Class Diagrams

Christian Rodríguez Bustos

Edited by Juan Mendivelso

Object Oriented Programming

Agenda

1. Modeling Classes

2. Completing the exercise

1. Modeling Classes

1.1 UML

1.2 UML Class Diagram

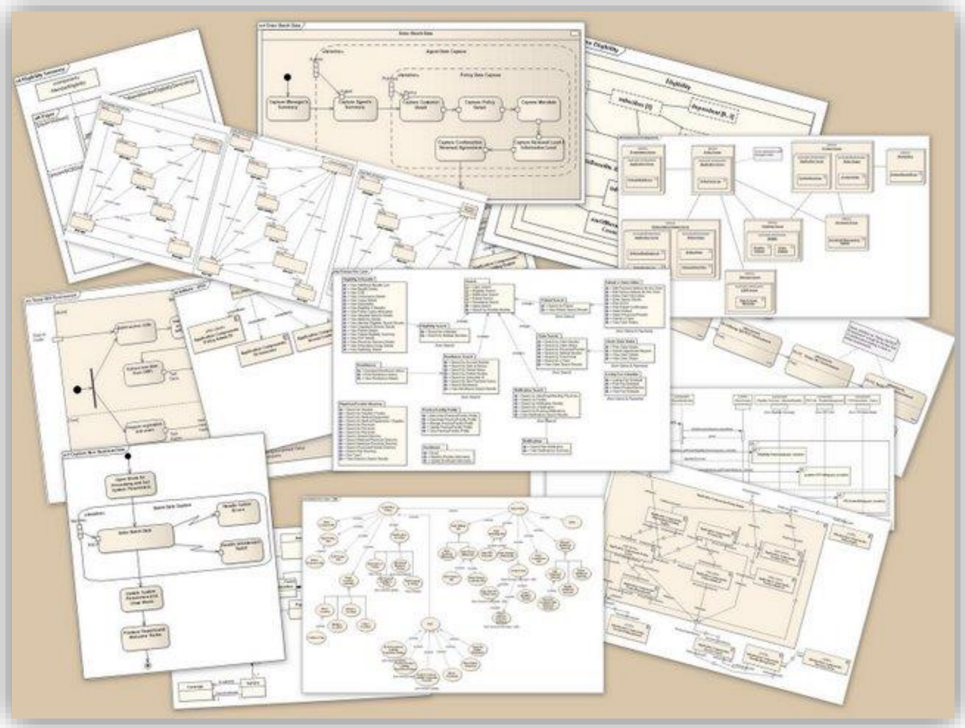
1.3 Relationship between objects: A closer approach

1.1 UML

UML is a standardized general-purpose modeling language

Unified Modeling Language (**UML**) is a standardized general-purpose modeling language in the field of **object-oriented software engineering**

UML is a standardized general-purpose modeling language



There are several diagrams in UML for **modeling object oriented systems**

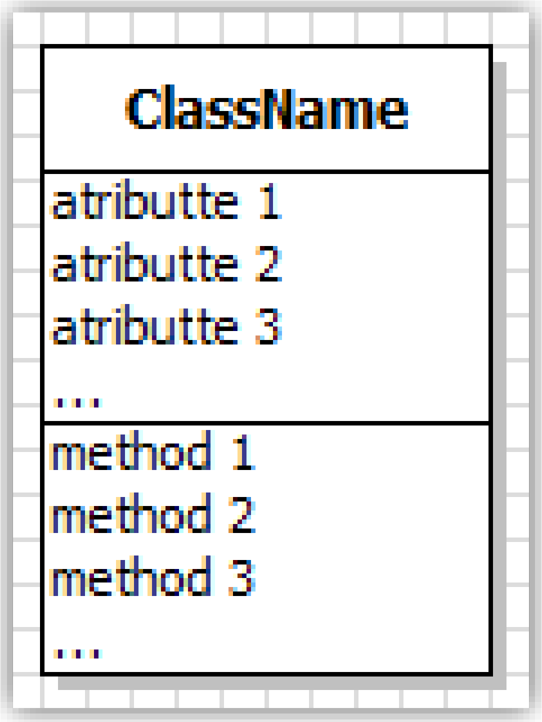
Activity diagrams and **class diagrams** are two examples

