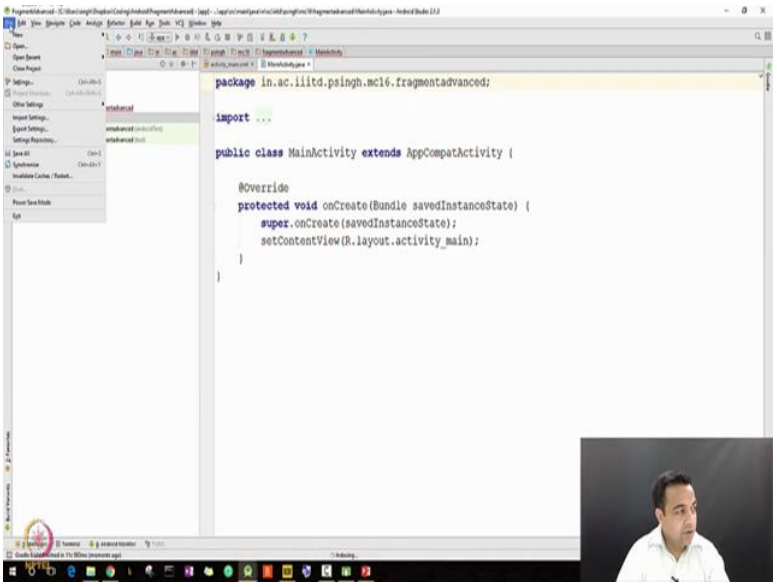
Mobile Computing
Professor Pushpendra Singh
Indraprastha Institute of Information Technology Delhi
Lecture 26
Fragment

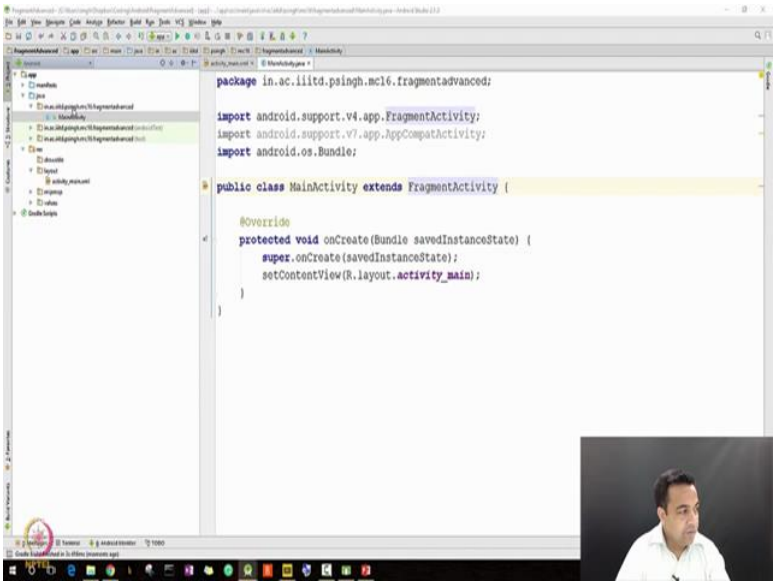
Hello, last lecture we created a basic fragment and we call it in the activity using code. Today we will advance the application and we will add fragments in the XML as well as call fragment from the code. We will also do replace operations; we will also implement a Calback in the fragment that will call a method in the activity. And we will also get a reference of the fragment in the activity just to show the concepts that we learned in earlier lectures.

(Refer Slide Time: 1:47)



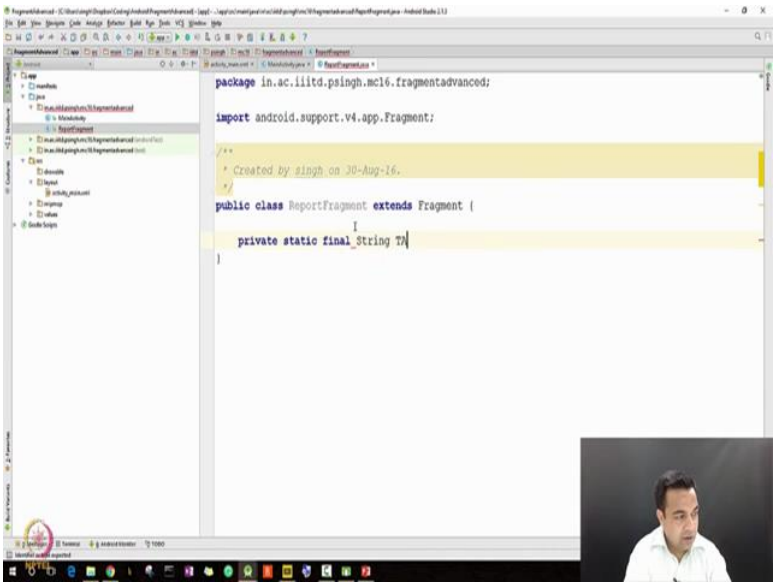
So let us start with the new project for fragment advance. So I start a new project going to the file new project I will call it fragment advanced choose phone in tablet I am using API 23, choose empty activity. I keep the same name and I press finish. My new project has been created, let me close the older project again the first step is I will add the support library. Go to file, I will go to the project structure in the file, go to the app, go to dependencies, go to plus library dependency and I add the support dash report library and I press ok.

(Refer Slide Time: 2:29)



The first step is again to convert my activity to a FragmentActivity. I will make sure that I am using the FragmentActivity of the report. The onCreate is already there that is fine, I want to see my layouts. Now in this program we will actually be adding two fragments and we will be working side by side with them. So for example we will be having a fragment called detail fragment and another fragment called ReportFragment or ListFragment let us say.

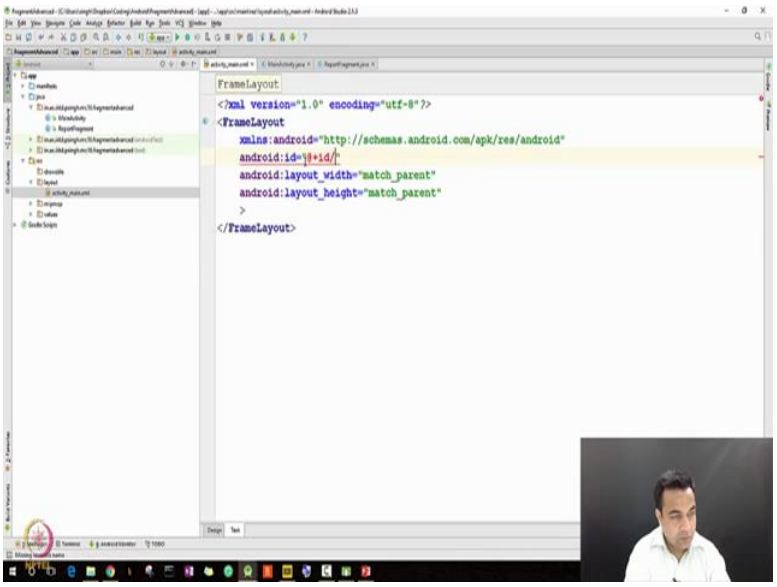
(Refer Slide Time: 3:56)



So yes let us first try to create few fragments, so again I will use the simpler method let me call it ListFragment with ReportFragment. We can see another file I will use the same extends Fragment to make a (()) (03:33) Fragment. I am making sure that I am importing the support library. I will add a string tag just in case where I need to print some long messages.

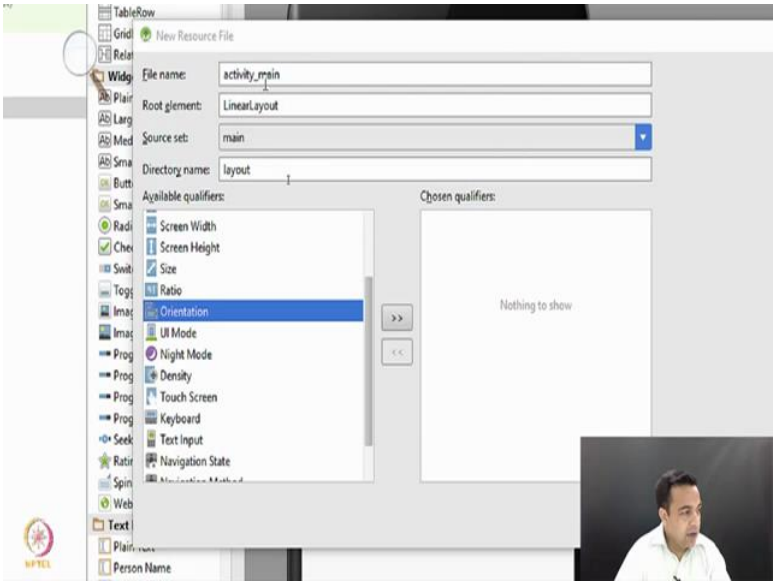
String TAG equal to REPORT underscore FRAGMENT. Now I will have to first implement the onCreateView, but like last time before going to onCreateView we will have to create some layout files. Actually we will be doing multiples, we will not only be creating layout files for our fragment but we will be creating different layout files for even our activity. So our first layout file for our activity will be same as we created in the last example that is a very very basic layout file which uses frame layout does not do much except an id field.

