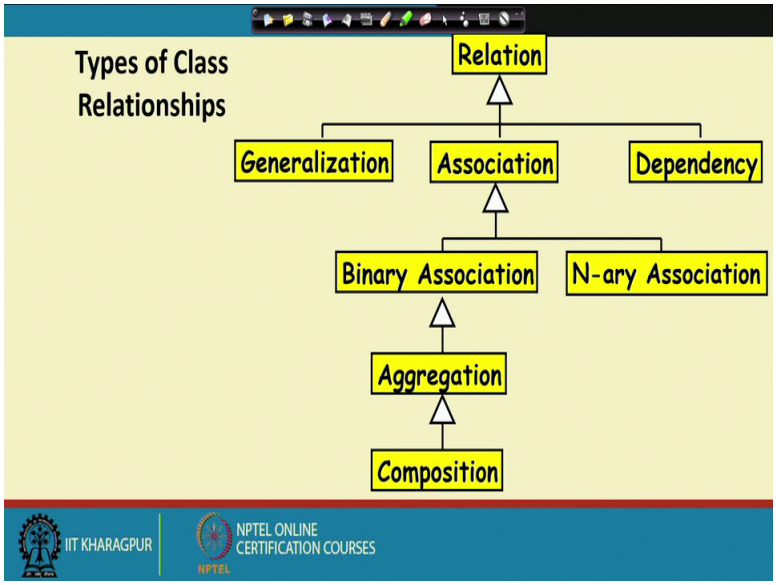
Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 36
Aggregation / Composition and Dependency Relations

Welcome to this lecture in the last lecture we were looking at the association relationship between classes. And we saw that the association relationship can imply lot of information between the types of association between two classes.

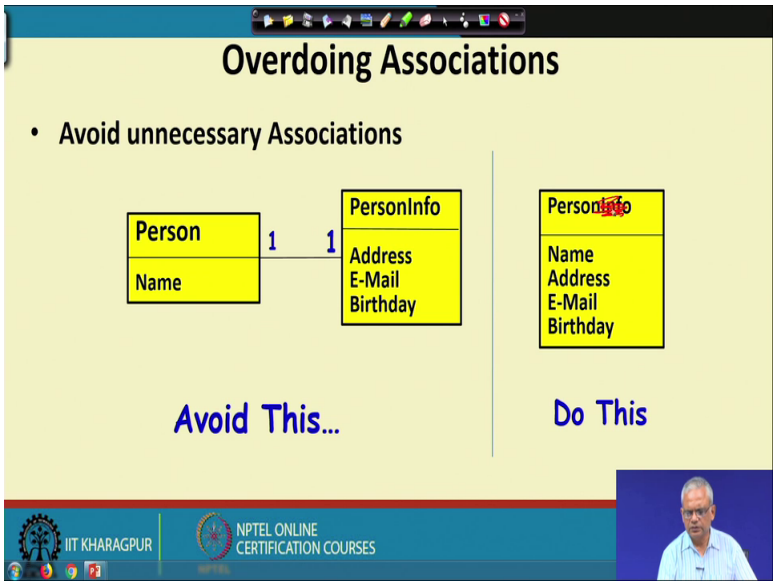
(Refer Slide Time: 00:44)



The types of relation that so far we have discussed between 2 classes; we will look at the relationship they can be generalization specialization association or dependency. And the association can be a binary association or N-ary association and a special type of binary association is aggregation and composition is a special type of aggregation.

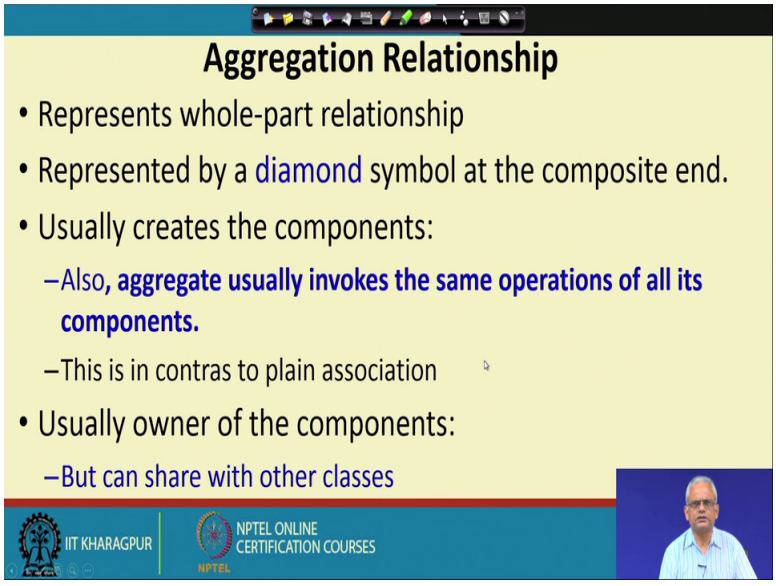
So, please observe that we are saying aggregation is a special type of association; it is a association with some extra characteristics. Soon, we will see what we mean by aggregation and why we consider it to be an association relationship with some extra characteristics. And similarly what exactly is composition relationship and why we consider the composition relationship to be an aggregation with some extra characteristics.

(Refer Slide Time: 01:42)



But before we look at the aggregation relationship in the composition; one thing we must mention here is that overdoing an association, unnecessarily complicates an design. For example, we might create a fancy diagram like this that a person is associated with person information, but actually this information are the attributes of the person. It would have been much better if you had written person information and then sorry a person and person has an; the attributes are name, address, email and birthday.