1.2 UML Class Diagram

UML Class diagram

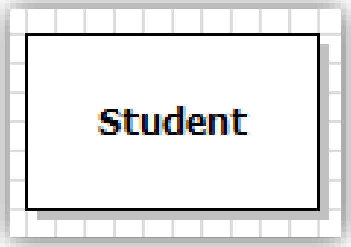
Describes the static structure of a system showing

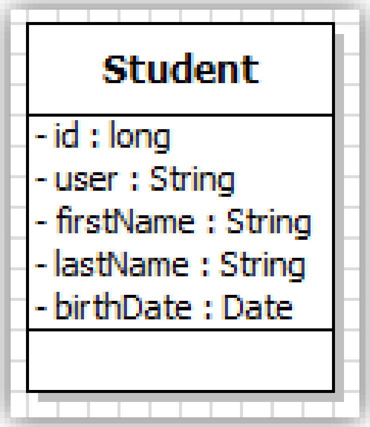
- Classes:
  - Name
  - Attributes
  - Methods
- Relationships between classes

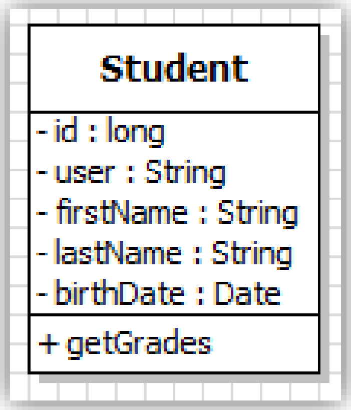


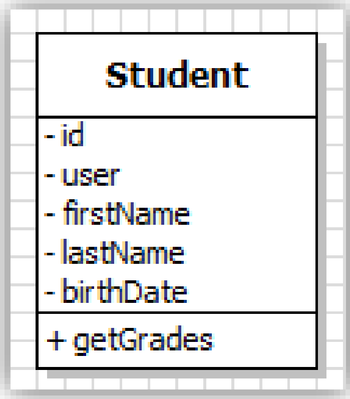
Showing classes

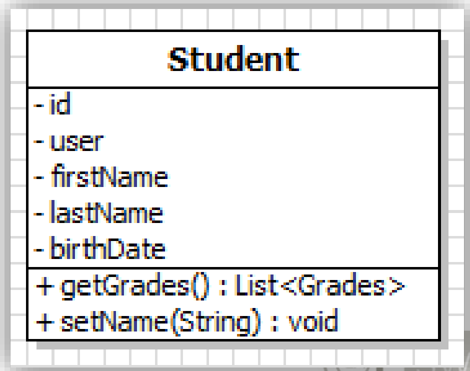
Classes can be shown at different detail level



