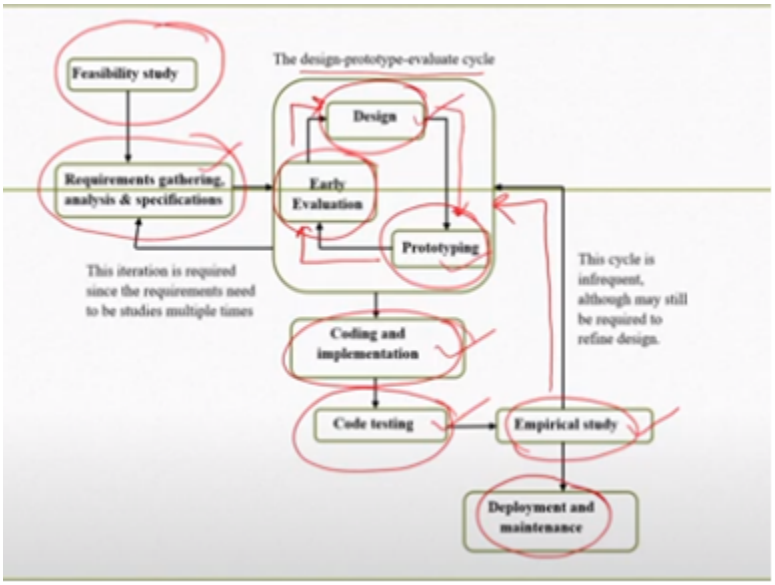
Design & Implementation of Human – Computer Interfaces
Prof. Dr. Samit Bhattacharya
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Module No # 05
Lecture No # 24
Introduction to Object Oriented Design

Hello and welcome to the NPTEL MOOCS course on design and implementation of human computer interfaces lecture number 21. Where, we are going to start discussion on object oriented design before we go into the topics of this lecture.

(Refer Slide Time: 01:03)



Let us first briefly recap what we have learned so far, if you may recollect we are talking about design and implementation of software systems which are interactive systems. Now in order to go for a systematic design approach we learned about design life cycles, for interactive systems we learned about specific design life cycles that we are following in this course. Let us quickly recap the stages of the design life cycle so the interactive system design life cycle consists of several stages.

Starting with feasibility studies which; we did not discuss in detail in this course. So we will skip it. The first stage that we discussed in detail is the requirement gathering, analysis and specification stage if you may recollect in this stage. We gather end user requirements where end

user requirements, actually refers to both client requirements as well as the actual user requirements.

This is followed by a group of sub stages together we call it design prototype evaluate cycle. So it consists of a design stage, a prototyping stage and an evaluation stage together they form a cycle as shown here. So once we gather the requirement we then go for design of the system then we prototype it and get it evaluated to see if there are any issues with the design if there are issues found in the evaluation.

Then we refine the design, go for prototyping again, go for evaluation again and this goes on till we arrive at a stable design that is followed by the coding and implementation stage; where we write the actual code for the software and we implement the whole system. The next comes the code testing phase where we test the code that has been written to implement the system. Then comes, empirical study stage in this stage we go for testing the usability of the interactive system.

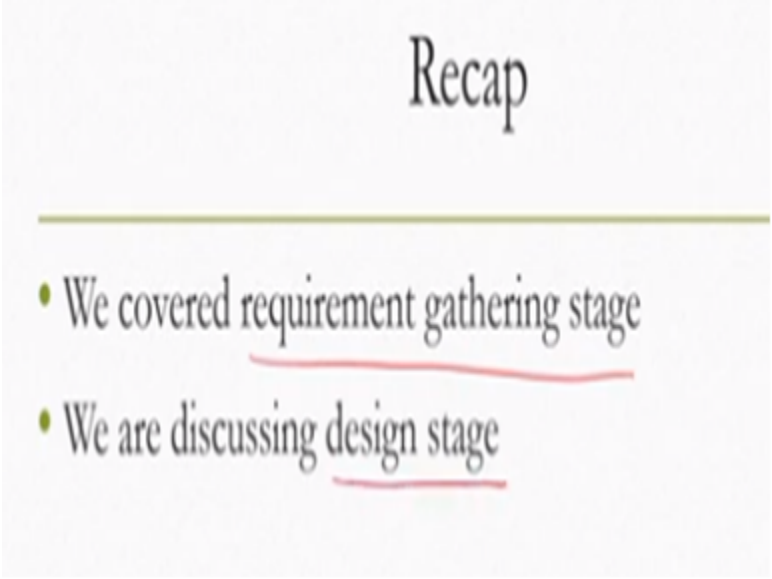
Recollect that usability is the most important concept that differentiates interactive systems or software systems that are interactive from other software, and in this stage we go for systematic study of usability of the end product. Now at this stage if we find some usability issues we may go back to the earlier stages as shown here and repeat the stages in the sequence. However this cycle is or ideally should be very infrequent maximum once or twice otherwise the overall cost and turnaround time of the project will be significant.

Once we arrive at an execution-able and usable system we go for the next stage that is deployment of the system as well as the maintenance phase. This is in summary the stages of the interactive software development life cycle, through which we get our system developed and deployed. You may also recollect that at the end of each stage we generate some output for the requirement gathering stage; the output is an SRS or software requirement specification document.

After the design stage we get the design document, after the prototyping stage we get the prototype document or prototype systems, after evaluation we get evaluation results. After implementation coding and implementation stage we get a code, after code testing we get the

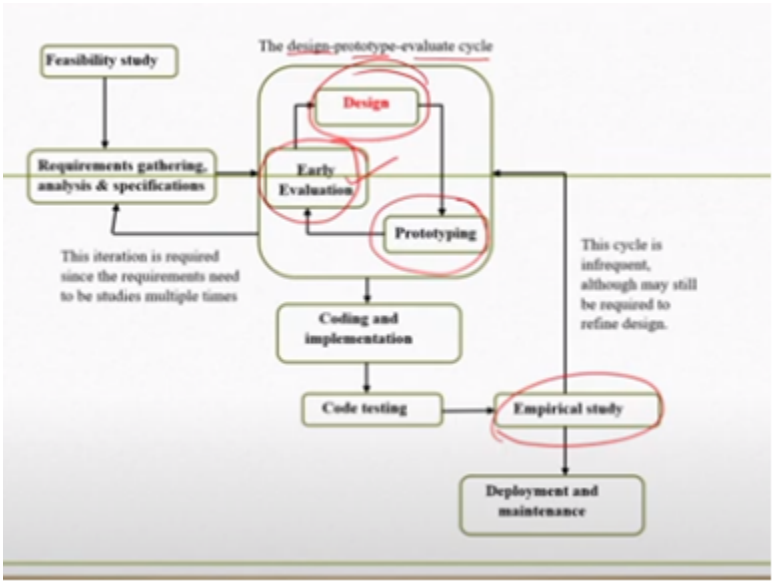
testing documents, after empirical study we get the usability evaluation document; so these are outcomes of the individual stages.

(Refer Slide Time: 05:37)



So far we have covered in detail the requirement gathering stage including how together and how to specify functional, as well as non-functional requirements currently we are discussing the design stage.