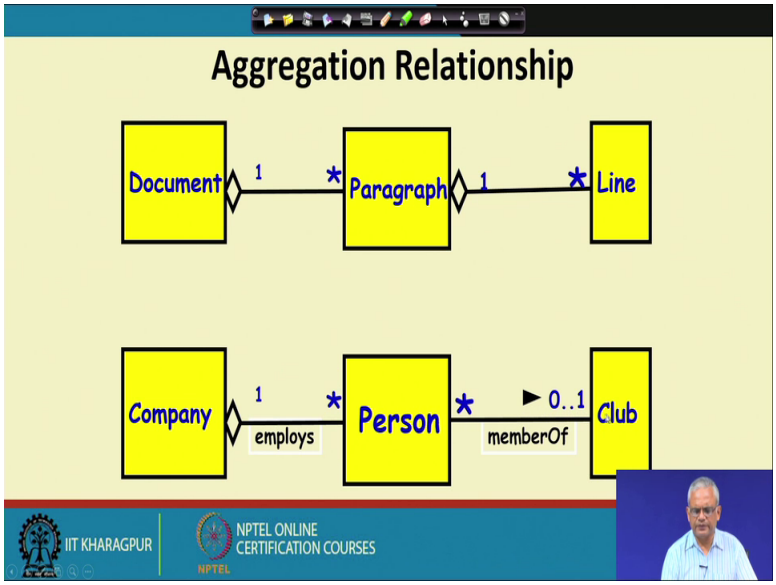
(Refer Slide Time: 02:44)



Now, let us look at the aggregation relationship between 2 classes; the aggregation relationship indicates a whole part relationship represented by a diamond symbol. And the aggregate class creates many components and it is also true that aggregate class; in books the same operations of all the components.

We will see what we mean by this and this is in contrast to plane association where a class does not invoke the; the same operation of all the classes to which it is associated and the aggregate classes typically the owner of the components.

(Refer Slide Time: 03:38)

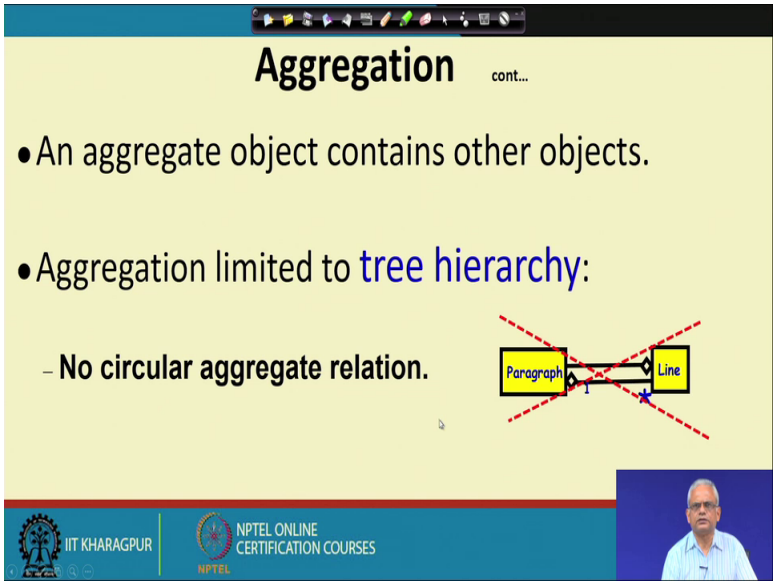


Let us look at some aggregation relationship and then we will see what we meant by saying that an aggregate class is a owner of the component classes, it creates the component classes. And it usually applies the same operation to all its component classes; this is an example of aggregate relationship the document consists of many paragraphs.

So, the document class creates the paragraphs and a paragraph is the aggregate of many lines, the paragraph creates the lines. And if we need to let us a search some word in a document; the document will apply the search operation to all the paragraphs and similarly the search operation each paragraph will apply to all the lines. So, that is what we meant by saying that the same operation is applied to all the components of an aggregate class.

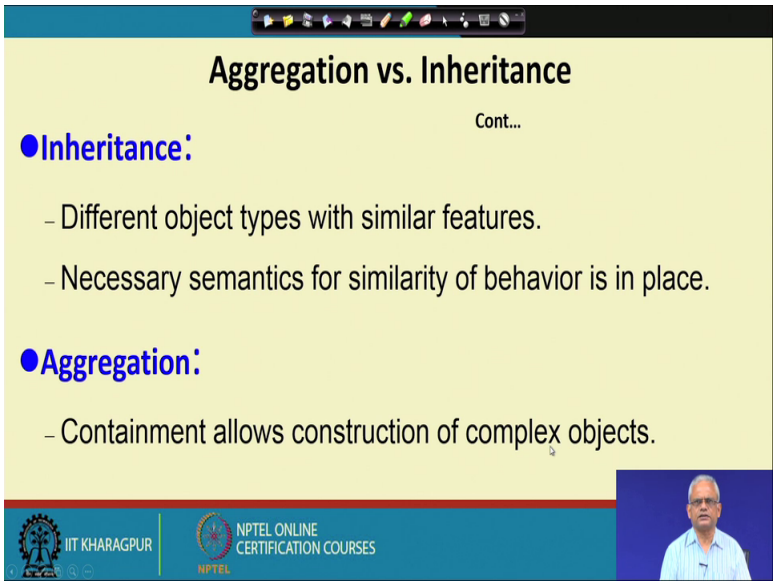
This is another example of aggregate class and this is an example of a association; let us say a company employs many persons. If a company is a aggregate of persons, then it is typically the case that the company creates the persons, applies the same operation on all the persons; for example, print salary etcetera. Whereas a person is a member of 0 to 1 club. This association relationship and a club has many members, many persons as its member. This association relationship in the club does not create the person where is here the company creates the person.

(Refer Slide Time: 05:58)



The aggregation is typically a tree hierarchy and no circular aggregate relation is possible; like a paragraph contains many lines, but at the same time a line cannot contain many paragraphs, this is a wrong aggregation relationship; it will have no implementation.

