

# Contenuti

- ◆ Programmi e programmazione
- ◆ Oggetti software
- ◆ Esempi di oggetti software
  - robot e labirinti
- ◆ Classi
  - costruzione di oggetti software
  - componenti di una classe
- ◆ Il linguaggio UML

# Introduzione

Nel paradigma di programmazione orientato agli oggetti, l'esecuzione di un programma consiste nella cooperazione di un insieme di componenti software chiamati oggetti software

- il lavoro del programmatore comprende l'identificazione delle tipologie di oggetti software necessarie per un programma, nonché la definizione del progetto di ciascuna tipologia di oggetto software

Questo capitolo presenta le principali nozioni del paradigma di programmazione orientata agli oggetti

- viene introdotta la nozione di oggetto software, anche mediante degli esempi
- viene discussa la progettazione di oggetti software
- la trattazione è abbastanza indipendente dai linguaggi di programmazione che possono essere utilizzati in pratica

# Programmi e programmazione

Un **programma** (o **applicazione**) è usato da un utente per far eseguire un insieme di operazioni a un calcolatore

- ciascuna operazione costituisce uno strumento per l'utente del programma
- l'esecuzione di ciascuna operazione corrisponde allo svolgimento di una sequenza di azioni da parte del calcolatore

Un programma mostra a 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

Le frasi di un programma servono a descrivere

- il modo con cui devono essere rappresentati i dati della realtà di interesse del programma
- le azioni che devono essere svolte da parte del 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

# Il calcolatore per un programmatore

Un calcolatore è un sistema composto da hardware, software di base (che comprende il sistema operativo e gli strumenti di programmazione) e software applicativo

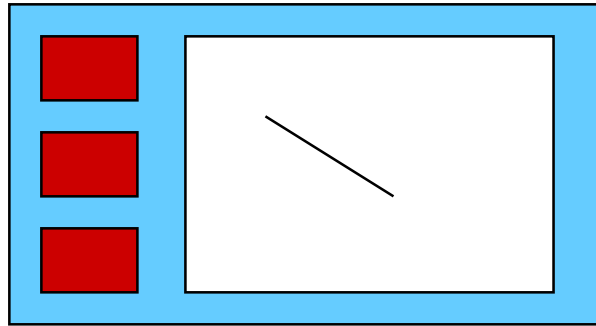
- il calcolatore può essere schematizzato come una cipolla



- il programmatore è normalmente interessato solo agli strati più esterni di questa architettura
  - gli strati più interni riguardano altre figure dell'informatica

# I punti di vista dell'utente e del programmatore

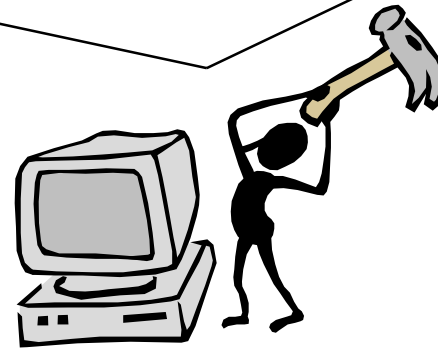
interfaccia del programma



strati nascosti dall'interfaccia del programma



utente



programmatore

# Paradigmi di programmazione

Esistono diversi approcci alla programmazione, chiamati paradigmi di programmazione

- si consideri ad esempio il problema di calcolare dei risultati a partire da certi dati
  - nel paradigma di **programmazione imperativa**, un programma specifica le azioni che devono essere eseguite in sequenza per calcolare i risultati a partire dai dati
  - nel paradigma di **programmazione funzionale**, un programma è la definizione di una funzione, parametrica rispetto ai dati, che calcola i risultati
  - nel paradigma di **programmazione logica**, un programma è la descrizione delle proprietà verificate dai risultati rispetto ai dati

# Programmazione orientata agli oggetti

Questo corso segue un approccio orientato agli oggetti (o, semplicemente, approccio a oggetti)

- questo corso introduce e utilizza il paradigma di **programmazione orientato agli oggetti**
  - l'approccio a oggetti è basato sul concetto di oggetto software (che sarà introdotto tra poco)
- la programmazione a oggetti viene presentata come una estensione del paradigma di programmazione imperativa
  - in realtà, l'approccio a oggetti presenta delle caratteristiche trasversali rispetto a quelle dei vari paradigmi di programmazione, e può essere ad essi ugualmente applicato

# 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**

Nel paradigma di programmazione orientato agli oggetti, gli elementi della realtà di interesse sono dunque rappresentati da componenti software chiamati oggetti software

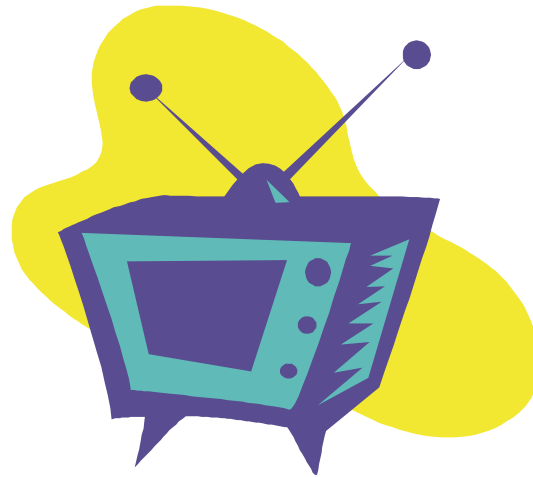
- un programma modella una realtà di interesse come una collezione di oggetti software che cooperano

Una delle caratteristiche fondamentali degli oggetti software è la loro modularità, che ne consente la componibilità e riusabilità

## Un oggetto reale

Vengono ora introdotti alcuni aspetti del paradigma orientato agli oggetti con riferimento a un oggetto del mondo reale

- il televisore

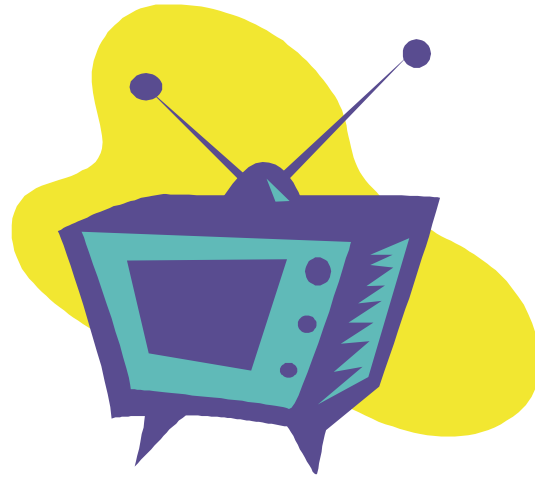


Un oggetto ha un **nome** (in questo caso, “il televisore”)

- il nome di un oggetto permette di far riferimento all’oggetto
  - ovvero, di referenziare l’oggetto

# Operazioni e comportamento

Il televisore sa fare delle cose — sa eseguire delle **operazioni**



Quali operazioni sa eseguire un televisore?

