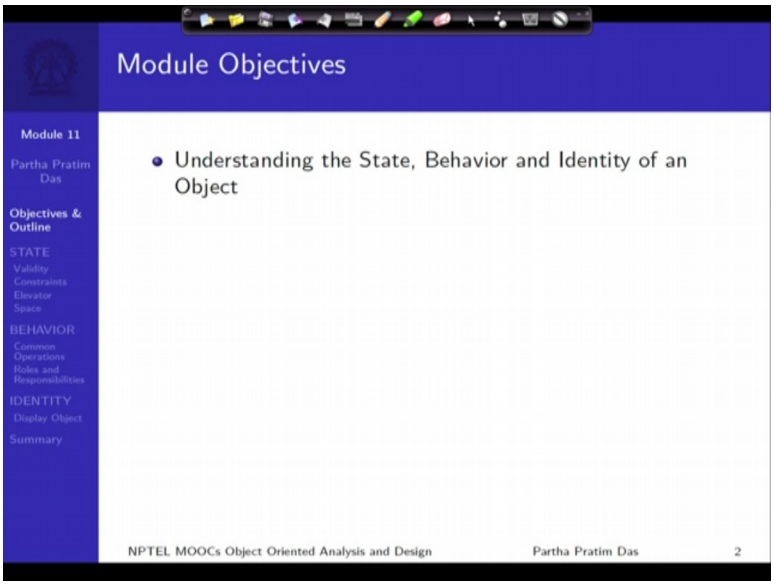
Object-Oriented Analysis and Design
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 18
Nature of an object: State, Behavior and Identity

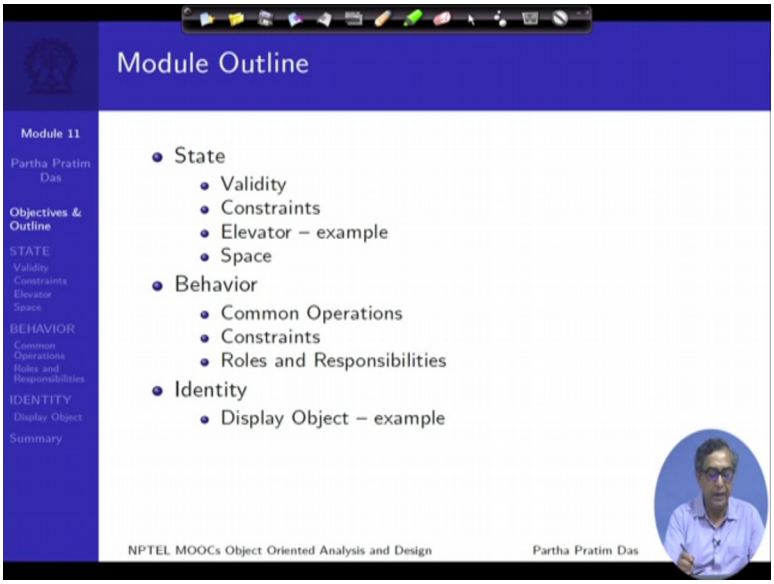
Welcome back to module 11 of object-oriented analysis and design. In the last couple of modules, we have discussed about different elements of object models. We have seen that there are 4 major and 3 minor elements which will guide, which will be the principles on which object models, should be built up. Based on those, now from this module onwards, we will actually get hands on with starting to define the details of objects, their relationships and classes and the relationships of classes and slowly get into actual exercise of identifying objects and classes in an object-oriented system.

(Refer Slide Time: 01:20)



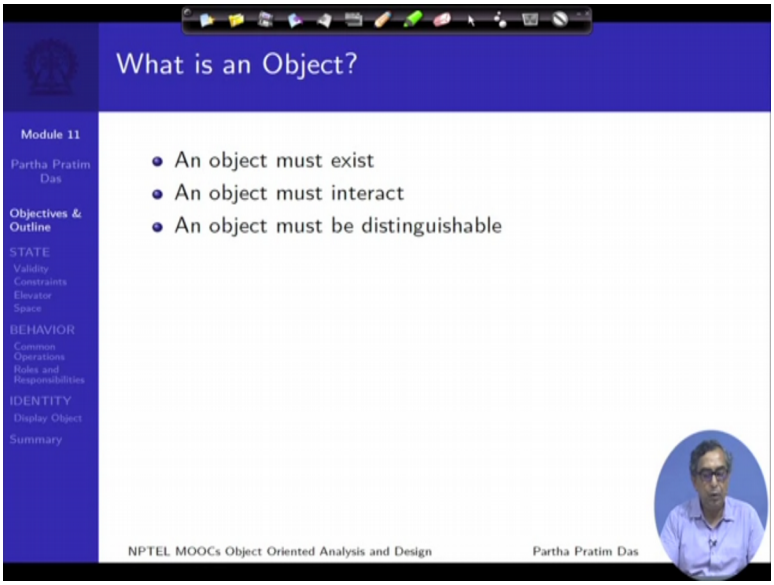
So here the in the current module we start by discussing the nature of an object, the objective of this module is to understand the three aspects known as state, behavior and identity of an object.