(Refer Slide Time: 5:20)



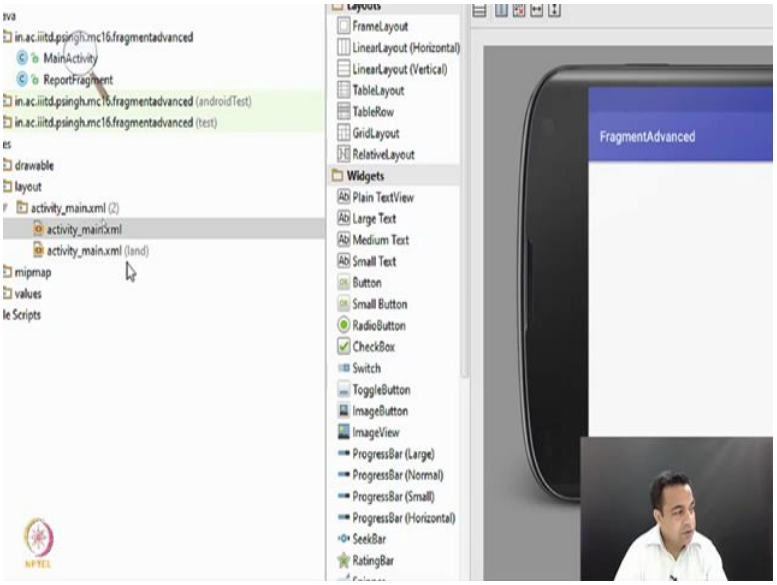
I will give it the same name as last time container that is fine. I can see the design it is very simple. Now I want to add another layout file which corresponds to the landscape mode of my activity. What I want to say is that when my phone is in the portrait mode I want to see only one fragment, but when my phone is in the landscape mode I want to see two fragments.

(Refer Slide Time: 6:33)



So let us add another layout file and this we have done earlier my file name is the same name as another layout file which is Activity_Main, the root element is linear layout I will currently leave it and I will choose the orientation. Let me switch on a magnifier so I have given the activity main everything is same and I have chosen the orientation, I change it here portrait is already there I do landscape and then I press ok.

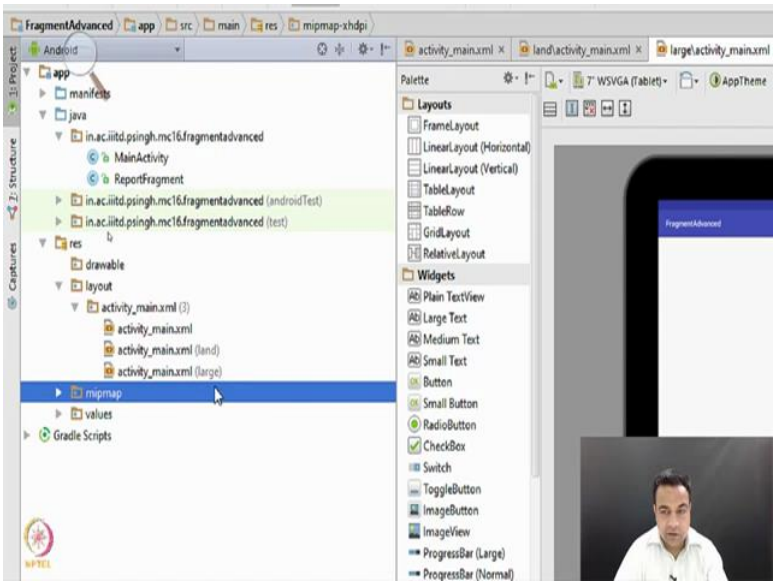
(Refer Slide Time: 7:01)



As you can see when I do this another Activity_Maine has been created and you can also see that it shows land which refers to the landscape. I will again do the same thing and will try to create another layout for a large device such as a tab. Let me again go to this right click new layout resource file, I will give again the same name we give the same name because we want

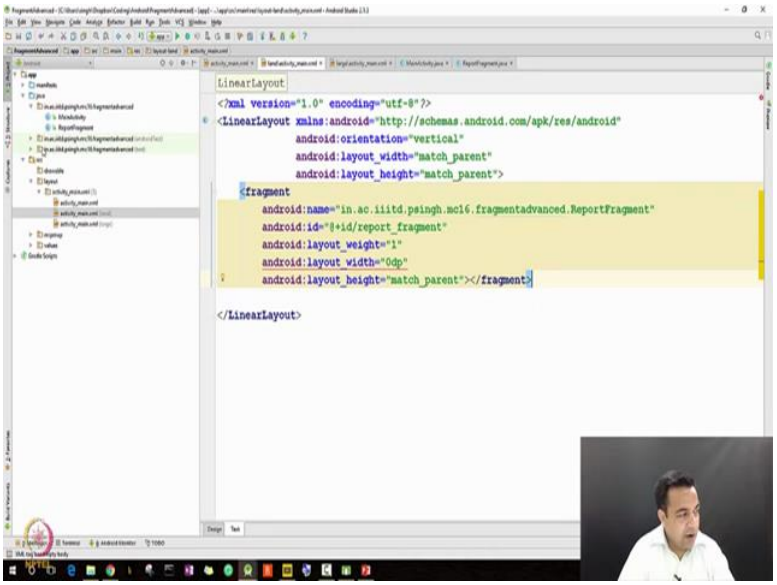
to have the same resource id LinearLayout and this time I am more worried about the size and I will say I want to make it further large size.

(Refer Slide Time: 7:56)



Now you can see that there is a third layout file has been added for the large. So now we have a portrait we have a landscape we have whenever we are using a large device such as Nexus 9 tablet, ok. So now let us kill the magnifier and we will now try to change the layout of the landscape and the large for the landscape as you see currently it is exactly same, but we will soon be changing it to adding two fragments into it. And we will make sure that these fragments are added using the XML method. In last program we saw the method of calling back a code let us try the using the XML method. So in my linear layouts I again start by call a fragment android layout, this is the first field that I want to give it that what fragment that I will do.

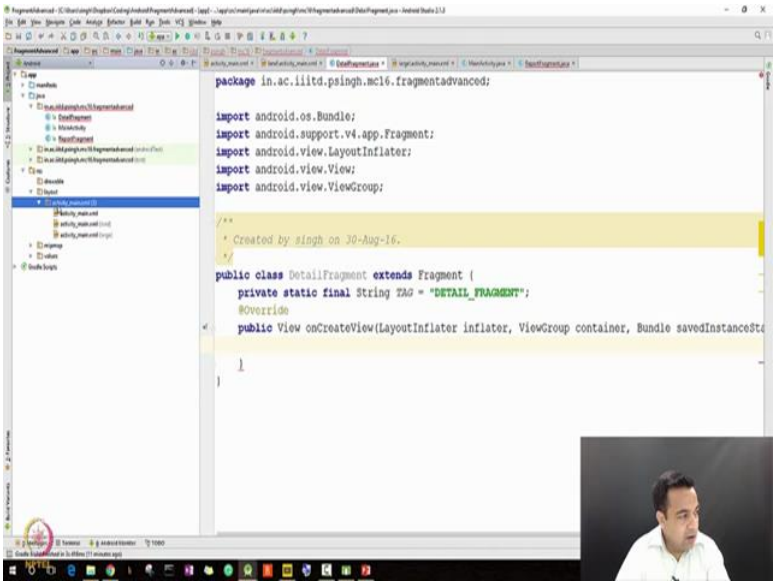
(Refer Slide Time: 9:16)



So this we define by calling by an activity call android name. And the name refers to the class file that we will be using for the fragment. So the only fragment that I currently have is the ReportFragment, so you see that I am giving the complete path to the ReportFragment. Then I will give it a id android id equal to @+id/report_fragment. I will choose a variable called layout_weight I will very soon explain you why I am doing this and I will give it a value of 1. Now for layout_width I will chose a value called 0dp and for layout_height I am chosing a value called match_parent.

