IIT Madras

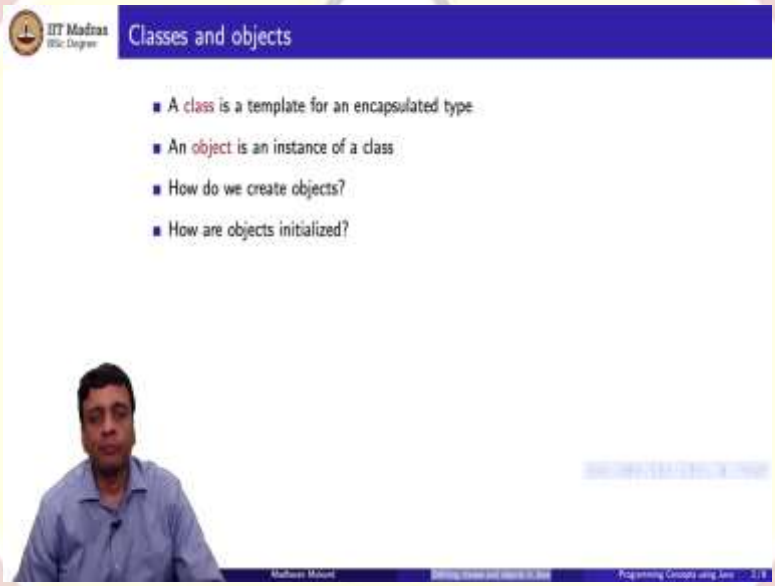
ONLINE DEGREE

Programming Concepts Using Java
Online Degree Programme
B. Sc in Programming and Data Science
Diploma Level
Prof. Madhavan Mukund
Department of Computer Science
Chennai Mathematical Institute

Lecture 10
Defining Classes and Object in Java

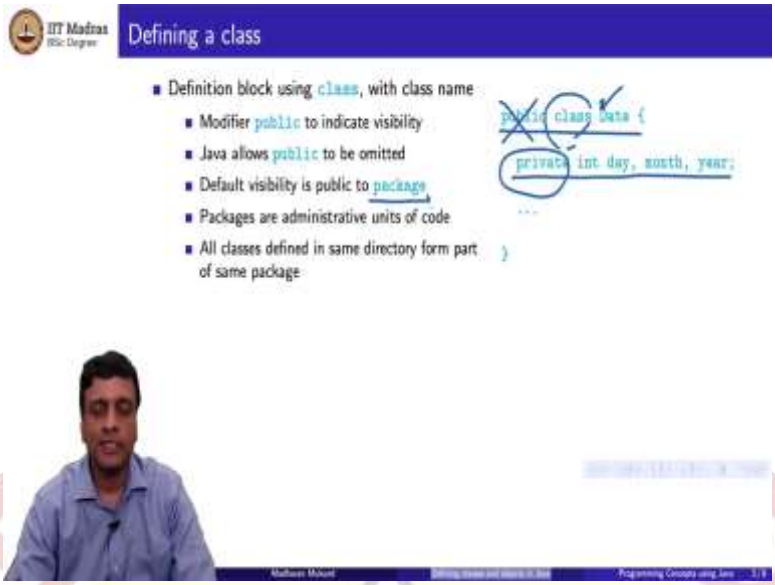
So, we have seen basic built-in data types in Java we have seen standard control flow in Java. So, now let us get down to business. So, object oriented programming. So, how do we define classes and objects in Java.

(Refer Slide Time: 00:26)



So, remember that a class is a template. So, a class encapsulates a data type the values that are stored in the data type and the functions that manipulate it an object is an instance of the class. So, how do we define classes how do we create objects and when we create an object what happens to the values inside the object how are they initialized. So, that is what we want to discuss.

(Refer Slide Time: 00:48)



So, we have already seen class definitions but the kind of class definitions we have seen have not been very interesting because they have only contained functions which are static. So, if we have a real class a class is intended to create objects dynamically and each object will have some local data in the form of instance variables. So, for instance suppose we want to store dates. So, a date consists of three parts a day month and year and lets for convenience assume that all of them are stored as numbers.

So, a date day will be something between one and thirty one month will be between one and twelve and here will be some 4 digit thing typically. So, at the top we have this date as the name of the class. So, we have this reserved word class which indicates that this is a class definition public of course indicates that this class is public remember that every public class defined must go in a file of the same name.

