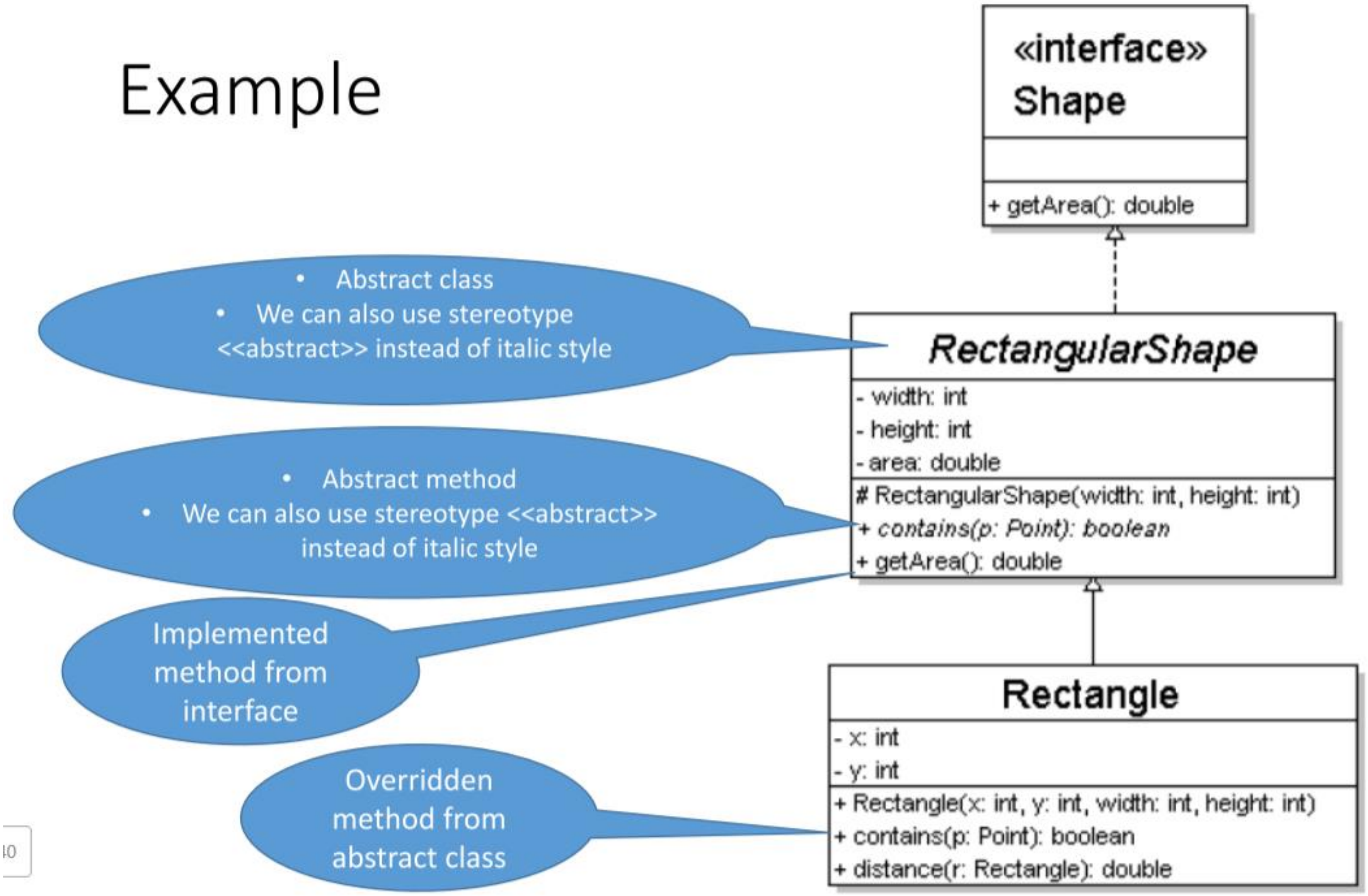
Example


«interface»
Shape
+ getArea(): double
RectangularShape
- width: int
- height: int
- area: double
# RectangularShape(width: int, height: int)
+ contains(p: Point): boolean
+ getArea(): double
Rectangle
- x: int
- y: int
+ Rectangle(x: int, y: int, width: int, height: int)
+ contains(p: Point): boolean
+ distance(r: Rectangle): double

