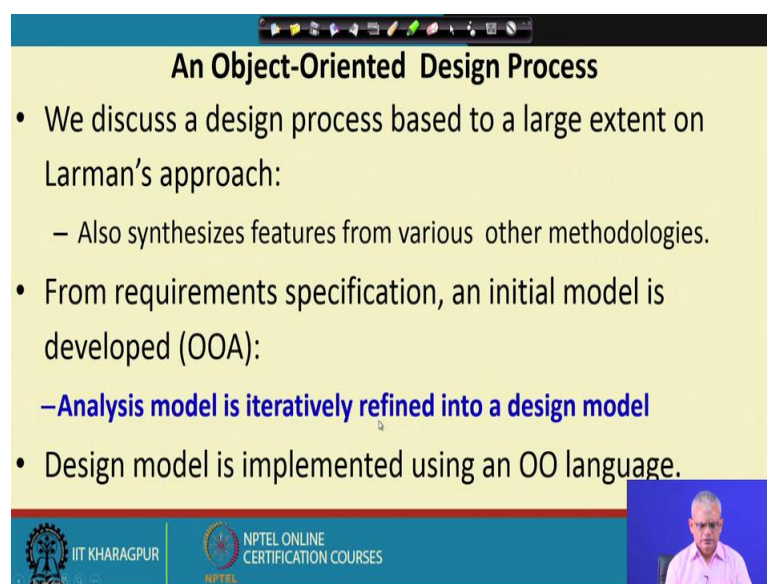**Software Engineering**
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 40
An Object - Oriented Design Process

Welcome to this lecture, in the last few lectures we looked at the UML, the UML is a modeling language and different types of views of a problem can be created. But we looked at only 3 or 4 diagrams. We looked at the use case diagram, class diagram, object diagram, sequence diagram, collaboration diagram and the state machine diagram. There are of course, other diagrams for we did not discuss those and those diagrams are not really needed in very simple problems. For specific category of problems, we need activity diagrams and so on. Now we look at a design process; the design process will tell us, will give us information about how to go about, what are the steps through which will carry out of design and we will use the UML to document the results of our process.

So, let's look at this design process and if you understand this well, for any given problem we can use this process or the number of steps that are there here and come up with a reasonably good design and the UML is of course used for documentation and helping carry out the steps.

(Refer Slide Time: 02:21)