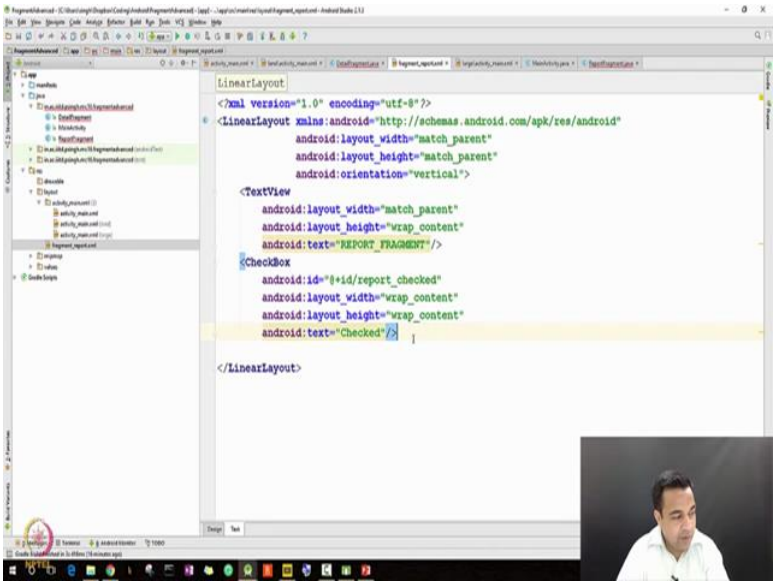
Now when we give 0dp what happens, number 1 that we are giving a weight, so our weight will decide that how much area it occupies? Height we have already defined, in fact I can just move it here and define it next line as weight I do not want to do match_parent I do not want to do wrap_content I will just say 0dp, ok. This may sound bizarre to you but it is not that because the width field will control how wide my fragment is.

(Refer Slide Time: 12:17)



But first let me add another fragment here. So first I will go to add another fragment and let us say I will call it DetailFragment. I do the same thing I extend Fragment I make sure that I am using the support library. I implement the onCreateView method at override public you know now why we do public because it will be called from outside onCreateView takes three parameter ViewGroup container and a bundle of let us say savedInstanceState. It will require a view to be return and that view both for detail and both for ReportFragment. Let us also implement the same code in the ReportFragment and to all the inputs. Similarly I will do a TAG to the detail DETAIL_FRAGMENT fine. So now our empty word is above our fragments are ready, in order to implement in order to create the view that this method requires you have to create the layouts.

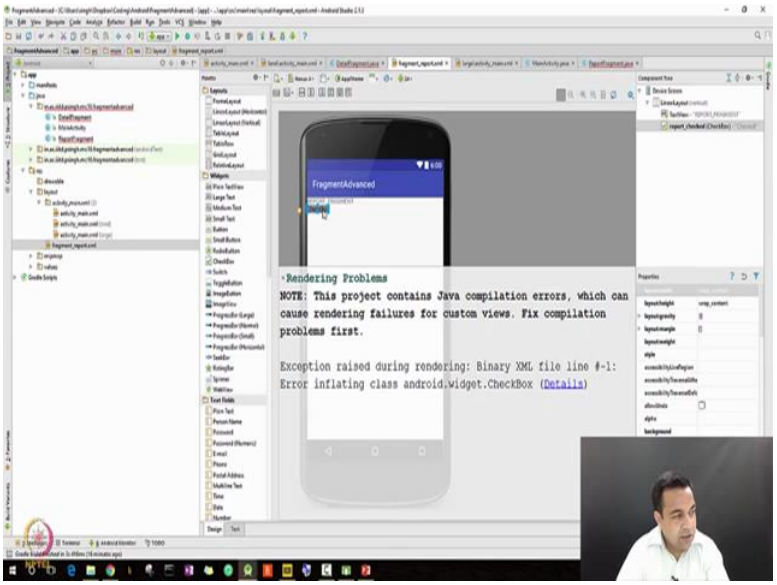
(Refer Slide Time: 15:15)



So let us first get on to the layout of our individual fragments. So I am going to the layout and I am going to add a new XML of the layout XML file wait a second. So let us call a first let us create the layout file for the report. So XML layout files what name you want to give it, let us give it the name fragment underscore report finish. Let us go into the text what do you want, this is fine, this is fine, this is fine let us add the orientation android orientation vertical. Now I just want to create a very simple layout where I will be having a TextView and I will be having a CheckBox because the idea is of today's program is only to show you certain concepts let me just have a TextView which is very simple TextView just shows a title so that we can distinguish which fragment it is so width is match_parent, height is wrap_content as (()) (16:19). And we are just doing a text let us hard code the text further time being and I will call it REPORT_FRAGMENT ok.

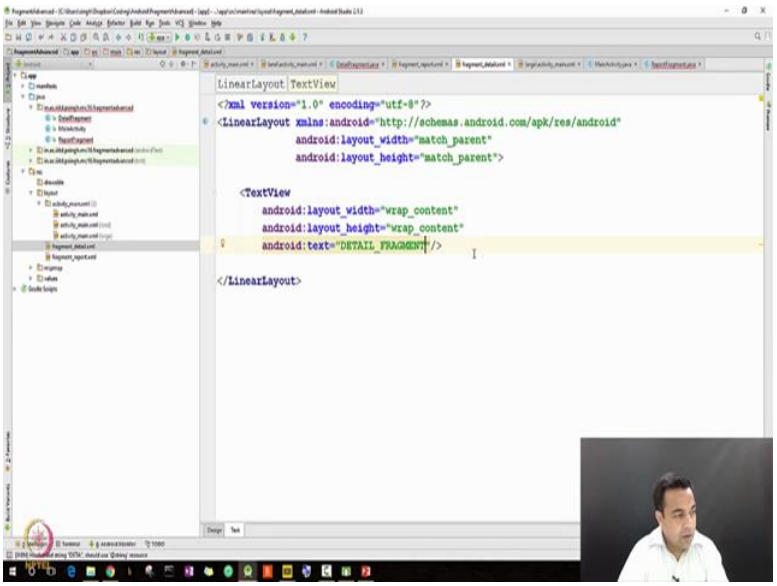
This is fine and then we will create a CheckBox, why we are creating a CheckBox because I want to associate a listener to the CheckBox and through that listener we will I will try to show you how to go from a fragment to activity and how to go from a activity to fragment. So let us add that CheckBox width wrap_content, height wrap_content for the CheckBox because we want use it our program we will also do an id@+id/ report_checked, I will give it also a text android text Checked. Again I am not using a very good approach here I am hard coding it but we have already learned how to create a string resource, so here we just want to learn about the fragments. Let us see what design it generates.

(Refer Slide Time: 18:15)



So it is generating design where it displays something there is a CheckBox here, let me refresh I will currently not worry about it too much. I will go here and I would actually like to create some margins let me just add those margins in the list and let us say a 16 dp margin so that we can see our final fragment also I will just add a 16 dp margin. Let us only so that we can see our fragments better so that is fine that is our report fragment layout.

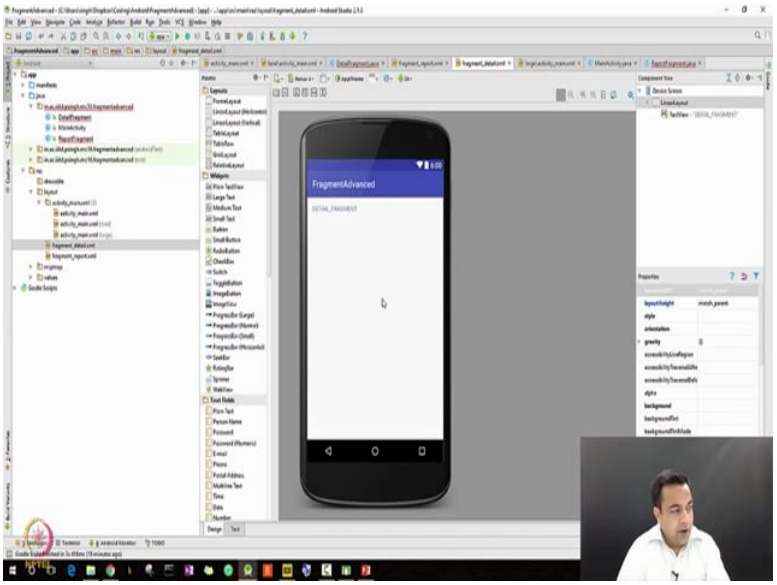
(Refer Slide Time: 19:48)



Similarly, I will create a for our DetailFragment, fragment_detail linear layout this is all good fragment_detail match_parent, match_parent. I will just add a TextView where I will just display a text so that we can see the text here. So TextView width wrap_content, height may

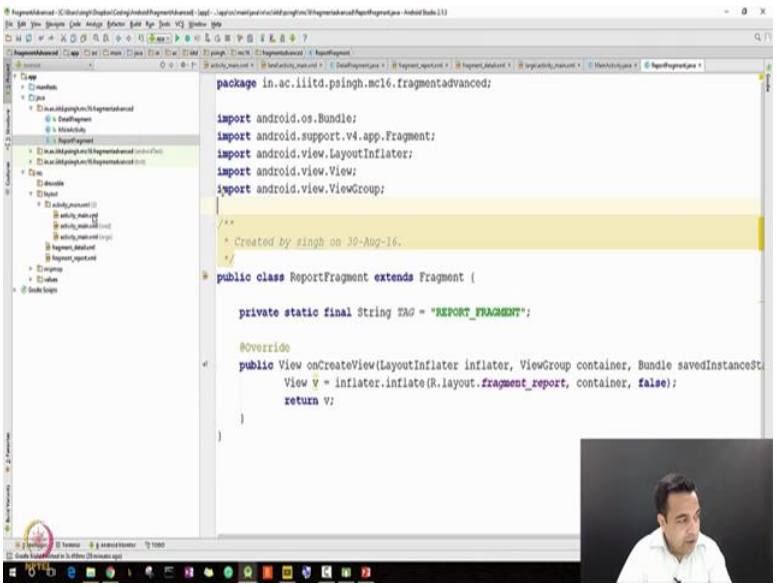
be wrap_content that is sufficient and just add a text which shows detail DETAIL_FRAGMENT.

