

Object-Oriented System Development Using UML, JAVA and Patterns
Professor Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 16
Class Diagram Exercises

Welcome to this session.

Over the last few sessions, we were trying to represent classes in UML diagrams, the different types of relations that exist between the classes and how to represent them on a UML diagram, and their translation to Java code. We saw that for most of the classes and relations, the translation to Java code is more or less systematic and that is the reason why many case tools can automatically generate code from the class diagram.

Given a problem, one of the important skill is to identify the classes from a text description and to identify the relations among them and represent them in a UML diagram. And after that we can generate code either automatically using a case tool or at least we can systematically write down the Java code corresponding to the class diagram.

Now, let's do few more exercises. In last class towards the end we have been giving very small sentences and trying to develop the skill to identify classes and their relations from the text description. Let's do a few more exercises and later we will look at larger problems and how to solve those that will look later, but right now look at small sentences and from the sentences identify the classes that are obvious in the sentence. Now let's get into exercises.

(Refer Slide Time: 2:29)

Identify Classes and Relations

- A square is a polygon
- Shyam is a student
- Every student has a name
- 100 paisa is one rupee
- Students live in hostels
- Every student is a member of the library
- A student can renew his borrowed books
- The Department has many students

The diagram shows two boxes labeled 'Student' and 'Hostel' connected by a red arrow pointing from 'Student' to 'Hostel'. The arrow is labeled 'live in' in red handwriting.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

A square is a polygon: The two classes here are square and polygon and the relationship between the square and polygon 'IS A' or in other words, a polygon is the general class and the square is the special class. A square is a special type of polygon and therefore, polygon is the base class and square is the derived class. So, it's a inheritance relation.

Shyam is a student: Here we must note that Shyam is a proper noun and therefore it is not a class. It is an object of the student class. Shyam is an object of the student class.

Every student has a name: Here name is not a class because the name has only one attribute, and we cannot associate any other attribute or methods with the name class and therefore, name itself is attribute of the student class. We cannot associate any meaningful methods with this name class and therefore it is an attribute and student is the class and name is a attribute of the class.

100 paisa is one rupee: Here paisa and rupee are the classes and rupee is composed of 100 paisa objects and since a rupee cannot have 99 or 101 paisa, the cardinality here is fixed and therefore it is a composition relation, a rupee consists of 100 paisa.

Students live in hostels: Here student and hostel are the two classes and 'live in' is the association relation between the student and the hostel.

Please note that the name of the class here is student not students, if we represent that in the UML diagram (shown in the above slide), we will represent that as the 'student' class. So the association relation here is 'live in' and each hostel many students live, the cardinality on the other side is by implicitly '1'.

Every student is a member of the library: So, here Student is a class and library is another class and 'member of' is the association relationship between the student and the library.

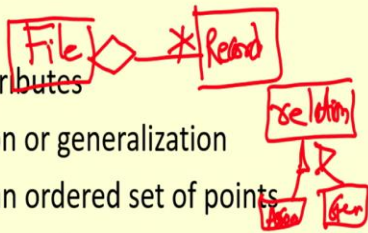
A student can renew his borrowed books: So, Book and student is the class. 'Renews' represent association relationship between student and borrowed books.



Department has many students: So here department and students are the class and the 'has' here indicates aggregation. The department has many students, so department is an aggregate of many students. We used a composition for the rupee and paisa because once a rupee is created, the 100 paisa is created, but here the students are free to join, some students may leave and so on and therefore this is the aggregation relation. We can draw the corresponding UML diagram, I hope everybody will be able to draw it.

(Refer Slide Time: 8:28)

Identify Classes & Relations

- A country has a capital city
- A dining philosopher uses a fork
- A file is an ordinary file or a directory file
- Files contain records
- A class can have several attributes
- A relation can be association or generalization
- A polygon is composed of an ordered set of points
- A programmer uses a computer language on a project



2

Now let us do some more problems.

A country has a capital city: So, here country is a class and capital city is a class and ‘has a’ indicates an aggregation or composition relation and since the country has exactly one capital city and therefore we will use a composition relation here. Country is composed of a capital city.

