Ereditarietà

Ereditarietà:

- Problema
- Definizione
- Esempi

Ereditarietà: *il problema*

Problema:

Cosa fare nel caso sia necessario progettare una classe **B** se, allo stesso tempo, fosse disponibile una classe **A** quasi identica a **B** tranne per la mancanza di alcuni attributi e operazioni?

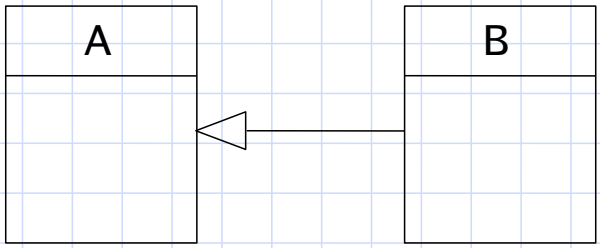
- Soluzione banale: duplicare attributi e operazioni di **A** per collocarli in **B**. Difetti:
  1. lavoro aggiuntivo di duplicazione
  2. lavoro di manutenzione

- Soluzione migliore: far sì che la classe **B** "chieda di utilizzare le operazioni" della classe **A**. Questa soluzione prende il nome di **ereditarietà**.

Ereditarietà: *definizione*

- L'**ereditarietà** (di una classe **B** da una classe **A**) è il meccanismo tramite il quale **B** ha implicitamente definito su se stessa ciascuno degli attributi e delle operazioni della classe **A** come se tali attributi e operazioni fossero stati definiti per **A** stessa.

- **A** è **superclasse** di **B**, mentre **B** è una **sottoclasse** di **A**



Ereditarietà: *modo d'uso*

L'ereditarietà rappresenta un'altra delle caratteristiche principali grazie alle quali la tecnologia a oggetti si distacca dagli approcci dei sistemi tradizionali:

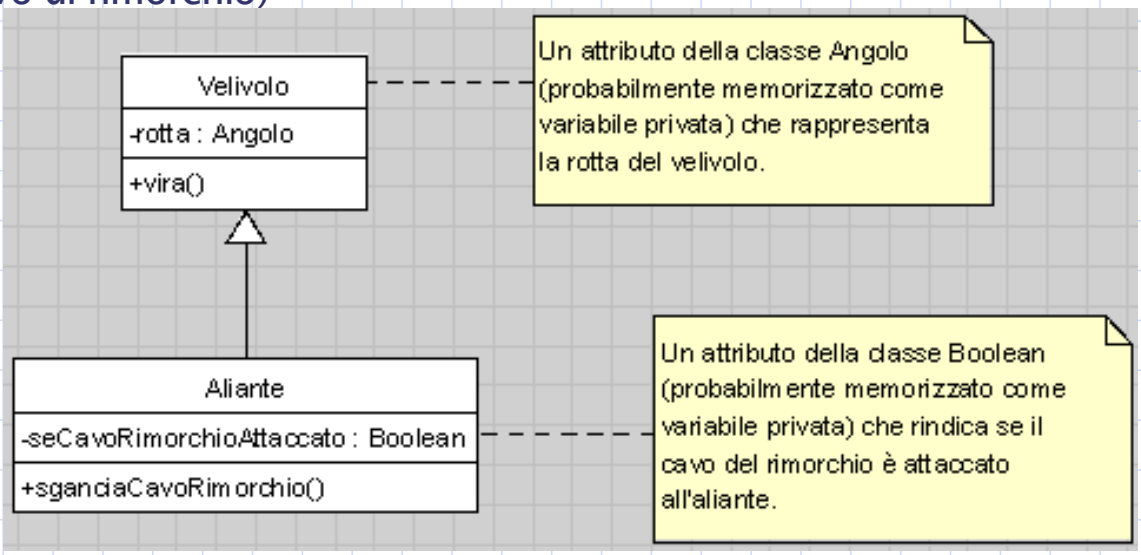
permette effettivamente di costruire il software in modo incrementale distinguendo due fasi fondamentali:

1. Si costruiscono le classi destinate a far fronte alle situazioni più generali.
2. Per affrontare i casi particolari, si aggiungono classi più specializzate che ereditano dalle classi generali.