(Refer Slide Time: 05:55)

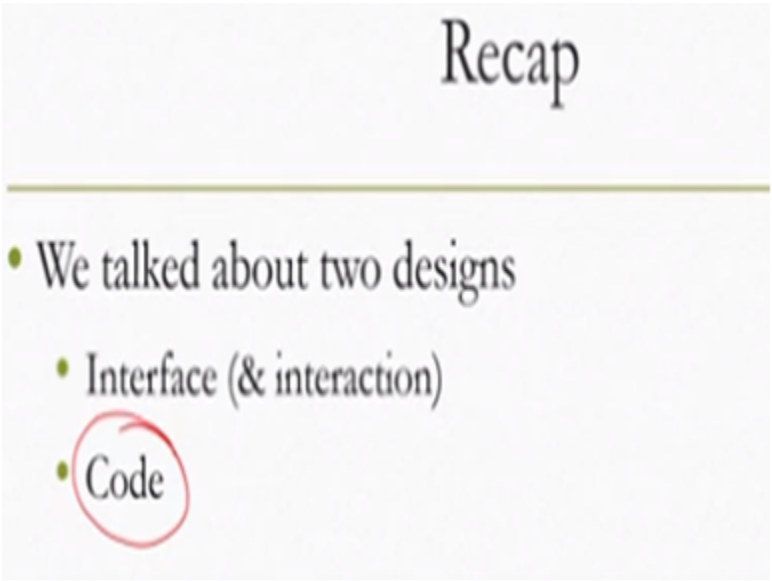


Now we are currently located in this stage here, now if you may recollect then when we talked about design we actually refer to 2 types of design; one is interface design one is code design. So

when we are talking of interface design the design prototype evaluate cycle is important. Here when we are trying to design for an interface which is usable we first come up with the design idea then we prototype the idea and get it evaluated for usability with expert users.

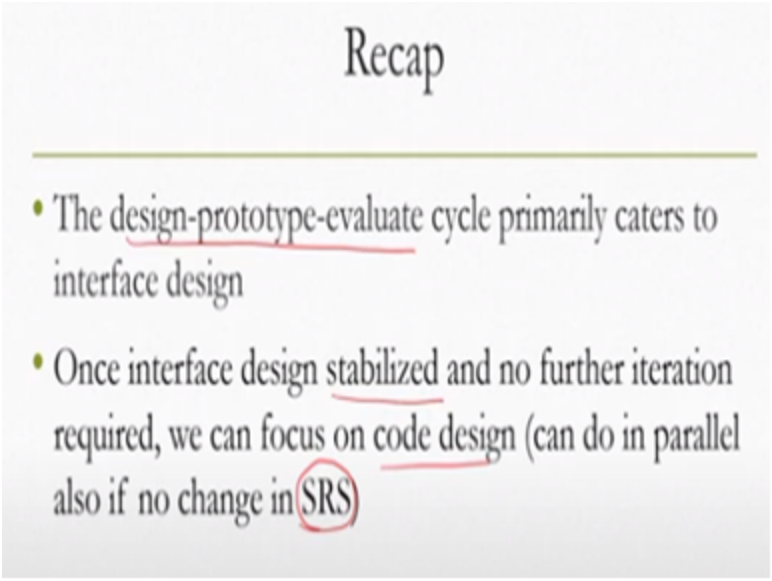
So this is different from the other usability evaluation that is done at this empirical study stage, in the sense that in this stage; early evolution stage we get the design tested by domain experts rather than end users. And this cycle goes on as long as we do not get a stable interface design with a minimum number of insignificant usability issues.

(Refer Slide Time: 07:07)



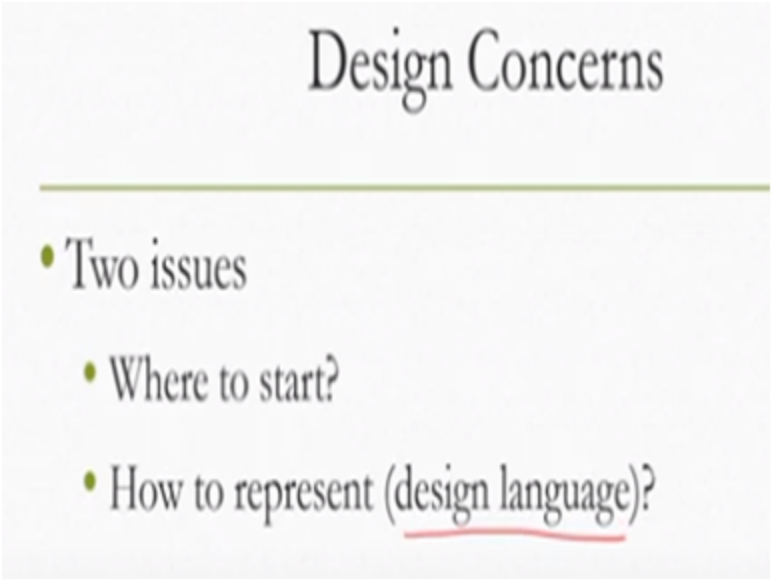
The other design; so once we get an interface design done through the multiple iteration of the design prototype evaluate cycle then we go for the next phase of design that is code design. Here we try to come up with a design of the entire system in terms of modules and sub modules which can be directly implemented as code or program.

(Refer Slide Time: 07:38)



So the design prototype evaluation cycle primarily caters to interface design; it has no role to play in the code design. Once the design stabilized interface design is stabilized and no further changes are required we focus on code design, it can be done in parallel also. If we already know in advance that no further changes will take place in the software requirement specification document that we get at the end of the requirement gathering stage.

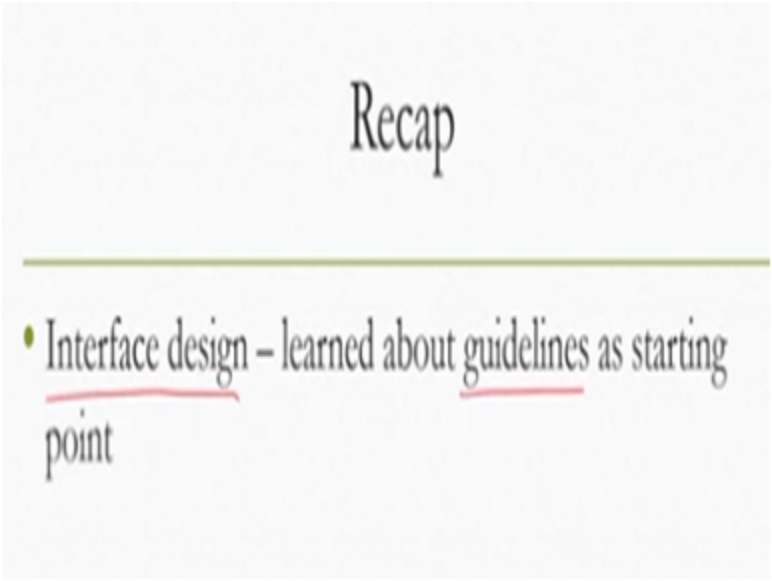
(Refer Slide Time: 08:18)



So when we are dealing with design there are primarily 2 issues one is where to start the design process, so design is also a process we can think of as a process so where we should start this