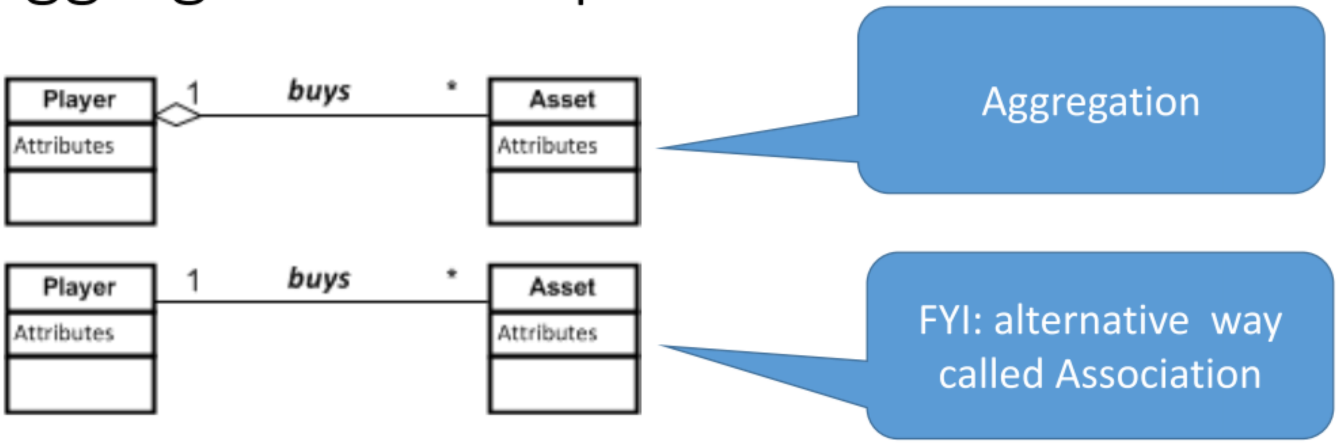
- Abstract class
- We can also use stereotype <<abstract>> instead of italic style

- Abstract method
- We can also use stereotype <<abstract>> instead of italic style

Implemented method from interface

Overridden method from abstract class

Aggregation Example

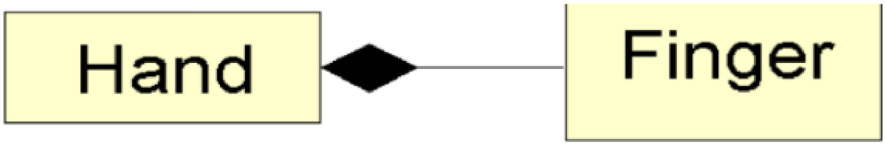


Aggregation

FYI: alternative way called Association

```
class Asset { … }
class Player {
List assets;
public void AddAsset(Asset newlyPurchasedAsset)
                    {
                    assets.Add(newlyPurchasedAssest);
          }
… }
```

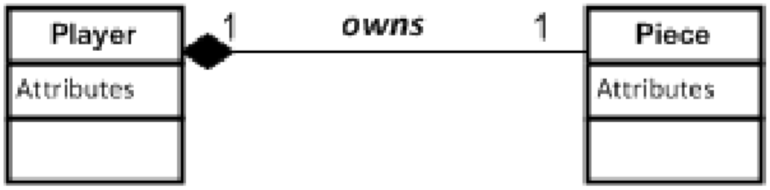
Composition



Composite

Part (Contained)

