Tecnologia ad oggetti

Concetti:

3. Conservazione dello stato
4. Identità degli oggetti
5. Messaggi

3. Conservazione dello stato

- Quando un modulo procedurale tradizionale (funzione o procedura) restituisce il controllo al proprio chiamante senza alcun effetto collaterale, il modulo cessa di esistere, lasciando solo il suo risultato come lascito.
- Quando lo stesso modulo viene nuovamente richiamato, è come se nascesse per la prima volta, perché non ha alcuna memoria di qualunque cosa gli sia accaduta nelle vite precedenti.

Conservazione dello stato: *esempio*

Invece ...

- Un oggetto è consapevole del suo passato: mantiene le informazioni al suo interno per un periodo di tempo indefinito.
- Per esempio, potrebbe accadere che il "chiamante" di un oggetto dia a quest'ultimo una particolare informazione e che poi lo stesso chiamante, o un altro, chieda all'oggetto di offrire nuovamente tale informazione.
- Allora, un oggetto non muore al termine della sua esecuzione, ma rimane in attesa di tornare nuovamente in azione.

```
var a1: Robot := Robot.New();  // creazione di un oggetto Robot
var p: Posizione := Posizione.New(); // creazione di una Posizione

a1.avanza(); // il robot cambia il suo stato e resta disponibile
p := a1.getPosizione(); // lo stato di p è uguale a quello della
                        // posizione del Robot
p.stampa();
```


Conservazione dello stato: *riassunto*

- Con un'espressione tecnica diciamo che un oggetto conserva il proprio stato (per *stato* si intende l'insieme dei valori detenuti internamente da un oggetto tramite i propri attributi).
- Per esempio, il robot conserva indefinitamente la conoscenza della posizione su cui si trova e della direzione in cui è rivolto.
- *Attenzione*: *ricordate che il modo in cui l'oggetto sceglie di conservare il proprio stato è solo un suo problema interno.*
- Vedremo in seguito che anche gli oggetti **nascono** (creazione) e **muoiono** (distruzione)

4. Identità degli oggetti: *definizione*

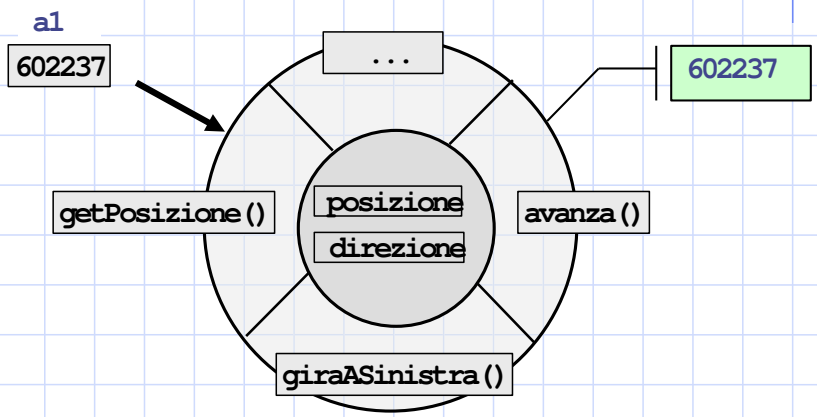
L'**identità degli oggetti** è la proprietà per cui ciascun oggetto (a prescindere dalla sua classe o dallo stato corrente) può essere identificato e trattato come una distinta entità software.

- Ogni oggetto ha una caratteristica unica che lo distingue dai suoi simili
- Tale caratteristica viene offerta dal meccanismo della maniglia *(handle)* dell'oggetto.
- La maniglia è nota formalmente come *identificatore di oggetto (OID, Object IDentifier)*
- La maggior parte degli ambienti a oggetti crea automaticamente questo OID.