A dining philosopher uses a fork: Here philosopher is the class and fork is a class and ‘uses’ represent dependency. Philosopher dependent on instances of fork.

A file is an ordinary file or a directory file: the ‘is an’ is a giveaway for inheritance relation and therefore a file is a general concept and then we can have special files like ordinary file and directory files.

Files contain records: The contain is an aggregation and we draw it as a file contain many records. We can just draw it here; we will draw here file contain many records (in the above slide).

Now, let us look at the other exercises.

A class can have several attributes: Here attribute is not a class, it is an attribute of the class here. This is not an aggregation, but it is an attribute is a data member of the class.

A relation can be association or generalization: Here relation is a more general concept and association and generalization are special types of relation. Association is a relation and generalization is a relation and therefore, this is an inheritance relation. If we want to represent in the form of a UML diagram, we will write relation is the base class and there are two special types of relation, one is association and the other is generalization (shown in the above slide).

A polygon is composed of an ordered set of points: Here a polygon has set of points and therefore point is a class and polygon is a class and this is a composition relation, assuming that once we draw the polygon we don't change the number of points.

A programmer uses a computer language on a project: There is a dependency between 'programmer' and 'computer language' and association between 'computer language' and 'project'. So, there are three classes here, the 'programmer', 'computer language' and 'project'. We draw these three classes and draw the dependency relationship between programmer uses computer language and association between language and project.

(Refer Slide Time: 14:22)

Exercise 1: Draw UML Diagrams

Some persons keep animals as pets.

```
classDiagram
    class Person
    class Animal
    Person "*" -- "*" Animal : Keeps
```

The diagram shows two classes, 'Person' and 'Animal', represented by rectangles. A line connects them, labeled 'Keeps'. There is an asterisk '*' at the end of the line near 'Person' and another asterisk '*' near 'Animal'. The diagram is drawn in red on a yellow background.

NPTEL ONLINE CERTIFICATION COURSES

IIT KHARAGPUR

Let's do an exercise. We want to draw UML diagram for a given statement (in the above slide). Let say some persons keep animals as pets. How do we draw this?

We need to identify the classes here and the relations. 'Person' is a class. Let's underline this. A 'person' is a class and 'animal' is a class. But pet is not a class, 'pet' is the role of the animal which the person keeps.

Once we identify these two classes, the person and the animal, we need to identify the relation and relation is 'keeps' and 'pet' is the role of the animal. we can draw it in the UML diagram we will write 'person' as a class (please observe that the name of the class is always singular) and 'Animal' as another class and 'keeps' is the association relation.

But one thing that is not clear here, whether some persons keep one animal that means the animal belongs to multiple persons. Like two persons have one animal as pet, but that's not clear from the sentence but in the general case it appears that multiple persons can keep a single animal as pet and in that case, we will also have multiplicity here given as '*' in the person side.

Now, let us do one more exercise, slightly larger exercise as we already said that as we proceed in this course we should be able to handle moderately large problem statements and be able to draw various UML diagrams and so that either we write the Java code ourselves or it can be written automatically by a case tool.

(Refer Slide Time: 18:59)

Exercise 2: Draw UML Diagrams

A company has many employees and undertakes many projects. Each project is carried out by a team of employees.

```
classDiagram
    class Company
    class Employee
    class Project
    Company o-- "*" Employee
    Company o-- "*" Project
    Project -- "*" Employee : carried out
```

NPTEL ONLINE CERTIFICATION COURSES

Now let's do one more exercise. A company has many employees and undertakes many projects. Each project is carried out by a team of employees.

If we try to identify the classes here, 'company' is a class, 'employee' is a class, 'has' is the relation, which indicates an aggregation. Company has many employees, undertakes many projects. So, the company undertakes many projects and 'project' is a class. Each project is carried out by a team of employees or in other words a project is carried out by many employees. If we represent this in the form of a UML diagram, 'company' is a class, company has many employees and company undertakes many projects.

And each project is carried out by many employees or a team of employees. 'Carried out' is the association, the multiplicity on the project side is 1 and that is default association, we do not have to write the multiplicity if it is 1 or if we feel like we will just write 1 here, so that will be the class diagram for this problem (as shown in the above slide diagram).