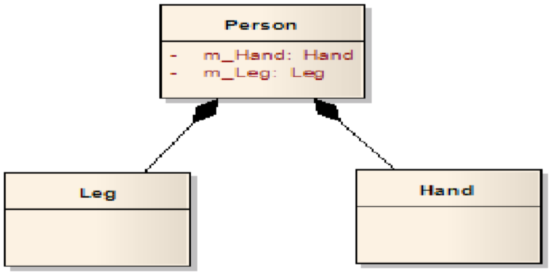
Composition



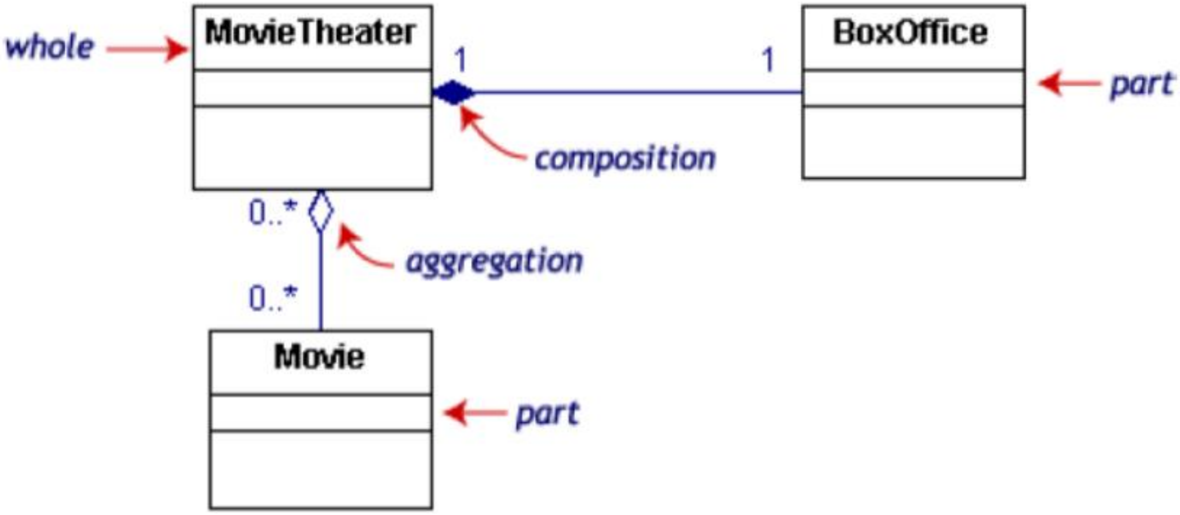
```
public class Piece { … }
public class Player
{
Piece piece = new Piece(); /*Player owns the responsibility
of creating the Piece*/
…
}
```

Composition

- Example 2: a person has a hand and a leg
- You can work and complete the human structure
  - E.g. another hand & leg, head, abdomen …etc.



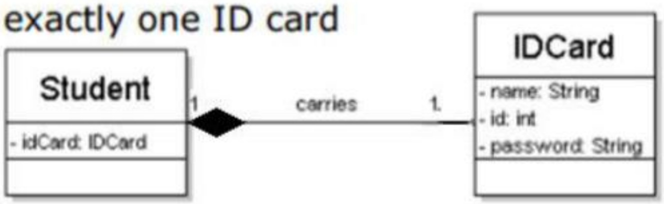
Aggregation and Composition Example



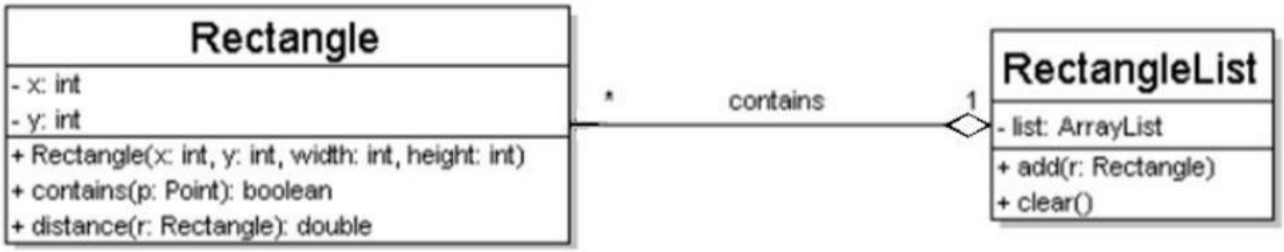
If the movie theater goes away
so does the box office => composition
but movies may still exist => aggregation

