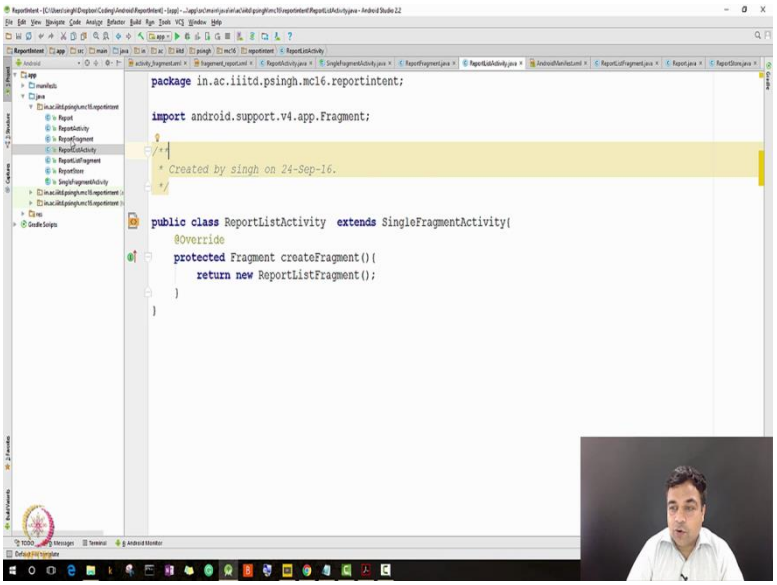
**Mobile Computing**
**Professor Pushpendra Singh**
**Indraprastha Institute of Information Technology Delhi**
**Lecture 36**

Hello, so we will continue our lecture on developing a larger size application using fragments which uses many other concepts. So far in the code you have only seen what we had already discussed in the lectures, that is fragments, etc. Today, you will see something which is new it is called RecyclerView. RecyclerView is used when you want to display lots of items on a smaller screen, so let us get a started with our recycler.

(Refer Slide Time: 0:50)

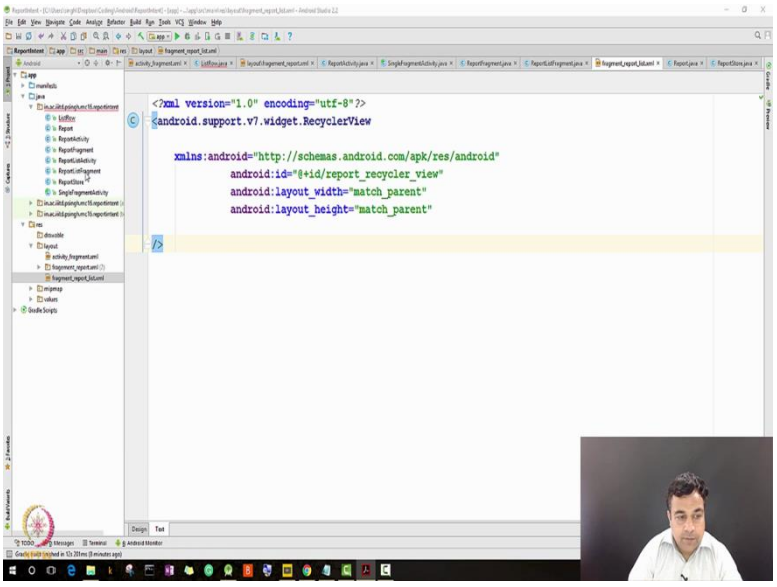



So just to remind this is our previous code. We have got multiple files ready here, one is the ReportListActivity, one is ReportFragment, and one is ReportListFragment which is currently doing nothing. And, then one is our SimpleReportFile, then a ReportStore and then we also have a abstract class for SingleFragmentActivity. We also did few layouts, so our first layout was this activity fragment layout which is nothing but just a container. And, our second layout was the layout of the individual report item. So now let us move forward and start developing something new. So now essentially what we want is that on our main display we want to show a list of items. So I would like to report here, report number 1, report 2, report 3, report 4, report 5 and things like this.

