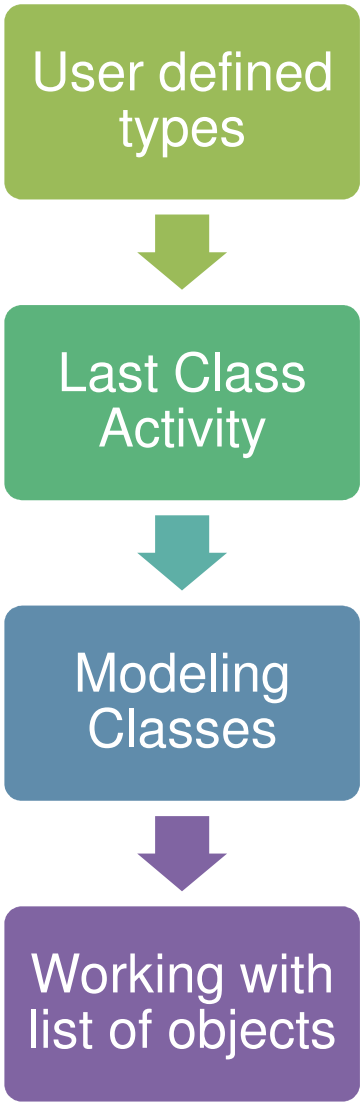
Objects & Classes II

Christian Rodríguez Bustos

Object Oriented Programming

Agenda


User defined types
Last Class Activity
Modeling Classes
Working with list of objects

User defined types

Data redundancy & Data integrity

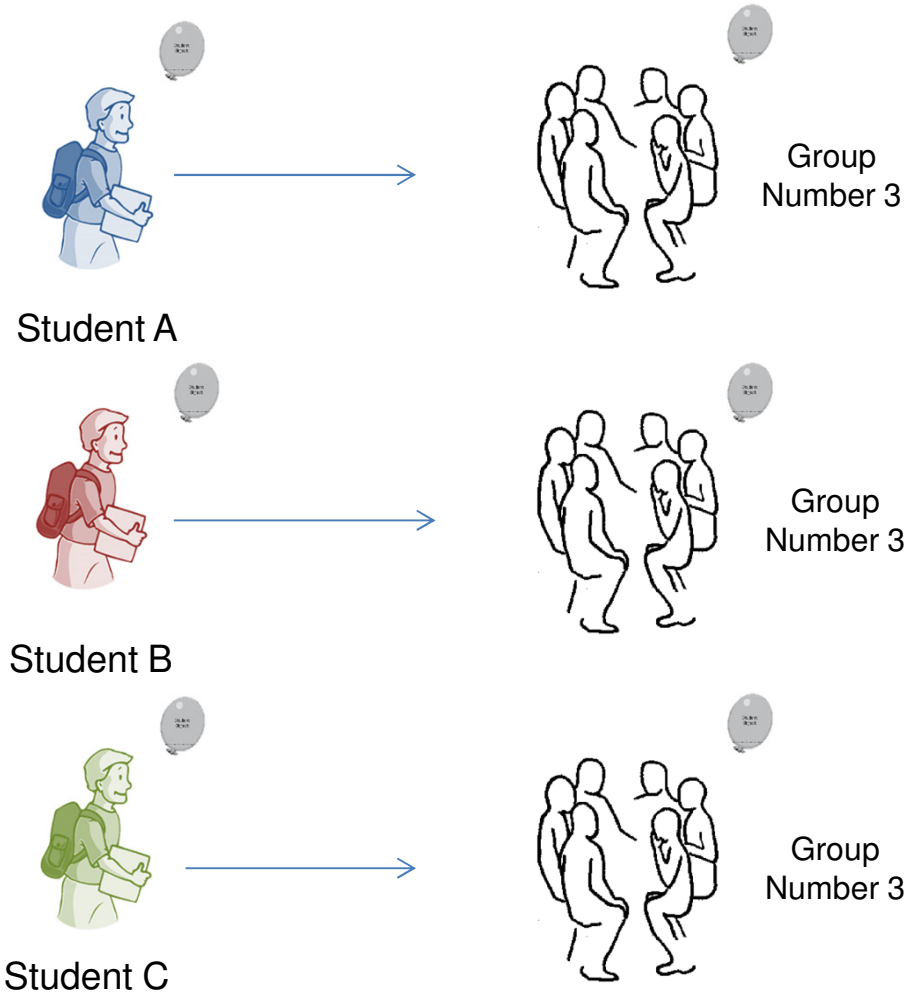
Objects as attributes

Do you remember the Student Class?