So, if I write this I must write this in a file called date dot java and the other thing to remember about Java is that like python capital letters matter. So, if I write capital date is the name of the class the file name should also start with a capital D. So, date dot java will contain this. So, here at the top this is defined outside. So, remember that every function in Java lives inside a class that is how we wrote the class for say the hello world and even some of the examples that we wrote to some of the arrays and all to illustrate the control flow.

But outside these functions you define these instance variables and for those functions which are out variables which are outside the function which are instance variables you

have to specify their visibility. So, here we are saying that these three values day month and year are going to be instance variables of date and they are going to be private that is for each object will have its own copy of this but it will not be visible from outside date.

So, now private is of course a useful thing to have public is a useful thing to have. So, is there a default. So, if you were very serious about object oriented programming you would assume that the default should be private everything is private unless you explicitly make it public. Now remember that there is a tension between what is theoretically desirable and what people are used to.

So, unfortunately when Java was created there was some public pressure in some sense to make it compatible or easy to use. So, the default is that if you do not say anything it is more or less public. So, private is something that you have to think about and write if you forget to write public it is not an error in Java if you forget to write this. So, you will see many examples of Java code on the internet or in text books where the word public does not appear and yet that code compiles.

So, what this means is that implicitly if something is not labeled as private or public it is kind of public. So, it is visible between what is within what is called the current package now the notion of a package is completely administrative it has nothing to do with functionality of the code it is just a way of organizing your code. So, it is a bit like libraries and so on or like putting your files into a folder or your mailbox into separate folders.

So, it is an organizational unit. So, if I do not say public explicitly then the visibility is public within the current package and what Java assumes is that all the classes which are defined inside say one directory or one folder in a system are all public to each other. So, the there is a kind of default package if you write something dot Java then all the other dot Java files in that same directory are deemed to be part of the same package.

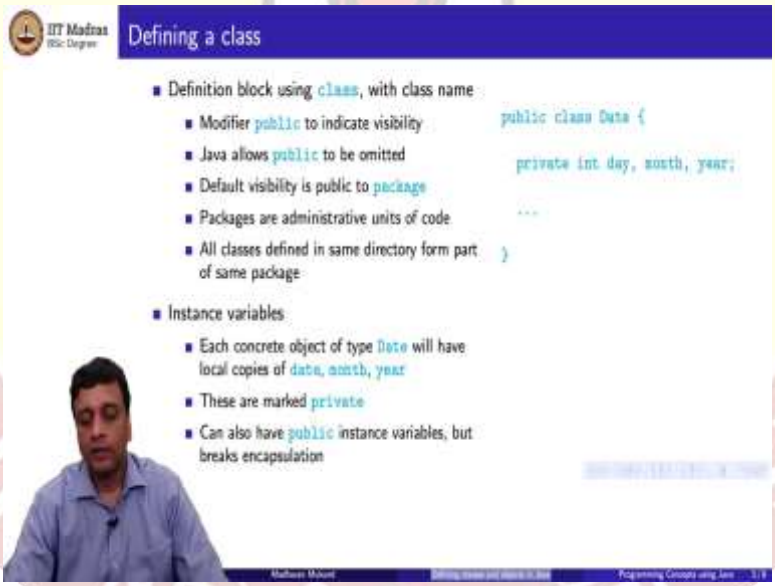
So, that is why you can get away with not writing public. So, again this is. So, we will keep seeing this kind of tradeoff. So, first of all we saw that Java has been has used these standard data types like intent float because it is cumbersome to make everything an

object. Then we saw that it uses this very strange C style for loop and switch statement because those are conventional and people are used to them.

So, here again because people might not want to be forced to write public and private everywhere remember the goal of object oriented programming is not to give you a choice the idea is that you can be a disciplined programmer without writing public and private this is what is expected of a python programmer. But the whole goal is to not give the programmer this choice to make it the compiler's duty to enforce these things.

So, that code can be type checked at compile time but still Java is forced to compromise and if at all once something should be default it should be the private is the default but actually public is the default. So, if you leave out public it is public within the package if you leave out private it is public within the package. So, you write explicitly private it is private otherwise you can assume it is sort of public.

