Object-Oriented System Development Using UML, Java and Patterns

Professor Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture 32

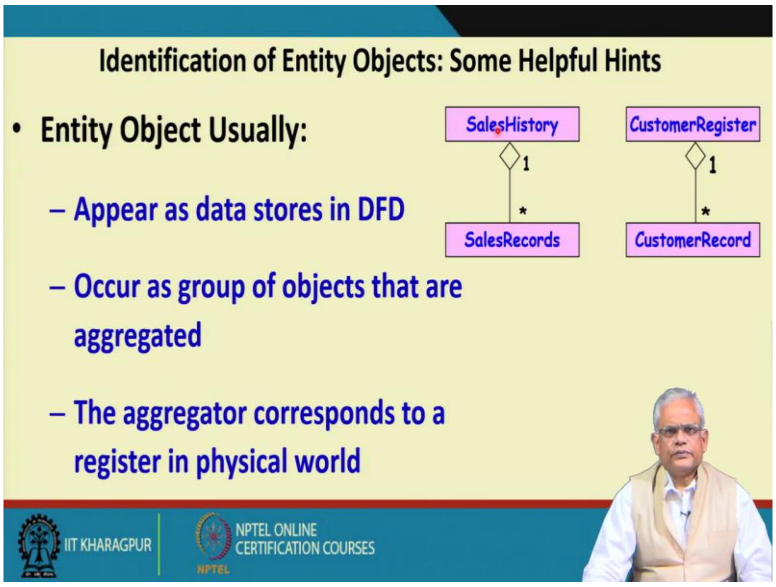
Example Application of OOAD

Welcome to this session. In the last session, we had discussed about identification of the domain objects which is a very important activity during design. The boundary and the controller objects are effortlessly identified just from an inspection of the use case diagram but identification of the entity classes requires some experience and in the last session we were discussing how to identify the entity classes based on noun analysis.

The entity classes typically appear as nouns in the problem description but then not all nouns are entity classes, we had discussed how to eliminate some of the nouns to get the entity classes. With a simple example, we underlined all the nouns and eliminated those nouns which cannot be entity classes.

For example, those who do not store data, we cannot associate methods, synonyms, noun form of verbs and so on to them but the good thing is that with a little bit of practice once we do few problems we do not really have to formally do a noun analysis. Just by going through the problem statement, it becomes clear. By reading the problem statement couple of times, it becomes clear which are the entity classes and I am sure that all of you will achieve that level of expertise after doing few problems.

(Refer Slide Time: 2:33)



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 a register in physical world

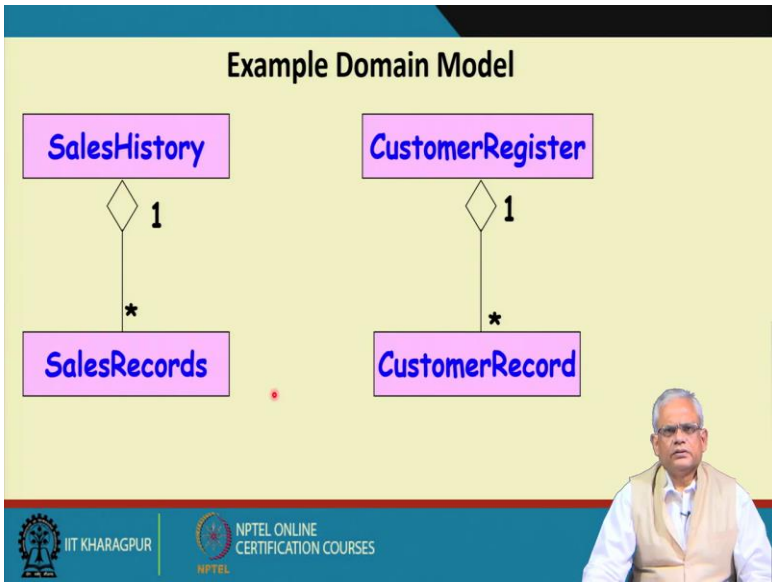
Now, let us proceed from where we had left last time about entity class identification. These are some of the helpful hints which can help you to identify the entity classes from the problem description. One is that if you have experience in procedural design, then the entity classes are the data stores.

