

Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 37
Interaction Modelling

Welcome to this lecture. In the last lecture we looked at the Class diagram. We looked at the Class relations. There are 4 types of relations that can exist among classes and also we looked at how to identify the classes from a problem description and also to identify relations and to represent them in a, them in a diagram. But, we mentioned that even though it appears very simple to identify the classes just by reading a statement and also to identify the relationships, but lot of practice is needed.

(Refer Slide Time: 01:06)

The slide is titled "Identify Classes & Relations" in a yellow box. It contains a list of statements on the left and a hand-drawn class diagram on the right. The diagram shows two classes, "Student" and "Hostel", each in a rectangular box. A red arrow points from "Student" to "Hostel" with the text "lives in" written below the arrow. The slide also features logos for IIT Kharagpur and NPTEL Online Certification Courses at the bottom, and a small video inset of the professor in the bottom right corner.

- A square is a polygon
- Shyam is a student
- Every student has a name
- 100 paisa is one rupee
- Students live in hostels

Identify Classes & Relations

```
classDiagram
    Student --> Hostel : lives in
```

In the last lecture towards the end, we had done some practice on identifying class relations. Today, let us start with some more practice, let us I will display a simple statement. We need to identify the classes and also the relationship that exist between them and if are able to identify we can represent them in a class diagram ok. Let us get started. The first one is a square is a polygon.

So, what are the classes here and what relationships exist between these two classes. This is rather straight forward. The two classes are square and polygon and there exists a generalization, specialization relationship; it is a either relationship we can see the is a

phrase there which quickly identifies the type of relationship between square and polygon.

But the question is which would be the base class and which is the derived class? Is square derived from polygon or polygon derived from square. So, here the polygon is the base class and square is a special type of polygon. So, the inheritance relationship between polygon and square; polygon is the base class and square is the derived class.

The next one is Shyam is a student. Please identify the class and relations. If you look at this statement student is a class and Shyam is an object; it is not really a class just an instance of a student. So, Shyam is an instance of the student class. Every student has a name. So, what are the classes here; student is a class and name is an attribute of the student class.

100 paisa is one rupee; here, the 2 classes are rupee and paisa and there is a composition relationship. One rupee is a composite class and it consists of 100 paisa. But then, we need to be careful that this is not an aggregation relationship; it is actually a composition relation because a rupee consists of 100 paisa, we cannot make it 99 paisa or 101 paisa. So, then paisa in 1 rupee is fixed and therefore, this is a composition relation and we all know how to represent the composition relationship. Please draw the diagram even though I am not drawing here; please look up the syntax and please draw the diagram rupee is 100 paisa.

Students live in hostels. So, here the 2 classes are student and hostel. We need to use the singular form; the class name is not students, the class name is student and the other class is hostel and there is an association relation between student and hostel. But then, what about the multiplicity of the relation? A student lives in one hostel and a hostel can have many students. How do I represent that in a class diagram? Let me just draw this one. Student and hostel; there is an association relation and the student is the reading direction, student lives in hostel.

But then, we have to give the multiplicity and for that we need to identify one object of one of the class is associated with how many objects of the other class and vice versa. So, a student lives in one hostel. So, we can write here 1; but if you do not write 1, by default it is 1. So, to be safe we can write 1; but if you do not write that still because by default it is 1. But if you read it the other way, a hostel in a hostel many students live. So, this will

be star. So, the correct class relations is association relation between student and hostel and we should get this kind of relation.

(Refer Slide Time: 07:55)

The slide is titled "Identify Classes & Relations" in a yellow box. It contains a list of statements:

- 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 slide also features the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a small video inset of a speaker in the bottom right corner.

Now, let us look at the next one. Every student is a member of the library. So, here library aggregates all students is aggregation relationship because every student is a member of the library and we know that aggregation is a special type of association and if you write this is an association relationship still it is acceptable, but then, we have to give the multiplicity the association name reading direction etcetera correctly.

Next one, student can renew his borrowed books. Here there are 3 classes; student, book ok; there are 2 classes, student and book. Student can renew his borrowed books and we can draw an association relationships. So, this implies the student can renew. So, there is a association relationship between two classes student and book there are two classes here.