Attribute	Type
Id	Long
User	String
First Name	String
Last Name	String
Birth Date	Date
Group Taught	**int**

```java
public class Student {

    private long id;
    private String user;
    private String firstName;
    private String lastName;
    private Date birthDate;
    private int groupNumber;

}
```

Data redundancy !!!

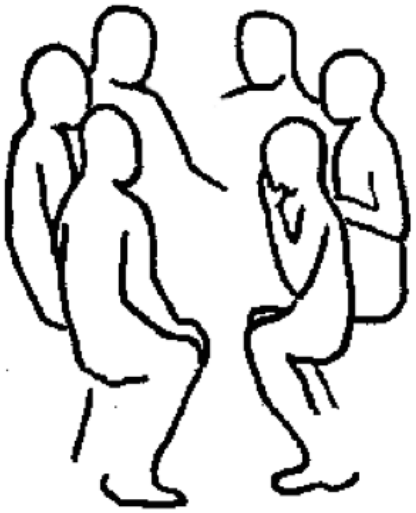



```
public class Student {

    private long id;
    private String user;
    private String firstName;
    private String lastName;
    private Date birthDate;
    private int groupNumber;

}
```

Data redundancy !!!

All students are linked to different groups

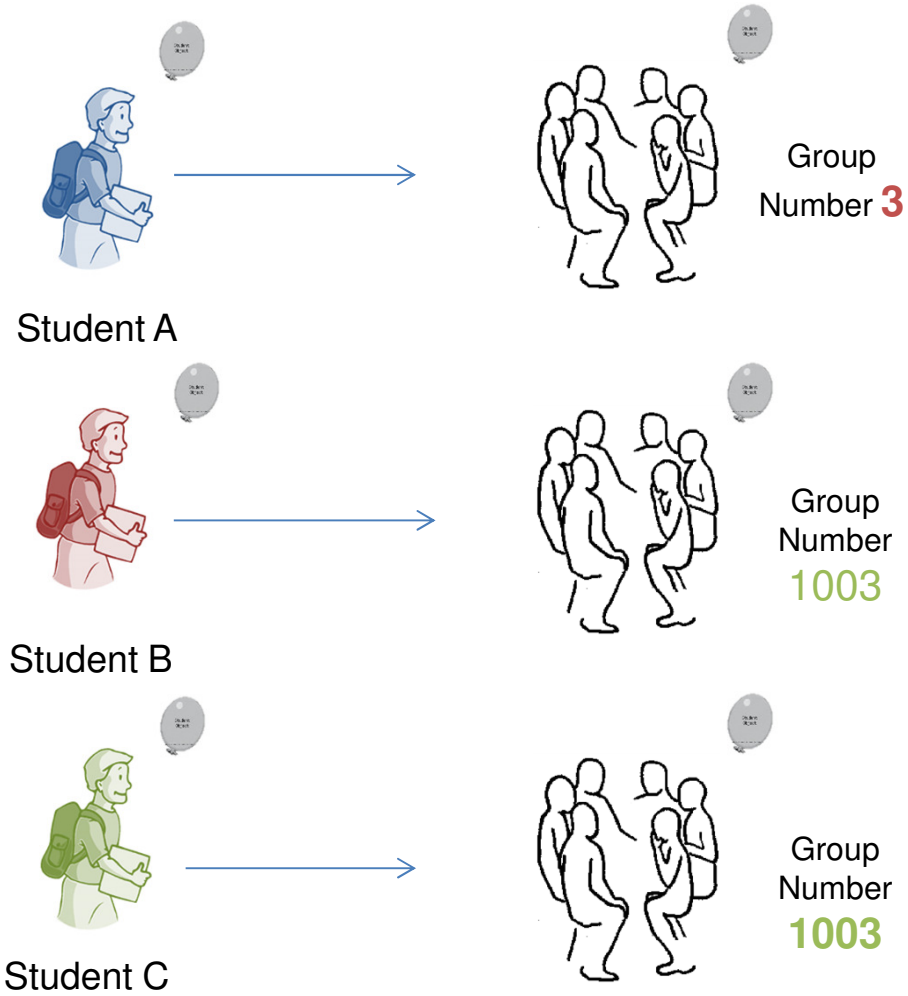
Changing enviroment



The old group number is 3

The new group number is 1003

Loss of data integrity




```
public class Student {

    private long id;
    private String user;
    private String firstName;
    private String lastName;
    private Date birthDate;
    private int groupNumber;

}
```

Loss of data integrity

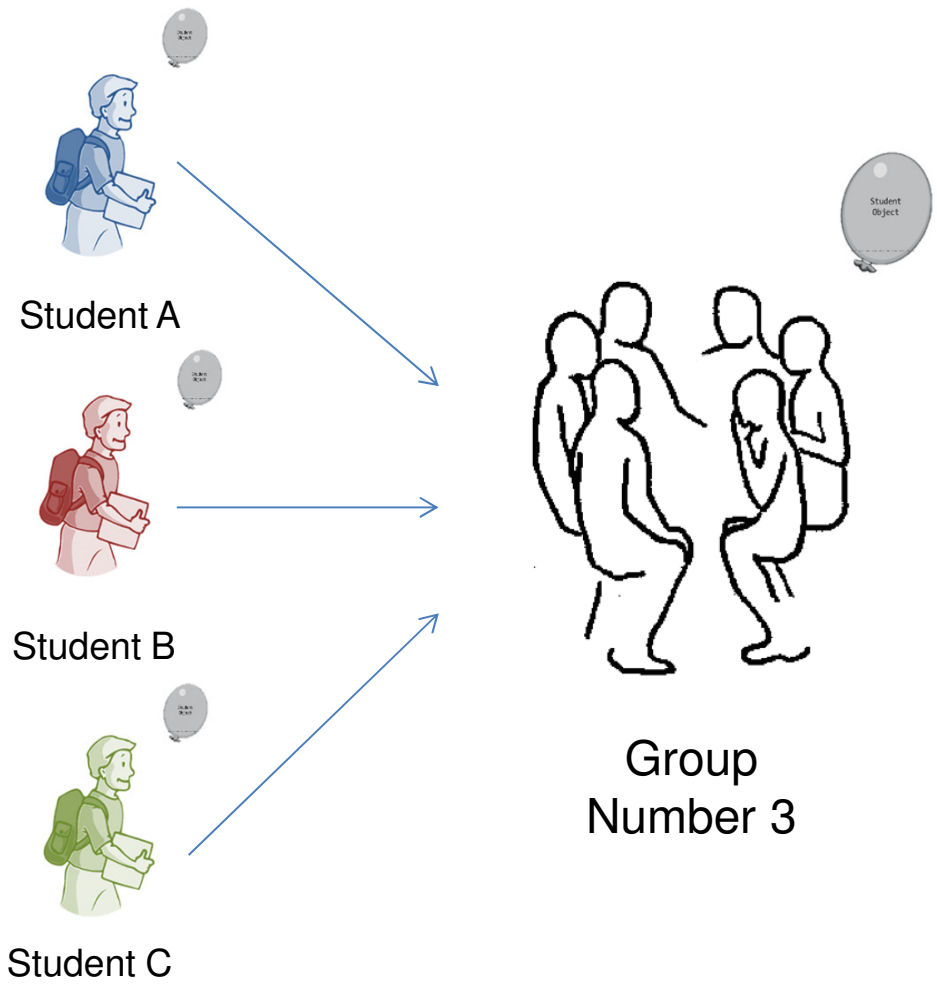
All students are using different instances and probably do not work with an update object

