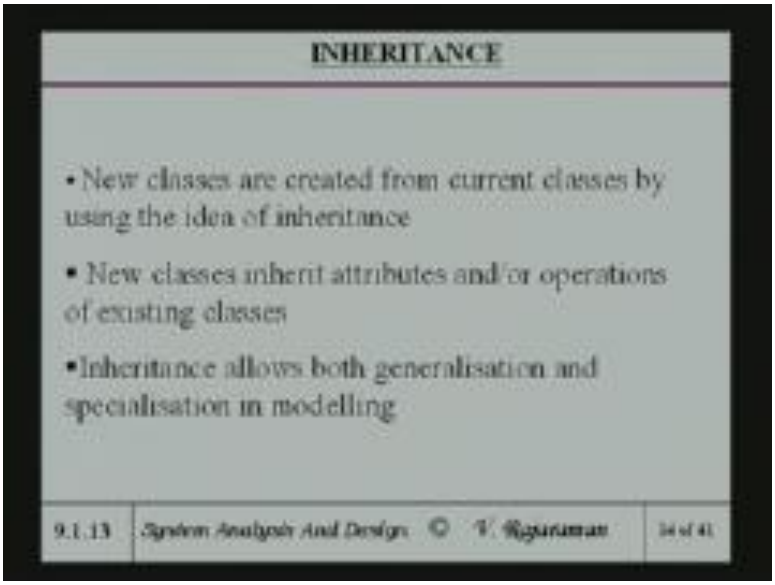
Systems Analysis and Design
Prof. V. Rajaraman
Department of Super Computer Education & Research
Indian Institute of Science, Bangalore

Lecture – 27

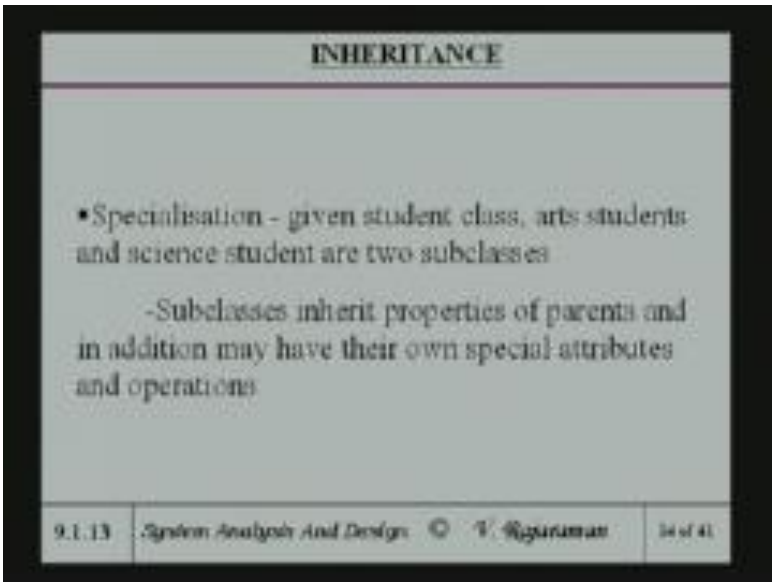
Last time, we started discussing the idea of inheritance, which is an important part of the entire object oriented modeling idea. Inheritance is something where new classes are created from current classes.

(Refer Slide Time: 01:39)



And the new classes inherit the attributes and or operations of existing classes. In other words, it is a kind of a, you might say specialization. Inheritance allows both generalization and specialization in sense that, you can I will show example. So, that I mean by generalization and what I mean by specialization.

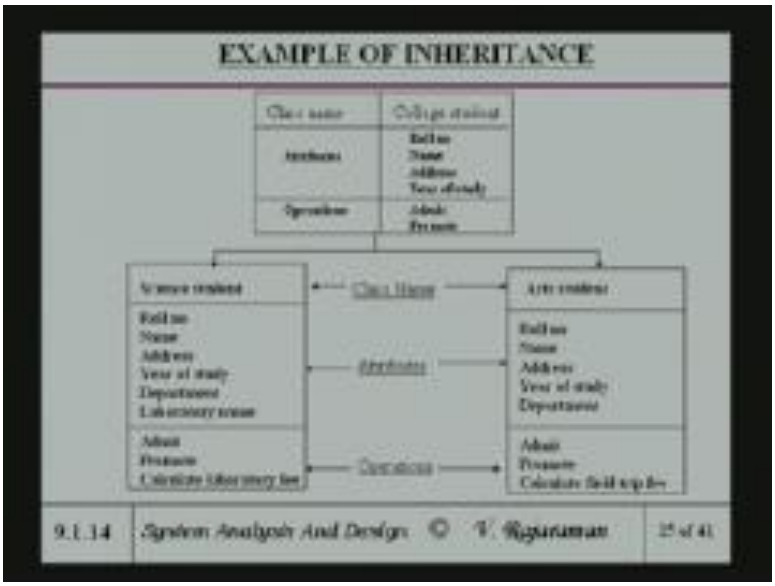
(Refer Slide Time: 02:08)



Specializations see given a college, in which there are both science students and arts students. Then, the science students or arts students are two sub classes of the class students. And sub class, the two sub classes inherit properties of the parents. Parents are being student, addition to that they may have their own special attributes and operations. In other words, a science student as well as an art student is a student.

But, the operations and may be even attributes, which are important. As far as the art student is concerned, may be different from that of a science student. But there will be some common attributes. So, the specialization really is based on the fact that the sub classes inherit the properties from main class. Some other properties and have their own special properties.

(Refer Slide Time: 03:16)



Like for instance, the class is a college student. And the attributes of a class are roll number, name, address, year of study. That is common to all students. And operations, which are common for all students is admit a student or to promote a student. And then, you can have sub classes which will need it. Some of the attributes like science student roll number, name, address, year of study.

They will again be repeated because, that is common for all students. Besides that, I am putting a department name or department code and laboratory name. Because, he may belong to a particular department and to a particular laboratory in the department because, departments are sometimes classify into labs of different types. And apart from admit and promote, it may have a new operation of calculating fee.

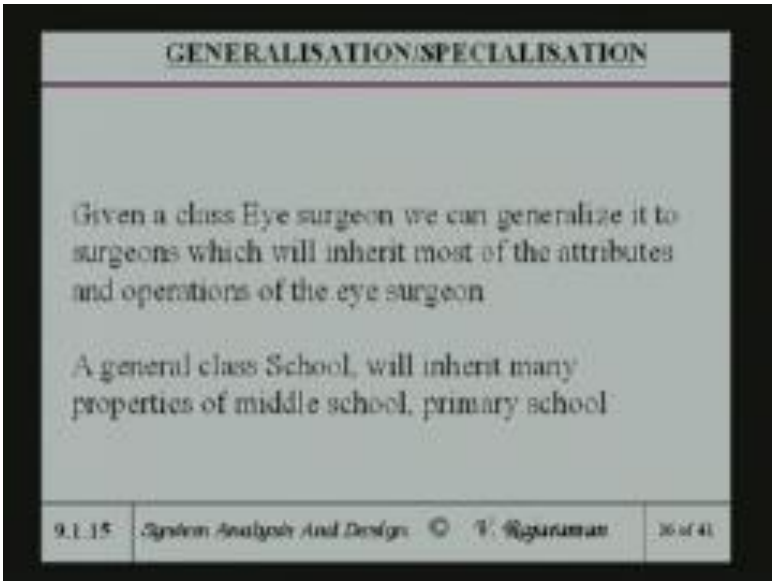
Because, the fee structure for science students may be different from a fee structure for the arts student. So, when you want to calculate the fees the operations as I pointed out in last lecture, are effectively methods or programs which are encapsulated into the object. So, the collection of fees may be based on certain rules. I have given a very simple example of calculating laboratory fees. Normally the science student has lab fee. And art student does not have lab fee.

So, there is a calculation of lab fee. But in general, the different two different student classes namely the art student and science student may have different fees structures. And implementation of fee structure may really depend on the rules, which are relevant

to them. So, calculate fee. In fact, I put laboratory fees in calculate field fee. But, if you just put calculate fee then also, it is alright.