(Refer Slide Time: 01:41)



Outline is given here and like in every module will be carried on the left of every slide.

(Refer Slide Time: 01:50)



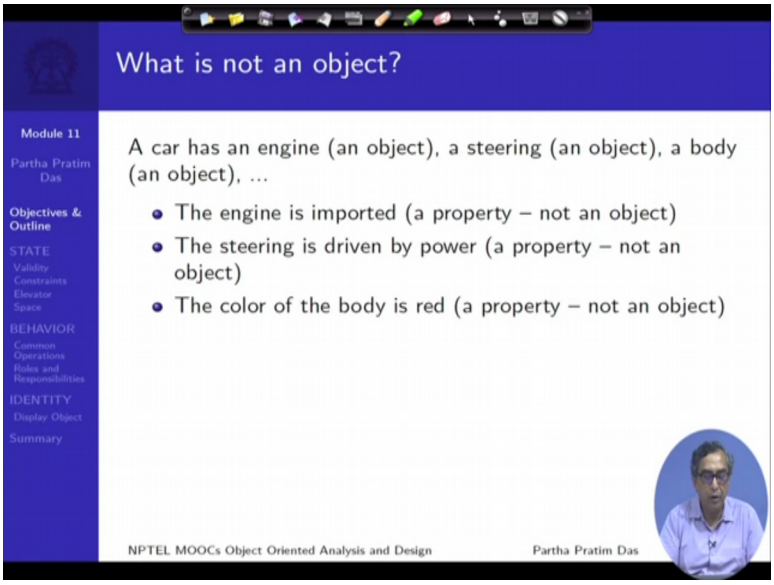
So just to quickly recap what is an object? Now we have talked about objects in terms of abstraction. Now if we finally want to come down to design, then what we say will be an object. The first condition or the first concept that we will try to enforce is that object must exist. There must be some kind of an existence for the object. So, objects could be tangible that is we can physically touch them. It may be somewhat intangible for example a process being followed in a manufacturing company could also be an object.

You certainly cannot touch the process, but it exists. It could be visible; it may be less explicitly visible and so on. different various aspects but fundamental to everything for an object is it must exist. The

second is an object that exist must interact with others that is a system is not interesting if it has objects which just ex exist by themselves and do not do anything with others. So, if a car has objects like steering, like body like engine, like wheels, like seats and so on.

These objects must interact between themselves and that is how we will get the existence of a bigger abstraction or a bigger object like a car. Third and a very important factor is object must be distinguishable that is it is not like the dust in the air or the water vapor in the cloud. But it must be like people, like chairs, tables, items, processes, practices and so on. so that one object must be distinguishable from another. so, these are the basic premise on which an object will be identified.

(Refer Slide Time: 04:22)

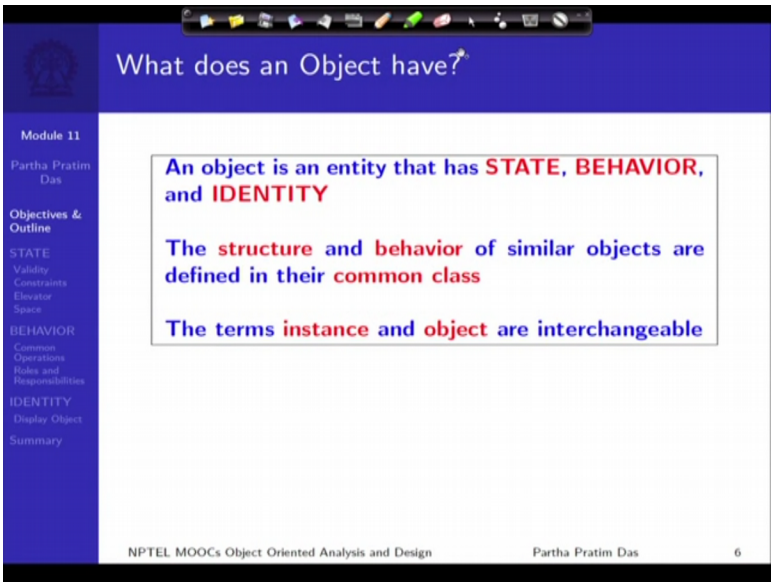


Now certainly it will be in order to also talk about if these are objects then certainly there is a lot which is not an object and what is not an object is certainly a more fuzzy somewhat vaguely defined concept and certainly something that does not satisfy one or more of the aspects that we just talked about will not be an object. For example, suppose a car has an engine, so if we just look at, it has an engine. So naturally an engine is an object.

