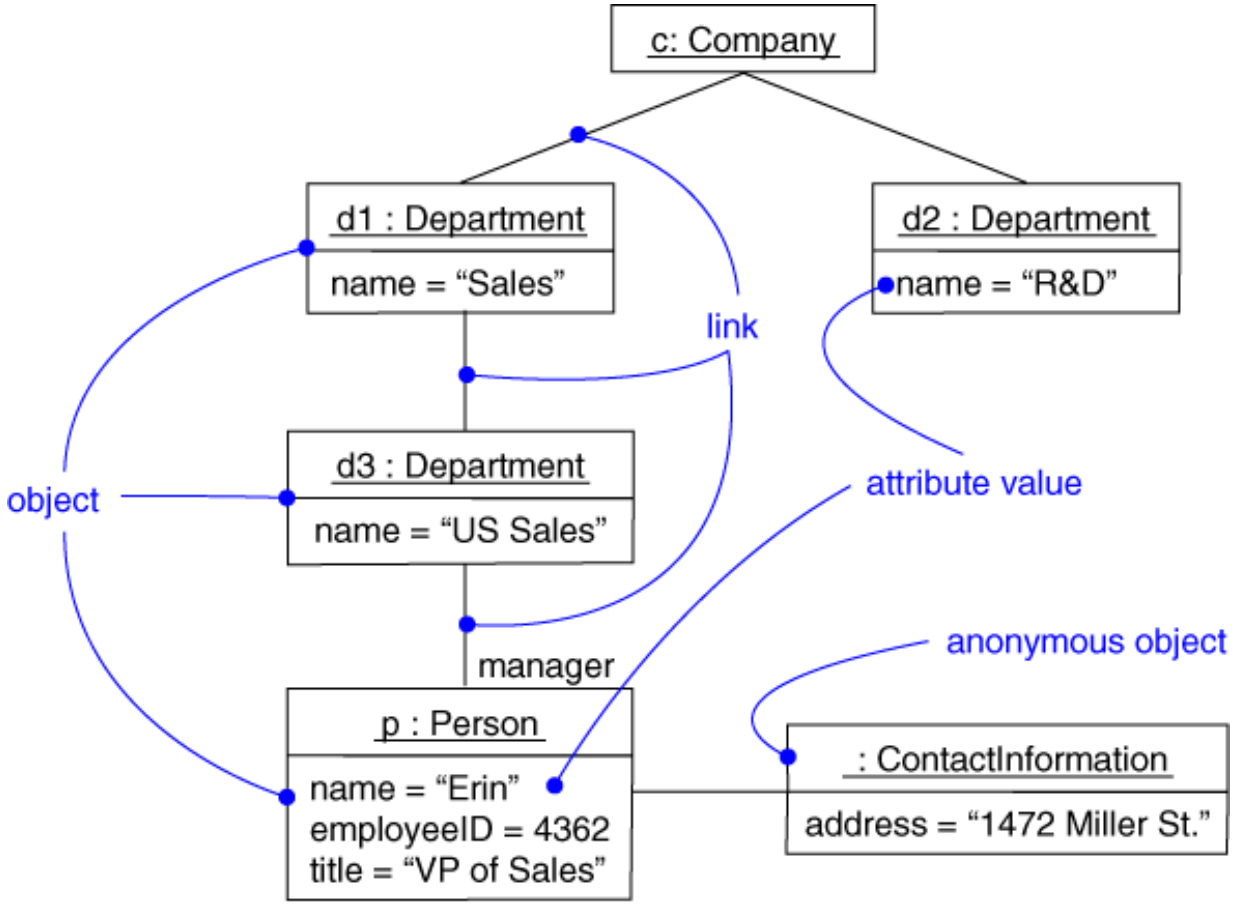
Software Design
Models, Tools & Processes