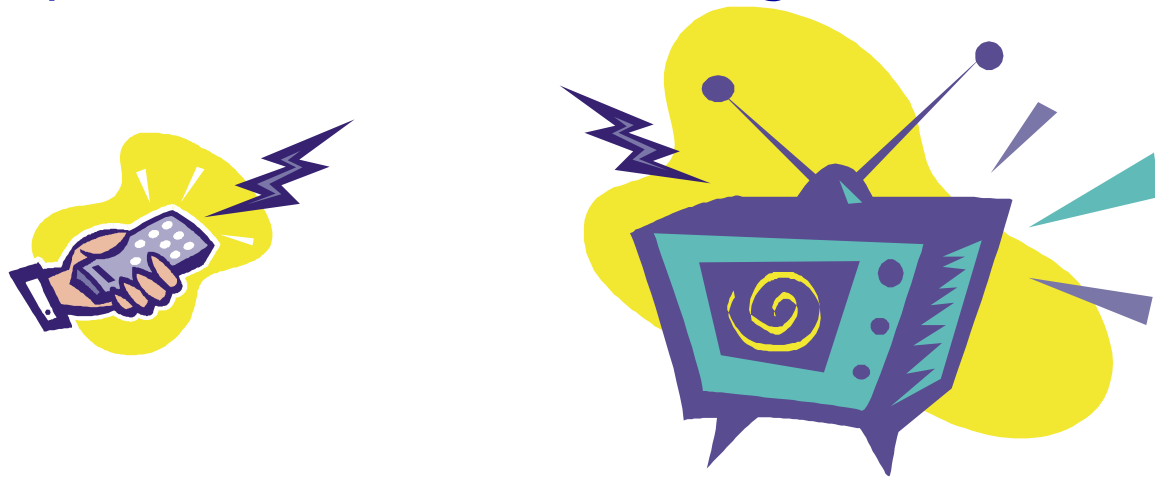
- sa accendersi
- sa sintonizzarsi su un canale
- sa variare il volume
- sa spegnersi

Le operazioni che un oggetto sa eseguire caratterizzano il **comportamento** dell'oggetto

# Operazioni e messaggi

Come si può chiedere al televisore di eseguire una operazione?

- come si può chiedere al televisore di accendersi, di sintonizzarsi su un canale, di variare il volume, di spegnersi?
- è possibile chiedere al televisore di eseguire una operazione premendo un tasto del telecomando
  - il telecomando invia al televisore la richiesta di esecuzione dell'operazione sotto forma di segnale elettromagnetico



Nella terminologia della programmazione orientata agli oggetti, la richiesta dell'esecuzione di una operazione viene fatta mediante l'**invio di un messaggio** a un oggetto

## Interfaccia e uso

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

- la descrizione di una operazione comprende
  - il formato del messaggio che deve essere inviato all'oggetto per fargli eseguire l'operazione
  - la descrizione del significato dell'operazione

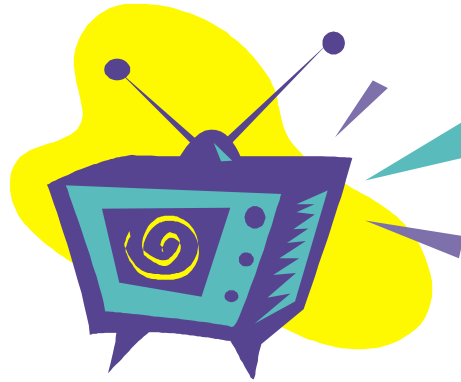
Descrizione di una operazione che il televisore sa eseguire

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

L'interfaccia di un oggetto è il “manuale d'uso” dell'oggetto

- ciò che serve per usare un oggetto è conoscere la sua interfaccia
- non serve sapere come l'oggetto è fatto dentro

# Proprietà e stato



In ogni istante di tempo, il televisore è in un certo stato

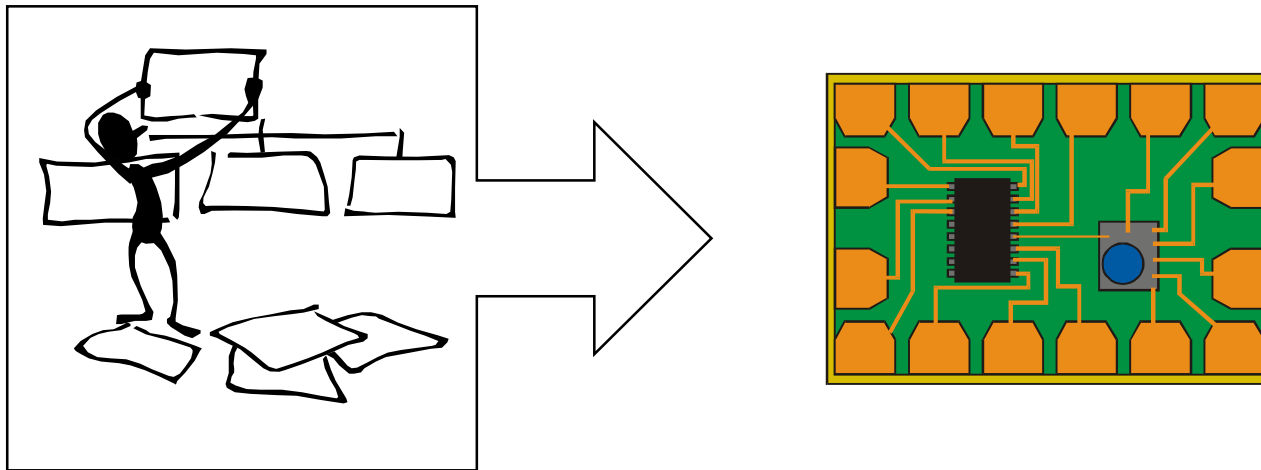
- lo **stato** di un oggetto è descritto da un insieme di **proprietà**
- ad esempio, proprietà di un televisore sono
  - accensione — il televisore è acceso o spento
  - canale — il canale su cui il televisore è sintonizzato
  - volume — il volume del televisore
- ciascuna proprietà è caratterizzata da un nome, dal valore corrente e dall'insieme dei valori ammessi

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

# Progettazione

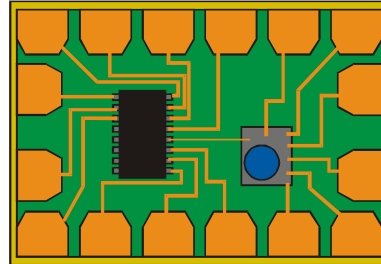
La progettazione del televisore viene effettuata da un progettista

- viene prima definita l'interfaccia del televisore
  - il progettista stabilisce come l'oggetto potrà essere usato dall'utente
- viene poi realizzato il progetto del televisore
  - il progettista definisce lo schema elettrico del televisore



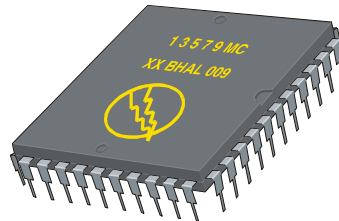
La **progettazione** di un oggetto è la definizione delle caratteristiche dell'oggetto e dei dettagli per la sua realizzazione

# Progettazione e riuso



La progettazione del televisore consistere dell'assemblaggio di

- oggetti (componenti elettronici) standard, disponibili sul mercato
  - che vengono usati conoscendone l'interfaccia