Because, we will come back to this question, I mean it is a same operation which is name is same. But, depending on the class the actual program which is implemented would be different. So, if I just give an order calculate fee to this, then it will based on the specialized rules for the inherited class. It will actually calculate fee for that calculate or do the work which is assigned for that.

(Refer Slide Time: 06:10)



Similar generalization is given a class eye surgeon. We can generalize it to surgeons, because eye surgeon is only a part of a surgeon. And which will inherit most of the attributes and operations of the eye surgeon. You can look at it kind of backwards from a class general, general class like a surgeon. I can go to an eye surgeon, which inherits the properties of the general surgeon. And also, it got some special properties or special ((Refer Time: 06:41)) operations.

And if it is an orthopedic surgeon, that person also is a surgeon. It also, he will also use an operation theatre. He will also do a operations and so on. But, the fee charges may be different. And many others, the operation may be different. And the attributes also may be different. So, instead of having separate kind of all types of surgeons, you can look at the surgeon class. And so, the generalization is that you go from small to big. And from big, you get to several small sub classes.

So, kind of get to the class surgeon and then, make it a specialized set depending upon different types of surgeries. Similarly, a general class school will inherit many properties of middle school, primary school and all that and vice versa. A school is more general than the middle school, primary school and so on. In other words, the reason why we talk like this, one might say you know why not we always start with the school, and go to middle school, primary school and so on.

Because we know that specialized case of classes, we have schools. But, you know let us look at an example. May be if there is a school, which is currently a primary school. And they would not upgrade to become a secondary school and later on high school. At the beginning, they may have a class which is relevant only to primary school students. And the class has certain attributes and operations.

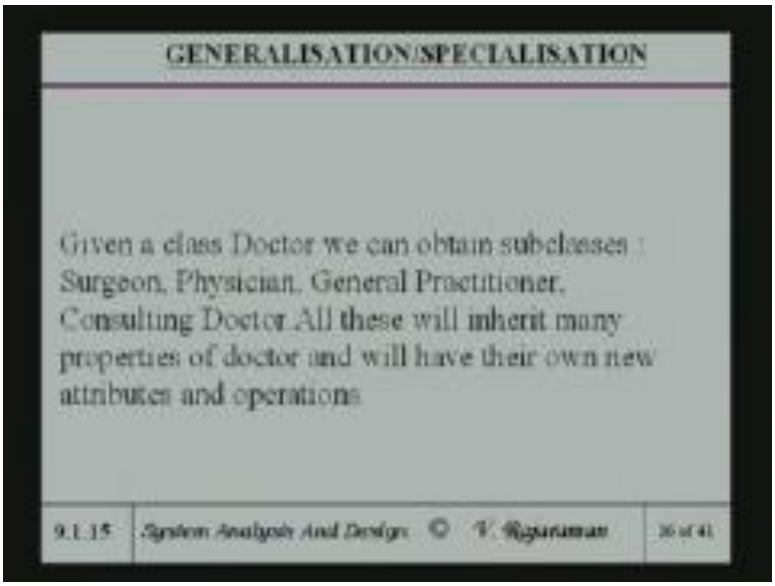
When the school expands to have a secondary classes also then, you would kind of a secondary school students will inherit some other properties of the primary school students. Now, you can in fact say, that you have general class students. And now, you realize that the school may become high school tomorrow. And also become a PUC type institution with 2 years, of almost college study after the school.

So, in order to accommodate changes, which may come into future? It is always a good idea to look at a general situation. And make primary student even those school may be only primary school, make it a sub class of this class of student. And then, later on we add new things. You can painlessly add new sub classes of that super class, namely the student class. That is the whole idea of inheritance which in other words you can always generalize. And from there, we inherit properties and specialize.

And so, you can a look a little bit ahead of what may happen in the future? And look at some generalized classes, which may not currently have too many sub classes. But, as time progresses more sub classes may occur which inherit many of the properties of general class. This is the general idea. Of the fact, that the object orientation primarily promotes, what I would call and change? It accommodates change.

And so, while you actually do a design you have to keep in mind that the whole idea in software is primarily to accommodate change. And while you do that, it always good to generalize and then, from there specialize. That is a whole idea, if inheritance allows this kind of generalization and specialization.

(Refer Slide Time: 10:48)

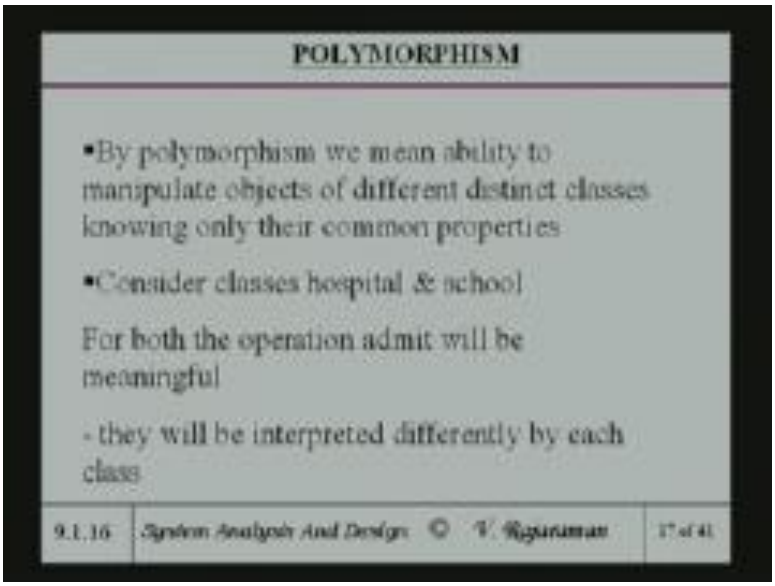


So, given a class Doctor, you can obtain sub classes. In other words you can go beyond surgeon. Because, particular hospital normally you would have normal surgeons. But, physicians it will have specialist physicians of in different areas and so on. And it may have a general physician or in general doctors. If you look at a class Doctor, there are specialist specialized doctors who are consulting doctors and general practitioners, who effectively or you might say your family doctor, who is not specialized in anything.

If you have a small illness, you go to a local your family doctor. And if it is a very complex illness, he will suggest that you go to a specialist doctor. Or is a surgical case, refer you to a surgeon and so on. In other words, you can always again look at doctor is a class. And doctors have certain common attributes regarding, whether the surgeons, general physicians, specialists and so on.

So, all these will inherit properties of doctor. And you will have their own new attributes and operations. So, this essentially what the same idea like in school. Now, polymorphism is somewhat slightly different idea.

(Refer Slide Time: 12:40)



By polymorphism, we mean the ability to manipulate objects of different distinct classes, knowing only their common properties. In other words, there are some common properties which are there for both and those. But, then the operations or methods which are encapsulated within the object, are really based on the attributes of that particular object class. So, if you look at object classes of hospital and school, both have some common operations like.

Both the word admit will have a meaning for both the school and the hospital. In a school context, it will be admit a student. And there will be a certain kind of rules we follow for admission. When you say admit in the hospital context, then it essentially would mean admit a patient to the hospital, as an inpatient. Both are meaningful, but they are interpreted entirely differently. Depending on the class, in which they are put.

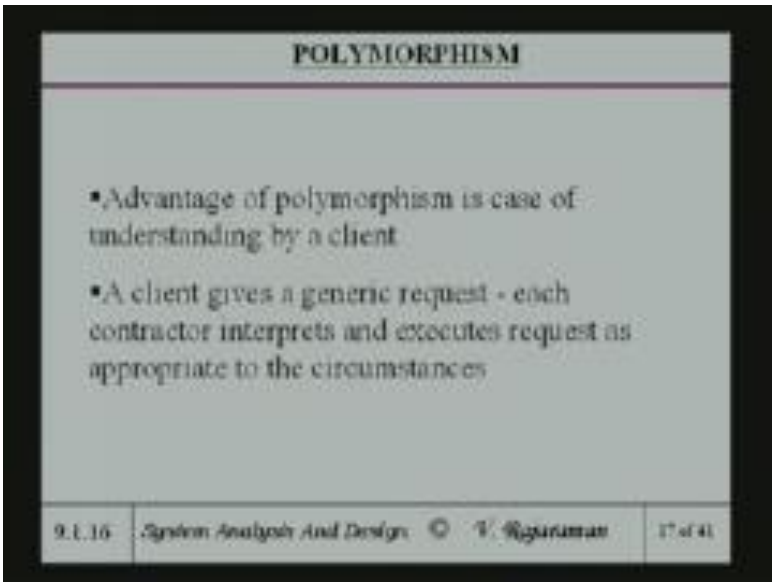
The advantage of using this that is in other words, the word the operation of attribute the operations admit takes on two different avatars. In one avatar, it is actually a related to the students. And the other avatar, which goes into the admitting in the hospitals. So, they are that is a reason what is called a polymorphism? There is a same thing has two different meanings, depending on the context. And one may ask, why should we have the same words used at two places?