Lecture 3: Addendum

Cecilia Mascolo

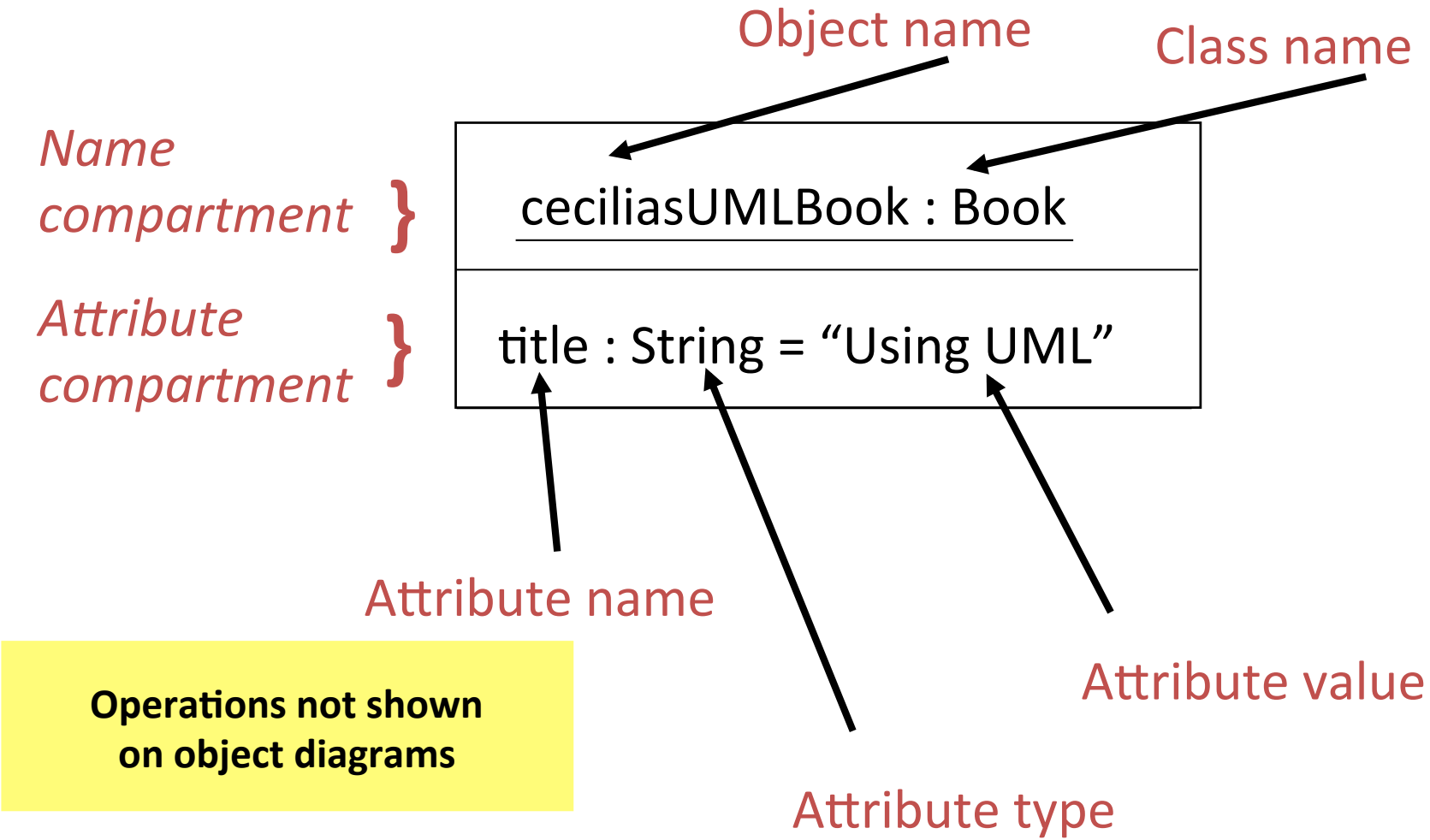
Example object diagram



Taken from [Booch 1999]

RATIONAL SOFTWARE

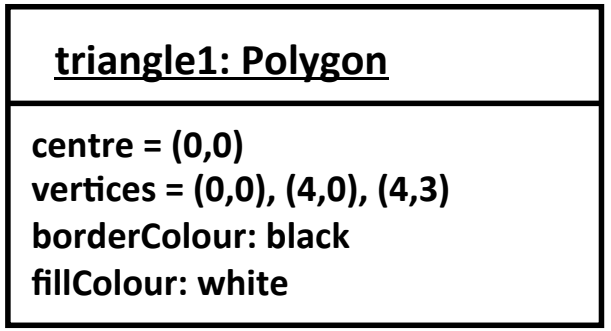
Notation for objects – an object *icon*



Example object

objectName: className

attribute name: type = value

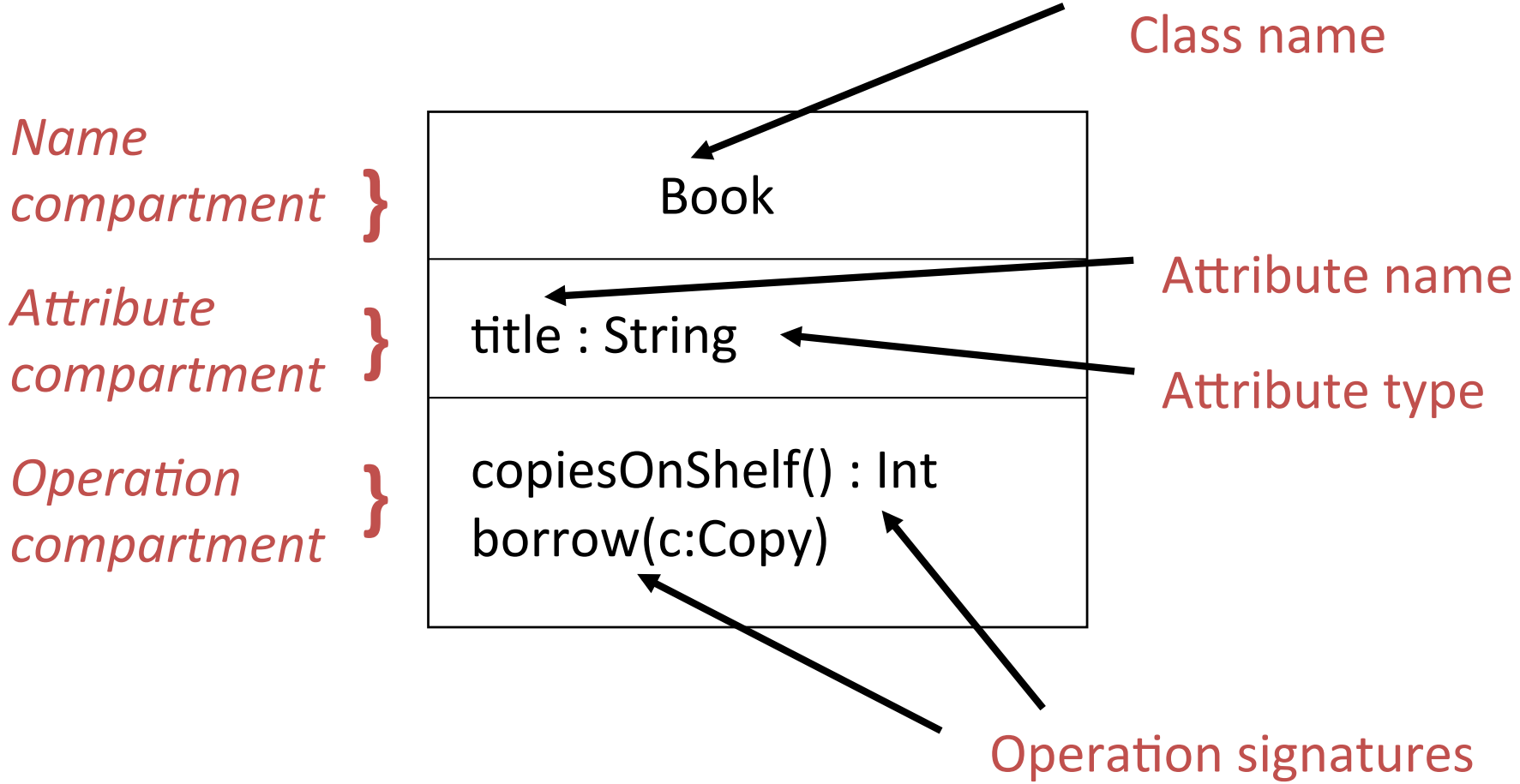


(same operations for all instances of a class)