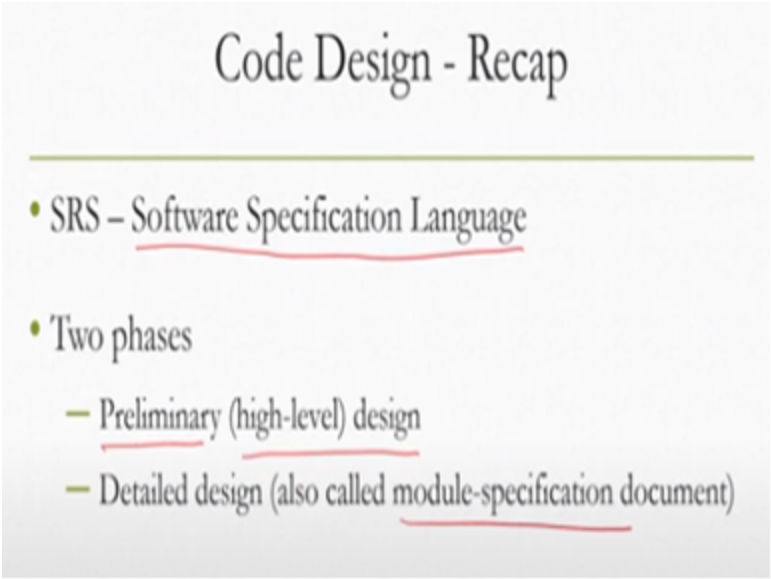
process. And second is how to represent our design in other words what language we should use to represent our design.

(Refer Slide Time: 08:45)



Earlier we talked about interface design, now that we have learned where to start we actually mentioned that we can start the design process based on our own experience or intuition. Also if we do not have sufficient experience or intuition or we want to augment this capability we can go for a starting point as the design guidelines. So there are several guidelines available. We can make use of those guidelines to start our design process. We learned about a couple of guidelines namely Schneiderman's eight golden rules, Norman's seven principles.

(Refer Slide Time: 09:31)



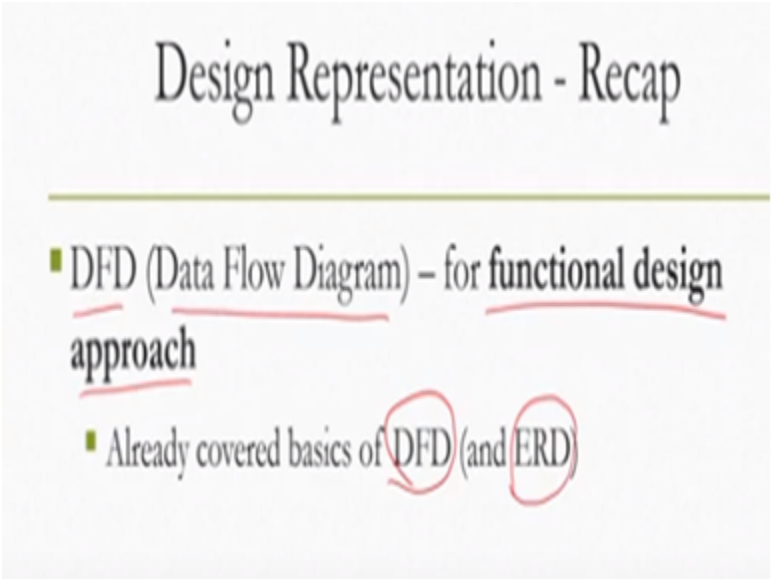
In contrast when we go for code design our starting point is the SRS document or the software specification document requirement specification document which we use as our starting point for code design. Now in the design process if you may recollect we mentioned 2 phases, so when we are going for code design we go for this design process in terms of 2 phases. The one phase is the preliminary phase also called high level design phase, and the other one is the detailed design phase also known as module specification document generation phase.

So these 2 phases primarily guide our approach to design of the code for implementation of the system.

(Refer Slide Time: 10:29)

Now when we talk of the approaches, there are broadly 2 design approaches used for code design one is function oriented. So in this case when we think of the design we think of the whole system as a collection of functions, so the basic abstractions used in order to design the system is the idea of functions.

(Refer Slide Time: 11:00)

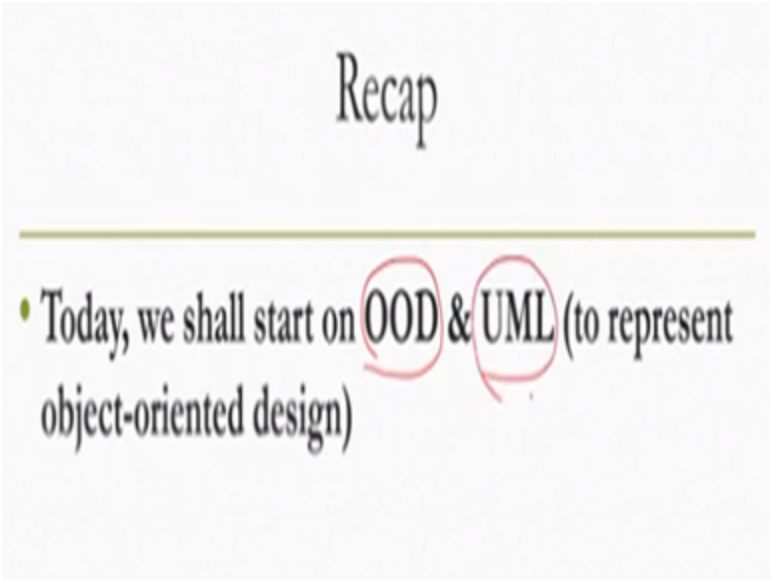


The other approach is object oriented design approach in this case the basic abstractions that are used to go for design of the system are objects, which in turn are instantiation of a concept called class. Now earlier we learned in detail about DFD or data flow diagrams. As we mentioned this

is the language using which we can express the function oriented design of a system. So this is primarily useful for functional design approaches and we learned various aspects of DFD.

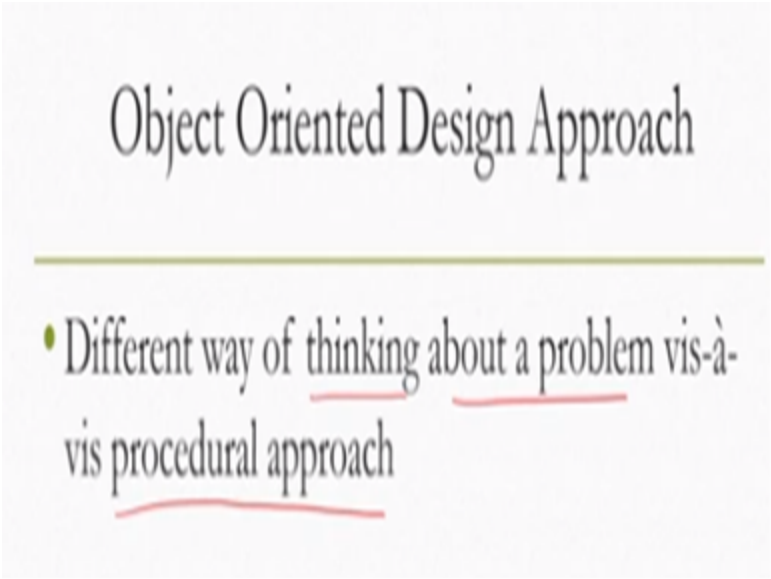
We have covered briefly a related concept called entity relationship diagram which is useful to represent data storage that are part of the design data storage elements that are part of the design.

