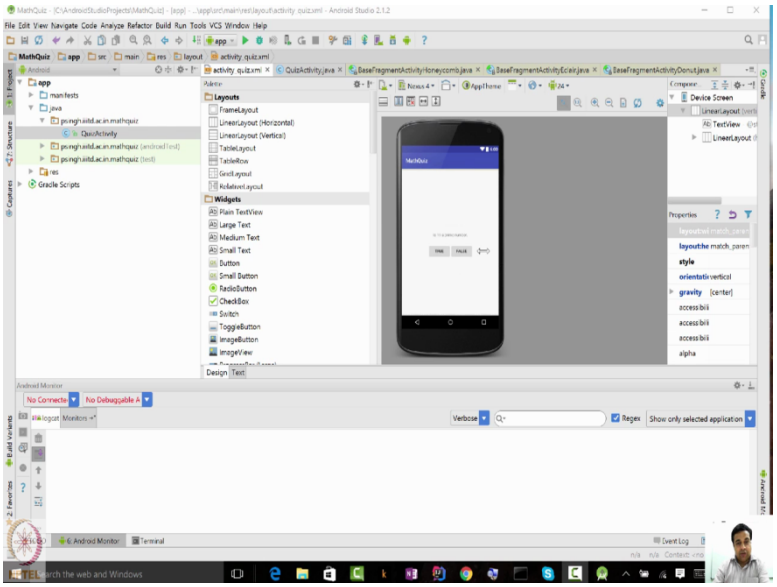
Mobile Computing
Professor Pushpendra Singh
Indraprastha Institute of Information Technology Delhi
Lecture 13

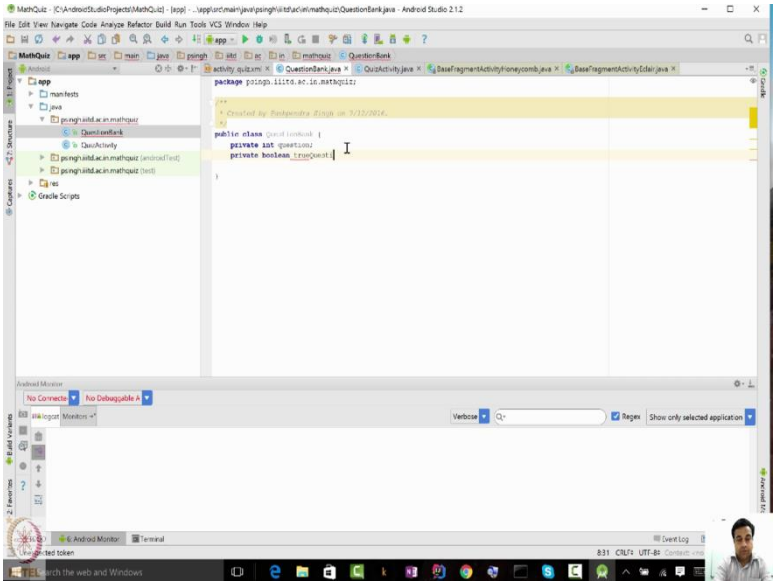
Hello, last class we created an application which displayed us a mathematics question. When we pressed one of the buttons, it also displayed us what was the correct answer. In this class and in couple of next lectures, we will try to extend our very simple application to a slightly better application by elongating the functionality to ask us more questions. In this class we will also briefly discuss what is called a model viewing controller design better or in short MVC design better. So let us get started, please start your Android Studio now and open the older project.

(Refer Slide Time: 1:14)