Multiplicity

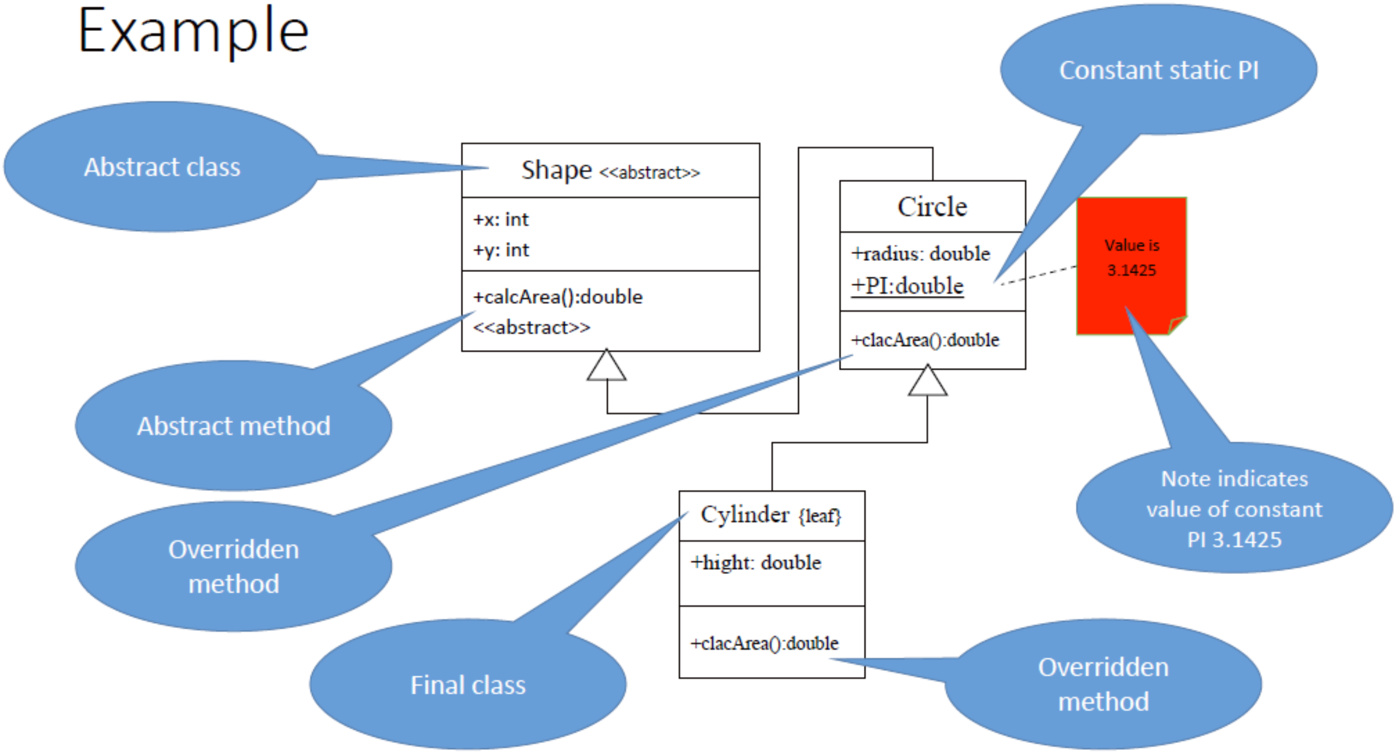
- one-to-one
  - each student must carry exactly one ID card



