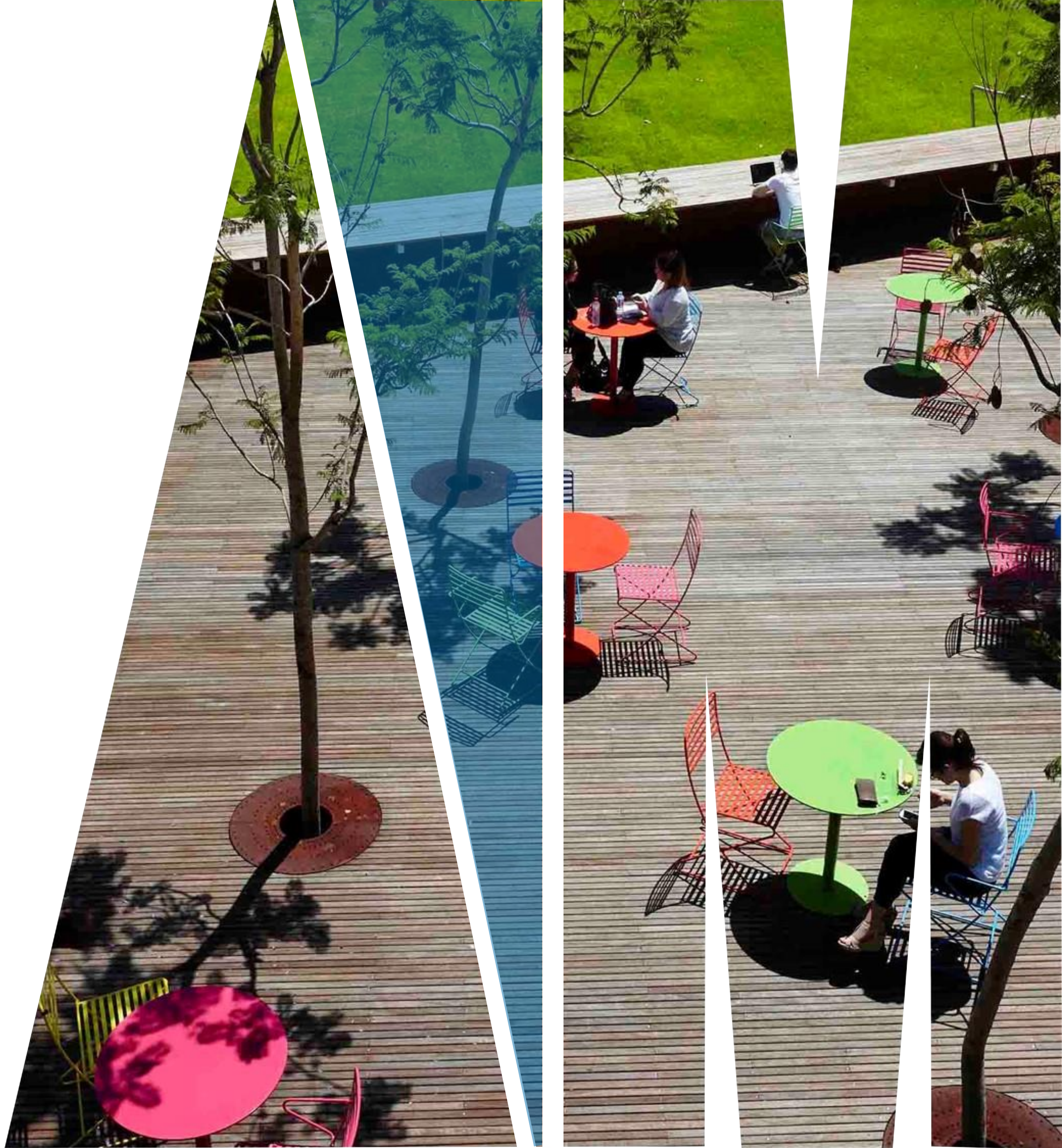
MONASH University

FIT2099 Object-Oriented Design and Implementation

Dependencies and associations



MONASH University

Outline

Dependencies and associations

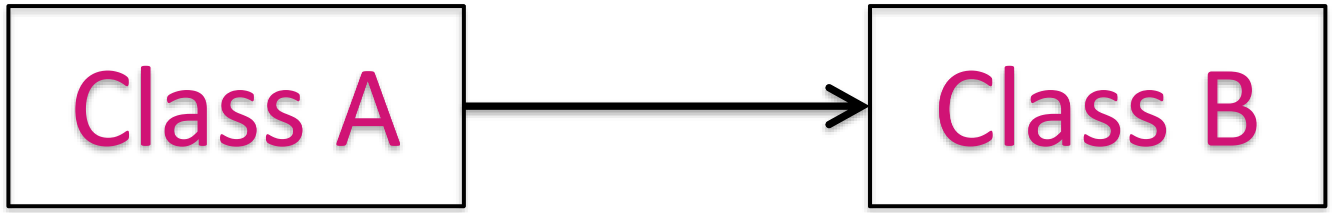
UML notation

Other association subsets

- Composition
- Aggregation

WHAT IS AN ASSOCIATION?

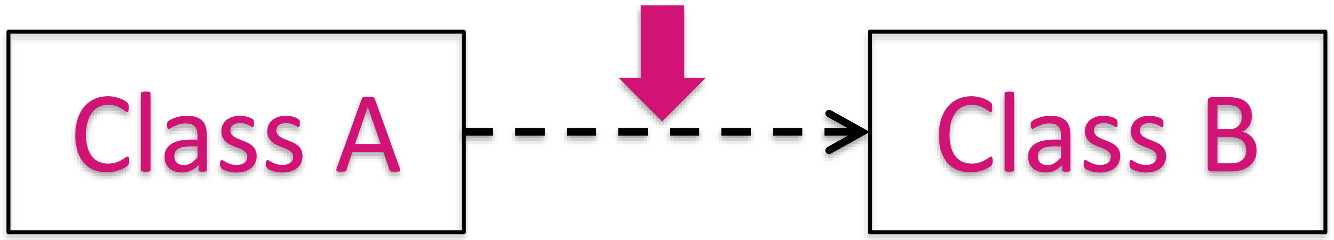
An **association** almost always implies that one object **has the other** object as an attribute.



