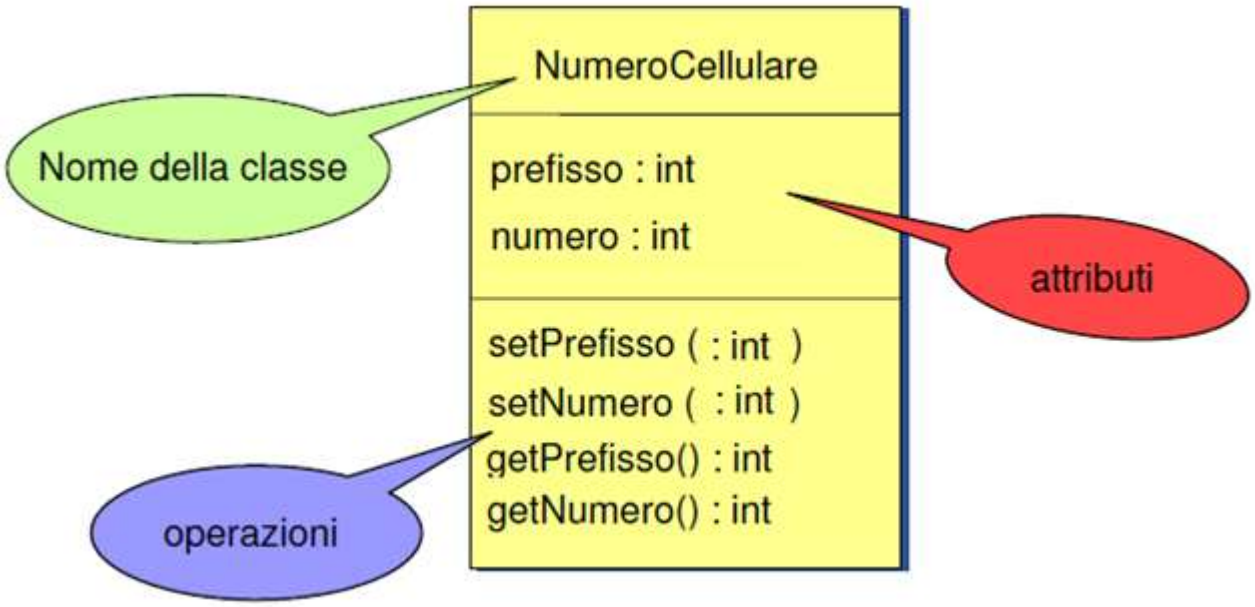
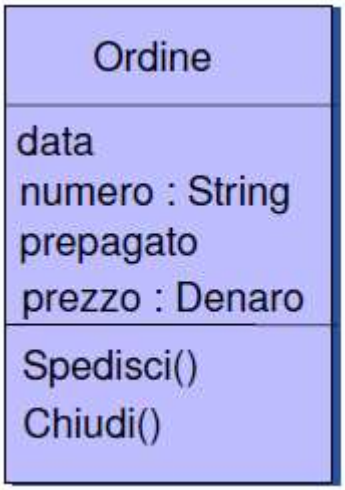
Implementare le classi descritte dai seguenti diagrammi UML
ed evidenziare le opportune modifiche per buono stile

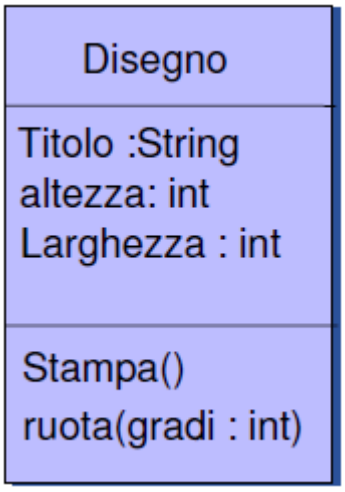
Esercizio 1.UML)



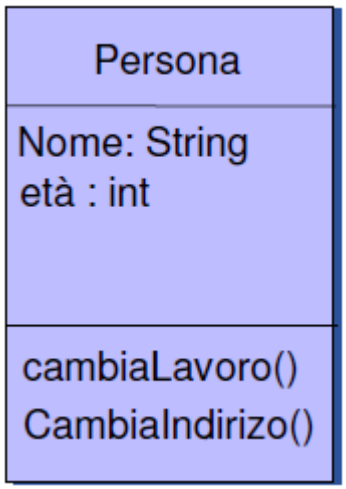
Esercizio 2.UML)



