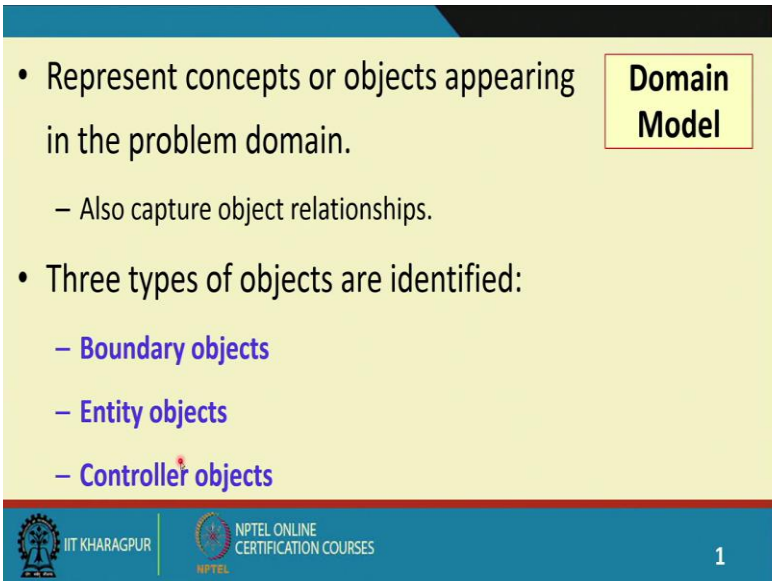
Object-Oriented System Development Using UML, Java and Patterns
Professor Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 31
OOAD - II

Welcome to the session. In the last session, we were discussing about Object Oriented Analysis and Design. We said that the object oriented analysis and design actually consists of two parts, the object oriented analysis and subsequently the object oriented design. The object-oriented analysis part is more of elaboration of the problem statement and the design part is design of the solution or the model of the solution.

In the analysis part, we said that, based on the initial use case diagram, use case construction is also analysis part, where we elaborate the problem description in text format and also construct the graphical use case format and possibly the most important thing in the analysis part is the domain model. We are discussing how to construct the domain model for any given problem, the domain model consists of the conceptual classes, concept level classes we just identify the names of the classes, that is what we call as conceptual classes and the relations among the classes. This is a very important skill for any designer to come up with the design model and we had said that we need to look for three types of objects in the problem description, one is the boundary objects, the second is the controller objects and third are the entity objects. And we are just discussing what are these objects and how to identify, let us proceed from that point onwards.

(Refer Slide Time: 2:32)



Domain Model

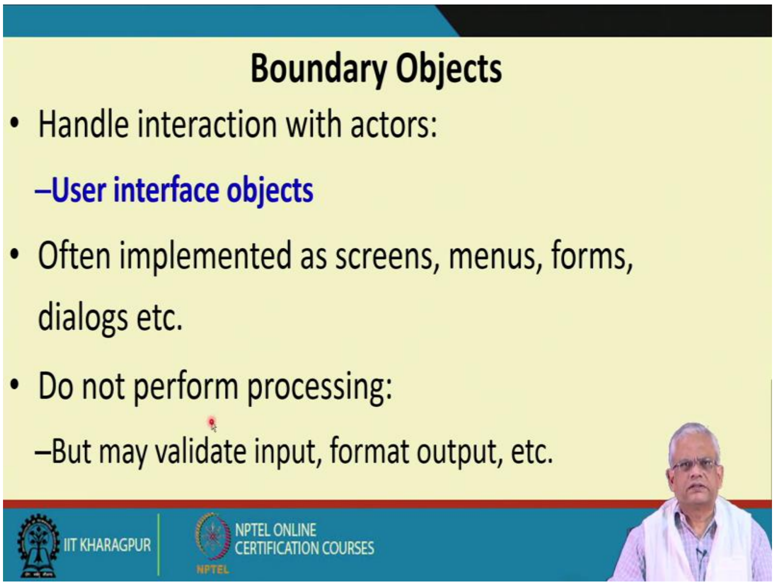
- Represent concepts or objects appearing in the problem domain.
  - Also capture object relationships.
- Three types of objects are identified:
  - Boundary objects
  - Entity objects
  - Controller objects

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
1

Domain modelling is an important part of the the analysis process. Here, we identify and represent the concepts that appear in the problem domain and also we identify the relations. Three types of objects are identified, the boundary objects, entity objects and controller objects. These three types of objects have different ways of identifying them, the boundary and controller objects are rather straightforward to identify for a problem description.