MONASH University

WHAT IS A DEPENDENCY BETWEEN TWO OBJECTS?

A **dependency** typically (but not always) implies that an object accepts another object as a method parameter, instantiates, or uses another object.
A **dependency** is implied by an **association**.



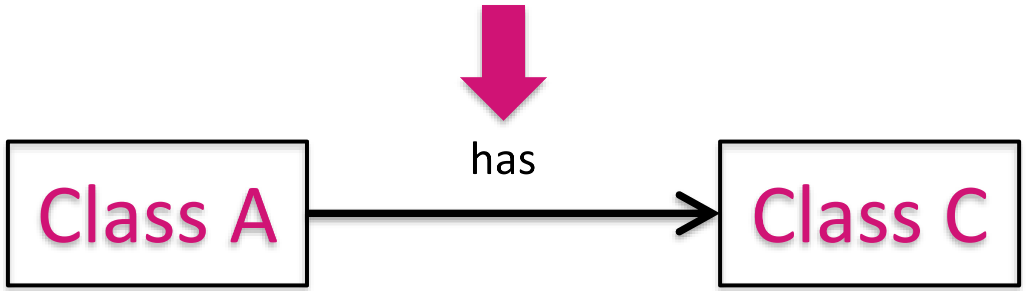
HOW DOES THIS LOOK LIKE IN JAVA CODE?

Association --> A **has-a** C object (as an attribute)

```java
public class A {
    private C c;



}
```



