Programmazione Orientata agli Oggetti in Linguaggio Java

Ereditarietà e Polimorfismo: Polimorfismo - c

Eredità del Tipo e Binding

versione 1.2

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Il Problema
- Cast Applicato a Riferimenti
- Ereditarietà del Tipo
- Algoritmo di Esecuzione
  - ⇨Binding Statico
  - ⇨Binding Dinamico

Il Problema

- Un passo indietro
  - ⇨riconsideriamo lo stile di programmazione che vorremmo adottare
- Esempio: gestione appuntamenti
  - ⇨stampa degli appuntamenti del giorno
- Nota
  - ⇨il codice dei metodi è semplificato rispetto a quello delle applicazioni sviluppate

Il Problema

```
public static void stampa (ArrayList lista) {
    for (int i = 0; i < lista.size(); i++) {
        Impegno impegno = (Impegno)lista.get(i);
        System.out.println(impegno.toString());
    }
}
```

L'output atteso:
- riunione con Marco
- lezione di Prog. Oggetti
- riunione della CIP

gli oggetti prelevati dalla lista sono trattati come oggetti di tipo "Impegno" indifferentemente dalle differenze di tipo

a ciascun oggetto viene chiesto di eseguire il metodo toString() nel modo più appropriato per il tipo di impegno

Il Problema

- Un passo avanti nella soluzione
  - ⇨sappiamo che è possibile definire Impegno come nuovo tipo (tipo = interfaccia)
- Intuitivamente
  - ⇨vengono prelevati dalla lista riferimenti ai vari impegni (riferimenti di tipo Object)
  - ⇨viene effettuato il cast di ciascun impegno sul tipo comune definito dall'interfaccia Impegno
  - ⇨ciascun tipo di impegno esegue opportunamente il metodo toString()

Il Problema

- In dettaglio, però
  - ⇨ci sono due questioni non completamente chiare
- Questione n. 1
  - ⇨qual è la semantica del cast ?
- Questione n. 2
  - ⇨qual è l'algoritmo secondo cui viene eseguito il metodo toString() ?

Cast Applicato a Riferimenti

- L'operatore di cast
  - ⇨operatore che cambia il tipo di un valore
  - ⇨semantica nota nel caso dei tipi di base
- Cast applicato ad un riferimento
  - ⇨il cast in realtà coinvolge l'oggetto
  - ⇨intuitivamente sto chiedendo all'oggetto relativo di "cambiare tipo"
- Quanti tipi ha un oggetto ?

Cast Applicato a Riferimenti

- Un esempio
  - ⇨consideriamo una riunione:
    Riunione r = new Riunione();
  - ⇨il cui riferimento viene inserito nella lista:
    lista.add((Object)r); // il cast è implicito
  - ⇨successivamente viene recuperato:
    Object o = lista.get(i);
  - ⇨e trattato come oggetto di tipo Impegno
    Impegno impegno = (Impegno)o;

Cast Applicato a Riferimenti

- Durante queste operazioni
  - ⇨vengono effettuati ripetute operazioni di cast
  - ⇨da Riunione a Object
  - ⇨da Object a Impegno
- Il senso di queste operazioni
  - ⇨manipolare l'oggetto con riferimenti di tipo diverso (r, o, impegno)
  - ⇨ognuno dei quali richiede un oggetto del tipo opportuno

Ereditarietà del Tipo

- In effetti
  - ⇨i riferimenti sono strettamente tipati
  - ⇨ma all'interno della gerarchia c'è un fenomeno di "ereditarietà del tipo"
  - ⇨per cui un oggetto può assumere "tipi diversi" e quindi essere compatibile con rif. diversi
- La semantica dell'ereditarietà di tipo
  - ⇨anche in questo caso è spiegabile sulla base dell'associazione gerarchica di oggetti

