reale nel senso che ne rappresenta le operazioni significative nell'ambito di una certa realtà di interesse e/o rispetto a un certo punto di vista

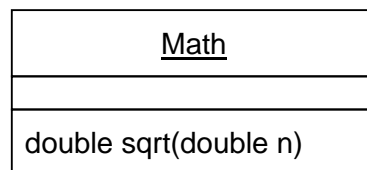
## L'oggetto software Math

I linguaggi di programmazione consentono di manipolare dati numerici mediante gli operatori aritmetici

- somma (+), prodotto (\*), differenza (-), divisione (/)

Molti linguaggi di programmazione non forniscono operatori per valutare altre funzioni matematiche, come la radice quadrata o il calcolo di potenze

- **Math** è un oggetto software bravo in matematica

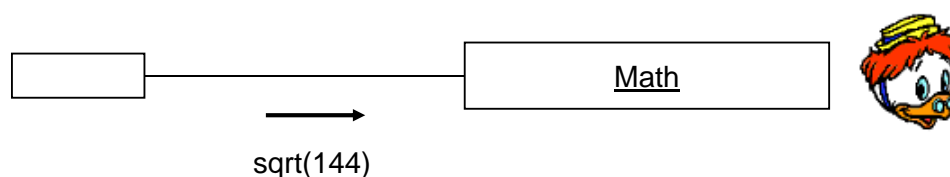


## Math – l'operazione sqrt

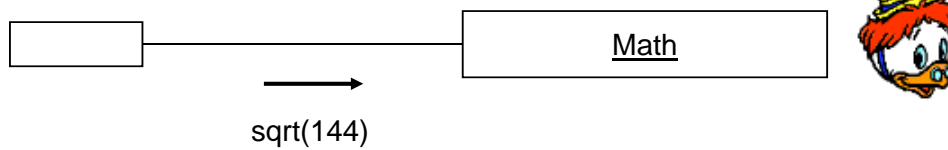
L'oggetto software **Math** sa eseguire un'operazione che calcola la radice quadrata di un numero

- questa operazione si chiama **sqrt**
- l'operazione **sqrt** ha un parametro
  - il numero reale (**double**) di cui si vuole calcolare la radice quadrata

Per calcolare la radice quadrata di 144 bisogna inviare a **Math** il messaggio **sqrt(144)**

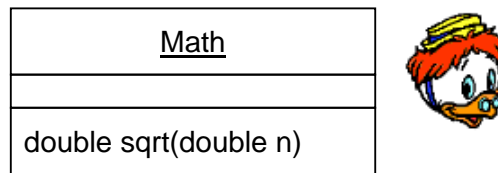


## L'operazione sqrt restituisce un valore



Che cosa deve fare **Math** quando riceve il messaggio **sqrt(144)**?

- deve calcolare la radice quadrata di 144 (12)
- deve restituire il valore calcolato all'oggetto software che gli ha inviato il messaggio

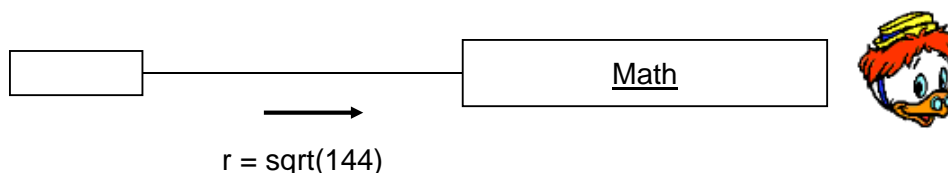


- un'operazione può avere un **tipo di ritorno**

## L'operazione sqrt restituisce un valore

Che cosa deve fare l'oggetto software mittente del messaggio **sqrt(144)** ?

- probabilmente deve memorizzare il valore calcolato da **Math** per poterlo utilizzare in calcoli successivi
- serve una variabile e un'assegnazione



## Variabile

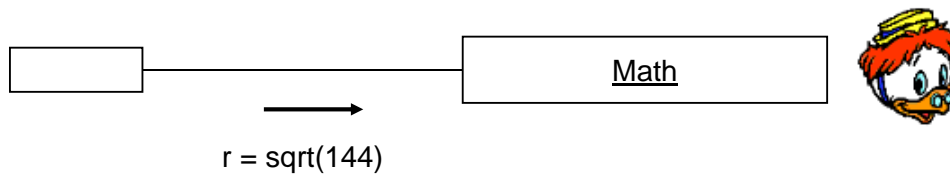
Una **variabile** è un contenitore che può memorizzare un valore

- una variabile è caratterizzata da un nome
- una variabile è caratterizzata da un **tipo**

Il valore della radice quadrata di 144 può essere memorizzato in una variabile **r** (per radice) di tipo **double**

r 12.0  
**double**

## Assegnazione



Un'**assegnazione**

- calcola il valore di un'espressione – **sqrt(144)**
- memorizza il valore calcolato (**12**) in una variabile (**r**)