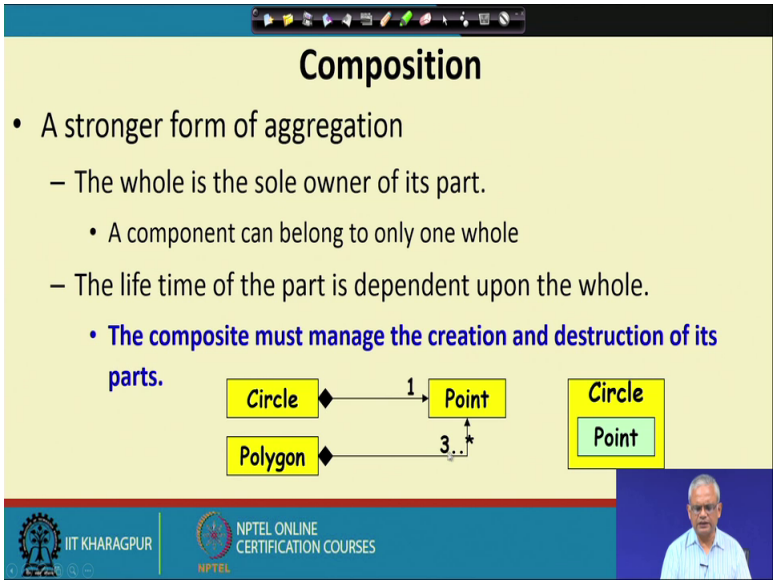
(Refer Slide Time: 06:25)



How does aggregation and inheritance compare? There are some similarities and dissimilarities; using the inheritance relationship we create many objects that have similar features. And the necessary semantics for similarity in behavior is in place because they all inherit the attributes and operations of the base class.

So, there is similarity of all the objects that are created by inheritance; on the other hand the aggregation is a containment relationship; it only helps to create complex objects, but these objects might have different characteristics.

(Refer Slide Time: 07:18)

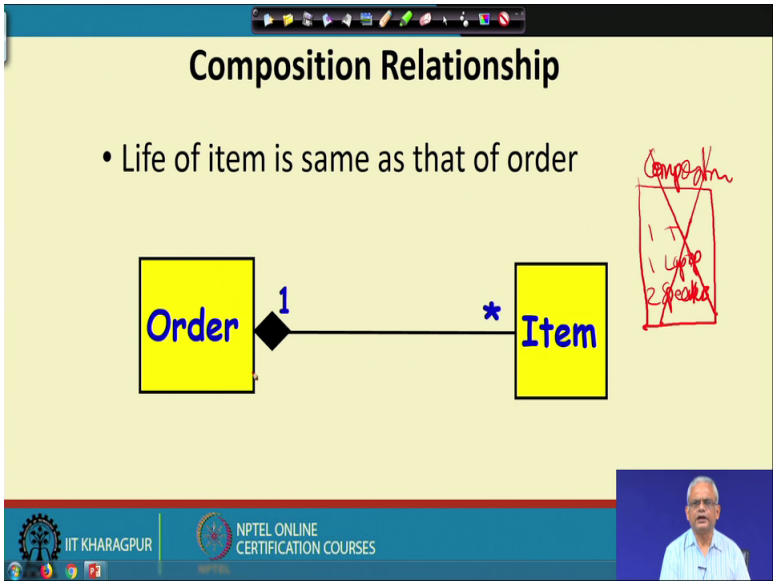


A special case of aggregation relationship is composition represented by field diamonds; it is a stronger form of aggregation and here this is the sole owner of this point; circle contains 1 point and a polygon contains 3 or more points.

So, you can represent a circle having 1 point; so, the circle class circle object is sole owner of a point. When the circle is created along with at the point object is also created in the circle is destroyed; the corresponding point object is destroyed. Similarly when a polygon is created 3 or more point objects are created is part of the polygon and when the polygon is destroyed, the point objects are also destroyed.

So, here in composition the composite class manages the creation and destruction of its parts. This is unlike the aggregation relationship where the aggregate relationship the aggregate class may create the component classes, but it does not destroy them; so, their life lines are different.

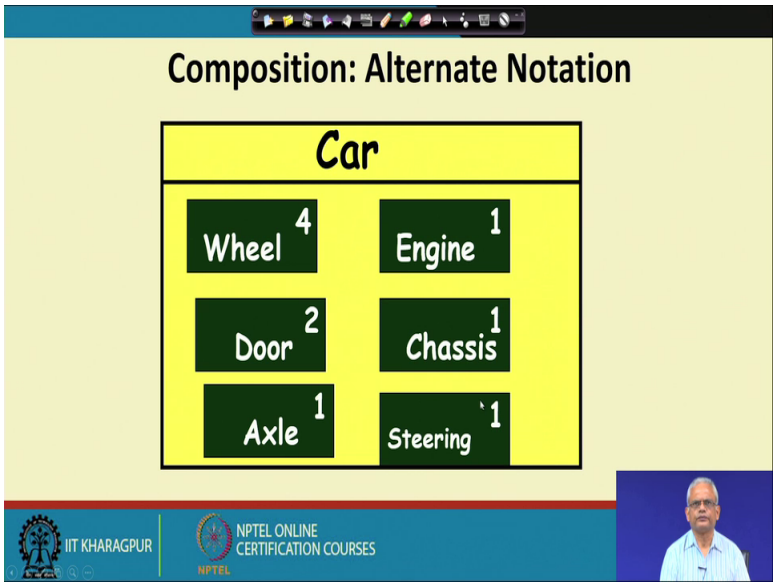
(Refer Slide Time: 09:05)



Let us just explain the difference between a composite relationship; a composition relation and aggregation relation with help of some examples. In this example, an order contains many items and when the order is created the items in it are created. And when the order is destroyed those items are destroyed, but we cannot change an item inside an order. For example, we create an order by entering the necessary data; let us say we have 1 TV, 1 laptop and 2 speakers; the order is created, but now we cannot.