Let's look at the design process. We look at it to a large extent based on the Larman's approach, but if we look at any other design process, there are only minor variations. In our approach we also use features from other methodologies. Here we have the requirement specification available to us. Based on the requirement specification document, we develop an initial model which we called as the analysis model. And then we refine this to obtain the design model and the design model as we proceed in the design step. Good amount of code is also automatically generated and then during the implementation, we need to fill up some of the machine code.
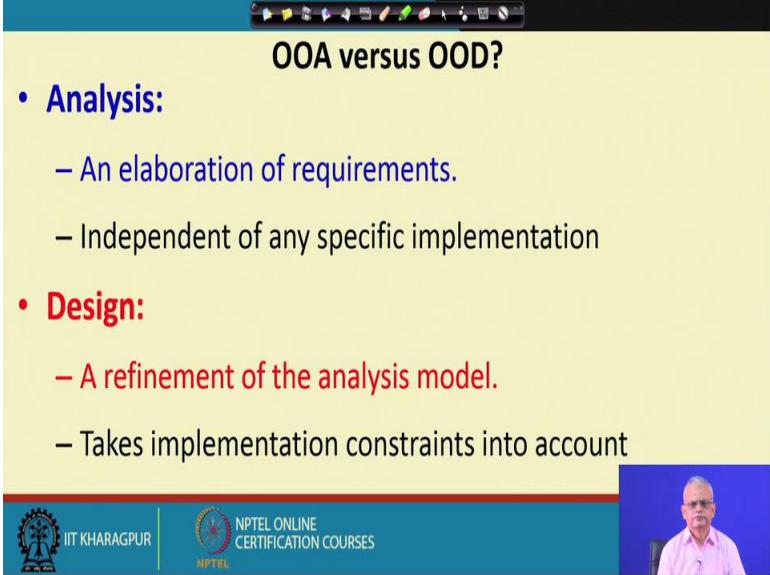
(Refer Slide Time: 03:35)
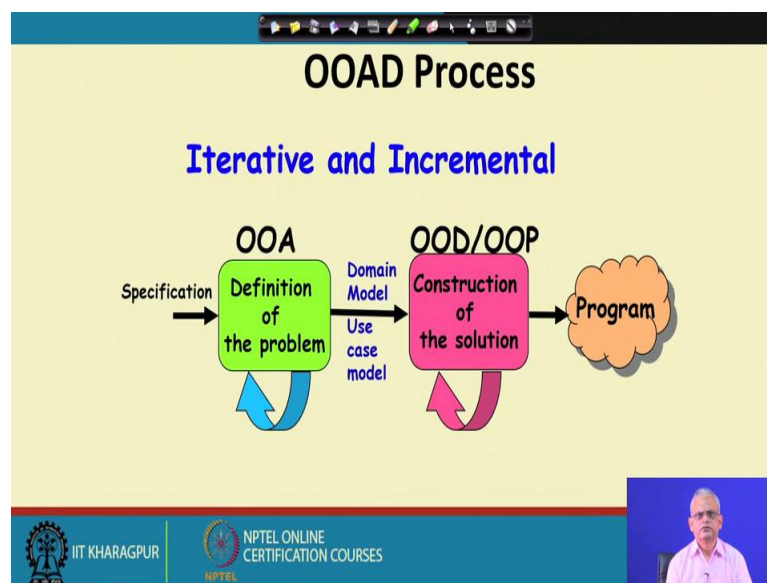
If we diagrammatically show our approach at the very overall overview of the approach is that we have an analysis stage; call it is an object oriented analysis. Here, based on the requirement specification, we analyze the requirements and develop an analysis model and this analysis model results in a domain model and use case model. So, the outcome of the analysis here are 2 diagrams. One is the use case model and the other is the domain model and then we use these 2 diagrams to carry out the design process and also the programming object oriented design and some code is also generated during the design process. Here we construct the solution. In the analysis problem we analyze the problem and model the problem. This is not really design, because we don't generate the code from here directly; it's the analysis model. But the design model takes into consideration the specific implementation that we need and the design that needs to be performed for that.

So, it is a refinement of analysis model and here some code is also generated and based on the design we can complete the program.

(Refer Slide Time: 05:23)

Let us be clear how the object oriented analysis is different from object orient design. We have already discussed in some form in the previous slide discussion but now let us be clear how object oriented analysis is different from object oriented design. In the analysis, we only elaborate the requirements, we model the requirements and we create detailed models of the requirement and the analysis model can be implemented for in any situation.

So, it is a very general purpose one and it is not specific to any implementation. On the other hand in the design step we take the analysis model and look at our constraints that we have for the solution and then we create the design model. As we proceed this point will become clear that initially we develop the analysis model, this is a very general purpose model of the problem. And then depending on the specific constraints we create the design model by refining the analysis model.

(Refer Slide Time: 06:55)
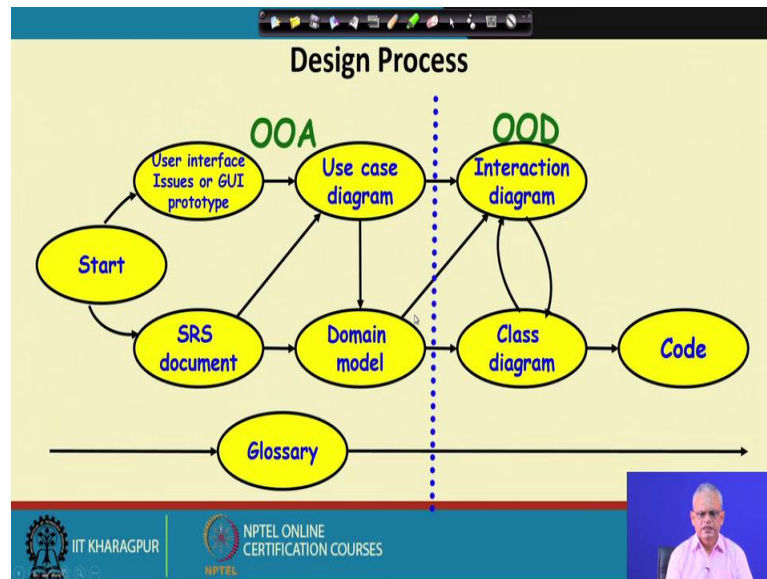
In our design process, we generate the use case diagram in the domain model during the analysis stage: these are general purpose diagram. For specific implementation, we do the object oriented design based on these two diagrams. Here we start with the SRS document available to us and we will see given the SRS document how we develop the use case diagram. For developing a use case diagram the GUI prototype is helpful to come up with a good use case diagram and we use the SRS document and the use case diagram to develop the domain model, we will see what exactly the domain model is.