The reason is, from a point of view of a user or a person who is going to use a system in the case of a context of a school, the admit will have a meaning. And in the context of

the hospital, the same word admit will have a different meaning. In fact, when we talked about the collection of fees for the sub classes art student and science student. I said the rules for fees may be quite different for the art student and for the science student.

So, you can just do collect fee for both these. In fact, you are using polymorphism in a way. In the sense that, when you say collect fee the rules which are implemented or the program or the methods which are encapsulated within the object for a sub class art student will be different from sub class science student. So, this is the whole idea of polymorphism. The same operation has two different meanings, based on the context.

(Refer Slide Time: 15:36)

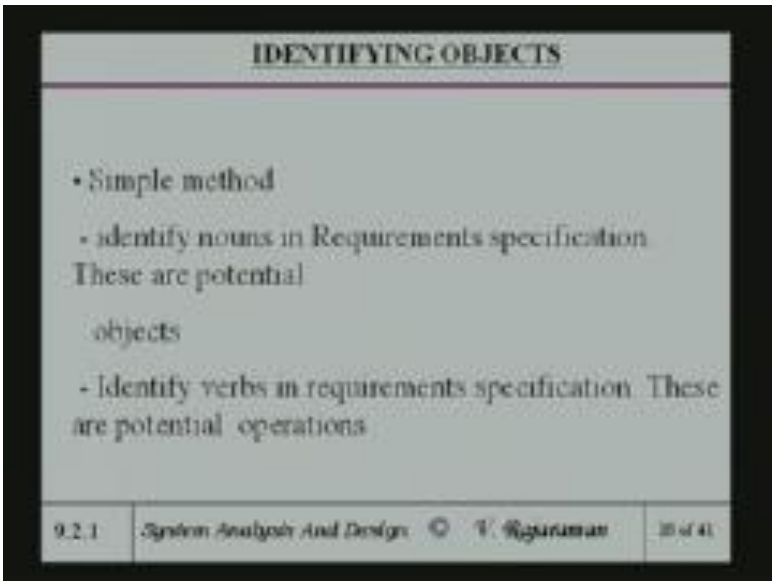


Advantages of polymorphism is as I said, ease of understanding. When I say collect fee you know, it collect fee is clear for an art student or science student. In fact, collect fee may even be the same is relevant for hospital or a school. In one case, it may be a fee, doctors fee the other case may be a school fee. But collect fee has got different contextual meaning, depending upon where you use it. So, a client use general request.

So, you can look at if I look at the object as contractors. So, the contractors will interpret this generic request. And execute it an appropriate circumstances. So, if there is a contractor who is doing the contract, then the contractor will interpret depending upon the contract you are given, like finding out the bill. So, you may have two different contractors. One contractor giving furniture's other contractor may be catering.

But, most cases you can say, if you give an order create a bill or bill the customer. Then, the each one we create this bill or bill the customer, based on their own individual rules and individual attributes and so on. So, from the point of view of the application it is a meaningful kind of a word. And the meaning is entirely contextual, based on the context.

(Refer Slide Time: 17:28)



Now, the greatest difficulty the most interesting problem, the most difficult problem in object oriented system design is to identify objects. What are the objects in an application? And that is even, there is no again algorithmic method of identifying objects. What I mean by algorithmic method is, you cannot essentially expect computer program to be able to identify objects.

The identification objects, because it is all semantic. It is got some meaning and because of that, and also contextual meaning. So, because of all these things it is a responsibility of a system analyst to be able to identify objects. And object identification is really very, very challenging part of object oriented modeling. And once you model, then this model can be used for system design. And then, at a lower level after you get programming.

But, modeling is the start of the entire cycle of an object oriented design, in general. So, normally you always start with a System Requirement Specification. SRS is the real start of your entire operation. So, the simple method is to identify nouns, the requirement specifications, their potential objects. See they are not, does not mean that every noun

which occurs in a system requirement specification is an object. Because, we normally find SRS will be very long statement.

Some cases, SRS may run into few pages. And you do not want to proliferate objects unnecessarily. Also identifying, if you kind of look at all nouns blindly, many of them will be irrelevant. So, you have to the most important part of it is having identified potential objects, which are all normally noun classes. Then how do we need out? And take out only those, which are relevant for a given situation.

So, that is the part which is challenging. Later on we will see in this lecture, that people have suggested many methods of doing this. In fact, one of the methods we will discuss requires that object identifications is not done by a single individual, who looks at the SRS. But, by a team which consists of both user and you are users or unless who may involved in this project. And the project head, who may who essentially looks at the entire project as a birds eye view.

So, a team normally is formed. And the team is the one which normally identifies. But, I am going to give an idea of, how the identification process is done? And later on, we look at a, so called modeling methods or the actual industrial practices you might say of doing this. Because as far as you are concerned, when you are in a college and learning. This you do not really come across very large systems.

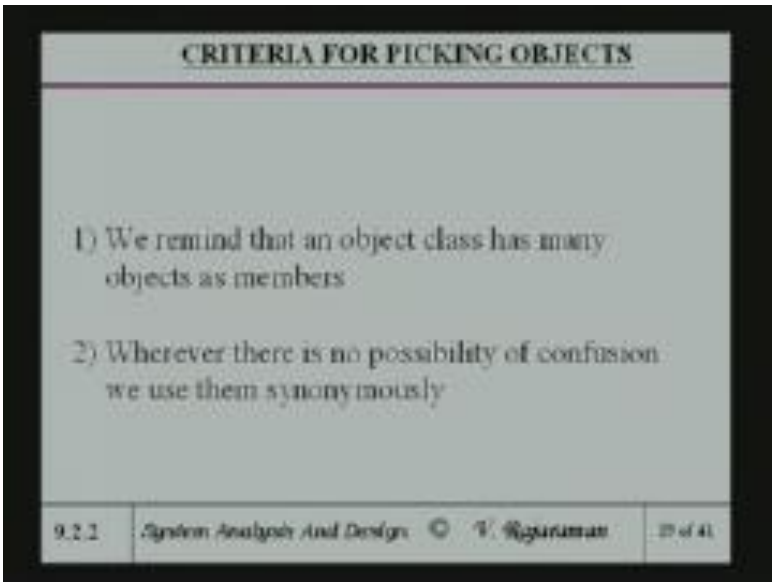
In fact, the entire object oriented modeling is relevant only for very large systems. A very small toy system, which encounter as examples which in fact, I should blame myself. Examples I have given, or really a toy example. Because, I have to be able to accommodate in a certain short span of time and I got a kind of explain, it the basic ideas. So, I cannot really take a real problem which is got.

You know something like ten pages of SRS and may lead to a hundred thousand lines of code and so on. So, by force we are we have to kind of look at only smaller problems. But, we have to keep in mind at the mind always, that the object oriented modeling and the entire object oriented design is much more relevant for very large problems. Where a large number of objects will be there and the idea of inheritance polymorphism and so on becomes extremely important.

In what I would say, programming in the large rather than programming in the small. Or equivalent might say information system design is the large versus information design small. We are essentially, these are problem these methods are relevant to the object orientation idea. It is relevant to the systems, when you do information systems may large.

