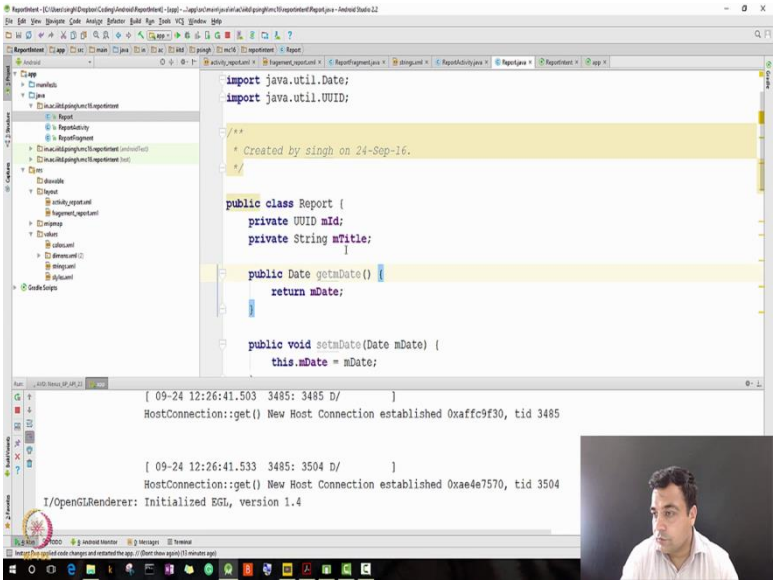
Mobile Computing
Professor Pushpendra Singh
Indraprastha Institute of Information Technology Delhi
Lecture 34

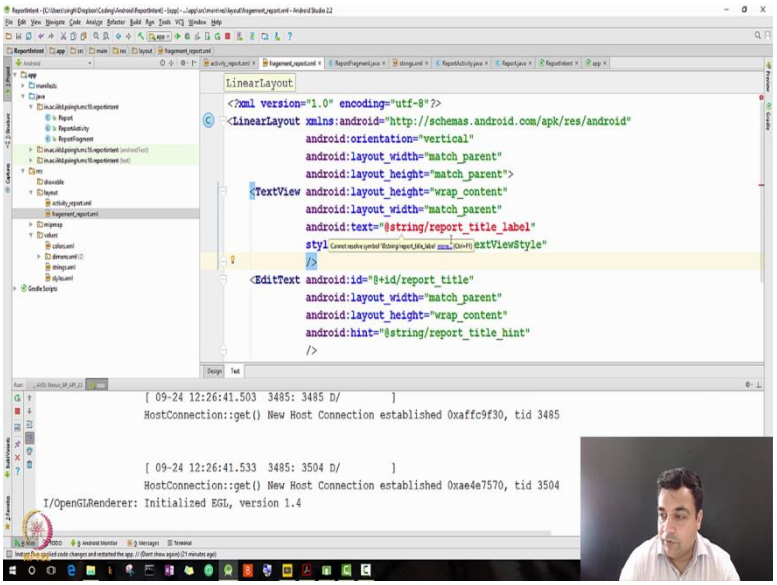
Hello, in this class we will continue that our last program and we will now update it with more features. So in the last program we created a simple fragment activity class and in which we added a fragment using the code, now let us extend that code to make it more functional.

(Refer Slide Time: 0:54)



Let us get started, so the first thing that we are going to do is to add more fields to the report class mainly we want a Date and a simple another variable which says that whatever we reported did it got resolved or not mResolved resolved, so for date I am getting an error I do, I do (())(01:16) and I can enter I have used the java.util.Date. This is fine and in my in my constructor I will also initialize my date to new Date and then I will go on to develop the getter and setter for my program. So let us say for mDate I develop a getters and setters and for and I would also do the same for my resolved, so where is my resolved? Generate getter and setter, now variable put it at the top, similarly I will put it at the top on the.

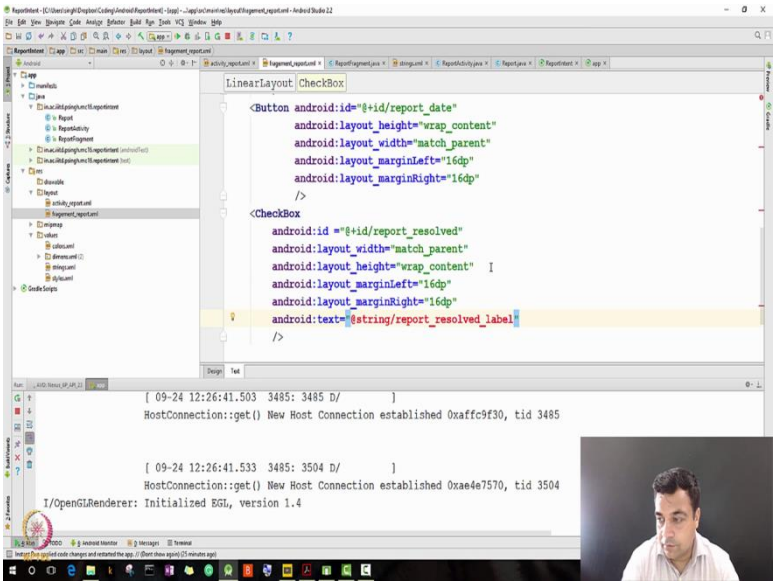
(Refer Slide Time: 3:51)



I am going to change this to isResolved setResolved, so I could directly generate my getters and setters functions and now I am saving it. The second step that I will be doing is updating the layout activity class. For updating the layout we are going to add two text views a Button and a CheckBox, so let us gets started on that first. So we opened the fragment_report layout xml file we already have our EditText here, because we are using the vertical, I will add one text view on the top, so let me add one TextView on top, TextView we have the height, height would be wrap_content, as you can see we have a width, width would be match_parent as you can see we will have the text android:text which let us say we will have to define a new string/report_title_label we will define it in the strings later on.

Now I am going to use something else as well which I have not used earlier, I am going to use style, and today we will be discussing in the lecture more about style and how to make good layouts. So we start with android I go to the, because we are creating a list I will go to the listSeparatorTextViewstyle and that is where we will end our TextView. So we have to add a string, let us remember it and add it later. Now we are going add another TextView, so let me just copy it put it here and width is fine, height is fine the text will obviously change and I will call it report_details_label and even the style is fine, and then I will add a button and a CheckBox.

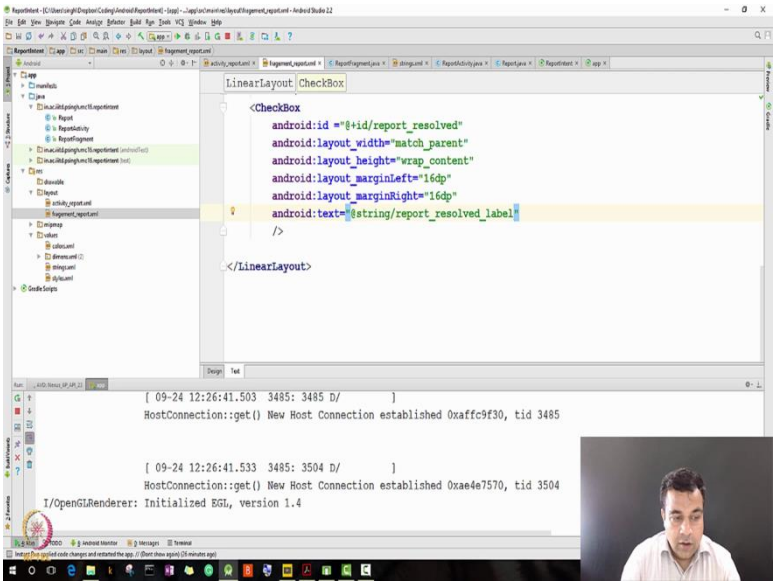
(Refer Slide Time: 6:37)



So let me add a Button android:id =, so as explained earlier to you we use id when we want to use that in our program. So by giving the id you can be sure that I am going to use it in the program and I will call it report_date and then the usual android:layout height and width I just copy them here, and then I will also add some margin because otherwise my Button will look just very close the bottom layout_margin and I will give it, in fact I will give it layout_marginLeft I will give it 16dp and android:layout_marginRight 16dp.