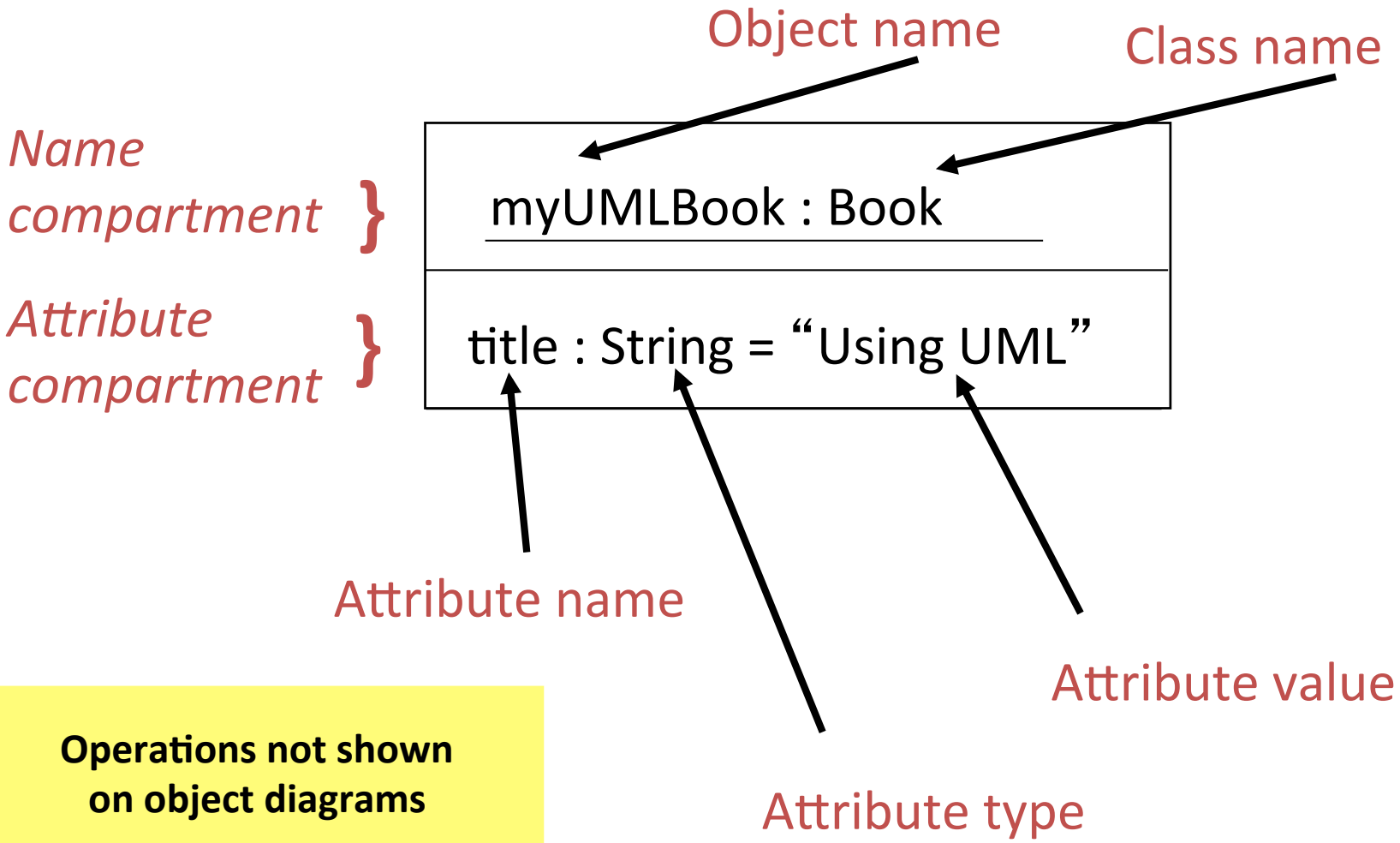
display (on: Surface)
rotate (angle: Integer)
erase ()
destroy ()
select (p: point): Boolean