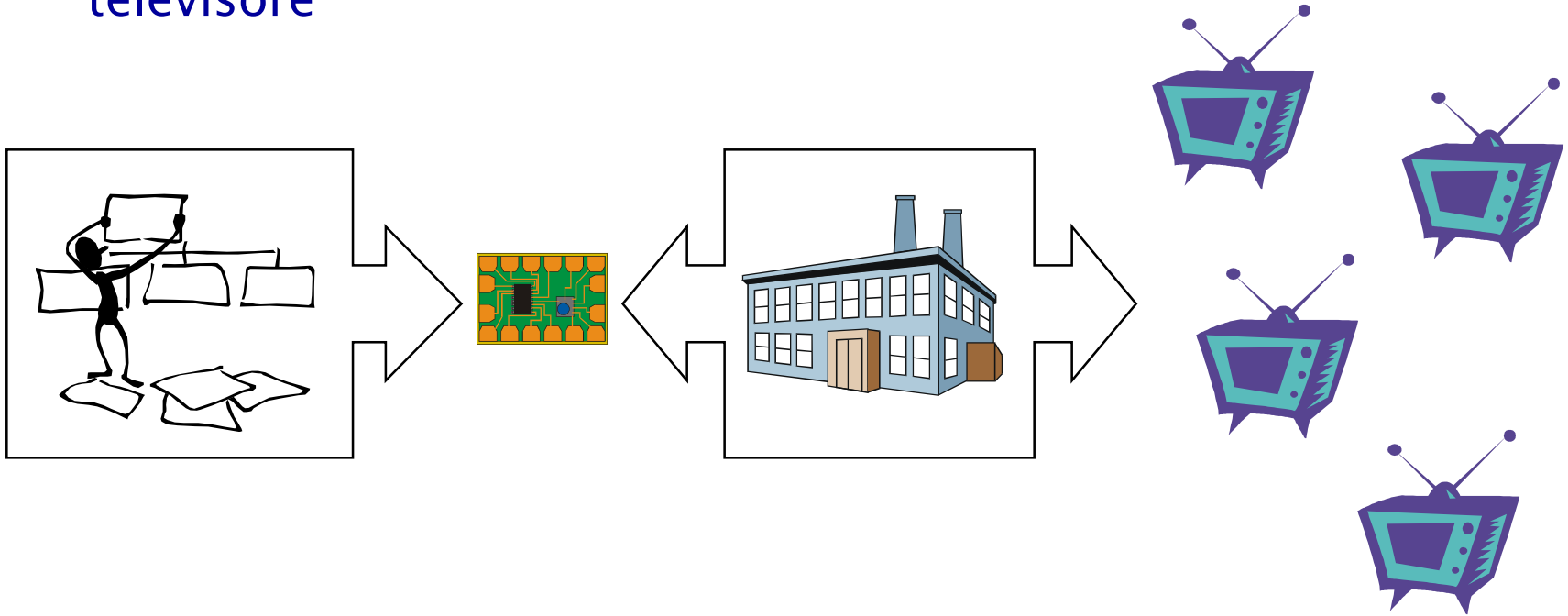


- componenti elettronici progettati in precedenza
  - che vengono riusati senza modifiche
- componenti elettronici progettati appositamente

# Costruzione

La costruzione del televisore viene fatta da una fabbrica

- la fabbrica costruisce televisori sulla base del progetto del televisore



Il progettista e la fabbrica svolgono ruoli complementari

- la complementarietà tra progettista e fabbrica è possibile perché viene adottato un linguaggio comune per la scrittura dei progetti

# 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 una 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
  - una astronave aliena, un documento, un libro, uno studente
- altri oggetti software modellano entità concettuali della realtà di interesse
  - la traiettoria di una astronave, una equazione, una sequenza di caratteri, un esame
- altri oggetti software vengono introdotti per esigenze realizzative
  - una finestra o un cursore sullo schermo
  - anche il programma è rappresentato da un oggetto software

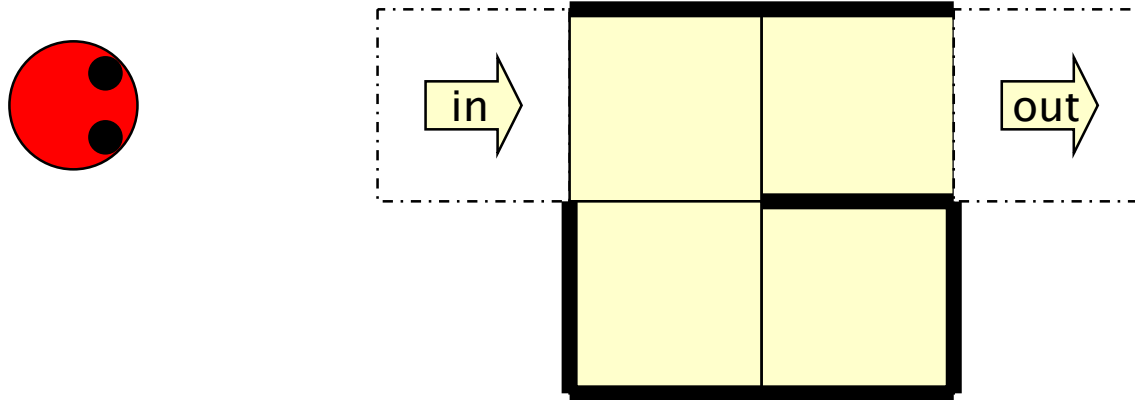
# Esempi di oggetti software

Vengono ora descritti, mediante degli esempi, alcuni oggetti software e il loro uso

- in questi esempi, gli oggetti software vengono descritti mediante un formalismo grafico (il linguaggio UML)
- anche la cooperazione tra oggetti software viene descritta utilizzando il linguaggio UML
- va però osservato che il linguaggio UML non è un linguaggio di programmazione ma, piuttosto, un linguaggio per la descrizione di oggetti e classi

# 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



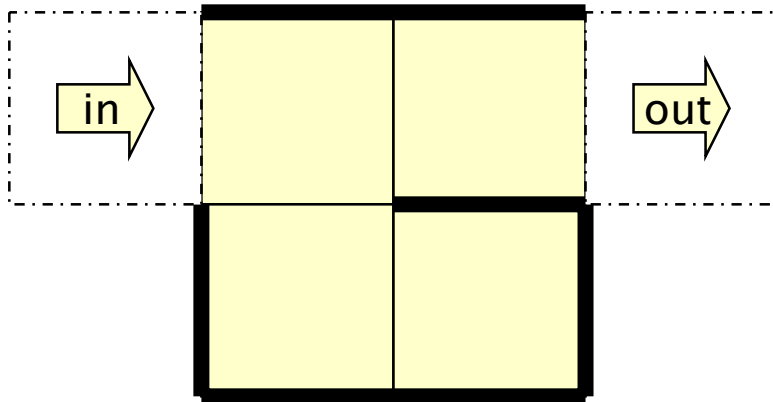
- il robot è modellato da un oggetto software
- anche il labirinto è un oggetto software

# Labirinti

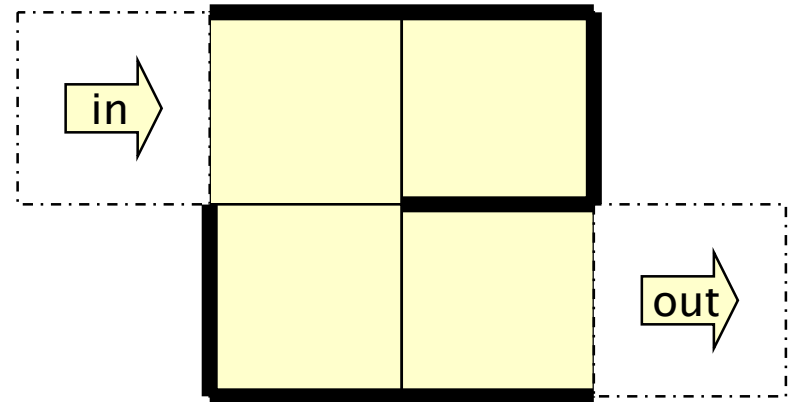
Un labirinto è costituito da un certo numero di celle quadrate

- celle adiacenti sono solitamente collegate
  - ma possono anche essere separate da un muro
- in un labirinto c'è una cella d'ingresso e una d'uscita
  - l'ingresso e l'uscita sono considerati esterni al labirinto
- c'è almeno un percorso dall'ingresso all'uscita

Esempi di labirinti



labirinto "semplice"

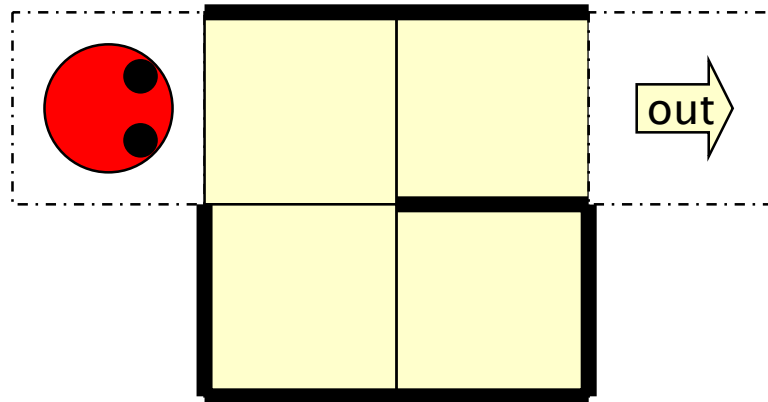


labirinto "facile"