MONASH University

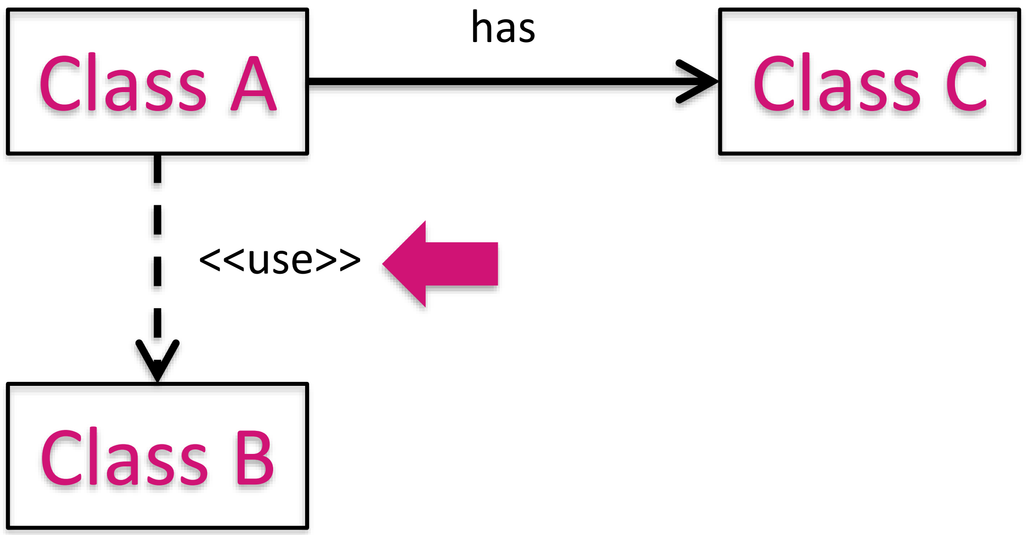
HOW DOES THIS LOOK LIKE IN JAVA CODE?

Association --> A **has-a** C object (as an attribute)

Dependency --> A **references** B (as a method parameter or return type)

```java
public class A {
    private C c;
    public void myMethod(B b) {
        b.callMethod();
    }
}
```

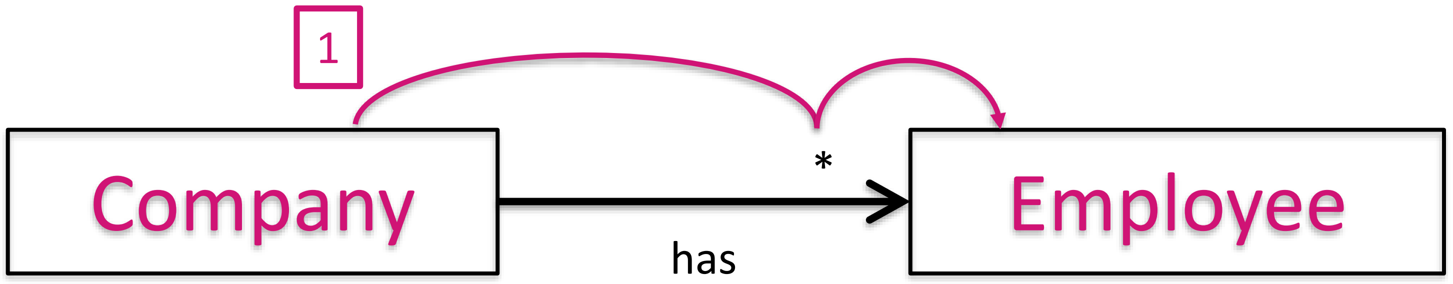
An object of the Class B is not an attribute of Class A



MONASH University

WHAT IS MULTIPLICITY?

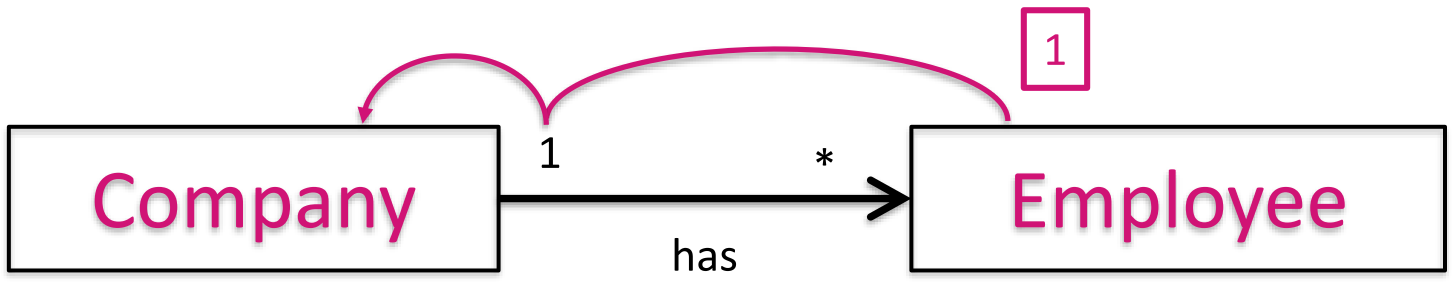
Place **multiplicity** notations near the ends of an association. These symbols indicate the **number of instances of one class linked to one instance of the other class**.