It has a steering that is an object. It has a body that is an object and so on. now we can say that the engine of the car, I want to put this information in the design that the engine of the car is imported. Now this the fact that the engine of the car is imported is a property and being imported is not an object. Kindly understand that this is different from import which will be an object. But here we are saying that the fact that the engine is important that it is manufactured outside the country and purchase from there is a property and is not an object.

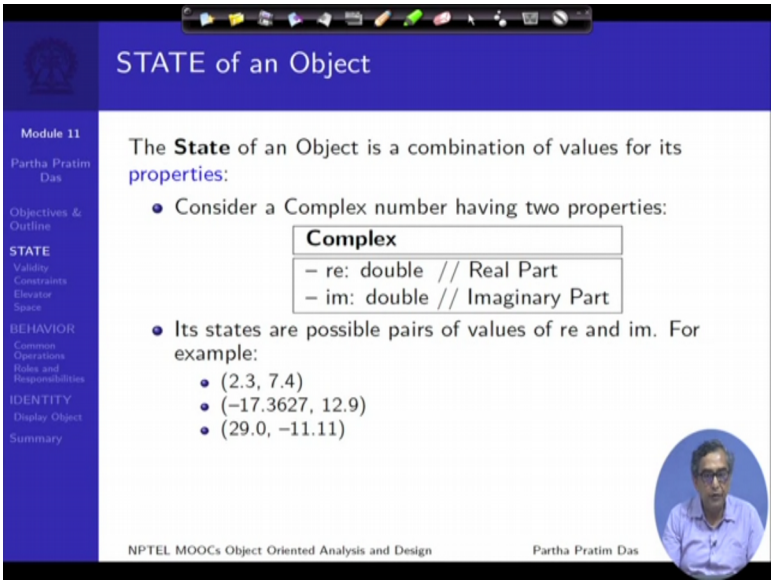
Similarly say that we have a power steering so we say the steering is driven by power. this fact will have to be somewhere put into the model, put into the design but it is again a property and not an object. The color of the body of the car is red. The fact that it is red is a property and that is not an object. So, then you could you could slowly think and expand on this and you will be able to identify, you will be able reason to yourself that what is an object and what is not an object.

(Refer Slide Time: 06:14)



Now if I say this is we now understand what is an object. Then we ask what does an object have. In object-oriented analysis and design from that perspective, an object is an entity which has 3 basic parts, a state, a behavior and an identity. The structure and behavior of similar objects will be defined in terms of their common class about which we'll talk more in the following modules and just for your reference the terms like instance of a class and object, we will be using them interchangeably one in place of the other.

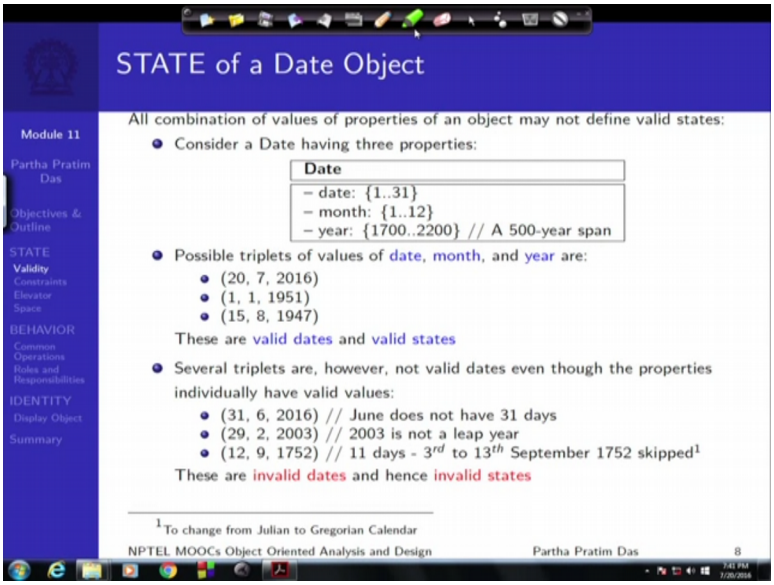
(Refer Slide Time: 07:09)



So now we will take a look into what are these state, behavior, and identity of an object are. So first talk about the state of an object. The state of an object is a combination of the values for its properties. Just note. So, it is dependent on the properties and a state is will be given by the combination of the value that the properties have. So, if I have a complex object, complex number of object, it has these 2 properties, real part and imaginary part, then a state will be since it has 2 components or 2 properties,

A state in this case will be a pair or a doublet giving the real part and the imaginary part and over different states potentially infinite, all different such pairs will define different states of a complex object. that is a simple idea, that is you consider what the properties are, considered what all different values they can take, all of these combinations give you a state.