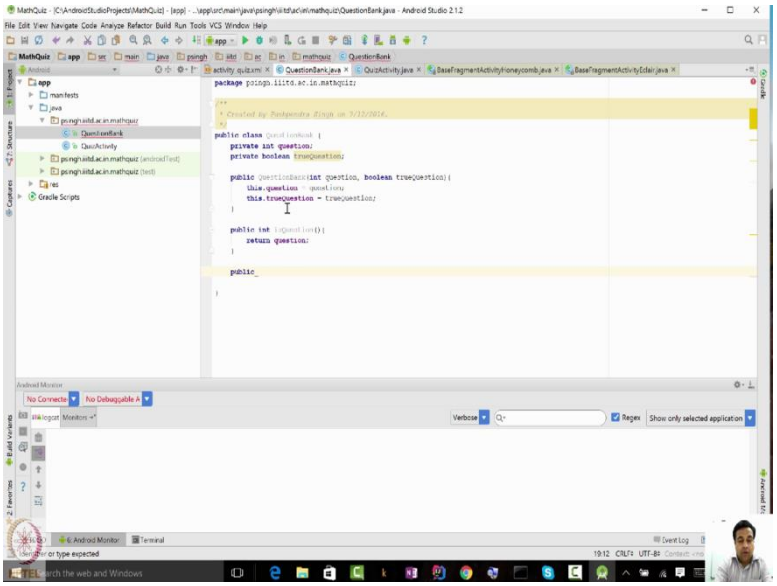
I hope you have now stared your application. As you can see, that our previously built application is already displayed here. Now if you want to ask more questions we will have to add a new class to our existing program. I am going to add a new class and to add that I directly choose the package in which I am going to add, I right click choose new and go to Java Class. I have named this class as question bank. As you can see Android Studio is building the class has out it in right package. Now the next step for me is to declare some private variables. Let me declare the first variable which I will call as question, I am declaring my question to be of type 8. I will declare second variable which I will call true question, true question.

(Refer Slide Time: 2:35)



Now in the beginning of the lecture I told you that we will be developing a class which shows different questions, and as you saw earlier that the question was a string, however so far I have declared question as an end. Could you please make a guess why I have done that? If you remember, we have described the resources and the resource ids. Each resource is tied with a resource id and using the resource id we can use the resource in our program.

(Refer Slide Time: 5:02)



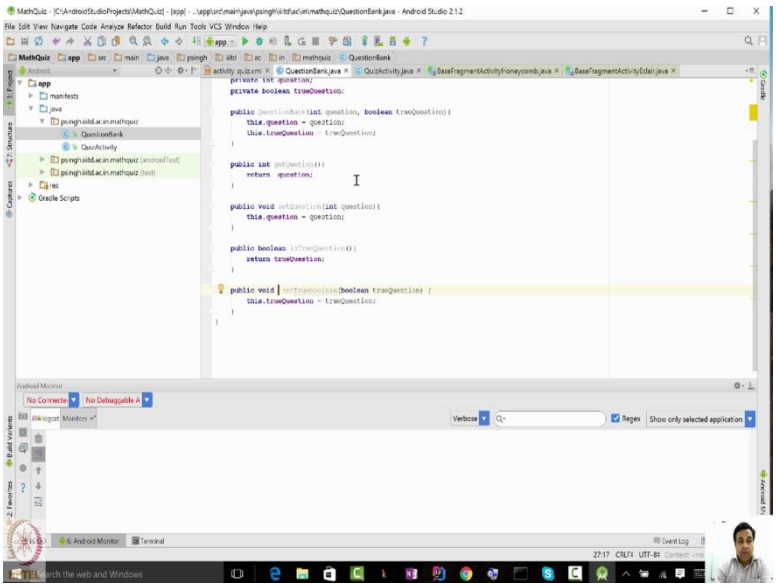
Our question which is an end will act as a resource id for different question strings. The next step is to declare a very simple constructor for our class. As you already know the constructor needs to be public method and it starts with a name same as the class name. I would like to initialize both of my private variables inside my constructor. My constructor is now complete and in my constructor I am passing two parameters which initialize my variables. The next

step is to write what we call as getter and setter methods in Java. The getter and setter method simply return us the value of the private variables or set the value of the private variables using public access methods. A getter normally starts with “is” and while a setter starts with “set”. So getter for 8 question will be called public 8 is question and or will do is return the question. While the setter will be called set question and will set the question.

Now we will describe what is called getter and setter questions. Getter and setter questions are used to access private variables. A getter returns the value of the variable while setter sets the value of the variables. So let us first write a getter setter method for our variable question and then for true question. A getter normally starts with get as you may have guessed already and it simply returns the value of the question.

While setter starts with set and simply set the value (())(7:07) error because I have given the wrong type of question, the right type of question is 8 and now my errors. Now, you may wondering that why I am writing getter and setter method and why I just do not declare my question and all true question to be publically accessible variable by using a modifier public. As I told you earlier that Java has this concept of encapsulation which hides the implementation details. I would now like you to know what kind of variables I have in my class.