# Robot

Un robot

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



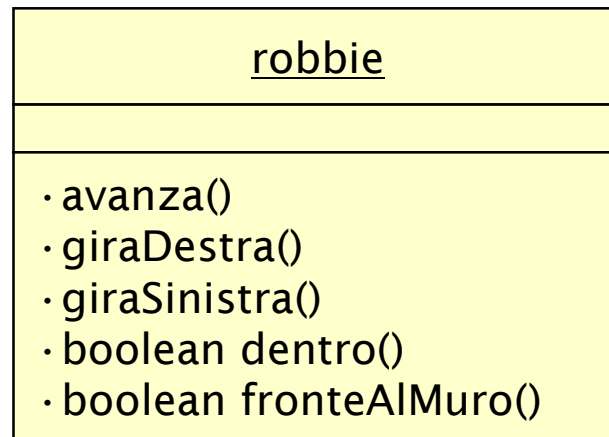
Questo robot

- è collocato in un labirinto “semplice”
- è posizionato nella cella d’ingresso del labirinto
- è orientato verso est, nella direzione d’ingresso del labirinto

# Un robot è un oggetto software

La figura mostra una rappresentazione grafica dell'oggetto software **robbie** che descrive anche la sua interfaccia completa

- l'interfaccia di un oggetto software è l'elenco delle operazioni che l'oggetto software sa eseguire

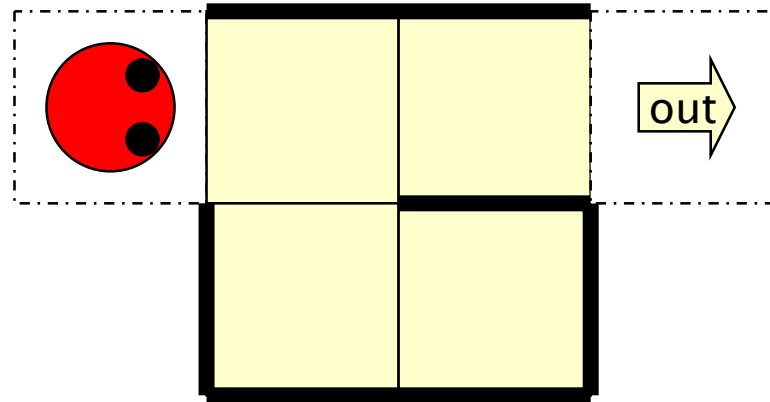


# Comportamento dei robot

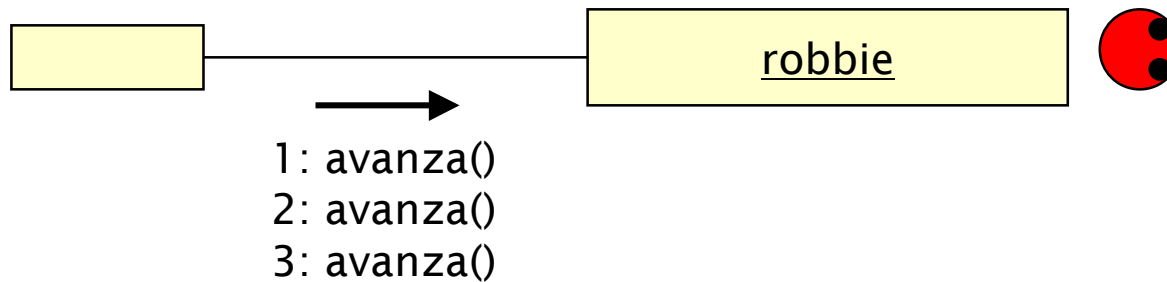
Operazioni che un robot sa eseguire

- **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 fronteAlMuro()**
  - il robot verifica se ha un muro immediatamente davanti a sé

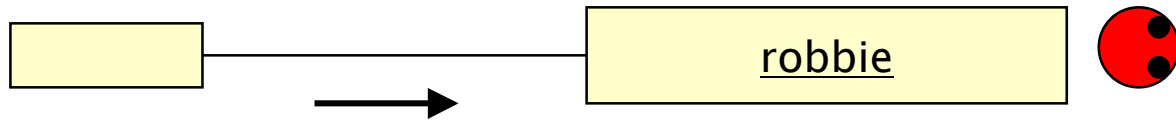
# Uso del robot



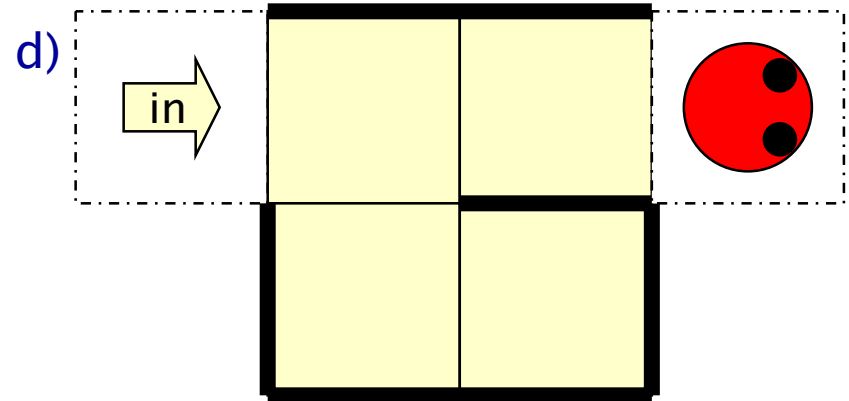
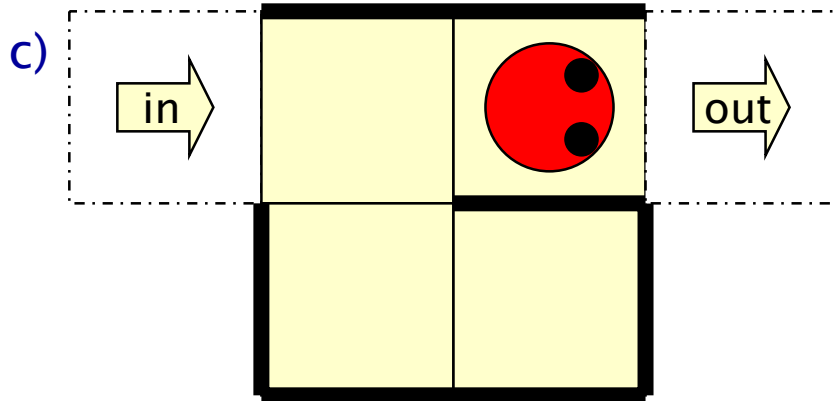
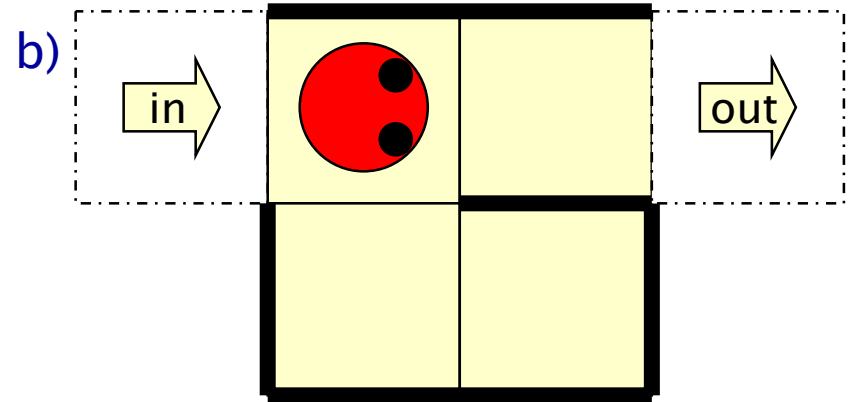
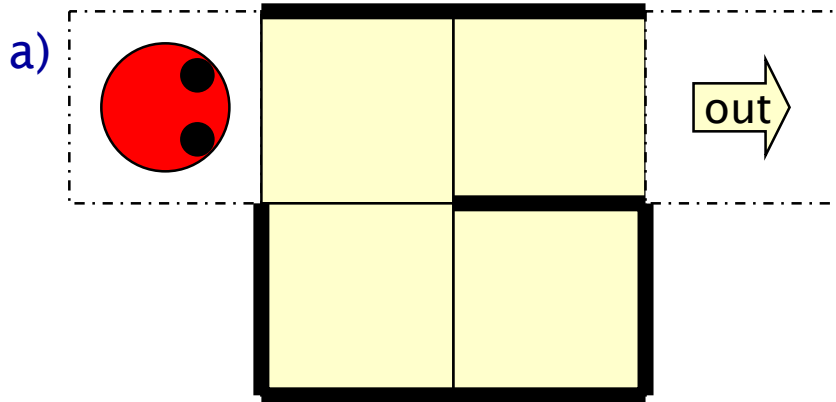
Per far attraversare questo labirinto al robot **robbie** (a partire dalla posizione che occupa) bisogna farlo avanzare tre volte