Identità degli oggetti: *esempio*

```
var a1: Robot := Robot.New();  // creazione di un oggetto di
                               // tipo Robot
```

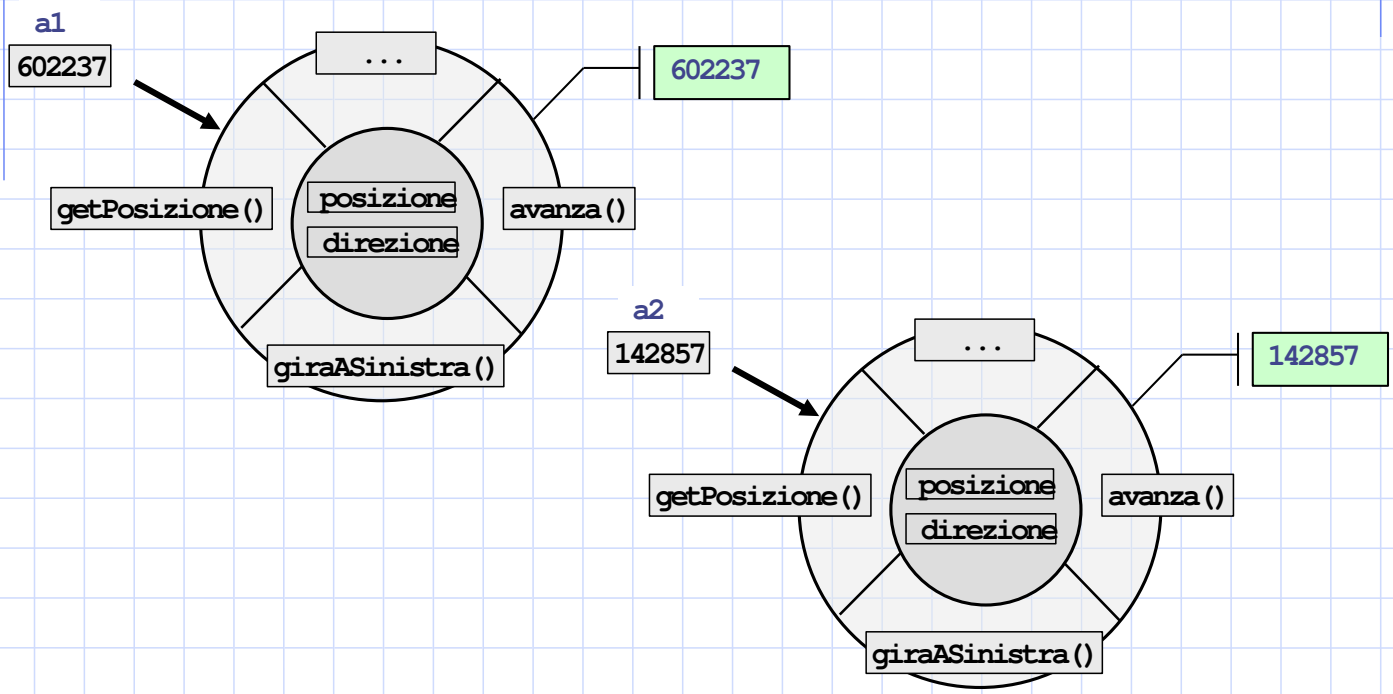
1. Un oggetto mantiene la stessa maniglia per tutta la sua vita