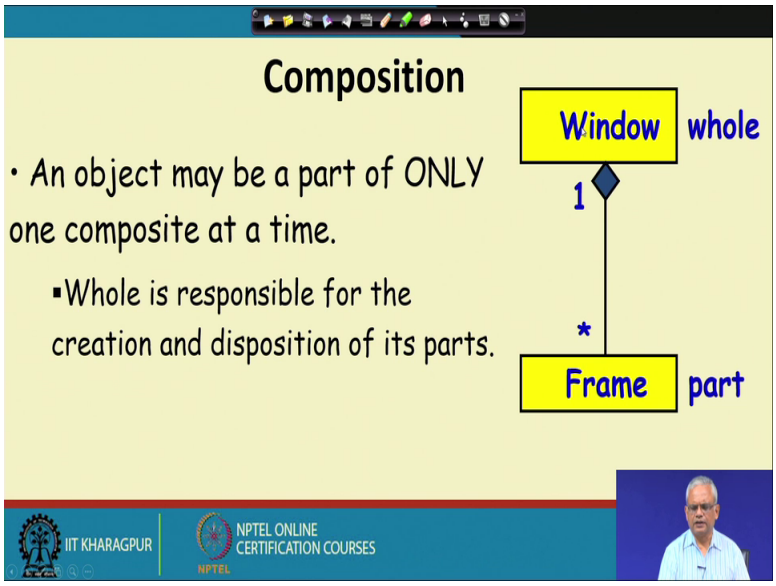
So, this are if it is a composite class; composition class it contains 1 TV object, 1 laptop object, 2 speaker object, but we cannot really changed it to 3 speaker objects; what you can do is you can delete this and you can create a new order class. But we cannot change the parts of it once it is created; the order during creation, the items are specified and the items are created along with the order. And the items are destroyed with the order there is no change possible to the order and the items in the order in between.

(Refer Slide Time: 11:04)



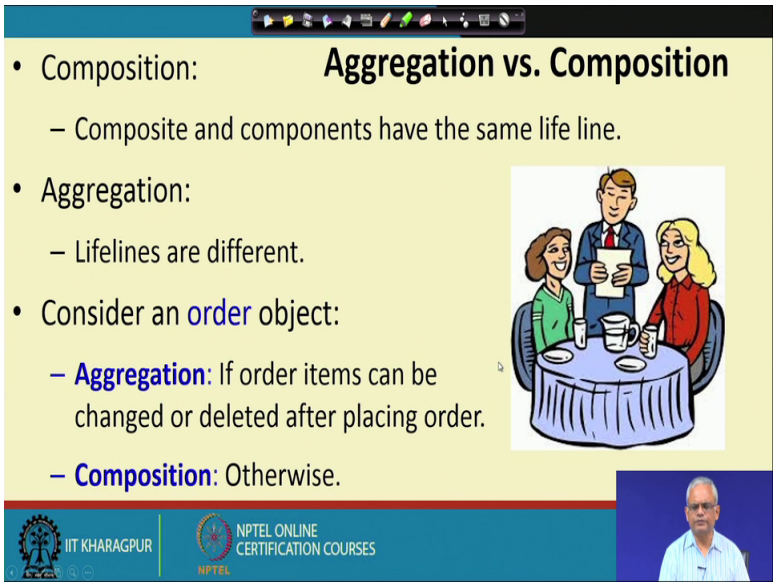
Your may also supports and alternate notation we can write the corresponding classes here inside the composite class and we can write the multiplicity of this. For example, a car contains 4 wheels, 1 engine, 2 doors, chassis, 1 axle and 1 steering wheel.

(Refer Slide Time: 11:32)



In the composition relationship and object may be part of only one composite at a time and the composite is responsible for creation and destruction of it part. For example, and window can have many frames; the window creates the frames during the creation of the window. And once the window is created no further frame alterations creation are relation is possible, but when the window is deleted; the frames also get deleted.

(Refer Slide Time: 12:12)

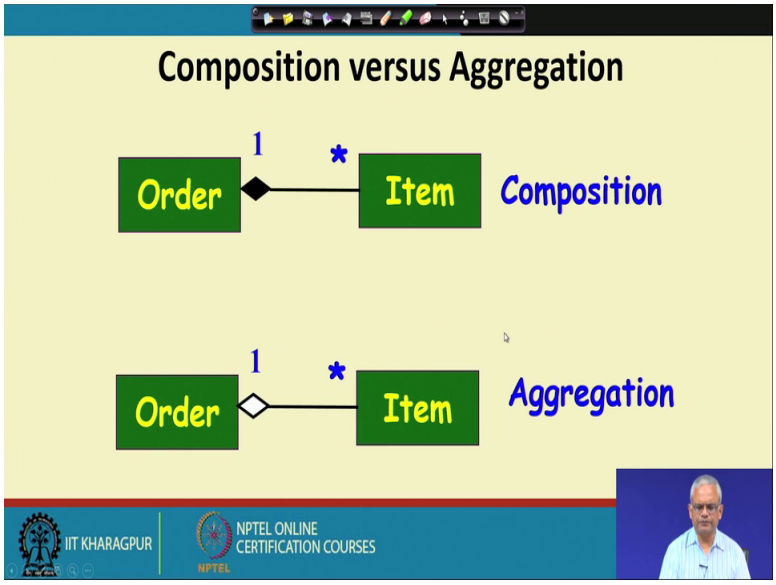


Just to explain the difference between aggregation and composition is just explained with the help of one example. Let us say you went to a restaurant and you ordered some menu

items. And if you say that see we forgot about the soup; please order the soup here and the waiter says yes I will order the soup. But if the waiter says that see your order is already entered you can do anything about it; you can only think you can do is you can delete that order and place a fresh order and that case it is the composition.