Objects as attributes

Solution is

Objects as attributes

Avoid data redundancy




```
public class Student {

    private long id;
    private String user;
    private String firstName;
    private String lastName;
    private Date birthDate;
    private Group group;

}
```

Avoid data redundancy and loss of data integrity

All students can use the same Group instance

Last Class activity

Class List

Class Definitions

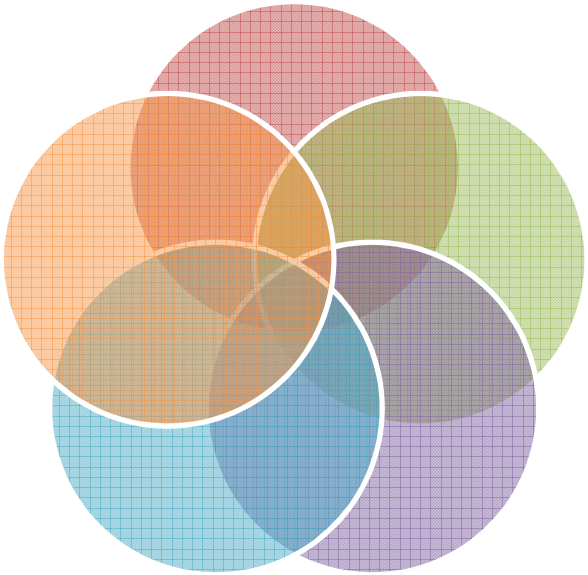
Last Class activity

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

Class List

Student

Grade



Professor

Group

Course

Class Student

Attribute	Type
Id	Long
User	String
First Name	String
Last Name	String
Birth Date	Date
Attends	List of Groups

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

public class Student {

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

}
```

List of **User define objects** can be used as attributes

Class Professor

Attribute	Type
Id	Long
User	String
First Name	String
Last Name	String
Birth Date	Date
Groups Taught	List of Groups

```java
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;

}
```

Class Course

Attribute	Type
Number	Long
Name	String
Offered As	List of Groups

```java
import java.util.List;

public class Course {

    private int number;
    private String name;
    private List<Group> offeredAs;

}
```

Class Grade

Attribute	Type
Student	Student
Group	Group
Grade	double

```
public class Grade {

    Student student;
    Group group;
    double grade;

}
```

Class Group

Attribute	Type
Number	Long
Days Of Week	String Array
Times Of Day	String Array
Semester	String
Represents	Course
Taught By	Professor
Attended By	List of Student
Issue	List of Grades

```java
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;