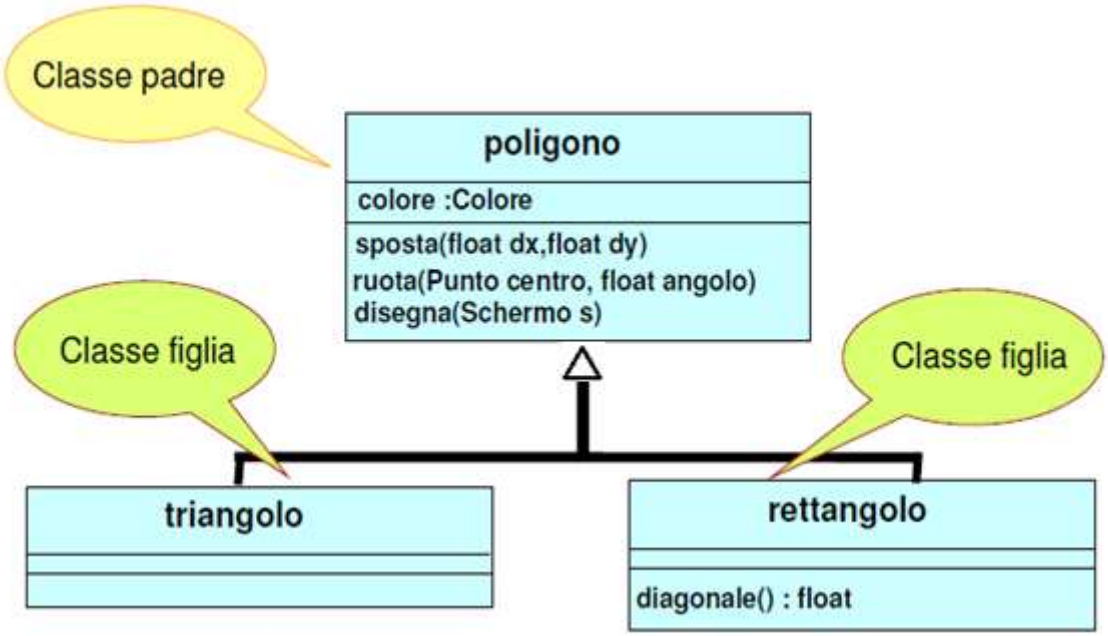
Esercizio 3.UML)



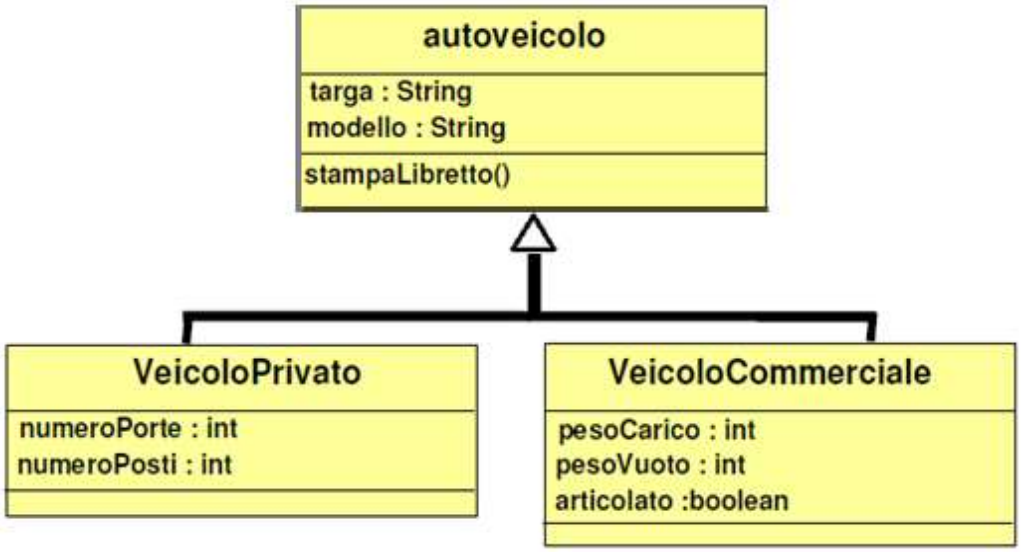
Esercizio 4.UML)



Esercizio 5.UML)



Esercizio 6.UML)