The entity classes in almost all cases occur as an aggregation. There are only few cases where entity classes may appear isolated as a single class and later once we discuss about the design patterns, we will look at the singleton pattern which deals with the entity classes which are singleton but in overwhelming cases 99 percent or more than that the entity classes almost always appear as groups.

So, that is a hint if some class is appearing as a large collection of objects, then it is likely to be an entity object and when there is an aggregation of the entity objects, we typically have an aggregator for those objects and these correspond to the registers like in the library earlier when it was manual we used to have a library register where all the issue book etc. are entered and also we had book register in the library which identifies all the books that are available in the library.

We can do similar thing here, if we have a sales records large number of sales record objects are aggregated into a sales history object: this is the aggregator, sales history object is the aggregator of the sales record. And similarly, for the customer records the large number of customer records and we have this aggregator customer register. For large number of books in the library problem we will have a book register. So, when we identify the entity objects, these will typically appear like entity objects which store bits of data and these are aggregated by the aggregator object like sales history and we know from our discussion so far that the sales history object is the one which creates the sales records calls- the new method on the sales record to create objects the sales record and stores their references.

(Refer Slide Time: 6:02)

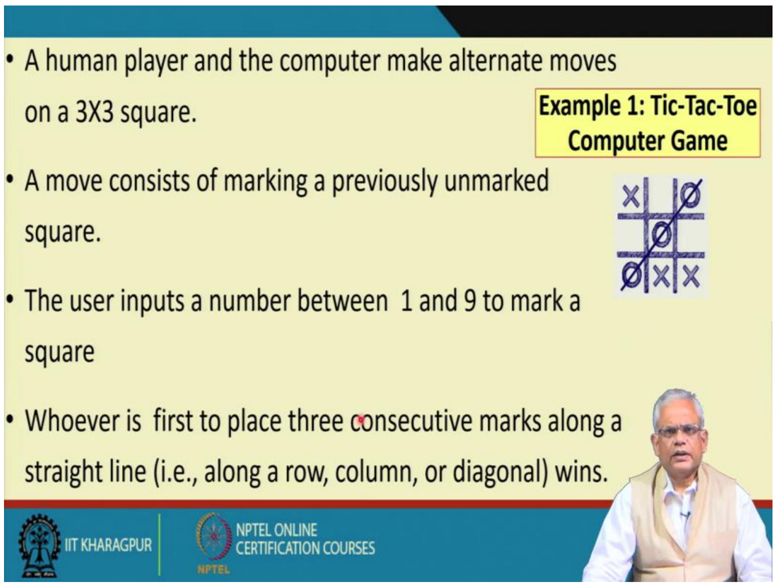


Example Domain Model

The entity objects typically appear in this form each of them store some data, little bit of data but together they store huge amount of data. For example, for the library there may be ten thousand books and the book details of every book is stored in one object and there are large number of objects. This is one important difference we had mentioned earlier that in procedural approach, the data is centralized.

We have a single array or array of structures which stores the data about books and this is the centralized global array which is accessed by all methods but here in object orientation the data is distributed among many small objects. We will work out few problems and there we will find that the entity classes typically appear like there is an aggregator typically called register or history or something and that aggregates the entity objects. We will have few practice problems for you and there also you would have to identify this kind of entity objects. This is the important experience that will be needed.