Now, let us look at the next one. The department has many students the 2 classes are department and student. But, we can debate whether it is an aggregation or a composition relation, to resolve this let us see that whether the department when it is created at the students fixed there.

Because the implication of the composition relation is that when the whole is created the parts are also created and when the whole is destroyed the part is also destroyed; the

parts cannot be added or deleted. But then, to the department students come and go; students may leave the department, graduate or otherwise they may leave the department, new students join.

So, the department and student is a aggregation relationship department is the composite class and student is the component and department has many students and that is aggregation relationship, please draw the aggregation relationship between department and student.

(Refer Slide Time: 11:19)

• A country has a capital city

• A dining philosopher uses a fork

• A file is an ordinary file or a directory file

Identify Classes & Relations

```
graph TD; File[File] --> OrdinaryFile[OrdinaryFile]; File --> DirectoryFile[DirectoryFile];
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us look at few more exercises. Some more exercises because as I said that we need some practice to identify the classes and the relations. The first one here a country has a capital. The country has, a country has a capital city; the two classes are country and capital city and what is the type of relation the giveaway here is the has a relation and we know that has a implies aggregation or composition and since the country has a fixed capital city. Normally, we can make it a composite relation, country has a capital city; it is a composite relation.

But then, we may consider that even the capital city can change and in that case if that is allowed then, we can make it a aggregation relationship; has a is typically and aggregation relationship even though it is just one instance. Aggregation is normally many components, but here there is only one, but because of the has a, we can represent it is an aggregation relation.

A dining philosopher uses a fork. So, the 2 classes are philosopher the dining philosopher and uses a fork. So, the dining philosopher is a derived class of the philosopher, if you think so otherwise, we can just write the philosopher is the class and use as a fork. So, there is a association relation between the 2 classes fork and philosopher and the association relationship is uses.

A file is an ordinary file or a directory file. So, the giveaway here is the is a relation. A file is the base class and ordinary file and directory file are the derived classes. So, there are 3 classes here; file, ordinary file and directory file. So, we can draw here file is the base class and ordinary file and the derive file sorry directory file, these are the 2 derive classes. Now, let us look at the next one. A file contains many records; contain is the giveaway here.

(Refer Slide Time: 14:41)

The slide features a yellow background with a list of examples for identifying classes and relations. A yellow box on the right contains the title 'Identify Classes & Relations'. At the bottom, there are logos for IIT Kharagpur and NPTEL Online Certification Courses, along with a small video inset of a speaker.