(Refer Slide Time: 9:29)



Instead, I will provide you access to those variables only using public available methods. This is also lead to change the implementation of my class whenever I want as long as I support the publically accessible methods. Now let us slide the getter and setter for another

variable true question. As you may have guessed that I am writing first a getter method. For the Boolean variable we do not start with get but we start with is, which is a more natural form for a Boolean because we are asking a question is, it true or false and then we return the variable.

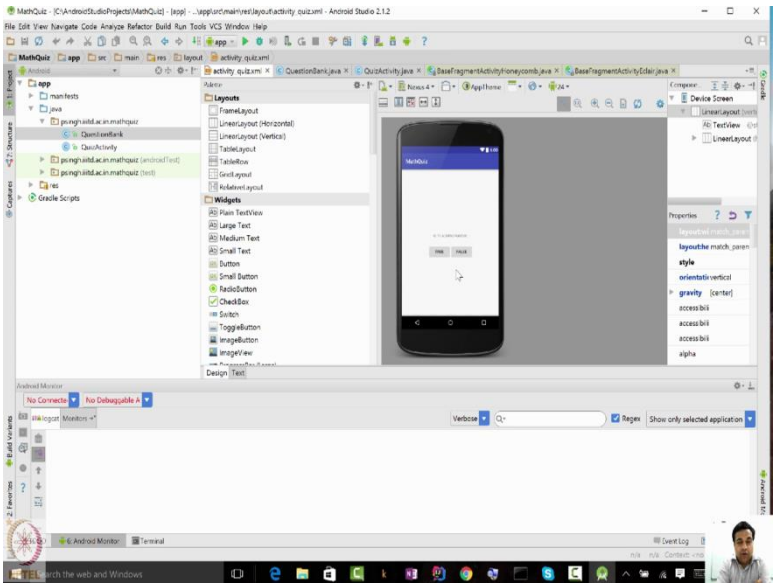
Setter is similar to any other setter and it starts with a set. Please note that the getter and setter and the terminology is only a convention. You can write the method with any name and it still assign the same functionality. However it will help other programmers to understand your program better if you use a standard terminology. Question I need to do a summary in, Oh I made a mistake I should have created a void which I did not.

Now my getter and setter functions are ready and my new class question bank is ready. What we are going to do in our program now is we will be creating an array of question bank objects and once we have this array of question bank objects, we will be displaying these questions using our original task quiz activity. What I am trying to use here is something called a model view control. While I will be discussing what will be controller in great detail in another lecture, I will give you a brief idea. The model we control a design pattern allows our application logic to be divided into three categories, models, views and control.

The model hold the application data and the business logic. While the few knows how to drop and display and the controller connects the model and the view using the application logic. So in our example, our question bank class will be the model, our quiz activity class will be the controller and our buttons which we have on our application will be the field. In a separate lecture I will be describing model view controller design pattern in great detail, it will explain more intricacies of the design patterns.

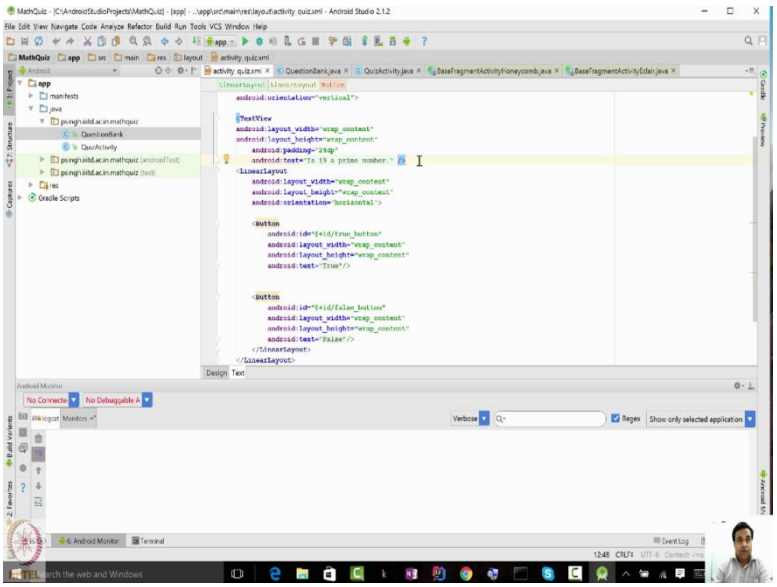
However I would mention the simple benefit of model view controller. Modern view controller allows you to separate the functionalities. Tomorrow you may want to just chase the view to display your logic in a completely different manner or you may want to just change the controller by modifying your application logic or you may want to just change the models to show a different business logic while keeping the other two components same. Now let us get back to our class. Ok now our question bank class is ready, we will now try to update our view.