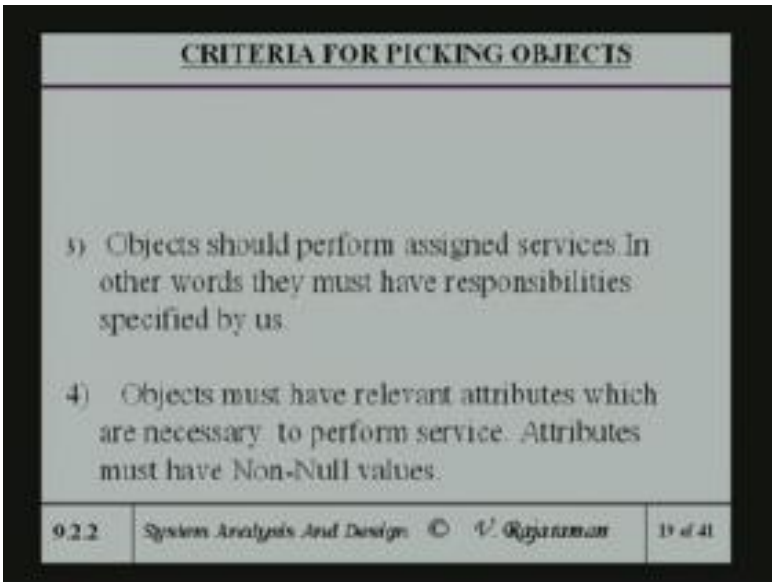
But of course, we all understand what this mean. Then, you identify what is in the requirement specification? These are potential operations. Not every game not every work as an operation. It is a potential operation. And so, we all to kind of again read out, some which are not absolutely essential.

(Refer Slide Time: 23:55)



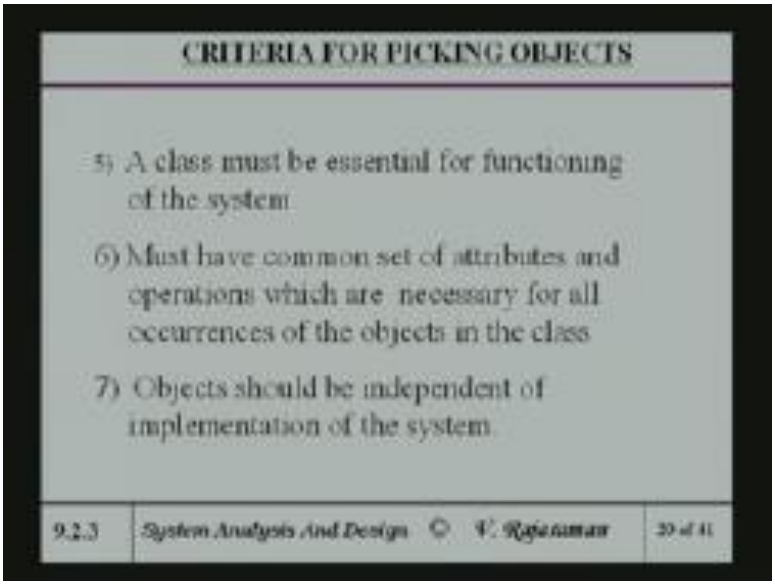
You remind again that, object class has many object as members. As it if the distance between the object class and an object individual object. But of course, when we will using it synonymously the ((Refer Time: 24:07)) also does it synonymously. Wherever there is no possibility of confusion, we use them synonymously. That is what we have been using. Object should perform assigned services.

(Refer Slide Time: 24:23)



In other words, they must have responsibilities specified to them, Specified by us by the system designer or system analyst to them. Objects must have relevant attributes which are necessary to perform service. Of course, attributes must have normal values. In other words, attributes must have values. In other words, why should be there. So, every attribute have put in to object must have some value. And well to play, it is no well to play. It has no business, being part of the attribute set.

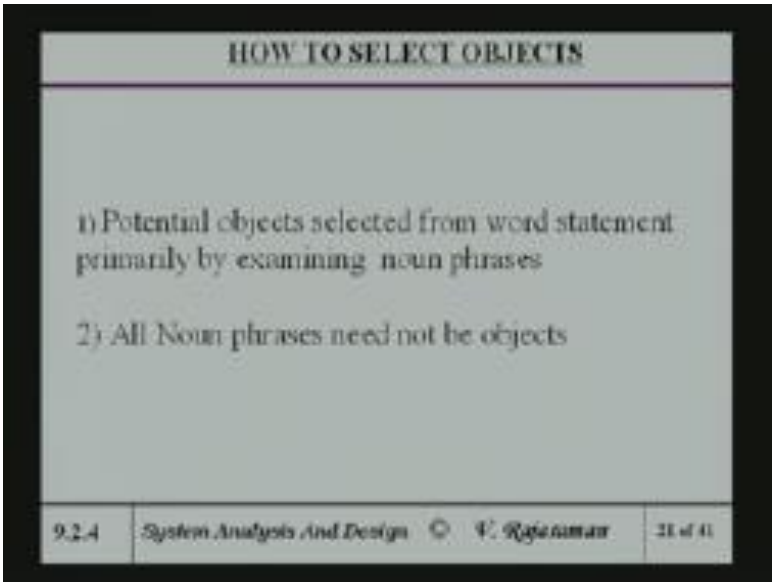
(Refer Slide Time: 25:06)



A class must be essential for functioning of the system. Other words, the information in the class must be preserved. Preserved in the sense that, is if it is not preserved system will not function properly. That is what really mean by a class being essential. Because we cannot lose it states or lose its attribute. If we lose its attribute or state, then system will not work. Must have a common set of attributes and operations, which are necessary for all occurrence in the object in the class.

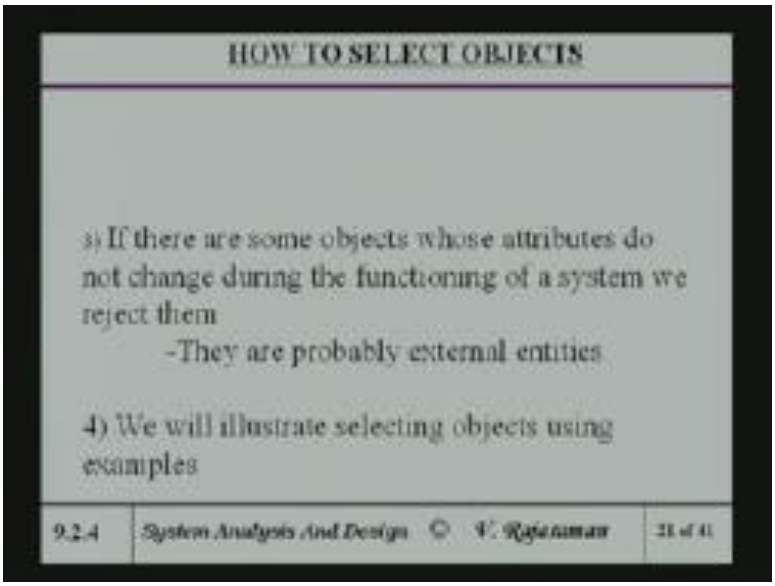
Because, the point is obvious from objects, all objects together for an object class. So that means, every object must have a attributes are common to object class. Object should be independent of the implementation system. This cardinal rule implementation, particularly the modulate time, implementation ideas are not at all relevant. Implementation ideas become relevant, only when you start programming. So, in this modeling there is no question of there is no you do not worry about implementation.

(Refer Slide Time: 26:26)



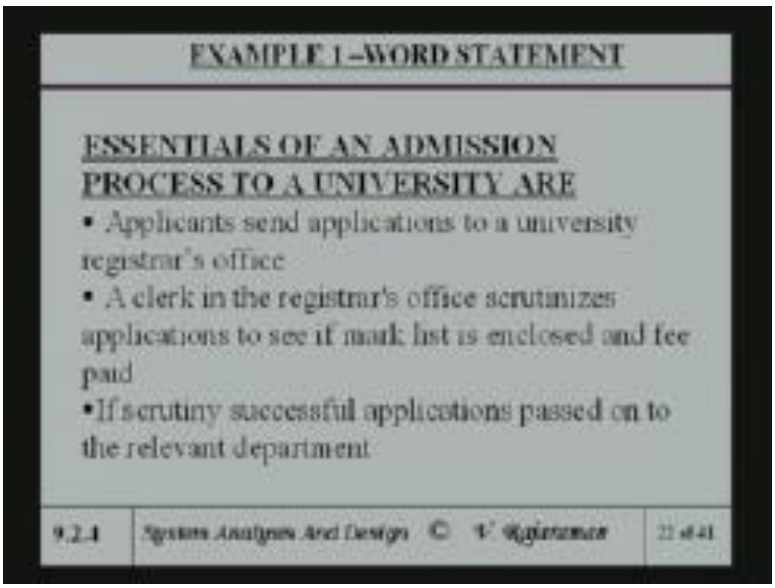
So, you select potential objects from word statement, primarily by examining noun phrases. And as I said the all nouns need not be. If there are some objects whose attributes do not change during the functioning of a system, we reject them. They are probably external entities.