(Refer Slide Time: 20:49)



And see the design and go here and I will add a margin of let us say 16 dp, fine. So we have created two layouts because we want to distinguish between the layouts.

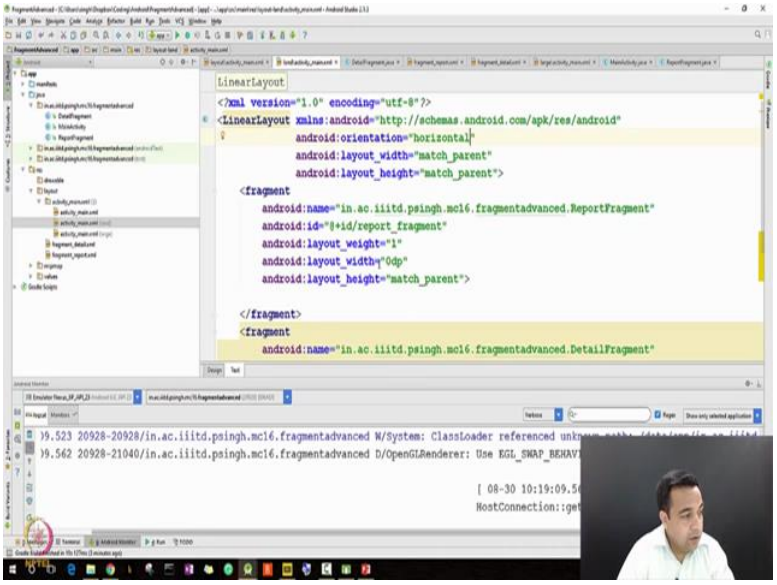
(Refer Slide Time: 21:15)



Now let us go back to our programs first to the DETAIL_FRAGMENT I do not have to do anything except just create a view object by calling the inflate method. This time I will give it the layout id of the fragment_detail everything else is same as we did in the last program. This is fine; we will just return v and as you see all the errors are go. Similarly move it to the

REPORT_FRAGMENT, so first we will fix the onCreateView method by doing pretty much the similar thing inflator.inflate (R.layout.report.container.a container false) and then let us say return v, ok. So now we have created two fragments and we have created one activity let us go back to the activity main it has nothing but let us go to the landscape.

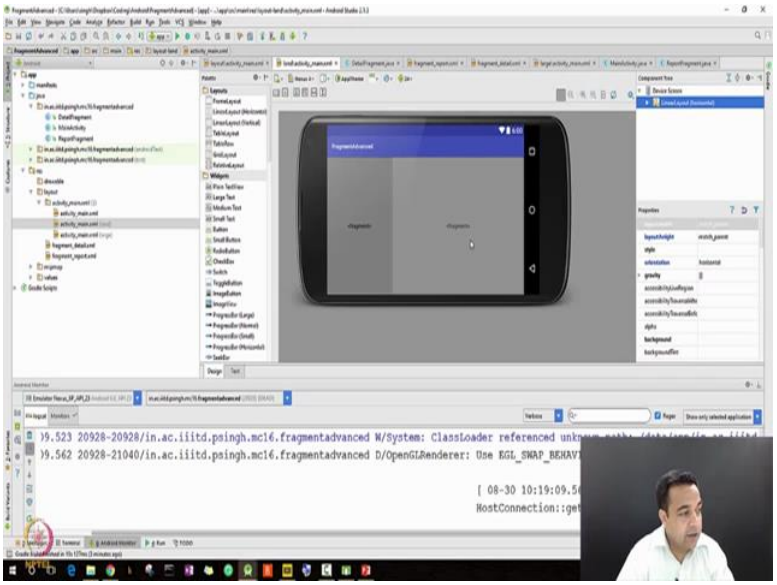
(Refer Slide Time: 22:53)



In the landscape we had first fragment now let us add another fragment except for this fragment we will want it to be the DetailFragment and with an id called detail fragment id. I will give it a weight of 2 because I would like to show you a how the weight plays the role. Do a refresh go here and not to worry about these for the timing I just want to run it and see what happens, ok. So we are getting this error let us first try to fix it, ok. So we have made a small mistake here we should have made the orientation horizontal because what we were giving is that as you see the error is gone let me explain what happen.

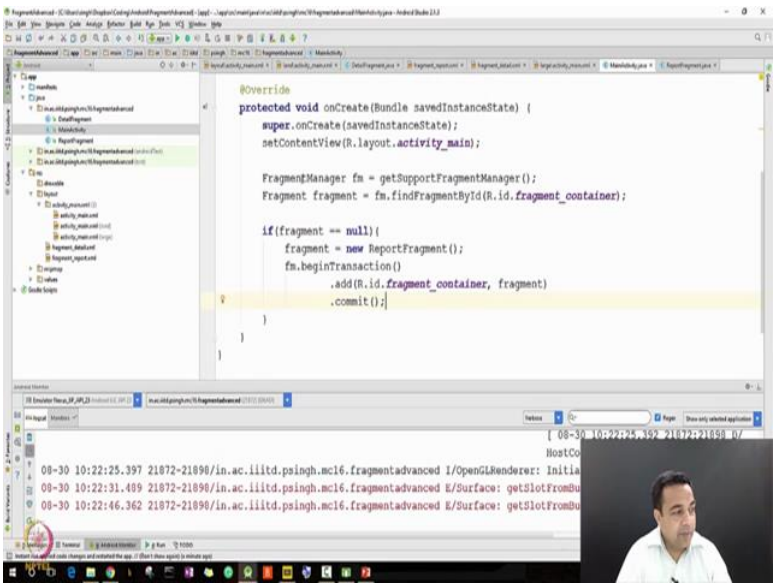
When we did the vertical and we give it the match_parent height with a 0dp width obviously this was an error because in the vertical the individual view elements are placed from top to the bottom. This should be the horizontal now what it means is that it is horizontal, so the height is equal to the parent height but the width will be decided by the weight.

(Refer Slide Time: 24:38)



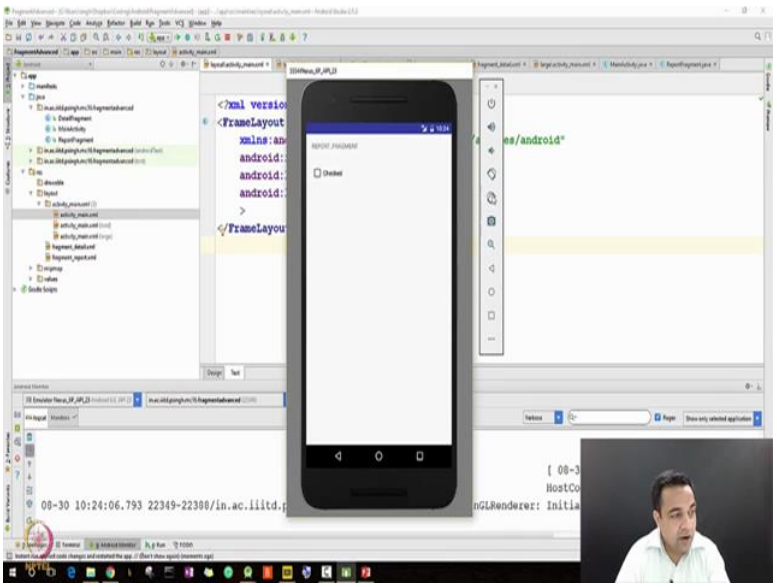
Let us see how it will look so yes it will look roughly like this. Let us now run this program and see it. So how does it do so it start that is the portrait we will change it to landscape and now you can see our REPORT_FRAGMENT has come on the left and our DETAIL_FRAGMENT has come on the right. So from one single activity we were able to call both. Go back to the portrait in the portrait we are not calling any fragment. Now let us kill it and go back to fixing our activity files. Number one that let us even from our portrait mode, let us just add the code which we had in our last program and that is the call by the code and see what happens. So I am just copying from our last program which we created in our last lecture and we are adding it.