For example, **one company will have one or more employees**, but each employee works for one company only.

WHAT IS MULTIPLICITY?

Place **multiplicity** notations near the ends of an association. These symbols indicate the **number of instances of one class linked to one instance of the other class**.

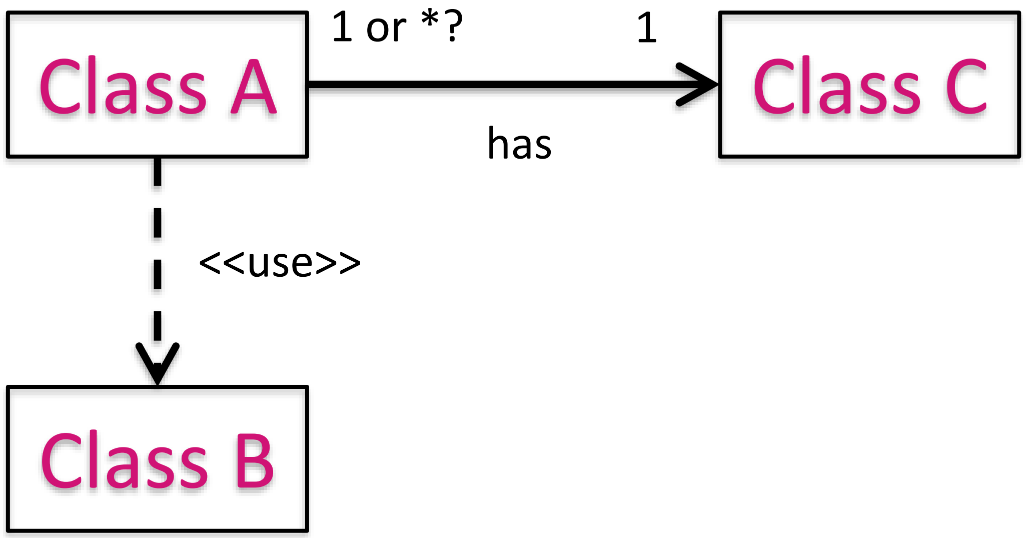


For example, one company will have one or more employees, **but each employee works for one company only.**

WHAT IS MULTIPLICITY?

Association --> A has **exactly one** C object (as an attribute)

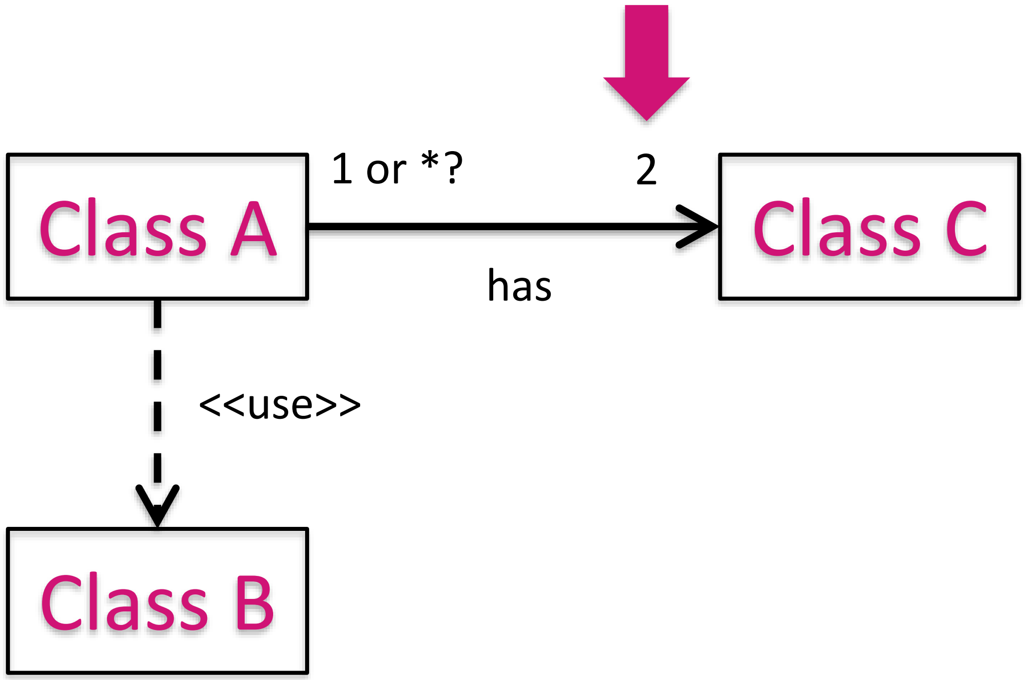
```
public class A {
    private C c;
    public void myMethod(B b) {
        b.callMethod();
    }
}
```



WHAT IS MULTIPLICITY?