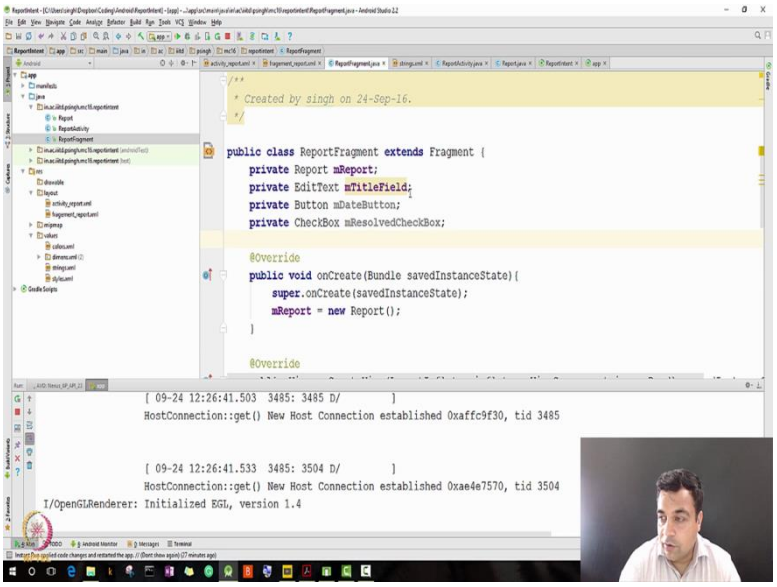
This ends my button, then the last item in our layout which is the CheckBox, let us add a CheckBox as well, CheckBox width so again match_parent add the wrap_content, we do want to give the CheckBox also an id, so we go here we go @+id slashreport_solved or rather call it resolved as we have been called in it and then we are going to give the same same margin so I will just copy that and then we would like to use an android:text string/report_resolved_label one two three, so we have to create three more strings.

(Refer Slide Time: 9:56)



So let us go and create three more strings in our strings file I will just copy one let us say v, v, v let me just change the name, report_title_label and this is (())(10:05) Resolved and you just give it the name, so this is an important step, but not always the most interesting part, report_resolved_label. After defining these strings ideally our layout file should not have any red items, yes it does not, and this is all very good. Now, let us go to the ReportFragment class and start making use of these new widgets that we have added.

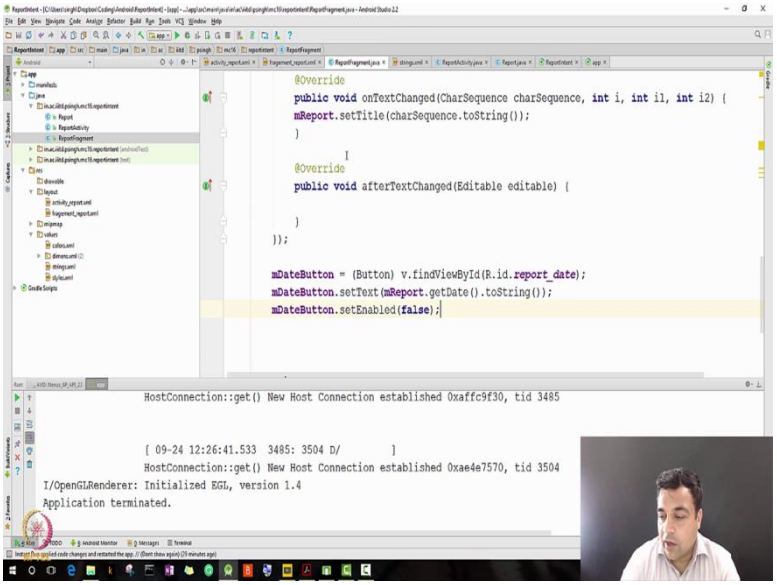
(Refer Slide Time: 11:36)



So I will go and, if you remember we created ids for only two so we are going to use only two of the widget in our program and add a CheckBox mResolved Resolved Button I would say Alt Enter.class yes, everything is fine. Now we will have to make you make use of these

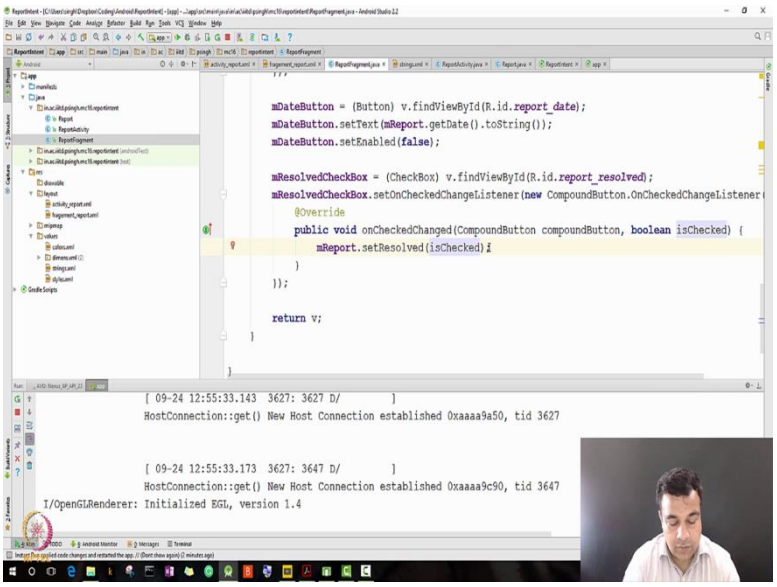
two new widgets we will go the onCreateView method which we earlier went and before return we will have to add some code so that we can make use of the Button and other widgets.

(Refer Slide Time: 12:34)



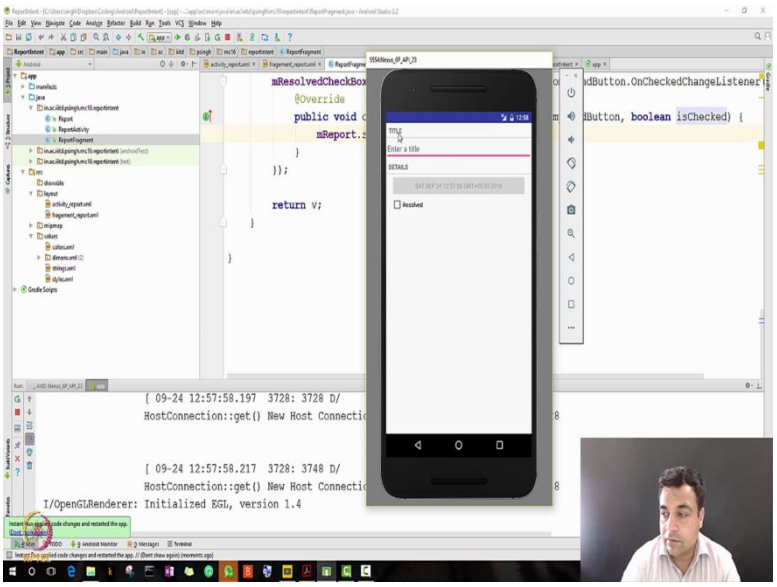
So mDateButton = Button and we will do it by v.findViewById our id was R.id.report_date for the Button and then I do mDateButton.setTextm.m, so whatever the date that we have created in our report file, so mReport.getDate.toString because this will take a string and then we will do mDateButton.setEnabled false, currently we are setting it as false, let just quickly run our program and see what happens uhh.