The Entity objects are the one which requires skill and experience, with experience if you solve a dozen problems, you effortlessly can identify the entity objects from the problem description but initially we need to do the noun analysis to get a hang of the things that we are looking for in the problem description. The boundary objects and the controller objects are easily identifiable from the use case diagram.

(Refer Slide Time: 4:08)

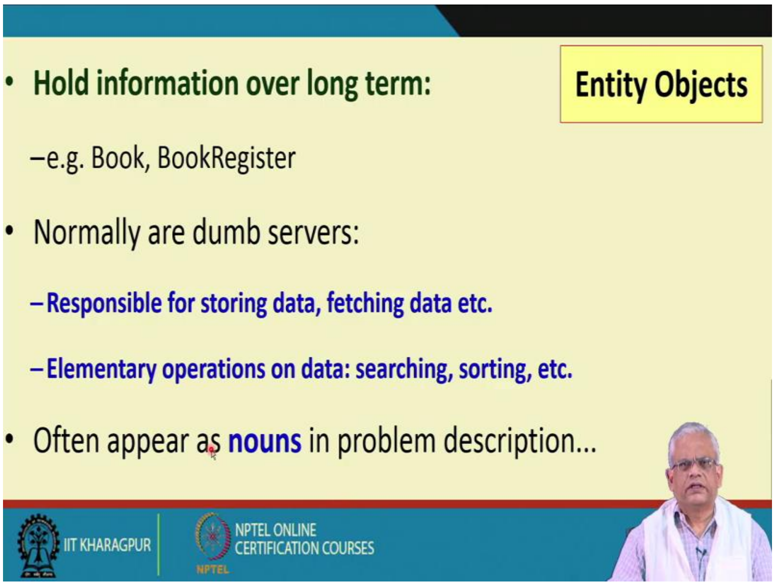


Boundary Objects

- Handle interaction with actors:
  –User interface objects
- Often implemented as screens, menus, forms, dialogs etc.
- Do not perform processing:
  –But may validate input, format output, etc.

The boundary objects are the interface between the actor and the use cases and the controller objects have the business logic for a use case. The boundary objects correspond to screens, menus, forms etc. and here as a good design principle, we should not have any processing embedded in the boundary objects other than getting the input and passing it to the controller object and similarly getting the output data from the controller object and properly formatting and passing on to the actor displaying or communicating to the actor. Other than that, the boundary objects should not have any processing activities associated with them.

(Refer Slide Time: 4:56)



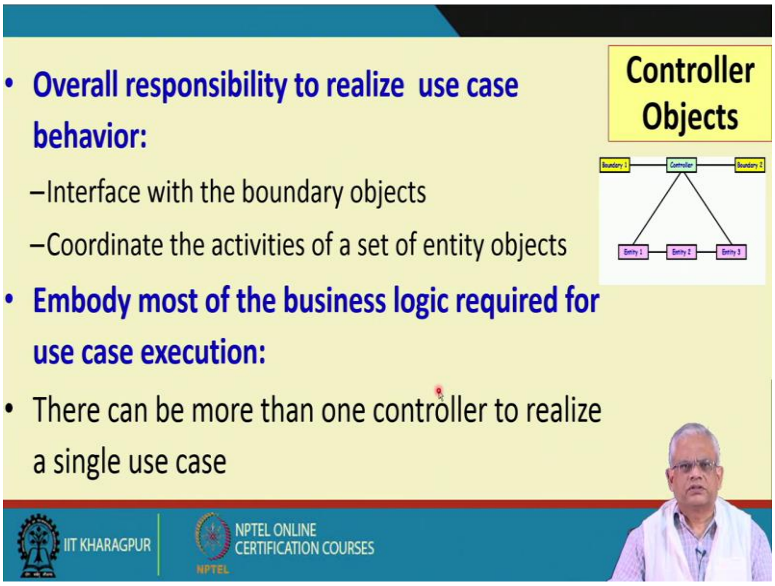
Entity Objects

- Hold information over long term:
  - e.g. Book, BookRegister
- Normally are dumb servers:
  - Responsible for storing data, fetching data etc.
  - Elementary operations on data: searching, sorting, etc.
- Often appear as **nouns** in problem description...