Now we want now we can do that for example we can just replicate this all over the place, let us say just a title of it. However, the problem in that approach is that if we have 100 element and we will have to display 100 patterns, which basically means that we are making memory for 100 text views while actually the user can only view the tittles which are currently visible

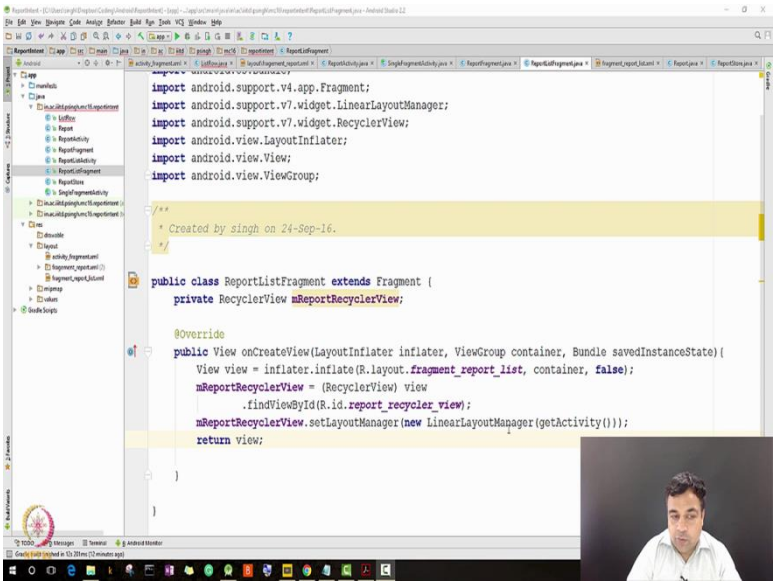
on the screen that is that fit the screen size and they may be 12, 20, but definitely not 100. So the recycle view solve this problem by reusing what has been created earlier. So let us get started with our RecyclerView and we will also create an adaptor and we will also create a view holder. So yes first things first let us do the program and then we will come back again, we have a fragment and a SQL report.

(Refer Slide Time: 3:33)




Now we let us create a Fragment_Report, now go to the text and first let us remove this part hmm and I will add some of my code here, android.support.V7.widget.RecyclerView, then we can have actually we no its going back the previous code. The only thing I am going to change is that the LinearLayout. I will make it android.support.v7.widget.RecyclerView and then we can simply write. Now xmlns http schemas android.com/apk/resandroid, we leave it as it is layout_width match parent, layout_height match_parent, we leave it as it is. We will just remove this line and instead we will add id because we are going to make use of it, so we need to create and id for it +id/slash report_Recycler_View yeah, so now our now our layout file is there, we will now write the code file.

(Refer Slide Time: 6:25)