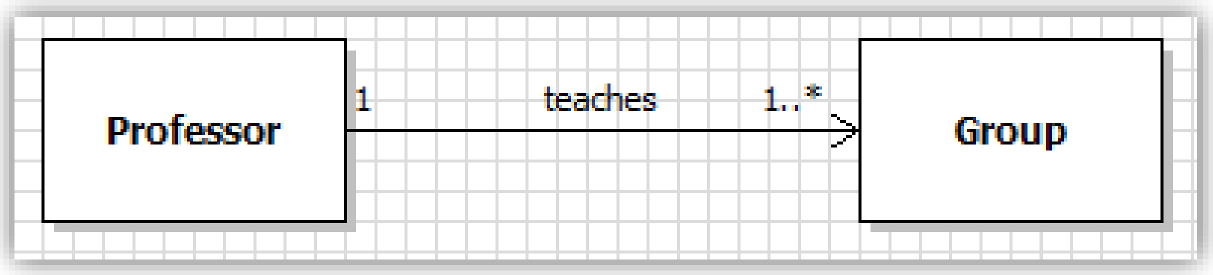






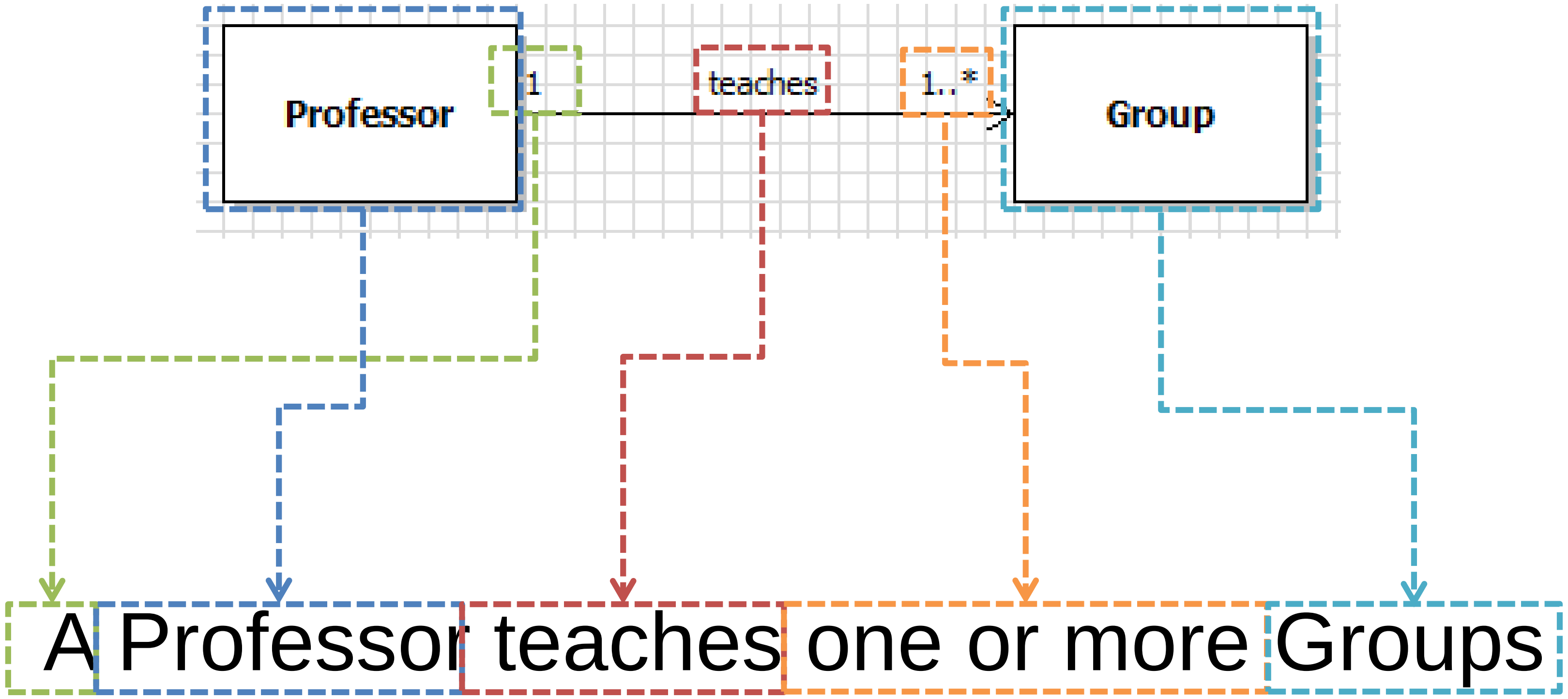


Showing relationships between classes



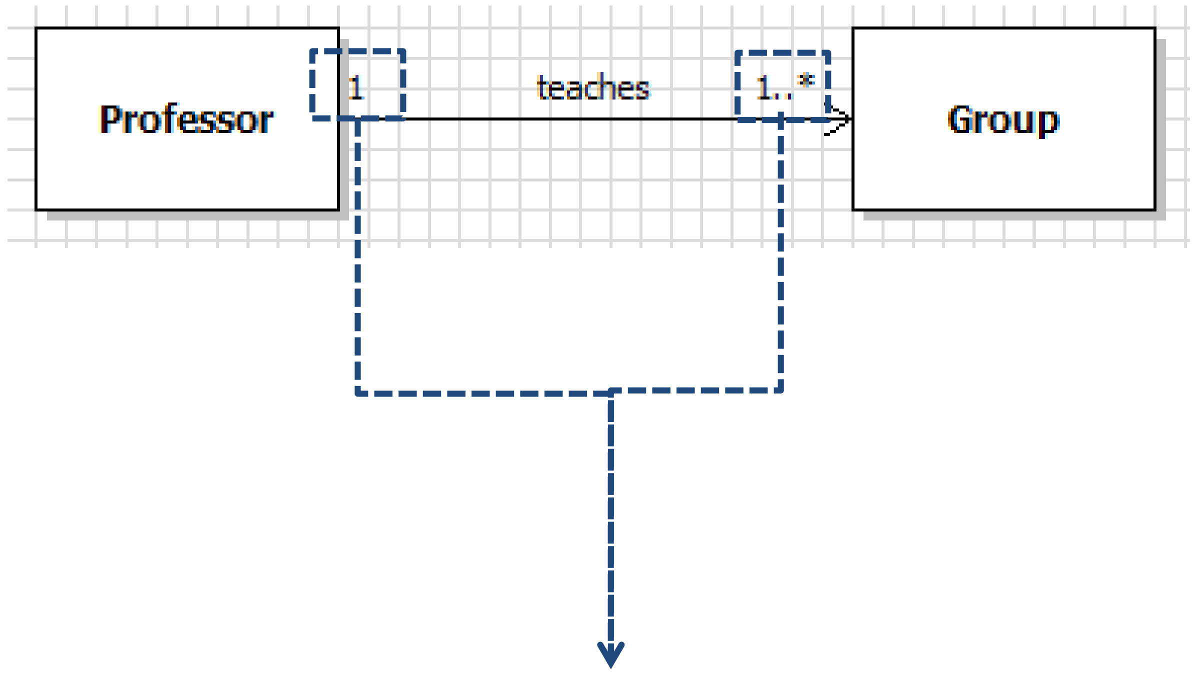
A Professor teaches one or more Groups

Showing relationships between classes



A Professor teaches one or more Groups

Showing relationships between classes