The entity objects, these are the crucial part of any domain modelling. The entity objects hold information for long time like once a book is created it remains for quite some time until the book is damaged, destroyed and written off. Similarly, the book register, one thing to note is that these entity objects kind of store data and on query return data and support things like searching, sorting etc. but the main purpose is to store data, for example in a library, large number of books exist and the books are the entity objects.

Each book object stores some data and the collection of all book object stores the information regarding all the books in the library. To identify the entity objects, initially will do noun analysis. We will look at the problem description, identify the nouns in the problem description, eliminate some of this and we will have the entity objects remaining, we will just illustrate that with an example but after we solve couple of problems we do not have to do a formal noun analysis. By reading the problem description it should become clear to us which are the entity objects.

(Refer Slide Time: 7:01)



Controller Objects

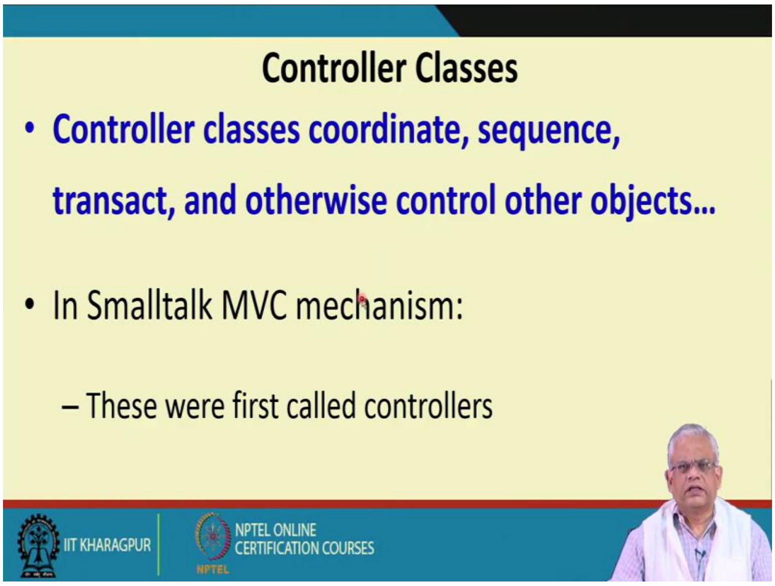
- **Overall responsibility to realize use case behavior:**
  - –Interface with the boundary objects
  - –Coordinate the activities of a set of entity objects
- **Embody most of the business logic required for use case execution:**
- There can be more than one controller to realize a single use case

The controller objects are possibly the most intelligent of the objects. The boundary does not do any processing, the entity objects do only some very elementary processing, then where does most of the logic implemented? The answer is the controller object each use case is implemented by a controller object or each use case is realised by a controller object what we mean by that is as soon as a user starts executing a use case like renew book or issue book, the boundary object gets the user data what the user really wants, that is the work of the boundary object and reports to the controller object.

The controller object knows is the intelligent object, it knows what all steps need to be done in the issue book or a renew book and then it initiates by requesting the appropriate objects in the simple example we have just it request one of these and then they in turn take help of other objects and return the data but in a more complex use case, the controller might request dozens of other objects at appropriate times to get the behaviour required for a use case implemented.

We can say that the controller object coordinates the activities of a set of objects. For every use case it has the required business logic or what needs to be done for use case execution is embedded in the controller object. If the use case has very complex behaviour requiring interactions with several dozens of objects, the controller object can become complex and to handle this situation we might have to split a controller object into multiple objects. In this case the use case will have multiple controllers but as a rule we will start with one controller per use case and later during refinement of the design we might split some of the controllers.

(Refer Slide Time: 10:03)