(Refer Slide Time: 25:48)



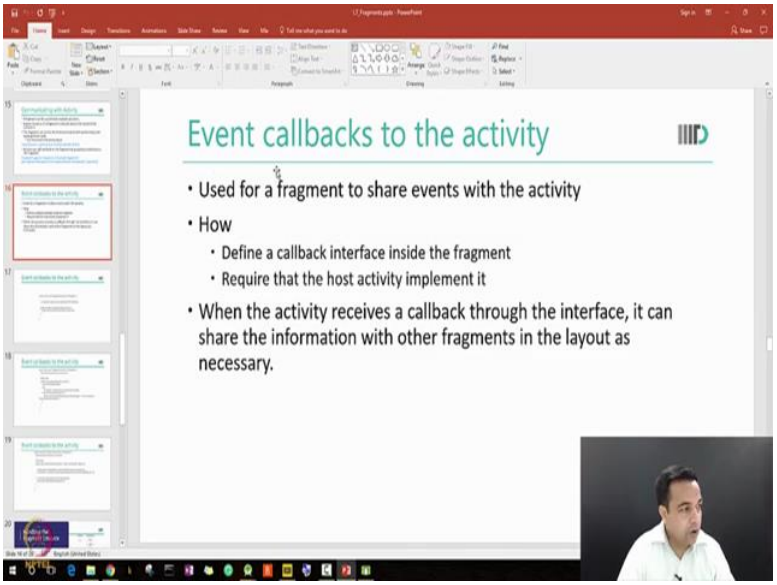
Let me explain it quickly for the people who may have not watched the video. We are calling a FragmentManager, we are creating a fragment, we are giving the id of the activity main then we check for the null. We create the new fragment we are creating a fragment of type ReportFragment and then we are adding it. So now in the same program at one time we are adding a fragment using a code and other time we are adding two fragments using the XML. Let us run it again and see when our program starts in the portrait mode it will use this code and it should show us the ReportFragment added and when we convert it to the landscape it will show us two fragments added, so one fragment and 2 fragments that is it.

(Refer Slide Time: 26:46)



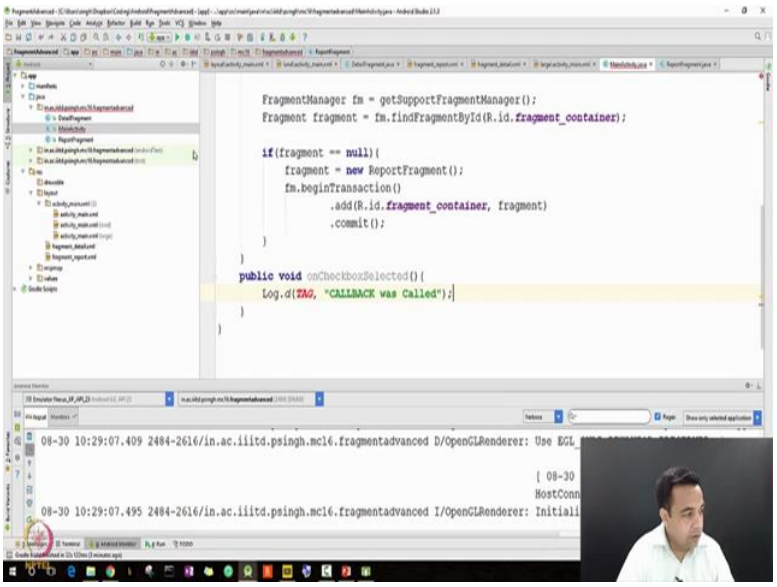
So it starts in the portrait mode it is showing just only one fragment now let me rotate it and it shows us the two fragments, ok. So now we will add few more functionalities we essentially want to add that how to use the Callback method to call an activity method from a fragment. And for that what we will do is that as you know we created a let our application come online that we created a simple CheckBox in one of our fragment. What we will do is we will put a listener to it and when we click it, it should call a method in the activity. So if you have forgotten what I am trying to show let us just quickly go to the our lecture slides and let me show you what I am trying to show to you.

(Refer Slide Time: 28:04)



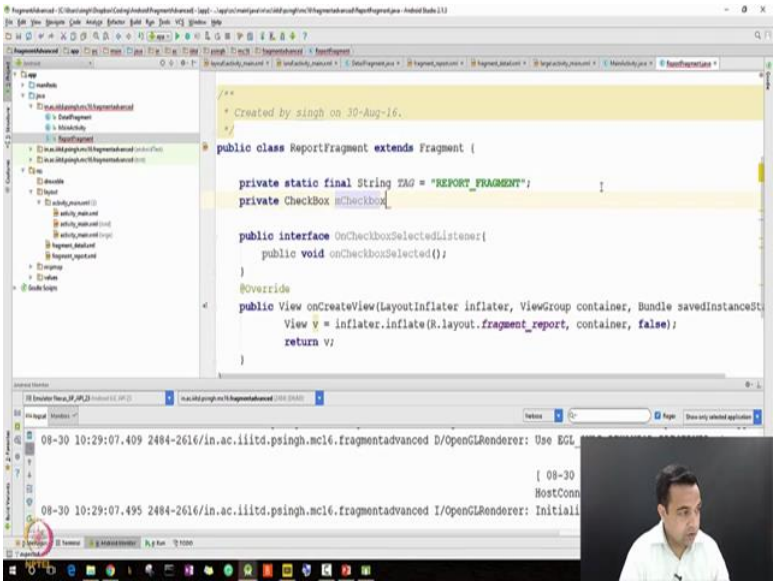
So I am trying to show to you this events Callback to the activity. So last in last lecture we had discussed how do you communicate with the activity and we have said that one of the method is to an event Callback to the activity, this is what we are going to do now. So let us go back yes so this is our application currently nothing is happening what we want is that when we click it, it should call a method in the activity using the Callback as we discussed in the last lecture. So we will have to do few things, number one is that we will have to first add a listener here which is related to our CheckBox and that listener should be called whenever the CheckBox is select.

(Refer Slide Time: 29:13)



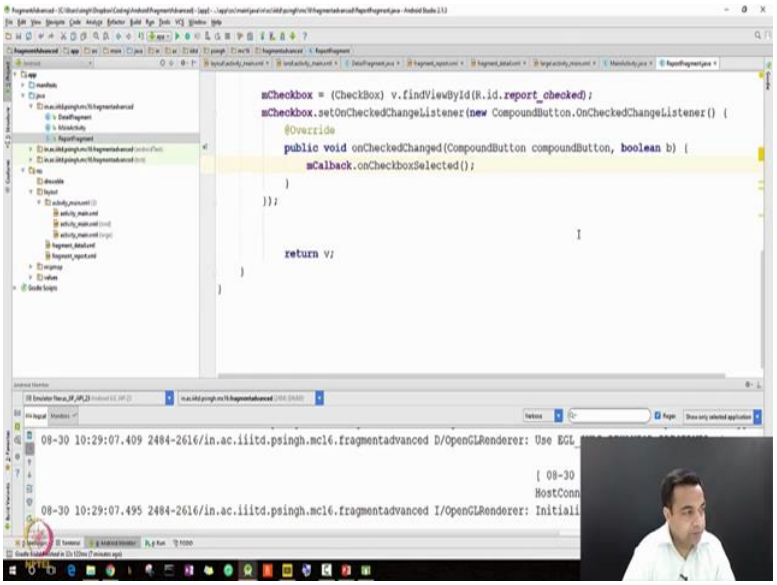
So a way to do it is that we go into our activity class and we add a method called onCheckboxSelected, so will have to override it. So let us override our method called onCheckboxSelected. Public void onCheckboxSelected, I am doing nothing but I am just adding a Log entry here because the purpose is to show you how the Callback works rather than doing something useful right away, so I will just write "Callback was Called". Now in order for the Callback to be called there are multiple other things that we need to doing. And those actions have to be taken in the fragments.