(Refer Slide Time: 26:37)



In other words, the attributes will change during the course of the operational system. So, again as we illustrate selecting objects from an example.

(Refer Slide Time: 26:59)



I will take two examples. One example, I am going to take is a process of admission in a university. And there is a set of system requirement specification or you might say, set of rules. The applicants send applications to university registrar's office. So, I am really giving a sequence of events. So, the sequence of events starts with an applicant or several

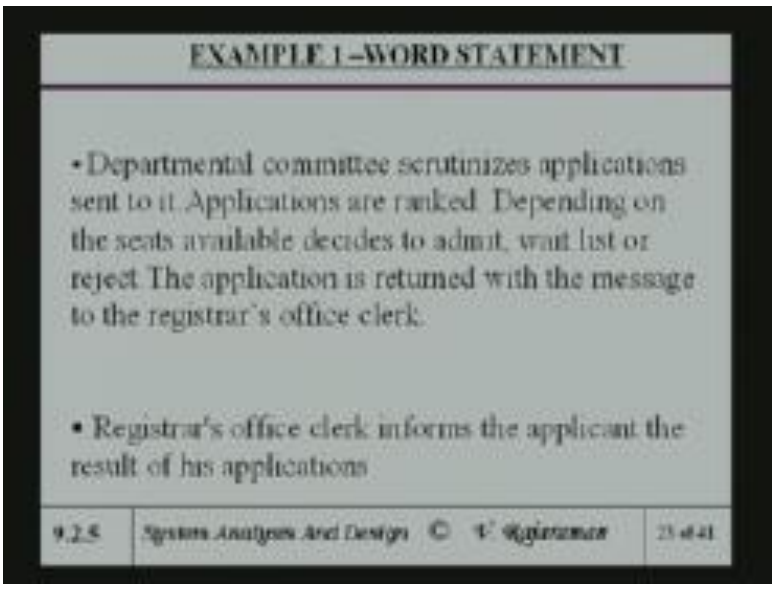
applicants. Who want to get admitted to a college by sending applications to registrar's office.

A clerk in the registrar's office, scrutinizes applications. To see if the marks list is enclosed and fee is paid. As far as registrar's office is concerned, they do not have any role to play in this example, in terms of the final decision to be taken. They just do the clerical job or checking, if the student application is complete, is got the mark list and it is got the fee required fees paid and so on.

And whatever requirements are there, along with the application whereas it may sent to the registrar's office. Whether they are fulfilled or not, is where the registrar's office does. If a scrutiny successful applications, the application itself are passed to relevant department. In other words, they will sort the applications. The application may be for different departments in the university. And each department, may have certain number of seats.

And they will have their own rules, regarding admission. In the sense, that the marks requirement may be different. And the minimum qualification may be different and so on. So, department is the age agency which does the does this kind of a final selection.

(Refer Slide Time: 29:28)



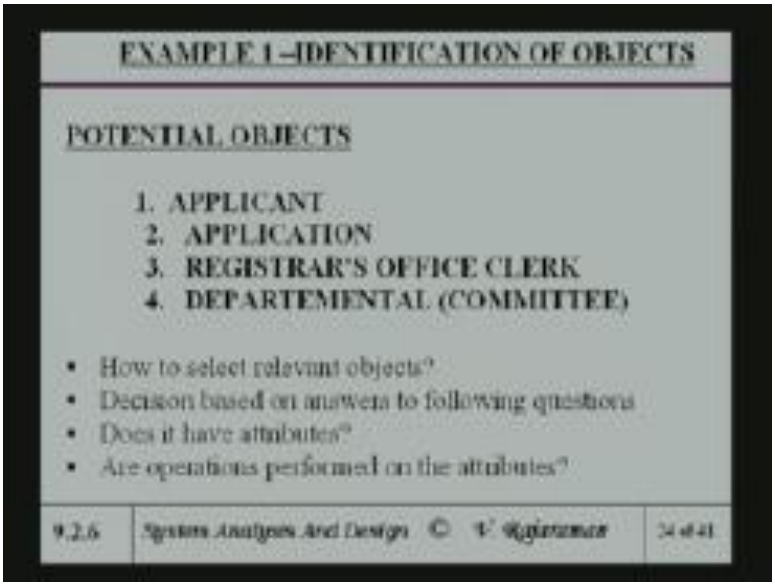
Departmental committee, there is department is got a committee which scrutinizes the application. Applications are ranked in other words, after scrutiny filing up the marks of

the student, other criteria. Whatever the criteria may be, they rank order the students in terms of a admission list depending on the seats available your rank order. Say if suppose you got 40 seats. You decide to send 40 admit letters.

And you may decide to have some something like 5 on the waiting list. That depends up on your earlier experience in terms of, how many people drop out after getting admission. That is judgment of the department. So, there may have some on the waiting list. And are beyond that, if they feel the student is not up to mark in for admission, reject. And having done this scrutiny and having taking the decision, they send back a message.

You might say registrar's office, thus the appropriate clerk who send that may be or you can send to the office. In a office, will do whatever you want to do with it. And I will send back the message that, so and so is admitted. So and so is wait listed. So and so is rejected. And the registrar's office clerk, when he receives this intimation from the department. His responsibility is to send a letter to the candidate informing, whether he is admitted, put on the waiting list or rejected. So, the result has been given to the student.

(Refer Slide Time: 31:42)



Potential objects, kind of looking at all the nouns or applicant. Applicant, I can say applicant is a noun. And applicant is looks like as I, you know beginning of the entire cycle of the applicant, give an application. And registrar's office has got a clerk, who does this job. And there is department committee is there. Now, the question is how to

select the relevant objects from this? Is it the clerk or is it the applicant or is it a application, you have to decide.

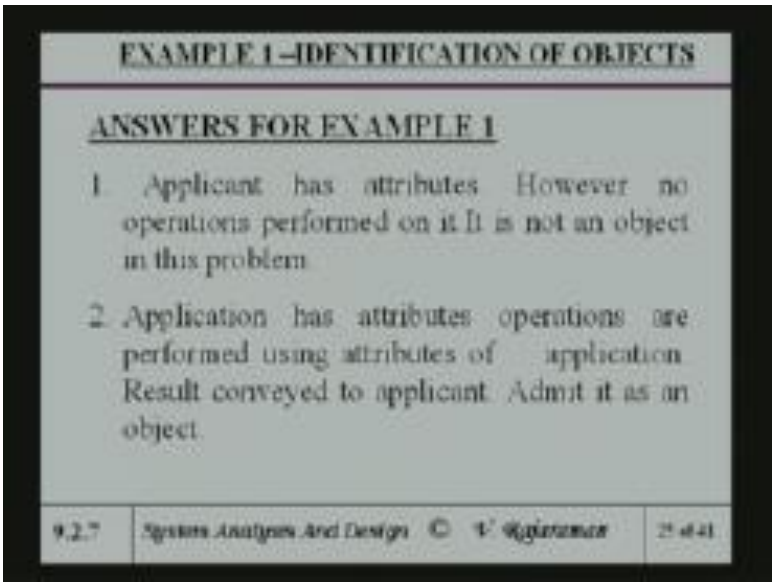
So, the decision is based on certain questions you ask. In other words, you try to do the filtering based on certain criteria. And the criteria is, certain questions you ask yourself as a modular. Does this satisfy these criteria? But, satisfy these criteria I will take it as a potential object. Potential object I would kind of take it as a real object. Otherwise, I will reject it or does the whether your attributes or operation performed in the attribute.

Now, these are two important questions, is in it? And if you look at applicant, which the potential object applicant of attributes. As the applicant name, roll number you know. Roll number is not may not would not be there, because in the applicant. There will be a serial number or something which is aside. But, that assigning will be done by the registrar's office, not by the applicant.