}
```

Modeling Classes

UML

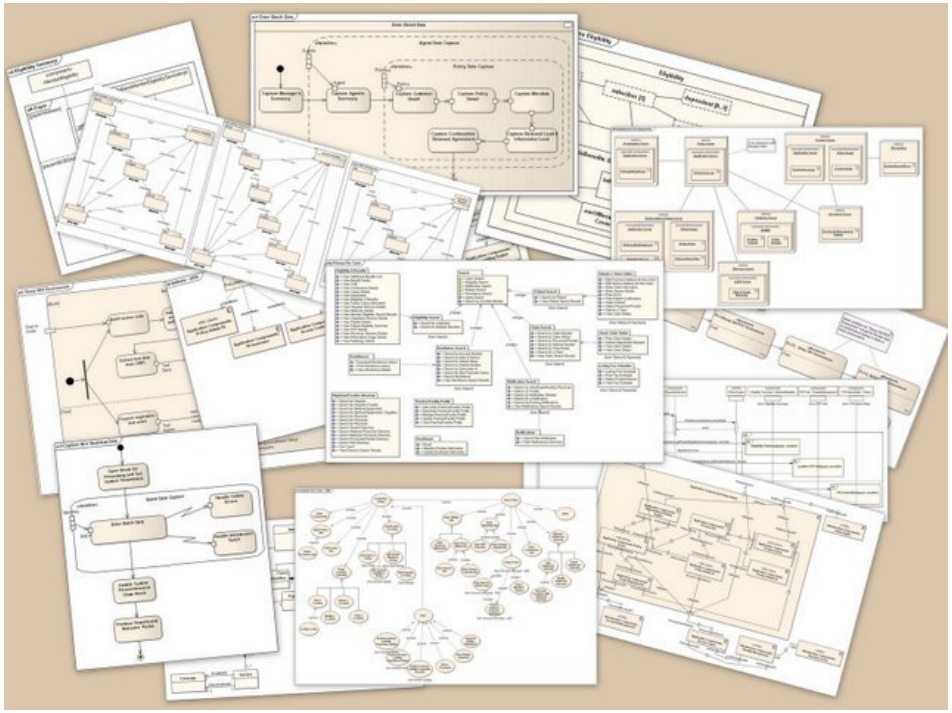
UML Class Diagram

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**

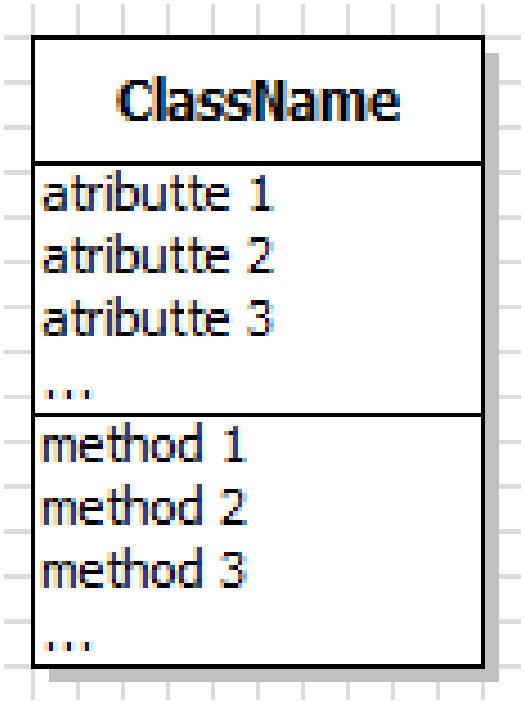


Activity diagrams and **class diagrams** are two examples

UML Class diagram

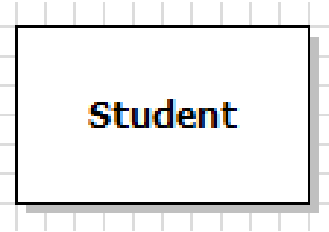
Describes the static structure of a system showing

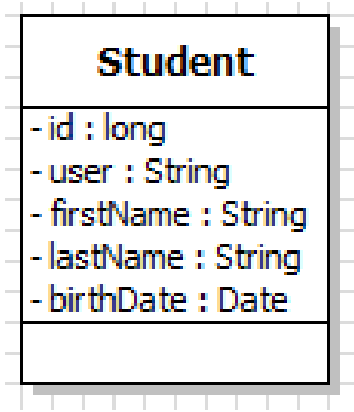
- Classes:
  - Name
  - Attributes
  - Methods
- Relationships between classes

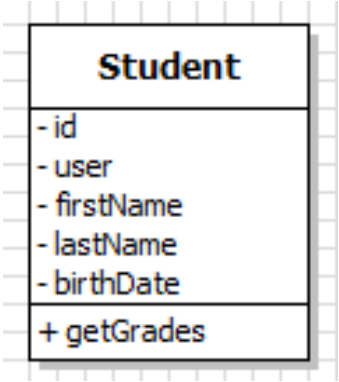


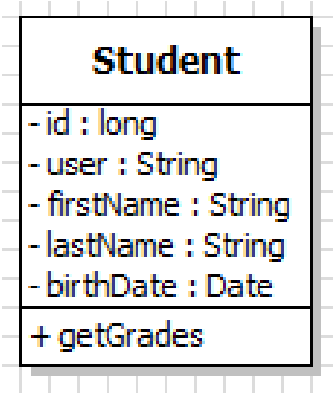
Showing classes

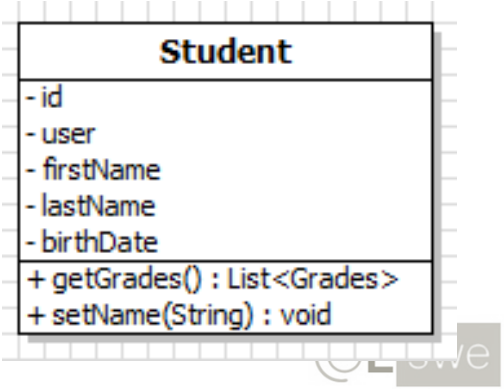