Appendice

UML: un class diagram rappresenta uno schema concettuale

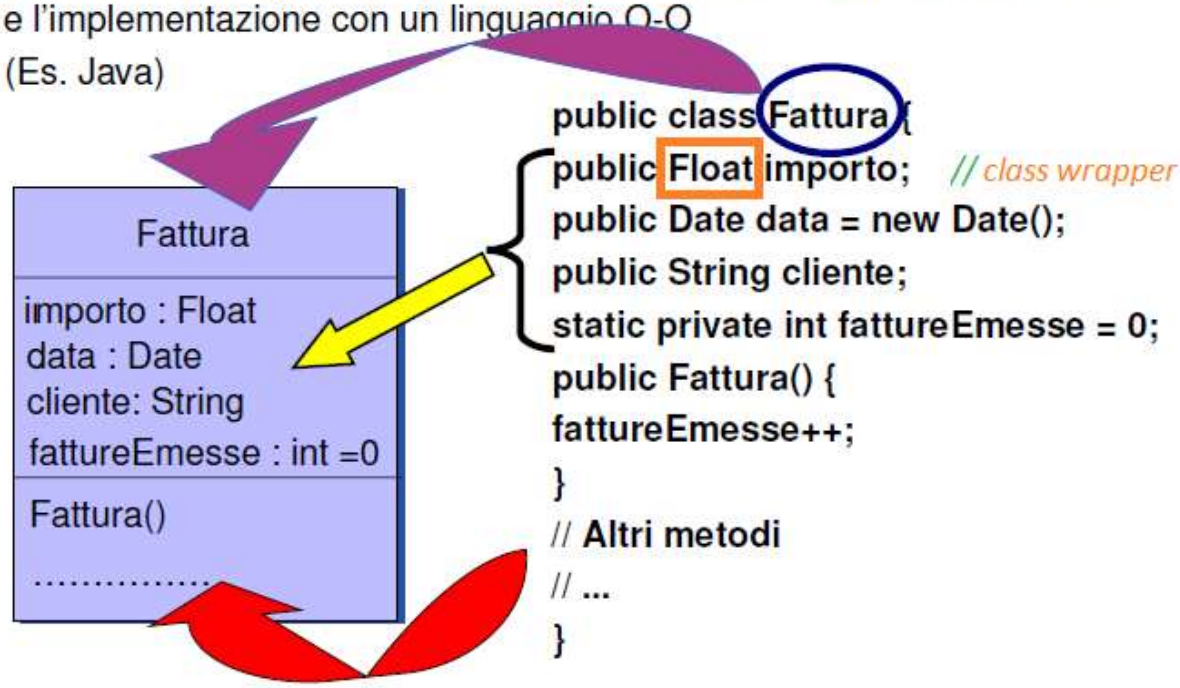
La **prospettiva** con cui si realizza il diagramma può essere

– **concettuale**
- studia i concetti propri del dominio sotto studio, senza preoccuparsi della loro successiva implementazione

– **di specifica**
- studia il software ma a livello di interfaccia e non di implementazione. Quindi l'attenzione è concentrata sulle responsabilità delle classi ma non sui dettagli concreti

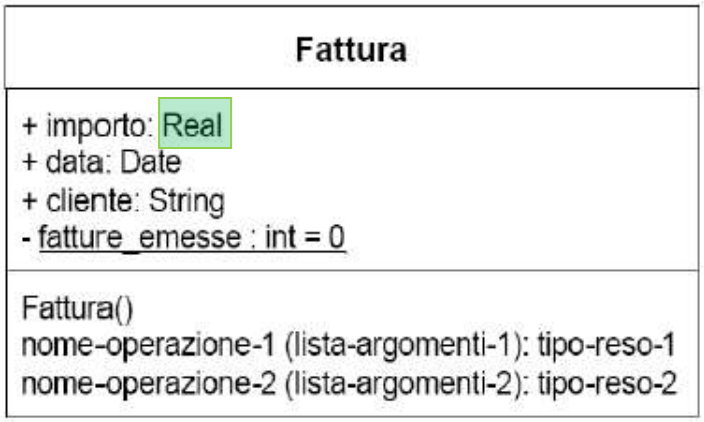
– **implementativa**
- il diagramma fa riferimento alle classi effettivamente realizzate con un linguaggio di programmazione OO e alle strutture dati effettivamente impiegate.