- 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 person uses a computer language on a project

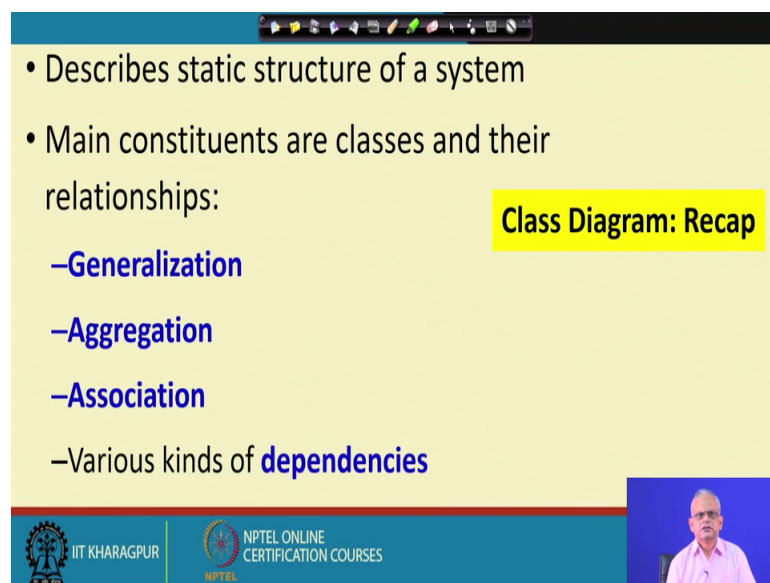
So, there is a aggregation relationship between file and record; file contains many records. Now, let us look at the next one. A class can have several attributes. Attribute is normally not a class because you do not have methods or operations and private data, public data etcetera associated with attribute just one value maybe. So, this is a several attribute, it is not really a class attribute.

So, there is class and attribute is part of it. Relation can be association are generalization. So, this is again a generalization, specialization relationship or inheritance relationship. Relation is a base class; association and generalization are the derived classes.

Now let us look at this. A polygon is composed of an ordered set of points. Here, the giveaway is the composed of and this phrase indicates that a polygon permanently contains set of points; you cannot just delete some point here, add some point here. When a polygon is created, a number of points in that is fixed and therefore, a polygon is a composite class of many points.

Now, let us look at the last one. A person uses as a computer language on a project. There are 3 classes here; person, computer language and project. But then, what about the relation between this? Person use as a computer language is an association and again language is used in a project. So, that is again and association, the 3 classes a person, computer language and project and where association relationship between the person and computer language and computer language and project.

(Refer Slide Time: 17:34)



The slide is titled "Class Diagram: Recap" and lists the following points:

- Describes static structure of a system
- Main constituents are classes and their relationships:
 - Generalization
 - Aggregation
 - Association
 - Various kinds of dependencies

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a speaker in the bottom right corner.

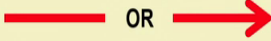




Now, let us just recaptured what we discussed in the class diagrams. The class diagrams describe the static structure of the system. Unlike the object diagrams where which is dynamic because as you said that the objects may created, deleted, link establish destroyed and so on.

But the class is the static structure of a system; association, relation, aggregation etcetera; they are permanent between classes. And therefore, we say that it is a static structure of a system.

The main constituents of the class relation, the different types of class relations are generalization, aggregation, association, dependency, composition.

(Refer Slide Time: 18:29)

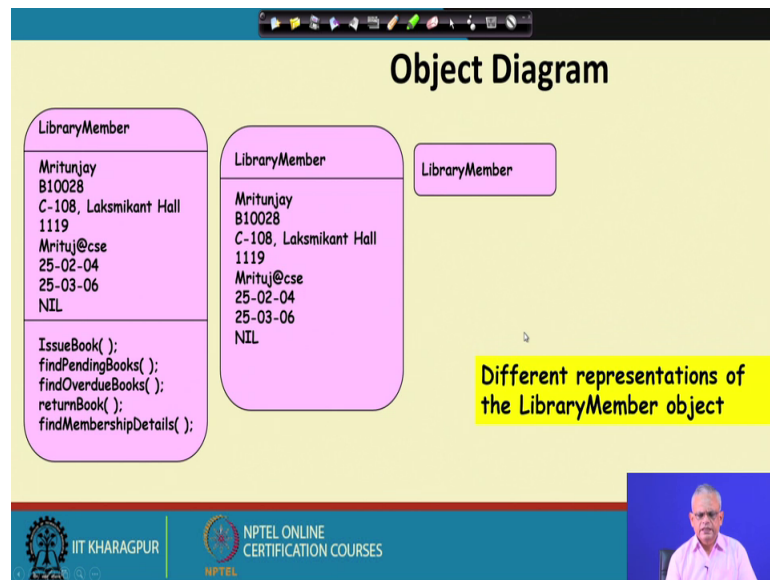
Summary of Relationships Between Classes

- **Association**
 - Permanent, structural, "has a"
 - Solid line (arrowhead optional)
- **Aggregation**
 - Permanent, structural, a whole created from parts
 - Solid line with diamond from whole
- **Dependency**
 - Temporary, "uses a"
 - Dotted line with arrowhead
- **Generalization**
 - Inheritance, "is a"
 - Solid line with open (triangular) arrowhead
- **Implementation**
 - Dotted line with open (triangular) arrowhead

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a speaker in the bottom right corner.

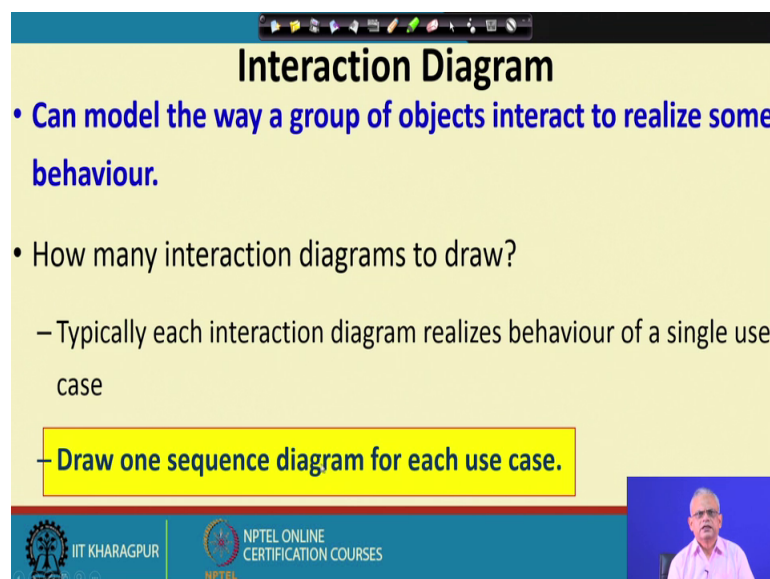