Based on the use case and domain model, we develop interaction diagram, the sequence diagram and its dual that is a collaboration diagram which can be automatically obtained from the sequence diagram; we developed that the interaction diagram. Based on the interaction diagram that we develop and the domain model, we develop the class diagram. As you might remember that we were discussing the domain model has only the class names and interaction diagram as we develop it adds the methods to the classes. Then we add the attributes and so on based on the methods and we obtain the class diagram and based on the class diagram we can write the code.

We have a glossary that is prepared throughout the design process. Here as we elaborate the requirements, we might come up with new terms, concepts etc. which you add to the glossary. During the interaction diagram and class diagram, we come up with several new terms and we write the meaning of those terms in the glossary. So, the glossary development occurs throughout the design process.

(Refer Slide Time: 09:23)
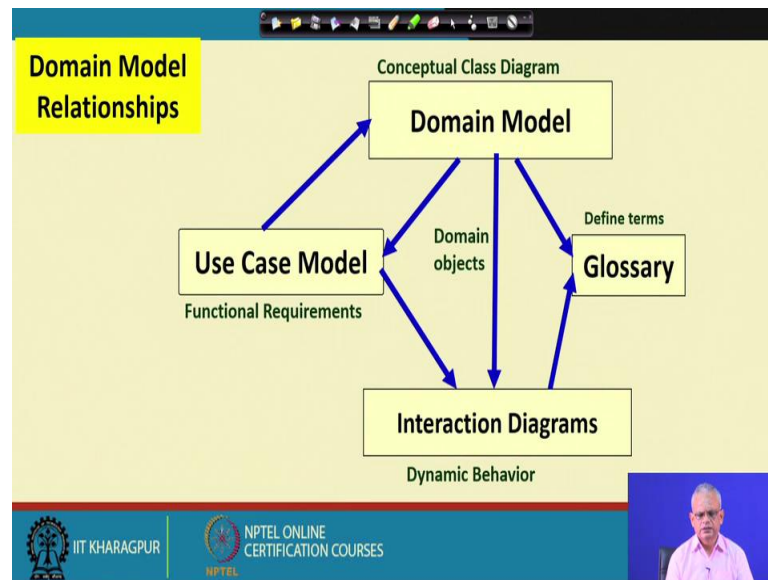
We can also model our design process in this way that initially we develop the use case model which is the functional requirement. We takes the SRS document and we create the use case model out of that and from that we develop the domain model which is the conceptual class diagram. Then based on the use case model and the domain model, we create the interaction diagrams and that will lead us to the class diagram and meanwhile we keep on defining the glossary.

(Refer Slide Time: 10:06)

So, the outcome of the analysis model is the domain model and the use case model. We have seen while we were discussing the use case diagrams how to develop the use case model. Therefore, we will not be discussing development of use case model here, we will discuss the domain model. The domain model is the initial fast cut class diagram.

We analyze the requirements document and we look at the use case model and develop the initial class diagram. We will say that it is not very complicated, if we have the use case diagram developed well and we have the requirements document available to us; domain modeling is not very complex. While developing the domain model we look for three types of objects: the boundary objects these are the user interface objects, the entity objects and the controller objects.

