

Object-Oriented Analysis and Design
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology – Kharagpur

Lecture 38
Class Diagrams: Part I (Class, Property and Operation)

(Refer Slide Time: 00:26)

The slide features a blue header with the IIT Kharagpur logo and the text 'Module 26'. Below this, a white box contains the title 'Module 26: Object Oriented Analysis & Design' and the subtitle 'Class Diagrams: Part 1 (Class, Property & Operation)'. The instructor's name 'Partha Pratim Das' is listed, followed by his department and institute. Contact information includes an email address and three phone numbers. A disclaimer at the bottom states that the presentation uses diagrams and examples from 'Object-Oriented Analysis and Design - With Applications' by Grady Booch et al. (3rd Ed, 2007) with permission. The footer includes 'NPTEL MOOCs Object Oriented Analysis and Design', the instructor's name, and the slide number '1'.

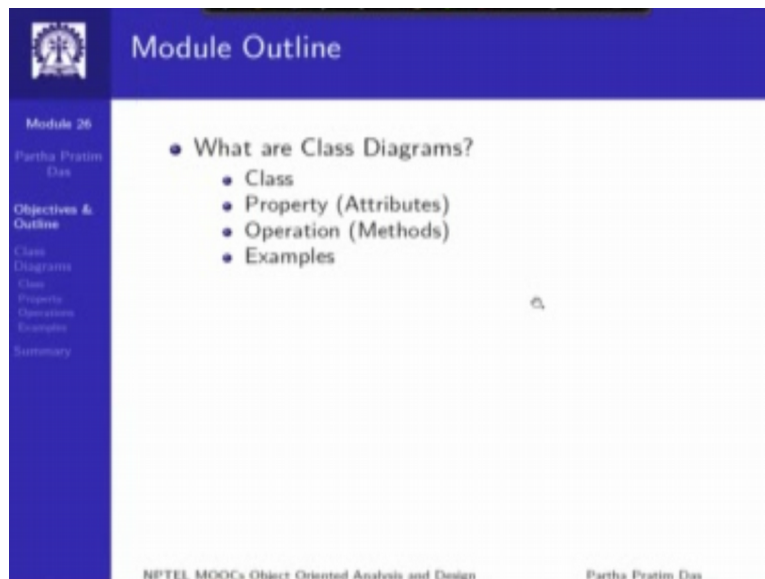
Welcome to module 26 of object oriented analysis and design. We have been discussing about UML diagrams, introducing variety of diagrams. We have already discussed about the use-case diagram at length. Next diagram we take up is the class diagram which is possibly one of the more important diagrams in the presentation of a system.

(Refer Slide Time: 00:56)

The slide has a blue header with the IIT Kharagpur logo and the text 'Module Objectives'. A white box contains the objective 'Understanding Class Diagrams'. The left sidebar is identical to the previous slide. The footer includes 'NPTEL MOOCs Object Oriented Analysis and Design', the instructor's name, and the slide number '1'.

We will, in this module start the discussion on class diagrams and this will go over to the next two modules as well.

(Refer Slide Time: 01:07)



The slide is titled "Module Outline" and is part of "Module 26" by Partha Pratim Das. It lists the topics to be covered: "What are Class Diagrams?", "Class", "Property (Attributes)", "Operation (Methods)", and "Examples". A sidebar on the left contains a navigation menu with options: "Objectives & Outline", "Class Diagrams", "Class", "Property", "Operations", "Examples", and "Summary". The footer includes "NPTEL MOOCs Object Oriented Analysis and Design" and "Partha Pratim Das".