(Refer Slide Time: 30:23)



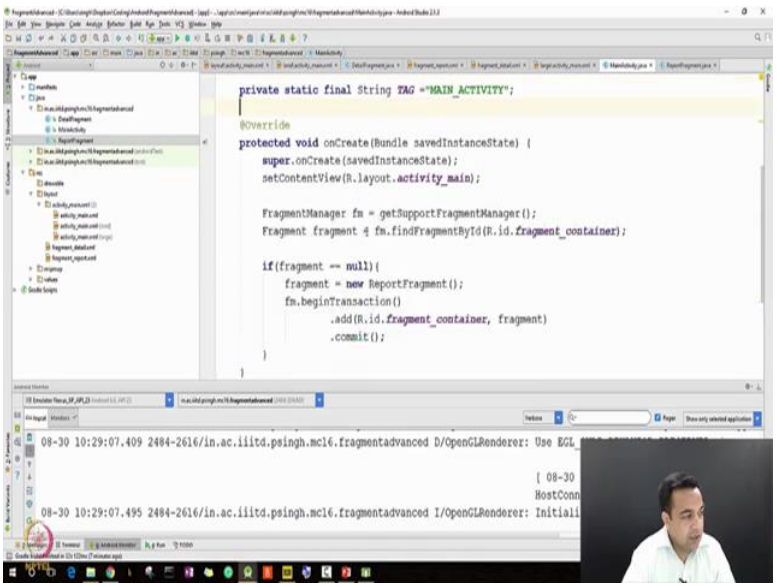
So our Callback is in the report, currently I am not ok. So I can possibly add the string to get rid of this error Private static final String TAG equal to MAIN_ACYIVITY yeah. Now go to my report fragment and that is where the majority of our work will be. So number one thing is that first we will have to create a interface, let me first define an interface. So our interface we define as we define any other interface so public interface OnCheckBoxSelected ok listener this only implements one method public void onCheckBoxSelected method ok. Then I will go on and I will first want to create some variables so that I can make use of it. So one of the variables is to make use of the CheckBox that we have in our layout mCheckbox currently we are not using it we have simple interface, then we go and we use the Checkbox inside our OnCreateView and we attach a listener to it.

(Refer Slide Time: 32:28)



So let me do mCheckbox equal to CheckBox we want to find out findViewByIdR.id.as you can see that this was the id that we have given mCheckbox.set (on click list) set OnCheckedChangeListener. Now we will have to give another anonymous inner class, so new this is a kind of a on. We have done this multiple times OnCheckedChangeListener, so I am creating another anonymous inner class and then I am overriding the OnCheckedChange as you see that android studio has already created a code for me the only thing that I need to do is to use our Callback from here to call our Callback from here.

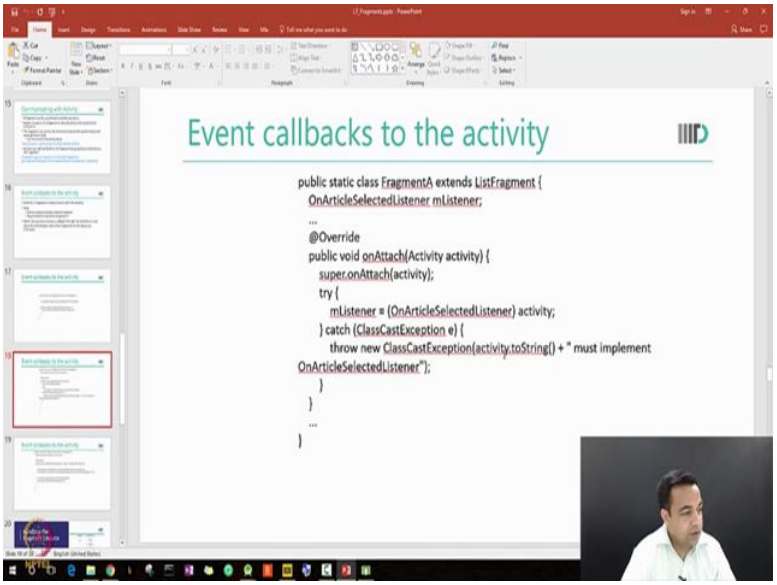
(Refer Slide Time: 34:46)



So first to in order to do that I will be defining another variable let us say of a type of interface OnCheckboxSelectedListener mCallback then I will be using my new variable

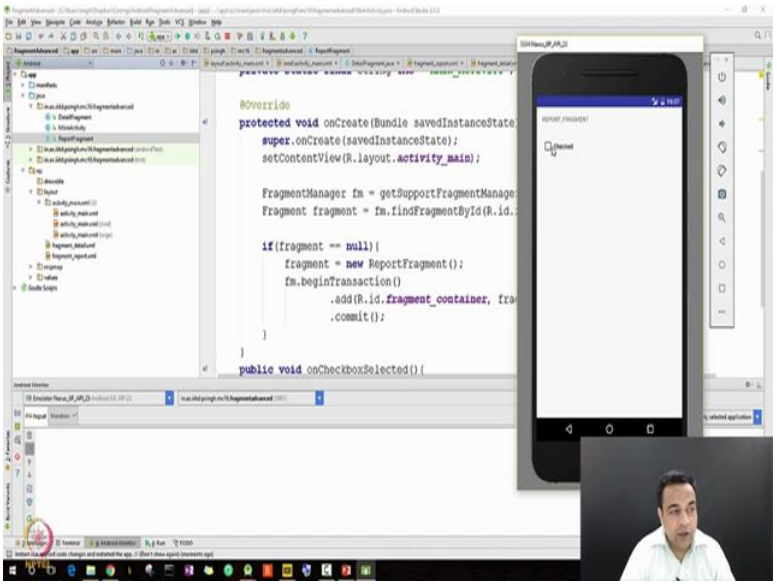
mCallback here to actually call the method OnCheckboxSelected write. Now we what we have done let us go back we have defined and interface, we have defined a variable of that interface and then we are using that variable to call the Calback method. And we already defined that Calback method into this but this is currently not link because we will have to say that our activity implements the OnCheckboxSelectedListener so here. So now our method OnCheckboxSelected associate with this listener is defined in the activity and the actual interface is and the interface is defined here. So if you look at the yesterday's slides that is what we were doing.

(Refer Slide Time: 35:34)



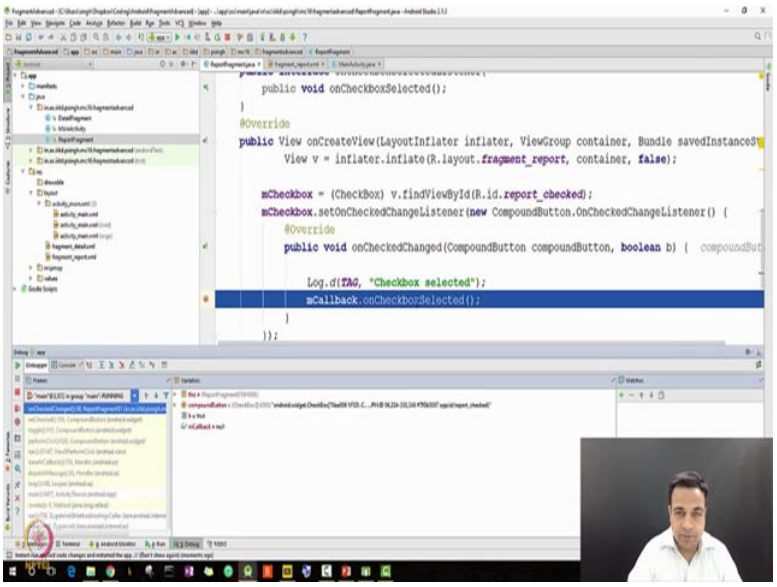
We are saying if you want to do the Calbacks the activity must implement the interface and the variable which calls it and that is how it is passed. So we go back here everything is ready from our side. So when we select the CheckBox now a method in the activity should be called. So we will be selecting the CheckBox in the fragment, but the method in the activity will be called. Let us run the program and see if it is really happening or not, so we will be keep watching our log cat window.