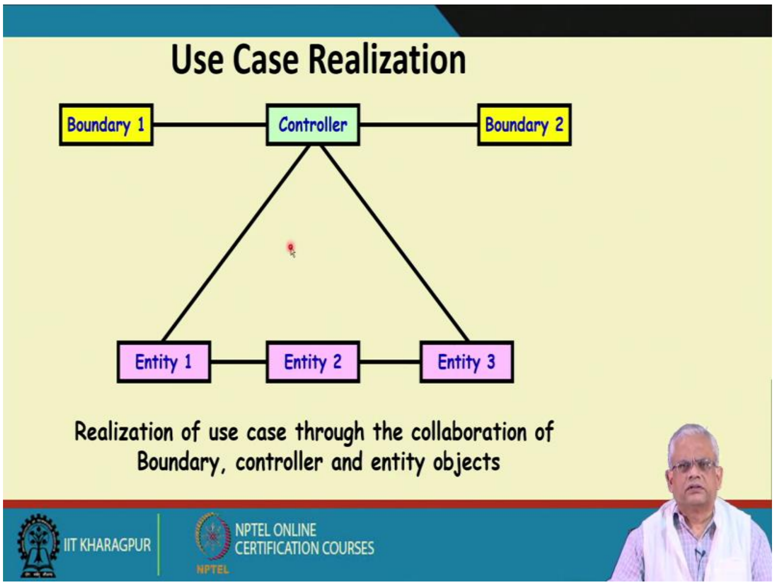
Controller Classes

- **Controller classes coordinate, sequence, transact, and otherwise control other objects…**
- In Smalltalk MVC mechanism:
  - These were first called controllers

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

The controllers have the responsibility of coordinating, sequencing and controlling other objects and the name possibly started with the small talk MVC mechanism.

(Refer Slide Time: 10:26)

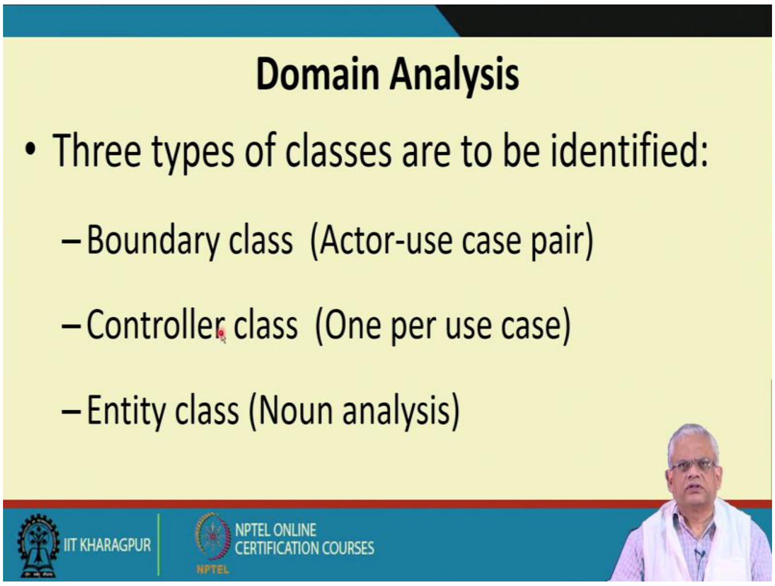


Use Case Realization

Realization of use case through the collaboration of Boundary, controller and entity objects

And we were discussing about the controller taking the help of other objects, entity objects to realise a use case. The boundary interacts with the boundary to get the input and display the output but it coordinates or requests several entity objects to get the behaviour done and of course the controller does some bit of processing.

(Refer Slide Time: 10:50)



Domain Analysis

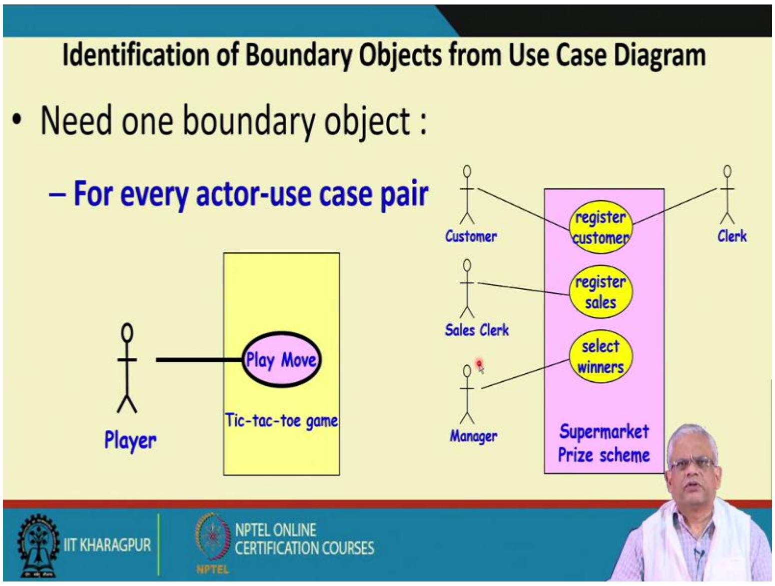
- Three types of classes are to be identified:
  - Boundary class (Actor-use case pair)
  - Controller class (One per use case)
  - Entity class (Noun analysis)

Now, during the domain analysis we have to look for three types of objects, boundary objects, controller objects and entity objects. The boundary objects we need to identify or we need to construct one boundary object for every actor-use case pair. One controller object for one use case for every use case we will have one controller and for every actor-use case pair will have one boundary object.