(Refer Slide Time: 7:43)



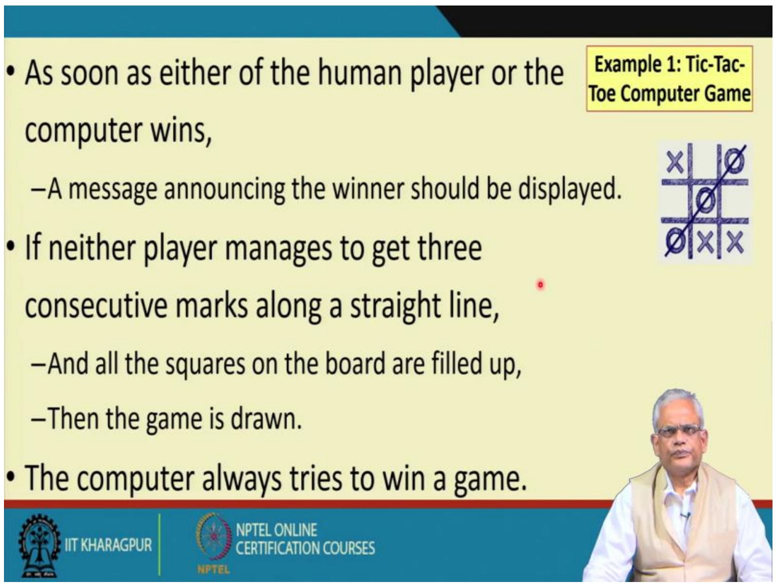
- A human player and the computer make alternate moves on a 3X3 square.
- A move consists of marking a previously unmarked square.
- The user inputs a number between 1 and 9 to mark a square
- Whoever is first to place three consecutive marks along a straight line (i.e., along a row, column, or diagonal) wins.

Example 1: Tic-Tac-Toe Computer Game

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Now, let us do one problem, the first problem is the simplest one is about a tic-tac-toe game. The tic-tac-toe game is written down here in the diagram but in a nutshell we will have a board which appears in the computer screen and then the user selects or marks a square by clicking on it and once he clicks, the computer displays a counter move, so the computer and the human player that take turns in marking the square and anybody gets three on any one of a straight line or a diagonal or in the row or the column wins. But if all the squares on the board are filled up and nobody wins that then the game is drawn. So, that is the game I hope that many of you would have played such games and the statement of that is written down here thatI just briefly explained this game.

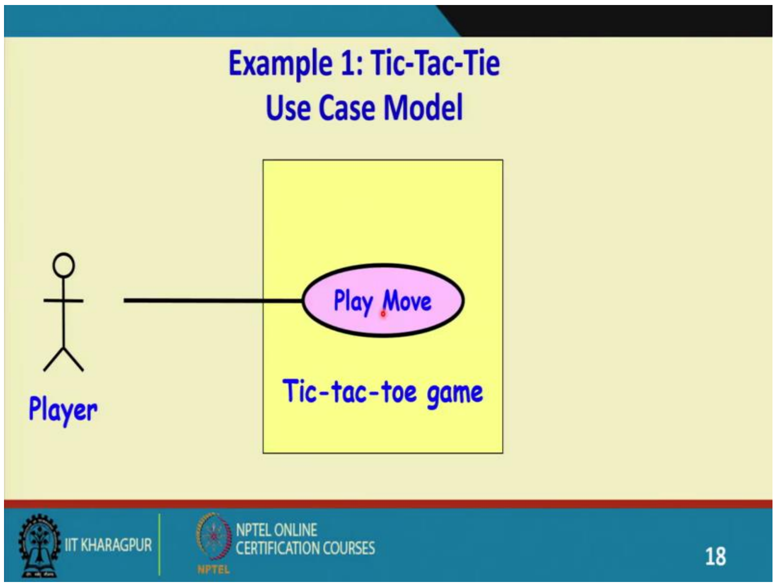
(Refer Slide Time: 9:03)



- As soon as either of the human player or the computer wins,
  - A message announcing the winner should be displayed.
- If neither player manages to get three consecutive marks along a straight line,
  - And all the squares on the board are filled up,
  - Then the game is drawn.
- The computer always tries to win a game.