(Refer Slide Time: 05:46)

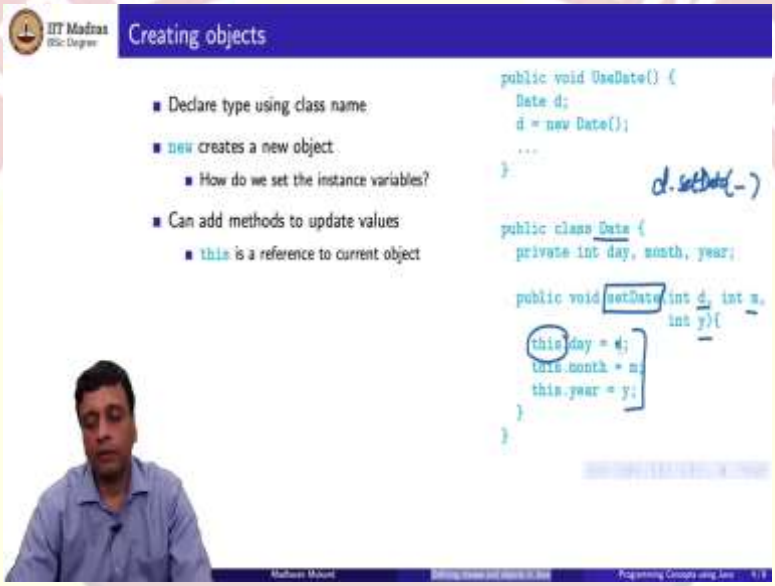


So, now these instance variables will be stored separately in every object of this type. So, every date object that we create will have its own copy of this. So, there will be multiple copies within that object they will be called day month in here but each of them will be distinct from others we can also have public instance variables as I said the whole purpose of object oriented programming and abstract data types is to encapsulate the data.

So, actually it is not meaningful normally to have publicly available instance variables because you do not want the internal data to be written or read from outside. Because if somebody is reading your internal data or writing your internal data then if you change your representation of how you keep the data then their code will break. So, you do not want other people's code to be affected by implementation choices in your code.

So, this is one reason to keep classes keep the instance variables private and only allow restricted access through public methods which you control.

(Refer Slide Time: 06:48)



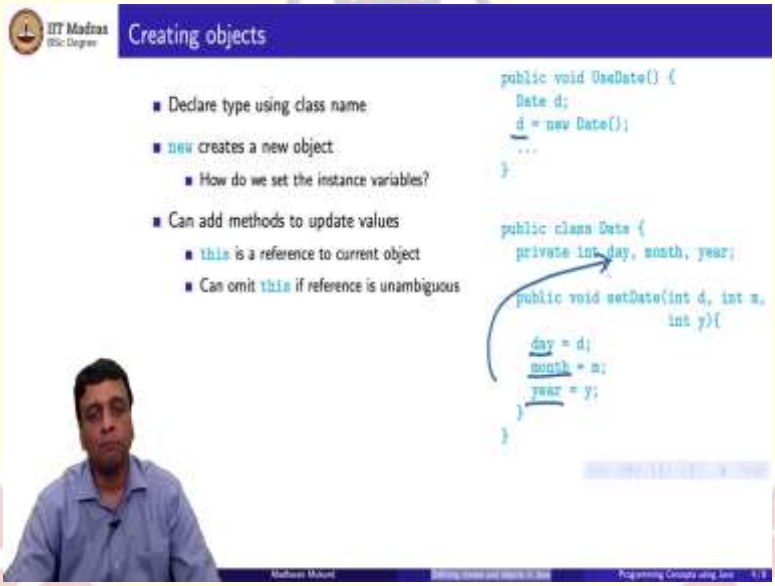
So, now to use a class we use the class name as a type. So, we can define a variable d and say that it is of type date where date now will be a class which Java knows about. So, if I write something like this in one file Java will go looking for a date dot Java file because it knows that there must be one such file available to me because there must be a public class which I am using if it does not find it, it will be a type error otherwise it will find it in that file.

So, this will tell it that there is a declaration here of date. So, there must be a date dot java with a definition of the class. Now I want to actually create. So, this does not create anything. So, this is we saw in the array case we just say int a this does not create an array. I have to say new something to create the array. So, the new actually comes from here. So, for any object we have to use this word new. So, we say day d is equal to new date.

So, this bracket we will see it is to do with setting up parameters but essentially you use the word new and the class name in order to create an object of that type. Now we have created an object it has some private day month in here. So, how do those get set. So, one way is for the class itself to provide functions to set it. So, for instance inside this class date I could have a function called set date which takes a day month and a year and it sets the instance variables to the arguments.