(Refer Slide Time: 12:05)



In this lecture we are going to start our discussion on the other design approach namely object; oriented design approach which is also known as OD. And in subsequent lectures we will also learn about a design language that can be used to express such object oriented designs called the UML or unified modeling language. Let us first learn about the basics, the basic idea behind object oriented design approach, what we mean by objects and how we can use these concepts for designing a system.

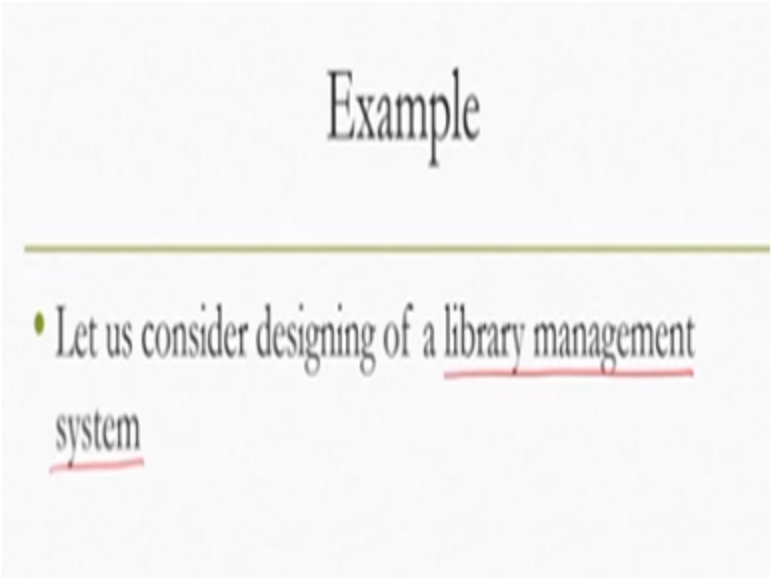
(Refer Slide Time: 12:47)



Let us start with the basic idea now when we are talking of object oriented design. Essentially this is a different way of looking at the design problem. We are changing our way of thinking about a problem vis-a-vis the functional or procedural approach. So earlier we learned about procedural approach or the functional approach both are same. When we learned about that approach intuitively we thought about the system as a collection of functions.

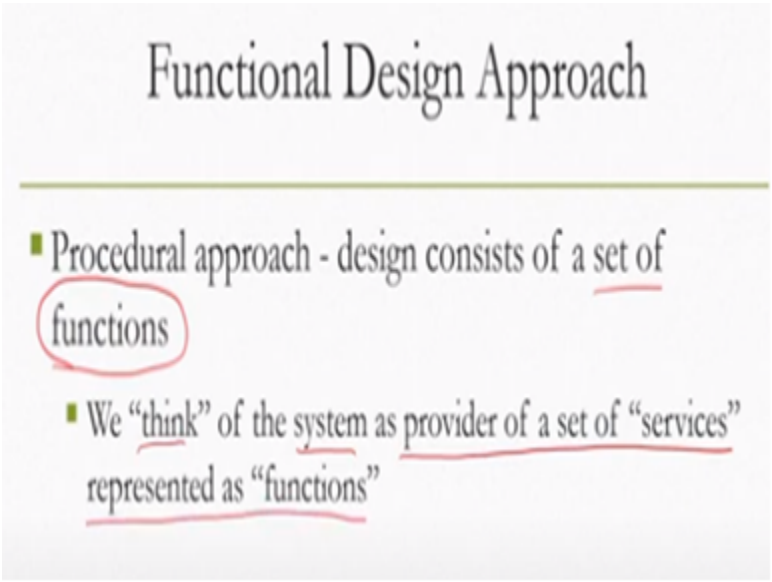
Here in object oriented design we are actually changing the way we think about a problem. Here our basic constructs become objects rather than functions as we shall see in the subsequent part of this lecture.

(Refer Slide Time: 13:44)



To understand the idea let us consider an example, let us consider a library management system so as the name suggests this system is meant to manage the operations that take place in a library. And suppose we are given the task of designing this system: how we can design the system, what the basic units are and; how they are related to each other. Let us try to address this design problem from both; the point of view or both the approaches namely the functional approach as well as the object oriented approach to better understand the difference between these 2 design approaches.

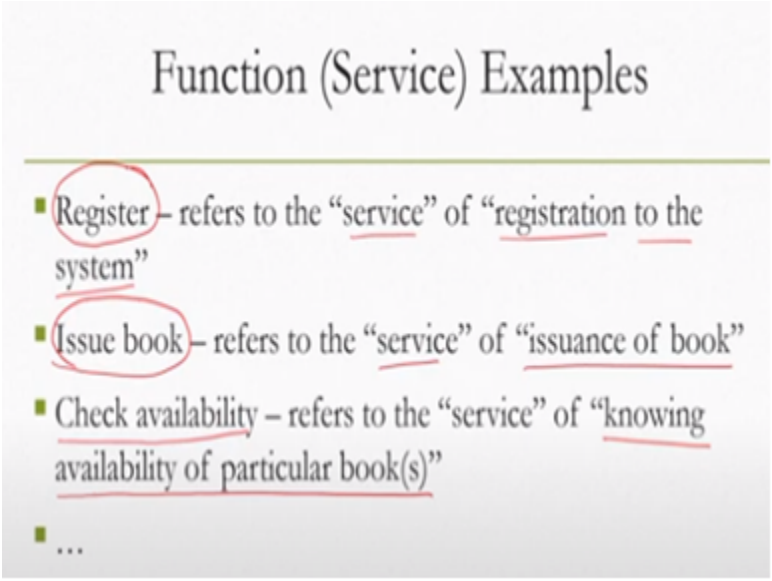
(Refer Slide Time: 14:33)



Let us start with the functional design approach for library management system design, as I said functional approach is also called procedural approach so procedure function these will be using interchangeably they are the same. When we are trying to design the library management system with the help of a procedural or functional approach there we can come up with a design that consists of a set of functions, what can be those functions.

Before we go into that, what does this signify? Let us try to understand that this signifies that we think of the system as a provider of a set of services. Now this is very important so when we are approaching a problem of system design and if we are taking help from a functional design approach then intuitively we think of the system as a provider of a set of services which are represented as functions. So that is our basic thinking that goes behind the design.

(Refer Slide Time: 15:56)



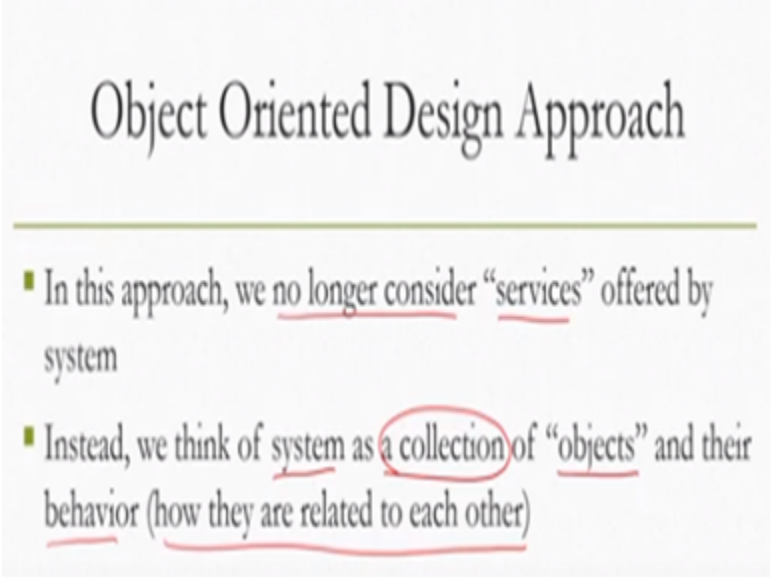
So in the case of the library management system then what are or what possibly can be the set of services represented as functions, let us try to enumerate a few important such services. One can be the service for registering a new user; so it essentially means that the system should provide a service of registration to the system for a new user. Similarly we can think of another service that issues a book, so this is a service of issuance of a book to a user.