And we had seen the notations for that and Association typically appears as a has a clause in statement and then, it can be represented by here a simple line or an arrow; Aggregation relationship a diamond with an arrow. A dotted line with this kind of arrowhead for a Dependency; Generalization is an arrow with this kind of depends is a line having an arrow with this kind of symbol and then, Implements again here it similar to a generalization inheritance relation; but here it is a dotted arrow.

(Refer Slide Time: 19:25)



The object diagram just like a class diagram, they can either appear with the name of the object or some attributes of the object or the methods that are available attributes. And here, it is a rectangle with rounded edges. So, far we have looked at the use case diagram, the class diagram and object diagram is simple within spend too much time. Now, let us look at some behavioral view of a system and will look at the interaction diagram.

(Refer Slide Time: 20:16)

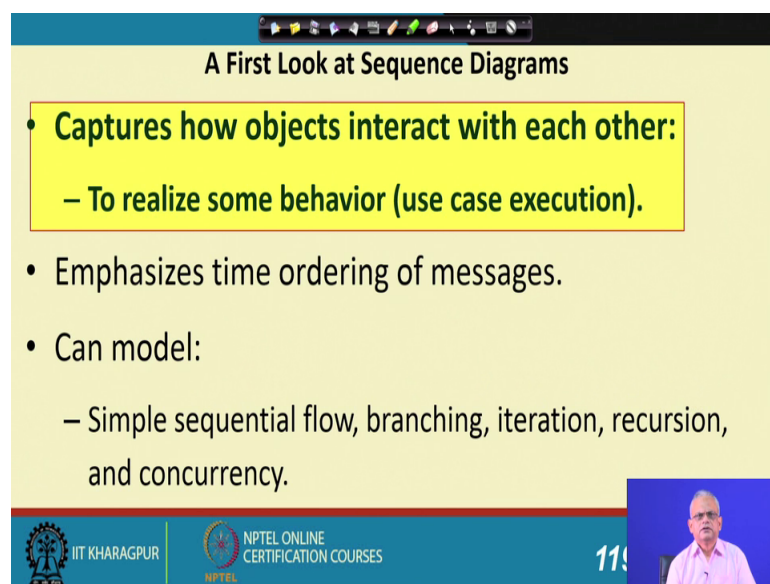


The interaction diagrams, they can model the way that objects interact during the run of the system to realize some behavior; but then, what can be a behavior of a system? The behavior is when we execute it done something for us for example, issue book return book. So, that is a behavior of the system and if we think of it the behavior is either execution of a use case or part of a use case.

So, we model here using interaction diagram that when I use case executes what are the objects that take part and what messages the interchange among each other. But the question here is that when a model a system, how many diagrams to draw; how many interaction diagrams? The answer is simple we draw as many use cases are there because each interaction diagram captures the behavior of a single use case and therefore, the number of interaction diagram is typically equal to the number of use cases. But of course, if the use case is very complex, we might split into multiple interaction diagrams; but if you have designed well, then we might have decomposed the use cases so that none of the use cases is very complex.

So, therefore, the number of interaction diagram should be equal to the number of use cases identified and we must remember so very important statement here that we need to develop one interaction diagram for every use case. We will see that interaction diagram either a sequence diagram or collaboration diagram and we need to draw one sequence diagram for you each use case.


(Refer Slide Time: 22:46)



A First Look at Sequence Diagrams

- **Captures how objects interact with each other:**
 - To realize some behavior (use case execution).
- Emphasizes time ordering of messages.
- Can model:
 - Simple sequential flow, branching, iteration, recursion, and concurrency.

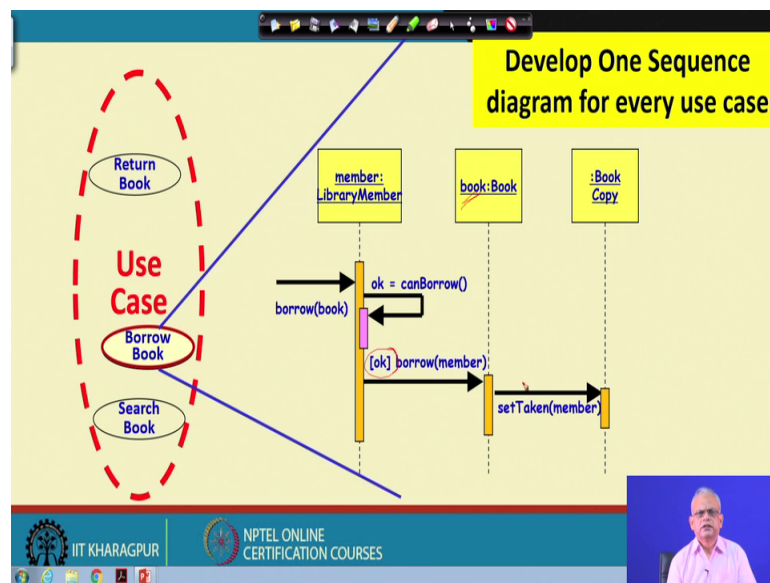
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 119



The interaction diagrams capture how the objects interact during the run of the software to realize some behavior; for example, use case execution and here we need to identify what are the objects that are interacting during the execution of the use case and what are the messages there exchanging among each other and the time ordering of the messages.