And therefore, construction of these two types of objects is almost straight forward just by inspecting the use case diagram, we can identify how many boundary objects and how many controller objects will be needed in the initial solution but possibly the most crucial part is the entity class require some experience and expertise by reading the problem description that is the SRS document and consulting the use case diagram.

We should be able to identify the entity classes with some experience, we will just try to solve couple problems to identify the entity classes and will have several problems displayed here and request you to solve those to get the required expertise to identify the entity classes and the quality of the solution represents is determined by how are entity classes identified and represented. These two of course all designers can easily do the boundary classes and the controller classes.

(Refer Slide Time: 12:51)

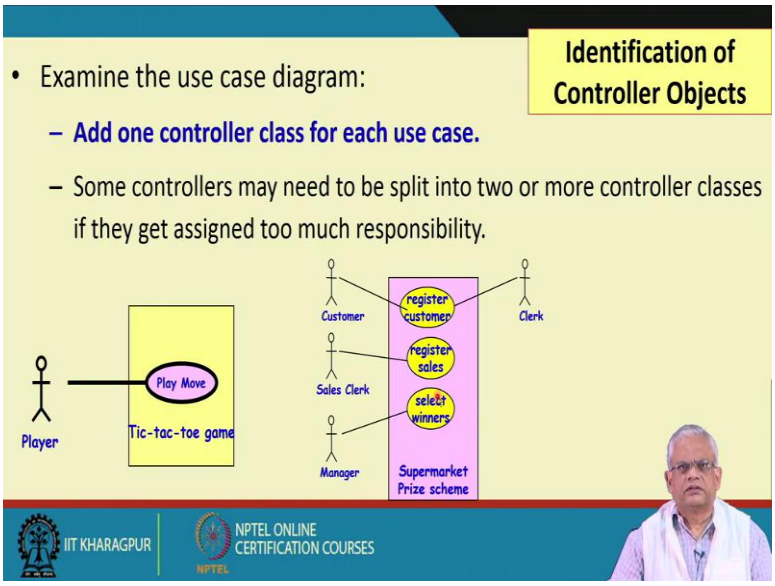


Identification of Boundary Objects from Use Case Diagram

- Need one boundary object :
  - For every actor-use case pair

The boundary objects straight forwar as in the use case diagram and for every actor use case pair, we need one boundary it has won boundary and we need one boundary object. Similarly, for this example as shown in the above figure, there are 4 actors and three use cases but there are four actor use case pairs and therefore we need 4 boundary objects for this problem. Given a use case diagram, everybody will find it very easy to identify the boundary objects and later in the design process these boundary objects will become user interface components like menus, checkboxes, text boxes and so on.

(Refer Slide Time: 13:55)

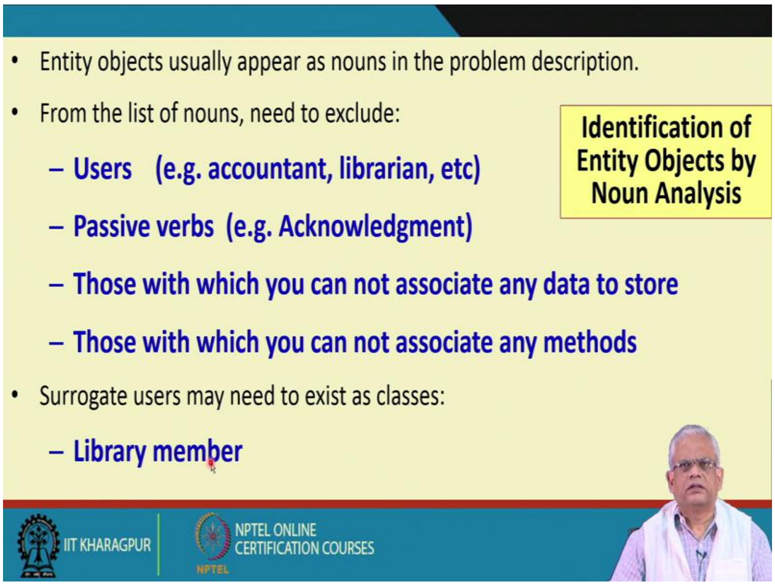


Identification of Controller Objects

- Examine the use case diagram:
  - Add one controller class for each use case.
  - Some controllers may need to be split into two or more controller classes if they get assigned too much responsibility.