Third service can check availability. This refers to the service of knowing availability of particular books so the system should allow the user to know which books are available for borrowing. There can be many such similar services that the system can provide and accordingly

we can come up with a functional design for the system. We can take; request the DFD notation to express that design.

Remember that in DFD we talked about processes so these processes are nothing but the functions that the system supports or rather the services that the system provides that represents those processes or functions.

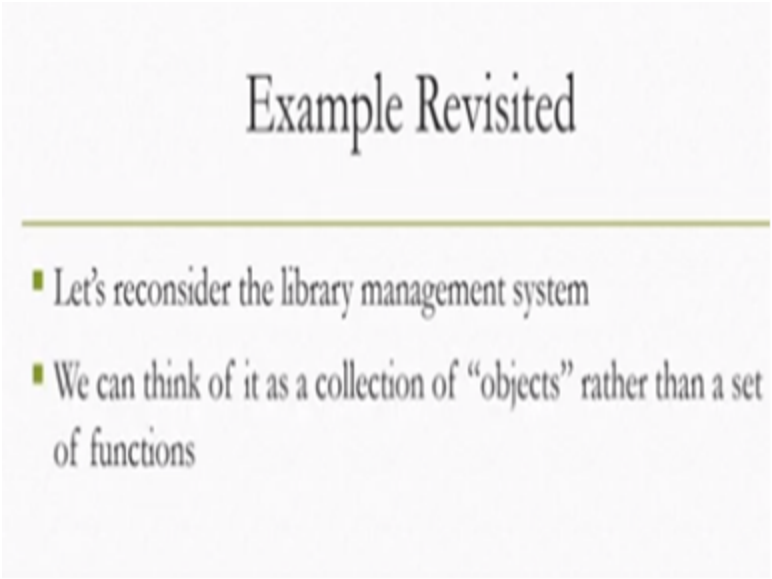
(Refer Slide Time: 17:37)



In contrast to this approach, when we try to go for; designing the same library management system using an object oriented design approach, what changes do we bring in? Let us try to understand that. So here in case of object oriented design approach we no longer consider services offered by the system, so those are no longer our primary concern. Rather what we do is we think of the system as a collection of objects and their behavior that means. How they are related to each other.

So earlier we are thinking of a system as provider of a set of services now we are thinking of a system as a collection of objects and their behavior or rather how they are related to each other, now this is the change in thinking

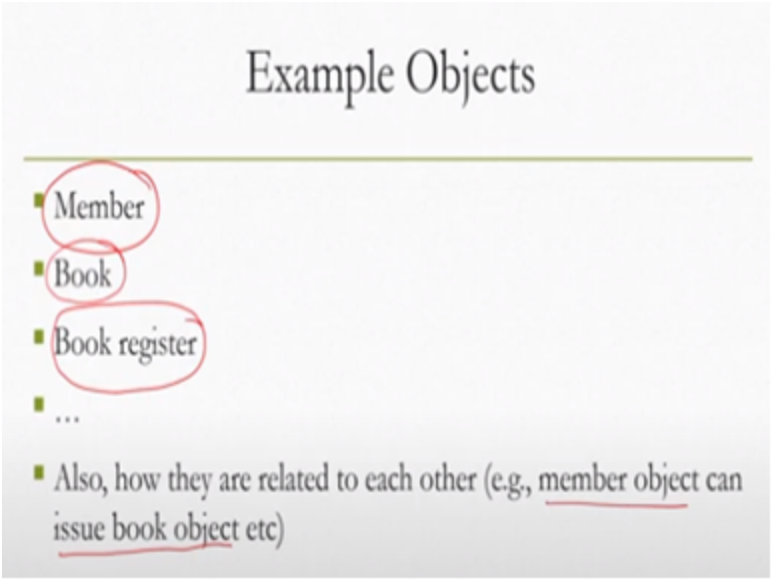
(Refer Slide Time: 18:41)



If we try to translate this idea into the design of the library management system, then let us see how we can now represent the system or rather the design of the system. So here now then we have to think of a set of objects that represent the design of the system not only a set of objects, but also how they are connected to each other, together they represent the design of the system. So earlier we thought of the library management system as a system that provides a set of services to its users.

Now we are thinking of the library management system as a system that consists of a set of objects and their behavior; that means how they interact with each other and how they are related to each other.

(Refer Slide Time: 19:34)



Let us see a few important objects that we can define for this particular library management system, one object can be a member object as the name suggests these objects refer to the members who are part of the library. Then another obvious candidate for an object would be a book, so a library consists of books so each book can be thought of as an object. So there is some register maintained to keep track of the books and their borrowing history so that a book registers can also be an object.

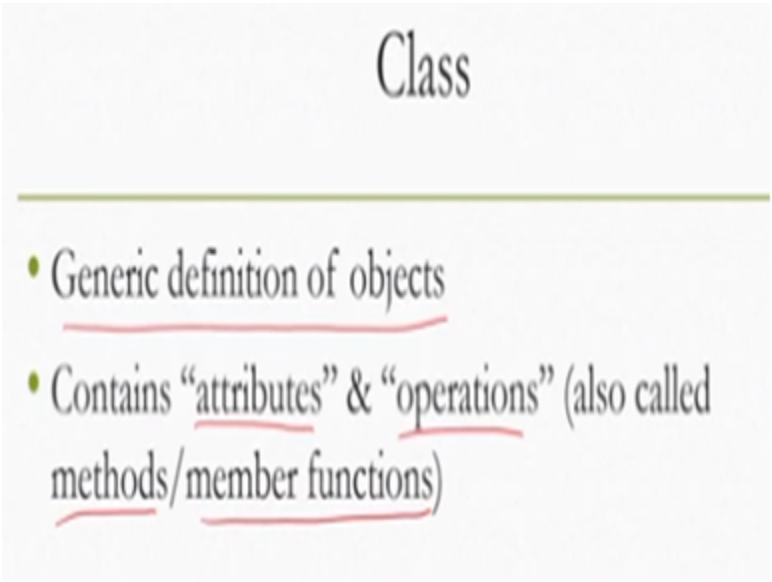
Similarly we can define other objects now not only these objects, how these objects are related to each other, that is how members are related to book objects, how book objects are related to book registers, how members are related to book registers; all these things we have to define or rather design. To come up with a complete design of the whole system, now if we are able to do that then what we are getting is an object oriented design rather than a functional design.

For example the behavior of a member object can be defined as a member object can issue a book, book object, similarly behavior of a book object can be defined as a book object can be issued by a member object, and a book object can have an entry in the book register object. So in that way we can define their relationship, their interactions and so on to complete the design of the library management system.