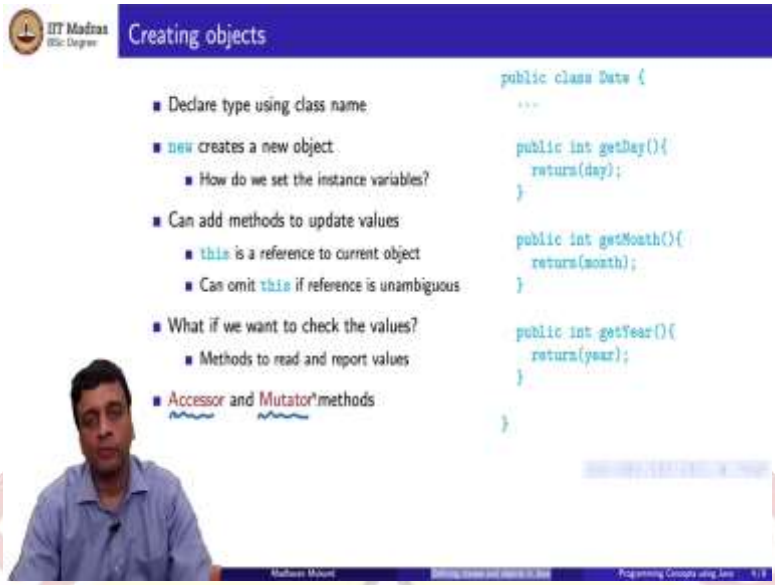
So, the word this here is specifying that we are talking about the instance variables for the current object. So, I am calling this I will say d dot set date something. So, this will mean that inside d it is referring to its own copies of this day month and year. So, it just used to disambiguate which day month and year we are talking about as we will see as we go along. So, this now sets this up and it will now allow us to update.

(Refer Slide Time: 08:51)



Now if there is no confusion about which one we are talking about I mean I am executing this say in the context of this date d that I created then I can omit this. So, if I do not say this and I use the name of an instance variable inside a function it must be my copy of the instance variable. So, this is implicit but sometimes to make the code explicit it is useful to use this.

(Refer Slide Time: 09:12)

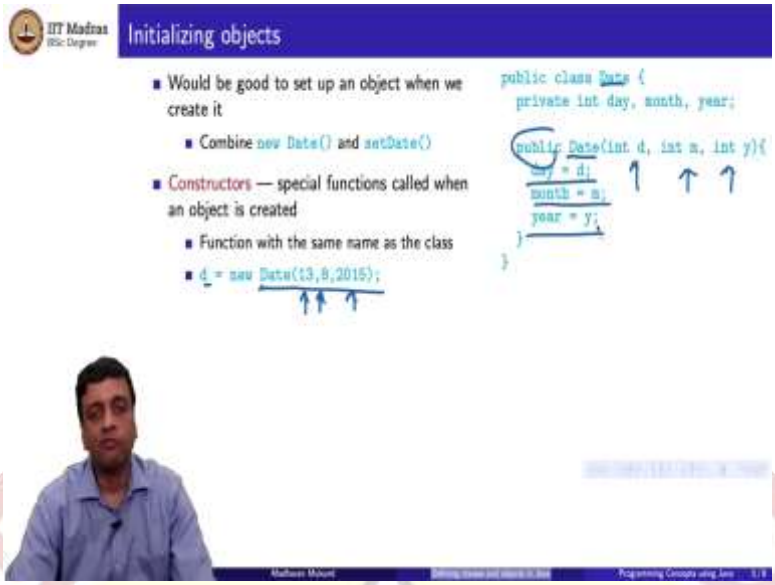


Symmetrically if I want to find out what is currently stored in a date I cannot look it up because it is private. So, I need a function which tells me. So, to avoid getting into complications of tuples and all that let us just assume that we have a function for each part of the instance variable. So, there are three integers day month and year. So, let us assume that there are three functions there is one function called get day which returns the current value of day inside this object.

We have get month which returns the current value of month and we have get here with the current value of year. So, these functions only report the current value they do not update it. So, broadly speaking we will have with any class which has these private instance variables we will have two types of functions to manipulate the data one which will update and one which will report.

So, it is conventional to call the ones that report accessors and the ones which update mutator. So, mutation is something which changes a mutator is a function which updates or somehow modifies the instance variable an accessor is a function which reports something about it may not report everything or it might report a summary or whatever but it tells you something about the instance variable without touching it.

(Refer Slide Time: 10:27)



So, this is of course a two step process I have to first create a empty date and then I have to call this set date variable to set it up. So, it would be nice to have a way of doing what we do with other thing. So, we say int i equal to 0. So, this is a declaration combined with an initialization. So, how would we do this for an object how do we how can we initialize an object when we create it.