So, these are the three categories of objects that will look for while doing the domain model and in the domain model we represent these three types of classes. We will discuss what exactly the boundary, the entity and the controller object are.
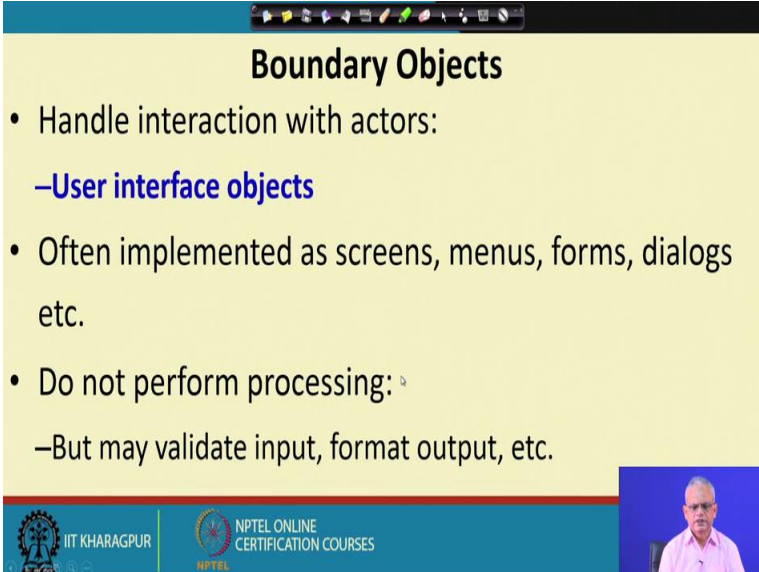
(Refer Slide Time: 12:03)

In every case tool we also use similar approaches to come up with the design and we look for three types of classes; that is boundary, control and entity. Since these classes are used frequently, we have stereotypes for that instead of writing boundary control and entity. We have the symbols in most of the case tools.

The boundary classes are the user interface for example, of cashier interface; we represent using the symbol in the figure above. These are not the UML symbol, but then most case tools, they use this symbols so that the design activity become simpler. The control; for example, a withdrawal manager is a control type of class and we represent using the corresponding symbol in the above figure. And the entity classes for example, and account class is represented by the symbol above.

So, if you are using a case tool and you come up with this kind of notation or in any other book or paper you can relate that in any specific problem during the analysis step we develop the domain model and in the domain model we look for 3 types of classes the boundary, controller and entity and these three we can have represent them using the symbols.

(Refer Slide Time: 13:58)

Now, first look at the boundary objects: these are the user interface objects and the actors they interact using the boundary objects. The different types of boundary objects are screens, menus, forms, dialog boxes, check boxes and so on. So, these are basic GUI objects and here the main function of the boundary object is to collect the user input and may be to validate and also to output the display to the user and may be format the output.

So, the boundary objects are concerned with input and output. Input data from the user using specific type of forms, screens, menus etc.: these are the GUI objects and the the

boundary objects are also used for displaying the results. But remember that these don't do any processing, they are just used for collect information from the user and to display the information.

(Refer Slide Time: 15:32)
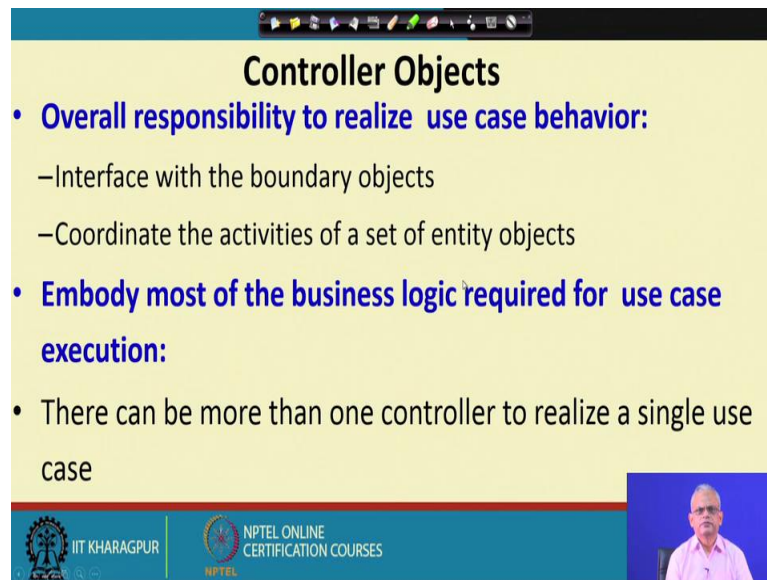
The second type of classes that we look for other entity objects: the entity objects store the information. For example, in a library we have large number of books and for each book as you create, we store the name of the book, the author, the ISBN number, the cost etc. So, this information is stored for long time. It is not that each time the book is issued etc., these are lost. Once book is created, it remains until after several years or something gets discard the book. So, the entity objects; the store data just like we saw that the book stores some data like author name, book name, ISBN number etc. Similarly book register is also an entity object. It contains what are the books available and which are issued out to whom.