(Refer Slide Time: 12:39)



The first step that we are going to do is to add a next button. We already have buttons for true and false, we would like to add a next button so that we can go to the next question. Let us see how we do that. What changes do you think we must be making to our fields? Number one, now that we have more than one question we will be using android text not as fixed hard coded string. Number 2, in order to do that, we will have to create a resource id for our question, and number 3, we will have to add a new button which will take us from one question to another question.

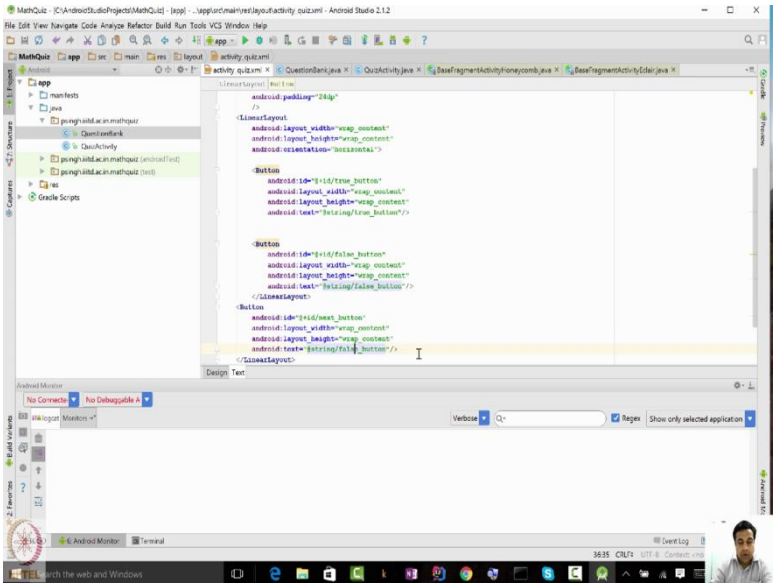
(Refer Slide Time: 13:26)



So let us start working on our view file. So in the text view we have the fixed hard board question let us first delete it, that is fine. Just like we added the resource id for our buttons we

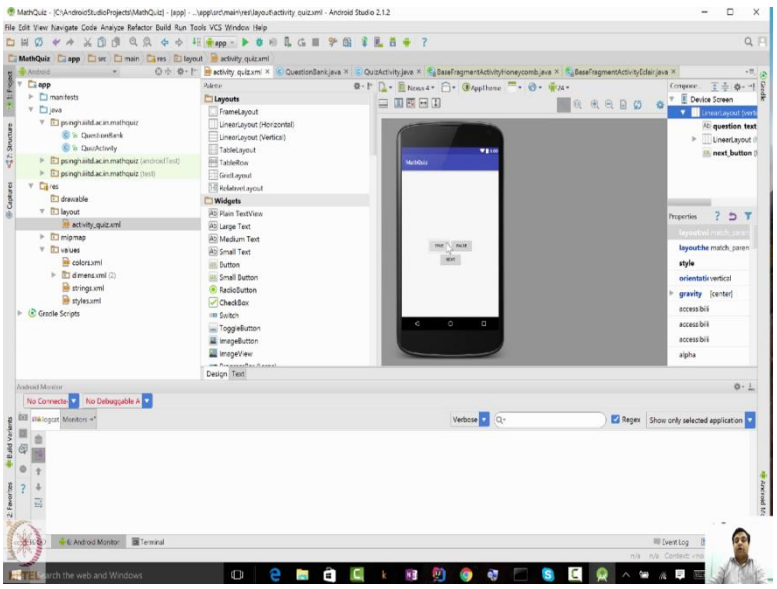
are going to add resource id for our string that we will use to ask the question. Apart from these two patterns we are going to add another button which we will call as next button. Let me copy, previous button code just make modifications into the next button.