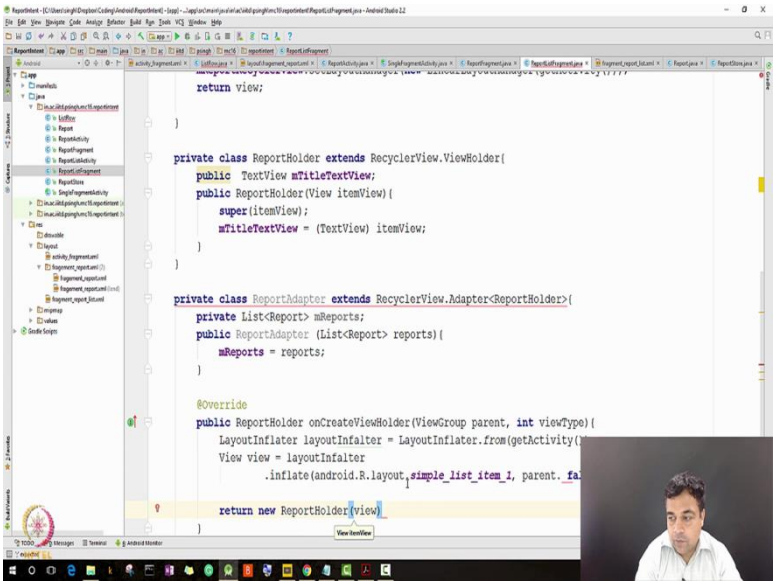

So we created a ReportListFragment earlier used to extend fragment. Today, we are going to add some code into it, so private RecyclerView and RecyclerView and we have to only override one method, which is PublicView onCreate view layout inflator inflator, view group container, bundle saved instance state, that is it we have to press Alt Enter. Now what kind of view do you think we will we will call later here. Well the one which we just created, so let us say View, view, inflator inflator.inflate R.layout.fragment report list, container false like last line. And let me solve it mreportRecyclerView = RecyclerViewview.findview by idr.id.report RecyclerView, then mreportRecyclerView.setLayoutManager new LinearLayoutManager getActivity.

After that we return view which we just created. Now as you will see that the moment we created a RecyclerView we gave it a linear layout manager, so the RecyclerView always requires a linear LayoutManager. If we do not pass a linear LayoutManager into it will crash. Now let us move forward so here in our case the RecyclerView responsibility is to recycle the text view that is the title of the report and to position them on the screen, but the RecyclerView does not do the job of positioning items on the screen itself that is passed down to the LayoutManager that (())(10:23). So this LayoutManager over here will handle the positioning of the items and as well as will define what happens if you scroll. So if the LayoutManager is not there as you can see that RecyclerView will simply crash.

Now, there are many built in LayoutManagers in android but we are using here linear LayoutManager which will position our list vertically because that is how we want our list to go vertically, if you want it into a grid then possibly you can use that grid layout or if you

want. So, it depends on the requirements that you have and based on those requirements which use the LayoutManager that you are going to use.

(Refer Slide Time: 11:34)

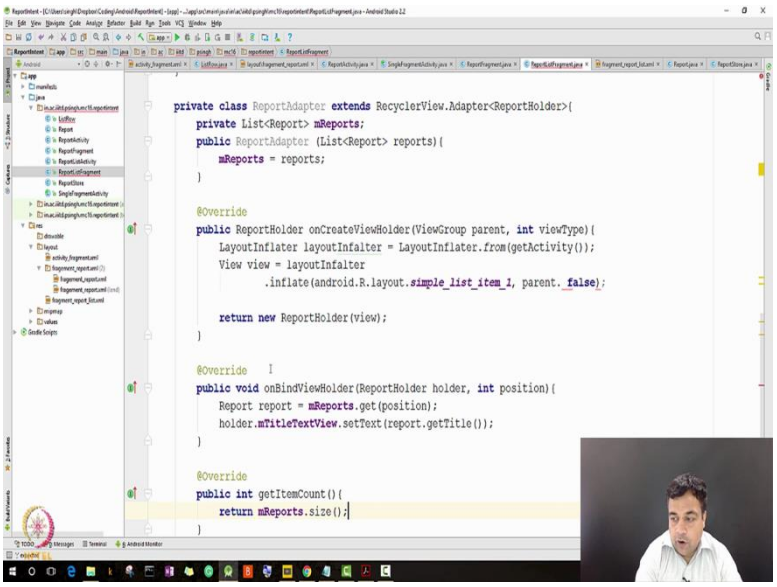



Now let us start defining the viewholder and we will do this as a inner class of our ReportListFragment. So we continue adding code so as you know from your java knowledge that we can define inner classes and that is what we are doing here. So report holder will be an inner class which extends RecyclerView.viewholder and we just do some basic development which is declaring a basic text view that we want so we give the name mTitleTextView and then we develop public ReportHolder View itemView, super itemView, mTitleTextView = TextView itemView. Now we have created a viewholder a very simple viewholder as an inner class. So currently our viewholder is only working with a TextView which will be = the title of the report that we wanted to display, so this was (())(13:12). This is the title that we are capturing to correct.

Now after doing the ViewHolder the next item is to build the adaptor. So we will keep working in the same file and we will build the adaptor as another inner class. So if you remember in the earlier chapter, I had asked to read about inner classes and today you can see the use of that in android programming. So, this is our adaptor and this will be this will hold to the ReportHolder, so to the ReportHolder so you will have a list list of report to pic from Reports, then we will have a ReportAdaptor reports mReports = let us say whatever we are passing. Class must either we declared abstract or implement abstract method on point ViewHolder.