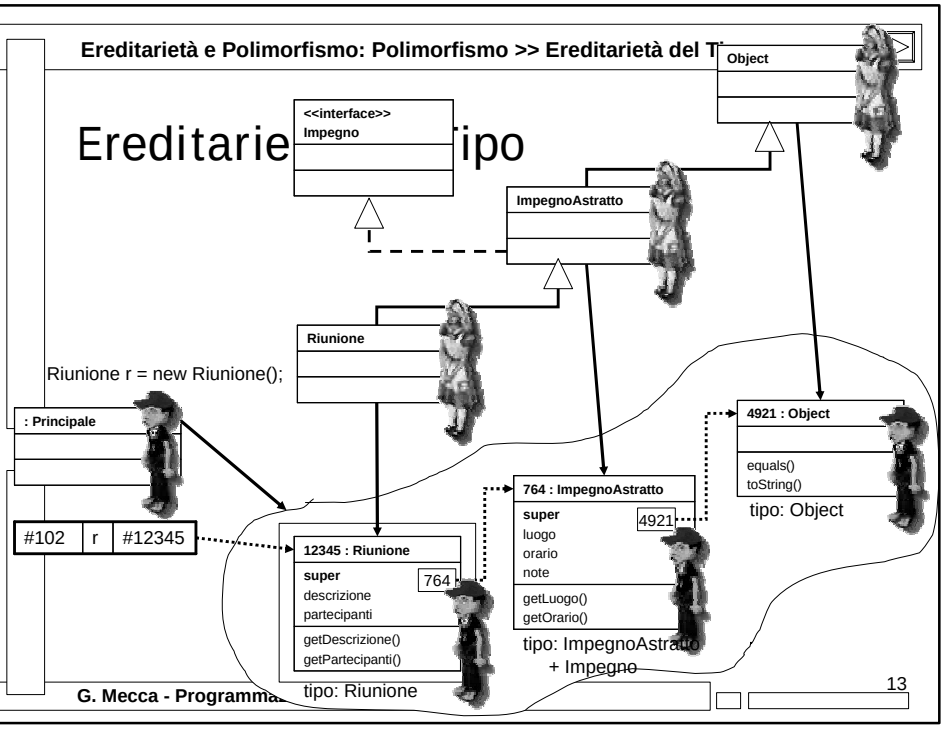
Ereditarietà del Tipo

- In particolare
  - ciascun oggetto della gerarchia è compatibile con uno o più tipi
  - collaborando, l'associazione può "assumere" ciascuno di questi tipi
  - e quindi proporsi come un "superoggetto" compatibile con riferimenti diversi

Ereditarietà del Tipo

- Per cominciare
  - alcune regole elementari sul tipo di un oggetto
- I tipi per gli oggetti
  - ogni oggetto ha il tipo della propria classe (ne ha l'interfaccia)
  - se una classe implementa un'interfaccia, gli oggetti di quella classe hanno il tipo definito dall'interfaccia (interface)

Ereditarie ipo



Ereditarietà del Tipo

ATTENZIONE
all'intuizione dietro il polimorfismo

- Comportamento polimorfo
  - ⇨l'associazione è in grado di presentarsi con "facce diverse", ovvero con tipi diversi
  - ⇨ovvero in modo "polimorfo" (molte forme)
  - ⇨questo avviene esponendo, un "oggetto di turno" diverso a seconda del tipo richiesto
- Oggetto di turno
  - ⇨oggetto incaricato di ricevere i messaggi da parte di un certo riferimento