objectName: className

Notation for classes - a class *icon*

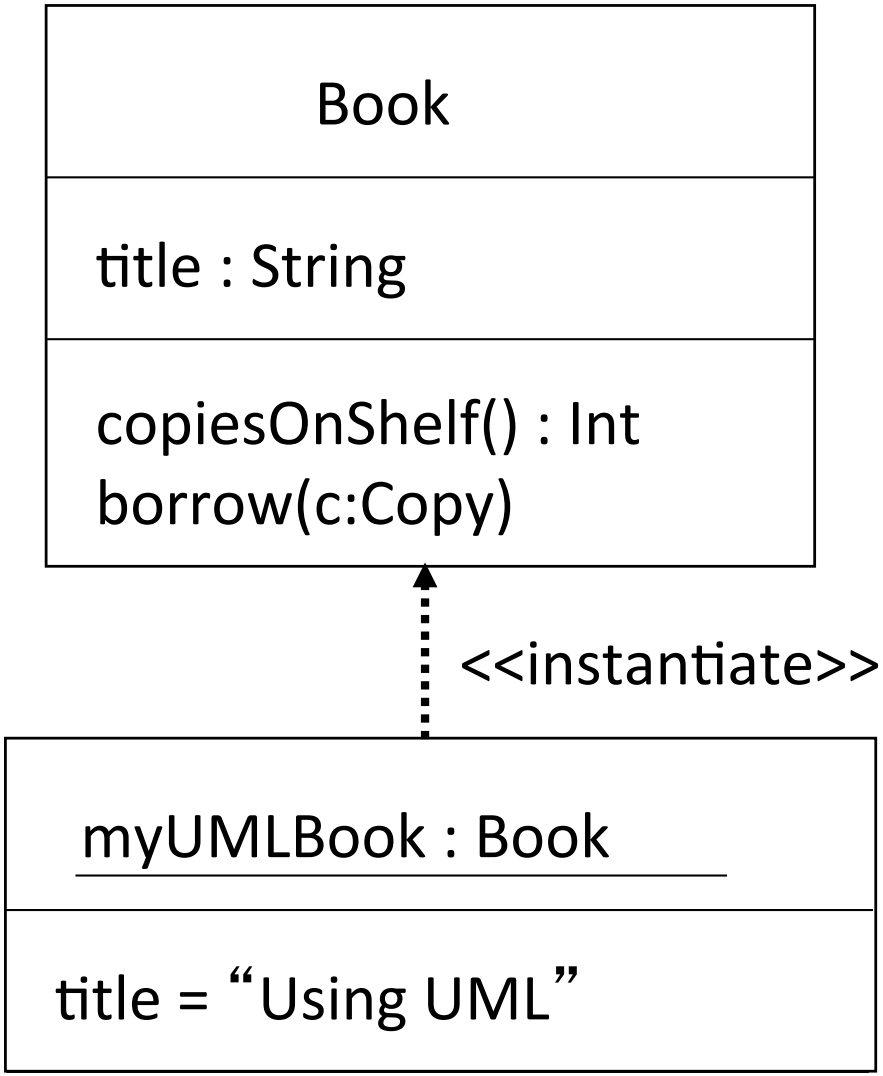


Notation for objects - an object *icon*



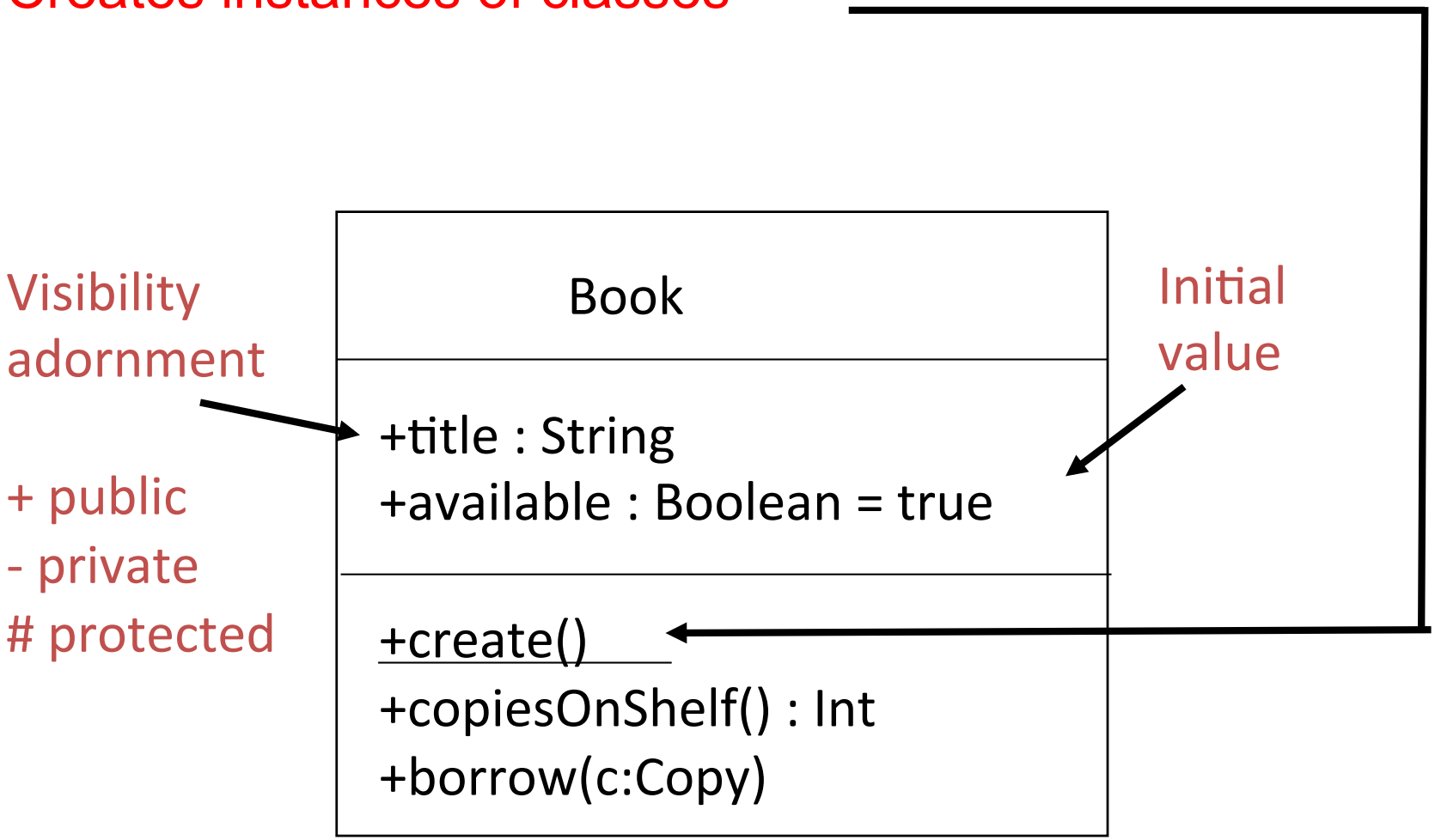
Operations not shown on object diagrams

Relating classes & objects



Constructors

Creates instances of classes



Relationships

- Relationships are connections between modelling elements - can be uni- or bi-directional
- Helps clarify understanding of the domain, describing how objects work together, & acts as a sanity check for good modelling
- We will look at
  - **Links** - relationships between objects
  - **Associations** - relationships between classes

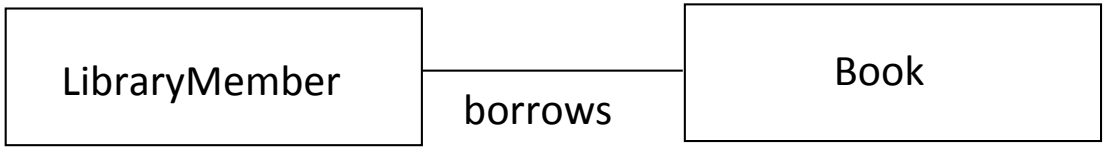
Links

- Objects send messages to one another to invoke operations
- To send messages, objects must have some way to reference other objects
- When an object has a reference to another object, a *link* exists between the objects
- Links are *instances* of associations of class diagrams