Example 1: Tic-Tac-Toe Computer Game

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES



Example 1: Tic-Tac-Tie Use Case Model

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 18

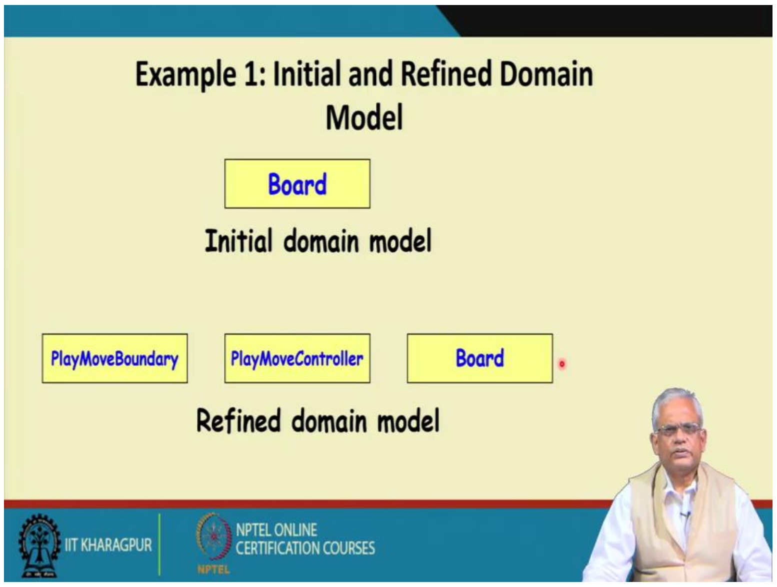
Now, how do we go about designing a solution for this? The tic-tac-toe computer game the first thing would be to draw the use case diagram for this problem we had earlier discussed that the use case diagram consists of only one use case that is the play move the only functionality that the user can invoke here in the simplest game is the play move of course we can make the problem statement more complex saying that we can set the level of the, player can set the level of the game, the game history can be stored and viewed the player details can be stored, etcetera, etcetera. But in the simplest form of this game that we have described here there is only one use case that is the play move use case and then there is a single actor which is the player.

Now, the next step would be to identify the domain model and in the domain model the boundary and the controller classes identification is almost trivial. There will be one controller class corresponding to the play move use case and we can call it as play move controller. There is only one boundary here we can call it as player play move boundary or simply we can call it as boundary because there is only one boundary, so there is one controller we can even call it as the controller.

Now, the crucial part is about identification of the entity classes. If we read through the problem, we will try to identify the nouns and which store information permanently or semi permanently have some meaningful methods associated with them that is the characteristic of the entity objects. If we read through the problem, we will find that there is only one entity class which stores data that is the board.

Once a mark is made either by the computer or the player it remembers who has marked there and then we can retrieve the marks and place the marks and that is the only entity class actually.

(Refer Slide Time: 12:25)



Example 1: Initial and Refined Domain Model

So, if we draw this will have this board as the only entity class and we have this we can add the controller and the boundary class we said that there is only one boundary class and one controller class and therefore this is the domain model. Of course, we can write a board consists of squares but then the squares are only nine and the entire nine square data can be easily stored inside the board class its not too much data and therefore we have the single class we do not use an aggregate here board contains many squares its not necessary that level because it will be an unnecessary overhead.

Now, the next step in the design process is to develop the interaction diagram namely the sequence diagram. Remember that for every use case, we need to develop one sequence diagram and here there is only one use case that is play move. To be able to develop the sequence diagram we need to thoroughly understand the problem description and then remember the steps that occur once the use case is triggered.