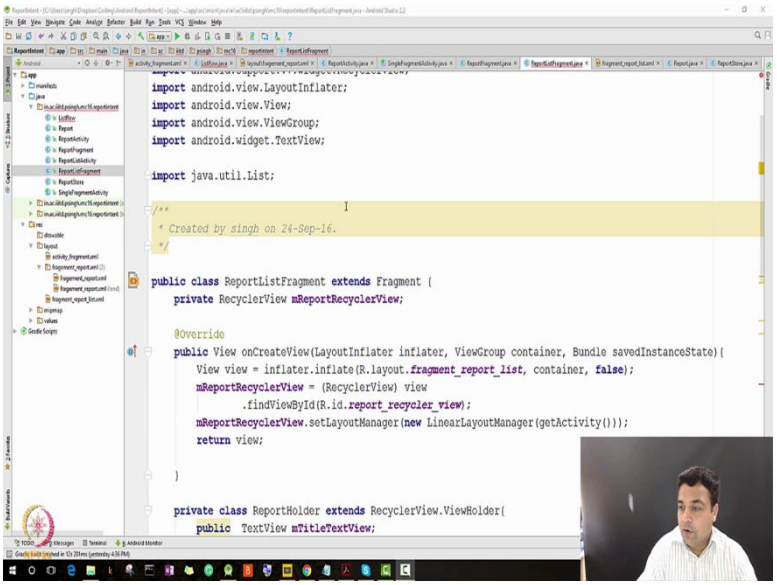
Ok, so let us hold on right now. So as you see there is an error, so this the code will not compile, this is nothing to worry we will add some code more, some more code here. But let us try to understand that what we are doing. So, now our RecyclerView will communicate with this adaptor when a ViewHolder needs to be created or connected with the report object. The RecyclerView will not know anything about the report object but that adaptor will know about the reports object and the details of the report. Now, let us implement few more methods in the report adaptor. So add override, public ReportHolder, onCreateViewHolder, ViewGroup parent int, viewType, LayoutInflator, layoutInfaltor is = LayoutInflator getAactivity. View view layoutInflator or view = layoutInflator.inflate (and) android.R.layout.simple list item one, parent false and we will return the new ReportHolder View.

(Refer Slide Time: 18:56)




The second method which we are going to implement is the on bind ViewHolder, that is a this is a mandatory method. Public void onBindViewHolder ReportHolder holder int position, Report report = mReports.get position Holder.mTitleTextView.setText report.getTitle. And the third method which we are going to implement is a simple get item count, @ Override public int getItemCount return mReports.size so, now our adaptor is a almost complete ok, now our implementation of the ReportAdaptor class is almost complete and we have done a lot of work in this class, so let us quickly go through that what we have been doing.

(Refer Slide Time: 20:52)

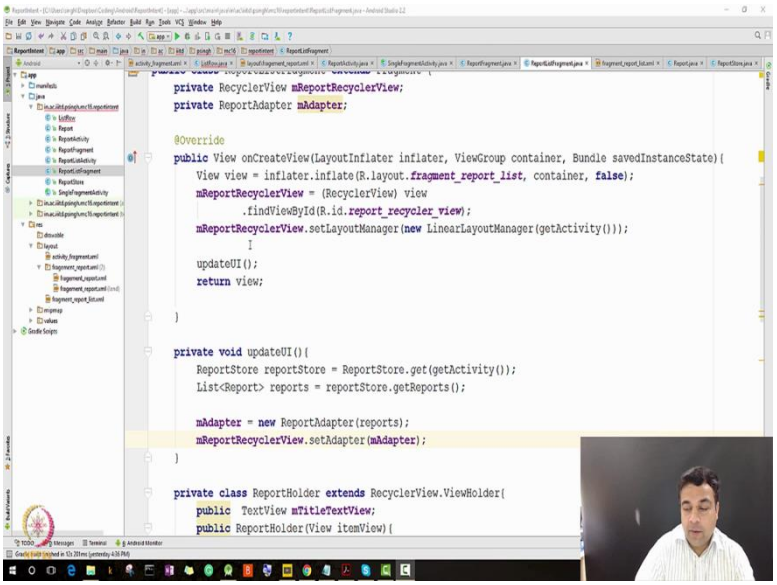



So, let us come first to the method called onCreateView as you can understand that this method will be called when we create the view. So this so it will inflate the view that we have created recently the fragment report list, this view and that is the job of it, it creates a view it does the find view id and then it uses the LayoutManager as I explained earlier. The next important work that we did was in the class private class the inner class report adaptor.