(Refer Slide Time: 15:04)



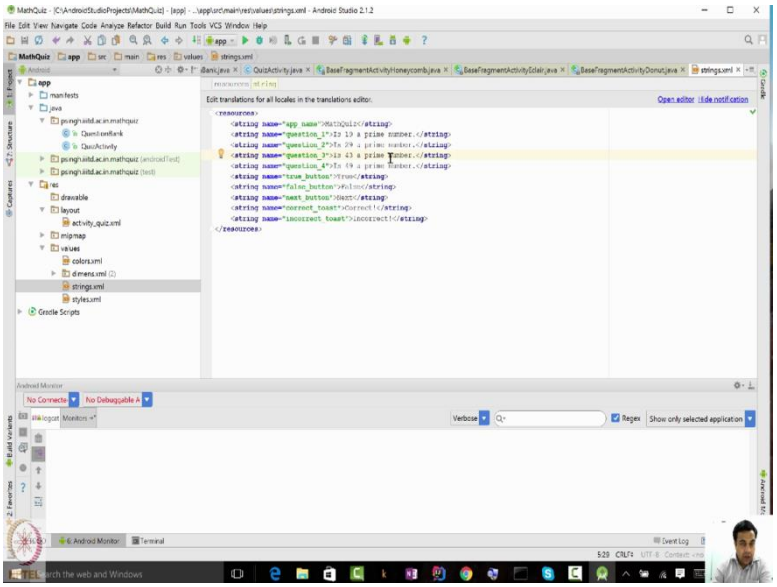
Number one that this button would be called Next button, other two are same and we will just change it to next. I think, you can see that the android studio is showing an error because it cannot resolve the symbol because so far we have not created a string which is called next button, please ignore this error as of now. The next go back to our rest folder. (())(15:43) the string and add the next button.

(Refer Slide Time: 16:31)



Now display next stated, let me go back check my error is now, my error is now gone and I can see next button, my fixed question is also gone. So instead of one question in a class span we will now ask 4 or 5 questions.

(Refer Slide Time: 17:37)

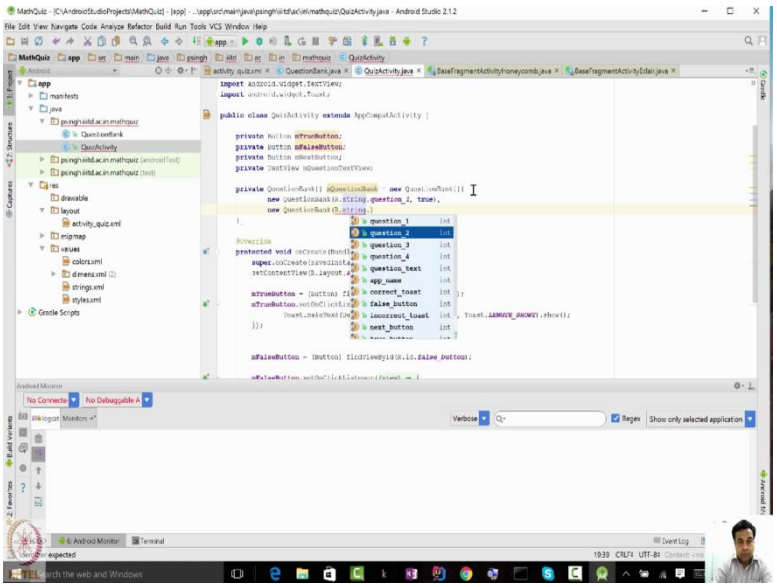


Let us go back to our string class and start changing of questions. Let us say my first question remains same, I will just copy it 3 more times, and create 3 more questions. Question 2, let me ask, let me ask only 3, let me ask for 49, change the string it is, save my strings time and I hope everything is ok. Go back everything looks just fine.

Now after updating my view file I would like to update my controller file which is the quiz activity file for us. As we saw last time in the quiz activity we created two buttons and we set what happens when those buttons are clicked, now first we have created now 3 buttons and now we will lead to set up what happens when we press the third button. The sum is that unlike the last time, where they were displaying a simple question this time they will be displaying different types of questions by pressing the next buttons. We will be making multiple changes to this file.