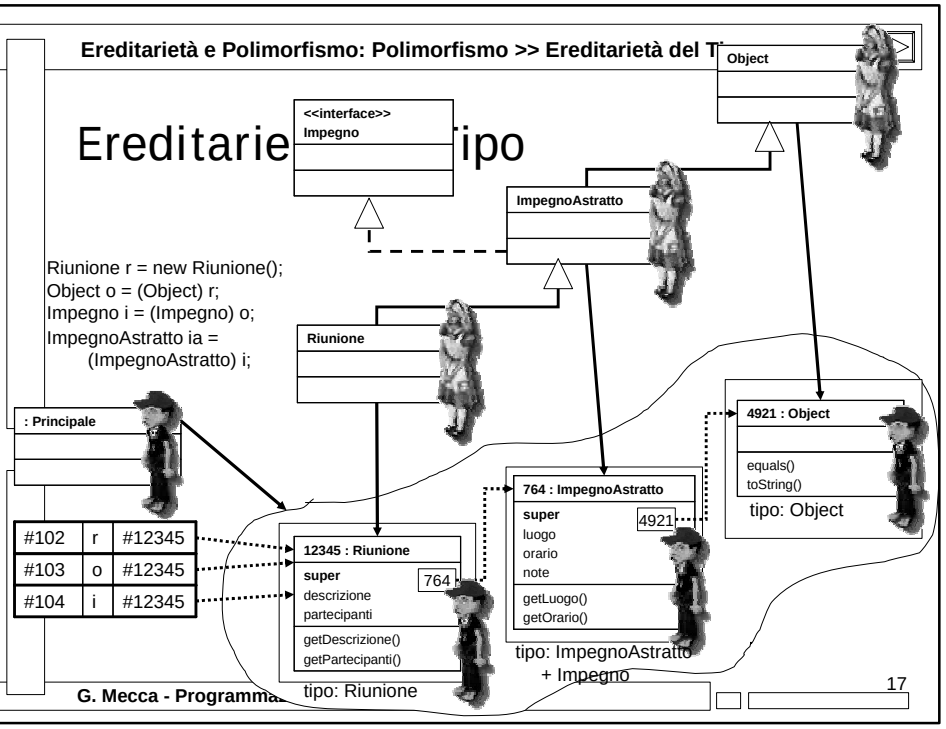
Ereditarietà del Tipo

- ATTENZIONE
  - ⇨ci sono quindi due oggetti importanti nella gerarchia
- L'oggetto puntato
  - ⇨quello il cui OID è mantenuto nel riferimento (l'oggetto la cui creazione ha dato luogo all'associazione)
- L'oggetto di turno
  - ⇨l'oggetto del tipo richiesto per il riferimento

Ereditarietà del Tipo

- Semantica del cast
  - ⇨il cast si applica ad un oggetto (attraverso un riferimento) per renderlo compatibile con un riferimento di tipo diverso
  - ⇨produce un cambio di turno nell'associazione gerarchica; l'oggetto di turno diventa quello del tipo compatibile con il tipo richiesto
  - ⇨da quel momento, l'oggetto di turno intercetta le chiamate inviate attraverso il nuovo riferimento

Ereditarie ipo



Ereditarietà del Tipo

- Metafora
  - ⇨l'oggetto di turno per un riferimento è incaricato di fare la guardia
  - ⇨sorveglia ed intercetta gli eventuali messaggi inviati verso l'associazione da parte del riferimento
  - ⇨cambiando la guardia, l'associazione "inganna" i riferimenti proponendosi di volta in volta con "facce" diverse

Ereditarietà del Tipo

- In questo modo
  - ⇨viene rispettato il tipo del riferimento
  - ⇨ciascun riferimento “vede” un oggetto di turno diverso, il cui tipo (interfaccia) coincide con quello previsto
- Dall’esterno
  - ⇨è possibile dire che l’oggetto originale di tipo Riunione è in grado di cambiare il suo tipo

Ereditarietà del Tipo

- Radiografia dei cambi di turno
  - ⇨passo 1: creo un oggetto di tipo Riunione
    Riunione r = new Riunione(); // OID #12345
    l’oggetto di turno per r è di tipo Riunione (coincide con l’oggetto puntato)
  - ⇨passo 2: lo aggiungo alla lista degli impegni
    lista.add(r);
    viene fatto un cast automatico a Object; l’oggetto di turno per il riferimento nella lista è di tipo Object; l’oggetto puntato resta 12345

Ereditarietà del Tipo

- Radiografia dei cambi di turno (continua)
  - passo 3: lo estraggo dalla lista
    Object o = lista.get(i);
    l'oggetto puntato è #12345; l'oggetto di turno per o è Object
  - il riferimento subisce un cast a Impegno;
    Impegno i = (Impegno)o;
    l'oggetto di turno per i è di tipo ImpegnoAstratto; l'oggetto puntato resta #12345

Ereditarietà del Tipo

- Meccanicamente
  - possiamo riassumere il fenomeno dicendo che ciascun oggetto di una gerarchia "eredita" tutti i tipi dei suoi superoggetti
  - ed è quindi possibile applicare all'oggetto il cast su qualunque di questi tipi
  - questo consente di manipolare l'oggetto attraverso riferimenti di tipi diversi

Algoritmo di Esecuzione

- A questo punto
  - ⇨resta da risolvere solo il secondo problema
- Ovvero
  - ⇨descrivere in dettaglio la semantica della chiamata al metodo toString()
  - ⇨in particolare, ci interessa la chiamata:

```
Impegno impegno = (Impegno)lista.get(i);
System.out.println(impegno.toString());
```