Similarly we might have members register all these are entity objects. We call these objects as dumb servers because they do only very simple processing like store some data, lookup some data or may be search, sort etc. The entity objects are normally identified by reading the problem description and finding the nouns there, the nouns as you will see the mostly indicate the entity objects.

(Refer Slide Time: 17:42)

The third category of objects are the controller objects. The controller objects are in overall charge of executing the use case. As soon as a use case is executed, the user uses the boundary to enter some request, then that gets reported to the controller objects. The boundary initial request is reported to the controller object and the controller object, it interfaces with the boundary object, gets the request from the user.
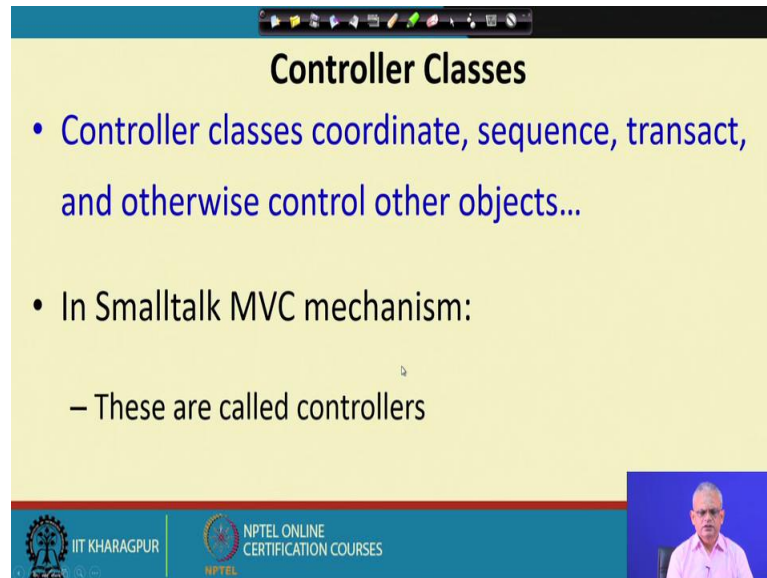
Once the request from the user comes the controller object, stores a logic inside. It knows that how to realize the required behavior i.e., which objects need to participate in this use case execution, it sends them specific messages, collects the information and sends to other set of objects and so on.

So, we can say that the controller objects are the one which have the business logic required for use case execution. Because at the start of a use case, the controller object is reported about the user trying to execute. Let us say one to issue a book then the controller object knows the business logic. It would check for example, whether the book is reserved before the book is issued to check, whether the member can actually issue, whether he has exceeded is quota and so on. Finally, it will issue it and store the information in a book register and finally print the issue slip.

So, the controller object is an important class object for every use case. The corresponding controller object gets the control and it determines what are the steps

through which the use case will executed and which objects need to interact and it request those objects to do their specific part of the execution.

(Refer Slide Time: 20:05)

We can say that the controller objects coordinate, sequence, transact and control other objects. So, these are the ones which are the brain behind the application. They coordinate the boundary objects and the entity objects and get the work done. The work that is required as part of a use case which are done by the controller object and the term possibly comes from the small talk model view controller mechanism and from there possibly in the name which is term as the controller classes.

(Refer Slide Time: 20:50)
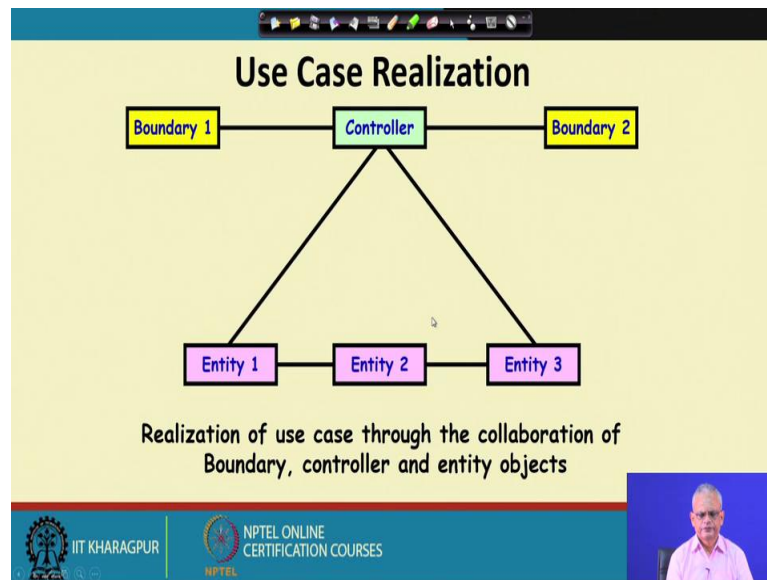
If we see how the use case execution proceeds, the boundary gets the request and it passes on the request to the controller. The controller knows the business logic that is which entity classes to request to get the work done. It sends the request entity 1 and the entity 1 may collaborate with entity 2, entity 3 and return to the controller the controller might request some other entities and so on.