(Refer Slide Time: 21:31)

So I hope you got the basic idea of what is the difference between a functional design approach and the object-oriented design approach. To recollect in a functional approach we think of the system as a provider of a set of functions and accordingly we go for the design. In an object oriented approach we think of the system as a collection of objects and how they are related to each other and accordingly we go for the design. Now with that basic idea let us now try to learn in a little bit more details the idea of objects and classes.

(Refer Slide Time: 22:10)

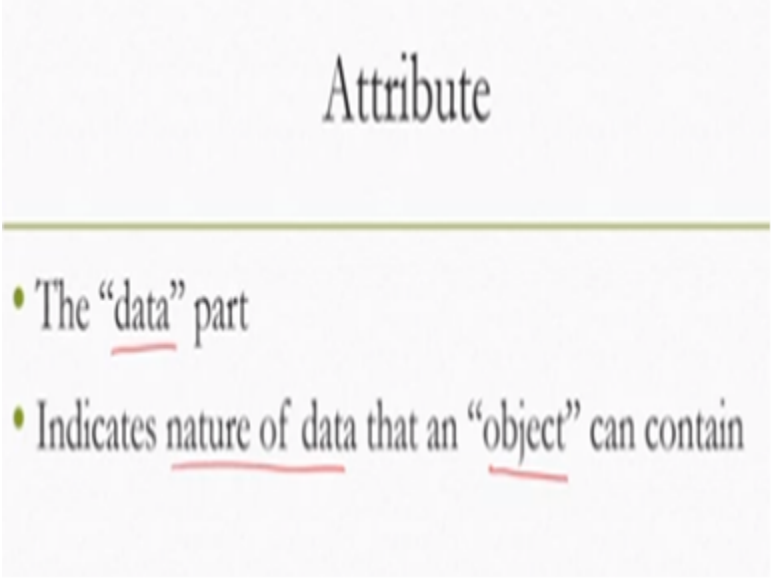


We will start with the idea of class, so what is a class whenever you are going for an object oriented design approach and going for implementation of object oriented design you will come

across this term of class. So class is a generic definition of objects that is the simple definition of a class. Now the class contains attributes and operations, now these operations are also known as methods or member functions.

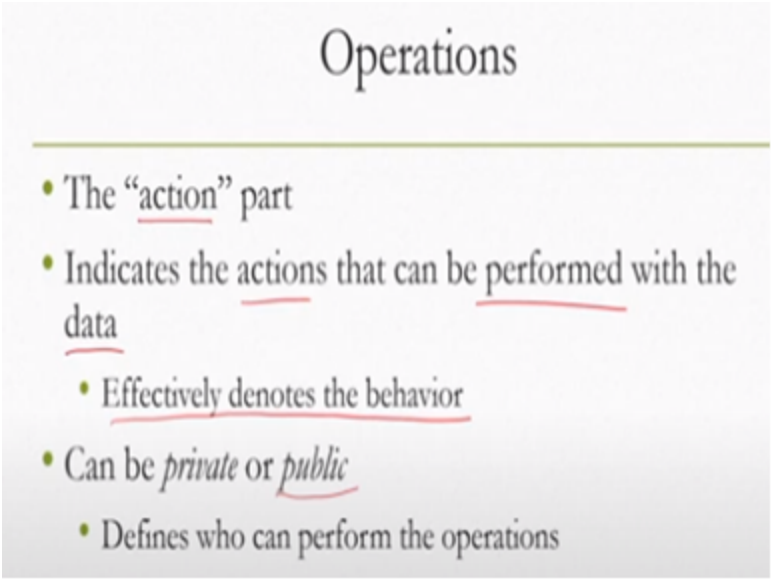
So there are several ways in which the operations are known. So in a class we generally have a component called attributes and another component called operations.

(Refer Slide Time: 22:58)



So what are attributes, simply speaking attributes are the data part of the definition of a class, so essentially it indicates the nature of data that an object can contain. So class is the generic definition of an object and class contains attributes as one component attribute essentially refers to the data that can be contained in the object, which results from the class.

(Refer Slide Time: 23:35)

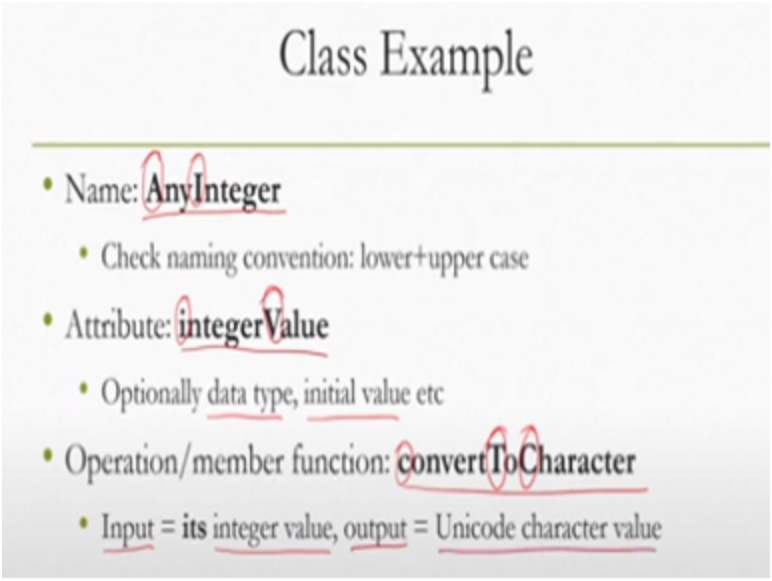


Operations are the action part so there is the data part and the action part is captured in the operation component of the class definition. Now these operations indicate the actions that can be performed with the data, so there are 2 things one is the data and other one is the operations that are allowed to be performed on the data. Now the data part is known as an attribute and the operations that are allowed to be performed on the data are known as operations or member functions or methods.

Now these operations or these actions that are allowed to be performed on the data essentially define the behavior of the objects that result from the class definition, so essentially the operations part defines the behavior of the objects. One important thing to note here is that these actions or operations can be defined as private or public. That means some of the operations or the operations that are defined as private can be performed by only a specific object and the access to those operations are restricted to the object, whereas if some operations are defined as public then other objects can also access those operations.

So in other words the private or public tag denotes who can perform the operations whether it can be performed by anybody or any object or it can be performed by only a specific object.

(Refer Slide Time: 25:29)



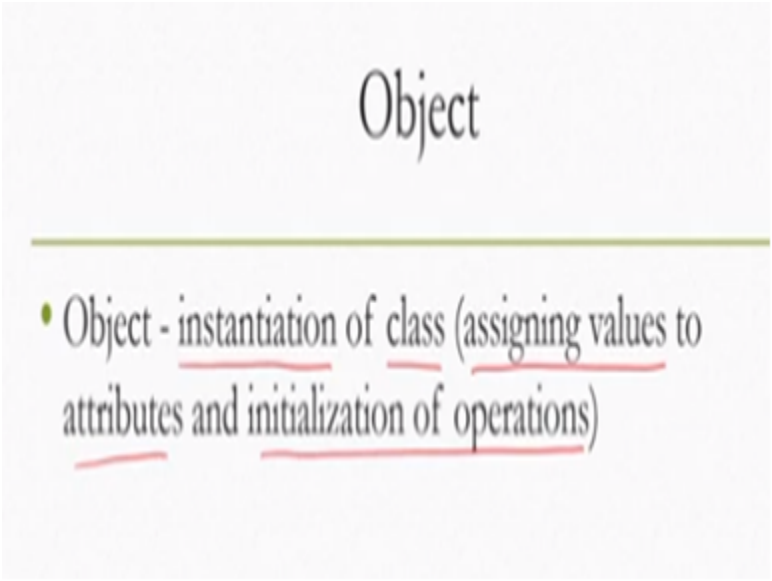
Now let us try to understand these concepts in terms of some examples, so let us start with one simple example that is let us try to define a class called any integer. So when we are defining a class name class and giving it a name some conventions are generally followed so if the name consists of 2 distinct words, then each word should start with a capital letter like shown here. If it is a single word then that word should start with a capital letter. This is a general naming convention and it is good if you can follow this.