Algoritmo di Esecuzione

- Sulla base di quanto detto
  - ⇨sia che l'oggetto originale sia una Lezione, sia che sia un Riunione, il riferimento impegno vede un oggetto di turno di tipo ImpegnoAstratto
- L'algoritmo visto in precedenza
  - ⇨prevede che la chiamata esegua toString() di ImpegnoAstratto

Algoritmo di Esecuzione

- Ma...
  - ⇨evidentemente non è così
  - ⇨infatti il metodo in questione è astratto e non potrebbe essere eseguito comunque
- Di conseguenza
  - ⇨l'algoritmo descritto è incompleto
  - ⇨è necessario descrivere in maggior dettaglio l'esecuzione dei metodi come toString()

Algoritmo di Esecuzione

- Il problema dell'"overriding"
  - ⇨l'algoritmo per rispondere alle chiamate di metodi ridefiniti è decisamente più complesso
  - ⇨ed è basato sul concetto di "binding"
- Il concetto di "binding"
  - ⇨tecnica per scegliere ("legare") quale eseguire tra le versioni di un metodo ridefinito lungo la gerarchia

Algoritmo di Esecuzione

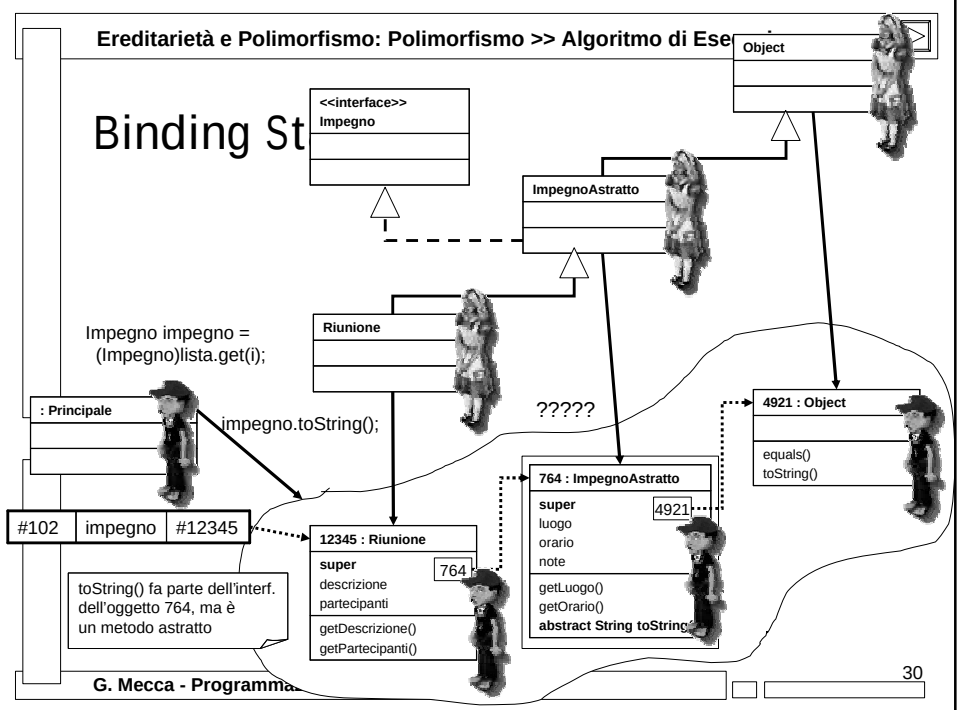
- Due tipi di tecniche di binding
- Binding statico
  - ⇨la chiamata viene eseguita prelevando la versione del metodo contenuta nella classe dell'oggetto di turno
- Binding dinamico
  - ⇨la chiamata viene eseguita affidandosi all'oggetto puntato (prelevando la versione più "profonda" del metodo tra le disponibili)

Binding Statico

- La tecnica di binding statico
  - ⇨coincide con l'algoritmo già visto
- Algoritmo di binding statico
  - ⇨il riferimento invia il messaggio all'oggetto di turno
  - ⇨se il metodo appartiene alla sua interfaccia, l'oggetto esegue il metodo
  - ⇨altrimenti, usa "super" per chiedere di eseguire il metodo all'oggetto padre
  - ⇨il processo può ripetersi