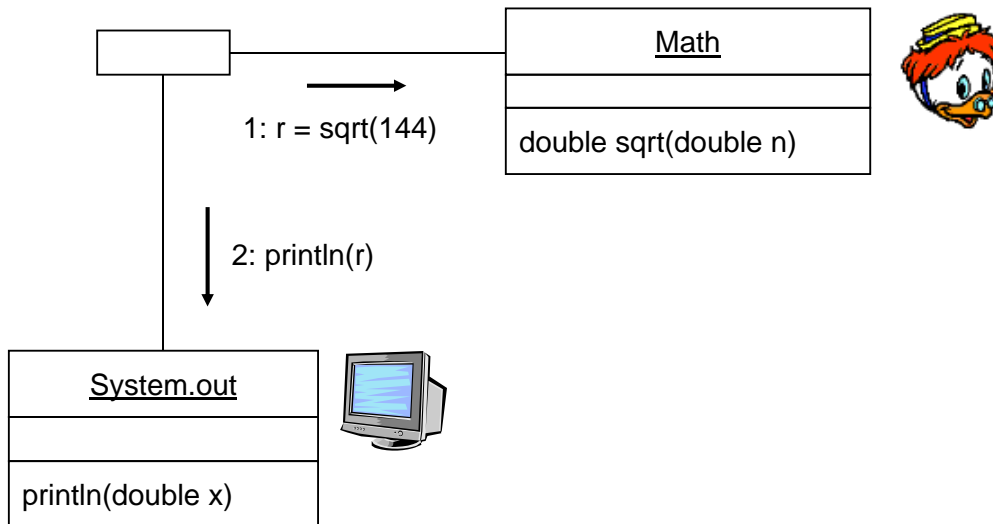
**=** è l'operatore di assegnazione

- a destra dell'**=** viene scritta l'espressione
- a sinistra dell'**=** viene scritto il nome della variabile

## Uso di Math e System.out

Calcolo e visualizzazione della radice quadrata di 144

- **System.out** sa eseguire l'operazione **println(double n)**



- di solito il nome di una variabile denota il valore della variabile
  - se non è scritto nella parte sinistra di un'assegnazione

39

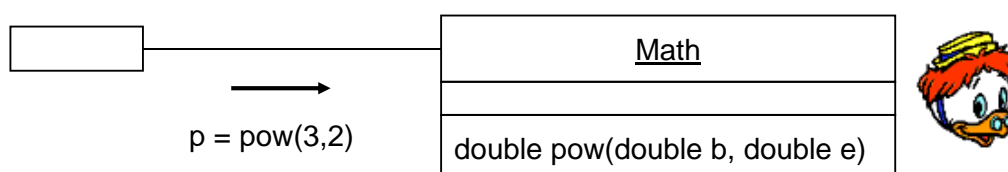
Oggetti software

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Math – l'operazione pow

**Math** sa eseguire anche un'operazione per il calcolo di potenze

- l'operazione si chiama **pow**
- l'operazione **pow** ha due parametri (di tipo **double**)
  - la base **b**
  - l'esponente **e**
- l'operazione **pow** restituisce un valore (di tipo **double**)
- ad esempio, per calcolare  $3^2$  bisogna inviare a **Math** il messaggio **pow(3,2)**
  - l'ordine dei parametri è rilevante



40

Oggetti software

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

# Un oggetto software per la lettura dalla tastiera

**Lettore.in** è un oggetto software che modella la tastiera del calcolatore

| <u>Lettore.in</u>   |
|---|
| char leggiChar()<br>int leggiInt()<br>double leggiDouble()<br>String leggiLinea()<br>boolean eoln() |



