EÖTVÖS LORÁND
UNIVERSITY | BUDAPEST

Class Relationship

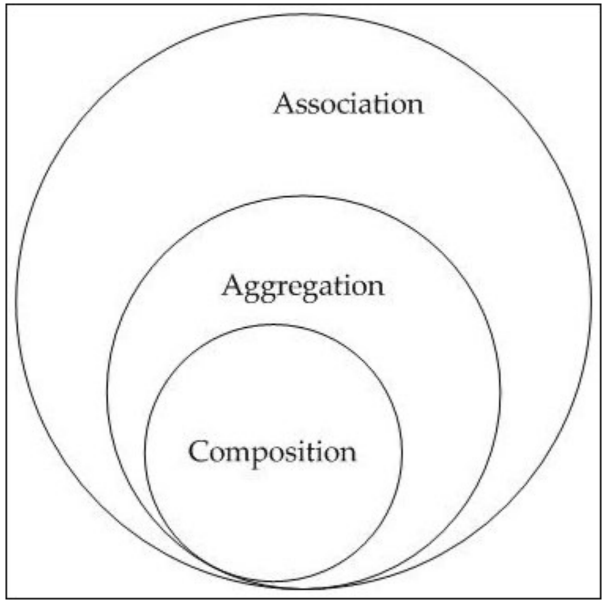
Association, Composition, Aggregation

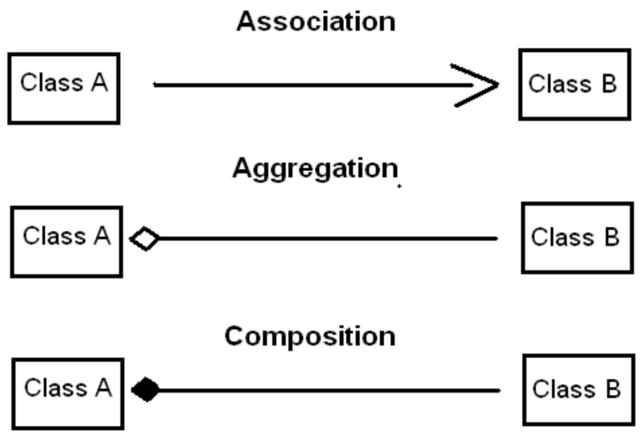
Object Oriented Programming | 2024 Spring
Practice 8

Presented by Tarlan Ahadli
Supervised by Prof. Teréz Anna Várkonyi

What is Class Relationship?

- Any relationship between object of two classes
  - one-to-one
  - one-to-many
  - many-to-one
  - many-to-many





Association

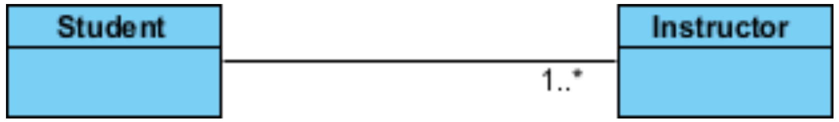
- **Definition**: A relationship where objects are independent and don’t affect each other's lifecycle. There’s no owner.
- **Navigation**: Can be unidirectional (one-way communication) or bidirectional (both ways).
- Example : Teacher – Student | Doctor – Patient | No one contains, no one.
    - Teacher has a student
    - Student has a teacher

Association | Teacher - Student

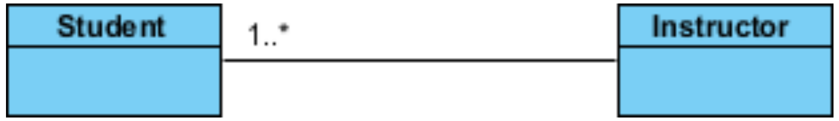
- Multiple students can associate with single teacher and single student can associate with multiple teachers
- There is no ownership between the objects and both have their own life-cycle.
- Both can be created and deleted independently.