(Refer Slide Time: 08:10)



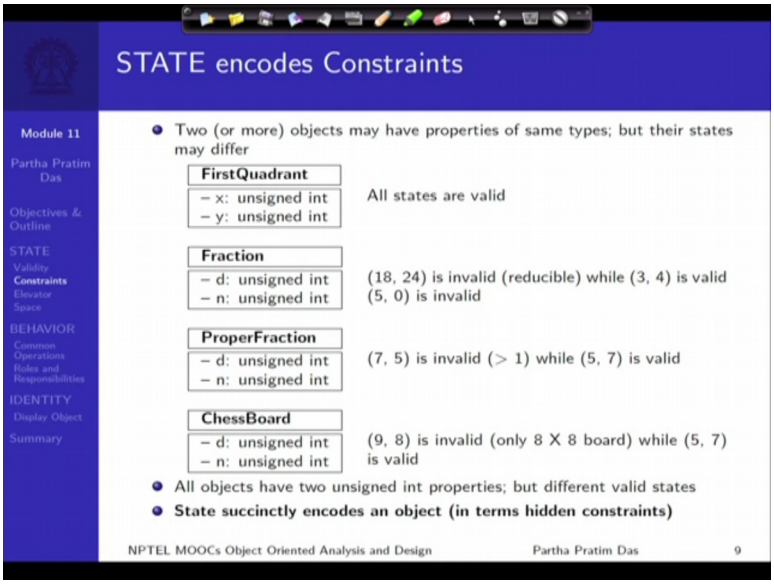
Let us look at a little bit of a different object. Lets us at it date object. So here we have I have just put a date object, simple date object which has a date as in the number of that particular day in a month. So, it can be between 1,2 this this notation means that date is a property which takes a value between 1 up to 31. This is defining a range, it has a month which is 1 to 12and just arbitrarily have taken a 500 year back, 1700 ad to 2200 ad.

So, a date object will have these 3 properties. So, we can we can look at different instances, different objects having these 3 properties like 20th July 2016, 9, 15th august 1947 and so on. these are all valid dates and therefore each one of them define a valid state but we said the state is a combination of possible values that the properties can takes for a date object, these are also possible values like 31st June 2016. We all know that is not possible, June cannot have more than 30 days.

So, this state for the date object is an invalid state. Similarly, 29th February 2003 is an invalid state because 2003 is not a leap year. Very interestingly in the calendar that we follow if we try to find the date, 12th September 1752, it will actually be an invalid date because it so happened that the calendar moved and 11 days from 3rd to 13th September in 1752 were simply wiped up out of the calendar. So, the reasons could be all different.

In some cases, a month does not have enough number of days, in some cases a leap year has to plug in, in some cases it is a one-time change in the date calendar that happened worldwide. All these lead to the fact that states may or may not be valid that is when we talk about all possible combinations, there is some underlying set of constraints that kick in could define what is a valid state and what is not a valid state.

(Refer Slide Time: 10:42)



So, let us look into little bit more this. Here I have framed and a collection of example objects for you where first you consider the similarity, the similarity is all of these objects have 2 attributes, 2 properties, all four of them and both of these have the same type. So, if I just consider the possible combinations of values that the properties can take, all of these objects is expected to have the same set of states. But now let us think of it semantically with the meaning. My first object is the first quadrant in the partition.

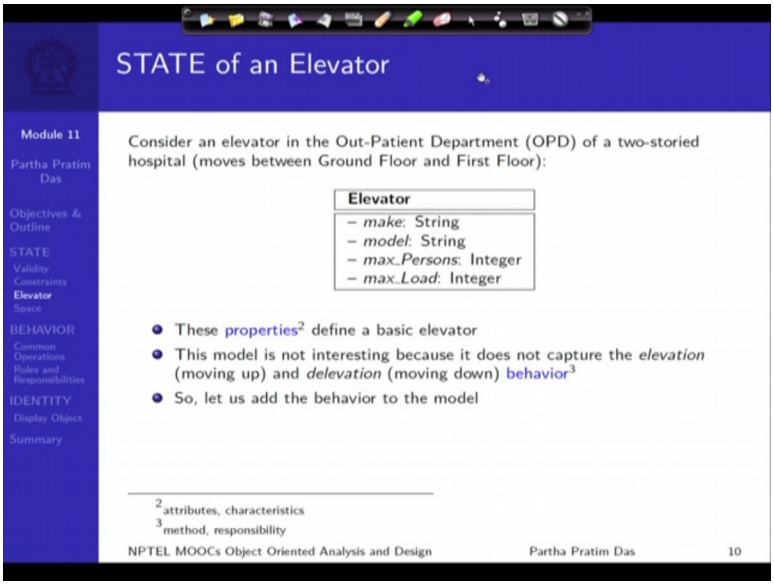
So naturally all possible quadrant values are acceptable, they are valid. So, all state are valid. Now think of a fraction. A fraction naturally is represented in an irreducible form which means that the denominator and the numerator must not have any common divisor. The gcd of these 2 should be 1. So, in terms of the properties, I can write 1824 as a pair where 18 is the denominator this is the numerator is actually invalid because they are reducible whereas when I actually divide them by the common factors it becomes a valid state, a valid fraction object.