Now this class can have attributes and operations so let us define an attribute to be an integer value which is an attribute for the class any integer. Now here if you again notice we are following one convention that is if it is a multi-word name for the attribute then the first word should start with a small letter, and the second word should start with capital letter. Now for this attribute we can optionally mention data type, initial value and so on that is also possible.

And let us define operations for this any integer class let us define only one operation which is converted to character, again multi word name. So we are following the convention. The first one is a lowercase letter starting with a lowercase letter but subsequent words start with uppercase letters as shown here. And if we are defining a member function or operation then we have to define its input as well as output.

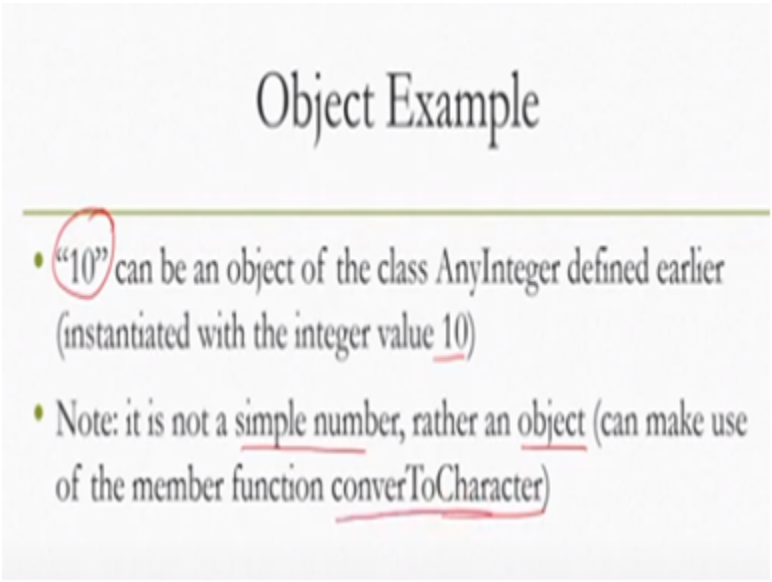
So we can define in this case the input to be its integer value and output is a unicode character value representing the character so that is the simple way of defining the class.

(Refer Slide Time: 27:48)



Now once this class is defined; recollect that we mentioned class is a generic definition of the object, so any integer is a generic definition of the objects that result out of this class definition. And when we instantiate the class with specific values then we get an object, so object is an instantiation of class. That means assigning values to the attributes and initialization of operations if required, so whenever we do that we get an object.

(Refer Slide Time: 28:32)

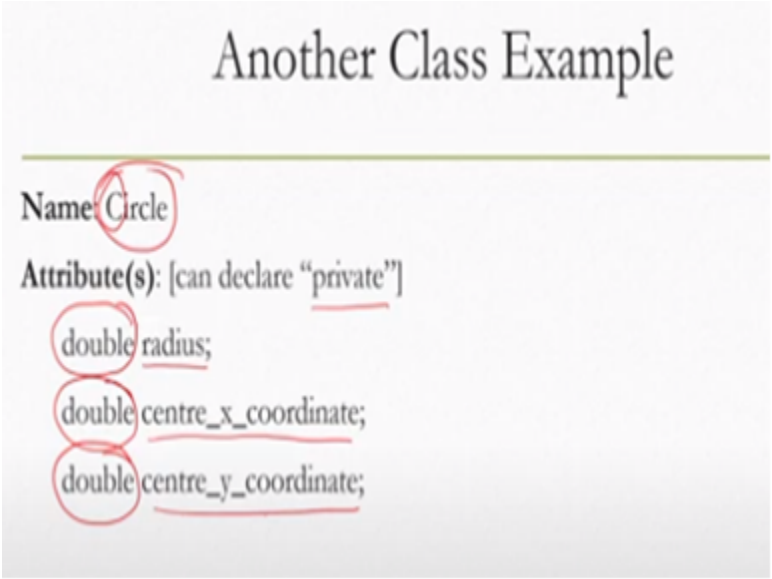


For example we can define 10 to be an object of class any integer here the attribute value is instantiated with or initialized with the integer value 10 and also we have initialized its member

function. Now in this case you should note that although we are using the number 10 in this manner, but 10 is not to be confused with a simple number it is an object that means to access its value we have to use its attribute rather than simply using this value.

And also we can use its member function to convert this object to a character value that is also possible by calling this member function convert to character. So these things we have to keep in mind that when we are defining something to be an object then simply using the object will not do. We have to use the attribute of the object if we want to get the value. And we have to use its operations if we want to perform some operation on the object.

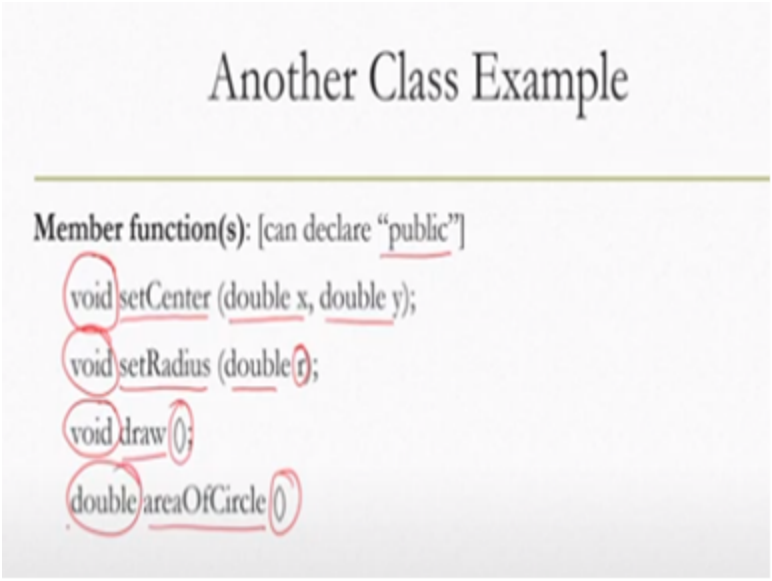
(Refer Slide Time: 29:52)



Let us see one more slightly complex example, let us define another class called circle; now here we are using a single word name so it starts with an uppercase letter and since there are no more letters so other letters are lowercase. We can define three attributes here now attributes can be defined as private or public like operations. So suppose here we are defining these attributes to be private so one is radius of the circle, note that here we have defined its data type that is double.

Similarly we are defining 2 more attributes one is x coordinate of the center again type double, y coordinate of the center again type double. So, all these are defined as private, which means only the object of this class can access it, others cannot directly access these values.