So, depending on the complexity of the use case the business logic that is stored in the controller can be rather trivial or may be very complex. It takes the control and coordinates the actions of other classes and finally returns the result to the user.

(Refer Slide Time: 22:04)
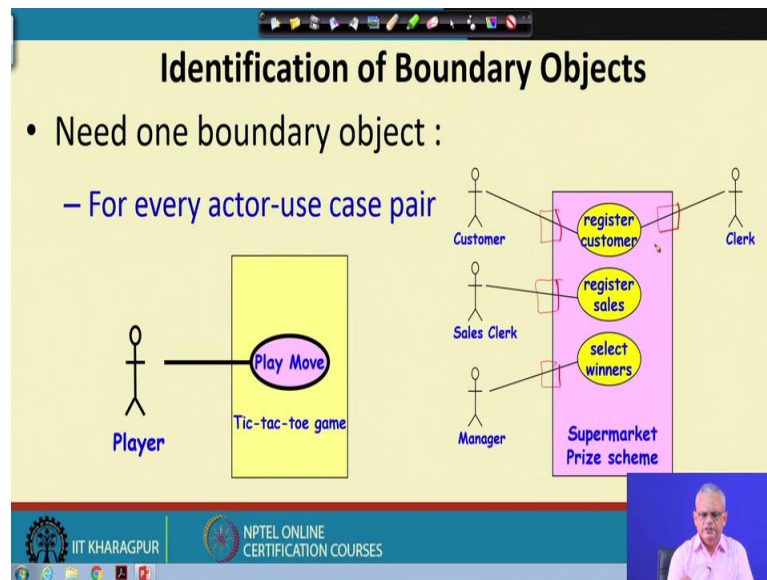
Now, let us see that how to identify these 3 types of classes. Given a problem description, we try to solve a problem, how do we go about to identify the 3 classes of objects? The simplest are the boundary objects for this we need to just examine the use case diagram. For every actor and use case we draw one boundary object that indicates the overall interface required between the use case and the actor. In this simplest use case diagram we have only one actor and one use case and therefore, we have only one boundary object.