But, the applicants name, address, degree passed and all that will be there. But, the operations are not performed on the applicant per say. We are performing operations on the application. See applicant is, as far as the procedure for admission is concerned. The identity of a applicant is irrelevant. Whether, it is Krishnan or Raman or Srinivasan, it does not really matter as far as the procedure is concerned.

The process, the process depends only on certain attributes like marks. The address of the applicant is not relevant, his name is not relevant. And only thing which may be or relevant is in terms of whether, he has obtained degree. What marks would he get? What subject did you passed in? As is fee paid and is a marks list enclosed, things like that. So, the applicant has no relevance but, the application is the one which is actually used by the clerks.

(Refer Slide Time: 34:51)



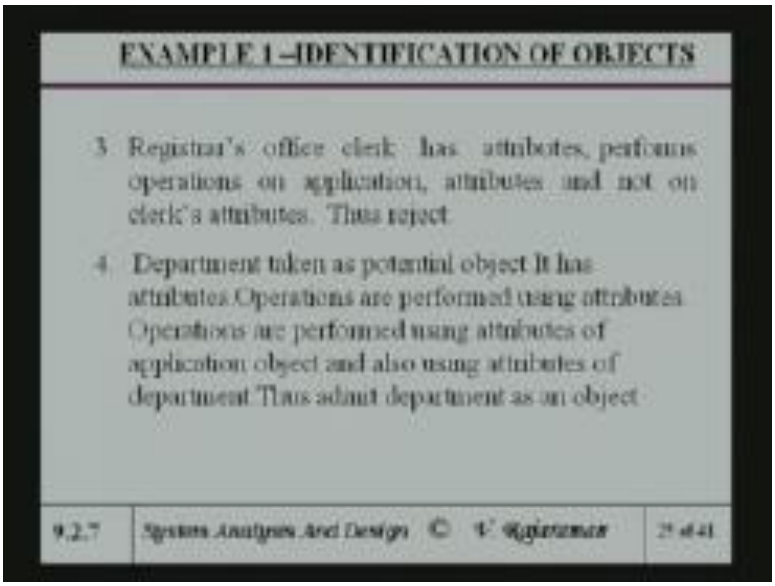
So, application of attributes and the operations are performed in the application. Whatever attributes are there like for instance the marks list, the fee paid. And all that attributes are used by both the registrar's office clerk as well as the departmental committee. To come up with a decision of whether, admit or wait list or reject. So, is got a application has got a attributes and operations are performed on using this attributes.

Result is conveyed to the applicant. But, the result is based on the application. And the address is required only for communicating the result. So, there is no operation. Only he gets a signal, he might say a message. So, this applicant might call is an external entity. See external entity, who sends some data and gets back some result. In this case, data is application result is admit or not admit or wait list or whatever.

So, admit applications are object. You might say application looks inanimate. See, but it is a kind of a tangible entity. And you are doing operations based on that, what is contained in the application. So, it is essentially a document which is got a meaningful existence, as far as the system is concerned. So in the model, application becomes a class. So, a specific application, it is a member of the class applications.

So, the class general class object may have individual objects, which are all several applications, a set of applications. Registrar office clerk, the clerk itself is got attributes like his age, his salary, his address and so on.

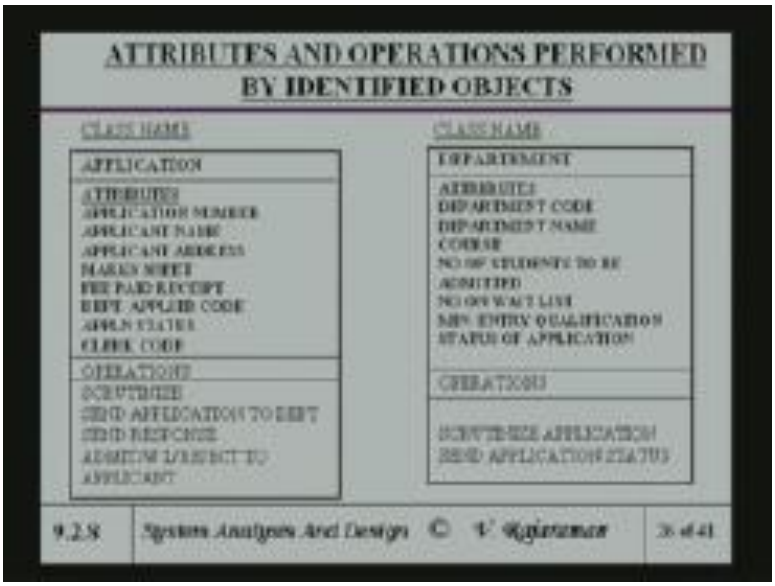
(Refer Slide Time: 37:05)



But, that they are irrelevant. What the task performed by that applicant of the clerk on the application are the only things which are relevant. So, reject the clerk as their object. But, registrar's office has some function to do. And the function which performed by the clerk, you might say. So, to that extent it is a you might that is the operations performed are relevant. But, that they are all performing the application. And so, application is a relevant to one.

And then, department there will be some operation will be performed by registrars office, which we will be come to. Department has taken a potential object. It has attributes. Operations are performed using attributes. And the attributes are used by the department. And so in other words, the operations which are performed use attributes namely the marks, mark list, what marks you obtained, which university passed from and so on.That is all used. And the operation also performed. So, department is a potential object.

(Refer Slide Time: 38:47)



So, application I am admitting as a class and departments admitting as another class. And operations on the application are stored in certain application department. And send response to the student, admit lately will reject the application. So, response is admit wait list, reject to the applicant see. And scrutinization is done by the clerk. And send application to the department, the clerk does it.

So, attributes of a application number, applicant name, because you normally assign a number as I said. The applicant name, applicant address, mark sheet, fee paid receipt, department applied code, applicants status is admitted or not admitted, because it will become a permanent record later on. If there is a later on question asked about a particular applicant. Because, may be the applicant did not receive or reject any letter.

So, if he enquires about a status of the application, immediately the office must be able to respond. And give a status, whether it is still being scrutinized or it is in the department or has letter already sent. So, there are three possible states. Being scrutinized in there in the off registrar's office, sent department that mean department scrutinizing. And third is, the decision taker and letter dispatched. So, that is important.

Now, I add a clerk code there, the application. Mainly because, if there is some normally you would like to track down, which clerk scrutinizes which application? The reason being, if suppose there is some later on problem. And you would like to find out, who

did this job? Suppose the fellow has grooved up. And you should be able to pin point responsibility. That is the reason, why we put the clerk code there.

And of course, if the requirement also says that the department looks if the clerk code and sends it back to that particular clerk. But, normally it is not done, because the department will just send it to the registrar's office. And registrar's office may use the clerk code to assign this task of sending admit reject letter, back to the student to a same clerk whose do that. So, to that extent clerk code is important from the for you might say.

So, to audit trail it keeps a trail of what happen. And if you look at the department, departments attributes are departments code, departments name, courses offered by the department, number of students they admitted to that particular course. Number to be wait listed, minimum entry qualifications of that course, status of the application. That is in other words, again status of the application is in this case.

It is being examined or it is being decision has been taken or its kept in abeyance may be. You see if they are, they have some problem with in terms of the authenticity of the marks and so on. They may put it on put it in abeyance or something like that whatever. So, depends on the particular detail depends upon the particular situation. The operation of course, scrutinizes the application, send application status.

That is after scrutiny, send the application status to the registrar's office, which will release this status information to act further. Now, this simple example where there are only two objects. And two objects communicated with one another. And external entity being a applicant. But, this if you have this they are brings out another important point. The point is that the admission process may be different in different college or universities, in the sense that the scrutiny and so on may be different.