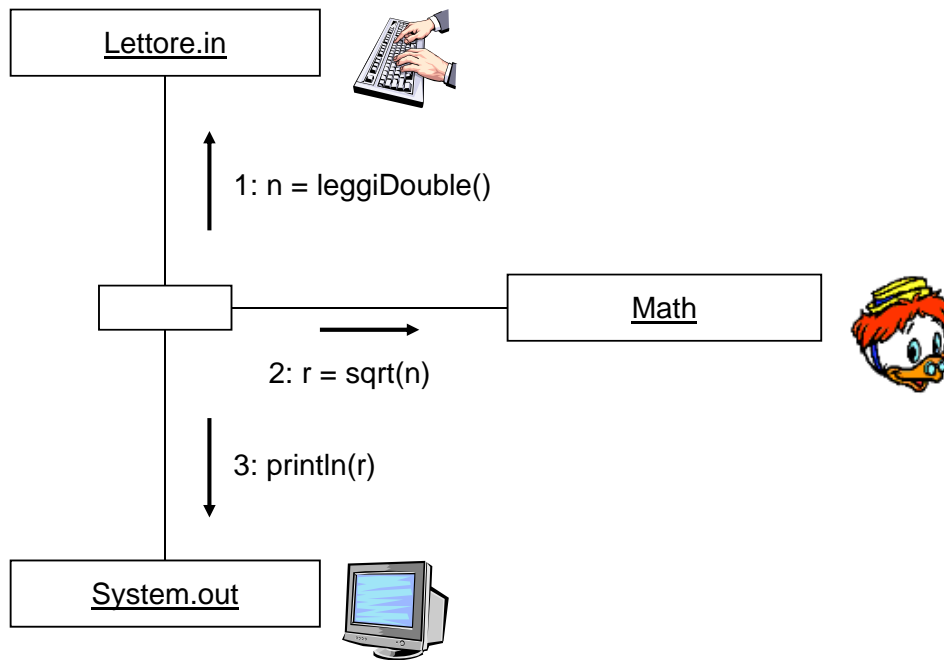
## Operazioni di Lettore.in

- **char leggiChar()**
  - legge un carattere, e restituisce il carattere letto
- **int leggiInt()**
  - legge un numero intero, e restituisce il numero letto
- **double leggiDouble()**
  - legge un numero reale, e restituisce il numero letto
- **String leggiLinea()**
  - legge una linea di testo, e la restituisce
- **String leggiString()**
  - legge una parola, e restituisce la parola letta
- **boolean eoln()**
  - verifica se la linea corrente non contiene altri caratteri da leggere — End Of Line
- **boolean eof()**
  - verifica se non ci sono altri caratteri da leggere — End Of File

## Esempio

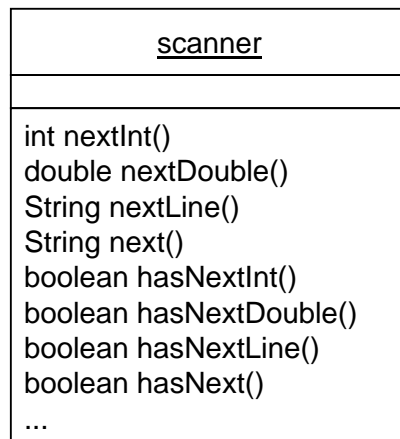


Un oggetto software che legge un numero dalla tastiera, ne calcola la radice quadrata e la visualizza sullo schermo



## Un oggetto software per la lettura dalla tastiera

Per rappresentare la tastiera del nostro calcolatore useremo un oggetto che chiameremo **scanner** oppure **in**

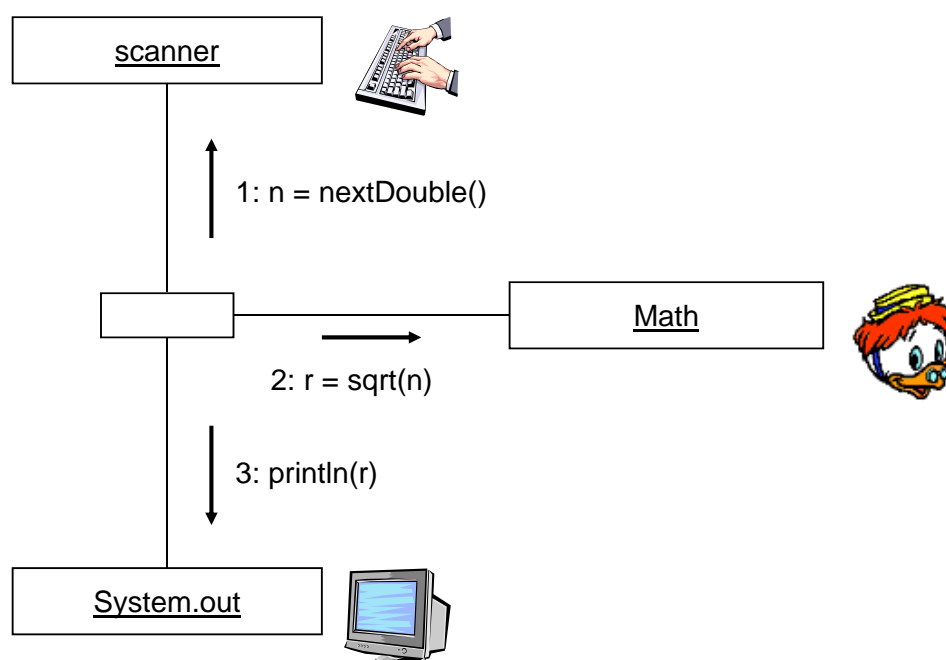


## Alcune operazioni di scanner

- **int nextInt()**
  - legge un numero intero, e restituisce il numero letto
- **double nextDouble()**
  - legge un numero reale, e restituisce il numero letto
- **String nextLine()**
  - legge una linea di testo, e la restituisce
- **String next()**
  - legge un “token” (intuitivamente, una sequenza di caratteri contigui e senza separatori), e restituisce il token letto
- **boolean hasNextInt()** – **boolean hasNextDouble()**
  - verifica se il prossimo token può essere interpretato come un numero intero/reale
- **boolean hasNextLine()** – **boolean hasNext()**
  - verifica se in input è disponibile una ulteriore linea/token