(Refer Slide Time: 13:45)



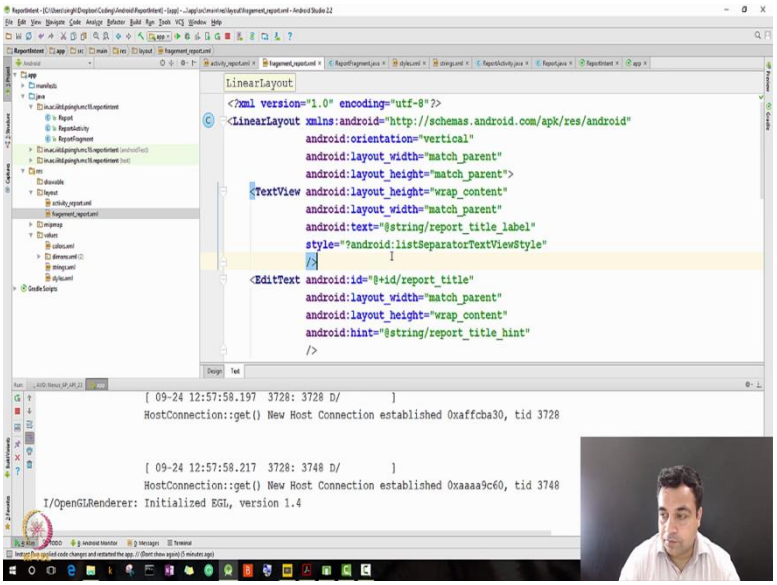
So now you see that more details have come our date is being displayed here, but the Button is disabled because we set it as disabled. Now the next step is to add the code for the CheckBox, so mResolvedCheckBox = CheckBox v.findViewById R.id.report_resolved that was the id for our CheckBox, if you are missing it then you can see here that was the id for our CheckBox report_resolved. So everything is fine report_resolved and the next thing that we are going to do is to set a listener here mResolvedCheckBox.setOnCheckedChangeListener that is, whenever we toggle our CheckBox to checked or unchecked new OnCheckedChangeListener once I add this android already fills in a lot of code for me onCheckedChanged and in this only thing I want to set is that my report.setResolved is similar to the Boolean variable which is given here, let me change the Boolean variable name to isChecked, now just say isChecked.

(Refer Slide Time: 16:07)



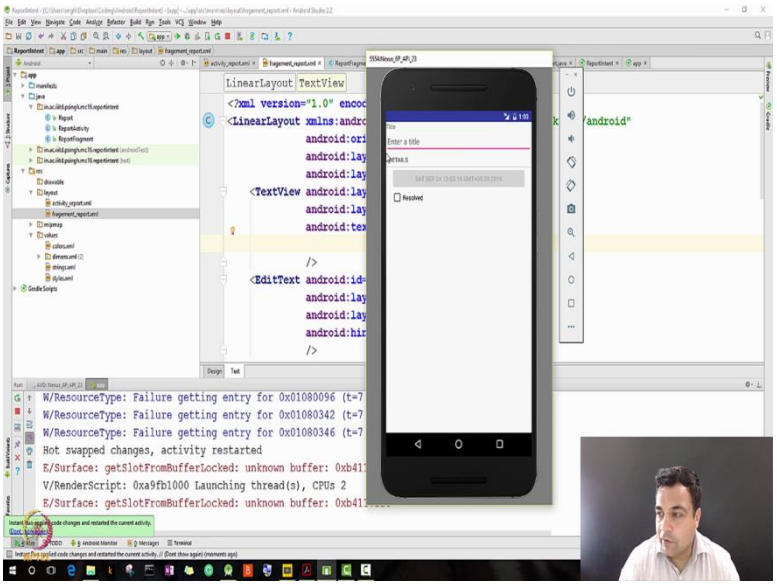
So this is fine, now we have added all of our widgets into our program and as we can say that all of them are now visible to us, so here we go all of them are here. So this is our very very basic UI for our application we will keep improving it because we do not want to add just one item, we want to add multiple items and then we want to see details.

(Refer Slide Time: 16:41)



Now, in this program we have used especially in the fragment report we have used some new things. So let me explain them to you, so one of the new things is this statement called style, and then a value was given. So let us for the time being remove this statement, let me run the program.

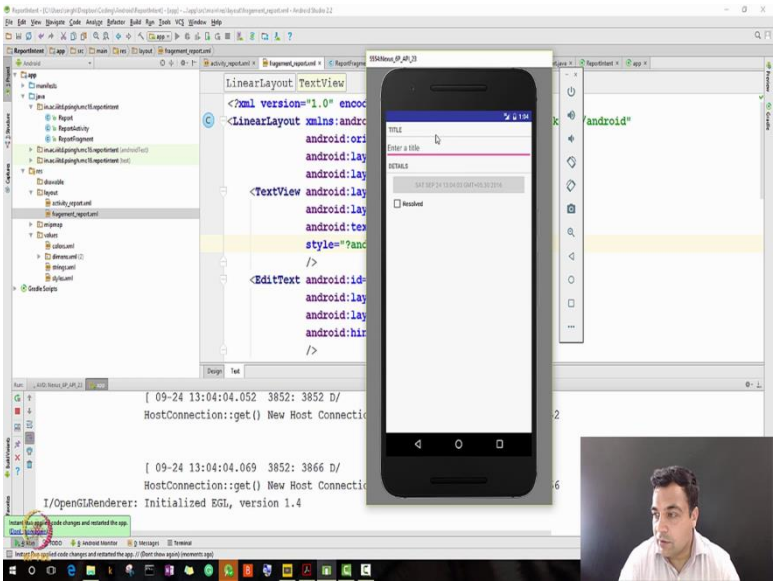
(Refer Slide Time: 17:02)



Now you see that earlier there was a kind of separator line coming here which is now no longer coming. So that separator line is still visible after details, because if you look at it and we are using we are using a style after the TextView of the details and that is why we can see this almost invisible part a very distinct line, which was earlier also after title but now it is

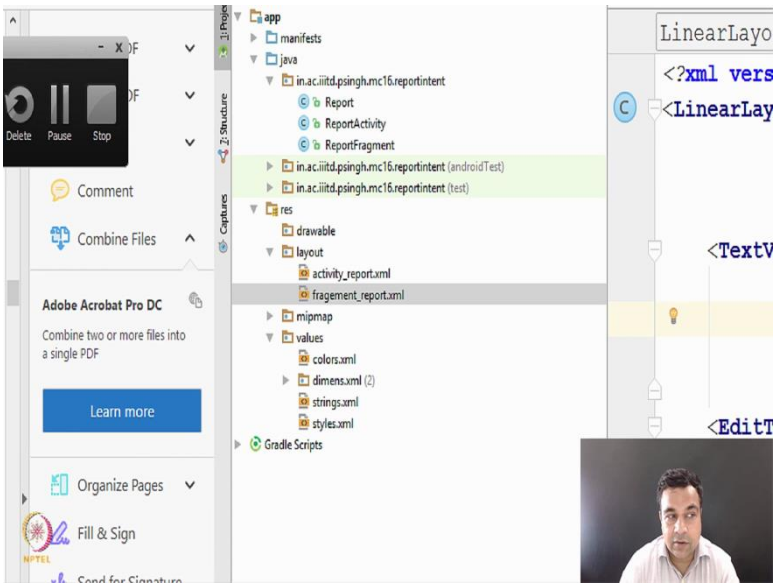
not. So this is what the style is let me, let me put it back stop, and run again so that you can see it very clearly, so yes, you see? This is the line.