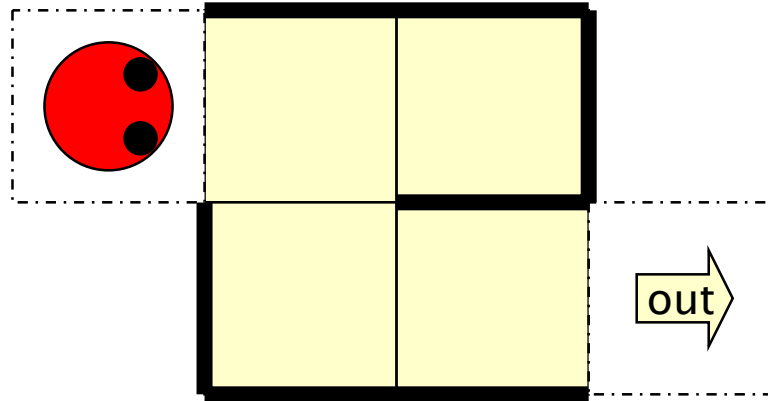
# Attraversamento di un labirinto "semplice"



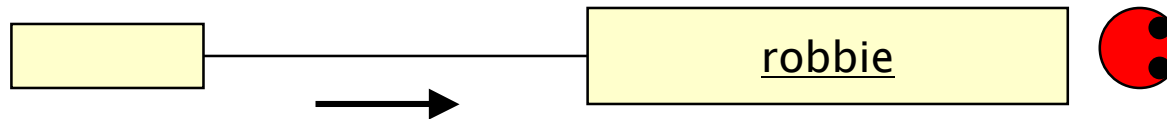
1: avanza()  
2: avanza()  
3: avanza()



# Attraversamento di un labirinto "facile"

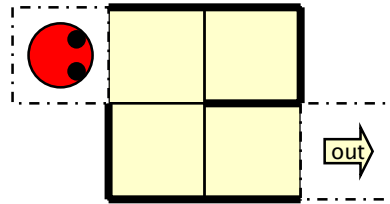


Messaggi che consentono al robot **robbie** di attraversare un labirinto "facile"



- 1: avanza()
- 2: giraDestra()
- 3: avanza()
- 4: giraSinistra()
- 5: avanza()
- 6: avanza()

# Stato di un robot



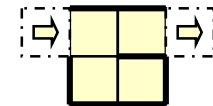
Lo stato di un robot è caratterizzato da tre proprietà

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

<u>robbie</u>
labirinto = labirintoSemplice posizione = (0,0) direzione = EST
· avanza() · giraDestra() · giraSinistra() · boolean dentro() · boolean fronteAlMuro()

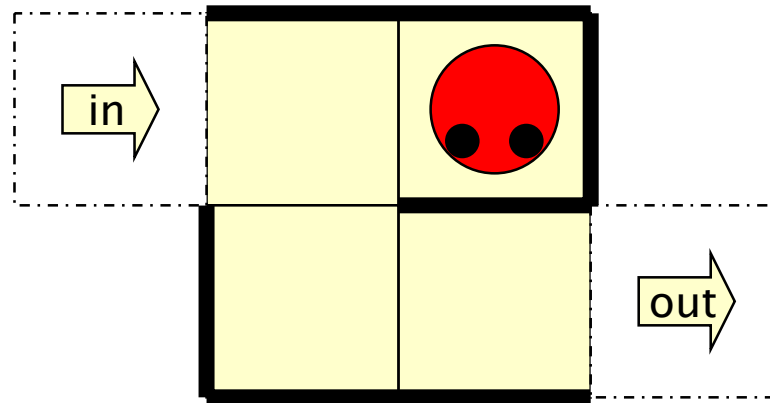


labirintoSemplice



## Pre-condizione di una operazione

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

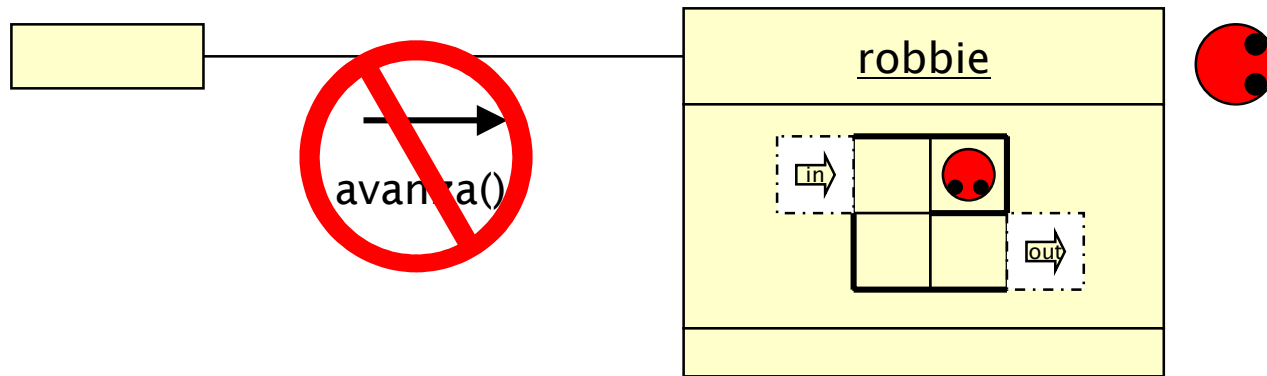


La **pre-condizione** di una operazione è l'insieme delle condizioni che devono essere verificate affinché sia corretto chiedere l'esecuzione dell'operazione

- ad esempio, l'operazione **avanza()** di un robot ha la seguente pre-condizione
  - il robot non ha un muro immediatamente di fronte a sé
- le pre-condizioni sono espresse in termini dei parametri e degli oggetti coinvolti dall'operazione

# Pre-condizione e responsabilità

Non è corretto richiedere a un oggetto software di eseguire una 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

# Classi

Gli esempi hanno mostrato l'uso di alcuni oggetti software

- l'uso di un oggetto software
  - è descritto dalla sua interfaccia
  - si concretizza mediante l'invio di messaggi
- per l'utente di un oggetto software è sufficiente conoscere l'interfaccia dell'oggetto software
  - l'utente di un oggetto software può ignorare come l'oggetto software sia fatto internamente

L'implementazione di un oggetto software consiste della

- definizione dell'interfaccia dell'oggetto software
  - la modalità d'uso dell'oggetto software deve essere definita in modo preciso
- definizione del progetto completo dell'oggetto software
  - definizione del progetto dello stato e del comportamento dell'oggetto software
  - il progetto di un oggetto software si chiama **classe**