We will be adding different variables and we will also be adding 4 extra buttons and then we will also create an array of questions with an index we will try to access those questions. Let us start working by first adding an extra button. As you saw that for Text view we are getting an error, then press alt and enter in Android Studio and it imports the right file, right type ok now let us move on. We should try to write some more code to make sure that we can have multiple questions.

(Refer Slide Time: 21.12)



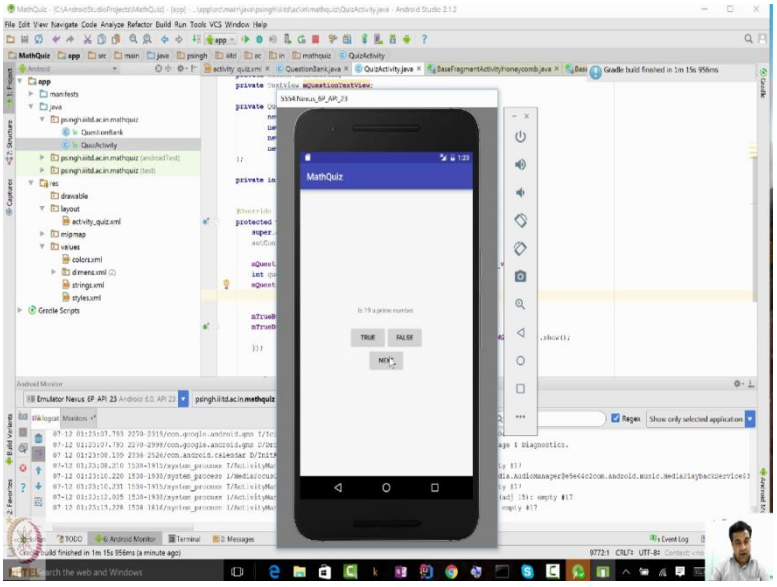
Now (())(21:14) now on I can give it a size that initialize the (())(21:21) or because I know that my array on these 4 elements I can try to initialize it directly here, without giving it a size because I know the size will be set to the initialization that I do here. Let us try the second approach now and try to initialize the array without giving the size exquisitely. Do you remember what our questions were? Well first let us think about question number 1, I am accessing question number 1 string by giving the resource id for question number 1 string.

The question number 1 was, is 29 a prime number? Answer to that is true. Question number 2 was also true so was question Number 3. However, question number 4 was to asking is 49 a prime number which is not true. So, now I have described all my array elements, my array is initialized with size 4 of type question bank. It may also describe another variable called current index. Current index will refer to the index of the array that we are trying to access. There are multiple ways to define model data. In this example we are using a very simple method. Later in more advanced application you may be seeing that the data is stored separately. For the time being, let us keep it simple and just create the array within the control.

Now let us go and create our text view, now let us start working with our end question text view. We will declare the text view earlier with the value and question text view. We are now come trying to initialize it. As you have seen that in our activity maths.quiz.xml file. We had removed the simple text view, we had created the text view with the resource id, and here is the id that we have now used. We are also using the current index to get the right place in the question bank and get the question.

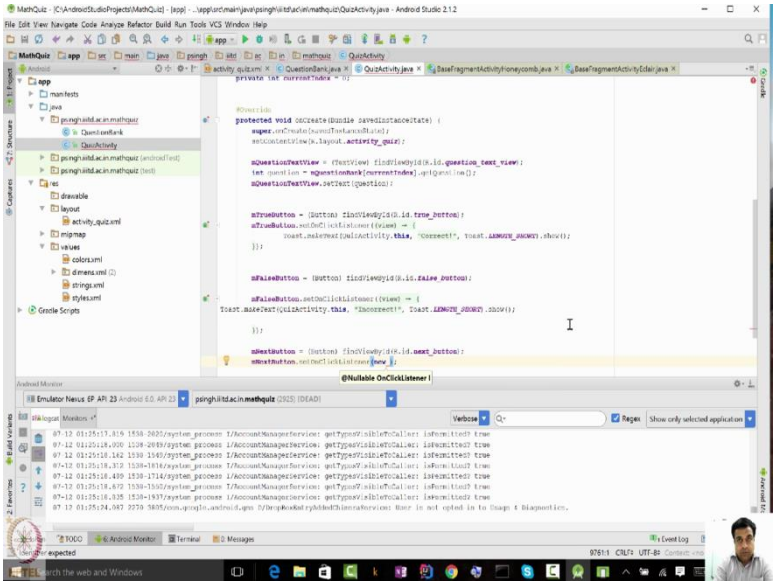
(())(27:15)are now, oops I believe that the previous file I have made a mistake, let me correct it, question text file. Ok so we have set up our text view, we are taking the right question, get it. Oh, let us just set the text, this question. This completes our code to display the right question, save our file and let us just run it. I am going to run it on emulator (())(No Audio 28:28 to 29:05)