The controller object identification is also straight forward, look at the use case diagram and for each use case we need one controller and in this simple example we need exactly one controller classes and we will name it as a play move controller. For this example we need 3 controller classes, the register customer controller, register sales controller and select winner controller.

(Refer Slide Time: 14:34)



- Entity objects usually appear as nouns in the problem description.
- From the list of nouns, need to exclude:
  - Users (e.g. accountant, librarian, etc)
  - Passive verbs (e.g. Acknowledgment)
  - Those with which you can not associate any data to store
  - Those with which you can not associate any methods
- Surrogate users may need to exist as classes:
  - Library member

Identification of Entity Objects by Noun Analysis

Now, let us try to identify the entity objects as this is the most crucial part of the design process. We must understand that the entity objects typically appear as nouns in the problem description, some of the nouns in the problem descriptions are the entity objects but then not all nouns in the problem description are entity objects. We need to eliminate once we underline the nouns, we need to eliminate most of this.

If it is the noun corresponds to a user, such as accountant, librarian, etcectra we will eliminate it because these are actors rather than entity classes. Some nouns are actually passive verbs, they are verbs but then in the text they are written like a noun, for example, acknowledgement, acknowledge is the verb and acknowledgement is the noun form of this verb and these also are not entity classes.

These are basically verbs like acknowledgement etc. are verb forms, even though they appear as nouns, these are verbs and later we will say that if the nouns correspond to the entity objects, the verbs correspond to methods. The nouns correspond to entity objects, some of the nouns and the verbs in the problem description they correspond to the methods but the question remains that which class will have which method and that we said that the interaction diagrams will help us to identify which verb or which method belongs to which class.

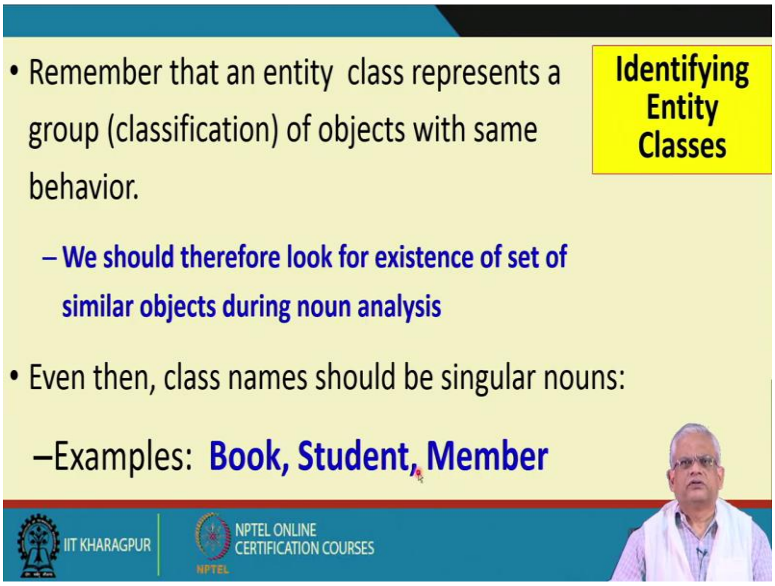
So, we need to eliminate the passive verbs from the identified nouns and we will also examine the nouns that we identified and find out with which we cannot identify any data to store because after all the entity objects need to store data that is the basic definition of entity objects. If we cannot think of any data that this entity object will store, then we need to eliminate it, it is possibly an attribute of another class. We will discuss some examples there, we will again remark this.

And those nouns with which we cannot identify any method at all, for example, store, get, etc., we should be able to identify such methods with the entity objects. If we cannot even identify any methods which will be associated store, get etc., then these are not entity objects. The examples that we discuss, we eliminate the identified nouns based on this criteria but of course, we should be careful that sometimes we actually have entity objects which appear as users.

Even though the library member is a user but then we need to have an entity object as a library member because it will store the member details, the books outstanding and so on. Many times we will have the users appearing as entity objects but not always. Whenever there is a user, the noun form user is present we need to check whether we need to store such information associated with the user.

For library member, we need to find out who are the library members, what are the books outstanding and so on. And we call this as the surrogate users.