- one-to-many
  - one rectangle list can contain many rectangles


Rectangle
- x: int
- y: int
+ Rectangle(x: int, y: int, width: int, height: int)
+ contains(p: Point): boolean
+ distance(r: Rectangle): double
*
contains
1
RectangleList
- list: ArrayList
+ add(r: Rectangle)
+ clear()

Example



Constant static PI

Abstract class

Abstract method

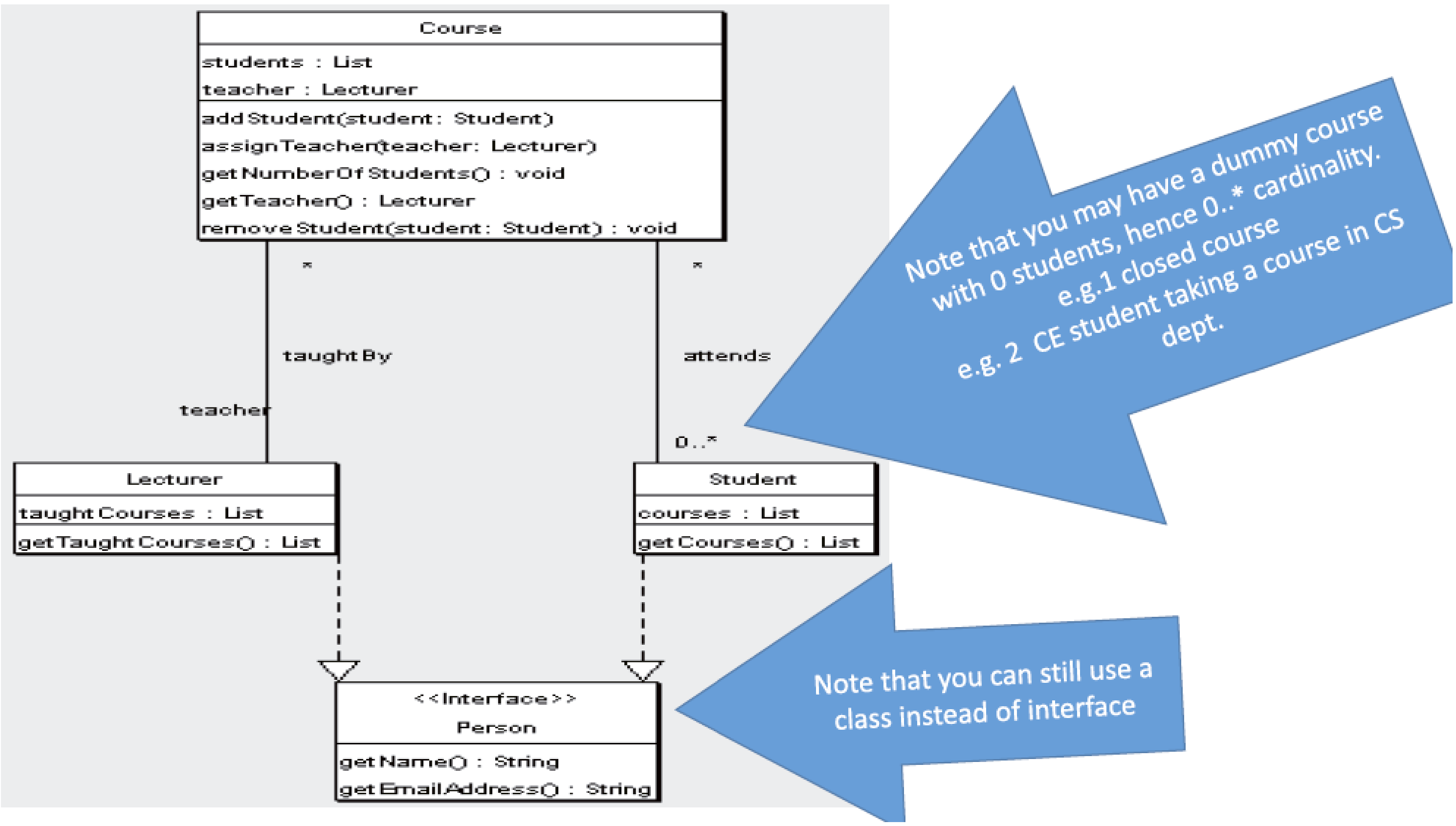
Overridden method

Note indicates value of constant PI 3.1425

Final class

Overridden method

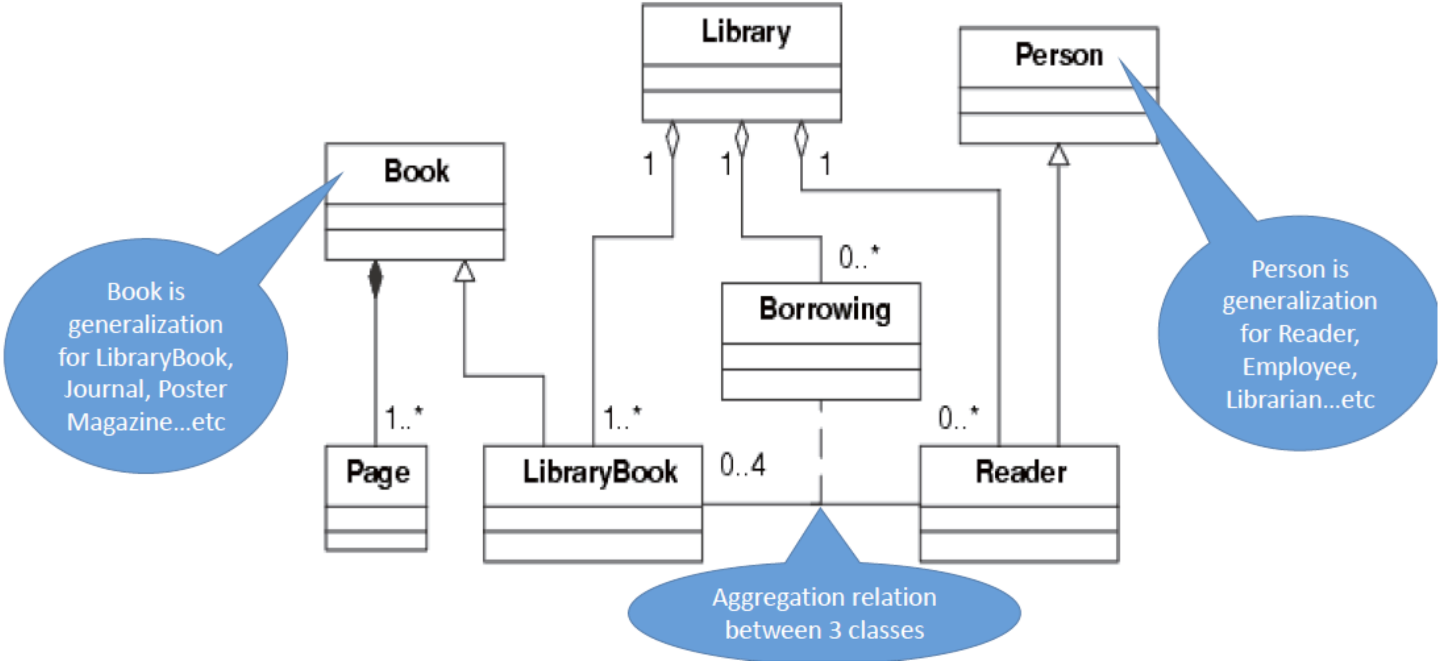
Example: Students, Lecturers & Courses



Note that you may have a dummy course with 0 students, hence 0..* cardinality.
e.g.1 closed course
e.g. 2 CE student taking a course in CS dept.