Proper fraction similarly would be those which are less than 1. So, 75 would be an invalid proper fraction. If I think about a chess board which is 8 by 8 naturally each component, each property of the chessboard must be a number between 0 to 7/ so all other states like 9, 8 will be invalid state. So, what it shows that it is not only the properties and their types but the state is dependent on some kind of hidden constraints or some kind of encoding of objects exist in those hidden constraints which actually define what the state's an object can have.

So, this in every case whenever you doing an object, you'll have to implicitly at least understand what

the states are and what the validity of the states are any more and more constraints you actually have which I incidentally often are hidden, they are not explicit in terms of the object design not at least with the way we have seen so far. Then you will have to make sure that you do not do a design which allows invalid states to become possible values in your system if it happens, then we will certainly have different kinds of errors coming from the system.

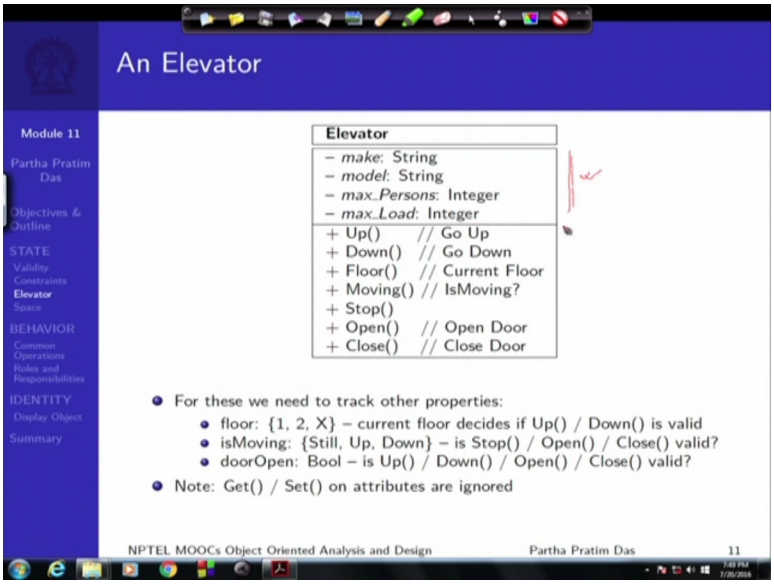
(Refer Slide Time: 14:01)



Let's take a little bit of an extended example to understand some more depth into the state. Let us consider that we are at the outpatient department of a hospital which is 2 storied. It has a ground floor and the first floor or floor number 1 and floor number 2. And there is an elevator which flies between them from ground floor, it goes up to first floor, from first floor it comes down to ground floor and there is an elevator object so this is this is the elevator object

So it talks about different properties like the make of the elevator, the model, the maximum number of persons it can carry, the maximum load it can carry, certainly it defines a basic these properties define a basic elevator but the question is do these properties actually define an elevator when he say it's an elevator, we expect not only information about which company made that or how many persons it can carry and so on but we certainly want the basic elevation and the delegation functionality to be available that it must move up and down.

(Refer Slide Time: 15:26)



So just to explore further, let us see what will be the impact on the state, if we add these operations, add these functionality to the elevator. So just look at what have done, this this remains same, this is exactly what it is, but I have added number of operations here I did a number of behavior to this which say that I have a method up which if pressed will make the elevator go up from floor 1 to floor 2, similarly a method down to go down, a floor method could be there to check which floor the elevator currently is in, it can check whether the elevator is moving