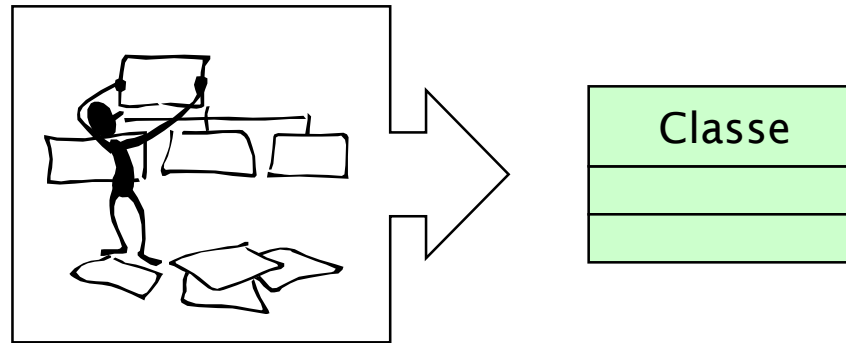
# Classe

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

- l'implementazione di un oggetto software consiste nella definizione di una classe

Il lavoro del progettista di un oggetto software

- definire la classe per l'oggetto software



# Costruzione di oggetti software

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

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

Come si costruisce un oggetto software?

- la definizione della classe per un oggetto software è scritta dal programmatore usando come formalismo un linguaggio di programmazione (orientato agli oggetti)
- la costruzione di oggetti software viene svolta dal calcolatore, usato come macchina virtuale in grado di eseguire programmi scritti nel linguaggio di programmazione scelto

# Modalità di costruzione di oggetti software

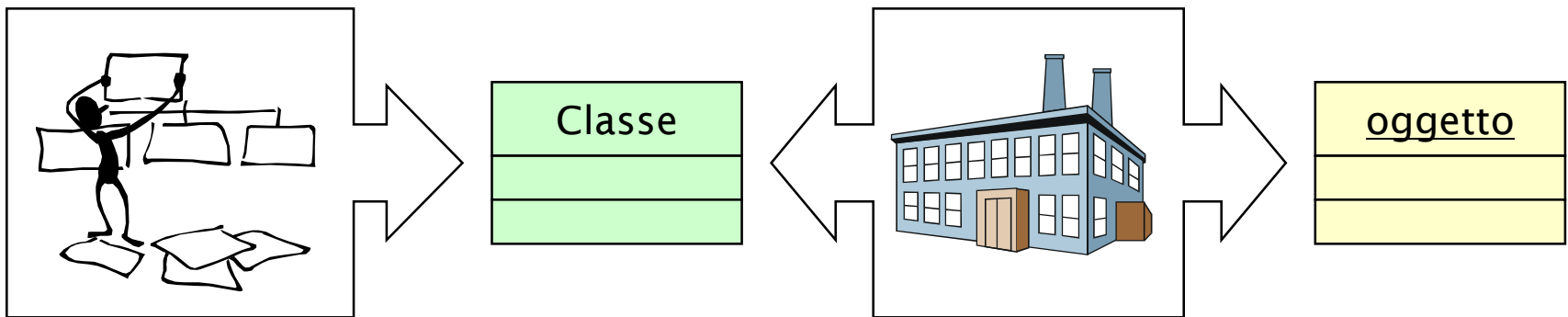
La costruzione di oggetti software a partire da una classe può avvenire secondo due modalità

- nella modalità di costruzione **dinamica**, la costruzione di un oggetto software dalla classe avviene normalmente solo in seguito a una richiesta esplicita di costruzione dell'oggetto dalla classe
  - prima di poter utilizzare un oggetto software con modalità di costruzione dinamica è normalmente necessario richiedere la costruzione dell'oggetto
- nella modalità di costruzione **statica**, invece, la costruzione dell'oggetto software dalla classe avviene in modo implicito
  - è possibile utilizzare oggetti software con modalità di costruzione statica senza doverne richiedere la costruzione
- la modalità con cui è possibile creare oggetti software da una classe è una caratteristica della classe
  - ogni classe può permettere una, l'altra o entrambe le modalità di costruzione di oggetti software

# Modalità di costruzione statica

Nella **modalità di costruzione statica** di oggetti software

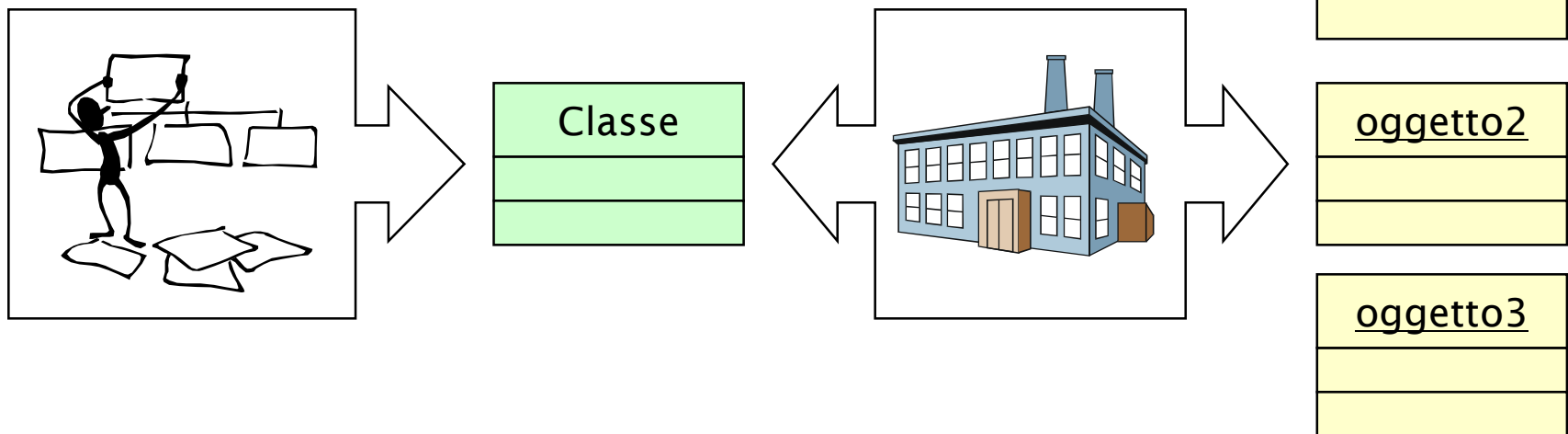
- una classe è il progetto di un singolo oggetto software
- la definizione della classe corrisponde essenzialmente anche alla costruzione dell'oggetto software dalla classe
  - l'aver definito una classe rende disponibile un oggetto software che ha le caratteristiche definite dalla classe



# Modalità di costruzione dinamica

Nella **modalità di costruzione dinamica** di oggetti software

- la classe è il progetto di una tipologia di oggetti software
- la costruzione di un oggetto software a partire dalla classe deve essere solitamente richiesta in modo esplicito
  - al momento della definizione della classe non viene costruito nessun oggetto software di quel certo tipo
- è possibile costruire tanti oggetti software di quel tipo
  - questi oggetti sono fra di loro indipendenti



# Modalità di costruzione statica e dinamica

Le due modalità di costruzione di oggetti software hanno finalità diverse

- la modalità statica permette di costruire esattamente un oggetto software che ha certe caratteristiche
  - utile se bisogna modellare un oggetto della realtà di interesse che è per sua natura unico
- la modalità dinamica permette di costruire (su richiesta) tanti oggetti software, tutti con le stesse caratteristiche, ma ciascuno indipendente dagli altri
  - utile se bisogna modellare una tipologia di oggetti della realtà di interesse (ad esempio, dei robot)
  - permette inoltre di specificare parametri da utilizzare nella costruzione dei singoli oggetti software