Esiste una corrispondenza tra la rappresentazione UML di una classe e l'implementazione con un linguaggio O-O (Es. Java)



classi wrapper (slides)

Esempio: class Fattura senza uso di classi wrapper




```
public class Fattura {
public float importo;
public Date data = new Date();
public String cliente;
static private int
    fatture_emesse = 0;
public Fattura() {
fatture_emesse++;
}
// Altri metodi
// ...
```

Visibilità:

- **Pubblica (+)**: l'attributo è accessibile da qualsiasi altro oggetto dotato di riferimento all'oggetto che contiene l'attributo in questione;
- **Privata (-)**: l'attributo è accessibile solo all'interno della classe di appartenenza (dichiarante);
- **Protetta (#)**: l'attributo è accessibile da tutte le istanze delle classi che "ereditano" da quella in cui l'attributo è definito;
- **Package (~)**: l'attributo è accessibile da qualsiasi altro oggetto istanza di classi appartenenti allo stesso package o in un altro ad esso annidato a qualsiasi livello.

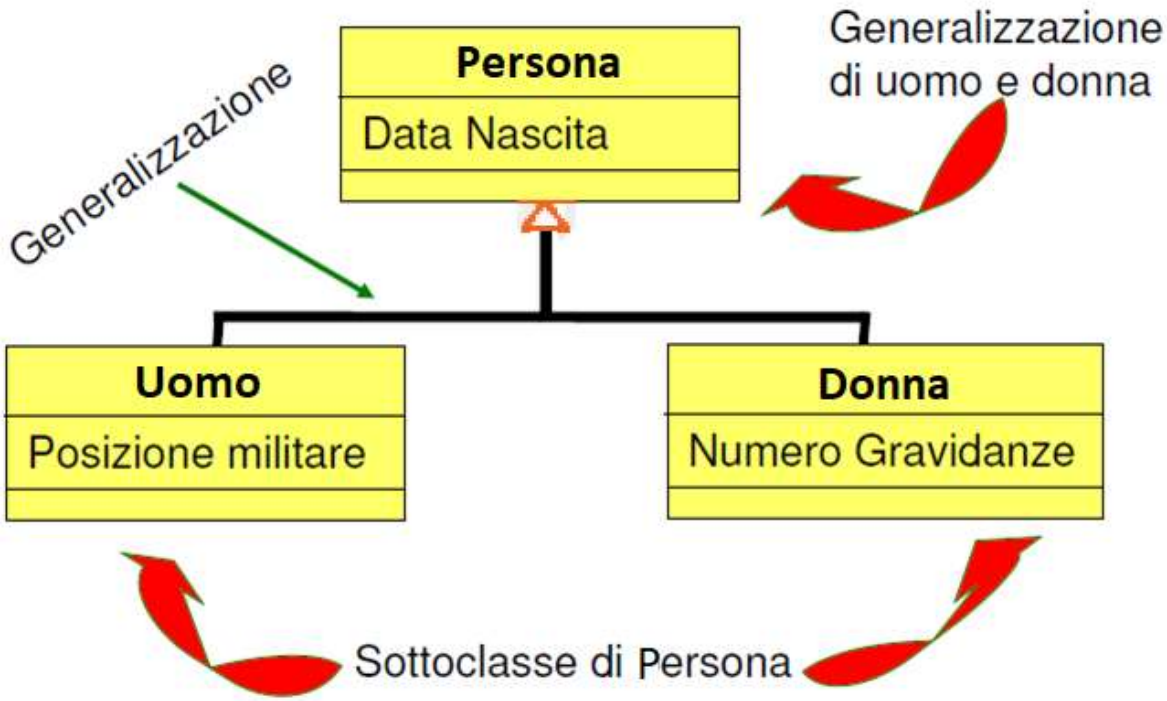
modificatore di visibilità	stessa classe	stesso package	sottoclassi stesso package	sottoclassi diverso package	diverso package (non sottoclassi)
public	OK	OK	OK	OK	OK
private	OK	KO	KO	KO	KO
protected	OK	OK	OK	OK	KO
(default)	OK	OK	OK	KO	KO