(Refer Slide Time: 19:30)



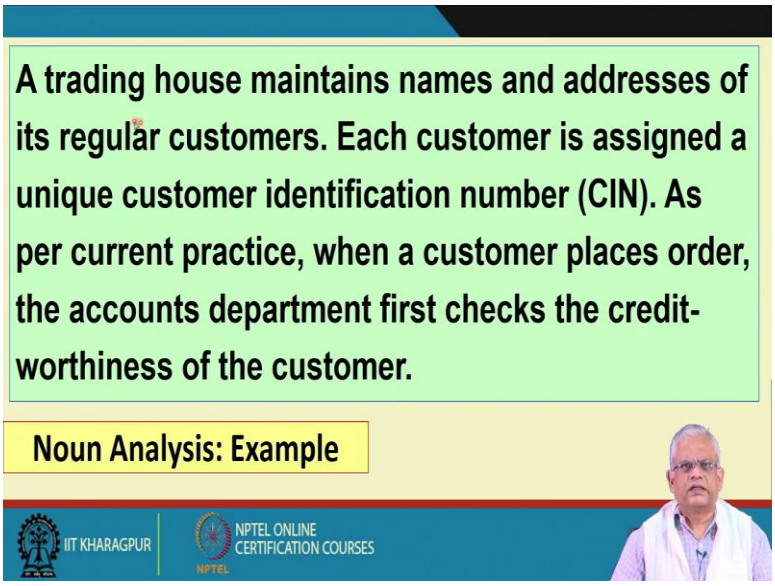
Identifying Entity Classes

- Remember that an entity class represents a group (classification) of objects with same behavior.
  - We should therefore look for existence of set of similar objects during noun analysis
- Even then, class names should be singular nouns:
  - Examples: Book, Student, Member

We will see that the entity objects typically occur in groups, for example, book, there are thousands of books they occur in a group, let us say library member, several library members and they occur in group. So, this is possibly a hint that if there are several objects of the same type, then it is possibly an entity object but even though this represent a large number of objects.

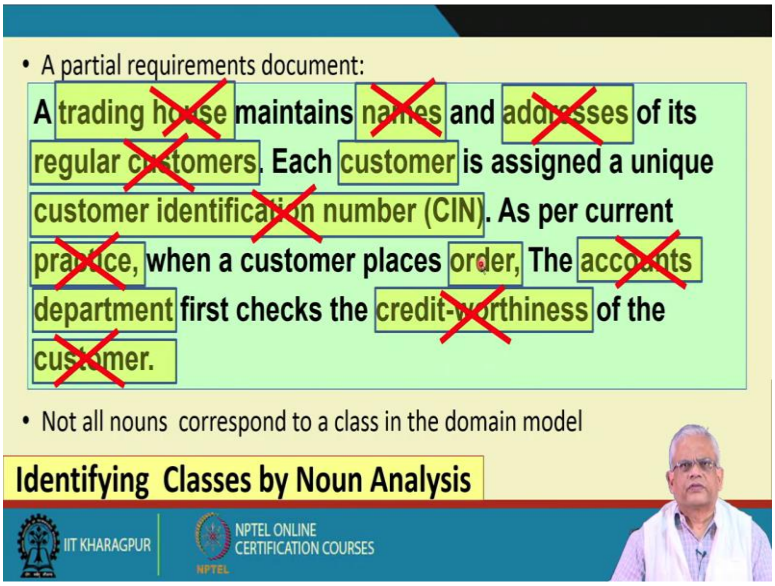
For example, the book, book class represents from where we instances thousands of books but please remember that the name of the class will be in singular, we do not have, we should not have a class called as books, the class name is book. The class name can be student, students is not proper, the class name students in plural is not proper, please try to give singular name to the identified entity classes, even though they represent large number of objects.

(Refer Slide Time: 21:02)



A trading house maintains names and addresses of its regular customers. Each customer is assigned a unique customer identification number (CIN). As per current practice, when a customer places order, the accounts department first checks the credit-worthiness of the customer.

Noun Analysis: Example



- A partial requirements document:

A trading house maintains names and addresses of its regular customers. Each customer is assigned a unique customer identification number (CIN). As per current practice, when a customer places order, The accounts department first checks the credit-worthiness of the customer.