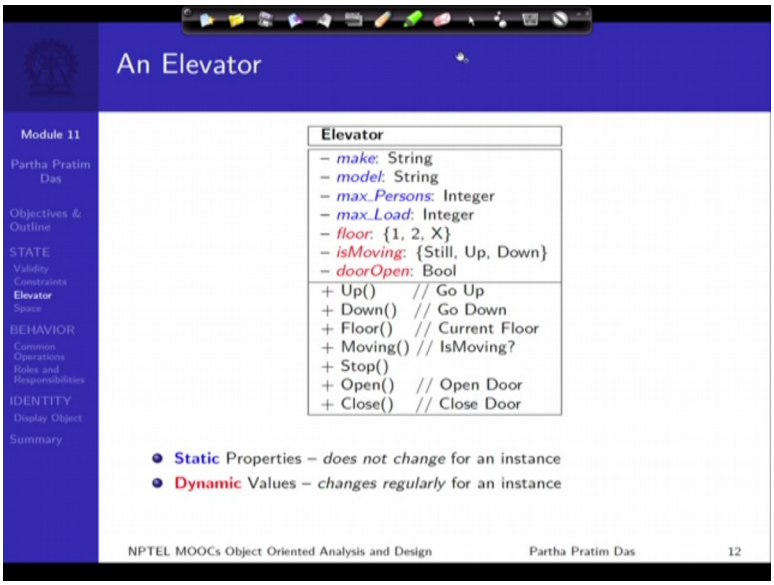
Or it is stand still I could stop the elevator, open the door, close the door and so on. so, these are all, there could be several other operations but am just focusing on some of the typical operations without which an elevator does not become an elevator. Now look into the state information this is your state information which says make, model, maximum persons and load and this is the functionality that we actually wanted in the object. So, we can easily see that the state is inadequately represented to carry the behavior. for example,

If I have to go up I need to be on the ground floor or floor number r1. Otherwise certainly I cannot go up, so to go up I need to know which floor am in. to go up certainly I would not like to go up in an elevator where door is open. So, I would need that the door has been closed before the elevator goes up or it goes down and so on. so, for these operations to be meaningful I will need an extension of the state in terms of a number of additional properties that this object should have.

For example, it should have the property say floor which could take 3 possible values 1,2 and x. 1 when it is at the ground floor, floor 1, 2 when it's in the second floor and x when it is moving that is this floor

is indeterminate. Similarly, it could be still, it could be going up, it could be going down. So is moving is a property which we will say that the door open will be a Boolean, if the door is open or is closed. So, these are these properties will help all these different operations to be actually realized.

(Refer Slide Time: 17:58)

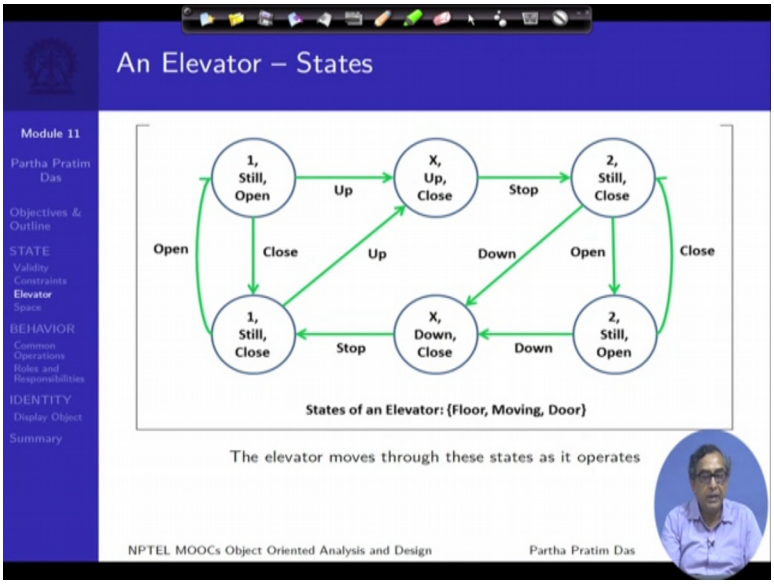


So, with this, naturally the state of the elevator now gets extended so now if you look at, these are the states or properties that we had and these are the properties that we have added, naturally with this addition the possible states of the elevator becomes a lot more. But you can see something the reason I am coloring them blue and red is you can see something about the elevator object, see the blue properties once an elevator has been installed in the hospital, then suddenly its make, model

Or the ability to carry the number of persons or the maximum load it can carry, these properties are not going to change. This will remain same as long as that elevator is in operation whereas these properties, the red properties that we have just added, the floor is moving, door open, these properties will regularly continue to change their values will continue to change as the elevator every time is gets into operation. So, I can we can say that in in in terms of defining the state, there is set of static properties which does not change or very rarely change for an instance whereas there are dynamic values for certain properties which changes regularly.

So naturally when I talk about the state of an object it is whenever such situation exists, it is more interesting to talk about this dynamic part of the state instead of talking about all possible values that can take part and define a huge state space of possibilities for the object.