Association | Teacher – Student | Diagram

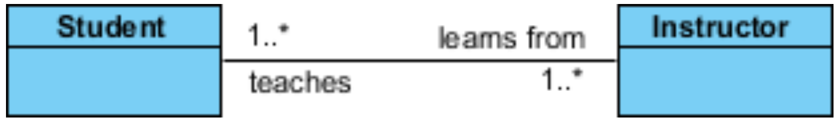
- A single student can associate with multiple teachers:



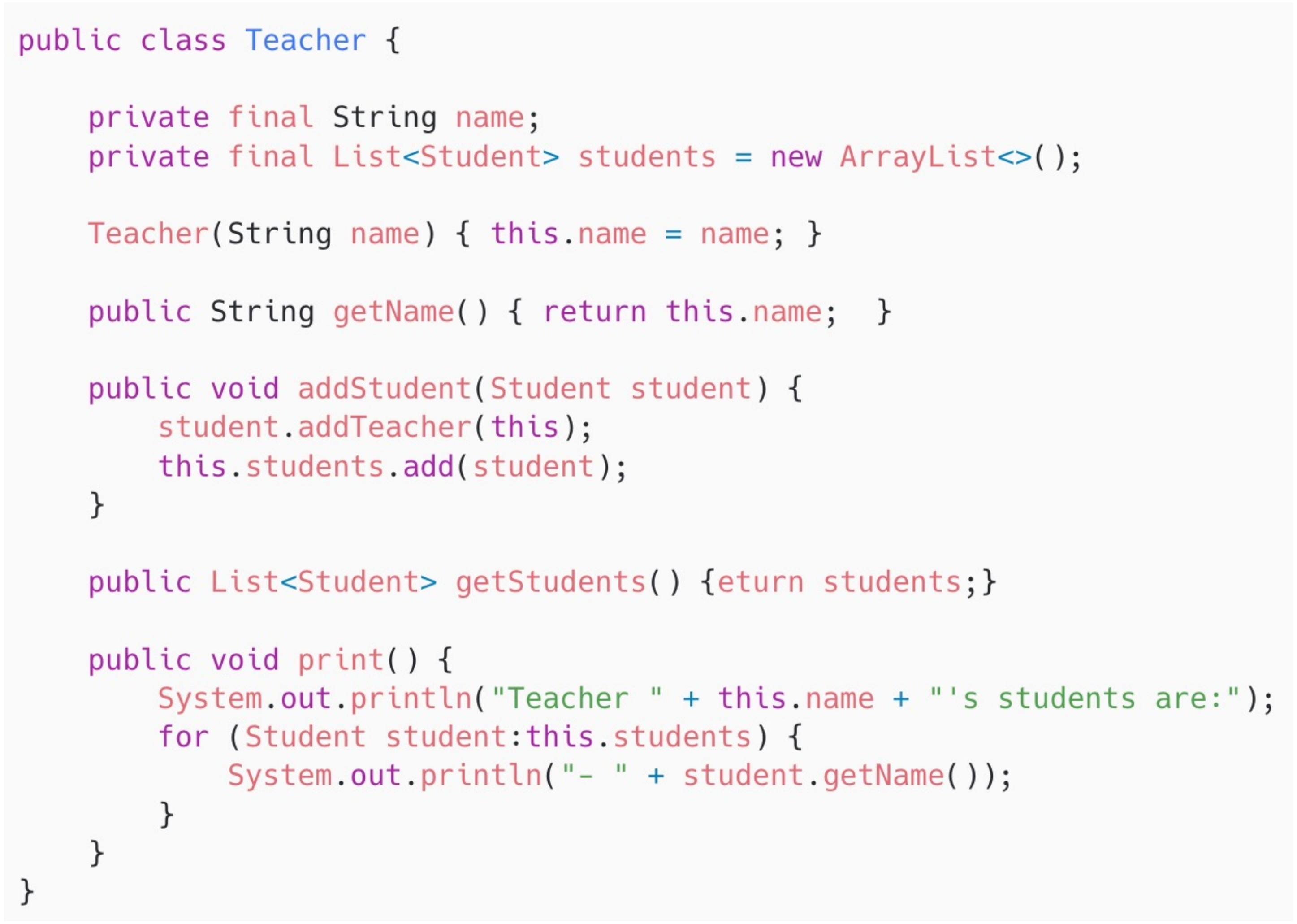
- A single teacher can associate with multiple students



- We can also indicate the behavior of an object in an association (i.e., the role of an object) using role names.