Now let's look at some other elements of UML diagrams.

(Refer Slide Time: 22:01)

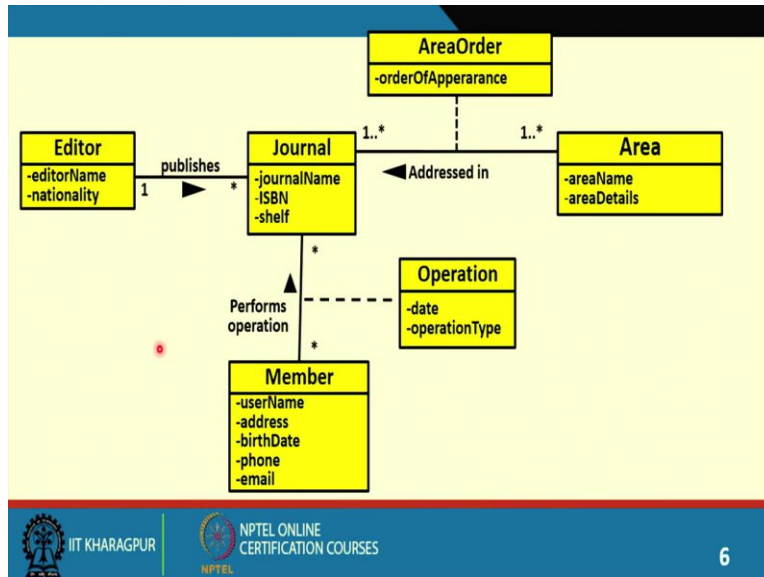
The slide features a yellow background with a blue header and footer. The header contains the word "Exercise" in a yellow box. The main content is a list of five bullet points. The footer includes the logos for IIT Kharagpur and NPTEL Online Certification Courses, along with the page number "5".

- Develop UML Class diagram for a software that we need to develop to manage a library.
- Each library member has a name, date of birth, address, phone number and email
- Each journal has a name, ISBN code, research areas, shelf in which located, and an editor. Each editor has a name and a nationality and publishes many journals.
- A journal may cover issues in several research areas. It is important to store the order of appearance of research areas, since these indicate their relative importance for the journal.
- It is important to keep track of the date on which a journal has been borrowed and returned, and the member carrying out the operation.

Let's do a larger exercise. This exercise is about a small problem, actually part of a problem and here we need to develop UML class diagram for a software. We need to manage a library. A library member has a name, date of birth, address, phone number and email. Here 'library member' is a class and these are the attributes: date of birth, address, phone number and email etc. A journal has a name, ISBN code and research areas and shelf in which located -- these are the attributes of the journal. Editor has a name and nationality and publishes many journals. You can see here that 'editor' is a class, the attributes are 'name' and 'nationality' and the relation between journal and the editors is 'publishes'. A journal may cover problems in several research areas, issues in the several research areas. It is important to store the order of appearance of the research areas, since these indicate their relative importance for the journal. Also, it is important to keep track the date on which a journal has been borrowed or returned by a member. The member carrying out the operation that means that a member can 'borrow or return'-- that is the association relationship between a journal and member. But then a member can borrow journal multiple times, and return multiple times on different dates and so on and we know that if we want to remember the date on which it has been borrowed and so on we use a association class.

But if we want to keep track or store the relative order for a research area of journal, we again need to use an association class. Please first try yourself and then compare with the below slide class diagram.

(Refer Slide Time: 25:12)



Now please compare with the above diagram that is given here which I have drawn here, that is a journal is associated with many areas, it addressed in a journal, more than one journal and the journal addresses more than one areas and we use association class to keep the order of appearance.

And similarly, there is an association between the member and the journal. The member performs operation and the operation has a date and operation type whether it is a borrow or return, this again we keep track in association class and the member can borrow even a journal multiple times, or a member can borrow several journals. And between the editor and the journal is an association. The editor publishes many journals. Please draw a diagram on your own for this problem and compare if you are able to draw nearly the same UML diagram. I think we are ready to move further and look at other issues of UML.

We will stop here and continue from this point in the next session.

Thank you.