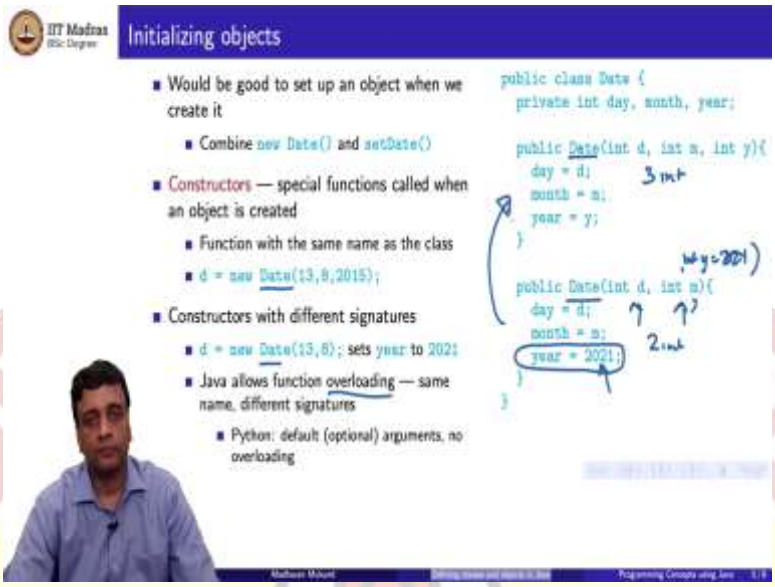
So, this is done through some special functions which are defined inside the class called constructors so in Java the constructor is given by the same name as a class if you remember in python the constructor is called init. So, there is a special function in python called init which is called when the class name is used to create an object in Java the convention is to use the name of the class itself.

So, this function does not have any return value its only purpose is to set up the values when the class is when the object is created you cannot separately call this function date at any later state. It is not a function available to you it is not a public function though it says public here is you cannot say at the later point d dot date something it is not this is not allowed. It is an implicit constructor which can be called only implicitly by creating it.

So, what you say is d equal to new dot date new date and then you pass as we know from python and the init function you pass the arguments that that constructor expects. So, the constructor here is written with three arguments and just like the set date that we had earlier it sets the instance variables to the parameters that it receives from outside. So,

this is a constructor. So, it is a special function called when an object is created cannot be called otherwise.

(Refer Slide Time: 12:19)



Now one interesting thing about Java is that it allows you to have multiple types of constructors here is another constructor where you just specify the day and the month and implicitly it sets the year to be 2021. So, this is something where one of the values is said to be a default value and only two values are provided. So, there are two different functions and internally also they have the same name.

So, technically what Java thinks of as the signature of a function is the name of the function its input parameters and its output parameter. So, here this has a this is called date and it has three integers whereas this is a date and it has two integers. So, these technically have different signatures. So, for Java these are two different functions. So, this is what is called overloading.

I can use the same name but I can distinguish between two different versions of the function by changing the parameter type. This is not allowed in python in python you cannot have two different functions with the same name and just have a variation in the. So, the way you can get around it in python is to define some default values. So, for instance you could say y is equal to 2021 here.

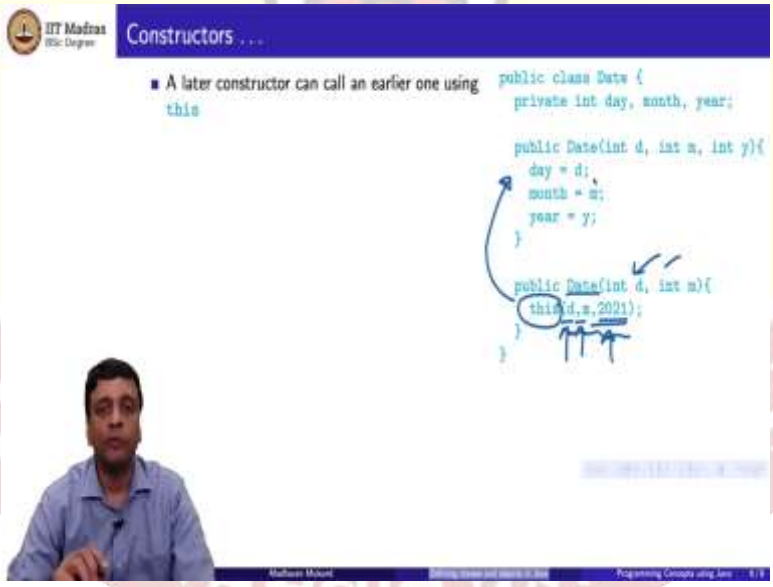
And then python will say that the last parameter is optional if I provide it I will use it if you do not provide it use 2021 but then this also fixes an order I mean. So, you have to

provide the optional parameters at the end and the non-optional parameters before and so on whereas Java allows is overloading. So, it allows the same. So, it hap here we have done overloading for constructors.