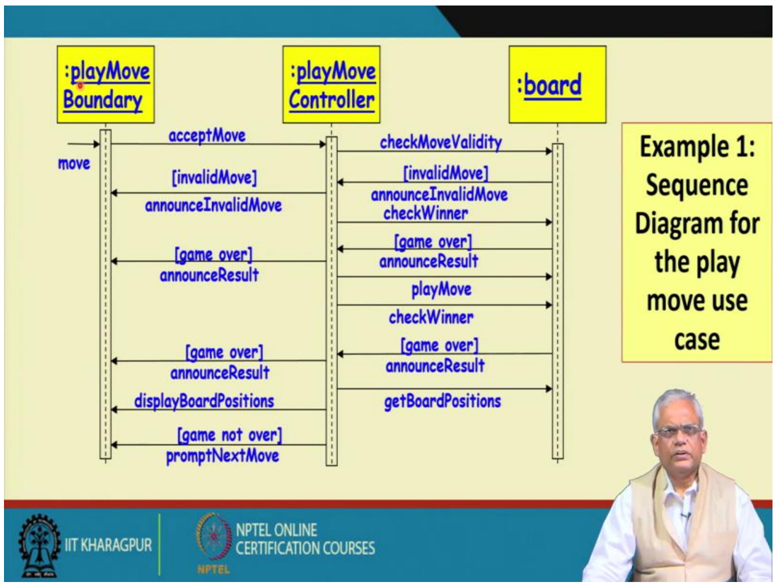
For example, in this case we must understand that once the use case play move is triggered, first it is checked whether the mark that the user is trying to place is a valid mark, he is not marking on something already marked and he is marking on a valid square its not marking outside the board.

If he does that will have to prompt for him to mark on a proper square and if the mark is proper then the mark is stored on the board on the appropriate square and also we need to check that if anyone has own with this mark is the player own by placing this mark that has to be checked that whether any of the marks are in a line.

And then in this use case execution the next step would be for the computer move because the human has made a move and if it is not a win or a draw, if it is a win then a congratulatory message has to be displayed. If it is a draw, drawn game has to be displayed otherwise the computer has to make the move and for this the computer needs to consult the board positions and make a move based on some algorithm.

And then the move of the computer needs to be stored in the board and also it needs to be again checked if with this mark placed whether the computer wins or the game is drawn in that case appropriate message has to be displayed. Otherwise the human player will be asked to play the next move the board should be refreshed and the human player asked to play the next move. If we understand that these are the activities or the steps that occur in response to the play of use case, you can represent that in the form of a sequence diagram.

(Refer Slide Time: 16:40)



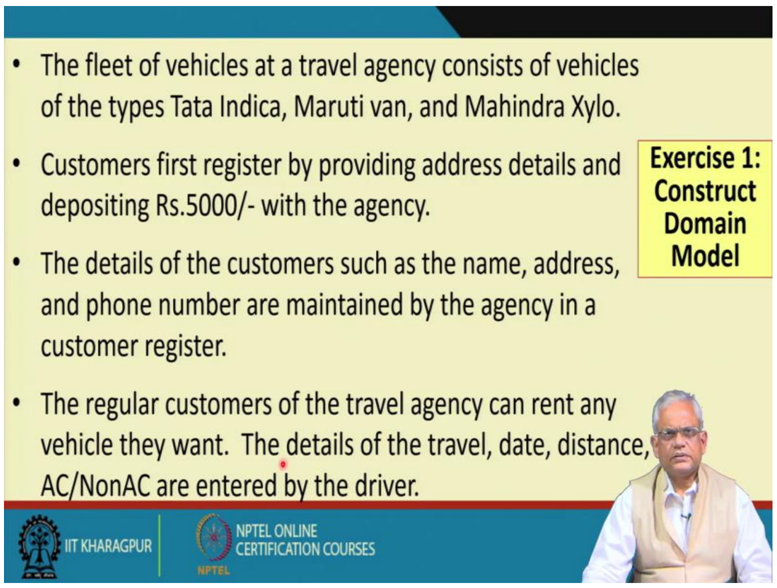
So here, we have these three, all the three objects participate in the use case play move, once the move is given by the user, then it reports to the controller, calls the method on the controller and the controller does not have the data to check whether it is a valid move because the controller does not store the board data, it requests the board to check move validity.