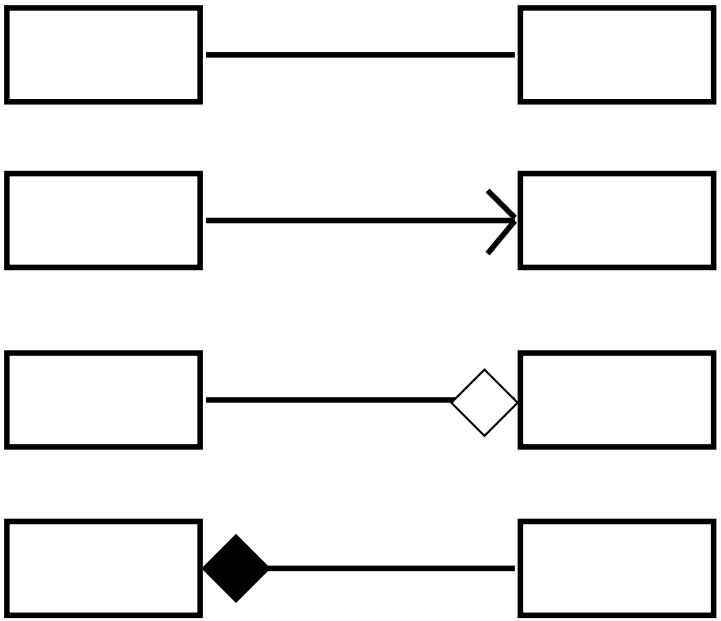
Associations

- Associations express relationships between classes. Class A and class B are associated if
  - Object of class A sends a message to object of class B
  - Object of class A creates an object of class B
  - Object of class A has attribute whose values are objects of class B
  - Object of class A receives message with object of class B as argument

- Real-world associations (e.g. a library member borrows a copy of a book)