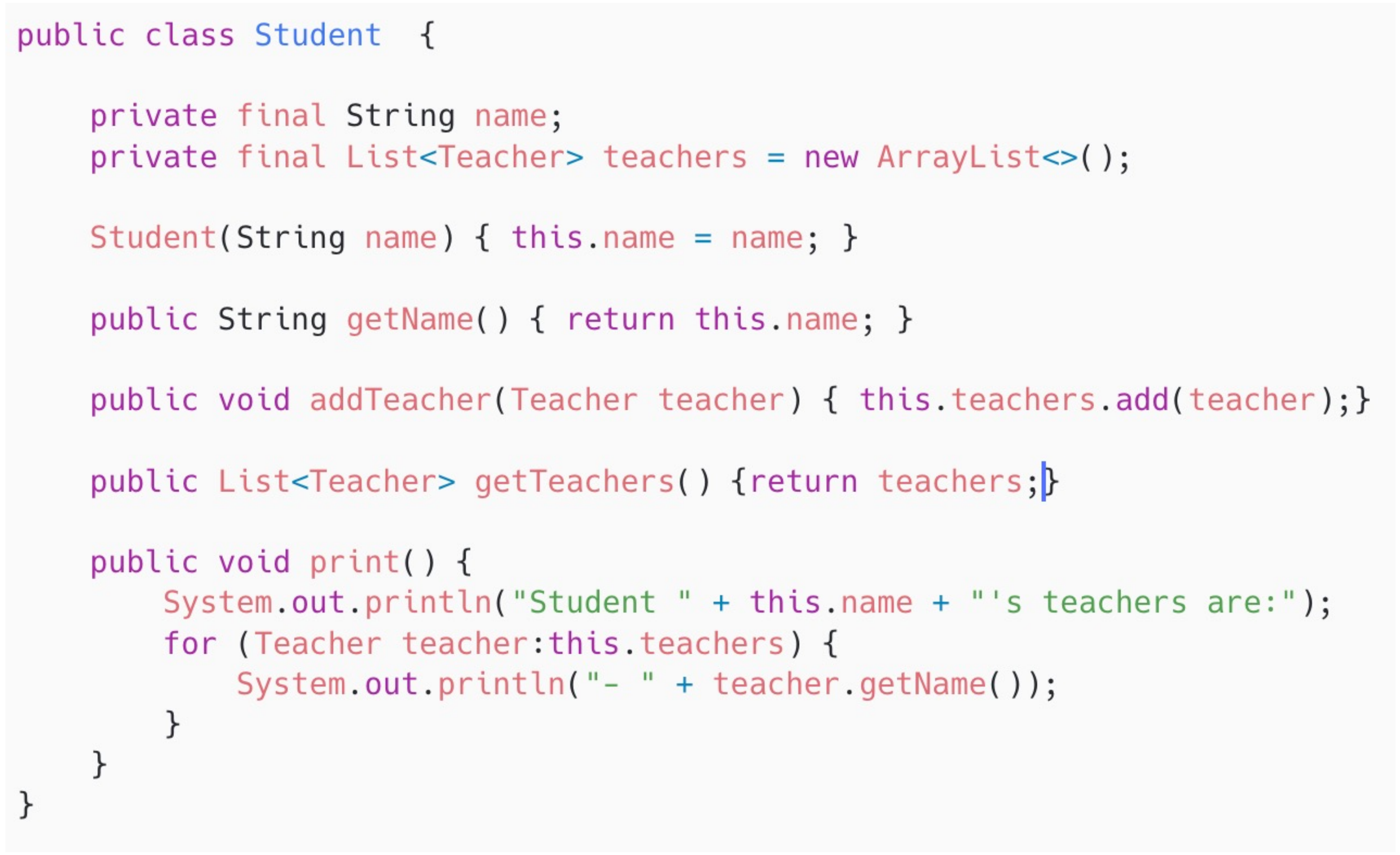


Association | Teacher – Student | C#



```
public class Teacher {

    private final String name;
    private final List<Student> students = new ArrayList<>();

    Teacher(String name) { this.name = name; }

    public String getName() { return this.name;  }

    public void addStudent(Student student) {
        student.addTeacher(this);
        this.students.add(student);
    }

    public List<Student> getStudents() {eturn students;}

    public void print() {
        System.out.println("Teacher " + this.name + "'s students are:");
        for (Student student:this.students) {
            System.out.println("- " + student.getName());
        }
    }
}
```