If the move is invalid, then it is announced that it is invalid move otherwise the move is recorded and check winner. If there is any winner that needs to be announced or the game is drawn that needs to be announced we have this guard here, that means these methods are conditionally executed and if the game is not drawn, then the computer has to make the move, but who can make the move? The controller cannot make the move because it does not have the data about the board positions and the board has all the data and therefore we need to associate this method here about play move.

And then after play move the controller which has all the business logic that what needs to be done, which step after which step etc. These are all encoded in the play move controller and the next thing it asks the board is to check for the winner and then if there is a decision there is a winner, the game is drawn it is announced and the board is displayed, gets the board positions and displays and then prompts for the next move.

And once we have drawn this sequence diagram, the three classes here the methods are automatically populated in a case tool or manually we can check here that board will have check move validity, check winner, play move and the get board position. These are the methods that should be there with the board and for the controller. Similarly, will have the accept move, announce invalid move, announce result, these are the methods that should be populated in the play move controller and similarly with the play move boundary, we can do that easily and I have not shown that here.

(Refer Slide Time: 19:56)

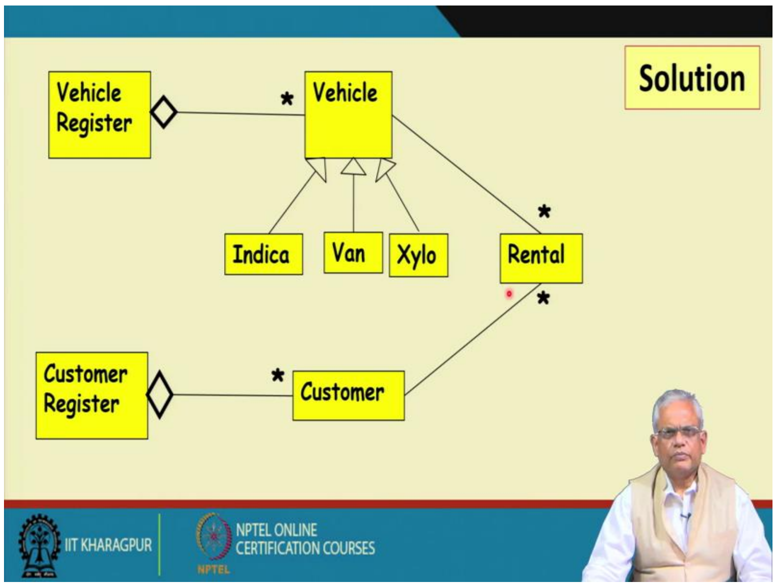


- The fleet of vehicles at a travel agency consists of vehicles of the types Tata Indica, Maruti van, and Mahindra Xylo.
- Customers first register by providing address details and depositing Rs.5000/- with the agency.
- The details of the customers such as the name, address, and phone number are maintained by the agency in a customer register.
- The regular customers of the travel agency can rent any vehicle they want. The details of the travel, date, distance, AC/NonAC are entered by the driver.

Exercise 1: Construct Domain Model

Now, let us try few more examples about the domain model, this is another problem the fleet of vehicles at a travel agency consists of different types of vehicle, the travel agency maintains vehicles like Tata Indica, Maruti van, Mahindra Xylo, etc. and the customers of the travel agency need to first register with the agency by depositing some amount of money and giving the details like name, address, phone number, etc. and the regular customers who have registered they can rent a vehicle and the details they need to provide which date they want to travel the distance, AC, non-AC etc. and then this is once that travel is complete the driver enters these details.