# 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) — e anche
- **istanziamento** (della classe)
  - gli oggetti software istanziati (creati) da una classe si chiamano **istanze** della classe

# Oggetti

Un oggetto software è chiamato

- **oggetto classe**

- se è l'oggetto software costruito automaticamente con modalità statica a partire dalla definizione di una classe

- **oggetto istanza**

- se è un oggetto software creato con modalità dinamica da una classe

Inoltre, si parla genericamente (e più semplicemente) di **oggetti** quando la distinzione tra oggetto istanza e oggetto classe non è necessaria 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

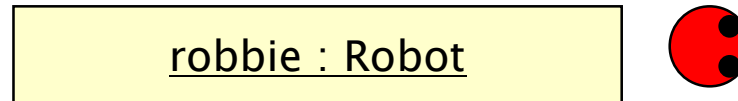
Viene fatta una distinzione tra componenti relative alla modalità di costruzione statica e quelle relative alla modalità dinamica

- per la modalità di costruzione statica
  - le variabili e i metodi da associare all'oggetto classe si chiamano **variabili di classe** e **metodi di classe**
  - le sequenze di istruzioni per l'inizializzazione dell'oggetto classe si chiamano **blocchi di inizializzazione**
- per la modalità di costruzione dinamica
  - le variabili e i metodi da associare a ciascun oggetto istanza costruito dalla classe si chiamano **variabili d'istanza** e **metodi d'istanza**
  - le sequenze di istruzioni per l'inizializzazione degli oggetti istanza si chiamano **costruttori**

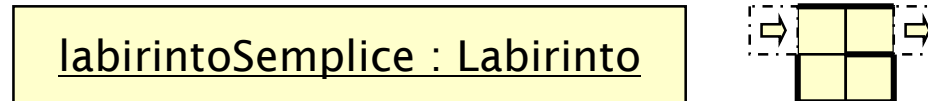
## Esempio — la classe Robot e la classe Labirinto

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

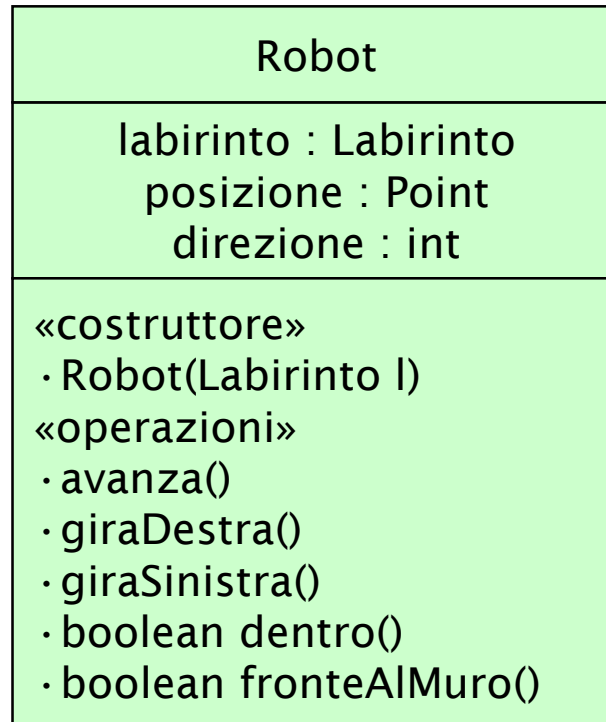
- la classe **Robot** è il progetto per robot in labirinti



- i labirinti sono istanze della classe **Labirinto**



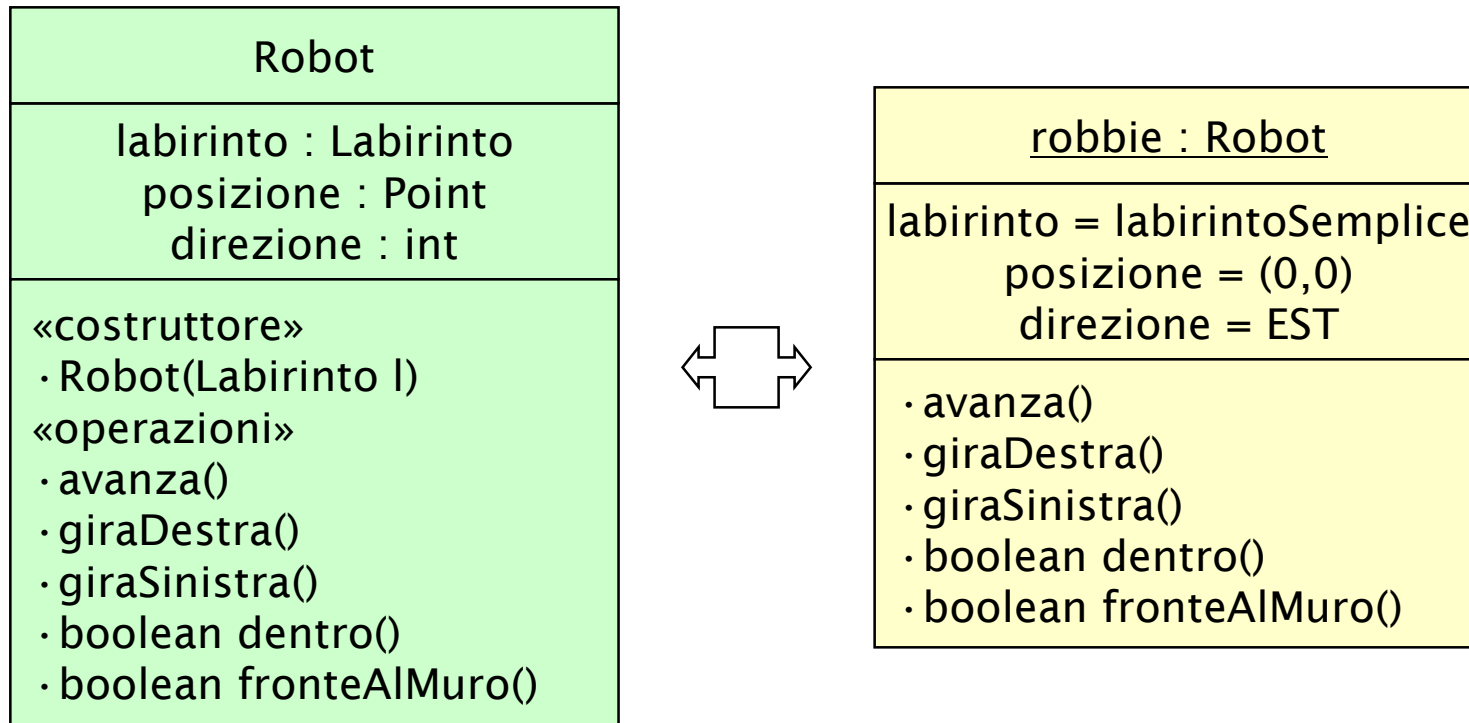
# La classe Robot



La definizione della classe **Robot** contiene

- la dichiarazione delle variabili per le proprietà di un robot
- la definizione dei metodi per le operazioni
- la definizione di un costruttore
  - il costruttore è elencato tra le operazioni, e ha lo stesso nome della classe

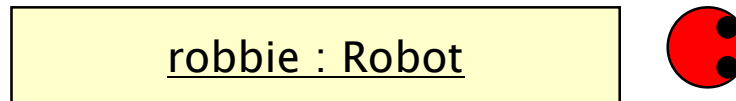
# Creazione di un oggetto Robot



- l'assenza della parola **static** indica che una proprietà (o operazione) è relativa alla modalità di costruzione dinamica
- al momento della creazione di una istanza, all'oggetto viene associata una nuova copia di ciascuna variabile (proprietà) e metodo (operazione)