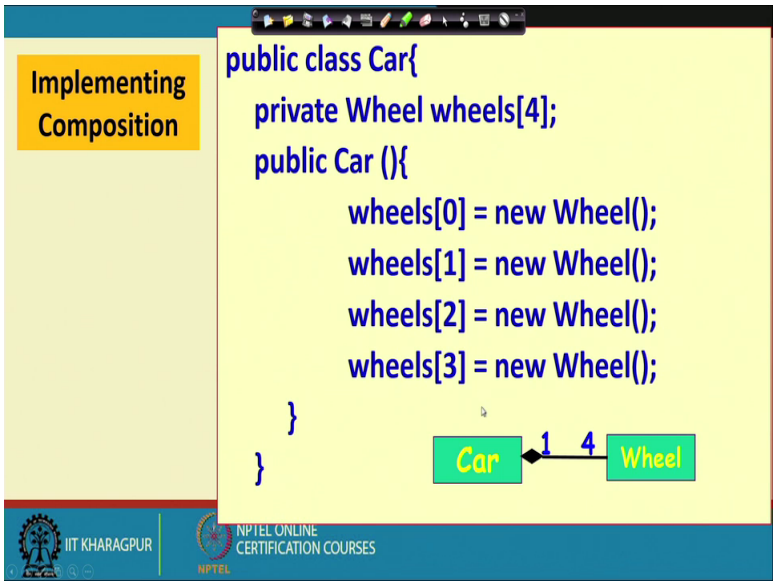
In aggregation, you can add new menu items in the order, delete some menu items and so on; whereas in composition once the order is created, the menu items are created with that and their lifetime is the same. You cannot change any items in between either you can delete the order and place a fresh order.

(Refer Slide Time: 13:32)



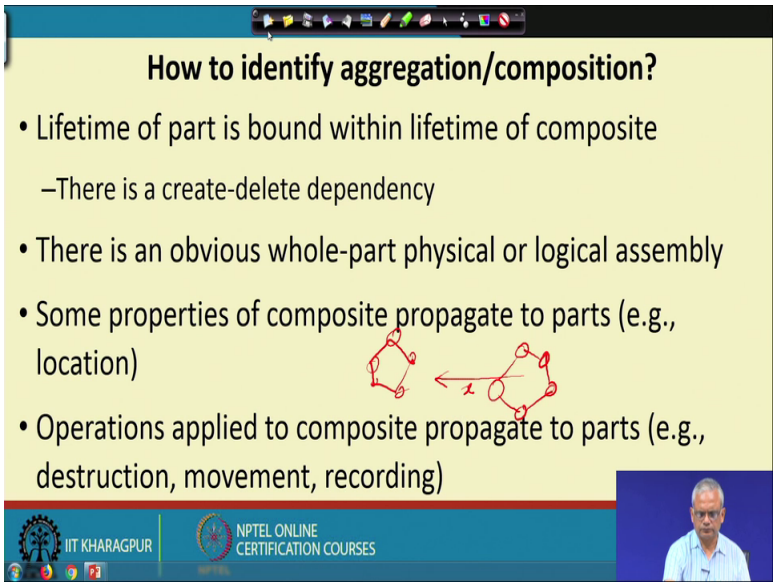
With respect to the representation; both are represented using diamond symbol, in case of composition it is filled diamond and in case of an aggregation design empty diamond.

(Refer Slide Time: 13:48)



But how is the composition relationship implemented? Let us say a car contains 4 wheels; when the car is created the 4 wheels must be created. So, can have public class car just 4 wheels and in the constructor of the car; we can a create 4 wheels. So, when the car is created the 4 wheels are created, cannot have a car with 3 wheels. So, as long as the car adjust the wheel exists and when the car is destroyed the wheels get destroyed.

(Refer Slide Time: 14:28)



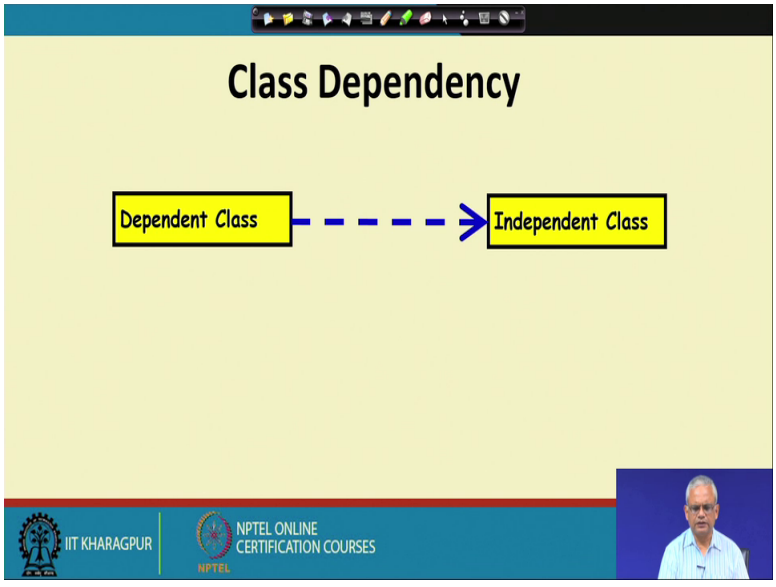
So, far we have been concentrating on the syntax; how to represent? What is the meaning of the symbol and so on. But now we come to the problem of given a description; how

do we identify aggregation composition relationship? If we find that the lifetime of the composite is the same as the parts, then that is a composition relationship.

There must be an obvious whole part relationship and some properties of the composite propagate to the parts; then it is either and aggregation are location sorry either and aggregation or composition. For example, let us say we have a car is an aggregate of its parts engine, chassis etcetera.