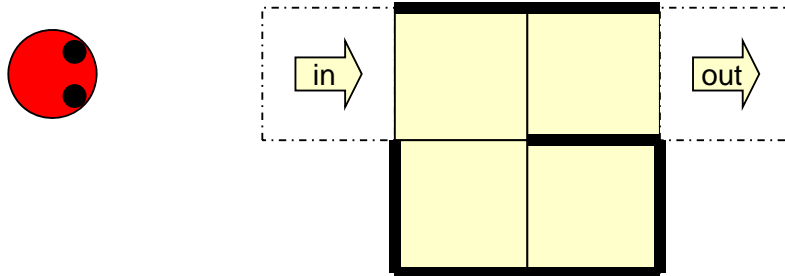
## Esempio

Un oggetto software che legge un numero dalla tastiera, ne calcola la radice quadrata e la visualizza sullo schermo



## Robot e labirinti

Si supponga di voler rappresentare un robot in un labirinto, e di voler controllare l'attraversamento del labirinto da parte del robot

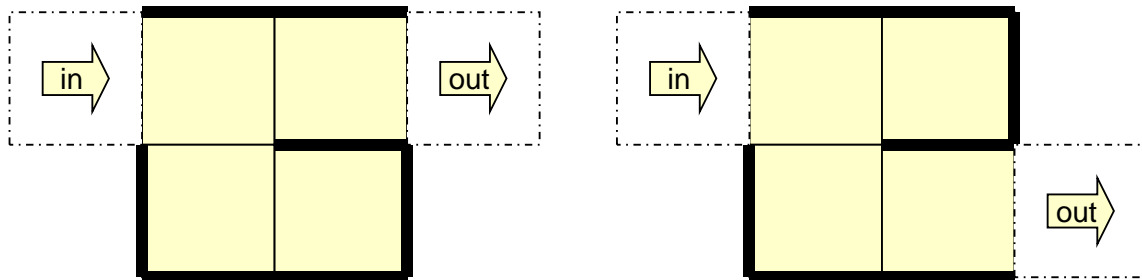


- il robot è modellato da un oggetto software
- anche il labirinto è un oggetto software

## Labirinti

Un labirinto è costituito da un certo numero di celle quadrate

- muri
- ingresso e uscita
- c'è almeno un percorso dall'ingresso all'uscita



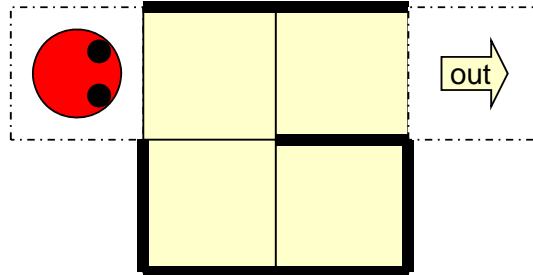
labirinto "semplice"

labirinto "facile"

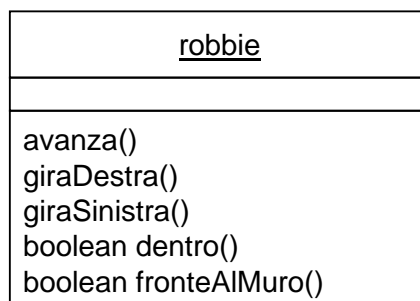
## Robot

### Un robot

- è collocato in un labirinto
- è posizionato in una cella del labirinto
- è orientato nella direzione di uno dei punti cardinali



## Un robot è un oggetto software



## Comportamento dei robot

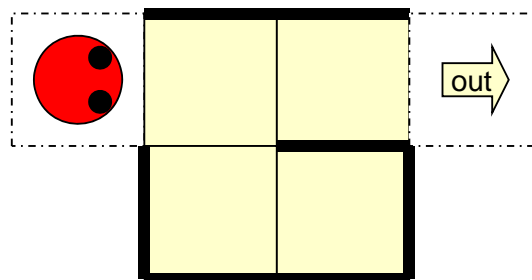
- **avanza()**
  - il robot avanza nella sua direzione corrente, spostandosi nella cella adiacente a quella in cui si trova
  - nota: il robot non deve avere un muro immediatamente di fronte a sé
- **giraDestra()**
  - il robot si gira verso destra, senza avanzare
- **giraSinistra()**
  - il robot si gira verso sinistra, senza avanzare
- **boolean dentro()**
  - il robot verifica se si trova all'interno del labirinto (le celle d'ingresso e d'uscita sono esterne al labirinto)
  - **boolean** è il tipo dei valori di verità (vero/falso)
- **boolean fronteAIMuro()**
  - il robot verifica se ha un muro immediatamente davanti a sé