# Il linguaggio UML

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



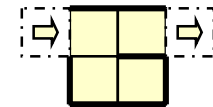
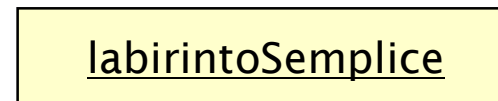
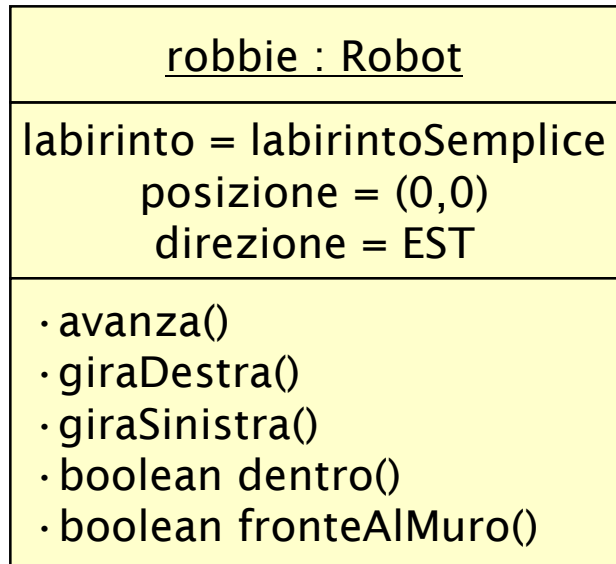
- **UML** è l'acronimo di **Unified Modeling Language**
- UML verrà introdotto in parte, in modo informale e graduale
- UML permette di descrivere solo alcune caratteristiche degli oggetti e delle classi
  - UML è un linguaggio per l'analisi e la progettazione orientata agli oggetti di sistemi software
  - UML non è un linguaggio di programmazione

Vengono ora descritte le tipologie di diagrammi che sono state già incontrate

- altre tipologie di diagrammi saranno introdotte nei successivi capitoli

# Diagrammi oggetti

Un **diagramma oggetti** descrive la struttura di uno o più oggetti

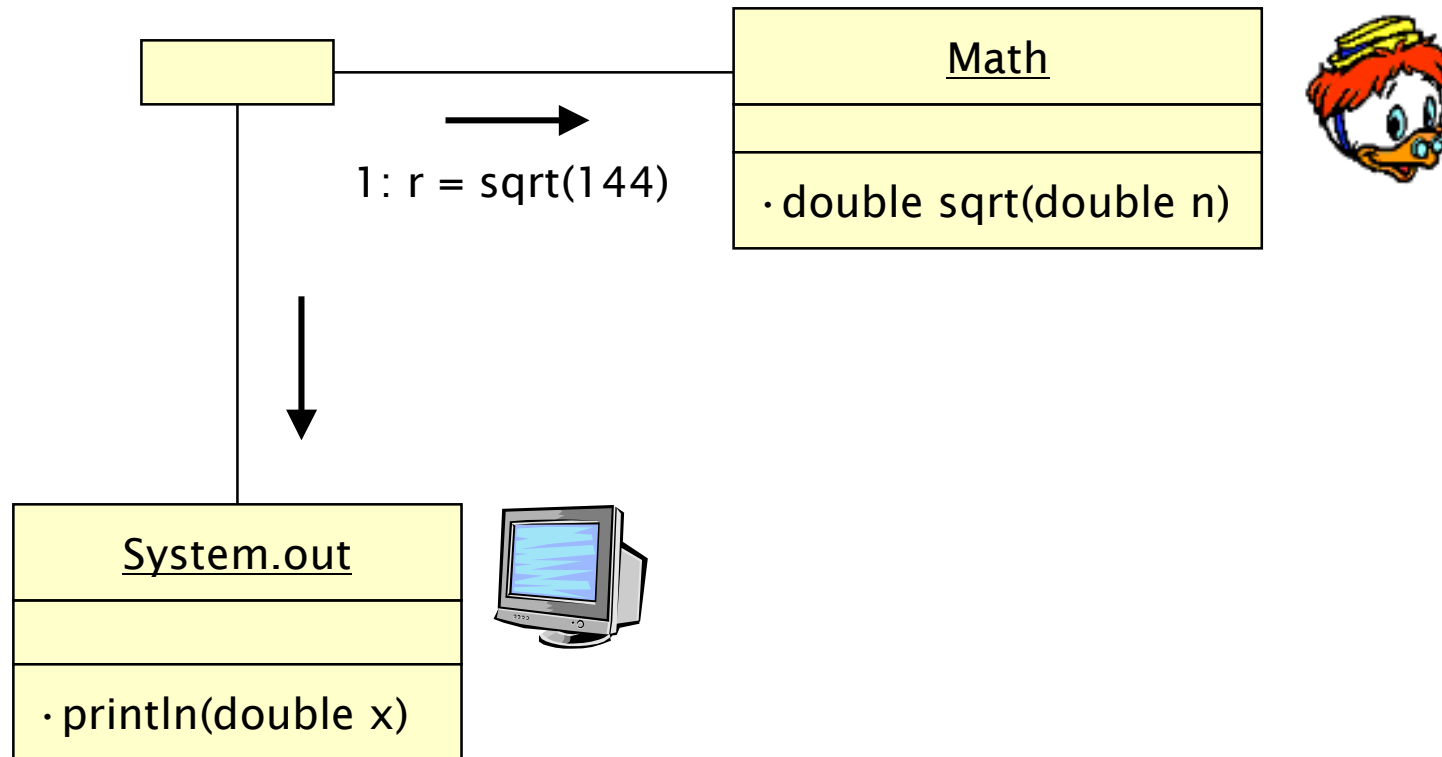


Un oggetto è rappresentato da un rettangolo diviso in tre parti

- la parte alta riporta il nome dell'oggetto
  - il nome è sottolineato, ed è eventualmente seguito dal nome della classe di cui l'oggetto è istanza
- la parte centrale riporta le proprietà dell'oggetto, con i relativi valori
- la parte bassa riporta le operazioni dell'oggetto

# Diagrammi collaborazione

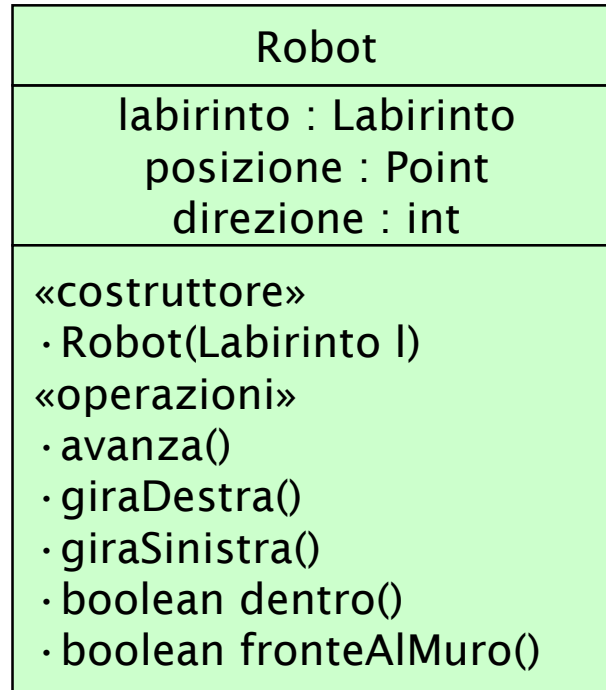
Un **diagramma collaborazione** mostra degli oggetti e una loro interazione basata sull'invio di messaggi



- i messaggi possono essere numerati, per mostrare l'ordine con cui vengono scambiati

# Diagrammi classi

Un **diagramma classi** mostra la struttura di una o più classi



Una classe è rappresentata da un rettangolo diviso in tre parti

- la parte alta riporta il nome della classe
- la parte centrale riporta le proprietà definite dalla classe, con i relativi tipi
- la parte bassa riporta le operazioni definite dalla classe, ed eventualmente i costruttori