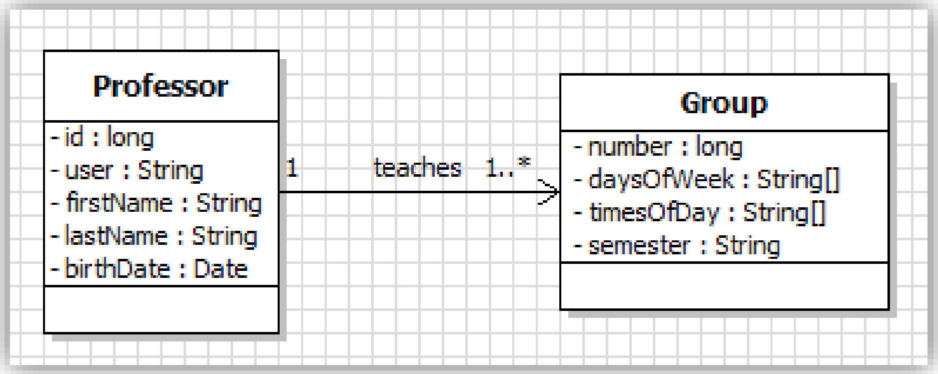
Multiplicity

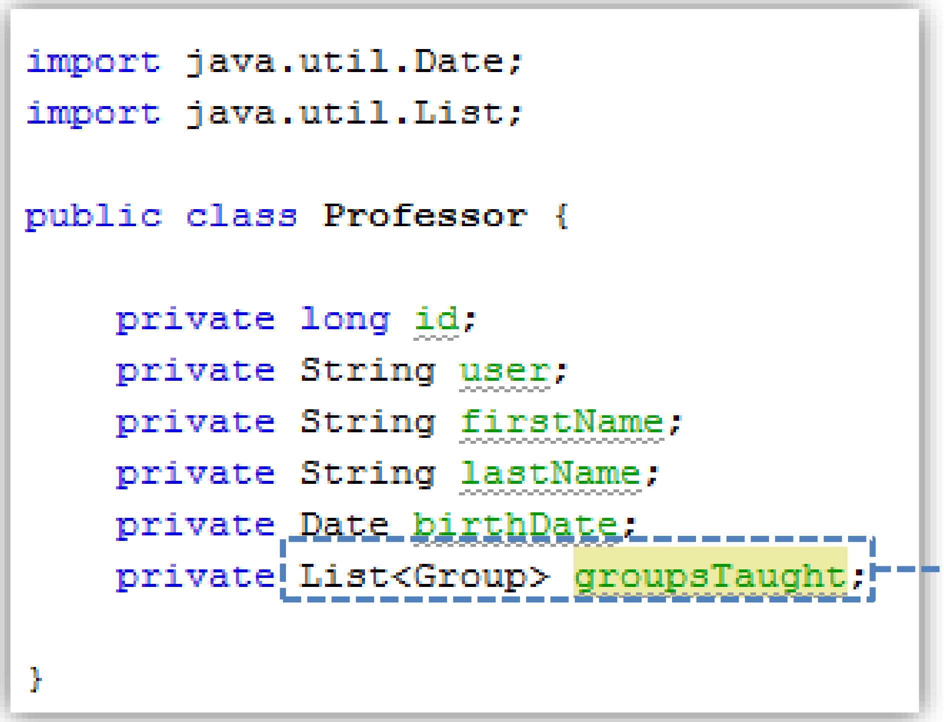
Multiplicity

Is the number of objects that participate in the relationship

0..1	No instances, or one instance (optional, may)
1	Exactly one instance
0..* or * or 0..n	Zero or more instances
1..*	One or more instances (at least one)