(Refer Slide Time: 29:41)



Ok as you can see here that we have three buttons and we have our first question displayed. If we press next then nothing happens because so far we have not added any functionality to the next button. The next step in our application development is to add some functionality to this text button. Let us go stop the app and start changing our quiz activity file again. First step, strike last line, get the Id, then I will have to set the behavior of what happens if next button is pressed.

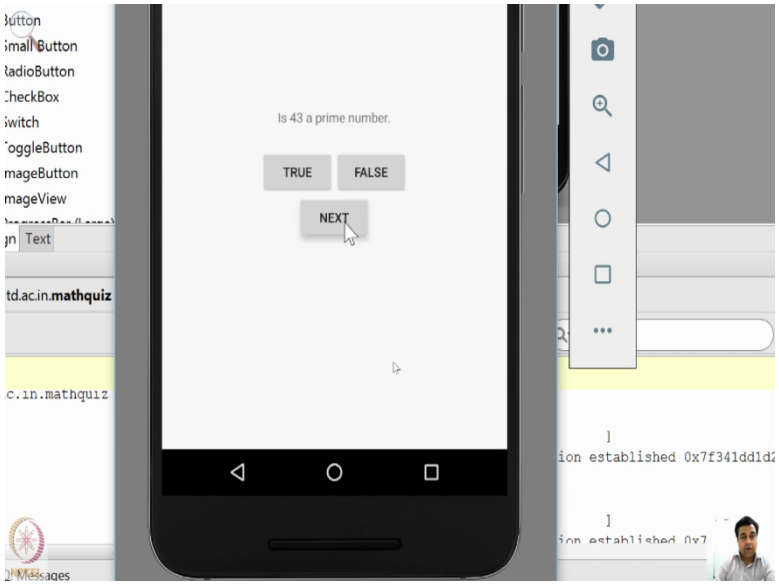
(Refer Slide Time: 31:20)



What do you think I should do here? Well we are going to increase the index of the array so that our pointer moves to the next element in the array and displays us the next question. We cannot simply increase to 1, because we have to come back when the array has been fully traversed. This we achieve by using following line of code. Now my array will go 0, 1, 2, 3, 0, 1, 2, 3 as the code progresses. (())(no audio from 32:43 to 33:24)

This completes the function for my next button. As you noticed that this piece of code is exactly same to this piece of code. Whenever we are in such a situation it is good to create a private method and then call that method. Let us create a private method called upgrade question. Copy this code there and then call update question whenever we need this piece of code, just copy it. The cells symbolize the code and also makes it easy to update question code later, and we will be sure that it is updated at both of the places. Now, let us run our program one more time. If you can notice our output is changing. I would just magnify the output for you 2, 3, 4 first question, second question third question and fourth question.

(Refer Slide Time: 35:55)