51

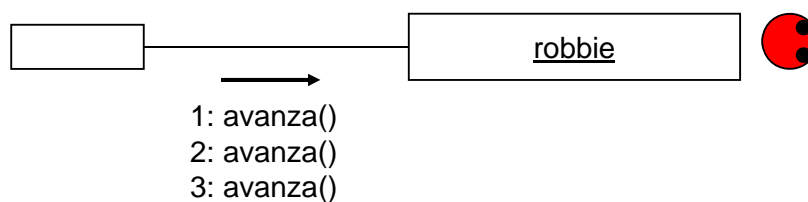
Oggetti software

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Uso del robot



### Attraversamento di un labirinto semplice

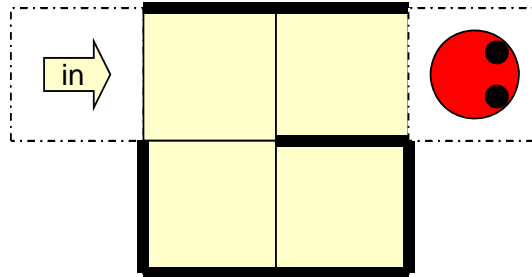
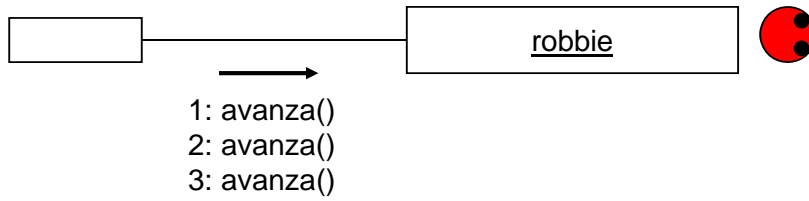


52

Oggetti software

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Attraversamento di un labirinto "semplice"

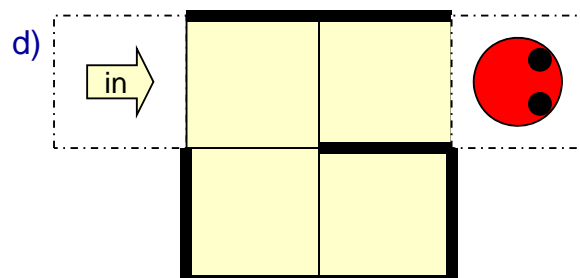
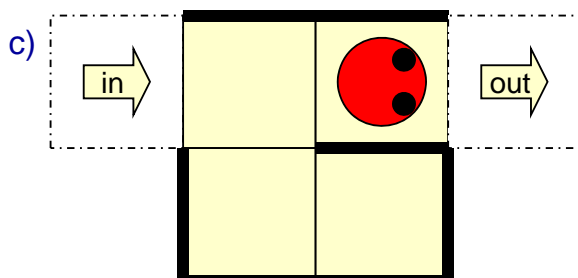
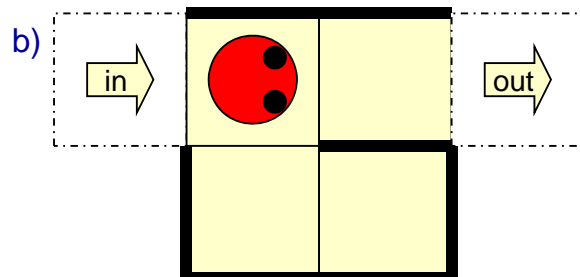
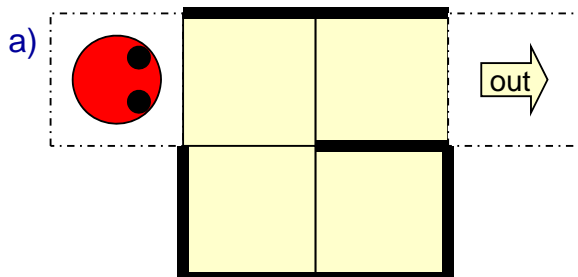
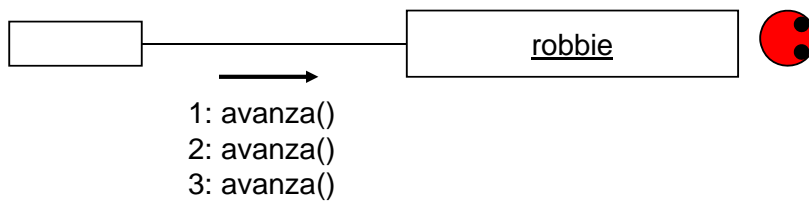


53

Oggetti software

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Attraversamento di un labirinto "semplice"