From model to code



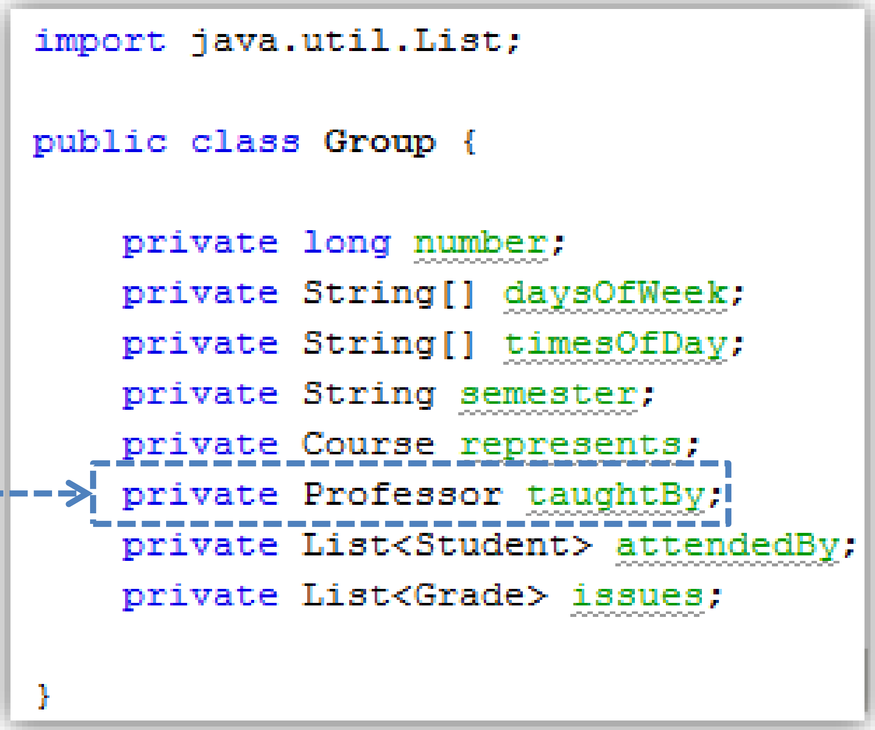


```
import java.util.Date;
import java.util.List;

public class Professor {

    private long id;
    private String user;
    private String firstName;
    private String lastName;
    private Date birthDate;
    private List<Group> groupsTaught;

}
```