(Refer Slide Time: 21:07)

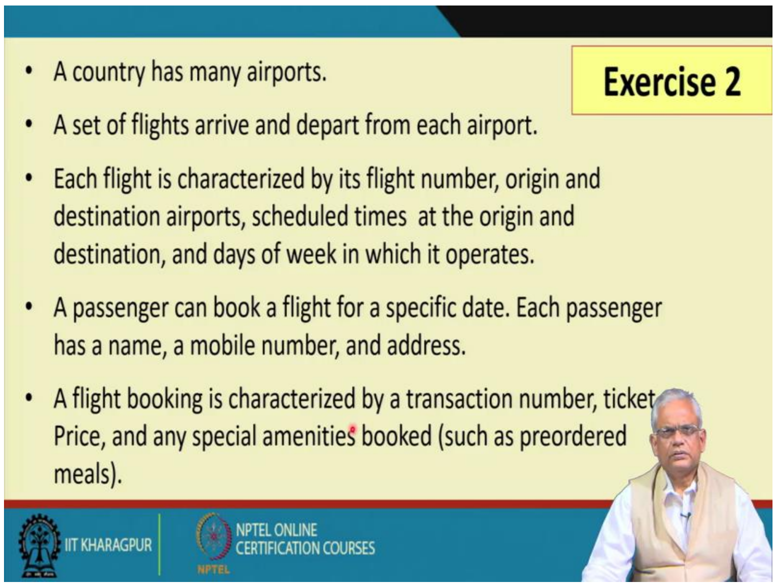


And we can draw the domain model here based on the problem description the first statement said that the travel agency maintains vehicles of different types, so vehicle is an entity class and there is a vehicle register which aggregates all the vehicles, the travel agency may have two dozens of vehicle, so these are all aggregated each vehicle the data is here but then the vehicle can be of three types Indica, Van or Xylo and that should have mentioned here.

And since vehicle is the base class and these are the derived classes in the array of vehicle either there can be Indica at a position, array position or can be a Van or a Xylo. Now, similarly the regular customers who register, we have an object here the customer object and aggregated by the customer register.

So, in the sequence diagram once the customer registers we will have object, customer object created here because the customer register aggregates the customers and then the customer may have a rental and the customer can take many rentals and that is what we have mentioned here a single customer one on this side can take many rentals. And a vehicle may be associated with many rentals, we can also represent this as the association class here between the customer and the vehicle. We can just use association class rental but both of these are equivalent, so this is the solution.

(Refer Slide Time: 23:22)

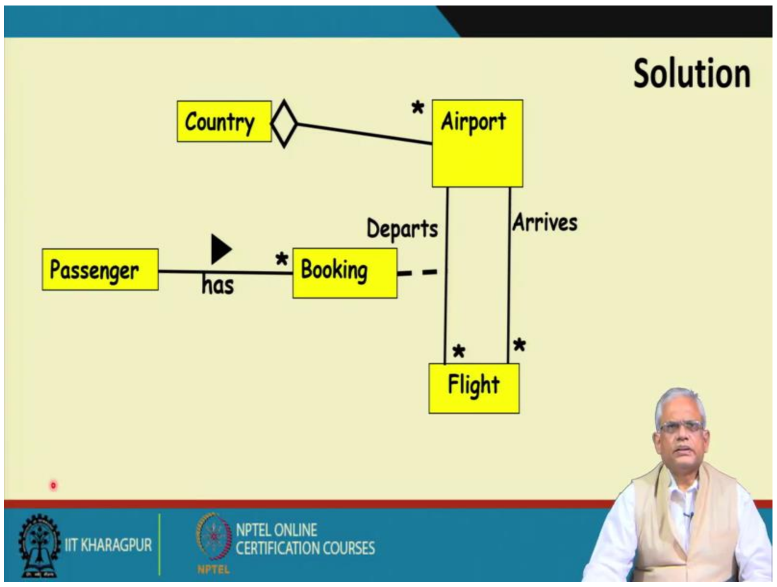

Exercise 2

- A country has many airports.
- A set of flights arrive and depart from each airport.
- Each flight is characterized by its flight number, origin and destination airports, scheduled times at the origin and destination, and days of week in which it operates.
- A passenger can book a flight for a specific date. Each passenger has a name, a mobile number, and address.
- A flight booking is characterized by a transaction number, ticket Price, and any special amenities booked (such as preordered meals).