Association | Teacher – Student | C#



```
public class Student  {

    private final String name;
    private final List<Teacher> teachers = new ArrayList<>();

    Student(String name) { this.name = name; }

    public String getName() { return this.name; }

    public void addTeacher(Teacher teacher) { this.teachers.add(teacher);}

    public List<Teacher> getTeachers() {return teachers;}

    public void print() {
        System.out.println("Student " + this.name + "'s teachers are:");
        for (Teacher teacher:this.teachers) {
            System.out.println("- " + teacher.getName());
        }
    }
}
```

Association | Teacher – Student | C#

```
public class Association {

    public static void main(String[] args) {
        Teacher teacher1 = new Teacher("Dr. Jhon");
        Teacher teacher2 = new Teacher("Prof. Mark");

        Student student1 = new Student("Ben");
        Student student2 = new Student("Jack");

        teacher1.addStudent(student1);
        teacher1.addStudent(student2);

        teacher2.addStudent(student2);


        teacher1.print();
        teacher2.print();
        student1.print();
        student2.print();

    }
}
```

Aggregation

- **Definition**: A "has-a" relationship where parts and whole can exist independently.
- **Ownership**: Parts do not belong exclusively to the whole; deleting the whole doesn’t delete the parts.
- Example: Car-Engine | Car contains engine | But engine can be removed and used somewhere else
  - Car has an engine – ~~Engine has a car~~
  - Library has a book – ~~Book has a library~~

Composition

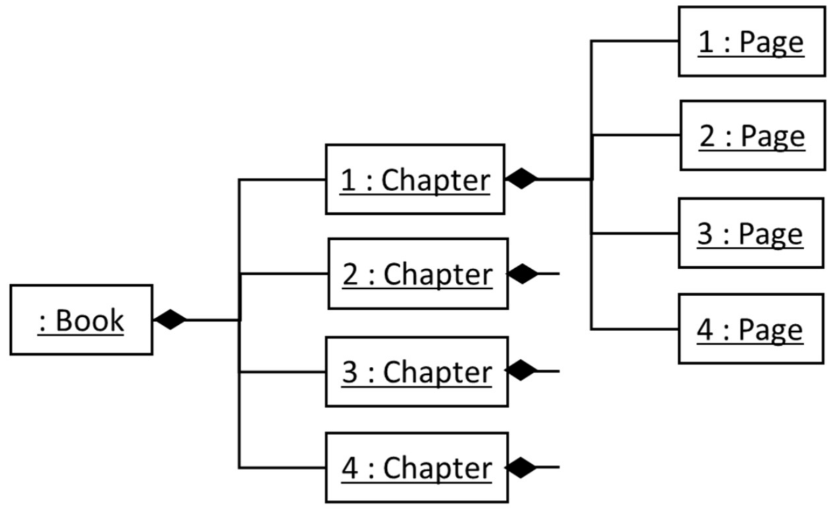
- **Definition**: A strong "has-a" relationship with dependent lifecycles between whole and parts.
- **Ownership**: Whole owns the parts; destroying the whole destroys the parts.
- Example:
  - House and rooms—rooms do not exist without the house.
  - Human and hand – hand doesn’t exist without human

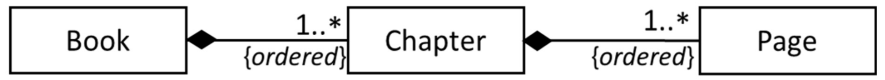
Sum Up

In essence, "association" is a general term used to describe a situation in which one class makes use of the functionalities provided by another class. We define it as "composition" when a parent class object owns a child class object, and this child class object cannot exist meaningfully without its parent class object. If the child class can exist independently, then the relationship is termed "aggregation."

Problem

- One book consists of at least one chapter, one chapter contains minimum one page. Give a class and a sample object diagram for the task!


: Book
1 : Chapter
2 : Chapter
3 : Chapter
4 : Chapter
1 : Page
2 : Page
3 : Page
4 : Page


Book
1..*
{ordered}
Chapter
1..*
{ordered}
Page