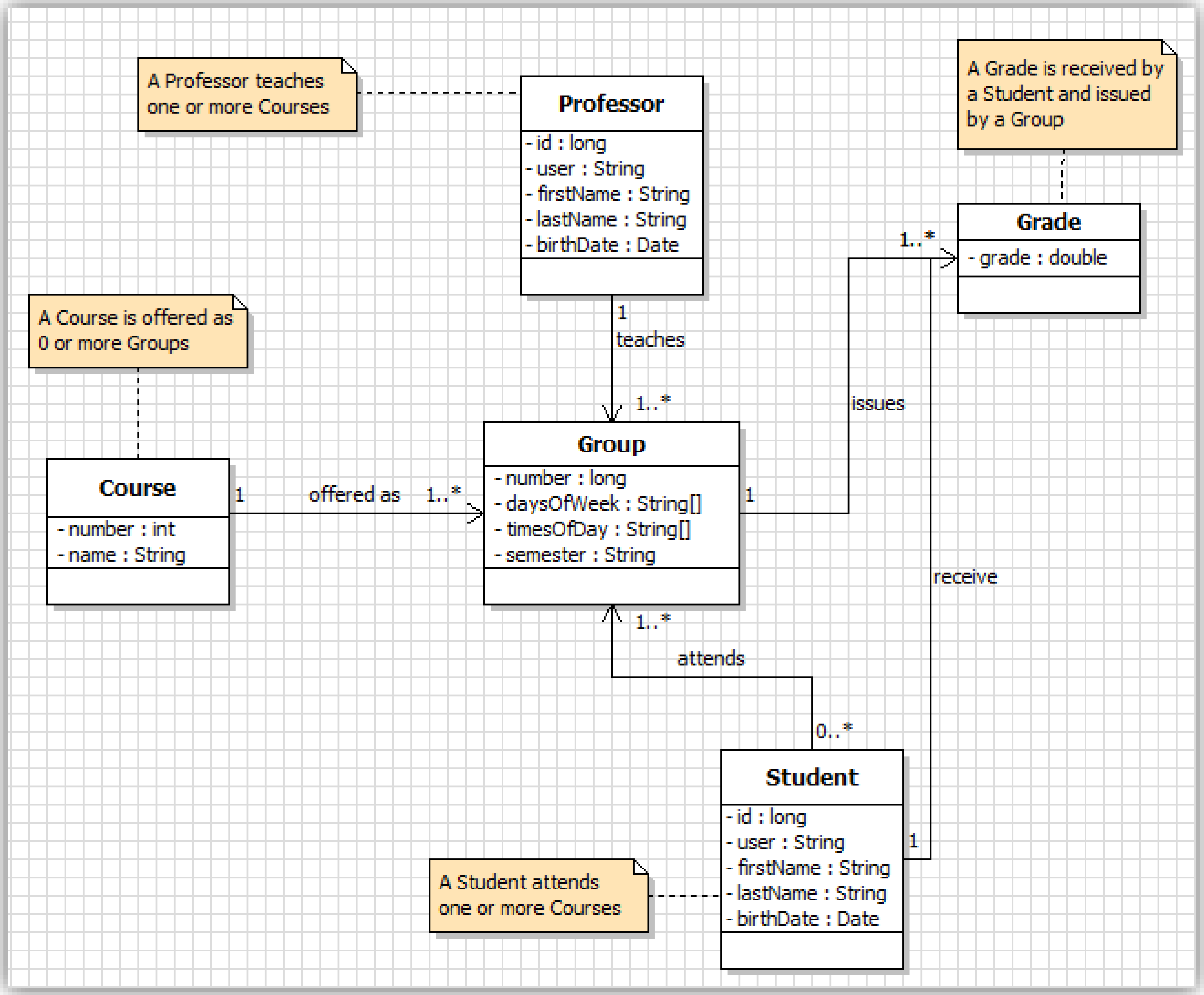


```
import java.util.List;

public class Group {

    private long number;
    private String[] daysOfWeek;
    private String[] timesOfDay;
    private String semester;
    private Course represents;
    private Professor taughtBy;
    private List<Student> attendedBy;
    private List<Grade> issues;

}
```