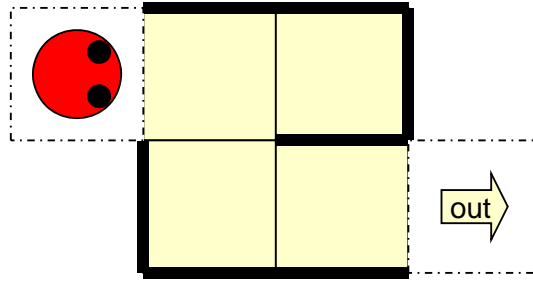


54

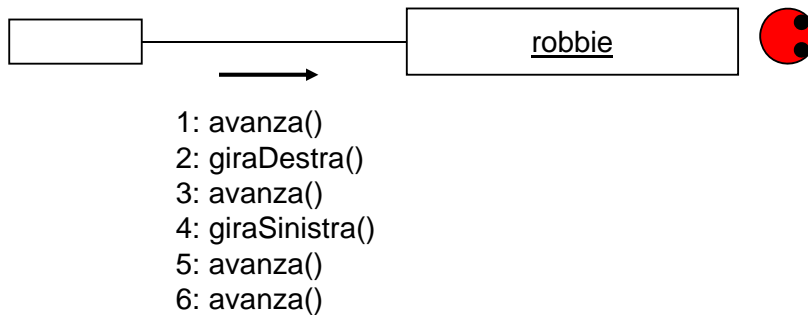
Oggetti software

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

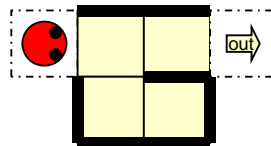
## Attraversamento di un labirinto "facile"



Come si può far attraversare a **robbie** un labirinto "facile"?

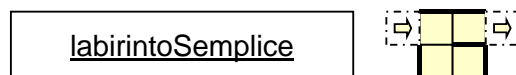
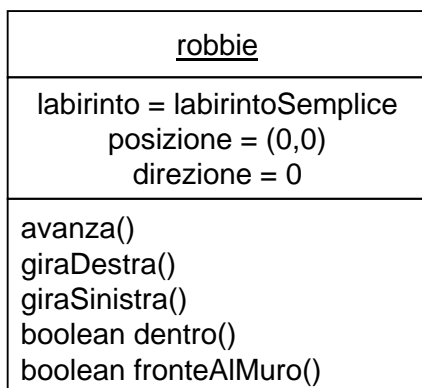


## Stato di un robot



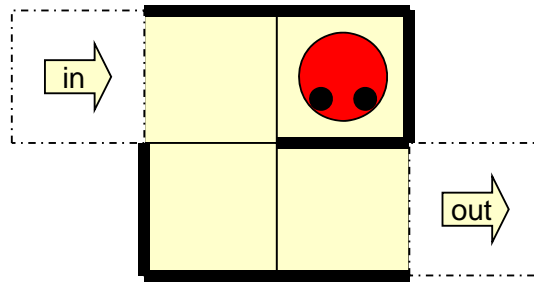
Lo stato di un robot è caratterizzato da tre proprietà

- il **labirinto** in cui si trova il robot
- la **posizione** che il robot occupa nel labirinto in cui si trova
- la **direzione** del robot



## Pre-condizione di un'operazione

Non è possibile chiedere a un robot di avanzare se il robot ha un muro immediatamente di fronte a sé



La **pre-condizione** di un'operazione è l'insieme delle condizioni che devono essere verificate affinché sia corretto chiedere l'esecuzione dell'operazione

- non è corretto richiedere a un oggetto software di eseguire un'operazione se la pre-condizione dell'operazione non è soddisfatta
- è responsabilità di chi invia un messaggio di verificare che la pre-condizione dell'operazione richiesta sia soddisfatta

## Oggetti software

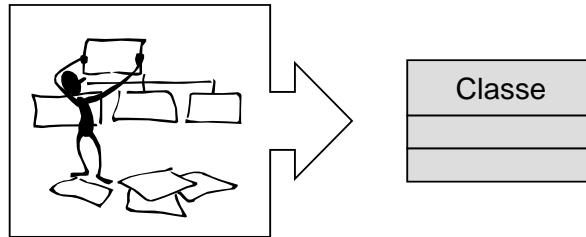
Un **oggetto software** è un'entità software dotata di comportamento e stato

- riferimento
- comportamento
  - operazioni
  - messaggi
  - interfaccia di un oggetto
- stato
  - proprietà

# Classi

Una **classe** è il progetto di un oggetto software

- una descrizione dettagliata, comprensibile da chi dovrà costruire gli oggetti
  - in un linguaggio di programmazione orientato agli oggetti
- implementare un oggetto software consiste nel definire una classe



Una classe è solo un progetto

- la costruzione degli oggetti software sarà svolta dal calcolatore – mediante una macchina virtuale

## Componenti di una classe – idee fondamentali

Una classe è il progetto di un oggetto software

- un oggetto software consiste di stato e comportamento
  - lo stato è un insieme di proprietà
  - il comportamento è un insieme di operazioni
- in corrispondenza, la definizione di una classe comprende
  - la dichiarazione di **variabili** – che rappresentano le proprietà
  - la definizione di **metodi** – che implementano le operazioni
  - un metodo è la descrizione di una sequenza di azioni che l'oggetto deve eseguire quando riceve un messaggio di richiesta di esecuzione di una certa operazione
- la definizione di una classe può comprendere anche altre parti