Classes can be shown at different detail level



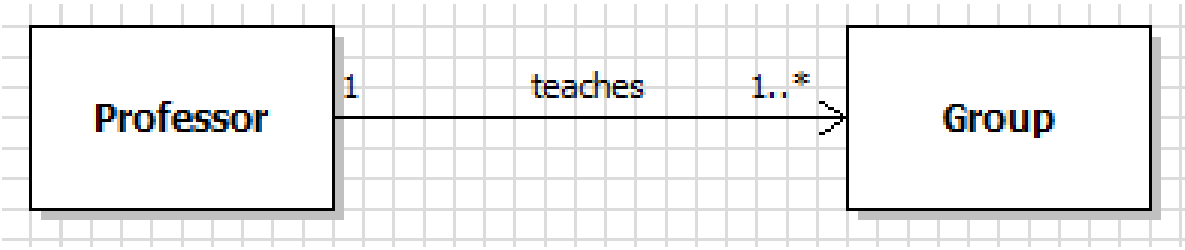






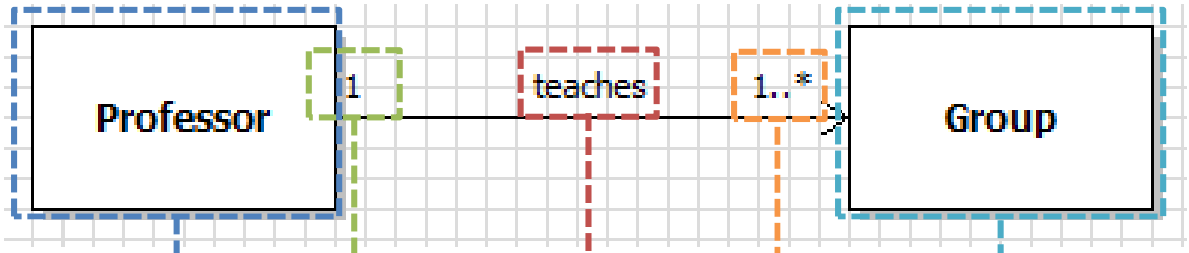


Showing relationships between classes



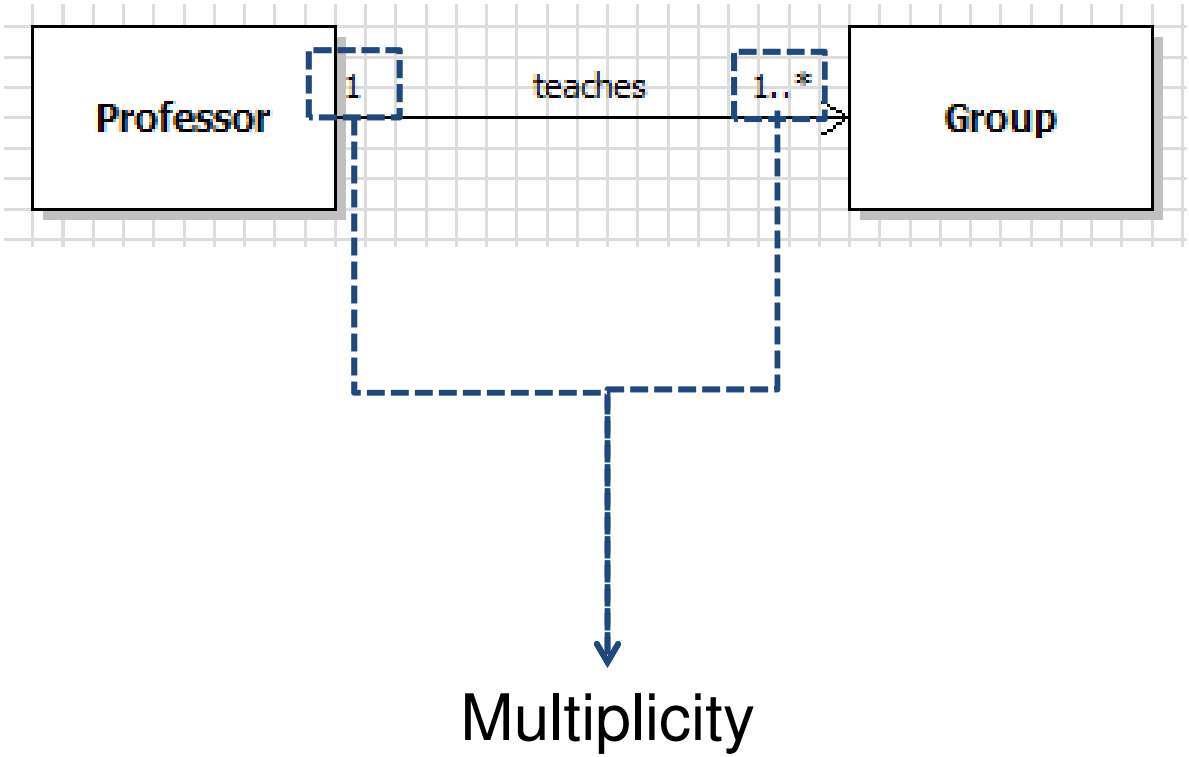
A Professor teaches one or more Groups

Showing relationships between classes



A Professor teaches one or more Groups

Showing relationships between classes

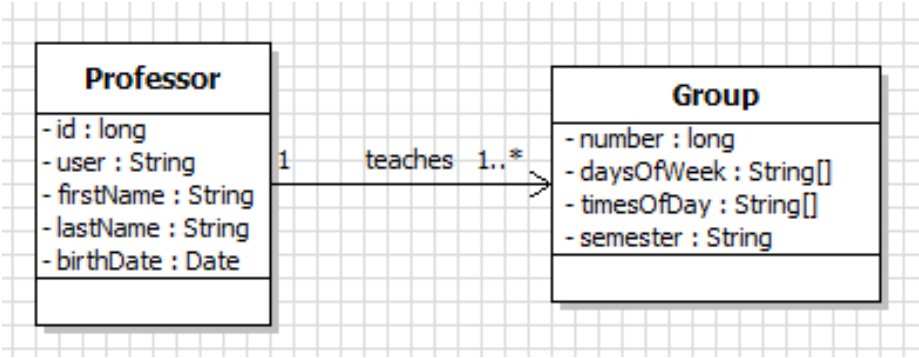


Multiplicity

Is the number of objects that participate in the relationship

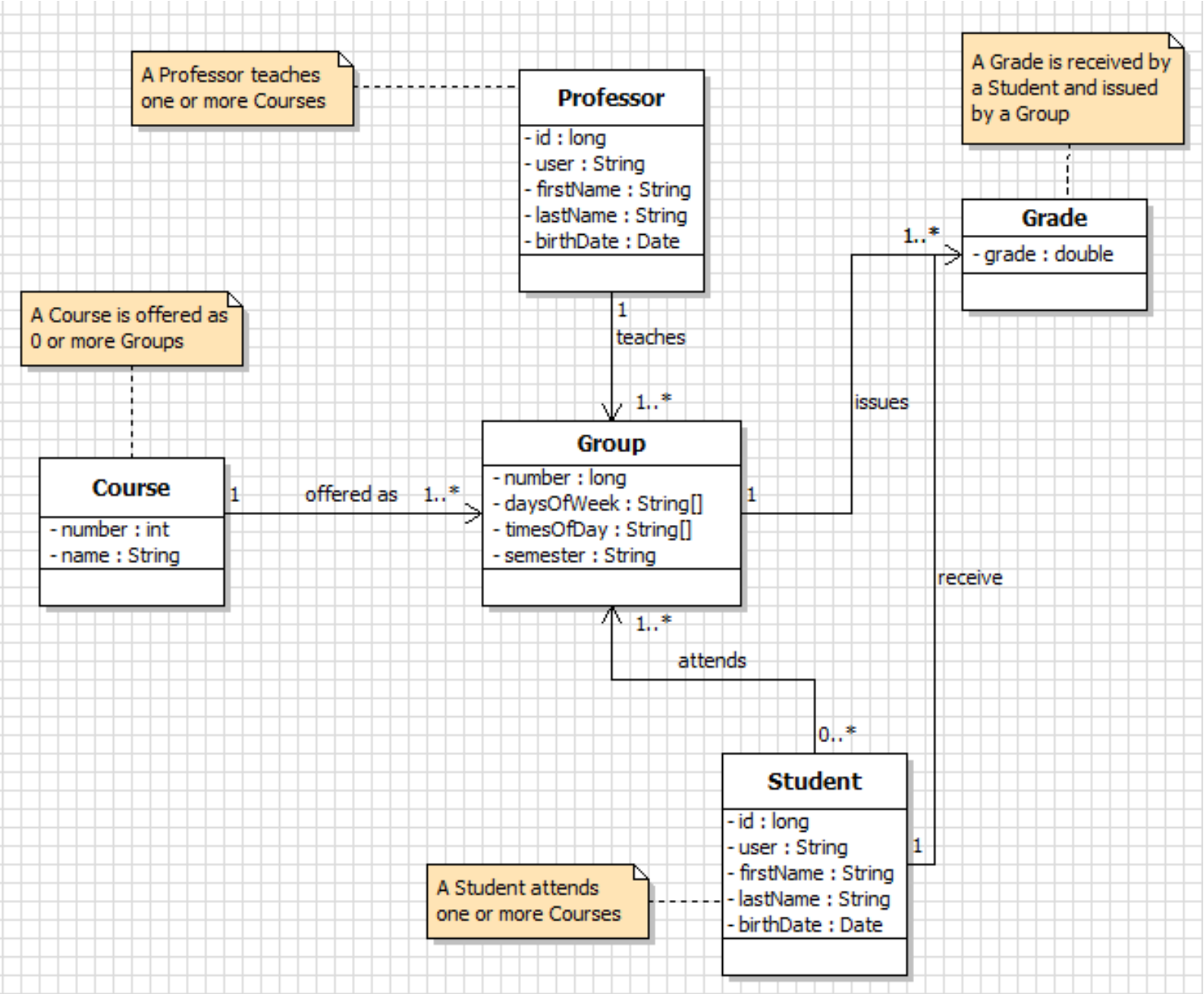
0..1	No instances, or one instance (optional, may)
1	Exactly one instance
0..* or * or 0..n	Zero or more instances
1..*	One or more instances (at least one)

From model to code




```java
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;

}
```

```java
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..*
A Course is offered as 0 or more Groups
1
teaches
1..*
issues
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

Working with list of objects

Array List Basics

Array List Basics

Instantiation

```
ArrayList<Student> students = new ArrayList<Student>();
ArrayList<Professor> professors = new ArrayList<Professor>();
ArrayList<Grade> grades = new ArrayList<Grade>();
```

```
students.add(student);
professors.add(professor);
grades.add(grade);
```

Adding elements

Array List Basics

Obtaining objects

```
student = students.get(0);
professor = professors.get(0);
grade = grades.get(0);
```

```
students.size();
professors.size();
grades.size();
```

Obtaining list size

Iterating through Array Lists

```
ArrayList<Grade> grades = new ArrayList<Grade>();
```

```
for (Iterator<Grade> it = grades.iterator(); it.hasNext();) {
    Grade grade = it.next();
    // Do Something Here
}
```

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**

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.