2. Due oggetti non possono avere la stessa maniglia

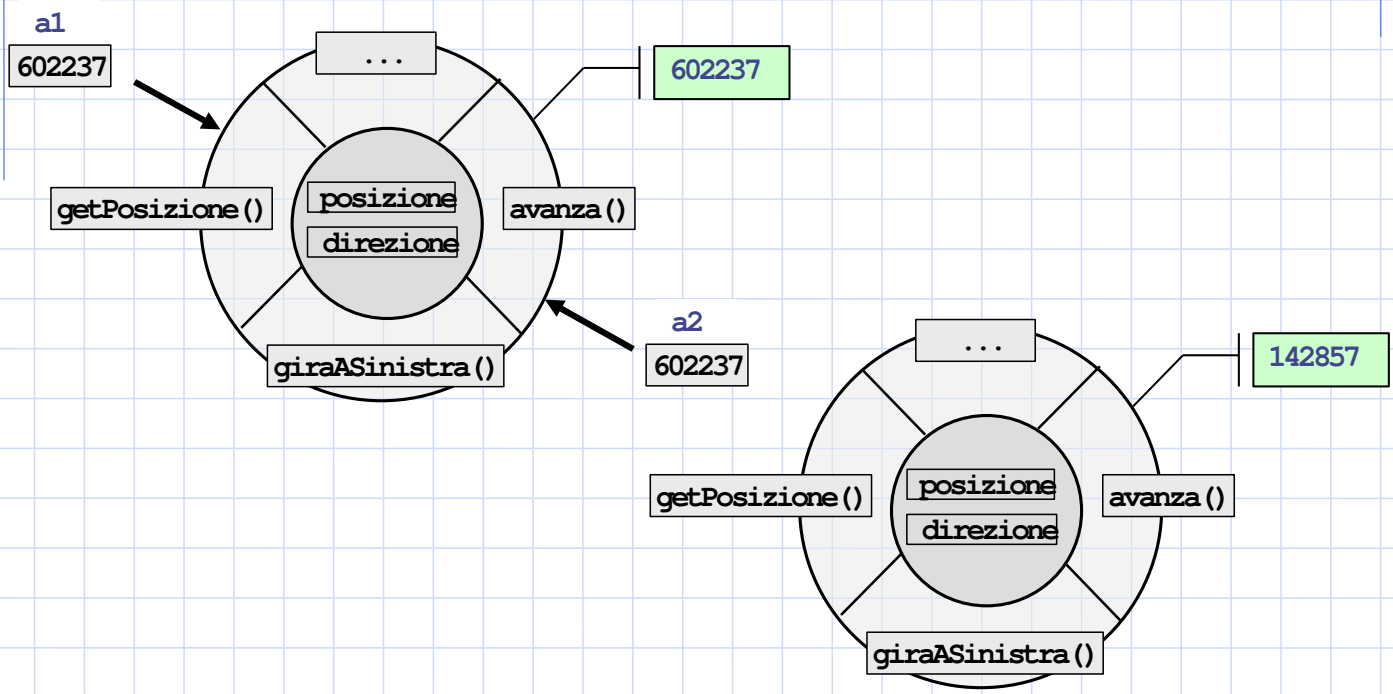
Identità degli oggetti: *esempio*

```
var a1: Robot := Robot.New();  // creazione di un oggetto di
                               // tipo Robot
var a2: Robot := Robot.New();  // creazione altro oggetto
```



Identità degli oggetti: *esempio*

```
a2 := a1;  // cosa accade con questa assegnazione??
```



5. Messaggi: *definizione*

Un **messaggio** è il veicolo tramite il quale un oggetto mittente **ogg1** recapita a un oggetto destinatario **ogg2** una richiesta affinché quest'ultimo applichi uno dei suoi metodi.