(Refer Slide Time: 17:45)



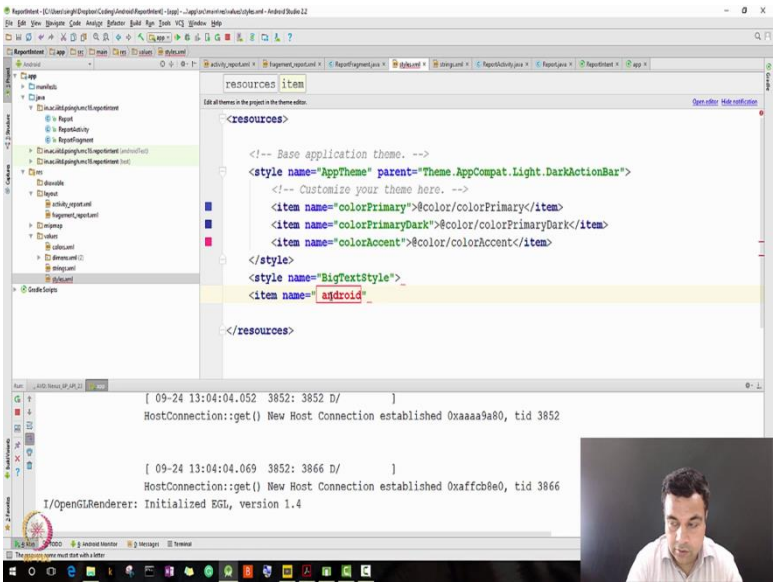
Now what is this line? So this line is actually a part of the style so it is giving a view of a separator. So what are styles? So styles are nothing but xml resources that contains attributes that describes how our widget should look and behave. So this is just same you have some xml resources here and it is similar to those xml resources, in fact if you look into the value let me, so if you look here in the values you will see that there is a file called styles.xml.

(Refer Slide Time: 18:25)



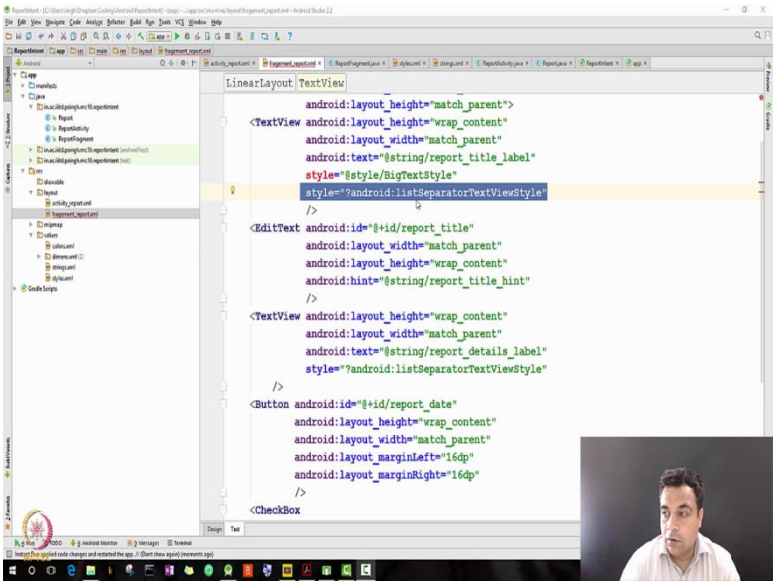
So this is the file that has your styles, this is the file that has your styles and you can click on it you can see some values, and in this style actually you can define new styles as well, so there is no need to just confined to what android has given to you, for example you may want that your text is displayed bigger, so in that case you may want to define a style called let us say new style let us put it here as sorry new style which I will give a name and my name could be just a BigTextStyle, will I want bigger text and I will just give it to a item.

(Refer Slide Time: 19:42)



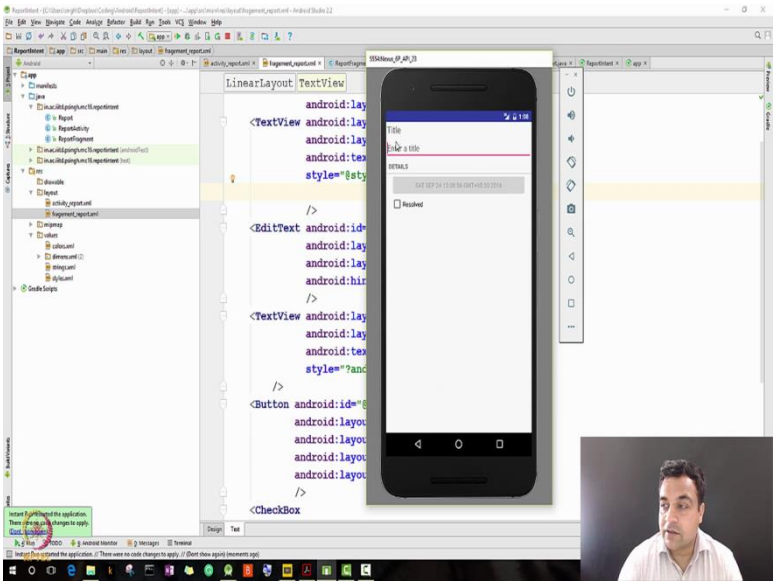
I will give it a item name like given here, I will give it an item name android:textsize and let us say 20sp. What sp is I will explain, and then I would say you know I would also give it some padding, so I will it some padding, so item name = android:padding 3dp and I go to and I finish my style. Now I have created a new style, which is my style and this style will display the text bigger and whenever I want to use my style I can use it just the way that I have been using the style given by the android.

(Refer Slide Time: 21:31)

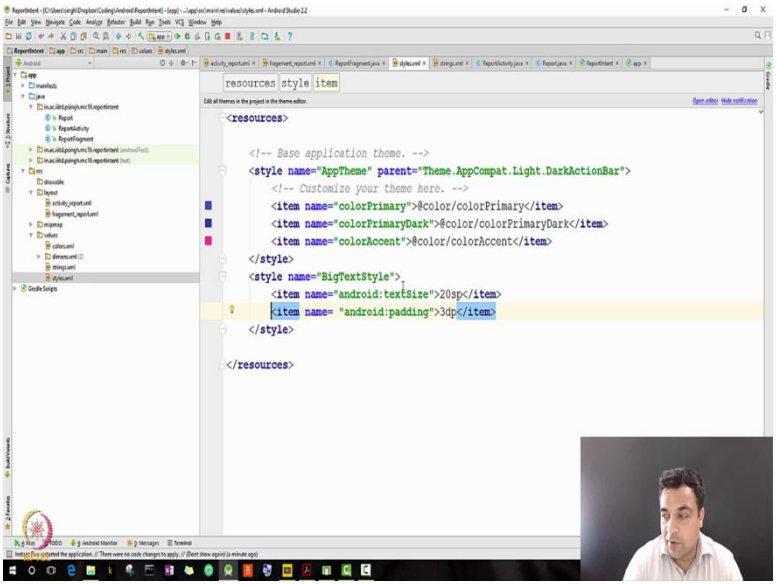


So, we were using a style here and a style here, we can simple use our own styles, you can say let me try a style, style = @style/BigTextStyle. So this all fine, I am now using my style, so let me close the second statement, because it will confuse. And now let me run the program to see how does my style looks like when I newly created this style.