So, I have two different constructors now I have a constructor in which I provide the day month and year when I set up the object there is another constructor where I only provide the day and month and it implicitly provides the year is 2021. So, this is not within my control of course whoever designed the class chose 2021. It might choose 2011 it might choose 1954 whatever it is.

But I do not set the year the year is set implicit. So, this is overloading. So, the overloading idea is not restricted to constructor it can be for functions in general not allowed in python is allowed in Java. Now notice that except for setting year to 2021. So, this is very similar to that. So, in a sense this is a special case of the second constructor is a special case of the first constructor where the year is set to a fixed value.

(Refer Slide Time: 14:44)

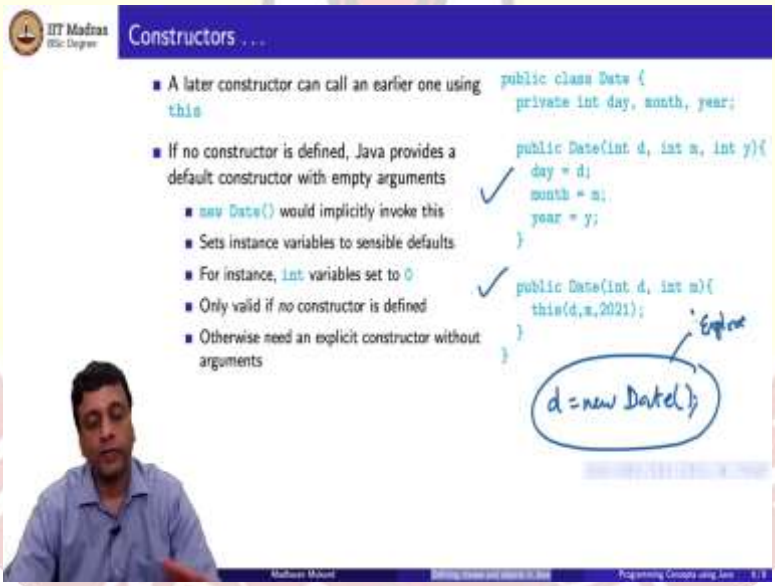


So, this can be actually reused in Java. So, what you can do in Java is you can make the second constructor reuse the first constructor by passing the d and the m that you get from outside and passing 2021 as the value. So, what we are saying is that inside this constructor when I use the word this I am saying look for another constructor with this signature. So, this here refers to myself.

So, I am saying this is like date but remember you cannot call date you are saying this is just a convention that you use this to say look for another signature with this look for another constructor with this signature and in that constructor I pass two values which I have got and I pass a default value of my choosing but the other constructor requires three values. So, it will correctly take these three values and set up the thing. So, in other words the second time I do not have to change the way the assignment to the instance variables is done I am just piggybacking.

So, I already have a way to do it I am just saying do it with this argument as a default for the third one. So, this is a reuse of code. So, if I change at some later point how I represent things then it will automatically be changed if I change only one constructor I do not have change every constructor. So, that is a general principle of reuse that if you change something then the change is automatically reflected everywhere you do not have to be careful about going everywhere and systematically making it consistent.

(Refer Slide Time: 16:09)



Now if you do not provide a constructor at all you just write a class and you do not provide a constructor then Java actually gives you a different. So, it is not obligatory to write a concept if I do not write anything then if I say d equal to new date this will still work and what this will do is that Java will set default values sensibly. So, this is one difference between Java and other languages.

In most languages when you declare something even python until you assign a value explicitly it has no well defined value. But in this setting of Java objects in general and

this is even true of arrays. If I create a new integer array in Java by default it is initialized to 0. Now it may not be a great idea to rely on this is probably better to initialize it yourself but this is done.