Association --> A has **exactly two** C objects (as attributes)

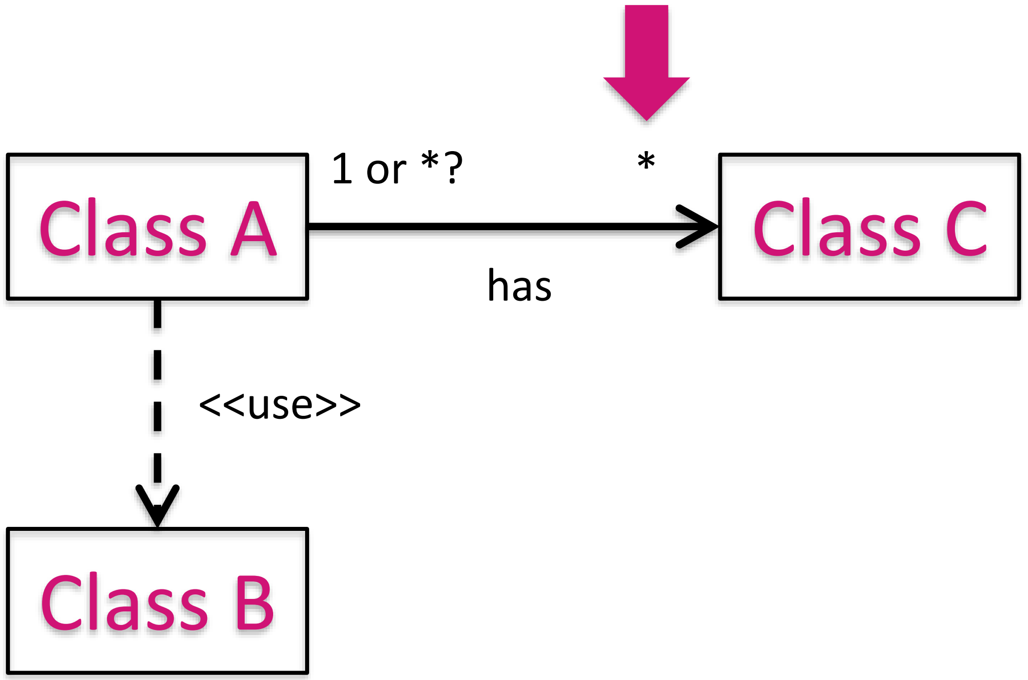
```
public class A {
    private C c;
    private C c2;
    public void myMethod(B b) {
        b.callMethod();
    }
}
```



WHAT IS
MULTIPLICITY?

Association --> A has **multiple** C objects (as an array or list)

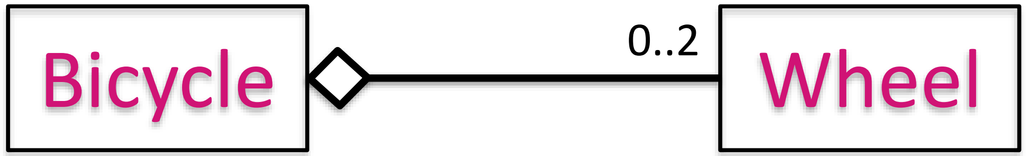
```java
public class A {
    ArrayList<C> cList = new ArrayList<>();
    public void myMethod(B b) {
        b.callMethod();
    }
}
```