(Refer Slide Time: 36:14)



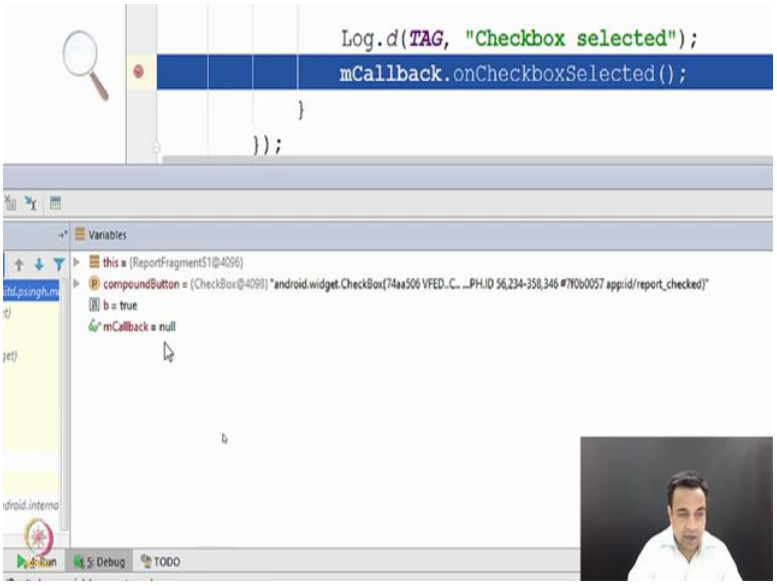
So yes, it is here now this is a fragment being displayed the fragments we call by the activity is in the code and hopefully in though I am interacting in the fragment it should call a Callback which is defined in the activity so let me click it. Now you just saw a bug in our program, I left this bug knowingly and now we will try to fix it. The purpose is to show you that bugs happen usually they happen because of our oversize, so let us see what our oversize is. So first let me run it again so that you can understand but this time I will run it using a debugger. So let us first put a debugger break point and now we will start with the debug and not with the run.

(Refer Slide Time: 37:29)



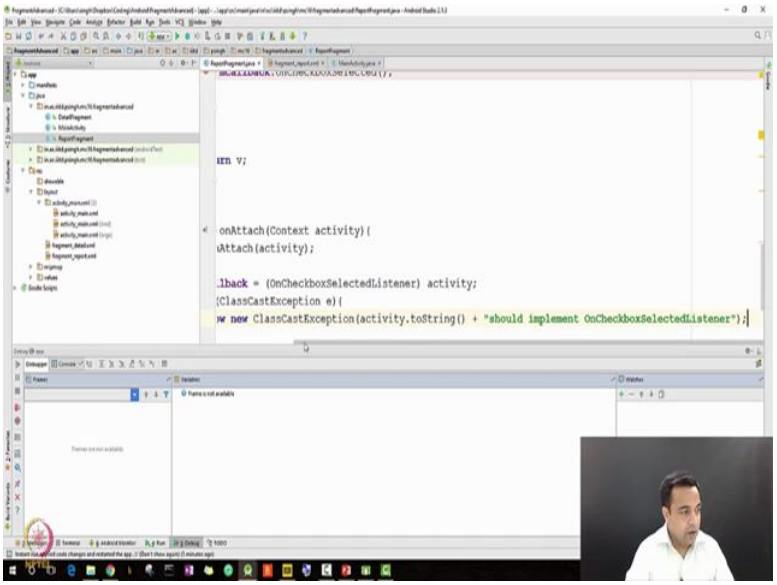
So when we debug our application starts I click the check and this time our execution stops at this point. Let us see what happens so on this point our execution has stopped we can see by the blue color. And now we are checking the variables at this point of time and you can clearly see that the mCallback is null.

(Refer Slide Time: 37:47)



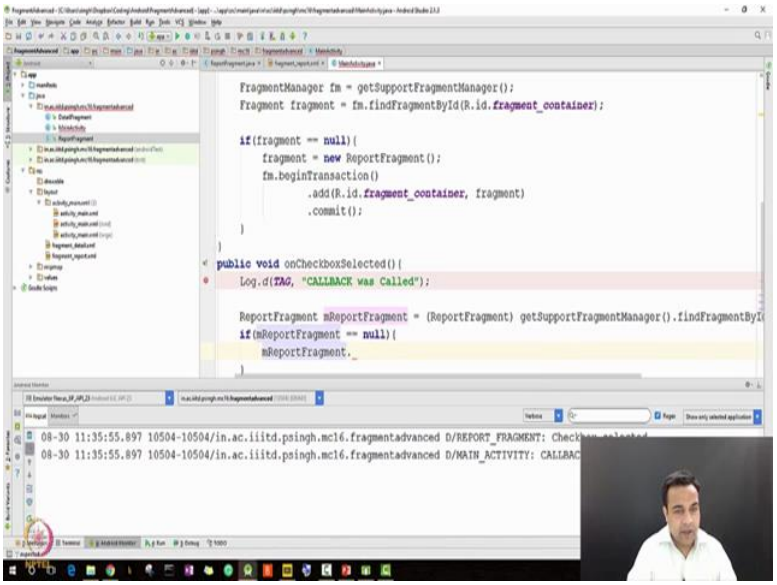
So mCallback here is null and when we call something on a null object we get a null pointer exception and that was the exception that we were getting and if we resume our execution we will get our exception and our application will correct. So let us now fix this bug, so let us see so we have missed one important thing in our program which is that onAttached method. So let us extent an override the onAttach method. Let us implement onAttach, we override public void onAttach takes one argument of type Context let us give it any name.

(Refer Slide Time: 38:50)



We go the first thing as usual we will call the super onAttach and then our actual work will start, I will also press Alt Enter so our error is gone. Here we will see that if we have really attached our interface correctly or not. So I will say a try and I will say OnCheckboxSelectedListener and I will give it a reference of the. So we have taken the context that we have got converted it into the right type and pass it to the on Callback. So this line was the line that was missing and just in case there is a problem we are asking it to through an exception. Now instead of our app crashing we now know how to handle the error if at all the error happens. So let us hope that everything will go fine but still we add a error message let us say should implement OnCheckboxSelectedListener right. So this reference is what we were missing earlier now we have provided it let us run our program again and see whether we get the right Log message that is this Log message or not.

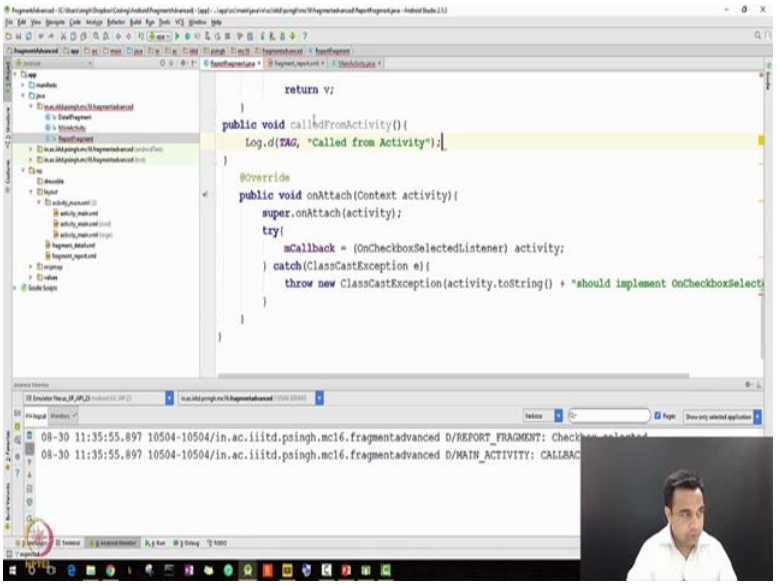
(Refer Slide Time: 41:01)



So let us clear our log cat wow so we have got our right log message that is we were successfully able to call a method in activity from our fragment. Now let us do the second part from our activity we will try to get a reference for the fragment and go back into the fragment. So what I will do is I will add some more code here which is very similar to the code that we wrote earlier, but this code what we will try to do is. So we will try to do two things first that we will try to call a method from the fragment from our activity and second thing we will try to do is to replace a fragment because that is another concept that we still need to try in our program. So we add some more code so ReportFragment let us say mReportFragment equal to (ReportFragment). So we learned in 2 lectures back that we need to get a FragmentManager and then call the method findFragmentById.

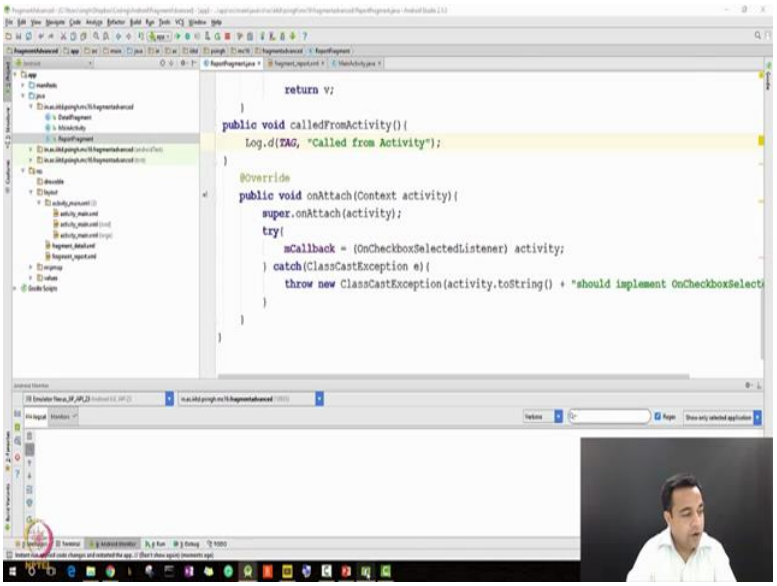
Now please think and tell me that what Id I should give here. So we go to R.Id and we will give it the id of the container because that is where we want our fragment to pick up. Now we are doing not much but we are just checking if we have actually got a correct object so we are just checking for null value and then we say mReportFragment and we want to call some method.