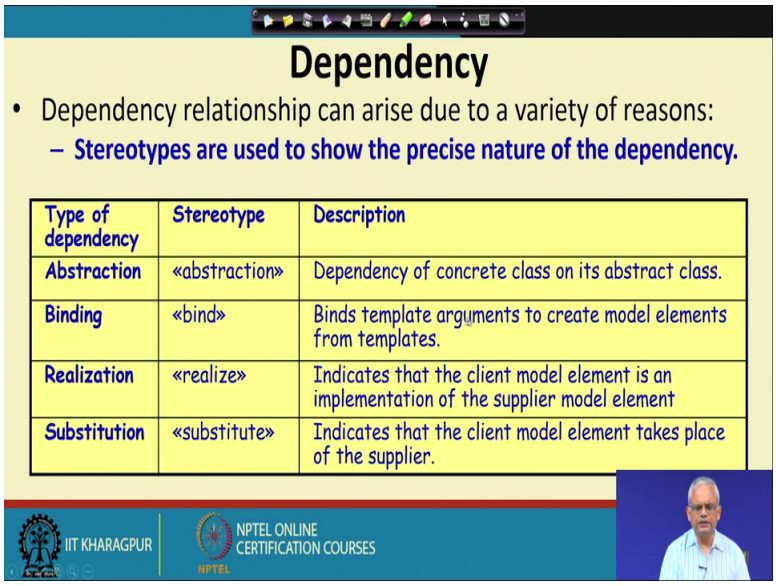
So, when the car moves the parts also move; when an operation is applied to the composite; it is propagated to the parts. For example, if you destroy the composite the parts also get destroyed, we move the composite let us say we have a polygon sorry; we have a polygon, here the polygon contains many point objects. And if we move the polygon by distance x then each of the component objects also move the distance x. So, the operation is applied on the composite object is applied to each of the component objects.

(Refer Slide Time: 16:50)



Now, let us see the last type of dependency the fourth type of dependency sorry fourth type of relation which is the dependency relation; represented using a dotted arrow from the dependent class to the independent class indicating that the dependent class is dependent on the independent class.

(Refer Slide Time: 17:12)



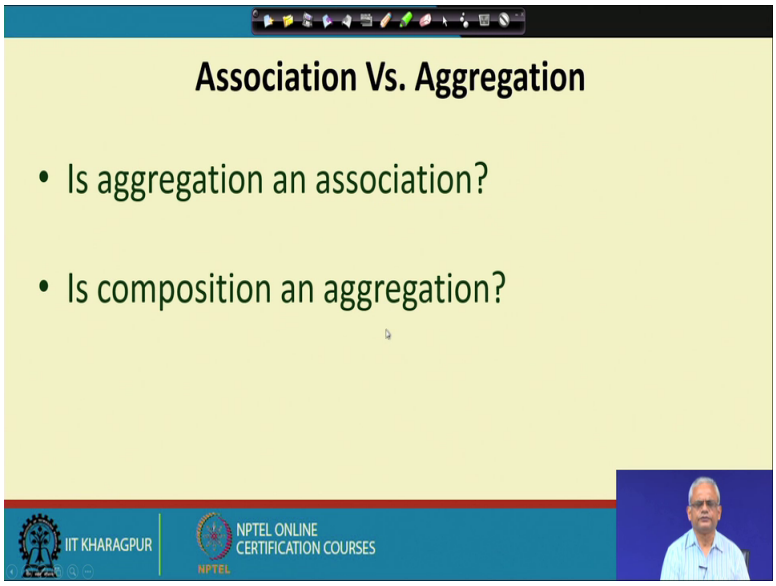
There are many reasons why dependency can arise between classes; one is due to abstraction. We have a concrete class which is dependent on the abstract class which means that if anything changes on the abstract class, the concrete classes also get affected the changes. Whereas abstract class is a independent class if you change the concrete class the abstract class is not affected.

Similarly, is the bind binding if you have the template arguments to create the model element from template. Let us say we have a stack class and we can have a integer stack or we can have a floating point stack or we can have a stack of strings and so on. We can bind the specific type inter, floater, string etcetera to the stack class and there is a dependency and the template arguments.

Similarly the interface realization if there is a interface class and the client classes are implemented implementation of the interface then if the interface changes then the client classes need to change. So, there is a dependency of the client class and the interface class; similarly the substitution a class can be used an object of a class can be used in place of another object.

So, there is a dependency there are many reasons why the dependency can arise; maybe there about a dozen reasons why dependency can arise and you have just listed few of them here.

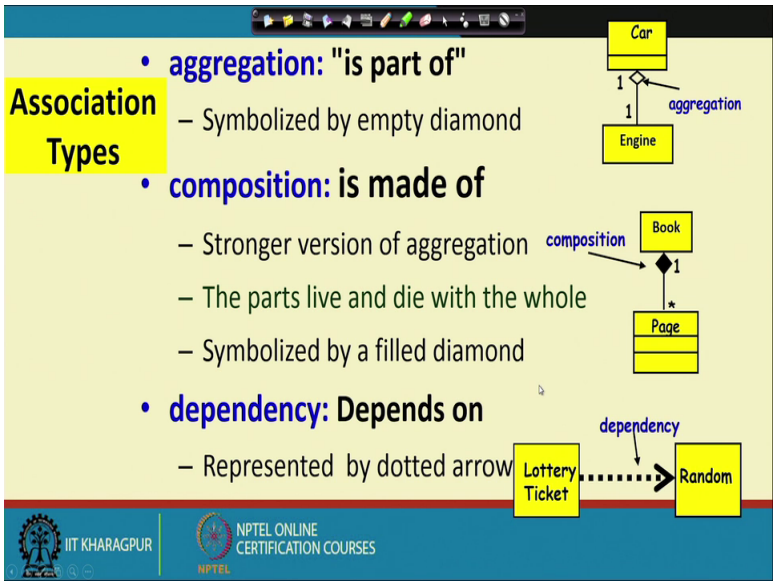
(Refer Slide Time: 19:24)



But each association is aggregation relationship in association; yes aggregation relationship is an association because the aggregate class is associated with its component class. It can invoke their methods, but what about a composite class is it an aggregation relationship?