Module 26
Partha Pratim Das

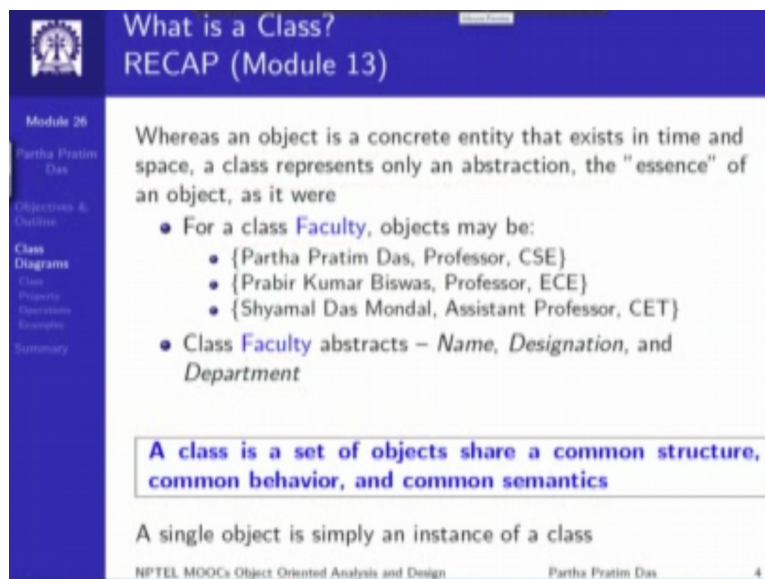
Objectives & Outline
Class Diagrams
Class
Property
Operations
Examples
Summary

- What are Class Diagrams?
 - Class
 - Property (Attributes)
 - Operation (Methods)
 - Examples

NPTEL MOOCs Object Oriented Analysis and Design Partha Pratim Das

In this module, we just specify what is a class diagram and discuss the basic representation of class, property and operations. We will take up examples as well.

(Refer Slide Time: 01:26)



The slide is titled "What is a Class? RECAP (Module 13)". It explains that while an object is a concrete entity existing in time and space, a class is an abstraction representing the "essence" of an object. It uses the example of a "Faculty" class, where objects could be "Partha Pratim Das, Professor, CSE", "Prabir Kumar Biswas, Professor, ECE", or "Shyamal Das Mondal, Assistant Professor, CET". It states that the "Faculty" class abstracts the "Name", "Designation", and "Department". A highlighted box defines a class as a set of objects sharing common structure, behavior, and semantics. It also notes that a single object is an instance of a class. The sidebar and footer are identical to the previous slide.

Module 26
Partha Pratim Das

Objectives & Outline
Class Diagrams
Class
Property
Operations
Examples
Summary

What is a Class?
RECAP (Module 13)

Whereas an object is a concrete entity that exists in time and space, a class represents only an abstraction, the "essence" of an object, as it were

- For a class *Faculty*, objects may be:
 - {Partha Pratim Das, Professor, CSE}
 - {Prabir Kumar Biswas, Professor, ECE}
 - {Shyamal Das Mondal, Assistant Professor, CET}
- Class *Faculty* abstracts – *Name*, *Designation*, and *Department*

A class is a set of objects share a common structure, common behavior, and common semantics

A single object is simply an instance of a class

NPTEL MOOCs Object Oriented Analysis and Design Partha Pratim Das 4

So before we get started let us quickly recap on what is class, so we have discussed the concept of class and objects at length, so a class represents the generic footprint or the generic layout of objects that exist. The concrete entity that exists are objects and we have seen that for the class like faculty, this could be the objecting stances and class as an abstract will have different properties or attributes that define this task.

(Refer Slide Time: 02:14)

The Canonical Form of a Complex System: RECAP (Module 04)

Module 26
Partha Pratim Das
Objectives & Outline
Class Diagrams
Class
Properties
Operations
Example
Summary

The Canonical Form of a Complex System

Source: Object-Oriented Analysis and Design – With Applications by Grady Booch et. al. (3rd Ed. 2007)

NPTEL MOOCs Object Oriented Analysis and Design Partha Pratim Das 5

We have discussed earlier also about canonical form of complex system, where we observed that a system could be looked at in two distinct views, one is the IS A hierarchy or generalization specialization view which is primarily the representation of the class structure. We have also seen that it can be looked at as a part of hierarchy which is the HAS A relationship which represent the object structure.