(Refer Slide Time: 43:16)



So let us create a method in our ReportFragment class and I am calling that method as nothing but a simple public void calledFromActivity again we will do not much but just print a log message send Tag and Called from Activity. Let us go back and call it from the activity, fine. Now if you see that when we press this CheckBox now first this log message should be printed and then this log message should be printed. So let us run our program one more time and see

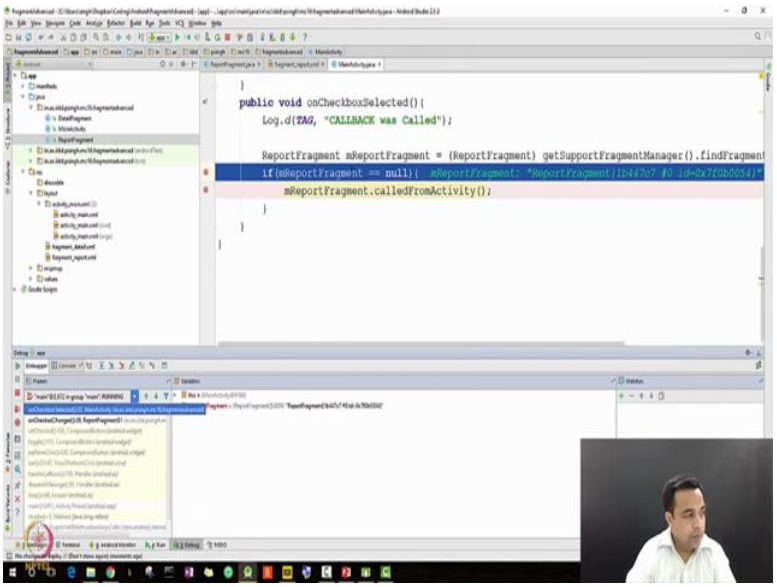
(Refer Slide Time: 44:12)



Here we are clear our log cat window click the check and you see that we have got the Callback was called, but somehow we have not got this log message, so let us run a debugger again and try to debug the program. You will see that I make very frequent use of the

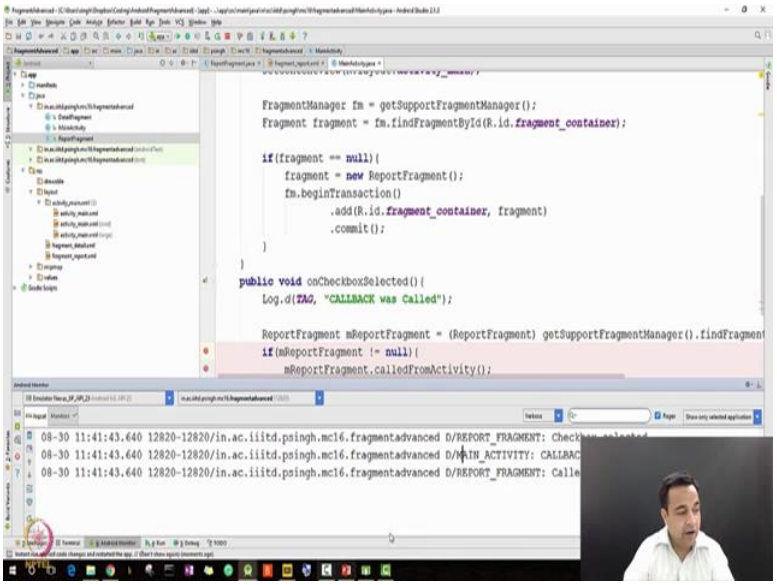
debugger, so this time we are here onCheckboxSelected we move forward we come here mReportFragment, I want to check the value of mReportFragment shows some valid value, then I move forward oh I should have stepped into let me start the debugger again.

(Refer Slide Time: 45:52)



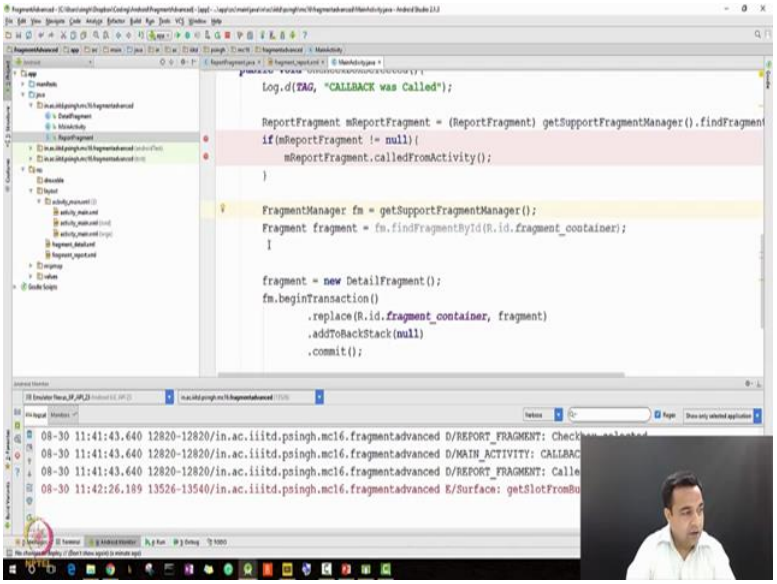
So select we resume we come here mReportFragment is definitely not null I will also say at watches so yes we have the mReportFragment here and we stepped into oh sorry ok. So we have made a mistake I set it like this actually it should be when it is not null then it should be call. So as you can see some simple mistakes you can make, but the debugger helps you because we were not getting in the, if we knew that there was some problem and once we set up the debugger we could figure it out.

(Refer Slide Time: 46:45)



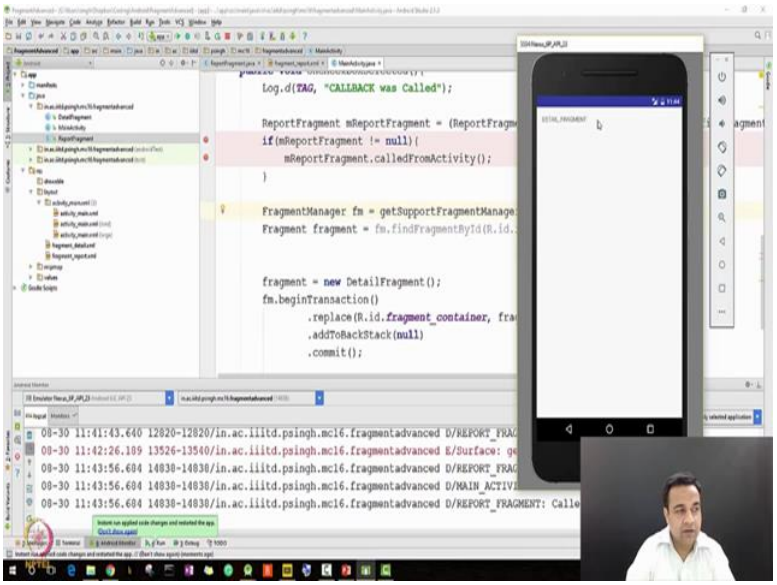
When I check and now you see all the correct messages are printed here Callback was called and then from that Callback we call we got the fragment reference and we call a method in the (()) (46:54). So we have gone back and forth from fragment to activity and then activity to fragment. Now one last thing is missing which is let us say using the back stack, so currently let me execute it again and try to show you what happens. Currently when I press the back button my activity is gone. Now what I want to do is that I want to make sure that if I do a replacement of the fragment then my back button uses the back stack to show me the fragment that was there earlier, ok. So let us do something we will go, we will add some more code and we will try to make sure that this time we so we are into onCheckboxSelected let us add some code right here only.

(Refer Slide Time: 48:41)



So what I will do is I will add the code which is similar to the code we did earlier that is the code which we use earlier in the onCreate but this time instead of that I will use the replace. And as you can see that I am replacing my ReportFragment with my DetailFragment so let us run this program and see where what effect does it make.

(Refer Slide Time: 48:57)



So I do come as you see that when I press checked my fragment has been replaced by the DetailFragment. If I press back with for a movement I come back to the previous fragment, but then it again goes because my selected state method is called. So now you see that in this video we have seen all what we had learned in the previous lectures. I will make this program available on piazza you can download it and check it but I will also request you to develop this program on your own, thank you.