Notation

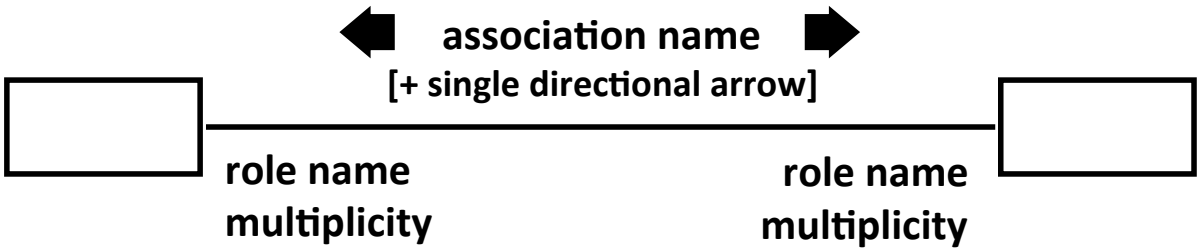


bidirectional / binary

unidirectional

aggregation

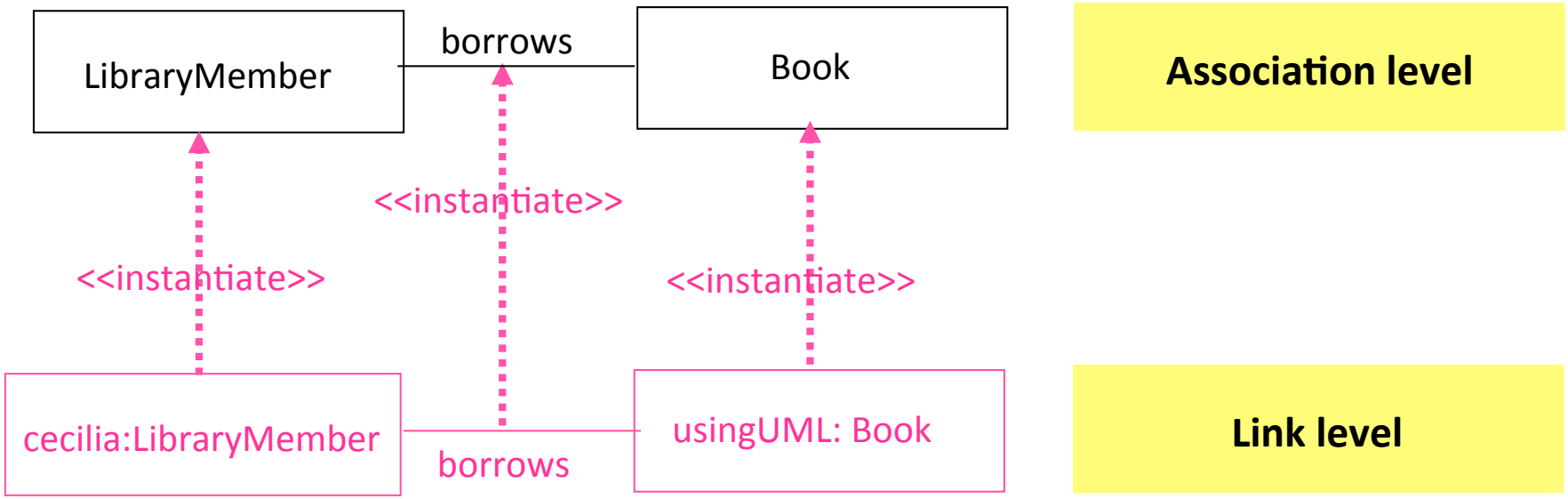
composition



supplementary characteristics

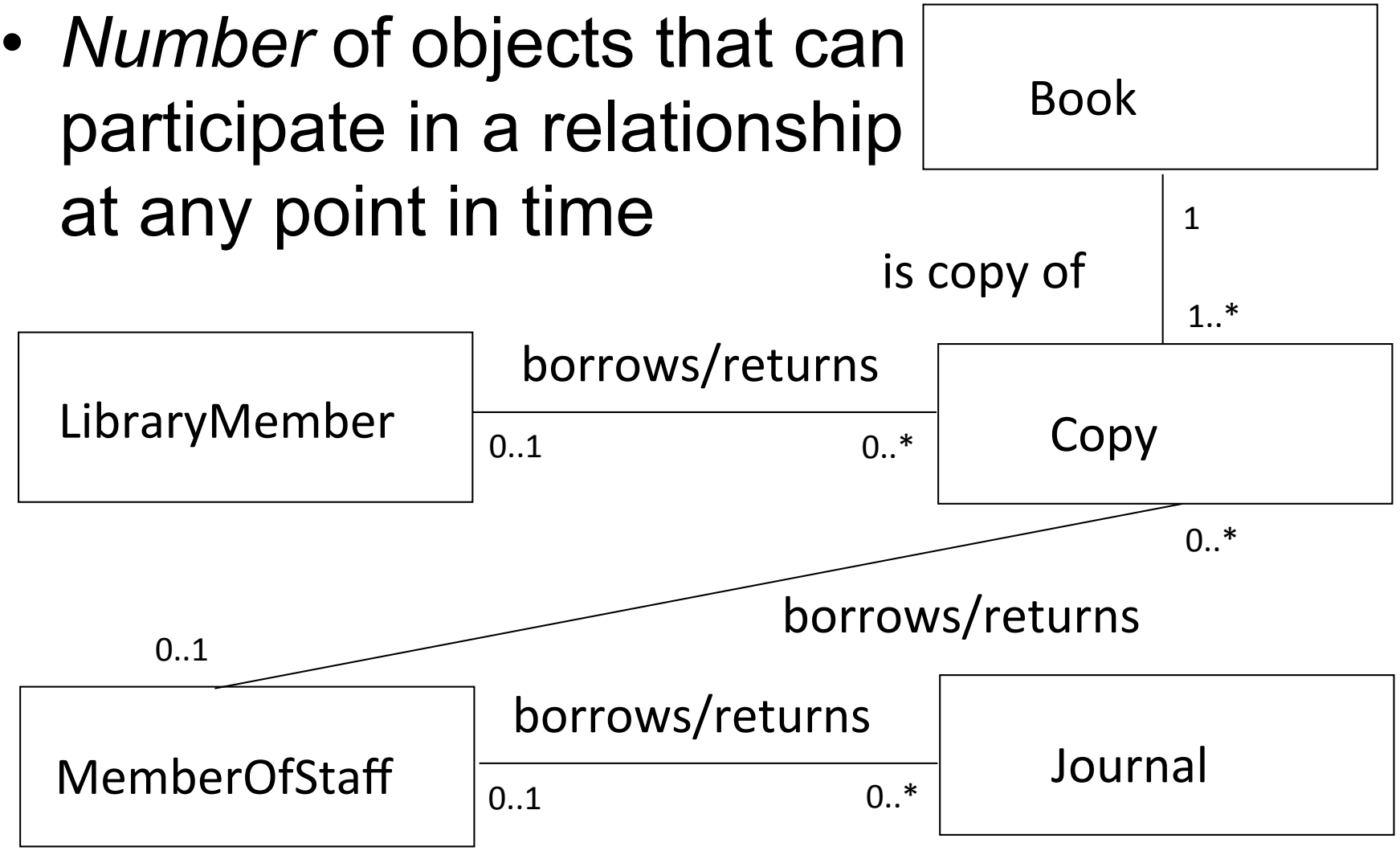
Links instantiate associations

- Links *depend* on associations



Multiplicity of an association

- *Number* of objects that can participate in a relationship at any point in time