Class diagram is unique in the UML representation because it provides us the mechanism to define the part of hierarchy or the object structure and the IS A hierarchy of the class structure together in the same diagram. So we will take a look in terms of how this is done.

(Refer Slide Time: 03:27)

Class Diagrams in SDLC phases: RECAP (Module 22)

Module 26
Partha Pratim Das
Objectives & Outline
Class Diagrams
Class
Properties
Operations
Example
Summary

- In the **Requirements Phase**, the class diagram is used to identify the major abstractions
- At this stage the attributes and operation of each abstraction m known
- Classes are identified as **domain models**

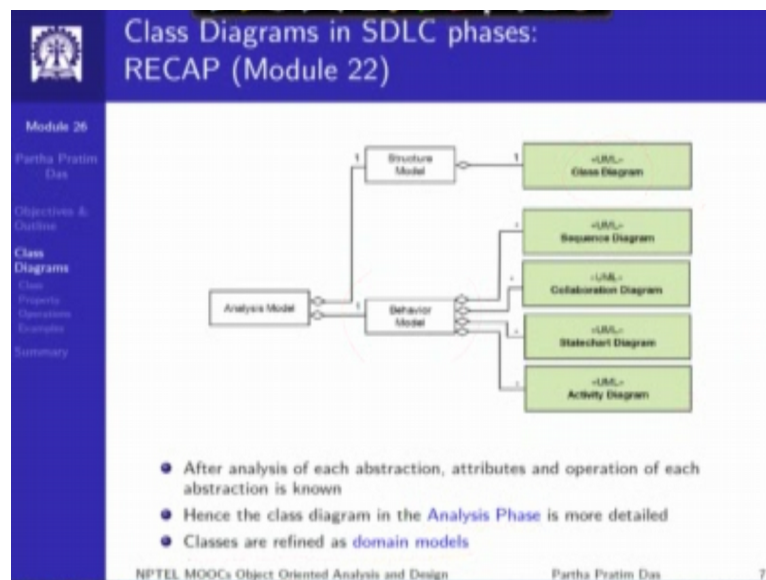
NPTEL MOOCs Object Oriented Analysis and Design Partha Pratim Das

I would also like to remind you that we have seen that at different phases of the software development lifecycle or SDLC, different UML diagrams are involved and right from the

beginning that is in the requirements phase, the class diagram along with the use-case diagram come in and start playing an important role, when you talk about class diagram at the requirement phase it is often that the attributes and operations, the properties and operations need not have been very clearly defined.

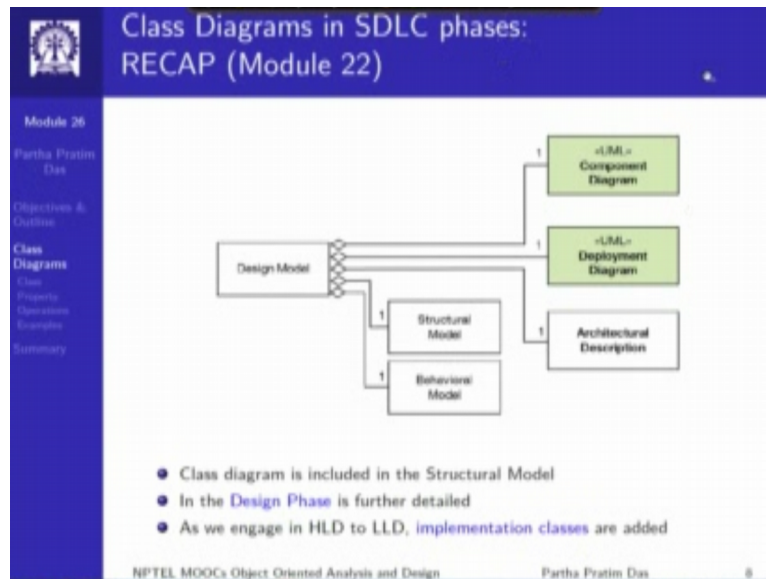
May not have been totally identified. It may be more that we have the abstraction, the concept of a class that we have been able to identify and we are still looking for, not maybe we have some of the properties and some of the operations but we may not have been able to complete that whole thing. So here the classes are being identified as domain models because we are trying to capture what the domain has. This is where the class diagram is supplementing the use-case diagram.

