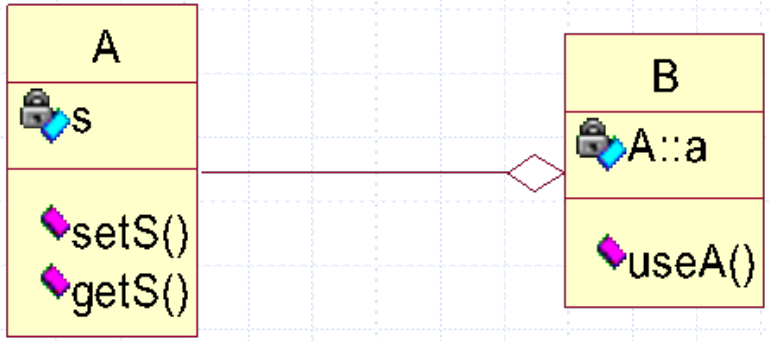
UML: Aggregazione

```
class A {
    int s;
    public void setS(int){…};
    public int getS() {…};
}
class B {A ob;
    public void useA() {…};
}
```

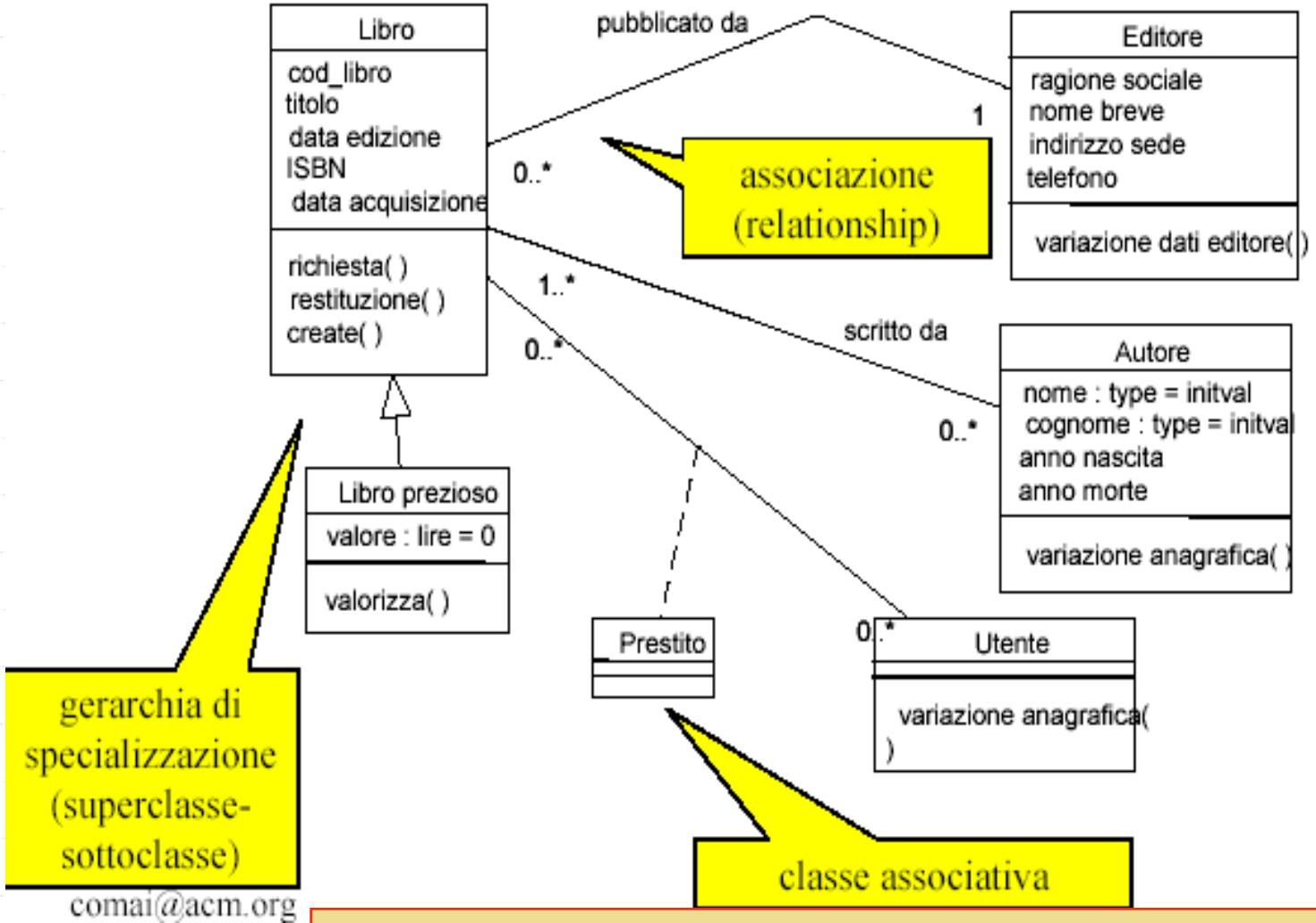


Aggregation - Composition

Use *aggregation (has-a)* when the lifecycle of the partecipating elements is different (one can exist without the other).

Use *composition (part-of)* when the *container* cannot be conceived without the *contained*.

UML – Class Diagram



comai@acm.org

Disegno ripreso da: Adriano Comai
http://www.analisi-disegno.com/a_comai/corsi/

Java

Generics

Uso di Generics nelle API di Java

```
// Removes 4-letter words from c. Elements must be strings
   static void expurgate(Collection c) {
      for (Iterator i = c.iterator(); i.hasNext(); )
        if (((String) i.next()).length() == 4)
          i.remove();
   }
```

Problemi?

Uso di Generics nelle API di Java

```
// Removes 4-letter words from c. Elements must be strings
   static void expurgate(Collection c) {
      for (Iterator i = c.iterator(); i.hasNext(); )
        if (((String) i.next()).length() == 4)
          i.remove();
   }
```

Here is the same example modified to use generics:

```
   // Removes the 4-letter words from c
   static void expurgate(Collection<String> c) {
      for (Iterator<String> i = c.iterator(); i.hasNext(); )
        if (i.next().length() == 4)
          i.remove();
   }
```

A partire da Java 5 molte classi sono state riscritte usando i generics

Vantaggi

No cast

Fail quick!

Uso semplice delle Collections

```
Collection insieme<Point>=new LinkedList<Point>t();

Point a=new Point(1,2); insieme.add(a);
Point b=new Point(3,4); insieme.add(b);

for (Point k:insieme) {
   System.out.println(k);
}
```