So in report adaptor we are first using this method called onCreateViewHolder. So onCreateViewHolder will be called by the RecyclerView, it is usually called by the RecyclerView, when it needs in new view to display an item. And just as you can see that we are using some called simple list item one, so this is an android layout to display list and that is what we are using because this is something which we find as useful in our scenario.

Then the on view onBindViewHolder this method will bind a ViewHolders view to our model object ok, it receives the ViewHolder so it is receiving the ViewHolder here and it is receiving the position and then it binds that. So basically we are in this we are doing nothing but whatever we want to show and because we want to show the title that is why we are using this and then the last one is simply to get the count or the size of the report. And now that we have created our adaptor we will try to connect it our RecyclerView and for that we will be implementing few more methods. So let us start with the coding again, so let us go to our file ReportListFragment which is this file and in this file we will like to first access the adaptor that we have just created.

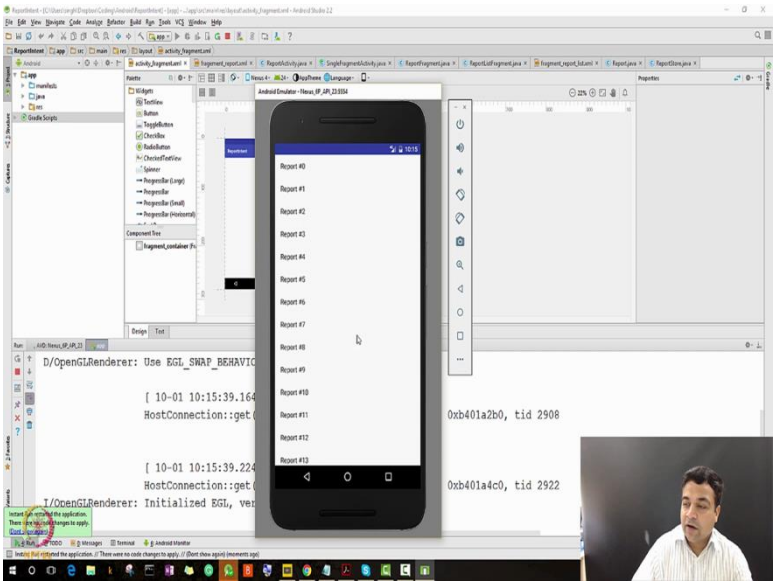
(Refer Slide Time: 23:24)




So, private ReportAdaptor and I will just call it mAdaptor ok and then in the onCreateView method we will actually in the last line we will just call another method let us call it update UI, we will soon be implementing update UI, so for there is no update UI that is why we are getting an error, but we will soon update we will soon implement this method so that we can update the UI. So let us start with updating the method with implementing the method. So I do private void update UI and in this basically we are doing report, so we are taking our report and what we are doing is, we are doing reportStore.Get getActivity. And then we are doing list report reports = reportStore.Get reports, then our adaptor we initialize it new ReportAdaptor reports sorry uhh.

And we will simply do a call our RecyclerView and we will setAdaptor as our (())(25:20). So that is the update UI method which will now we called in the onCreateViews format. Now first let us try to run our program and see how it looks like. We are still using it in X6P API 23, letter is here and yes.

(Refer Slide Time: 25:45)



Now you see that we are seeing a very nice list of reports that we will be creating, we can scroll through it. Now, let us add some more detail to what we are already seeing. So, what we are already seeing is simply the title and we may want to see more information for example, we may want to see a date and in some cases we want to see a sub title, some cases we may want to see some additional information. So what we will do is, we will now learn that how to customize this view, the view of an individual item.