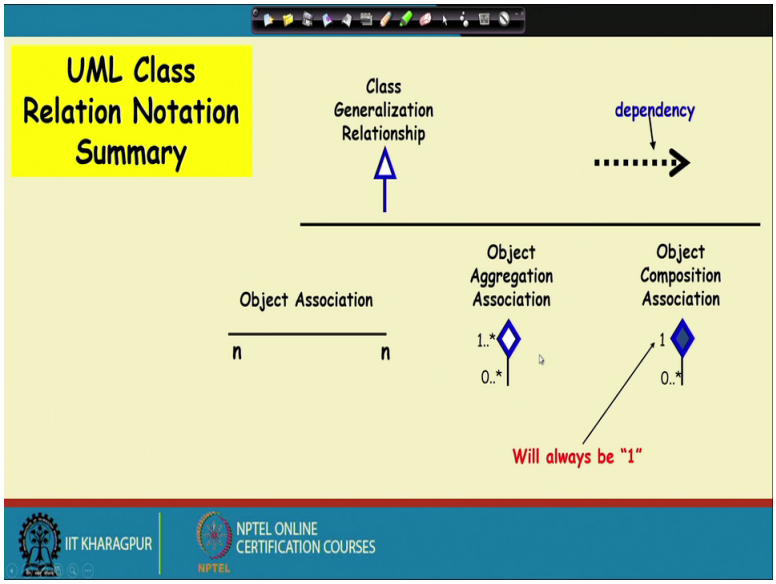
Yes, a composition is an aggregation relationship, but the only difference is that the composite class and aggregates the component classes their lifetime is the same; the composite class creates its components during its creation and also destroys the parts on its destruction. Whereas an aggregate class some of the parts can be created later on and they become part of the aggregate class or some of the items parts may be deleted.

(Refer Slide Time: 20:40)



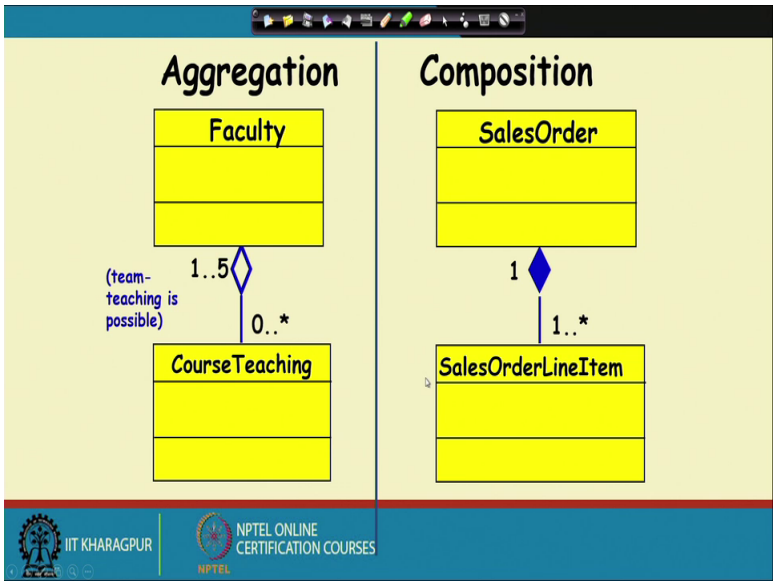
The aggregation relationship is represented by an empty diamond where is the composition by a filled diamond and the dependency by a dotted arrow.

(Refer Slide Time: 20:58)



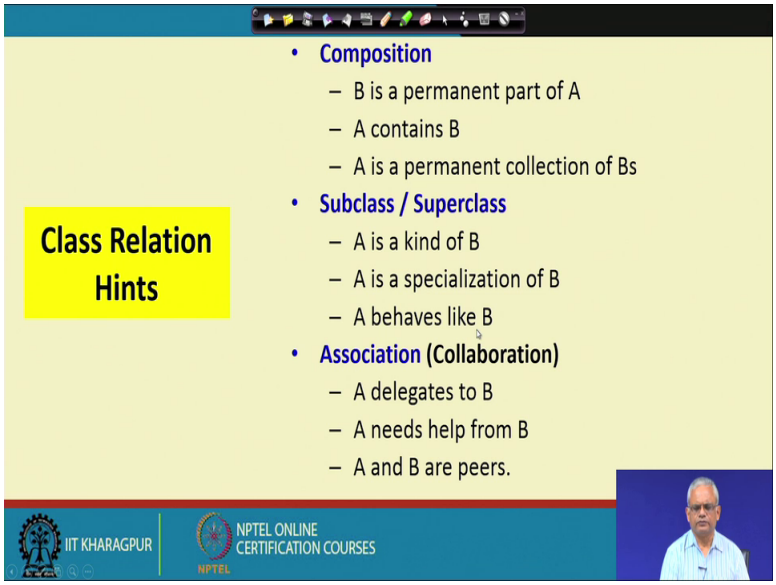
So, this is the summary of the notations we have so far seen the generalization relationship dependency association and this; the multiplicity we write down the association and the name of the association and the association line aggregation relationship composition relationship and here always we have 1 where is here it can be 1 to star.

(Refer Slide Time: 21:29)



Just to give an example a faculty can teach many courses, but a course may be dot by 1 to 5. So, that is a group teaching possible, but here it is always one the sales order contains many sales order items once the sales order created along with the sales order line items are created.