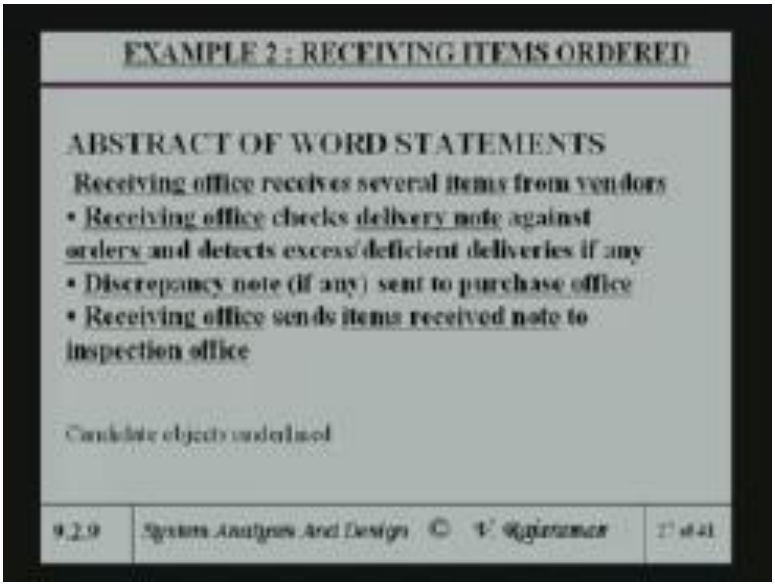
But, normally the attributes will be same, operation will be similar. So, in other words today we may be doing something for the university x. Tomorrow you will be ask to model for similar problem for a university y. In this case, you do not have to go back to a drawing board. And start from scratch for university y because, you already got university x. And you can reuse the same objects.

The reusing being that the idea of application department is will be common. The only the details of how that applications is scrutinized. And how the rejection or acceptance

takes place, would vary from university to university to university, or like college to college. So, the name of the operation may still be same, scrutinize and so on. But, the methods you implement may vary depending upon the specific university.

So, in other words these operations may be or polymorphic. You have different situations, you may implement different methods. But, attributes are somewhat common. So, that is the whole idea of saying that reuse is promoted by object orientation. So, reuse is possible.

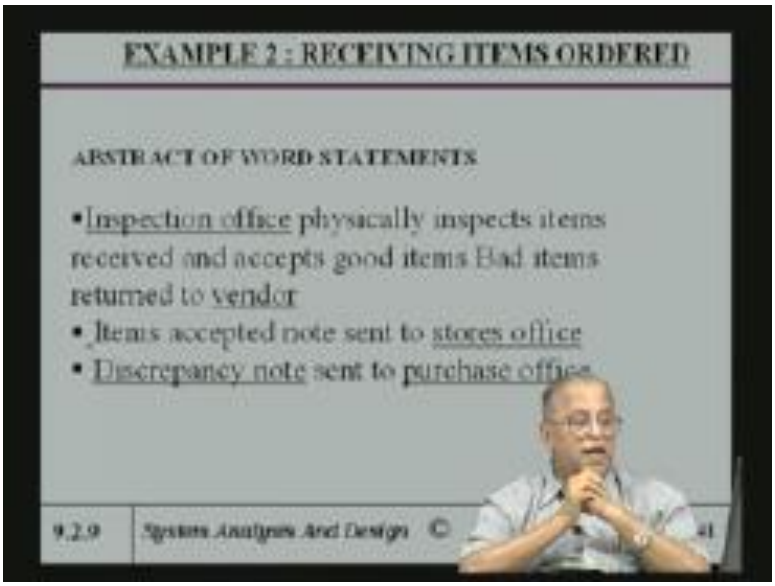
(Refer Slide Time: 45:45)



And then, one more little bit more complex example, which we started looking at the very beginning. This is an example of a company, which receives a lot of goods from vendors. And they are all received by receiving office. So, lots of items come into the receiving office from vendors and receiving office checks delivery note, which comes along with the item against the orders. In fact, we came up with a data flow diagram for this earlier.

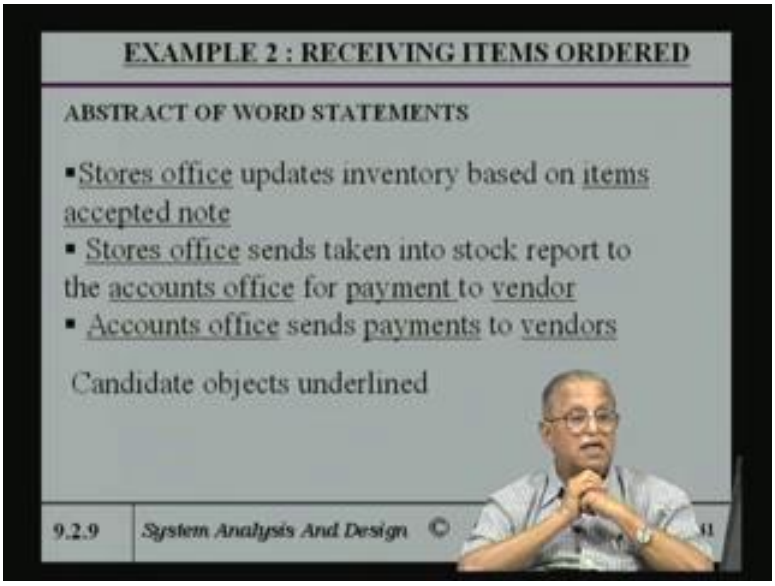
And against orders placed and detects excess or deficient deliveries, if any. So, if there is a excess delivery or a deficient delivery, then the normally the purchase office will be informed about this. So, discrepancy note is sent to the purchase office, if any. And the receiving office after receiving the items sends an item received note, which may be a copy of the delivery note. Or it may be somewhat different for inside internal structure of the internal data structure may be there to the inspection office.

(Refer Slide Time: 47:18)



Inspection office is the next entity. The inspection office physically inspects items received and accepts good items. So, which again we said earlier and bad items are returned to the vendor. So, items accepted note is sent to the stores office. Stores office gets the items accepted. Discrepancy note sent to the purchase office.

(Refer Slide Time: 47:51)

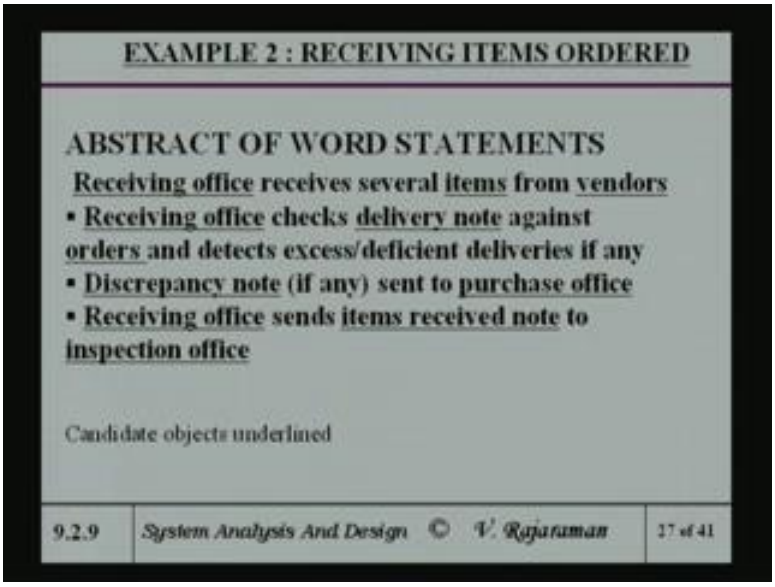


Stores office are update inventory, based on items accepted note. Whenever inspection, inspects or accepts then the stores will update the inventory. And once they update the

inventory, they send a taken into stock report to the accounts office for payment to the vendor. So, that means these number of items came, found good by the inspection.

And we have got a document saying that, these has been accepted and have been taken into stock. So, the stocks are inventory is being updated by the stores. And because payment has been made, after the inventory is updated the information is sent to the accounts office. And the accounts office, based on this note will send a payment to the vendor.