(Refer Slide Time: 04:53)



In the next SDLC phase which is the analysis phase, we analyze the class diagrams that we have captured earlier, tried to refine them further, continuing to represent the domain models, represent the information that the problem domain has and build up the UML representation of the problem along with several other behavioral models which we will discuss in subsequent stages.

(Refer Slide Time: 05:32)




Further in the SDLC, the next phase talks about the design, called the design phase, where we have seen that the design is further detailed in terms of high level design and low level design and possibly at this stage we are making use of the class diagram. The class diagram belongs here, the structural model and these are the new additional diagrams that are coming in.

Now in the design phase the class diagram is a further refinement of the domain models and in addition you may have implementation classes. That is now, we are in the design process, so we are in the output of this would be actual classes that we would start coding possibly in C++ or in Java and so on. So it is likely besides the domain models we will also have specific classes added to the design which will talk about the implementation aspect of the whole design.

So the class diagram as we see, play a role in all these 3 phases very critical. In other phases also it will be required. But usually the development of the class diagram will remain limited to these few phases alone.

(Refer Slide Time: 06:57)




Class Diagram

Module 26
Partha Pratim Das
Object-Oriented Analysis & Design
Class Diagrams
Class
Properties
Operations
Examples
Summary

- Class diagram is UML structure diagram which shows structure of the designed system at the level of classes and interfaces, shows their features, constraints and relationships – associations, generalizations, dependencies, etc.
- Some common types of class diagrams are:
 - Domain model diagram
 - Diagram of implementation classes

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



NPTEL MOOCs Object Oriented Analysis and Design Partha Pratim Das 9

So given that a class diagram is an UML structural diagram. You will recall that we have talked of structural diagram and behavioral diagrams. Class diagram talks about the basic structure of the design system at the level of classes and interfaces. Interfaces we will talk about more. Interfaces are like classes where we may not know the details of the properties that are inside, but we know just the operations that we want to satisfy.

Interfaces could be generalizations of various different class concepts and so on. Besides that, class diagrams will show the features, the constraints, the relationship, this is most important and we have talked about variety of relationships in terms of association and generalization, dependencies and so on and we will see how all of these in terms of the class diagram can be presented. Typically, class diagrams are of different kinds.

Most of them are either domain model diagrams where basically I am capturing the domain information from the problem which will significantly happen in the requirements as well as analysis phase and the other type they are the diagrams of implementation classes which will be added further as we moved further from the analysis phase to the design phase. So implementation class diagrams which represent implementation classes, naturally have far more details than the domain model class diagrams.

(Refer Slide Time: 08:45)

Features of a class

- **Non Static Features:** characterizes individual instances of class
- **Static Features:** represents some characteristic of the class itself
- **Structural Features (attributes):** is a typed feature of a class that specifies the structure of instances of the class
- **Behavioral Features (Methods):** is a feature of a class that specifies an aspect of the behavior of its instances


Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)

Now before we can start the representation of classes, let us remind ourselves that the major features of the class include that a class has non static feature that is where the individual instance or every object instance of a class will have these features. So these are called non static features. We have static features where there is some characteristics, some feature may be property of operation which is a characteristic of the class itself, not of the specific objects.