- Not all nouns correspond to a class in the domain model

Identifying Classes by Noun Analysis

Now, let us look at one simple problem description and let us perform noun analysis. A trading house, this is part of a problem description, we can do for the full problem description but let us just do for the part now. A trading house maintains names and addresses of its regular customers. Each customer is assigned a unique customer identification number or CIN. As per current practice, when a customer places order, the accounts department first checks the creditworthiness of the customer.

Now, let us do the noun analysis, let us identify all the nouns. Trading house is a noun we will underline that, name is a noun, address is a noun, regular customer is a noun, customer is a noun, customer identification number is also a noun, practice is a noun, order is a noun, accounts department is a noun, credit worthiness is a noun and customer is a noun.

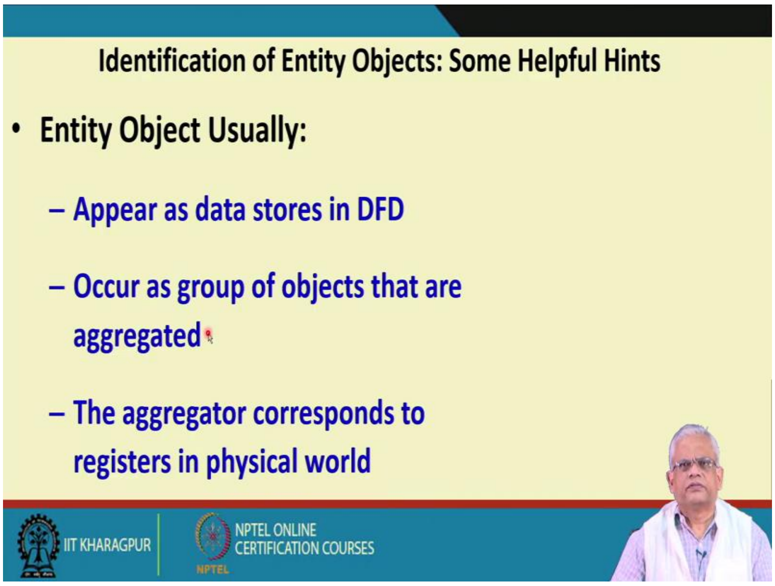
The first thing we should possibly do is to eliminate these duplicate ones. We should take one of that and let us apply the criteria that we are discussing that the trading house is the name of the organisation and in the software we do not have to really store data about the trading house, we cannot really associate any methods with the trading house, for example store what do we story about a trading house? So, we will eliminate it, what about name and address? To the name and address also we cannot associate any method, these are attributes actually of the regular customer, we do not change the name of a customer normally, and we do not separately store names and addresses. So, we will eliminate that regular customer, we need to store data about customer but then the regular customer and customer they are synonyms, so we will eliminate regular customer and we will just have the customer.

Like the customer identification number, we cannot store data about customer identification number, this is just one data item and its an attribute of the customer rather than an entity by itself just one data item and typically an entity stores multiple data items and we cannot really associate any meaningful methods with the customer identification number, it should be an attribute of the customer.

Similarly, practice is the noun form of verb we should eliminate it, order is a noun we can associate data and methods with the noun and therefore we should have this as a entity object. Accounts department, we do not need to really store data about accounts department, we will eliminate that. Creditworthiness is also not data to be stored, it is not entity object and customer is a duplicate, here we have appearing customer two times we will eliminate one of that.

So, that leaves us customer and order are the two meaningful entity objects based on the noun analysis here, we could identify the nouns, eliminate the ones which are not entity objects and got two entity objects from this problem description, you can read through the entire problem description and identify the entity objects but then after practice and problem solving for couple of problems you will see that just by reading here you can make out that customer and the order are the only entity objects in this text description.

(Refer Slide Time: 26:44)



Identification of Entity Objects: Some Helpful Hints

- Entity Object Usually:
  - Appear as data stores in DFD
  - Occur as group of objects that are aggregated
  - The aggregator corresponds to registers in physical world

But we should be, there are some helpful hints we can mention in identification of the entity objects. One is that if we know procedural design, then in the data flow diagram the data stores typically are the entities, if we look at the DFD model the data stores correspond to entity objects and the entity objects almost always occur in groups of large number of objects and these are aggregated like a book register aggregates large number of books.

And typically we use something like register or something member register to aggregate the members, book register to aggregate the books and therefore those and also entity objects, we identify the books is a large number of them and book is an entity object and also a book register is an entity object. We are almost at the end of the session, we will stop here and continue in the next session, thank you.