(Refer Slide Time: 22:03)



(Refer Slide Time: 23:20)



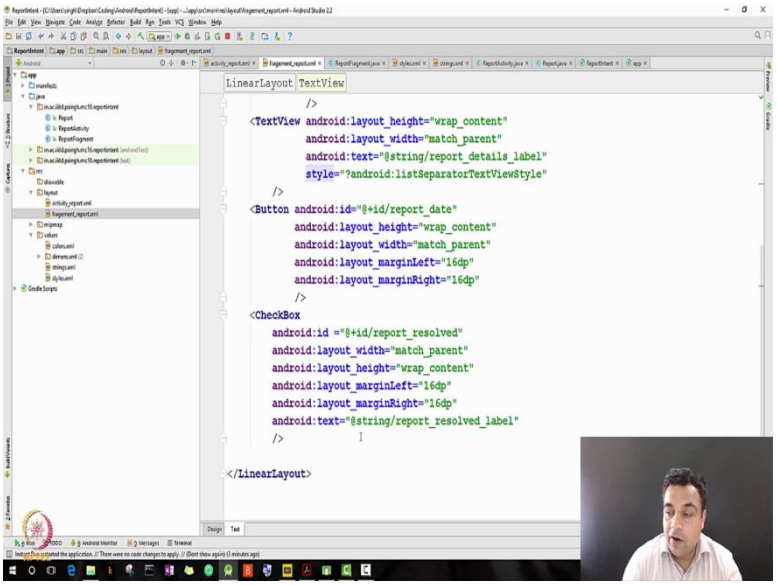
And as you can see that text size has increased, so this is all about styles you can create different types of styles that you want and then you can use in your program as you wanted it. So I will just remove it because we want to go back to our original style, which was the style given by the android. So android already supplies you lots of style and then it also supplies you with the capability to create your own styles. Now the second thing that we have been using in our program is, for example this values like dp and values which I just used, for example sp, so let us see let us discuss what this values are and hence I explained earlier that this are nothing but different units described how big your, for example in this case your text look like, so sp, stands for scale independent pixel.

So the scale independent pixels are density independent pixels that also takes into account the users font size differences. While dp, stands for density independent pixels and you will typically be using it for margins and paddings etc and for anything for which you would otherwise specify size with the pixel and so this is kind of giving you a pixel value size while this is giving you more like a scale independent pixels. The other values that you can use are pt, mm, in, where pt is points which is roughly equivalent to 1 by 72 of an inch, a millimeter is a millimeter mm and an inch is an inch.

So there is no need to get confuse, and if at all you ever get confused so just write a unit and run your program to see how it is acting. Now till another important thing that I want to discuss is that you may have seen already that some this parameters starts with the layout and some of this parameters do not start with the layout. So if you have not guessed already that

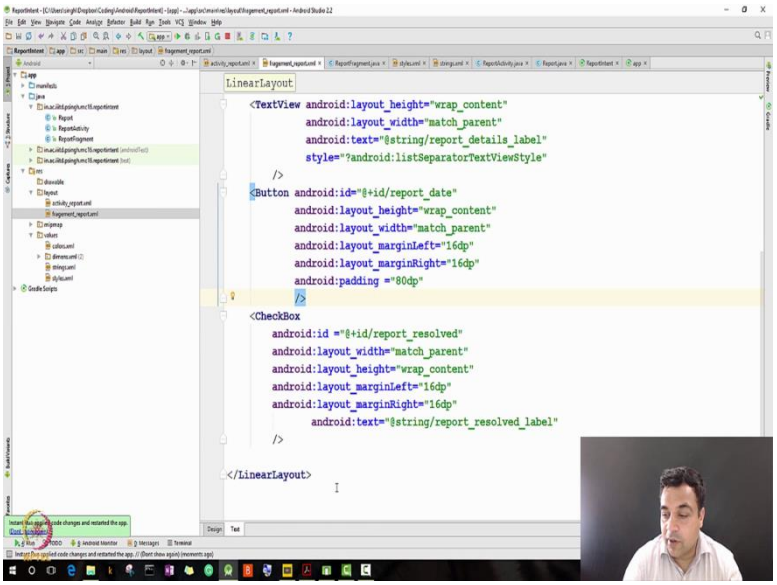
the parameters which starts with layout corresponds to the layout, while parameters do not start with layout corresponds to the individual widget.

(Refer Slide Time: 24:35)



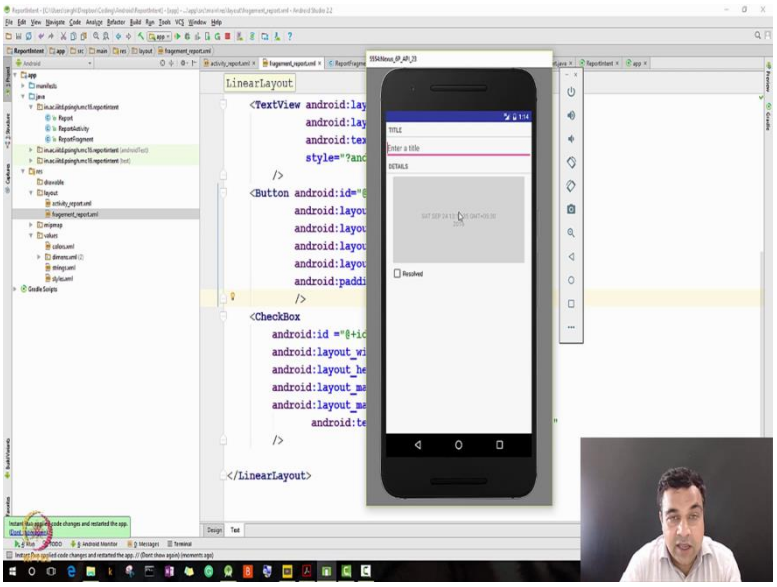
So, for example, in this case when I am saying the let me go back, let, this is a good example, when I am saying that I want the margin, I am actually giving direction to the layout that this element CheckBox should be displayed at a given margin and you will soon realize that some of these widgets you will have to add with some margin, while in case of padding, I am giving instructions to the individual widget that how it should display itself. So the difference between the margin and the padding is that margin is an instruction given to the layout, while the padding is an instructions given to the individual widget. So, we have already seen the that what the margin does so we can already see what the margin does, so here you can see that in this is slightly off from the very beginning, now let me just for the sake of experiment add a field called padding here and see how does the padding impact, and that will also make clear to you what is the difference between margin and padding.

(Refer Slide Time: 25:53)



So, we go, we define a padding of 80dp, currently our application looks like this, we will now restart it and you will see what this padding has done. So we are adding the padding to the CheckBox how, okay, no resource identifier, let me, this was probably the wrong ways to add padding, let me add the padding to the Button and run it okay let me add what is the, oh so our spelling is wrong and as you seen there was no error now let us run again any way you wanted to add the padding to the Button or the CheckBox.