## Costruzione di oggetti software

Prima di poter utilizzare un oggetto software, è necessario costruirlo a partire dalla classe che ne definisce il progetto

- è una costruzione di tipo software, puramente virtuale
  - gli oggetti software esistono solo nella memoria del calcolatore

## Modalità di costruzione di oggetti software

Due modalità di costruzione di oggetti software da una classe

- modalità di costruzione **statica**
  - un oggetto software viene costruito dalla classe in modo implicito
- modalità di costruzione **dinamica**
  - ci vuole una richiesta esplicita di costruzione dell'oggetto software
  - può essere parametrica

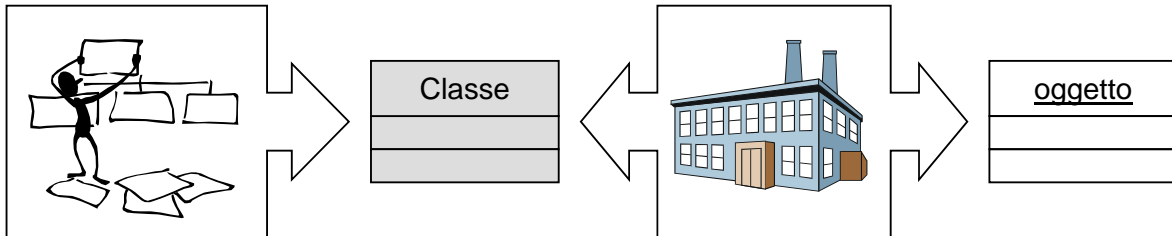
Una classe può offrire una o entrambe le modalità di costruzione

- dipende dalla classe

## Modalità di costruzione statica

### Modalità di costruzione statica di oggetti software

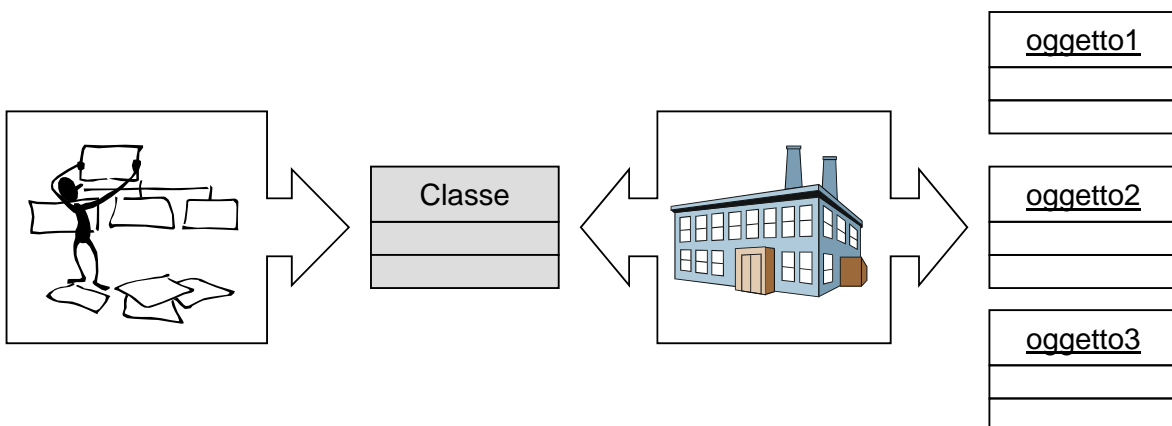
- una classe è il progetto di un singolo oggetto software
- la definizione della classe corrisponde essenzialmente anche alla costruzione dell'oggetto software dalla classe



## Modalità di costruzione dinamica

### Modalità di costruzione dinamica di oggetti software

- la classe è il progetto di una tipologia di oggetti software
- è possibile costruire tanti oggetti software di quel tipo – tra di loro indipendenti



## Creazione di oggetti software

La costruzione di un oggetto software a partire da una classe con la modalità dinamica si chiama

- **creazione** di un oggetto software (dalla classe)
- **istanziamento** (della classe)
  - gli oggetti software istanziati (creati) da una classe si chiamano **istanze** della classe

## Oggetti

Due tipologie di oggetti software

- **oggetto classe**
  - oggetto software costruito automaticamente con modalità statica a partire dalla definizione di una classe
- **oggetto istanza**
  - oggetto software creato con modalità dinamica da una classe

Talvolta, si parla genericamente (e più semplicemente) di **oggetti**

- se la distinzione tra oggetto istanza e oggetto classe non è importante oppure è chiara dal contesto

## Componenti di una classe

Una classe è il progetto di una tipologia di oggetti

- un oggetto software consiste di stato e comportamento
  - lo stato è un insieme di proprietà
  - il comportamento è un insieme di operazioni