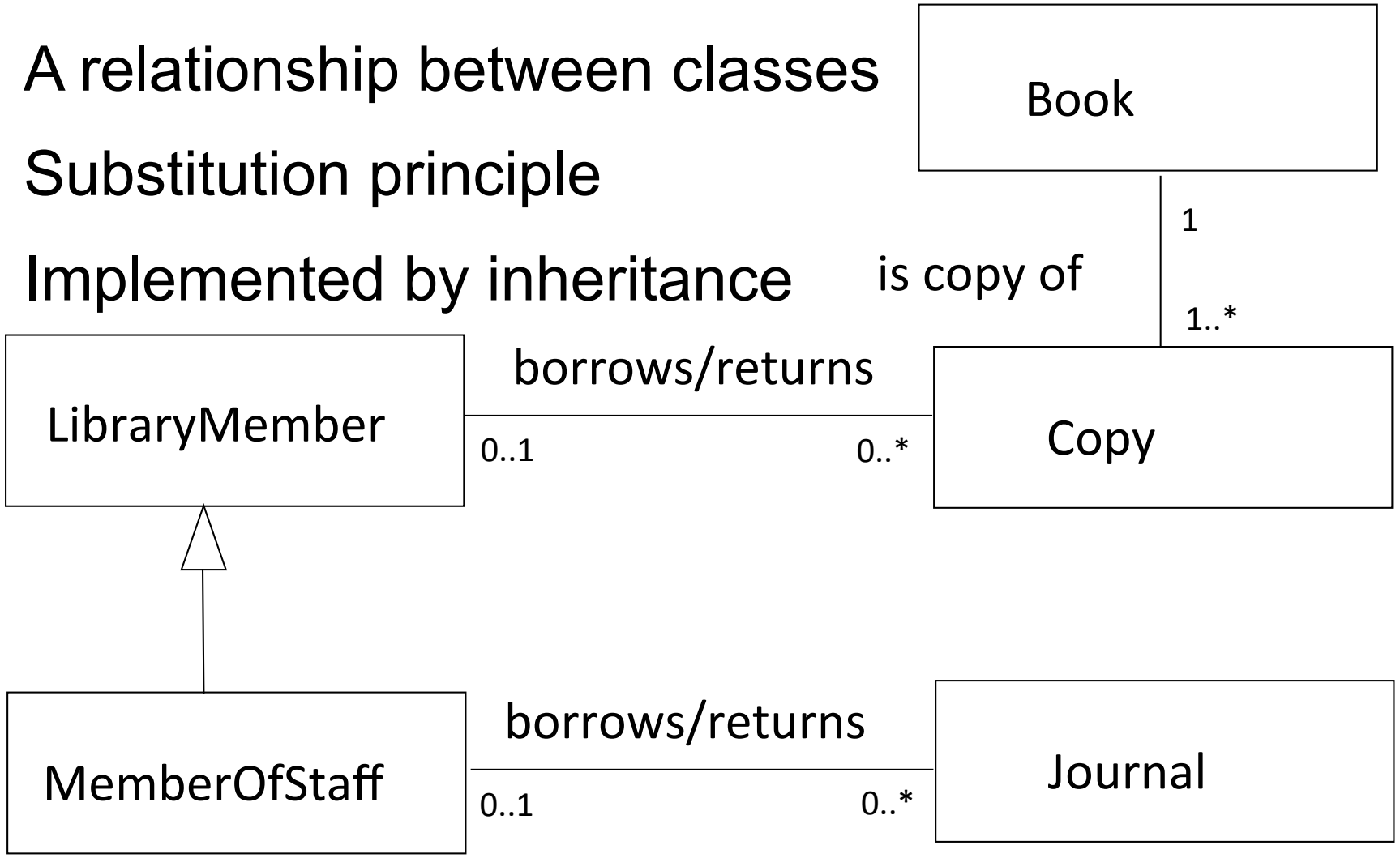


Exercise

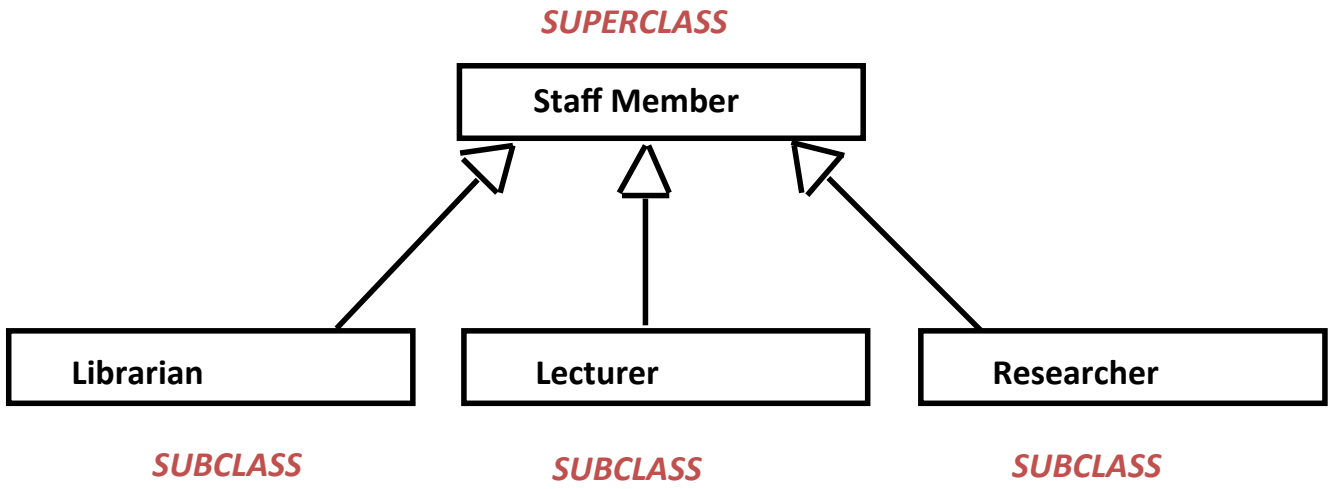
- Using your initial analysis class diagram for the library system, identify & add associations between the classes (NOTE: this is only meant to be a first approximation)
  - Consider the interactions in your earlier use cases
  - Consider the CRC cards you produced & the listed collaborators
- Add associations to your class diagram & provide each with: (a) a name; (b) multiplicity; & (c) navigability
- Your analysis model should now show classes, attributes, operations, named associations, multiplicity & navigability

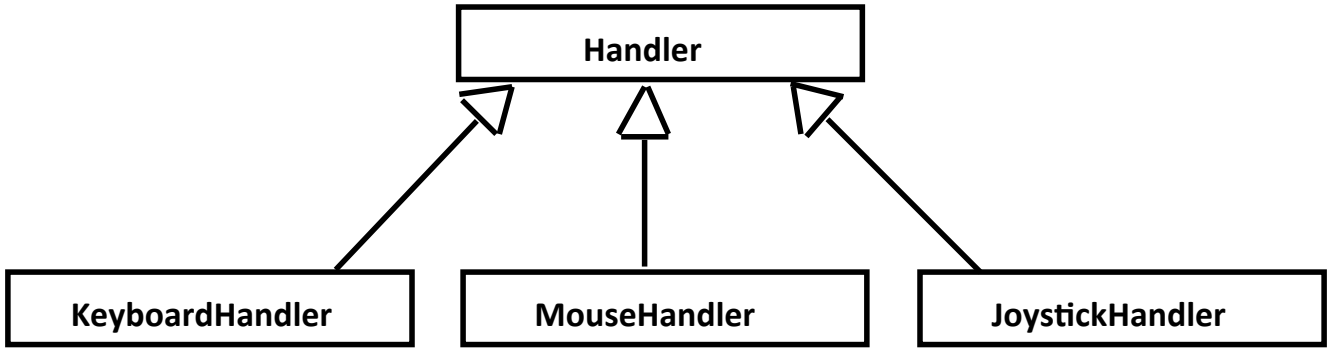
Generalisation & inheritance

- A relationship between classes
- Substitution principle
- Implemented by inheritance



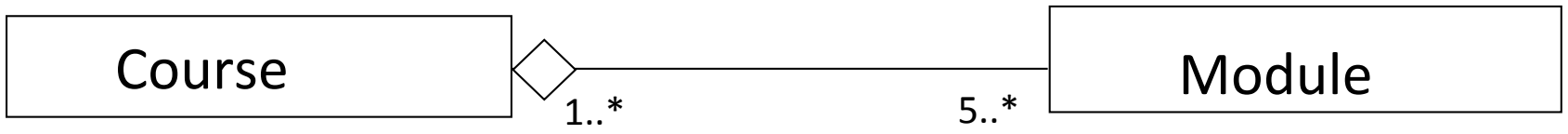
Example


SUPERCLASS
Staff Member
Librarian
Lecturer
Researcher
SUBCLASS
SUBCLASS
SUBCLASS

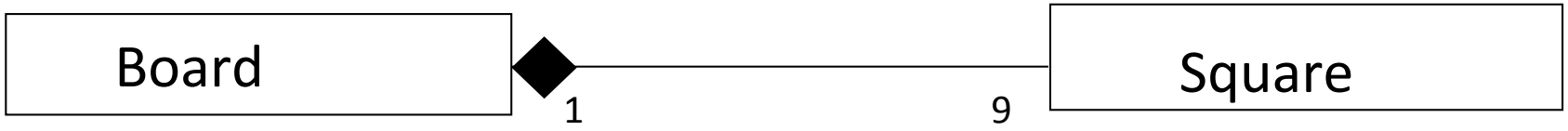

Handler
KeyboardHandler
MouseHandler
JoystickHandler

Part-of associations

- Aggregation
  - The part objects can feature simultaneously in any number of other objects