(Refer Slide Time: 19:45)



So this is the, we will talk about these kind of diagrams later on when we talk about uml but this is just a sample representation to show the different states an elevator could be in. for example, initially I could be here that I am in so this representation is like this, within every circle which denotes a state of the elevator object, I right 3 items, the first one designates the floor, the second designates the moving status of the elevator and the third designates what is the status of the door.

So, if I am initially here, then it is one still open which means it is in the ground floor, floor 1, it is still it is not moving and the door is open. So, at this stage what you can do certainly at this in this state you cannot go down because it is already in the ground floor but you can go up. If you go up so this is the action that you can do on this state and then the state changes like something like this where the state becomes indeterminate, am sorry the floor becomes indeterminate

Because it is moving, your direction becomes up because you are going from the floor 1 to floor 2 and your door must have been closed. So, when this this transition of state is happening then this values of these properties must change in the appropriate way as shown. Similarly, when the elevator comes to a stop because a stop operation has been performed naturally the floor of the elevator must have become 2 because it was moving up and it becomes still and the door is still open.

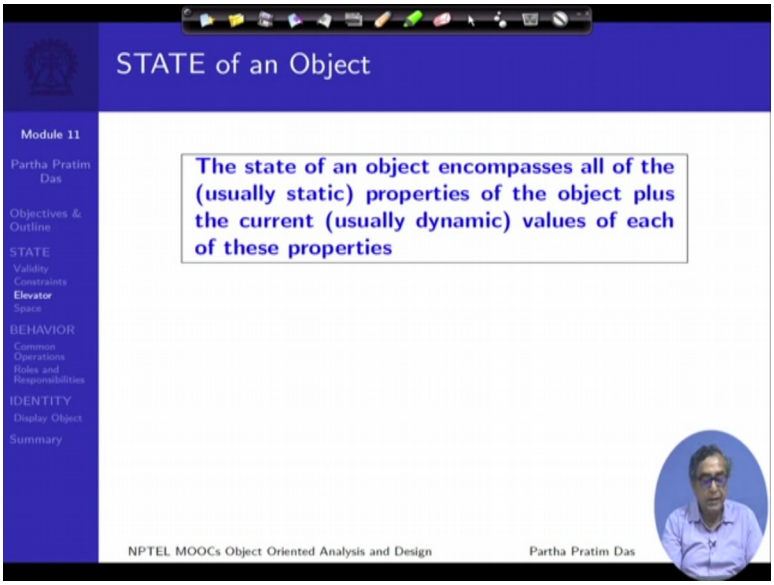
Now if you if people want to go out then certainly the can do an open operation and the state changes to 2 still open. You can switch between these 2 by opening and closing the door or you could from any of these states you could put down keep the elevator moving again but now in the downward direction and finally reach the floor 1 or the ground floor and become stationary. So, this is a we will say this is

called a state diagram or state subsequently will see this will be called a state chart in uml.

These are very powerful tool to represent how the state of an object could impact the behavior of the object in a very second manner how the constraints will apply. Here you can say find that though we are though we are having 7 properties in total the 4, make, model, maximum persons and maximum load are ignored in considering the state because they are kind of static properties and the 3 that we have which are dy dynamic values even there, we do not take into considerations all of the possibilities because all of the possibilities simply cannot happen.

For example, I cannot have a state where I have x, I have still and I have open or even close that is if my floor is interminable then my elevator cannot be in stationary mode. It must be moving. So, this will tell you that this kind of representation of states implicitly encodes the constraint that exists that an elevator must be at a well-defined floor if it is stationary if it is moving it must have an indeterminate floor, if it is moving the door must be closed and so on so forth. So that is that is the kind of a state information that an object would carry.

(Refer Slide Time: 23:48)