Note that you can still use a class instead of interface

Example: a library

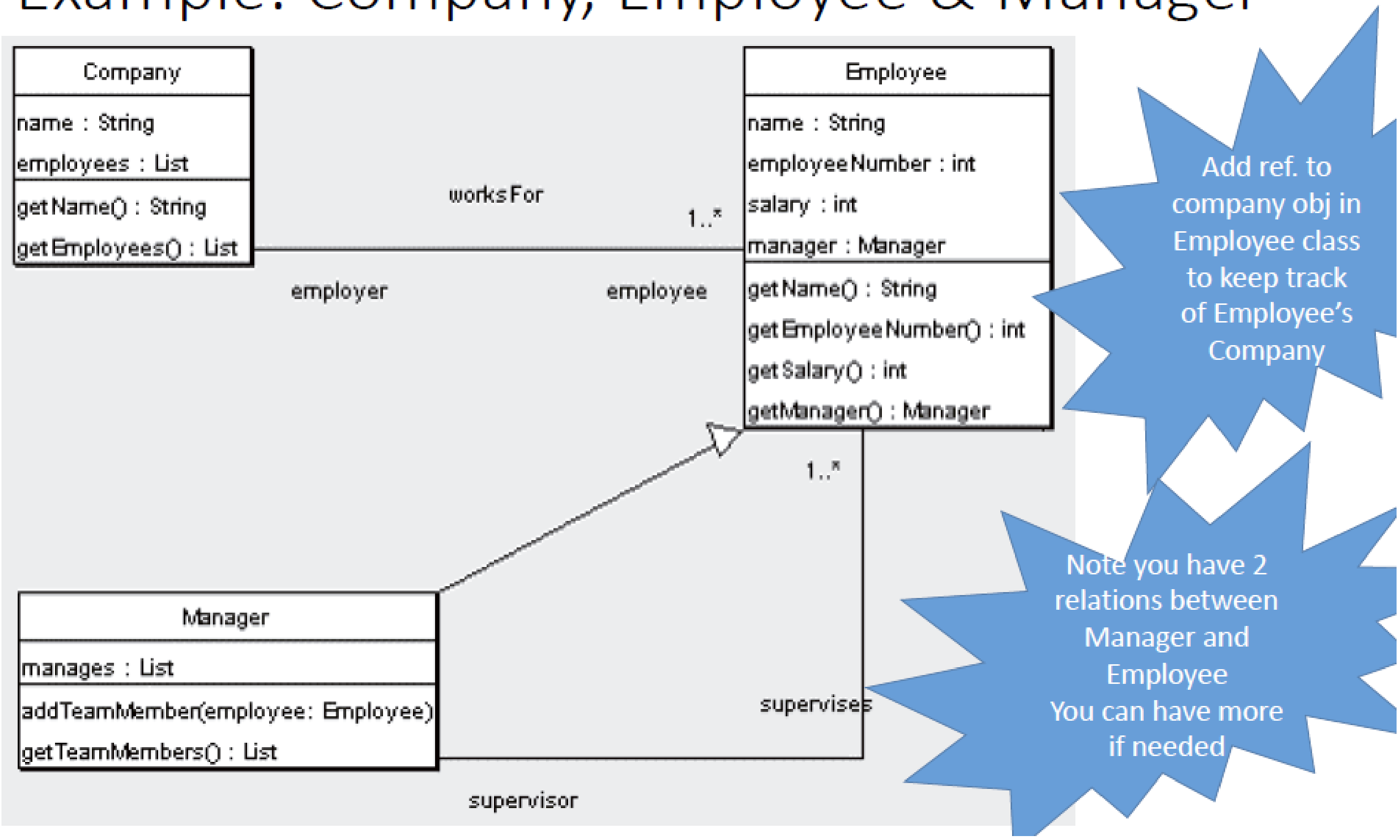


Book is generalization for LibraryBook, Journal, Poster Magazine…etc

Person is generalization for Reader, Employee, Librarian…etc

Aggregation relation between 3 classes

Example: Company, Employee & Manager

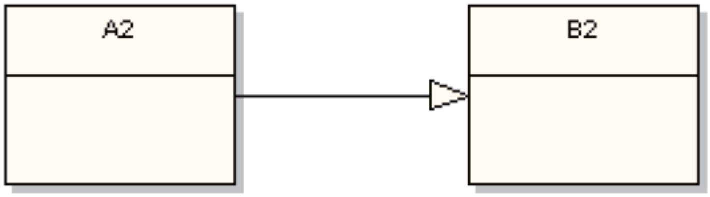


Add ref. to company obj in Employee class to keep track of Employee's Company

Note you have 2 relations between Manager and Employee
You can have more if needed