   Una classe specializzata avrà pertanto il diritto di utilizzare tutte le operazioni e gli attributi (operazioni e attributi sia di classe sia di istanza) della classe originale

Ereditarietà: *esempio*

- La classe Velivolo modella attività e informazioni relative a qualunque tipo di apparecchio volante
- Esistono però tipi speciali di velivolo che svolgono speciali attività e pertanto richiedono informazioni particolari.
- Un aliante svolge attività speciali (sganciare il cavo di rimorchio) e potrebbe dover registrare speciali informazioni (essere attaccato a un cavo di rimorchio)

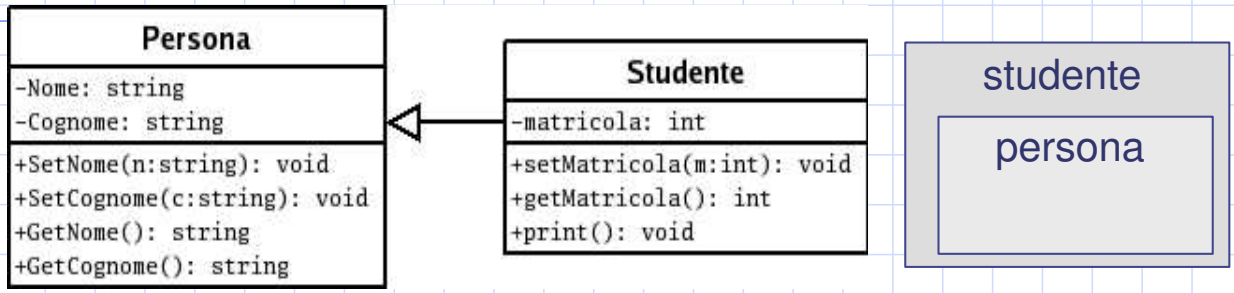


Ereditarietà: *esempio di codice*

```
var v: Velivolo := Velivolo.New();
var a: Aliante := Aliante.New();
v.vira(nuovaRotta, out viraOK);    (1)
a.sganciaCavoRimorchio();          (2)
a.vira(nuovaRotta, out viraOK);    (3)
v.sganciaCavoRimorchio();          (4)
```

1) L'oggetto **v** utilizza l'operazione **vira()** che è stata definita nella classe **Velivolo**
2) a utilizza l'operazione **sganciaCavoRimorchio()** che è stata definita nella classe **Aliante**
3) dato che **Velivolo** è superclasse di **Aliante**, l'oggetto **a** può utilizzare qualunque operazione di **Velivolo** e dunque anche **vira()**
4) *non funziona!* Nessuna operazione **sganciaCavoRimorchio()** è definita per **Velivolo**