For example, if we talk of that, we have a property to define count the number of instances of a class that we have constructed. This count will be a static feature because every object will not have a separate count because it is just one of those object but at a class level we will have a static count property which will keep the count of the objects that have been construct.

Accordingly, we could have operations which are also static, for example the operation which will actually update this count when an object is constructed, will update it when an object is distracted or when you want to know how many objects of a class exist and so on. In other terms there are structural features or attributes and they are behavioral features or methods or operations that will be involved in defining a class.

(Refer Slide Time: 10:30)



Notation for Class

Module 26

Partha Pratim Das

Objectives & Outline

Class Diagrams


Class

Property

Discussion

Exercises

Summary



- Class name should be centered and in bold face inside a solid-outline rectangle, with the first letter of class name capitalized

Student

Class Student - details suppressed
- Abstract Classes (which cannot be instantiated) have the keyword abstract mentioned within { }

Teacher (Abstract)

Abstract Class Teacher - details suppressed
- A class has optional compartments separated by horizontal lines containing attributes and methods in order


NPTEL MOOCs Object Oriented Analysis and Design Partha Pratim Das 11

Now coming to the notation. A class will be typically depicted in terms of a closed rectangle which has a solid boundary and in between that, within that boundary, the name of the class, class must have a name and the name of the class will be written in bold letter with the first letter in the upper case that is a basic convention, so we will not write a class like this where the first letter is not upper case. This is the correct way of doing that.

Some classes could be abstract. If a class is abstract, if a class is abstract then it can be, we will write within curly braces after the name of the class that it will be abstract. What it means is this class only has a concept but in the realization we will never actually instantiate the class and create objects. Abstract classes in an alternate form could be represented by writing the class name in italicized font as well.

Finally, a class may have optional compartments, that is I have a class, say this is the class, the name of the class is a book, I may have further compartments. One compartment for the properties and the other compartment for the operations. But these are optional I may have both of them, I may have only one of them. I may not have any further compartment at all.

(Refer Slide Time: 12:12)



Notation for Operations (Methods)

Module 26

Partha Pratim Das

Objectives & Outline

Class Diagrams

Class

Property

Operations

Exercise

Summary

● **Operation (Methods) specification format:**

Visibility OperationName (*ParameterName : Type*) : *ReturnType* { *Property string* }

- The visibility of the operations are denoted by +(public), #(protected) and -(private)
- OperationName is underlined if it is Static, and is italic if it is Abstract
- Return type is optional
- An operation may be *Read Only*, *Static*, *Ordered*, *Unique*, *Abstract*, *Sequential*, *Guarded* or *Concurrent*

Student
+name: String +date.of.birth: Date +roll.no: String unique +age: Integer +subject: Subject[] *
@recordAttendance(): bool +getCertificates(): Certificates[] {unique, ordered} -changeSubject(Subject s): bool +calculateAge(): Integer +bookMusicClassSlots(): bool {concurrent}

NOTES: MPCS's Object Oriented Analysis and Design
Partha Pratim Das
15

Now coming to the description of properties, a property is written, so typically you can see that, let us say this is one class, this is the class name, this is the first optional compartment where we issued the properties and let us say we are talking about this particular property. So what it has, certainly this is role number is a property which has a property name. So after the property name, we put a colon to separate it from the type of the property.

So string is the type of the property. So roll number colon string says that roll number is of type string, while it is assumed that string is something which is known already. If it is not then that will also have to represent it. So those of you who may be familiar with C++ and Java will relate that we keep the similar information in a C++ or a Java class but it is just that we write them in the different order.

We first write the type name and then we write the variable name. Here we first write the property name and then we put the type of which it belongs. So that is this part. Then it is prefixed with a symbol which could be one of plus hash or minus, they mean the visibility of that particular property of that particular data member. If a property is public, it is meant that any other class can access that property directly.