Binding Statico

- Caratteristiche dell'algoritmo
  - ⇨consente l'ereditarietà delle caratteristiche
  - ⇨le chiamate vengono inoltrate su per la gerarchia
  - ⇨non vengono mai inoltrate giù per la gerarchia
- Con questo algoritmo
  - ⇨il codice visto non potrebbe funzionare



Binding Dinamico

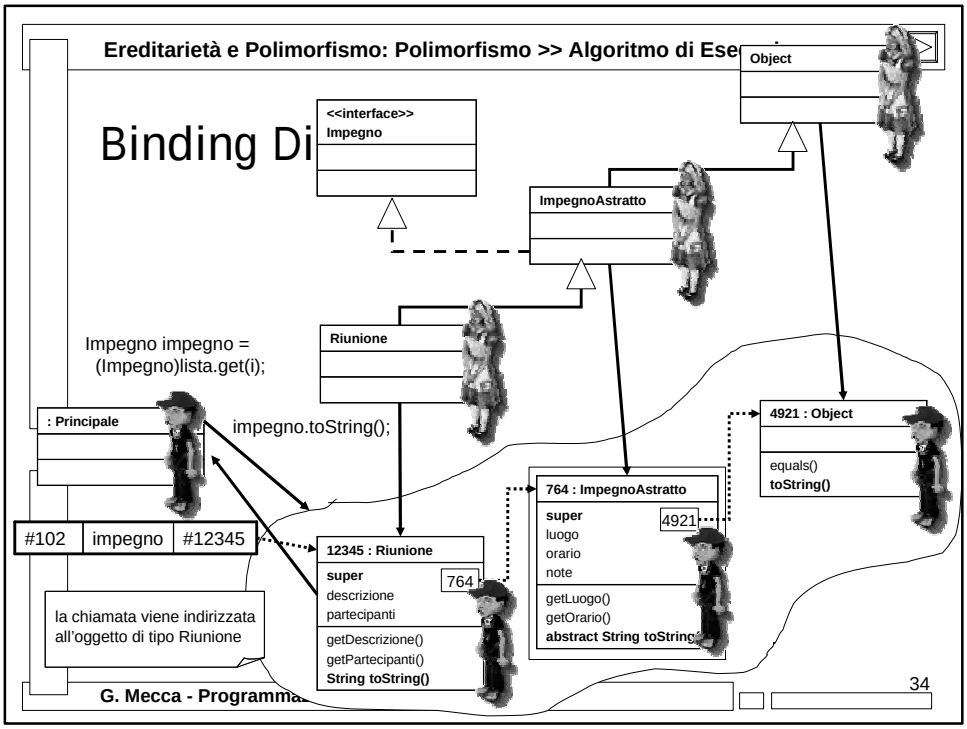
- La tecnica di binding dinamico
  - ⇨consente al codice visto di funzionare
- Algoritmo di binding dinamico
  - ⇨il riferimento invia il messaggio all'oggetto di turno
  - ⇨se il metodo richiesto non è ridefinito lungo la gerarchia, l'algoritmo procede come nel caso precedente

Binding Dinamico

- Algoritmo di binding dinamico (continua)
  - ⇨se il metodo è stato ridefinito lungo la gerarchia, interviene la macchina virtuale
  - ⇨la macchina virtuale inoltra il messaggio all'oggetto puntato
  - ⇨se l'oggetto può, esegue il metodo
  - ⇨altrimenti si rivolge al suo "super" e la procedura si ripete

Binding Dinamico

- La differenza
  - ⇨se il metodo è ridefinito lungo la gerarchia, ne viene sempre eseguita la versione “più in basso”
- Conseguenza
  - ⇨se l’oggetto in basso è di tipo Riunione viene eseguito il toString() di Riunione
  - ⇨se è di tipo Lezione, viene eseguito il toString() di Lezione



Binding Dinamico

ATTENZIONE
per i metodi soggetti a "override" viene applicato il binding dinamico

- Nei linguaggi a oggetti moderni
  - ⇨se il metodo è ridefinito, viene sempre applicato l'algoritmo di binding dinamico
  - ⇨altrimenti, l'algoritmo di binding statico
  - ⇨questo vale sia in Java che in C#
- A questo punto
  - ⇨siamo in grado di interpretare i due esempi di codice precedentemente visti