A Professor teaches one or more Courses
Professor
- id : long
- user : String
- firstName : String
- lastName : String
- birthDate : Date
A Grade is received by a Student and issued by a Group
Grade
- grade : double
1..*
1
teaches
1..*
issues
A Course is offered as 0 or more Groups
Course
- number : int
- name : String
1
offered as
1..*
Group
- number : long
- daysOfWeek : String[]
- timesOfDay : String[]
- semester : String
1
receive
1..*
attends
0..*
Student
- id : long
- user : String
- firstName : String
- lastName : String
- birthDate : Date
1
A Student attends one or more Courses

1.3 Relationship between objects: A closer approach

1.3.1 Association and Links

1.3.2 Aggregation and Composition

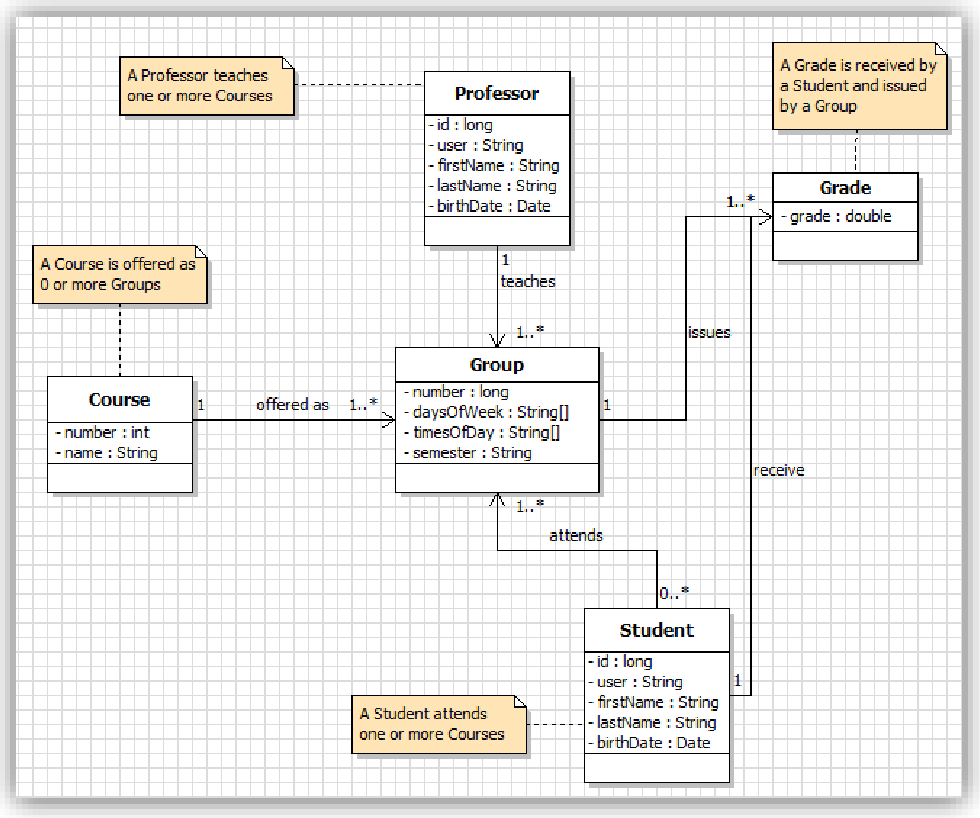
1.3.3 UML Notation

1.3.1 Association and Links

Associations are structural relationship

Associations are structural relationship that exists between **classes**

- A **Professor** *teaches* one or many **Groups**
- A **Course** *is offered* as one or many **Groups**
- Zero or many **Students** *attends* one or many **Groups**



Links

Links are relations between two specific **objects** (*instances*)