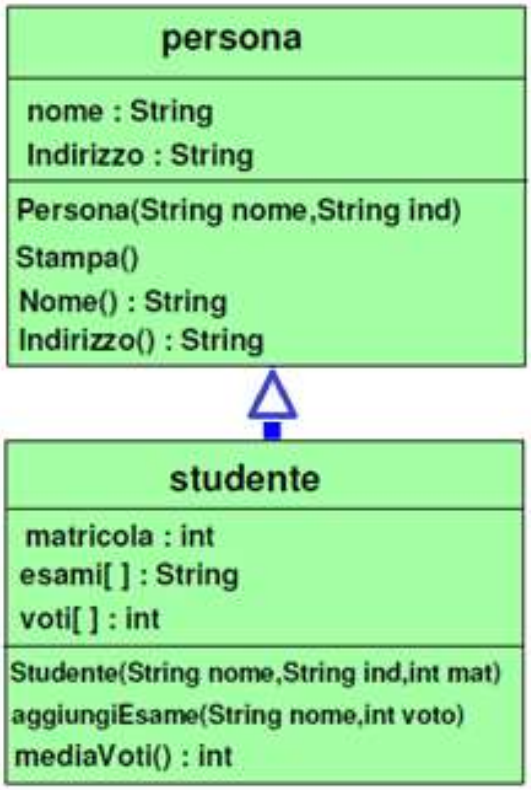
Nb: KO non visibile

altri modificatori

Relazioni tra classi

Generalizzazione = relazione "is-a“ → *Ereditarietà*





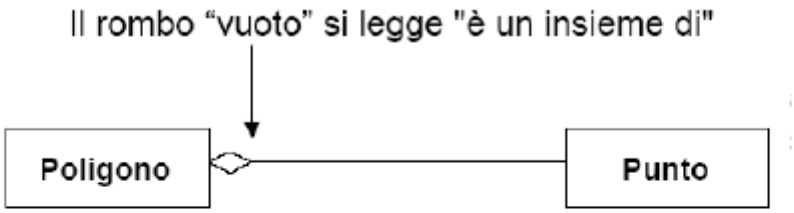
- la classe Studente eredita dal padre:
  - **attributi**
  - **metodi**
- Un oggetto Studente può essere trattato esattamente come un oggetto Persona

- In cosa solitamente può differenziarsi la classe erede ?
  - **aggiunta di attributi e metodi**
  - **i metodi possono essere ridefiniti**

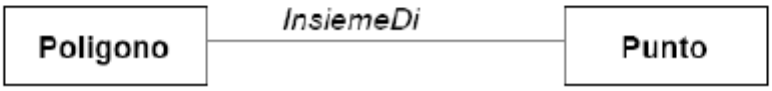
Associazione “ha-un”

Associazione del tipo: “è un insieme di” o *aggregazione*

Esempio:



Modello equivalente che usa una associazione convenzionale:



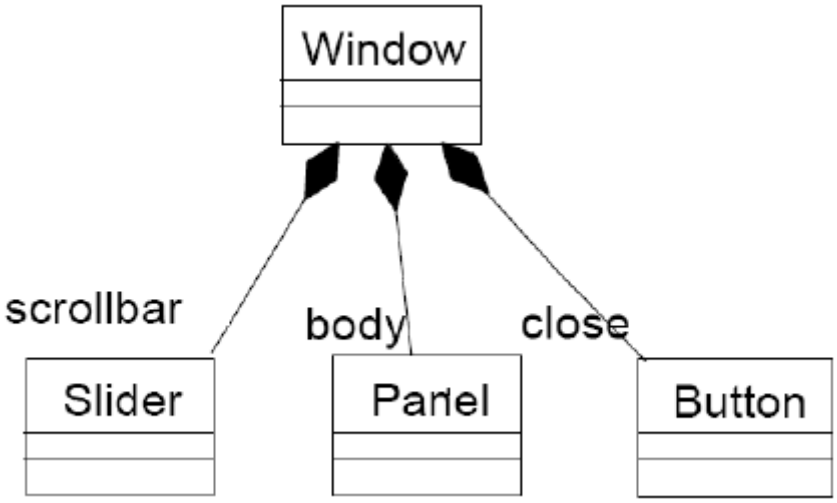
Caso particolare di aggregazione

del tipo: ”è composto da" o *composizione*

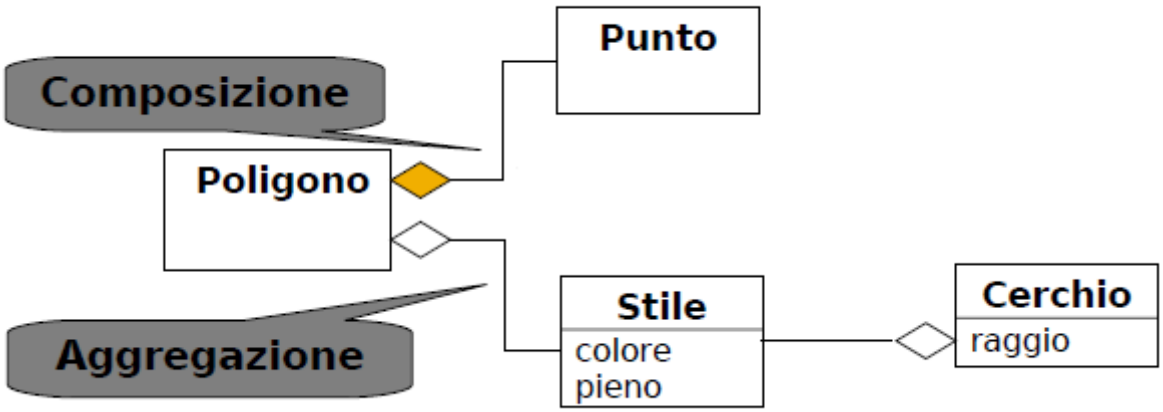
È una relazione più forte dell'aggregazione, *l'oggetto parte* appartiene ad un solo tutto e le parti hanno lo stesso ciclo di vita dell'insieme. All'atto della distruzione dell'oggetto principale si ha la propagazione della distruzione agli oggetti parte.

- I componenti non possono esistere senza il contenitore

Esempio:



Esempio:



- La cancellazione di una istanza della classe ***Poligono*** viene estesa ad ogni suo ***Punto***, ma non allo ***Stile*** ad esso associato

- Un'istanza della classe ***Stile*** può, invece, essere condivisa tra ***Poligono*** e ***Cerchio***

Relazione generica di dipendenza: "usa"

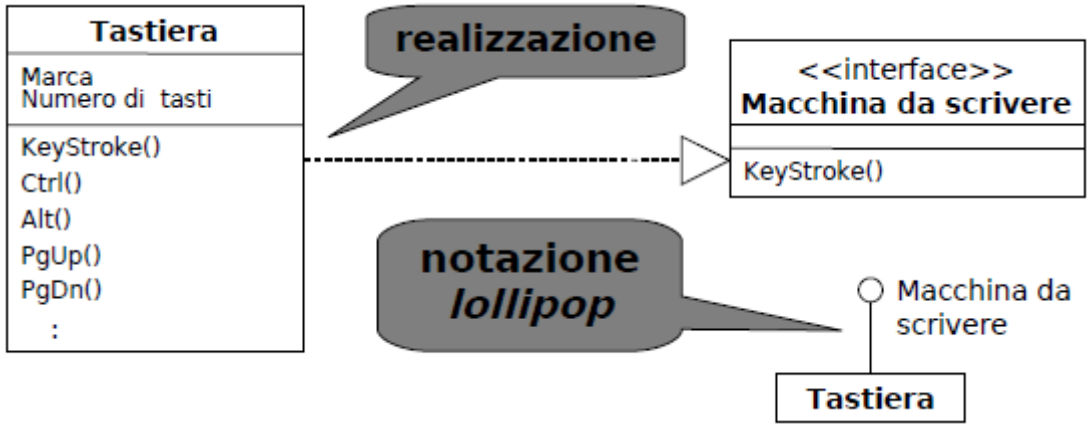
Dipendenza ------>

Interfacce e realizzazioni

- **Un'interfaccia viene modellata allo stesso modo in cui viene modellato il comportamento di una classe e rappresenta un insieme di operazioni che una classe offre ad altre classi**
  - Un'interfaccia non ha attributi ma soltanto operazioni (metodi). In UML per rappresentare le interfacce si utilizza un rettangolo con la dicitura «interface» o un piccolo cerchio (notazione *lollipop*, "a lecca-lecca").

- **La relazione tra una classe ed un'interfaccia viene definita realizzazione. Tale relazione è visualizzata nel modello da una linea tratteggiata con un triangolo largo aperto costruito sul lato dell'interfaccia o con una linea nella notazione compatta lollipop.**

Esempio:



La tastiera del computer è un tipico esempio di realizzazione di una interfaccia. La pressione di un tasto (KeyStroke) rappresenta un'operazione che è stata definita dall'interdaccia *Macchina per scrivere*. L'operazione KeyStroke() viene realizzata anche sulla tastiera dei computer. D'altra parte sulla tastiera dei computer si trovano un insieme di operazioni che non appartengono alla macchina per scrivere (Ctrl, Alt, PageUp, PageDown, ecc.)