(Refer Slide Time: 22:01)

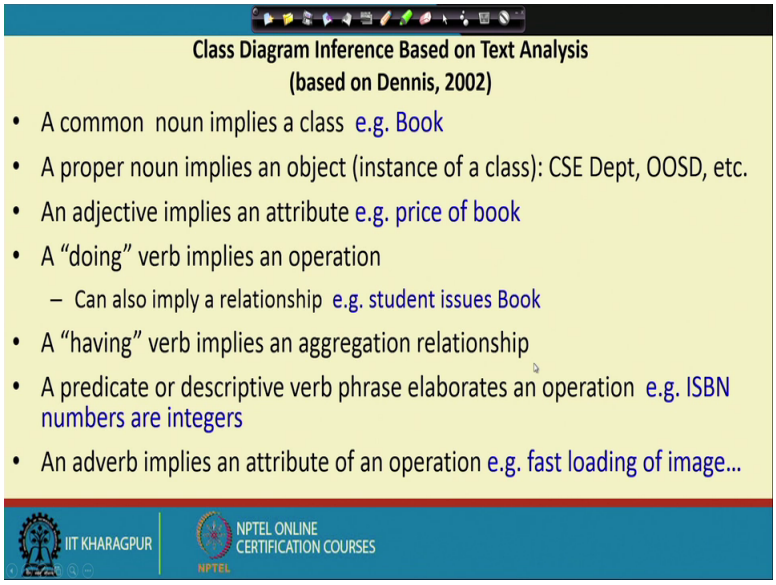


How do you identify the class relations given a description, how do you identify which is composition which is the subclass; superclass generalization, specialization relationship and which is an association relationship? If from the test desk description you can find B

is a permanent part of A. Or A contains B or we have A is a permanent collection of B; then we say that; that is a composition relationship. If A is a kind of B or A is a specialization of B or A behaves like B; we say that there is a subclass superclass relationship or generalization specialization relationship B is the base class and A is the derived class.

Similarly, if A delegate some responsibility to B; A needs help from B, A and B are peers if the text contains this kind of terms then we know that A and B are association classes sorry they have association relationship between them.

(Refer Slide Time: 23:13)



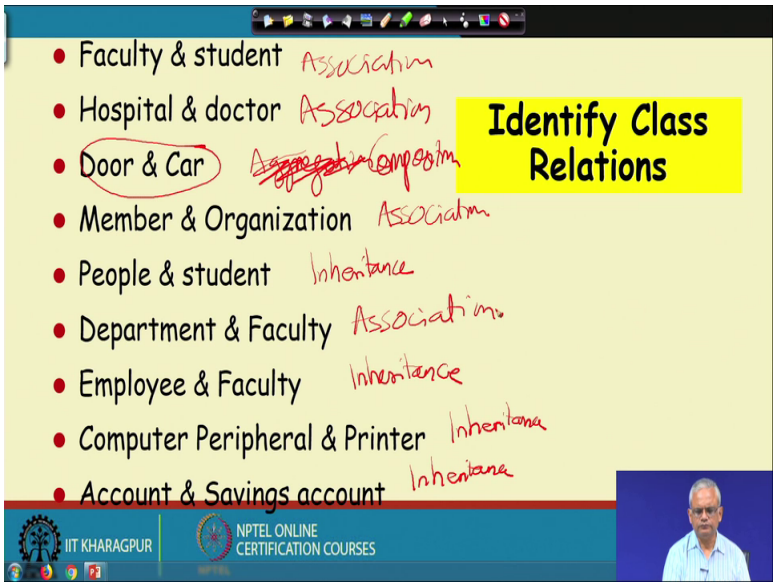
But how do we identify the classes from a text description? Because after all during the design process we need to identify the classes that is an important statement design when we look at the design methodology; we will see that identifying the classes from text description is an important skill.

Here is a hint how to go about identifying the classes based on the text analysis; grammar analysis you can do and identify the classes. A common noun implies a class for example, book is a common noun, a proper noun implies an object for example, CSE department, Object Oriented Software Design and so on. And adjective implies an attribute for example, price of book; a doing verb implies an operation for example, student issues a book.

So, issues is a operation; having implies an aggregation relationship, a descriptive verb phrase indicates an operation for example, ISBN are integer; it elaborates an operation. The descriptive verb elaborates an operation for example, ISBN operation numbers are integers, an adverb implies an attributer operation for example, fast loading an image.

So, it says that; what are the characteristics of loading it identifies an adverb fast loading identifies some characteristics of the operation and helps in implementing the operation.

(Refer Slide Time: 25:16)



Now, let us practice with few simple examples how to identify the class and relation? What are the classes here? The first one the classes are faculty and student, but what is the relation? The faculty teachers many a faculty teachers by teachers; many students a student dot by many faculties.

Similarly what is the relation between hospital and a doctor? A hospital employees many doctors; a doctor is employed by a hospital. A door and a car; a door is attached to one car a car has 4 doors, a member and an organization a member belongs to one organization; organization has many members people and student.

So, this is a inheritance relationship; these were association relations; this is a aggregation relationship. A car has 4 doors this is aggregation; these are all association relations, people and student this is a inheritance relation a student is a special type of people, department and faculty this is the association relation. Employee and faculty this

is a inheritance relationship; a faculty is a special type of employee. Computer peripheral and printer; this is also an inheritance relationship because a printer is a special type of computer peripheral. Account and savings account this is also an inheritance relationship because a savings account is a special type of account; we might have current account, fixed deposit account and so, on.

So, given 2 classes we should be able to identify what is the relationship that exists between them; whether it is an association, aggregation, composition ok. This is actually door and car is a composition relationship because a door is permanently attached to a car. So, we can write that as a composition relationship whereas hospital and doctor is an association relationship because doctor may leave a hospital; new doctors may join. Similarly, member and organization is an association relationship, department and faculty is a association relationship. We need more practice to identify the relationship between classes. We are almost at the end of this lecture.

We will stop here and continue in the next class. And we will give you some more exercises to practice because after all; designing is a skill which is a learnt by lot of practice. We will give some assignment. And, also in the next lecture we will give you some quizzes so that you pick up the skill and identifying the classes and their relationships.

Thank you.