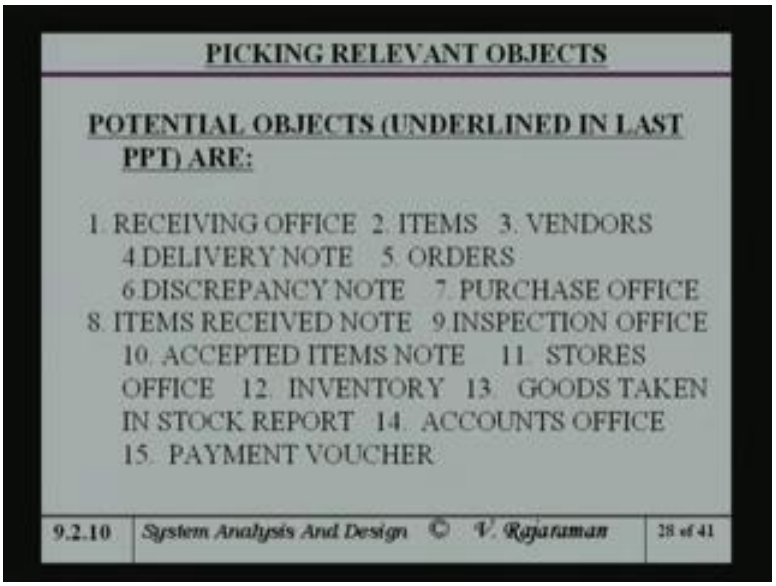
(Refer Slide Time: 48:54)



Now, what I done is underline all the candidate objects. Candidate that is potential objects, I call it as candidate objects. So, receiving office is a potential object items, vendors delivery note, orders discrepancy note, purchase office against receive of course, receiving office, again items receiving note and inspection office.

These are potential objects again inspection office, again items will be there vendor, items accepted items again. And stores office, discrepancy note and purchase office, again stores office, items accepted note, stores office, accounts office, payment and vendor accounts, office payment and vendor it is again come here.

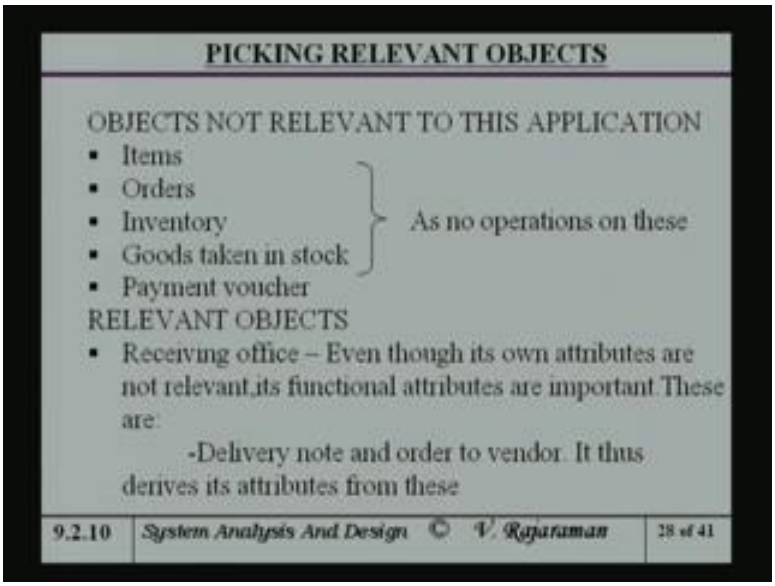
(Refer Slide Time: 49:52)



Now, we took all those underline potential objects. And list up receiving office, items, vendors, delivery note, orders, discrepancy note, purchase office, items received note, inspection office, accepted items note, stores office, inventory, goods taken in stock report, accounts office, payment voucher. Fifteen potential objects are there. Some of these are got are physical entities, like items vendors and so on. Some are documents like orders, delivery note, discrepancy note and so on.

And some are entities which do some job like receiving office, purchase office, stores office, accounts office and so on. Insurance got a function to play, based on the documents which flow. So now, we have a job of filtering out from the set of fifteen potential objects. Let us see, if there is any more there is fifteen potential objects here. And take out those, which seem to be relevant.

(Refer Slide Time: 51:29)



Now, how do we start picking out, what is relevant and what is irrelevant? This is the most difficult part. In an object oriented modeling to kind of do an informed selection or a common sense selection, you might say. Because, that informed in a common sense selection that is why I said very often, expert suggest that need not should not really be done a single analyst. Normally it should be a team, which kind of decides.

That is what many experts in this area say, I also agree with them. That you know, what is not? It can become arbitrary even particular person, does it. There is a small group, which does it. May be, they will debate. And based on debate, certain good ideas will come. And what can be included and what need not be included. So, from that point of view a team is always very useful. So items, orders, inventory, payment voucher all these are better attributes.

But, there you know actual operations performed on the items. Similarly, the orders has a dump dump document, which comes. You know operations performed in the order in that sense, which order has been placed already. Inventory as of is some kind of a database, which is cylinder. And payment voucher is again another document, which is not really any operations are not performed. It is actually result of a certain operation, formed as a accounts office.

So, I will feed to be look at the functional parts of functional entities, you might say. In the set of permissible objects, which are operations to perform on the attributes of

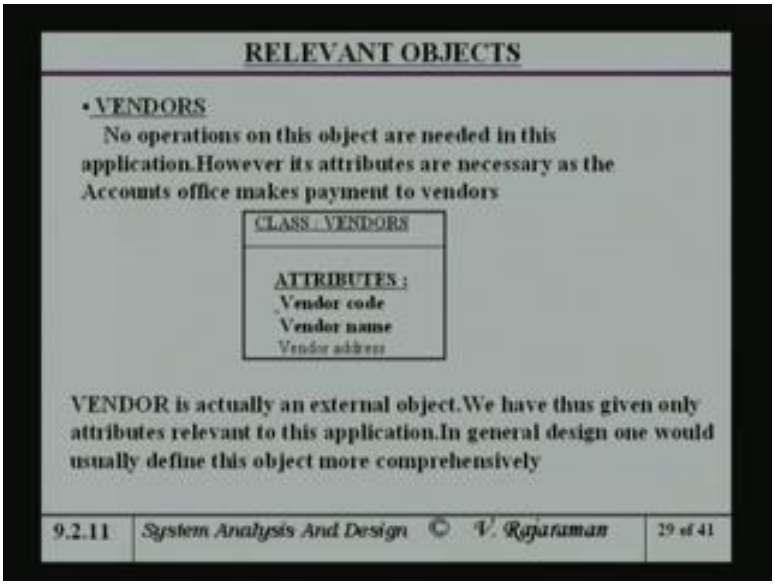
whatever documents they need to be able to perform the operation? As they have been always saying, object is an it is a something which got attributes encapsulate with operations. In other words, there are there is some data structure which is encapsulated along with the operations on the data structure.

The operations are triggered by some messages or request coming from a external source. Or another cooperative object, which may require which may send this request for cooperation. So, in general we look at an object oriented system as set of objects, which all has some independent existence. But, they collaborate or cooperate to get a particular job done. So, from that point of view receiving office even though, the office attributes are not relevant, its functional attributes are important.

And it derives the attributes from the, that you know in another delivery note comes to receiving office. It compares a delivery note against the order. It is in place of the vendor. Then, the comparison some action is taken action being sending, it a discrepancy note and also sending the items of receiving note to the next step, namely inspection. So, from that point of view you might say that, it is a potential object whose attributes are ((Refer Time: 56:09)).

They got it has to inherit the attributes of delivery note. And order to vendor that is, these two thus to inherit that attributes. And operations of course, specific for the what the receiving office suppose to do. So, those operations will be relevant. Look at vendors.

(Refer Slide Time: 56:33)



Vendors looks as a, it is a potential object, because there are an attribute like vendor code, vendor name and vendor address and so on. But, there is no particular operation perform on the vendor, in this case. Suppose, there was different situation there, we want to kind of a higher kind of new vendor added. Or the vendor has some operations can be performed on the vendor namely like, you know address change or things like that.

They of course, it will become a potential object. In this case, I am assuming that vendor is a no operations of object are needed in this application. The attributes are necessary for the accounts office to make payment, because address is required. So, vendor is an external object to with whom you interact. The sense that, the vendor is a person who sends the item he delivers an item with a delivery note. And the vendor is a ultimate person to whom the payment goes.

So, vendor essentially is an external entity. And attributes of vendor is important in point of view of the operations performed by some other object. But vendor itself is an object. It is not really relevant in this example. There may be other examples, where vendors may become relevant. If vendor plays a very central role in the entire, in this case some extent you can say is got a role to play.

But, there are certain operations which if suppose have to be perform in the vendor like, you might say reject a vendor and so on. There it becomes necessary. So, I think I continue from this point onwards. And look at other potential objects, because it is a very long example. And I cannot, I am running out of time. So, I think I will continue from here, and revisits the example next time.