If it is private it means that only the operations of that specific class can access that property but no external class, no other class can access that property. And finally if the visibility is protected then we have something in between this. If I have one class, say employee and if we have another class say manager, so that they are related by on a IS A hierarchy that

manager is a employee which means that every manager satisfy all the properties of an employee but may have some additional properties, some additional operations.

If such specialization exists then for a protected member in the base class, the derived class or the class that specializes the base class can access this protected members. But other classes cannot access the protected members. So protected visibility tells us a different kind of access restriction where for a child class or for a specialization protected members are like public, they can be accessed modified.

Whereas for any other class which is not, a specialization of this given class, the protected members are like private. So as we define the property we also define the visibility of that property. Then we may have an optional qualifier that follow that type name which say certain specific things about that property. For example, here we are saying that your roll number is, the style is to put it within curly brace and these are the different qualifiers that I could have.

So if I say unique it means that of all the instances of this class, this particular property will always take distinct values. In other words, I cannot have two students, we are representing the student class, I cannot have two student objects which have the same value on the roll number property. So several such are possible, I can make something read only. If I make something read only, then that property can only be read, it cannot be changed, it is whatever is set at the construction time of the object will remain as such.

A property maybe optional. If a property is optional then I can have null value for it, that is it is not necessary to have a specific value at the time of instantiating objects. A property could be static so that it operates only at the class level. A property could be ordered that I can say that within different values will have to occur in a certain order and so on. So the other way to say if a property is static is also to be able to underline that.

So if we underline that, that also means that the property is static. Now the property could have multiplicity. This is important that it could happen, I could have number of them. For example, here I have a subject, visibility is public, is of type subject which means that there is some other class subject, so we are saying that this subject property is an instance of this subject plus, but what we additionally have here is one dot dot star.

This defines the multiplicity which states that I can have one subject, I can have two subjects, I can have arbitrary number of subjects, of course more than one. So we can specify a fixed number here, we can specify a range like this, we can specify a range which is bounded and so and so forth. So that is the basic sense of multiplicity.

Certainly the multiplicity by default is one, for example here nothing is written, so the multiplicity is one. So there is only one data part property here. So if we do not say anything it is one and of course that is a minimum value that you can have. Then some properties like this one is written with a forward slash before the name.

So you are saying, this forward slash means that this is a derived property, which means that this property by itself will not be stored but there will be some operation through which the value of this property can be computed whenever I want to access it. If you try to see the reason, we are saying this property is age. For several applications, I will need to know what is the age of a person, age of a student.

Certainly we cannot store the age because certainly with every passing day the age is potentially changing. So we cannot update that data. Instead what we do? We store the date of birth and we will possibly have some method, some operation like say calculate age which will be associated with this derive property age. So that whenever we want to access the age property, this operation will be invoked.

This operation will consult the date of birth, consult the date that is today and based on that compute the age and give us the value. So in terms of the view of the object we will see, as if there is a age, but there is nothing be actually stored. So this is the concept of derived property and we will see the use of several derived properties in the design. Moving on, the next that we need to specify is certainly operations or methods.

So they written in this form, say every method has a name and the fact that it is a method is shown by having a pair of parenthesis after that. It is separated by colon from the return type. So this says, this will, get certificate is a method in student and it will return certificate. Record attendance is a method, is an operation and it will return a bool. So again you can see

that compared to the common object oriented programming languages like C++ or java, the style here of expression is little different.

We first write the method name and then we specify the return type and the return type itself is optional. So I may have a method which does not have a return type, which basically means that it is not expected to return any value. So if you think in terms of C++ then it's return type would be coded as void. Methods have similarly different visibilities and the visibility notation is same as of the attribute.


So they could be public, protected or private. Methods can be qualified by different qualifiers, like we are saying get certificate as a method is qualified by unique and ordered which means that this method will produce results which are a number of certificates. You can see the multiplicities star which means any number of certificates, zero or more. But what it says is that it is unique.