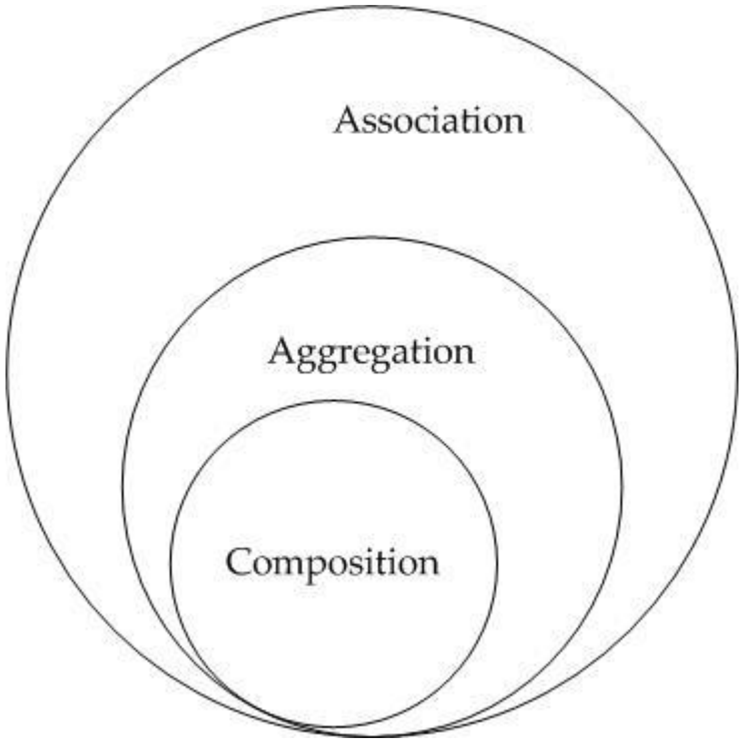


OTHER
ASSOCIATION SUBSETS

Aggregation is same as association and is often seen as redundant relationship. A common perception is that aggregation represents one-to-many / many-to-many / part-whole relationships (i.e. higher multiplicity), which can be represented by via association too (**hence the redundancy**).

Some developers use a hollow diamond to indicate aggregation.

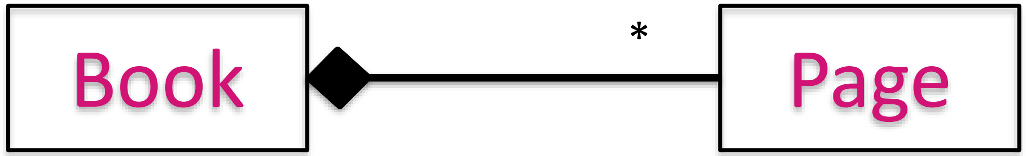


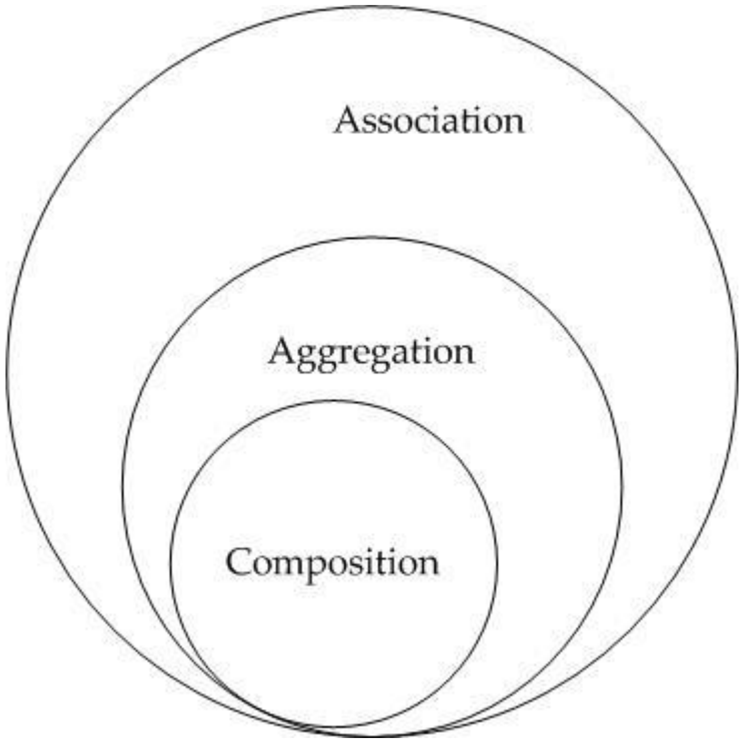


OTHER
ASSOCIATION SUBSETS

Composition relates to instance **creational responsibility**. When class B is composed by class A, class A instance owns the creation or controls lifetime of instance of class B.

When class instance A is destructed (garbage collected), class B instance would also get destructed.





OTHER
ASSOCIATION SUBSETS

In FIT2099 you will not be asked to use Composition and **Aggregation**. You can focus on Association, Dependency and Inheritance (this later to be covered in the near future).