Association: ______ *attends at* ______

A student Any group

Link: Bruce Wayne *attends at* Math 3B

A ***specific*** student A ***specific*** group

1.3.2 Aggregation and Composition

Aggregation and Composition

Aggregation: Is a specific type of association, is represented typically by "*consists of*", "*is composed of*" and "*has a*"

A Team ***is composed by*** one or more Students

A Department ***is composed of*** one or more Professors

A Club ***has*** Members

Composition: Is a strong form of aggregation, in which **the "parts" cannot exist without the "whole."**

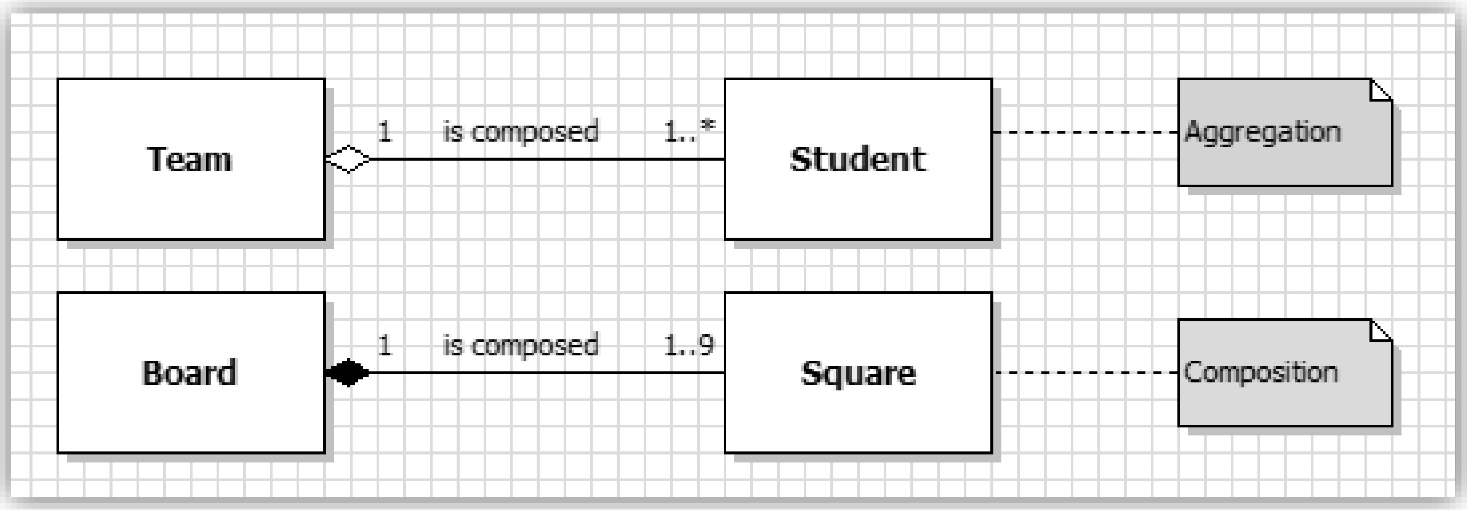
A Building ***is composed by*** one or more Rooms

A University ***is composed of*** Departments

A Board ***is composed of*** Squares

1.3.3 UML Notation

UML notation



Aggregation is depicted as an **unfilled diamond**

Composition is depicted as a **filled diamond** and a solid line.

2. Completing the Exercise

Now is your turn for using your model

1. Abstract the model to submit the grades of a student in the SIA (Classes, behaviors, attributes, etc)
2. **Create a Java project in NetBeans or Eclipse**
3. **Create the Java classes of the proposed model**
4. **Encapsulate the classes**
5. **Do all setters and getters for all classes**
6. **Test your classes with the following test class:**

   Test Class
7. **Do more tests creating new objects of other classes.**

References

- [Barker] J. Barker, *Beginning Java Objects: From Concepts To Code*, Second Edition, Apress, 2005.
- [Deitel] H.M. Deitel and P.J. Deitel, *Java How to Program*, Prentice Hall, 2007 - 7th ed.
- [Sierra] K. Sierra and B. Bates, *Head First Java*, 2nd Edition, O'Reilly Media, 2005.