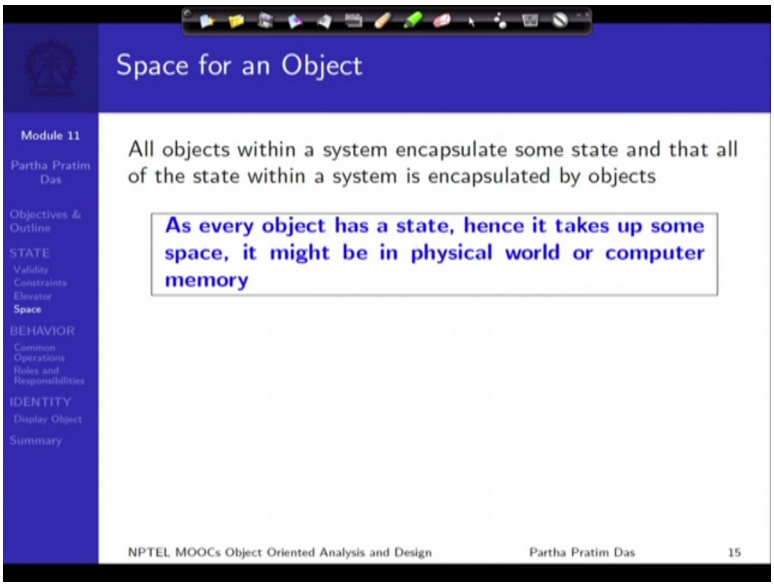


So to summarize on the state, we will say the state of an object encompasses all of the usually static properties we have seen usually static, the kind of properties we say for the elevator of the object, plus the current or usually dynamic values that is the floor is moving kind of properties and that together will define the state of an object and the state of an object may be valid or it may be invalid based on the constraints that are encoded by the state and our endeavor would be to design and define systems and their operations in a manner so that we always move from one valid state of an object to the next

and so on

And we do not ever get into an invalid state, getting into an invalid state is generating an exception in the system because if an object is in an invalid state that is if an elevator is moving with its door open then certainly the system would not know what could be the valid operations for that, what could be a consistent way of moving forward with that system.

(Refer Slide Time: 25:12)



Now along with this we should also note that the state actually satisfies the first condition that we talked about the object that an object mut must exist. That is all objects within a system encapsulate some state and all the state within the system is encapsulated by objects. This is this statement is very simple couple of words, but this is a very deep statement and please read it over and over again, try to go through examples to see that.

All that is being said that if I a system comprises a number of objects then every object will encapsulate some state in the system and on the other hand, the total state of the system if I think about the whole system as an object the total state of the system will be encapsulated by one or more objects in different parts and that is the basic notion of states in the system. So now therefore every object that I have in the system must have a state. Now if it is a state then it is to take up space.

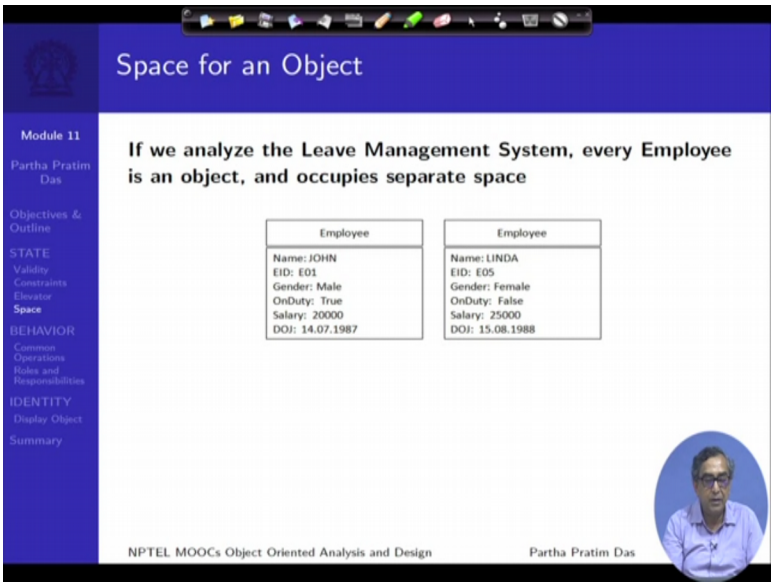
The state is memory, state is remembering, state is knowing in what condition I am. So, a state will always need some space, takes up some space. Very important that space could be into physical world or it could be in the computer memory, the elevator in the opd of the hospital physically exists. So, it

has states, it is flying people between the 2 floors, opening, closing door, moving up and down. It has a physical existence; the states have resulted in a physical existence.

So, it exists in the physical world. On the other hand, when I would like to design and implement a system for the whole hospital taking that elevator as a part, this elevator object will become a and will get an existence in the memory where in the computer memory I will have binary bit representation for all these different properties and we will do something about the methods so that it will occupy certain bits in the memory and it will have an existence.

So critical to to take back from this is states imply that objects finally have to have space, finally have to have a physical existence however abstraction we actually talked about in terms of their characterization, in terms of their design.

(Refer Slide Time: 28:07)



So, if we just look at a very simple some simple kind of example, talk about a leave management system, then every employee is an object and will occupy separate space. Now naturally physical employees of course will occupy space but their object representations as in here, I have one employee here, I have another employee, john and Linda are 2 employees, their objects will be represented in computer memory with all these values and they will take up physical space for that.

So, with this we come to the end of this first part of nature of object objects and here we have just discussed about the state. Next, we will talk about the other 2 aspects of objects.