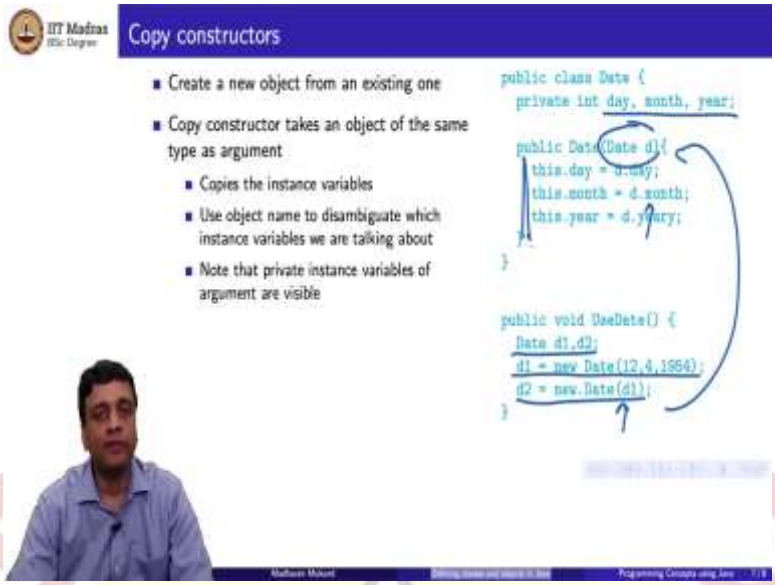
So, similarly here if you do not provide a constructor then Java will set up this object but not with random values but with some specified values in particular sensible defaults. So, typically numbers will default to 0, Booleans will default to false a string you would think would have default an empty string but it may default to the value null which is the equivalent of an undefined object.

But in general this default assignment is done for at the object level. So, you can get away with not defining a constructor and doing this but if you have defined these constructors. So, supposing you have these constructors and now I do this then what happens well then Java will not forgive you because you have defined constructors. So, the default constructor is not available to you.

The default constructor is available to you only if you have defined no constructors at all if you have defined even one constructor and then if you call it like this you must provide an explicit constructed with no arguments otherwise it is an error. So, when you call of new the way you call it must match one of the constructors if it does not match any of the constructors it is like calling a function that is does not exist.

And if you expect to call it with no arguments you must either provide a construct with no arguments or provide no constructors at all in which case you will get a default construct.

(Refer Slide Time: 18:25)



So, this is to create an object from scratch. I create an object and I want to populate its values from scratch what if I already have a date and I just want to copy it into another date I want to create a new date which has the same values as another date. So, this is something called a copy constructor. So, here again you create a function whose argument whose name is the same as the class but now the argument instead of being the same type typically as the instance variables is another object of the same time.

So, I will create a date using another date and what will I do I will copy the instance variables from the other date. So, remember that if I have an instance variable inside an object I use the object name dot the variable name. So, I am looking at the day month and year inside d. So, d dot day d dot month e dot here and I am going to copy it to this dot month this dot d and this sort here.

So, I am going to copy the instance variables from there to here. So, I am basically taking an existing date d and creating it. So, how would you use this typically well you would create typically two variables d1 and d2 initially you create one date by concretely providing a date. So, you say that I want a date which has 12th of April 1954 then I say I want a new date which has the same values as the old date.

So, I pass the existing date d1 as an argument to the constructor and it will automatically call this constructor because this type matches I have a date d there and I have a date d here. So, one thing to keep in mind here is that remember that these variables are private

and yet we are accessing them from another object. The crucial thing is we are accessing them from another object of the same type one date is looking at another date.

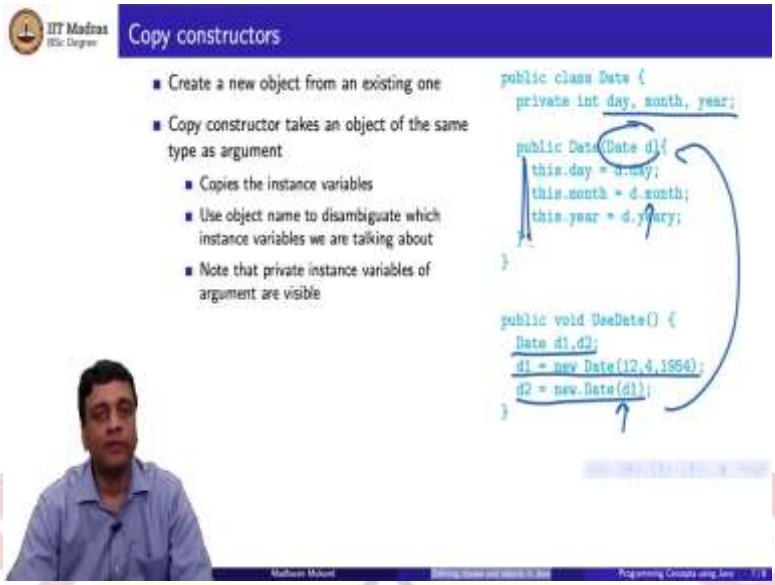
So, within objects of the same class privacy is not maintained. Now this does not we will discuss this later this does not create a problem why is privacy and public an important thing. So, you want to keep the implementation private. So, that other code does not change when you change your implementation but if I change the implementation of date then every object of type date is also changing.