1. Un oggetto manda un messaggio ad un altro oggetto per chiedergli di svolgere una precisa attività.

2. Inoltre, un messaggio può anche trasferire informazioni da un oggetto a un altro.

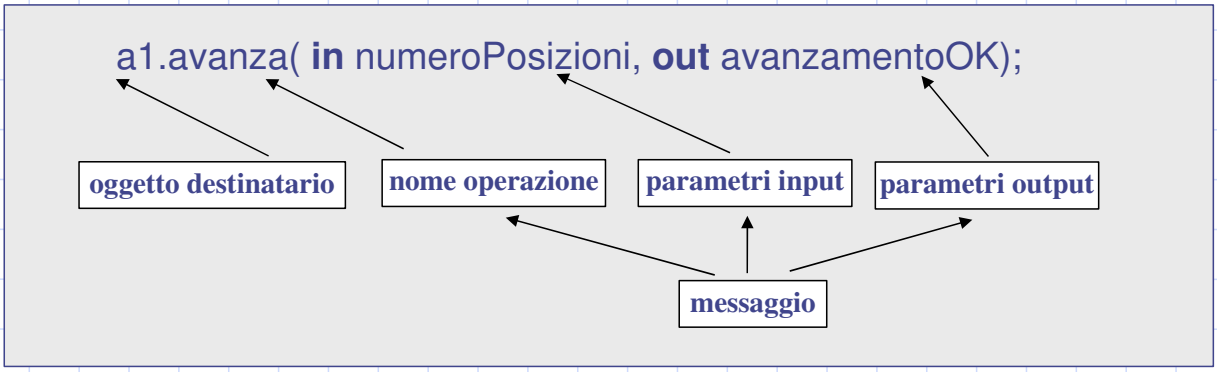
Messaggi: *conoscenze necessarie*

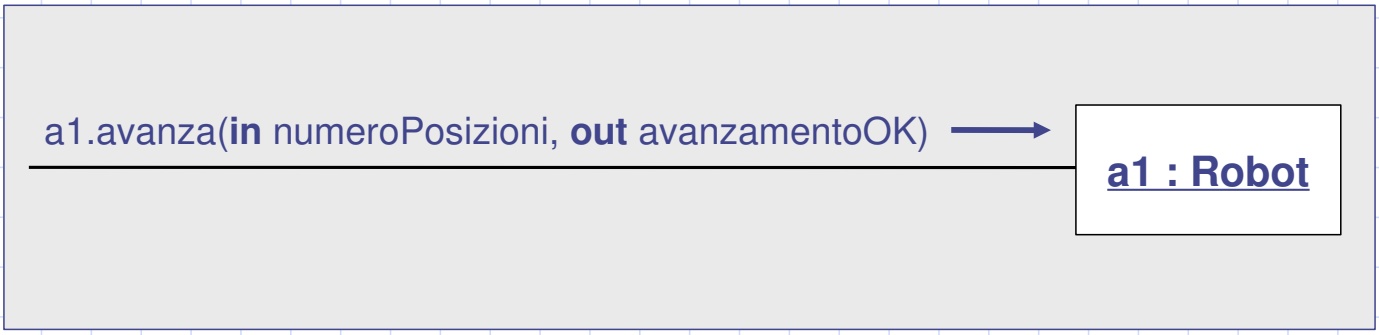
Affinché l'oggetto **ogg1** invii un messaggio sensato all'oggetto **ogg2**, l'oggetto **ogg1** deve conoscere tre cose:

- La maniglia di **ogg2** (ovviamente quando si invia un messaggio, si deve conoscere il destinatario).
- Il nome dell'operazione di **ogg2** che **ogg1** desidera eseguire
- Qualunque informazione supplementare (*parametro*) che sarà necessaria a **ogg2** per eseguire la sua operazione.

L'oggetto che invia il messaggio (**ogg1**) viene chiamato **mittente** o **sender** mentre l'oggetto che riceve il marcatore (**ogg2**) prende il nome di **destinatario** o **target.**

Messaggi: *esempio*



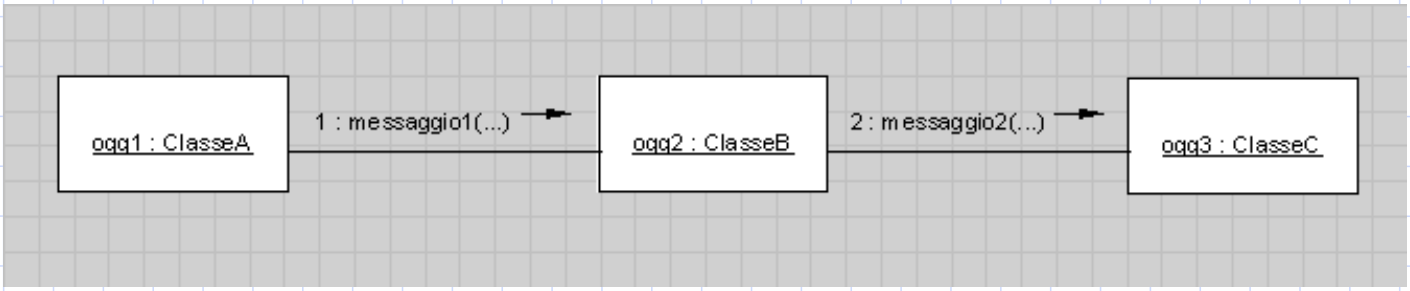


Messaggi: *ruoli degli oggetti*

In un messaggio, un oggetto può interpretare il ruolo di:

- mittente
- destinatario
- parametro

Rispetto a differenti messaggi, lo stesso oggetto può svolgere sia il ruolo di mittente che destinatario:



Messaggi: *tipi*

Un messaggio **informativo** fornisce al destinatario informazioni per aggiornarsi. E un messaggio "orientato al passato".

```
impiegato.sposato(dataMatrimonio: Data)
```

Un messaggio **interrogativo** richiede al destinatario di rivelare alcune informazioni su di sé. È un messaggio "orientato al presente".

```
a1.getPosizione()
```

Un messaggio **imperativo** richiede al destinatario di portare a termine qualche azione su se stesso, su un altro oggetto o sull'ambiente attorno al sistema.

```
a1.avanza()
```