In corrispondenza, la definizione di una classe comprende

- la dichiarazione di **variabili**
  - che rappresentano le proprietà
- la definizione di **metodi**
  - che implementano le operazioni
  - un metodo è la descrizione di una sequenza di azioni che l'oggetto deve eseguire quando riceve un messaggio di richiesta di esecuzione di una certa operazione
- la definizione di una classe può comprendere anche la descrizione delle azioni da svolgere al momento della costruzione di oggetti dalla classe

## Modalità di costruzione e componenti di una classe

Nella modalità di costruzione statica

- variabili di classe
- metodi di classe
- blocchi di inizializzazione statica

Nella modalità di costruzione dinamica

- variabili d'istanza
- metodi d'istanza
- costruttori

## Esempi di classi

**Math** è a modalità di costruzione statica

**Robot** e **Labirinto** sono a modalità di costruzione dinamica

## La classe Math

La classe **Math** è a modalità di costruzione statica

- **Math** è un oggetto che per sua natura è unico
- **Math** è un oggetto classe

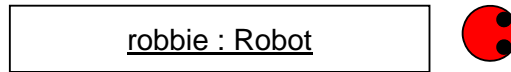
| Math  |
|---|
| static E : double<br>static PI : double   |
| static double sqrt(double n)<br>static double pow(double b, double e)<br>static double log(double n)<br>static double random()<br>... |

| «oggetto classe»<br><u>Math</u>   |
|---|
| E = 2.71...<br>PI = 3.14...   |
| double sqrt(double n)<br>double pow(double b, double e)<br>double log(double n)<br>double random()<br>... |

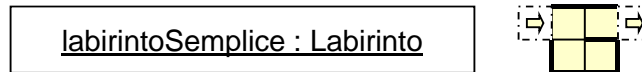
- le proprietà di **Math** sono alcune costanti notevoli
- **static** indica che una variabile o un metodo è di classe

# Le classi Robot e Labirinto

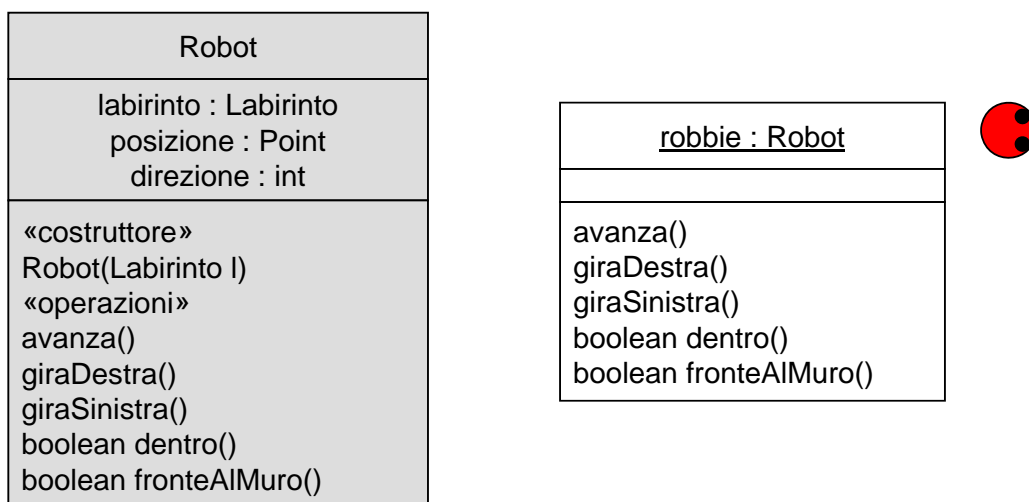
Il robot **robbie** è una istanza della classe **Robot**



I labirinti sono istanze della classe **Labirinto**

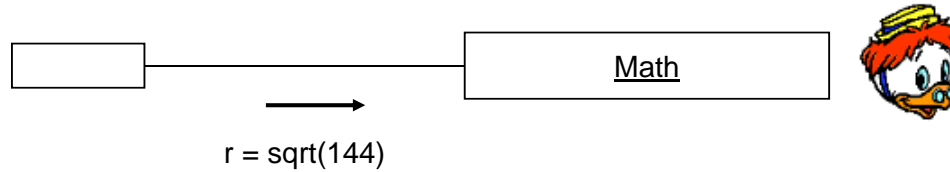


## La classe Robot



# Il linguaggio UML

In questo corso, le classi e gli oggetti vengono spesso descritti usando il linguaggio grafico **UML**



## Unified Modeling Language

- UML è un linguaggio per l'analisi e la progettazione orientata agli oggetti di sistemi software
- diagrammi strutturali
  - diagrammi degli oggetti
  - diagrammi delle classi
- diagrammi comportamentali
  - diagrammi di collaborazione (comunicazione)
  - diagrammi di sequenza