Ereditarietà: *principi fondamentali*

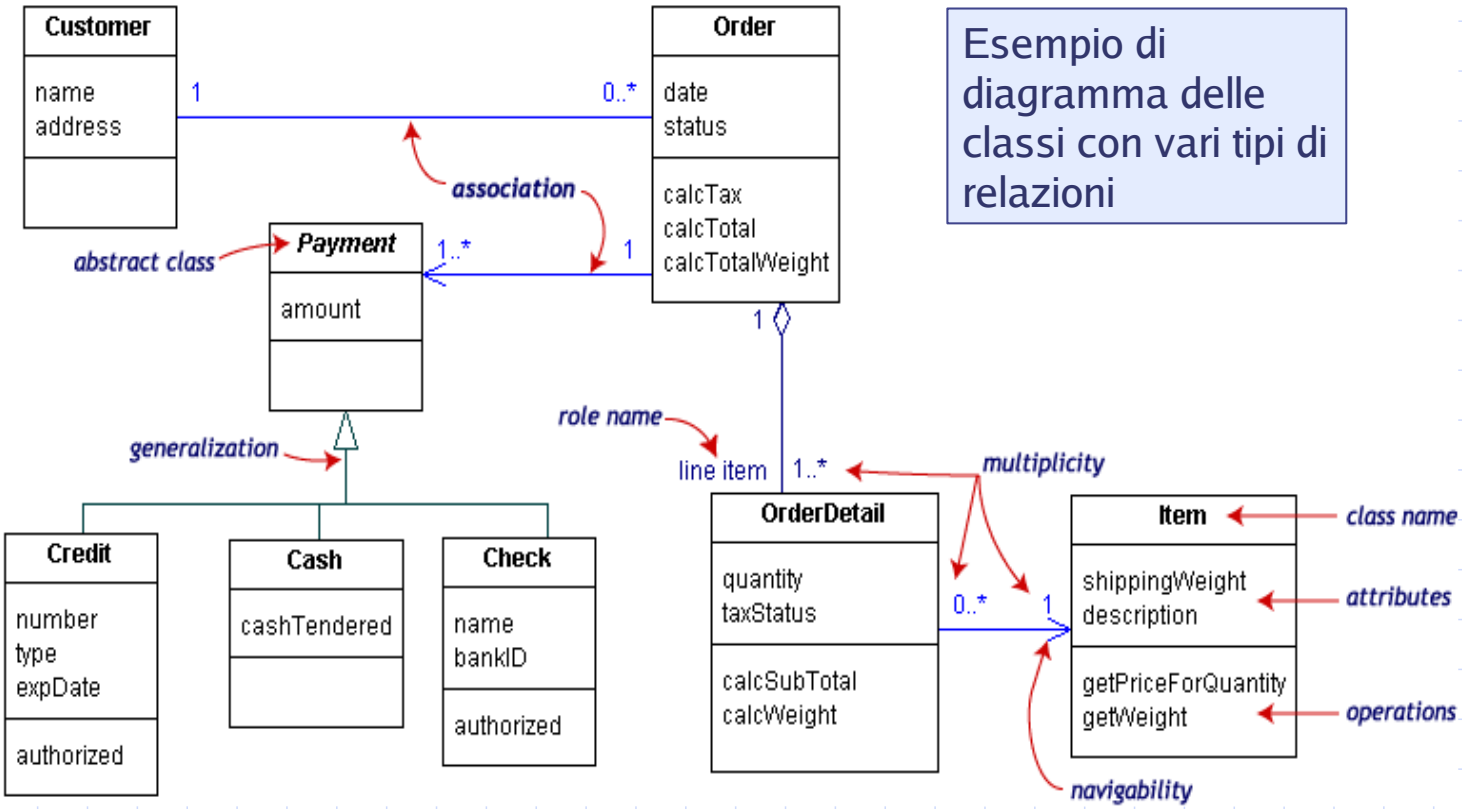



```
var p: Persona := Persona.New("Maria","Bianchi");
var s: Studente := Studente.New("Leo","Neri",12345);
p := s; // corretto!!
```

L'ereditarietà permette a un singolo oggetto di essere contemporaneamente un'istanza di più di una classe:

- Ogni oggetto della classe Studente è un *(is-a)* oggetto della classe Persona
- È possibile usare un oggetto della classe Studente in ogni punto in cui è atteso un oggetto della classe Persona
- La classe Persona contiene più elementi della classe Studente

Esempio



Esempio di diagramma delle classi con vari tipi di relazioni

Ereditarietà multipla

- Esiste anche il concetto di **ereditarietà multipla**. In tal caso ogni classe può avere un numero arbitrario di superclassi dirette.
- L'ereditarietà multipla introduce alcune difficoltà di progettazione (possibilità di ereditare operazioni o attributi in contrasto tra loro)

- Consentita da C++ ed Eiffel
- Non consentita da Java e Smalltalk