And we will see that we will have ways to represent some control information like whether there is sequential flow from one object to another object, the control just flows from one object to another object. There is a branching that is depending on some condition there; control flow occurs from one object to another object or some other object. Iteration multiple times control passes from one object to other object and returns, recursion, concurrency and so on.

(Refer Slide Time: 24:12)



But we must remember that for every use case, we need to develop one sequence diagram. Sequence diagram is one type of the interaction diagram. There are actually 2 types of diagrams; the sequence diagram and the collaboration diagram. If this is our use case model; then, we need to develop one sequence diagram for each of these use cases.

You can take any one use case and draw the sequence diagram for this, the sequence diagram will look something like this that what are the objects that participate during execution of the use case and what kind of messages they exchange and the time ordering of those messages. So, here this diagram at the top, we mention the objects that

interact. Even though, it looks like a class symbol, but then just look here we have underlined here library member is a class name and member is the object when we underline that.

This one is an instance of the book class, small book is an instance of the book class and then, here just here in this we have not even mentioned what is the name of the object and that is because it is an anonymous object; any book copy here. And the diagram is typically read from top to the bottom. This we call as the timeline and this arrow is here capture the sequence in which the messages are exchanged between various objects of these classes as the interaction between them occurs during the execution of the specific use case.

So, to start with, the user issues a borrow book request and he needs a specific type that is book as given as the argument here and then, this is a self method here and checks whether the member is eligible to borrow book; has not exhausted his limits, number of books that he can issue etcetera. And just see here there is a recursion here because the same one calling here and then, this small rectangle here is an indication that the object is active now it is executing its code.

And then, it depending on whether he can borrow the book that we write here in the form of this control statement that ok with the condition is true then invoke the borrow member. So, the book class the specific book, I mentioned here that this is a book here and then, the book for the book class the method borrow is activated and the member class waits here for the result to return and that is how we are not really stop this rectangle here it continues.

And once it comes to the book, the control comes to the book it gets active and then if the book has many copies and the specific copy that would be issued any one of the copies the member who is the issuing it, it is the registered there.

So, that next time query we query that who has taken the book copy, then you can find that the member has taken the specific member who was requesting has taken the book. So, this is the essence of a sequence diagram. Basically I will identify what are the objects and for execution of a use case in what way they interact; which messages the exchange and in what time model. We are at the end of this lecture stop here and continue in the next lecture.

Thank you.