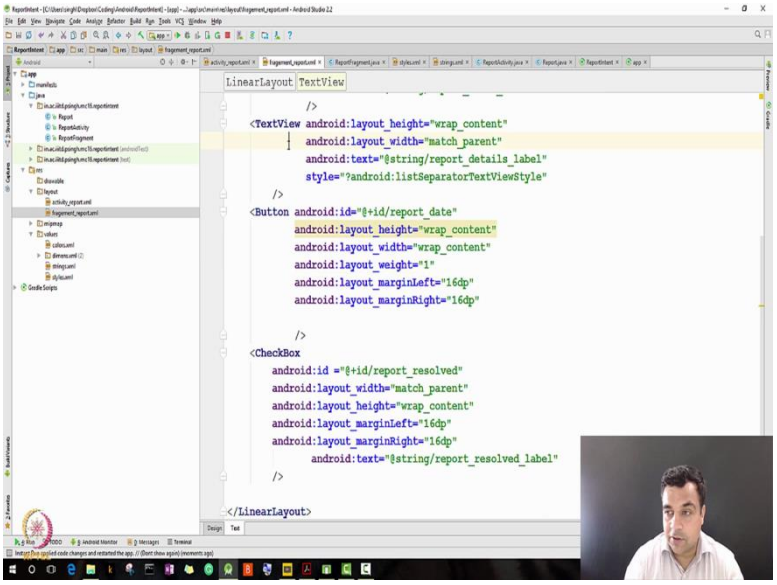
(Refer Slide Time: 26:57)



Now you see, now the button has displayed itself in a much bigger form and that is what the padding does. Padding is an instruction to the widget that how it should display itself while margin is an instruction to the layout that how it should place a particular widget. So when I

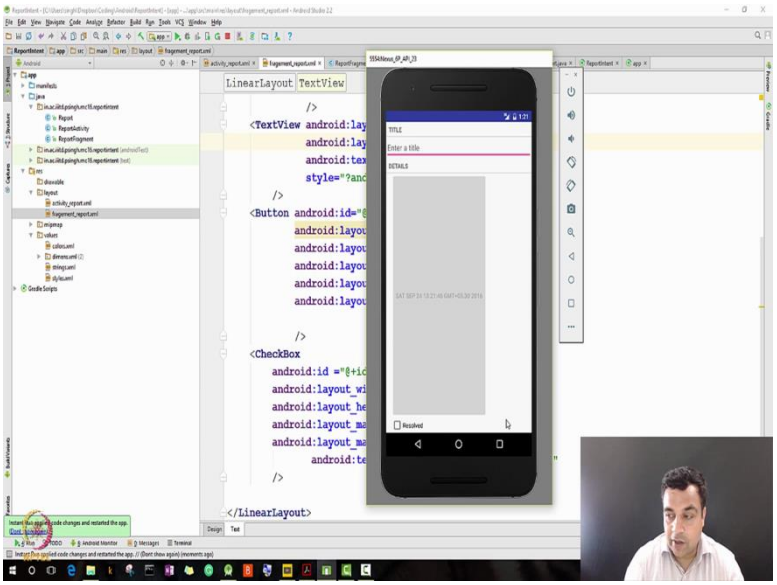
am saying that the margin left is 16dp is making sure there is a difference between the left, between the left side of this to the border of it, but when I am saying padding, and the padding is to the button and now the button is inflated giving padding size and that is what the difference between the padding and the margins.

(Refer Slide Time: 28:11)



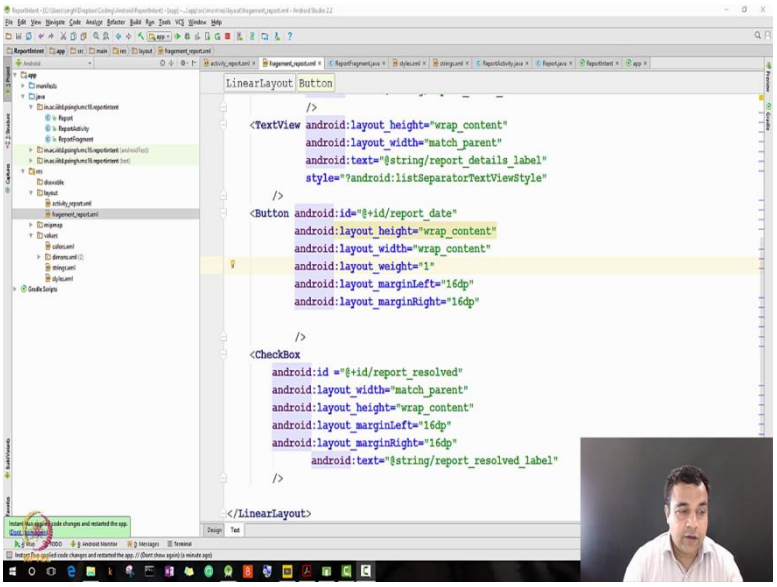
Let me remove it now, save it. Now another important thing that we need to see is the another important attribute that we often use is the weight attribute, yeah here, that we use. So let me try to provide some weight attributes and show that how does it impact. So before going to the weight I will have to make some changes to the button and let me just change the width to wrap_content, if I change the width to wrap_content as you can imagine that state of the button expanding to the whole width till now only expand till the content is, content in the button. Now, I have given the width and let me now give the weight to it, this is just for the experimentation purpose later on we will go back and let me give the value one. Now currently you can see that our program looks fine but our program is not occupying the whole screen.

(Refer Slide Time: 28:59)



Now let us see what happens when we give one of its variable the width, so I do this and so now the our UI is occupying the whole screen and every other widget is occupying the screen which they were earlier occupying in terms of area but the button widget has increased so that it occupies the rest of the screen space. And that is exactly what is the use of the weight variable, so in android the width and the weight works together, they work together to define how something is displayed.

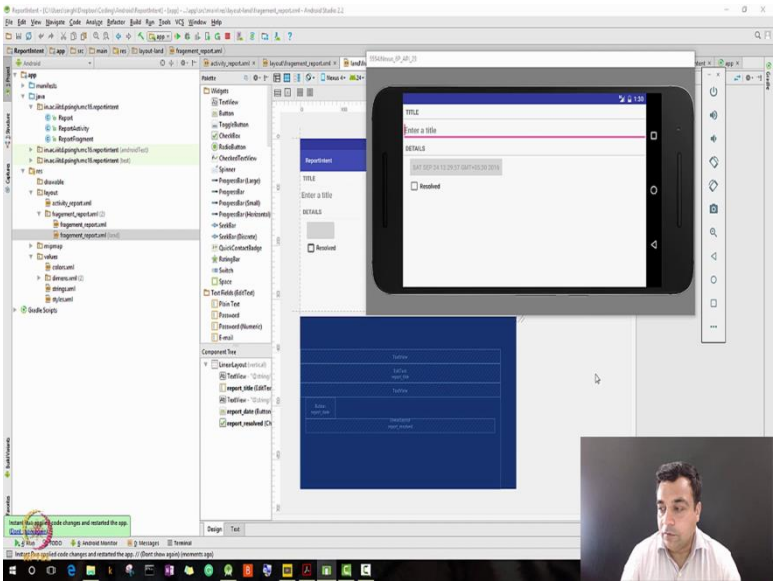
(Refer Slide Time: 29:45)



So the in the linear layout will make two passes we set the width of the view, in the first pass it will look only at the width and then it will in the second pass it will also look at the value of the weight. Now just for the argument let me give the weight and to show you how there are