But for example, we have 3 use cases and 4 users and we need a boundary between every user and every use case. So, we need 4 boundary objects here; so, we can say that the number of boundary objects we look at how many use case actor pair exist use case actor 2 use case actor pair 1 ones a total 4. So, identifying the boundary objects is a rather straight forward if we have the use case diagram available to us. We just look at every boundary here to the use case actor and have one boundary object for that.
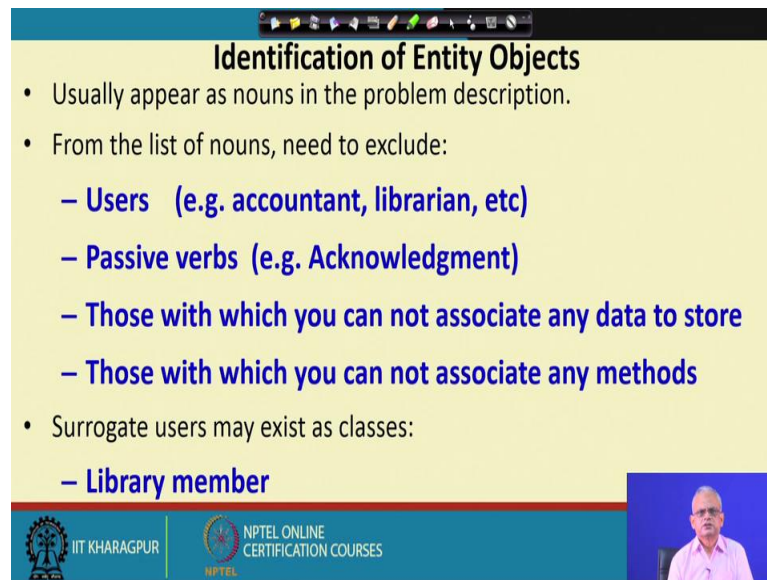
(Refer Slide Time: 24:04)
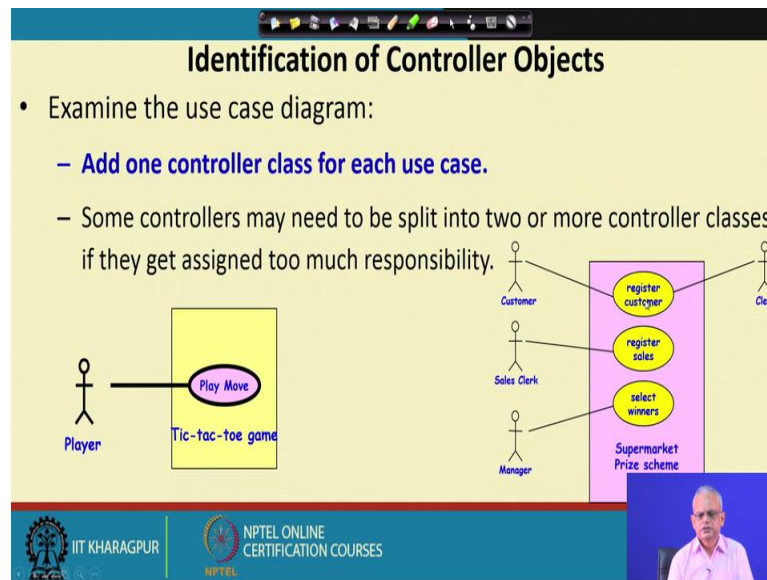
Now, let's see the controller objects. This is the second class of objects. Identifying the controller object is also straight forward. We know that each controller class realizes behavior of one use case and therefore by looking at the use case diagram, we just count the number of use cases and we need that many controller classes. Each controller class has the business logic to realize the behavior for that use case. For the first simple diagram above, we have only one controller required because there is just one use case.

For the second diagram above, we need 3 controller classes; the register customer controller, register sales controller and select winner controller. To start with we just have one controller class for each of the use cases for any problem. As we proceed with the design, we might see that for some use cases the controller class becomes extremely large with very sophisticated business logic. Then we need to split those controller classes to simpler multiple controller classes.

(Refer Slide Time: 25:33)

Now, that brings us to the third category of classes that is the entity objects. Actually these are the hardest. The boundary classes and controller classes are easily identified just by inspecting the use case diagram. We can very quickly identify the boundary classes and controller classes.

But entity classes, these require more experience and thought. We need practice identifying the entity classes, but some guidelines exists. Let us look at how to go about because this is the most crucial class. Others to identify boundary and controller classes are rather straight forward. Let's see how to go about identify the entity objects.

Typically the entity objects are identified as a noun analysis because the entity object occur in the noun. Initially when we are new to the object oriented design, we need to underline the nouns and then see which noun satisfy or they can be considered as the entity classes. But as you become more experienced, we do not have to really identify each of the noun just by reading the problem description mentally, we can determine which the entity classes are.

But to start with, we underline the nouns in the problem description and then all nouns are not entity objects, we exclude several types of noun. For example, if your nouns like accountant, library and etc. which are the users, we exclude them. But of course, sometimes we need to have some of the users also as classes for example, a library member is a class but many of this are not classes. The passive verbs also appear like

noun, but they are not really entity objects. we will look at the nouns first, eliminate the users and passive verbs and then will look at the nouns and try to associate some data and methods.

If we see that we cannot really have meaningful set of data associated with the noun and meaningful set of methods then also we eliminate that. Even though most of the users we eliminate, but sometimes we need to have some users of classes. As we were saying that library members are classes because we need to store information that who are the members, what books they borrowed and so on. These are called as surrogate user classes.

We are almost at the end of this lecture. We will stop here and we will see how to identify the entity classes for specific problems. Then we will take up few problems and then we will walk through the design process and see that how we develop the domain model, and then the interaction diagram. Finally, the class model that we will do in the next lecture. We will stop now.

Thank you.