We will just look at another problem because here practice is a very important thing. A country has many airports, so that means a country aggregates many airports and set of flights arrive and depart from each airport, flight is an object, airport is object and flight is associated to the airport there are two associations, one is that it arrives that is one association and depart is another association.

Each flight has some details that will be stored inside the flight object, passenger is object aggregated by a passenger register and then the passenger is associated with a flight and the passenger details are given here and then the flight booking is characterized by the transaction number and so on, so the passenger will make a booking and the booking is for a specific flight we can represent that here.

(Refer Slide Time: 24:41)

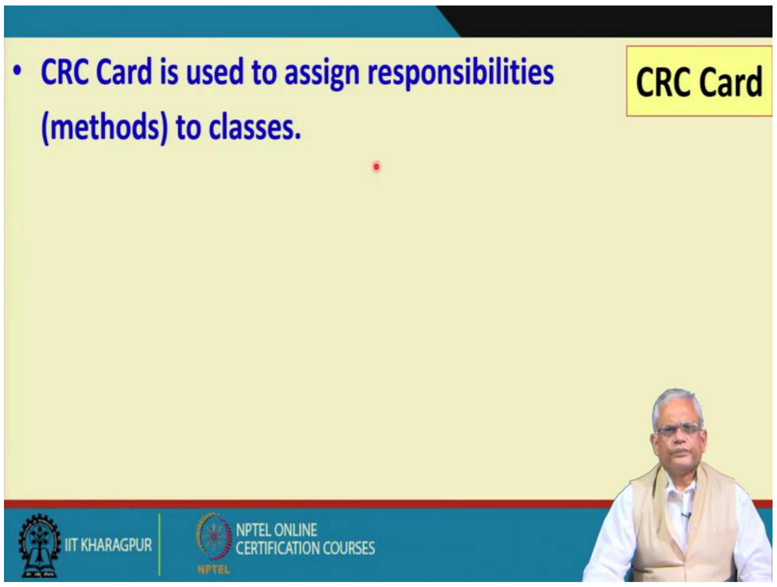


Solution

So, observe here that we have the country has many airports, so this is the first thing we identified and then a flight is associated with an airport, the flight departs from the airport and the flight arrives at the airport. So, in one airport multiple flights can depart and multiple flights can arrive and this is the notation we had discussed earlier and a passenger can have a booking, a passenger can have multiple booking with a specific flight and airport combination because the flight may be associated with many airports.

So, the flight airport combination the booking is for that flight a flight at a specific airport and that we represent in the form of an association class. So, this is the solution for that we will just display few more problems for you to try out because practice is very important here, please do practice, you can consider this as assignment problems.

(Refer Slide Time: 26:18)



CRC Card

- CRC Card is used to assign responsibilities (methods) to classes.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

We will display those just little bit later before that we will discuss about handling large problems, when there are three classes, four classes which participate in a interaction we can just proceed like we did for the tic-tac-toe problem but remember tic-tac- toe is a rather trivial problem.

In a real problem there may be a dozen objects or 8, 10 objects interacting during the execution of the use case. In that case it becomes difficult to draw the sequence diagram just by reading through the problem statement because our mind cannot really capture so much of information that we can just start drawing a sequence diagram where there are 8, 10 or 20 objects participating in a use case execution.

For handling such large problems, a systematic way has been developed and that is the CRC card. CRC stands for Class Responsibility Collaboration card, we need not use it for very simple use cases where only few objects 2, 3, 4 objects participate but as long as the number of objects participating is more, we need to use the CRC card. The CRC card uses is a group activity a few developers sit together with CRC cards and develop the sequence diagram.

We are almost at the end of this session, we will stop here and in the next session we will discuss about the CRC card and then we will display some practice problems for you and with that we will complete our simple discussion on design, the basic design or the core design issues and then we will start discussing about the principles for better design and then the design patterns. We will stop here, thank you.