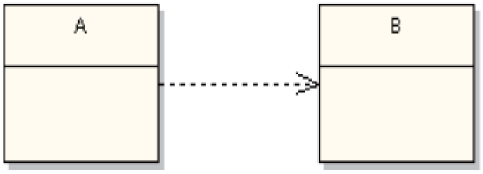
Generalization

```
class A2 extends B2{
.........
```

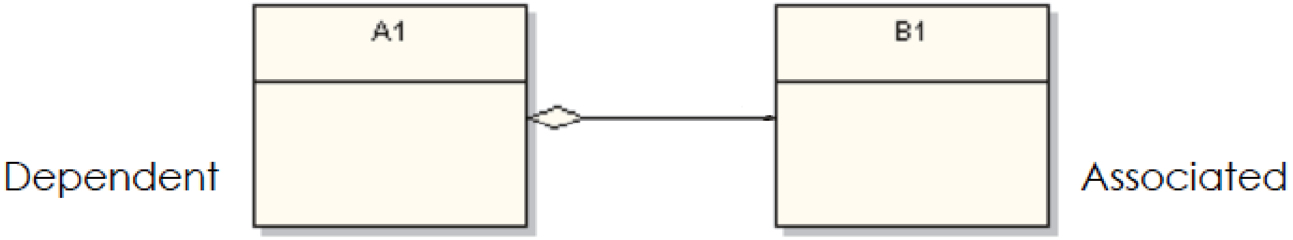


Dependency

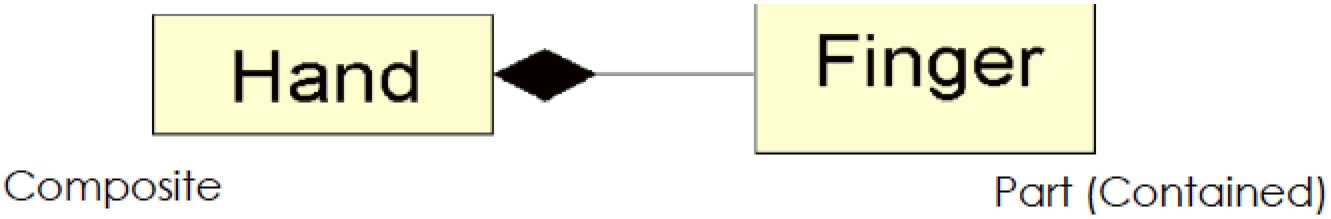
```
import B;
public class A {
                public void method1(B b) { // . . . }

}
```

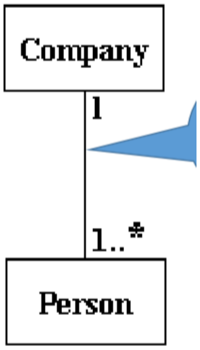


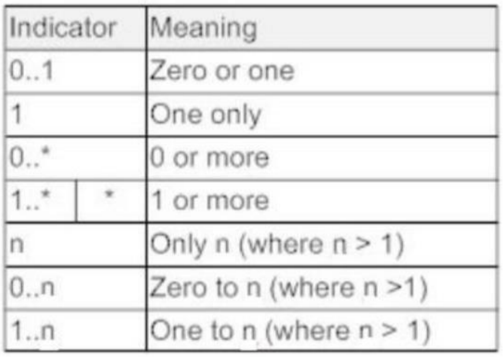
Aggregation



Composition







Indicator	Meaning
0..1	Zero or one
1	One only
0..*	0 or more
1..* *	1 or more
n	Only n (where n > 1)
0..n	Zero to n (where n >1)
1..n	One to n (where n > 1)