Binding Dinamico

```
public static void stampa (ArrayList lista) {
   for (int i = 0; i < lista.size(); i++) {
      Impegno impegno = (Impegno)lista.get(i);
      System.out.println(impegno.toString());
   }
}
```

Supponiamo che nella lista siano stati inseriti
- un oggetto di tipo Riunione
- un oggetto di tipo Lezione
- un altro oggetto di tipo Riunione

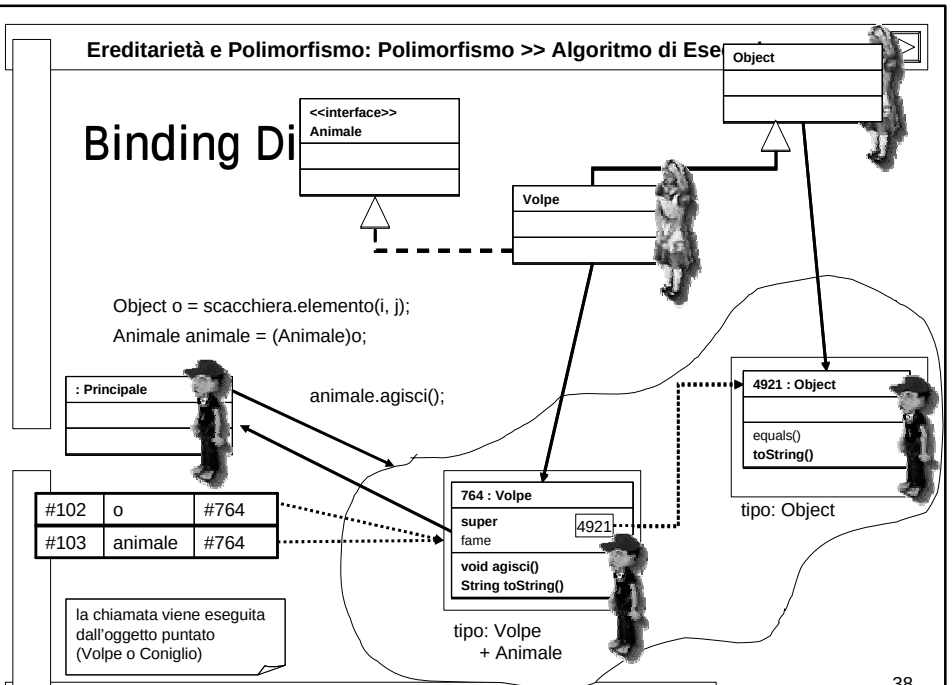
L'output ottenuto è del tipo:

```
8:30   -- riunione con Marco – luogo: ufficio – partecipanti: ... – note: ...
10:00 -- lezione di Prog. Oggetti – aula: Aula 1 – note: ...
15:00 -- riunione della CIP – luogo: sala riunioni – partecipanti: ... – note: ...
```

Binding Dinamico

```
public void simula(Scacchiera scacchiera) {
    for (int i = 0; i < scacchiera.getRighe(); i++) {
        for (int j = 0; j < scacchiera.getColonne(); j++) {
            Animale animale = (Animale)scacchiera.getElemento(i, j);
            if (animale != null) {
                animale.agisci(scacchiera);
            }
        }
    }
}
```

In questo caso:
- gli animali che sono Volpi agiscono andando a caccia
- gli animali che sono Conigli agiscono correndo



Binding Dinamico

- Attenzione
  - ⇨il binding dinamico ha dei costi in termini di esecuzione
- Infatti, nel caso di binding statico
  - ⇨sulla base del tipo del riferimento il compilatore può decidere staticamente quale versione del metodo verrà eseguita e fare le ottimizzazioni del caso
  - ⇨non è necessario attendere l'esecuzione

Binding Dinamico

- Nel caso del binding dinamico, viceversa
  - ⇨la versione del metodo non dipende dal tipo del riferimento, ma dell'associazione di oggetti lungo la gerarchia
  - ⇨viene decisa solo a tempo di esecuzione e richiede l'intervento della macchina virtuale
- Infatti
  - ⇨il binding dinamico si chiama anche "late binding"

Riassumendo

- Il Problema
- Cast Applicato a Riferimenti
- Ereditarietà del Tipo
- Algoritmo di Esecuzione (“Binding”)
  - ⇨Binding Statico
  - ⇨Binding Dinamico

Termini della Licenza