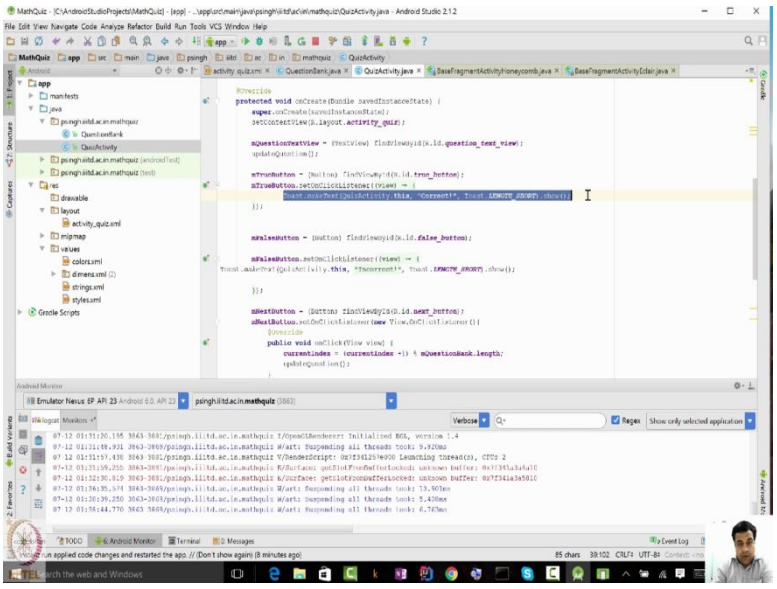


So now our Next button is working around true false working, ok, is pertinent true, no it is not. This is because we have so far not added functionality for determining whether our question and the answer that we get are correct or not. Now that our next button is working, let us also check if our true or false button is also working. Let me press it and it displays correct which is wrong as you may know. This is happening because so far we have not updated the functions of our true and false button, since we added new questions to our applications. In our next step we will be doing that.

So we will add a new method called quiz, we will add a new method let us call check answers. This method will accept a Boolean variable that identifies whether the user press true or false, then this method will check users answers against the answers in the current true false objects. Finally after determining, whether the user has answered correctly or not, it will make a Toast that displays the appropriate message to the users. Let us add this method in the quiz activity class. (No audio from 38:00 to 38:25) I declare a variable called press (())(40:30) then try to determine what the user pressed, if it is same what has been given as the answer in my question object.

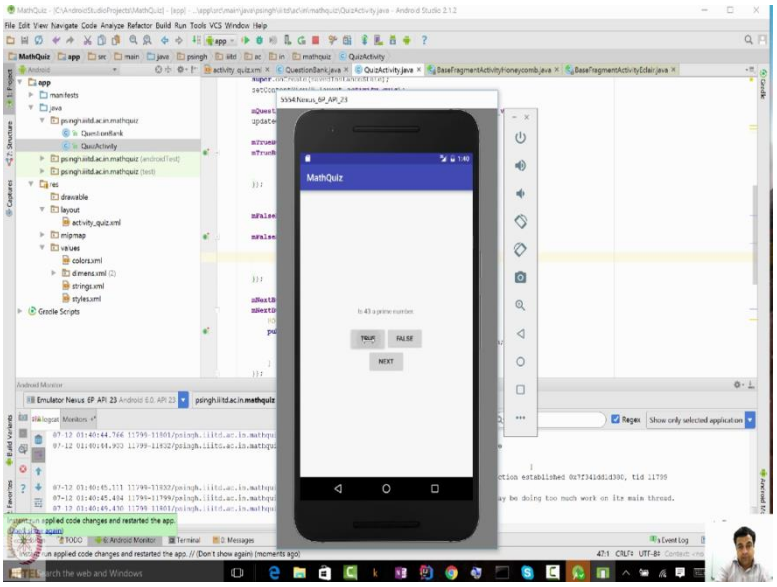
If this is indeed the code case, then I will make a message result Id of correct Toast else I will make message result id of incorrect Toast. I will then go ahead and show my toast. For this I will be using code so that what we used earlier, let us save our file, is the code now complete or do you need to do something else as well. Well within the button's listener you will have to call our new method called check answer which we have thus just described. Let us go and call check answer whenever we press true or false buttons.

(Refer Slide Time: 43:07)



Instead of simply displaying the Toast, I will call check answer, I will initialize it with true, this solves, similarly for the false button I will just call answer or else with false and hopefully I am done ok, so now let us test our application we are running it one more time.

(Refer Slide Time: 44:15)



Text is working, yes, incorrect, correct, good. So now our application is fully functional you can show us 4 different questions and if you make a mistake you can point it out, that is good development. As I move far, I would like to add a previous button to this application. As you may guess the previous button should take us to the previous question. This is it for this lecture we have successfully increased the functionality of our application, we got a very

brief, but important (())(45:08) to the MVS design pattern and we enhanced our application. For this example I have used the resources from the book Big Nerd Ranch Guide.

You may buy this book from Amazon or from any other book seller. The full name of the book with publisher information is given on the NPTEL website, Thank You.