(Refer Slide Time: 30:59)

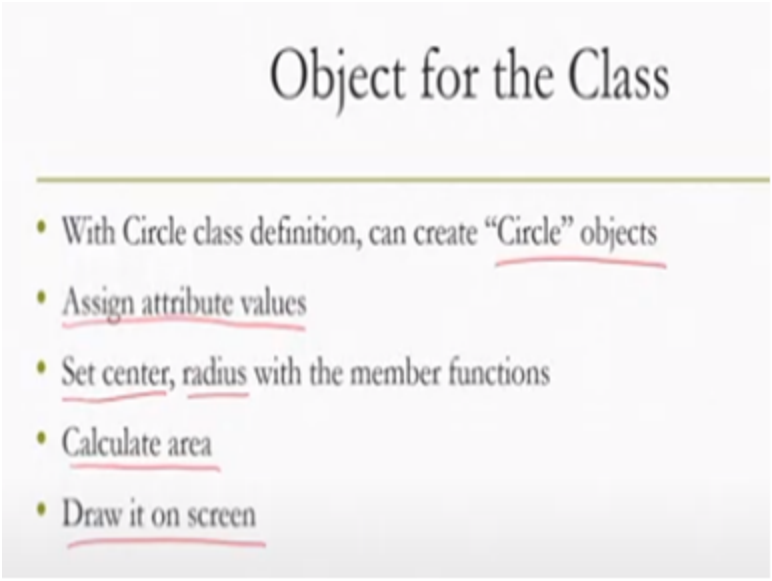


Similarly we can define its attributes as public also we can then define few operations or member functions or methods for this class circle. Let us assume that those are defined as public that means other objects can access these member functions so there are four member functions we can define. Set center as the name suggests this function is used to or rather this member function is used to set the x and y coordinate of the circle.

Accordingly it takes 2 inputs x coordinate and y coordinate both are defined as double data type and it does not return anything return type is void. Set radius sets the radius of the circle so it takes a radius value defined as double as the input it also does not return anything return type is void. Then we can define another member function as draw which does not take anything and does not return anything; it simply renders the circle on the screen.

We can also perform some computation like area of circle does not take anything but returns a double value which is the area value for the circle using the formula that we are all aware of.

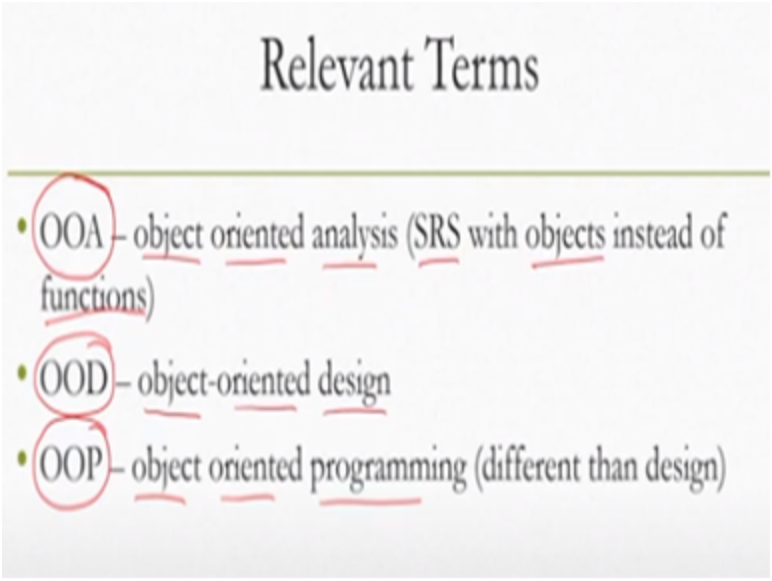
(Refer Slide Time: 32:38)



So then to define or to get objects for this class what; we can do we can initialize the attributes with some values and initialize the functions. So if we do that we will get circle objects each of these objects will have a specific set of assigned attribute values. We can also use the member functions to set the center of the circles, radius of the circles. You can calculate the area of the circles, and we can draw it on the screen using the draw member function.

So the idea is that circle class is a generic definition it; does not refer to any specific circle whereas circle object is a specific instantiation of the class. So if we say there is a circle having a center at coordinate values 2 and 2 respectively, and having radius of 2 units. Then that is an object of the class circle. Why it is an object is because we have instantiated the attribute values of the circle class with these values center x coordinate 2 center y coordinate 2 and radius value 2, hence we have created an object of type circle.

(Refer Slide Time: 34:03)

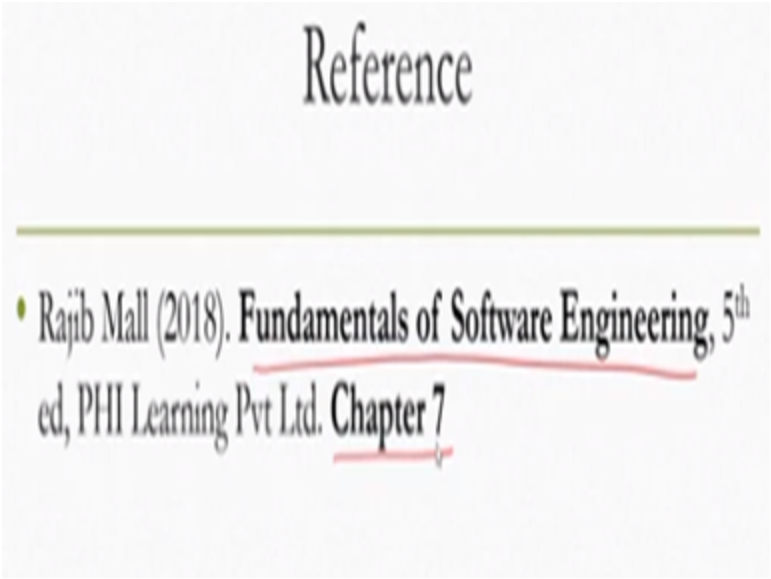


So that is briefly the idea of class and objects. These are the basic concepts that we are going to use in our subsequent discussion on object oriented design for systems. It is helpful if we know some relevant or related terms at this stage. So broadly there are three terms that you may encounter one is OOA or object oriented analysis which refers to creation of SRS where explicitly we mention objects instead of functions, so if we do that then we refer to that activity as object oriented analysis rather than simply requirement analysis.

Then we have this term object oriented design or od object oriented design also a very related term is object oriented programming OOP, or OOP object oriented programming. Now you may have encountered these terms before so it is always helpful to remember that object oriented design and object oriented programming are 2 distinct terms without any relation to each other. Object oriented design refers to the design approach that means designing the system in terms of object and object oriented programming is implementation of the system using object oriented programming languages.

It is not mandatory although it is generally used or followed. It is not mandatory that object oriented design always uses object oriented programming and vice versa. So today in this lecture what we have learned is basic concepts of object oriented design ideas of objects and classes. In subsequent lectures we will learn about; in more details how to use these concepts to go for design of systems.

(Refer Slide Time: 36:19)



Whatever we have discussed today can be found in this book fundamentals of software engineering fifth edition you can refer to chapter 7 of this book to learn in more details these concepts, that is all for this lecture I hope you understood the concepts we will continue this discussion in the subsequent lectures hope to see you in the next lecture thank you and goodbye you.