To quote Rumbaugh (one of the original and key UML creators):

"In spite of the few semantics attached to aggregation, everybody thinks it is necessary (for different reasons). Think of it as a modeling placebo".

Summary

Dependencies and associations

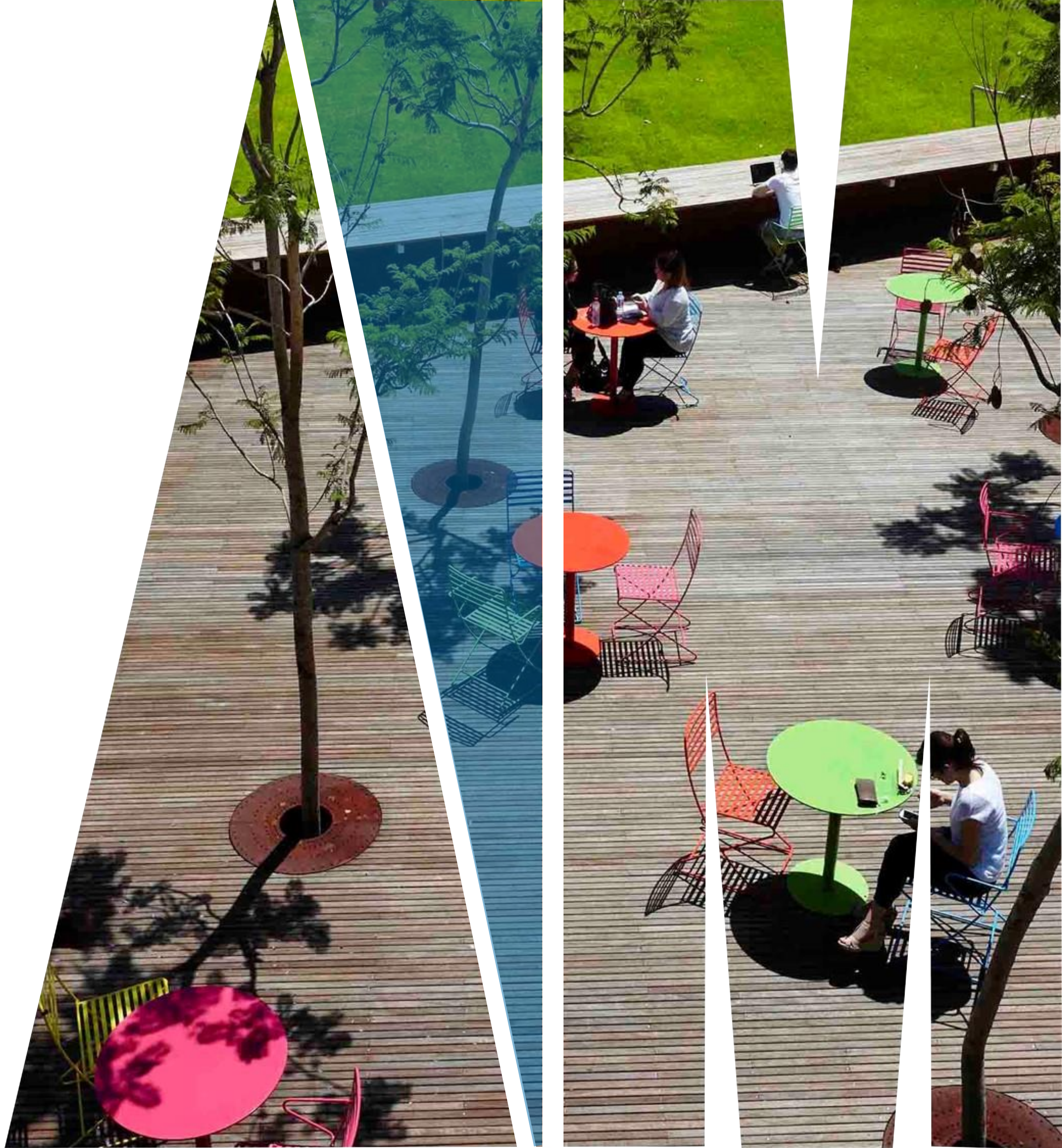
UML notation

Other association subsets

- Composition
- Aggregation

MONASH University

Thanks



MONASH University