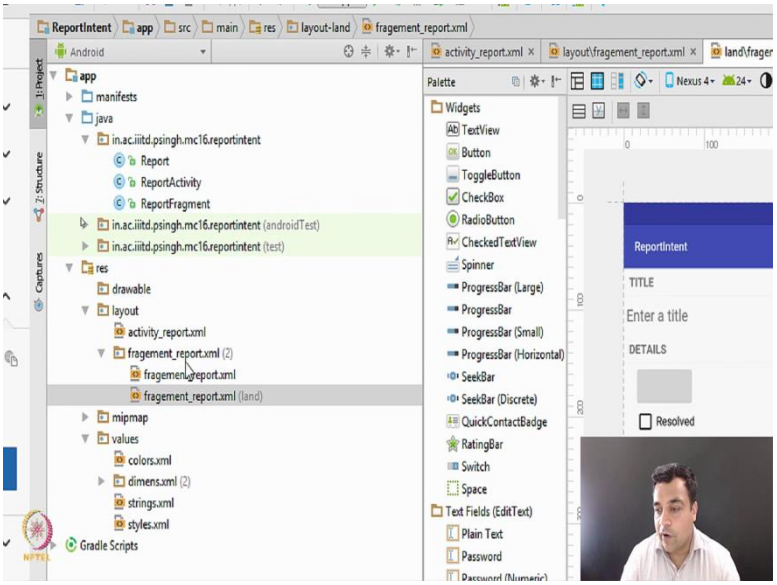
playing to each other, let me give it two here, and let us try to see what happens now, yeah and now you see that the resolved is occupying twice the space of the matter because one is two and resolved is CheckBox is two. So that is how the weight plays out with width and I have also already explained how the style, margin and padding work out. So you can make use of all this and create your layouts. Okay, now let us also see how to use the graphical tool of android to do some manipulations, thus so far we were doing the manipulation only in the xml.

(Refer Slide Time: 31:10)



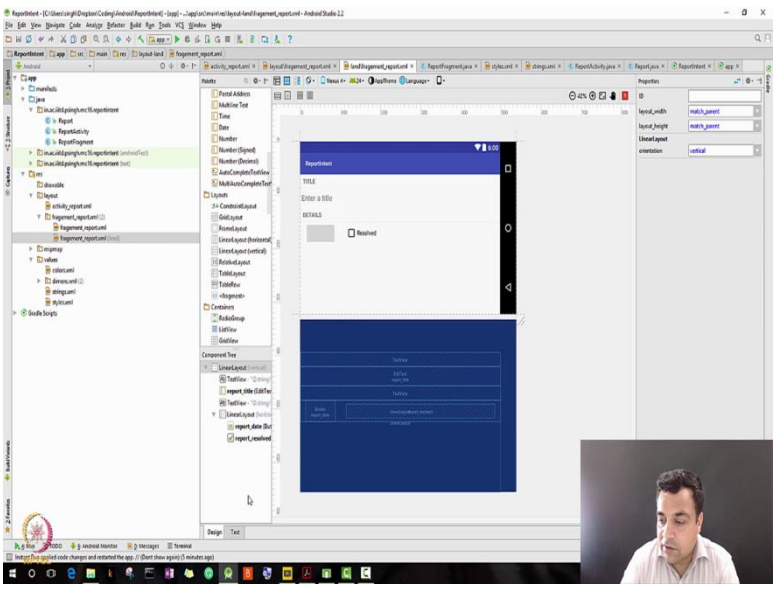
Now I have converted my layout to the landscape layout currently I have just converted it as it was in the portrait it is same thing as being displayed in the landscape, so in the portrait and in the landscape. But this is not looking good to me so I would like to actually use a different layout file when my application is in the landscape. So you can see that this button if you click on this button, then basically this generates you a different layout file. So now I have got two layout file here, let me magnify it for you.

(Refer Slide Time: 31:42)



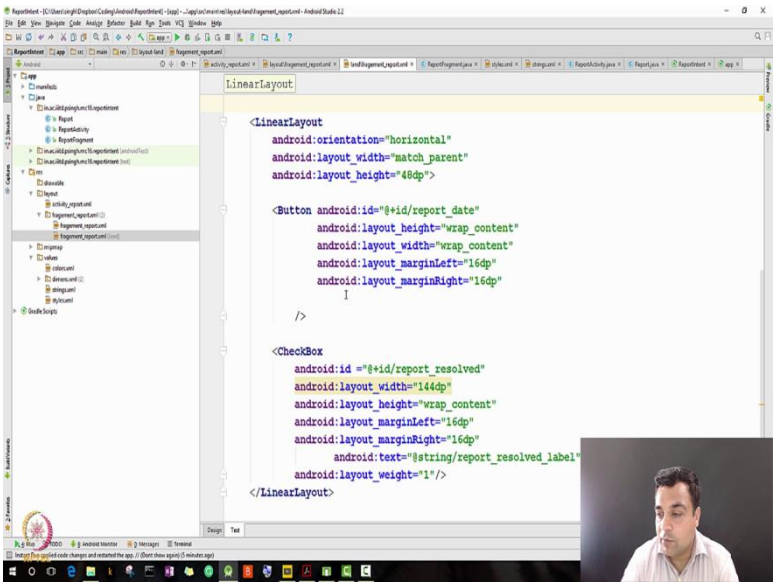
Now I have got two layout file here, one is the fragment report which I originally created myself, and one is a fragment report.xml(land) which refers to the landscape mode. Currently both of the files are same and it is just as good as turning by on screen, but now I will change the land file to have a different view and I will make these changes using the graphical tool of the graphical tool of the android, instead of making change in the code. So let us go back, again I open the landscape file I see the text this is currently everything is same, but now I will be making some changes, so one change which I want to make is that I want to bring this button in parallel to the to my bring my CheckBox in parallel in line with my button.

(Refer Slide Time: 33:10)



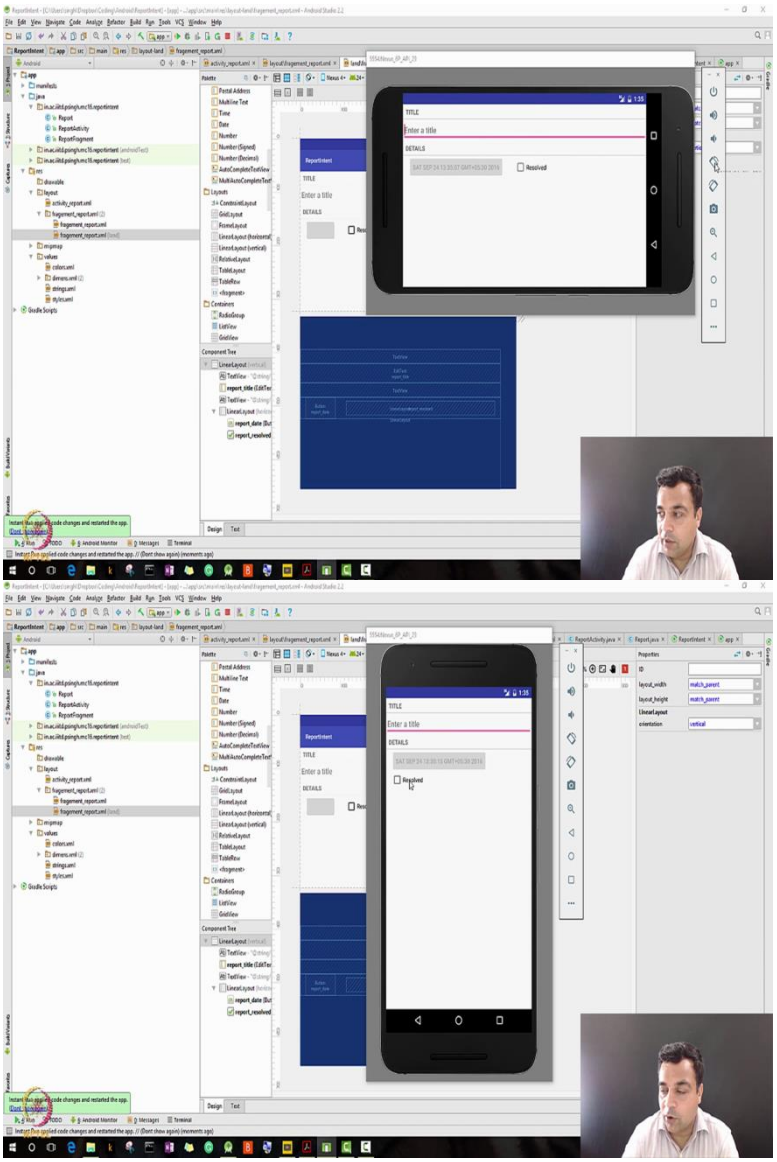
So, this is a very simple change, let me just use the graphical tool and hopefully I should going to move it now, so this is my button that is fine I am just trying to move my resolved CheckBox, okay let me go to the text file and actually add, let us say a linear layout here but we can first we also do it here as well so I want to add a linear layout here and report_date, report_title this is my widgets sorry so I was making a mistake I should take the linear layout here horizontal and I should add it add after the details. I add my linear layout and then inside that linear I added my linear layout, inside that linear layout I will add my button, I will add my button and I will add my CheckBox, so yes , so I am not using but let us see what change happens in the text file.