So, I can update all parts of date which affect this together. So, we talked about this at the beginning we talked about this object refinement. So, we said that when you have a bank account then if you want to change the way the transactions are reported you have to make a bank account more elaborate. And if you did not have it as a class then you would have to separately check about the functions.

But if the all the functions that one manipulate blank accounts are in that class then automatically when you change the representative data you will also change all the functions. So, similarly if I change the way the date is represented in one place then this object constructor will also change. So, it is not really it makes sense to say that private variables can be seen within a class.

And we will see other examples of this but just remember this that public and private refer to outside the class not just outside the object other objects with the same class can read the private instance variables for other variables of the same class.

(Refer Slide Time: 21:35)

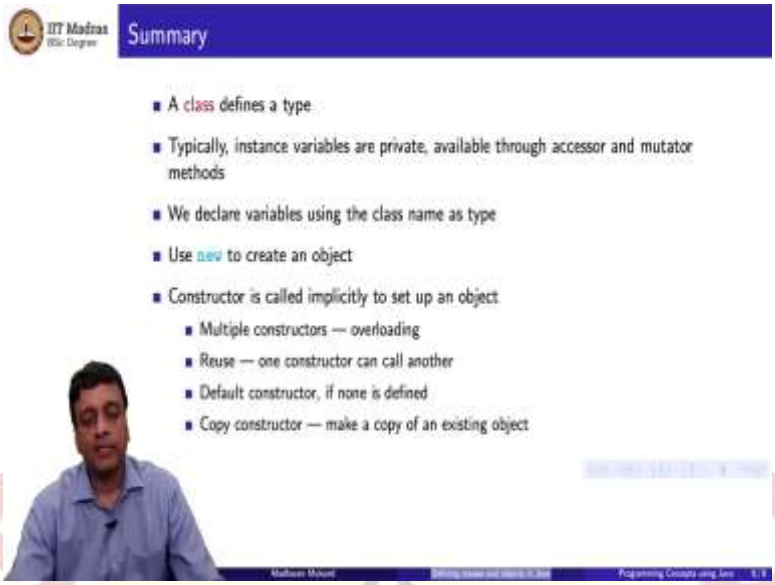


So, another subtle point here is that we are copying these things. Now this is the equivalent in python of copying immutable data but remember in python we saw that if you write l1 equal to l2 they are both pointing to the same list and the same happens in Java also if I write d one is equal to. So, I could have just said d2 is equal to d1 here. So, this would have been fine but then d2 and d1 are not two separate objects d2 and d1 are two names for the same object.

So, at the high level I avoid that by doing a copy construction but what if these in turn were objects then these two will be pointing to the same object. So, I have a problem of this, what is called a deep copy. So, superficially I copy the top level but inside we are pointing to the same object. So, if I manipulate something in d1 it will also get change in d2 and vice versa. So, we will come back to this later on.

So, there is a general issue about cloning an object if I want to take an object and I want to make a copy of it which is disjoint and does not interfere how do I do it.

(Refer Slide Time: 22:34)



So, to summarize as we know a class essentially defines a type it is an encapsulated type with the variables and the functions that manipulate it. Typically we expect the instance variable to be private and then they should be accessed through suitable methods which allow you to read and write them. So, these are called accessor and mutable the mutator methods and then we declare variables using the type name as the type I mean the class name is the type.

So, we write date which is a class name and d and this tells us that d is going to represent an object of type date and then we use new to create the object and when we use new, we would like to populate it. So, for that we use constructors. So, constructors can come in multiple forms and this is generally allowed by Java something called overloading. So, if the signatures of two functions are different and this could mean the same name but different arguments they are different functions.

So, you can use the same name to denote different variants of a function one constructor can call another. So, you can reuse the code and minimize the problems that happen with updates. If you do not provide a constructor you get a default constructor which takes no arguments and sets sensible values for all the instance variables. And finally you can use a copy constructor to make a copy of an existing object but you have to do the copying.

So, here for instance this part of the code you have to decide how you are going to copy. So, this issue of shallow copying versus deep copying is in the hands of the implementer it is not something that Java is going to guarantee.