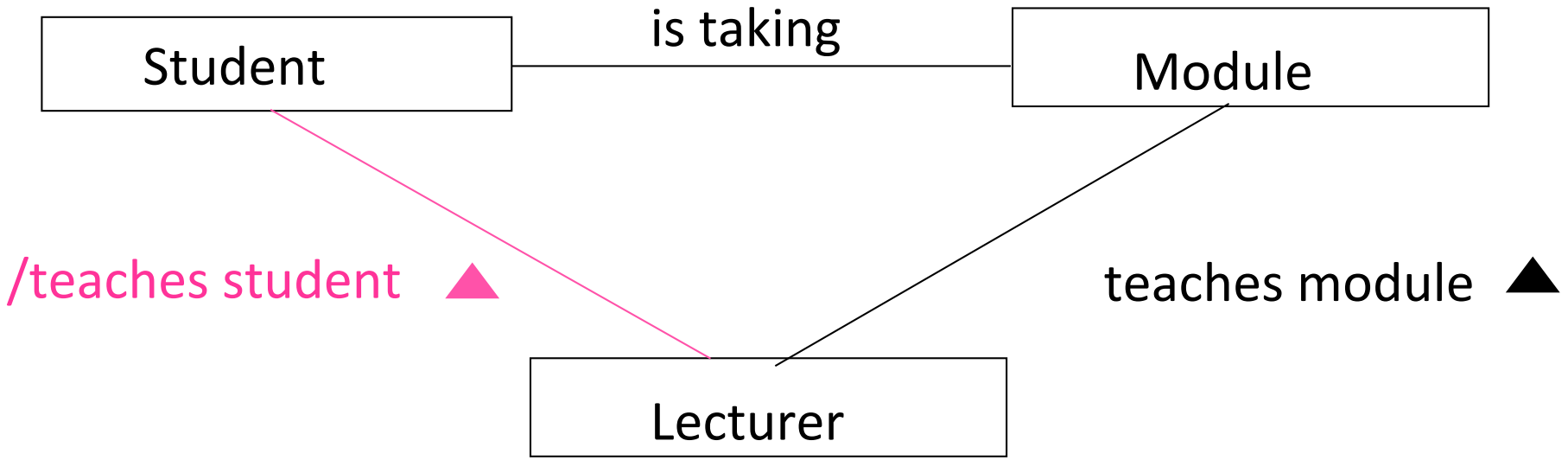


- Composition
  - The whole strongly owns its parts, so they cannot feature elsewhere



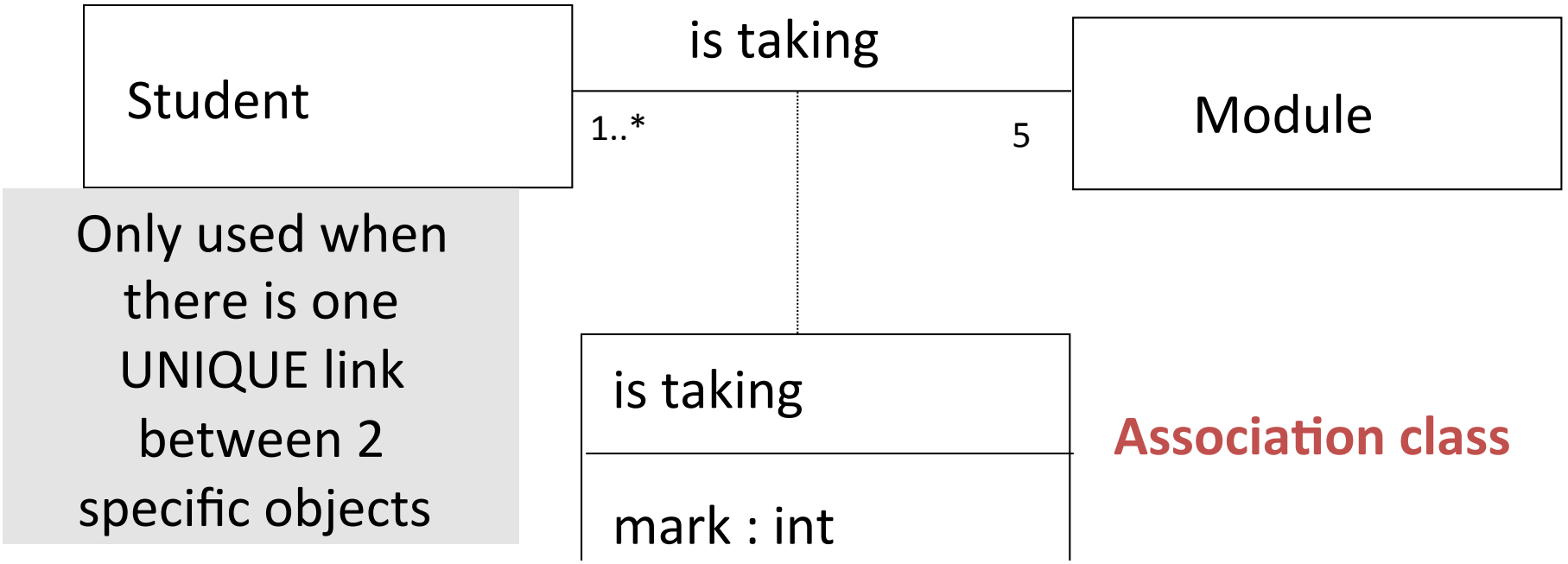
Derived associations

- Do you always need to show all associations?
- Sometimes associations that are not explicit can be deduced from the diagram



Association classes

- Used when it is required to add data to particular links (I.e. attributes not easily placed in original classes)
- Class icon & the association line must have the same name (as the same thing!)



Dependencies

- A relationship between two elements where a change to one element may affect information needed by the other element

- There are different kind of dependencies
  - Most used one is the one where you specify that one class is using definitions from another class

Exercise

- Examine your updated analysis model (i.e. the one that shows classes, attributes, operations, named associations, multiplicity & navigability)

- Update your model if there is anywhere you can apply
  - Inheritance
  - Association classes
  - Qualified associations
  - Other dependencies

- NOTE: modelling is an extremely iterative activity (hence why tool support is so desirable!)