(Refer Slide Time: 35:23)



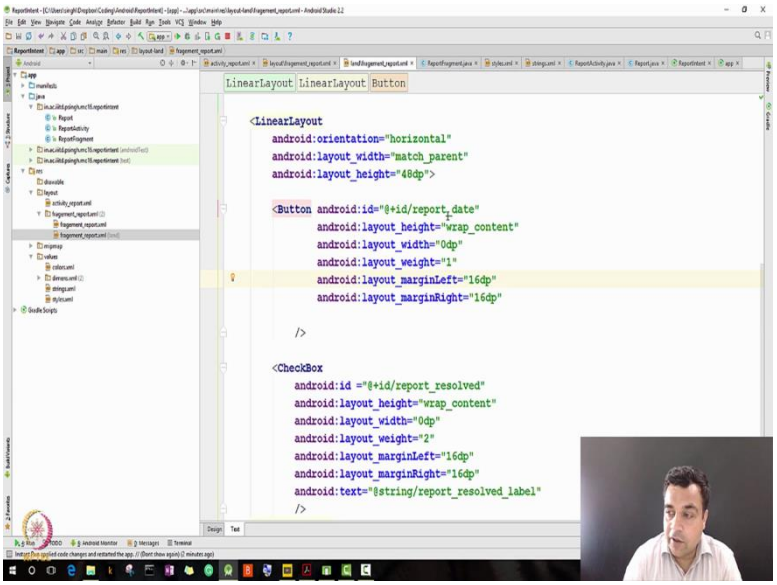
As you can see in the text file now a linear layout is added just before the button and CheckBox, button and CheckBox are arranged in the linear layout of orientation horizontal. So this we have done using only the design, and if we run the program, then our program should behave differently when it is rotated.

(Refer Slide Time: 35:45)



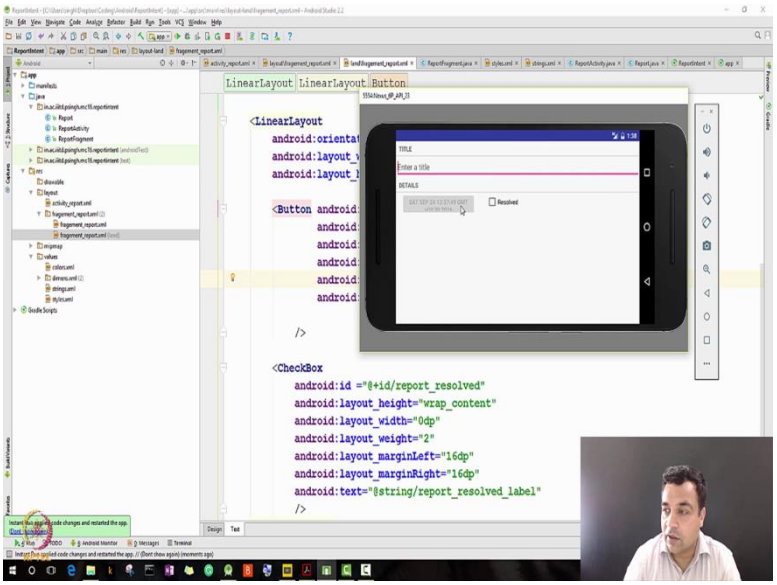
So now see that in the rotation it is showing me the CheckBox side by side and while in the portrait it is showing the button up and then down is the CheckBox.

(Refer Slide Time: 37:20)



Now let us also again see what we were doing earlier with the weight, and this is actually a very good opportunity, till now I would say android:layout_weight = 1 android layout, let me delete the width and just use the weight = 2, also I delete use it just once and let me delete the width from here as well. So earlier you were seeing that it was giving us a warning so I will not delete it, I will just make it 0 and I will just make it 0 okay 0 dp and 0 dp. I am not getting any warning I have converted the width to 0 but I have added a weight which I have given 1 to the button and 2 to the CheckBox, now let us see what happens.

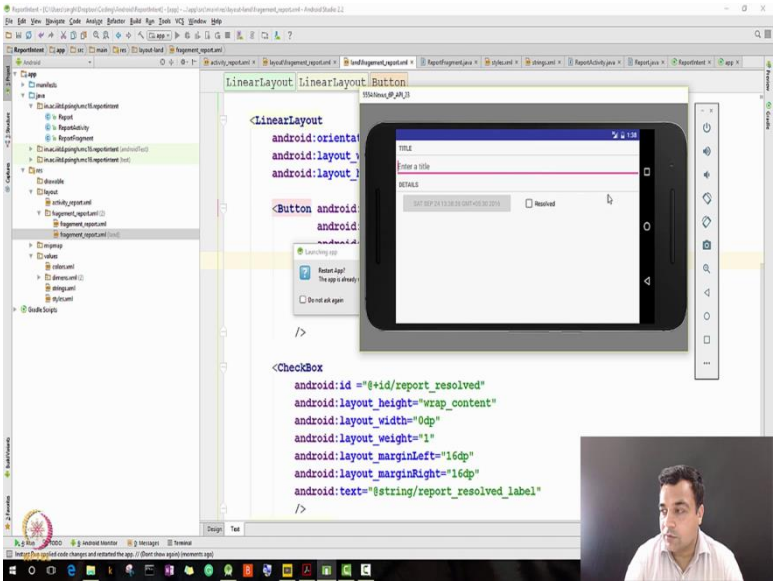
(Refer Slide Time: 38:19)



When we run our program everything is fine in portrait but we never made any change, now do the landscape, now you see that because the weight is double for the layout checkbox, the

CheckBox occupies two third while the button occupies one third. Let us make a change here, want a bigger button you want a smaller CheckBox, let us run it, so let us run it and this time the width is twice and the CheckBox is one third.

(Refer Slide Time: 39:07)



And similarly as you may have guessed if I give it 1, 1, and then both will occupy equal space, so both will occupy equal space. So, this is it, so this is it in this lecture you have seen how we use the graphical layout or we learned about styles, themes, margins, paddings and how to use the graphical tool to change the layout. So now let us go back to our program very quickly, so now we have we still have only one fragment and we have one activity fragment activity, but now we have got 2, now we have got two files to refer to our fragment layout, one when it is in portrait mode, one when it is in landscape mode and our another layout file is called main activity. So that is it, in the next lecture we will develop on this application and make it even bigger application with more functionality, Thank you.