So if it returns a list of three certificates then these 3 certificates will have to be unique. They are ordered. So there is some ordering between them. For example, the ordering could be the level of education for which the certificate is provided, the 10th standard certificate, the 12th standard certificate, the graduate, the under graduation certificate and so on. And these are the various different types of qualifiers are possible, ordered, unique, abstract, sequential, guarded, concurrent.

These sequential, concurrent, these are qualifiers to take care of concurrency in the system, so that there are situations where a particular operation maybe invoked by more than one object at a time. So if you say that a method is concurrent then it is possible to invoke it by multiple objects at the same time. But if you say it is sequential, then it necessarily means that you will have to do one and then you can do the next.

It could be guarded, operation could be guarded where in it means that the operation has a precondition, that is I can invoke this operation from an object provided certain guard condition is satisfied. So these are the typical ways to denote the operations or methods of a class.

(Refer Slide Time: 24:26)



Abstract Classes of LMS

Module 26

Partha Pratim Das

Objectives & Outline

Class Diagrams

Case

Properties

Operations

Exercises

Summary

● We represent below the two abstract classes of LMS

Employee (Abstract)

```

+name: String
+id: String
+gender: (Male, Female)
+onDuty: Bool
+salary: Double
+doj: Date
+reportsTo: String
+recordAttendance(): Bool
+requestLeave(): Void
+cancelLeave(): Void
+availLeave(): Void
+reportLeave(): Leave


```


Leave (Abstract)

```

+startDate: Date
+endDate: Date
+status: (New, Approved)
+isValid: Bool
+Type: ()
+approveLeave(): Bool
+id: String
+type(): String
+approveLeave(employee e): Bool
+isValid(): Bool

```





NPTEL MOOCs Object Oriented Analysis and Design

Partha Pratim Das

1

So if we just refer to our LMS example. So LMS as we have seen have two major abstract classes employee and leave. There is specialization which we have seen earlier. We are not talking about that. We are just talking about the abstract classes, so there could be several properties of employee name, employee ID, gender, salary, the date of joining and so on and there would also be several operations like recording attendance, cancelling leave, requesting leave and do on.

And we have seen how this work, how this can be extracted and here we are just making a representation for that. I would draw your attendance for this. For example, for leave we will have something called, Is Valid. So if Is Valid is true then that leave can be processed. If it is false, then that leave cannot be processed. Naturally we do not expect that while the leave is created, somebody is entering a value for its value.

But this will relate to some operations say validate leave which can have the result of a Boolean which will be set when I try to access this Is Valid property. Therefore, you can see this slash here which say that Is Valid is a derived property of the class.

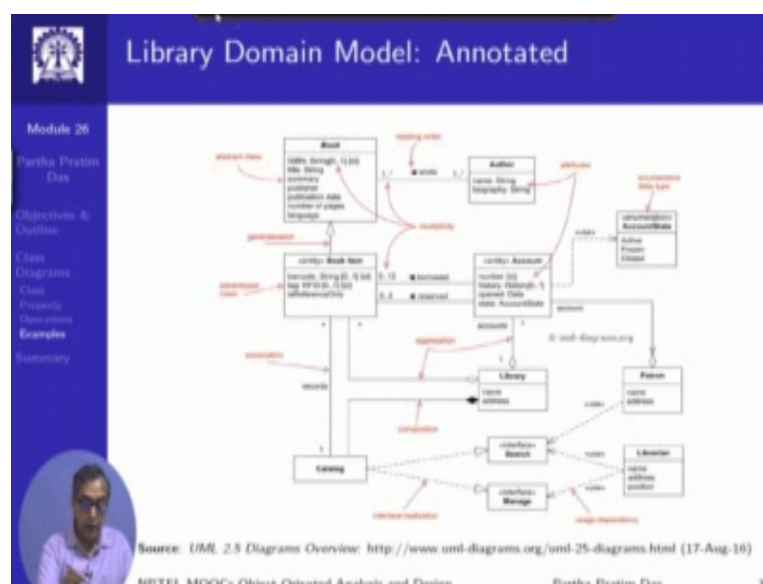
(Refer Slide Time: 26:06)

You can see that this is in a sense that little bit simplified model because we are talking about the domain, for example it is not showing the visibility specification for this different attributes because that will get decided based on getting all the different classes together and when we go towards actual implementation then we can decide as to what are the properties that we want to hide, what are the properties that should be publically available and so on.

But here since we are just capturing from the domain we have not yet done that exercise, so you do not see the visibility specification symbols. So similarly there are several other classes like, class defining an author which is a name biography, birth date of the author. The library itself is a class which has a name and an address whether it is the national library situated at Calcutta which is a central library of IIT, Kharagpur and so on.

So if there could be library instances coming in here, there could be librarian is another class which has different attributes and so on. So in this way you can see a whole lot of classes that are captured from the library domain to represent the, to build up, start building up the class diagram. You also of course see lot of other symbols and some of which we have casually talked of while we introduced different concepts, but we will come to discussing those, once we have actually talked of the relationships.

(Refer Slide Time: 30:04)

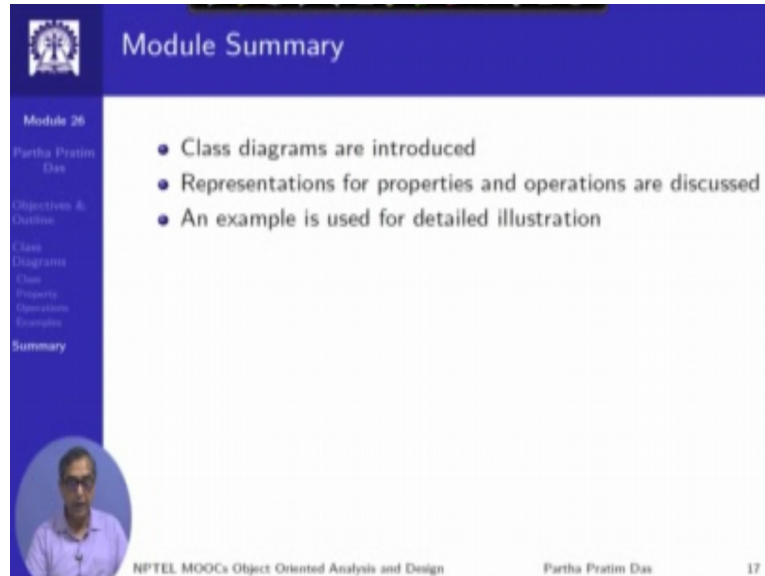


Finally, it is the same library domain model. The only thing that is provided here, that in addition to the actual model you have lot of annotations given, so these annotations, this will basically help you prepare for your understanding, so that it specifies that this is an abstract

class and you can see, how do you know this is an abstract class because you know that it is written in a italicized form.

So these are different attributes as marked here and rest of the stuff you have not done yet. So you can study this and understand and get familiar with the way the classes can be represented in the class diagram.

(Refer Slide Time: 30:56)



The slide is titled "Module Summary" and is part of "Module 26". The presenter is Partha Pratim Das. The slide lists the following topics:

- Class diagrams are introduced
- Representations for properties and operations are discussed
- An example is used for detailed illustration

The slide also includes a navigation menu on the left with the following items: Module 26, Partha Pratim Das, Objectives & Outline, Class Diagrams, Class, Property, Operations, Examples, and Summary. A small circular portrait of the presenter is located at the bottom left. The footer contains the text "NPTEL MOOCs Object Oriented Analysis and Design", "Partha Pratim Das", and the slide number "17".

To conclude we have introduced the class diagram. We have shown the representation of properties and operations in a class and we have shown a couple of examples. In the next module, actually in the next two modules, we will talk about different relationship amongst classes that we had studied earlier, how to represent them in the class diagram.