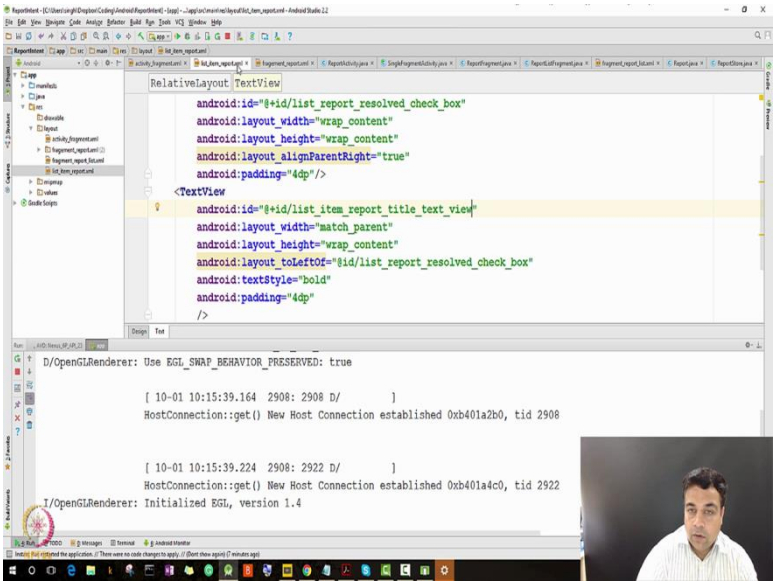
So this is called the item layout and we will be customizing it. So as usual we will create a layout file in the layout, so I go to new create layout resource file let us give it a name list_report list_item_report. Let us choose linear layout as its root element, directory name layout we press ok. And we have got our layout file or basic layput file which says nothing but simply a linear layout; we will be modifying it so that we can put this layout over there.

(Refer Slide Time: 27:30) 29:46

So first let us change it to RelativeLayout, because we want to put elements side by side not always in the vertical. We could have chosen in the RelativeLayout when we were creating the layout file, but it does not matter we can just change the layout here and it will be fine. And now we will be adding 2-3 items to this layout file 2-3 view items, ok. So first thing, that we will be adding is a checkBox, so let us add a checkBox here, checkBox the width is good enough to wrap_content and the height is also good enough to the wrap_content. We will be doing an android layout_alignParentRight and I will set it to true. I will very quickly show you why I am doing it and then we do small amount of padding, android padding = 4dp. And

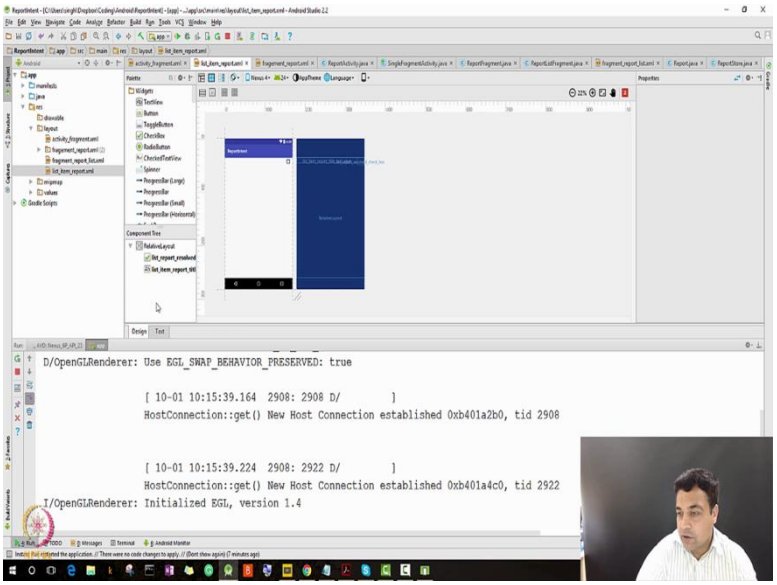
because we will be making use of it that is when we do a check or uncheck, we will have to assign it an id = @+id/list_report_resolved_check_box.

(Refer Slide Time: 29:52)



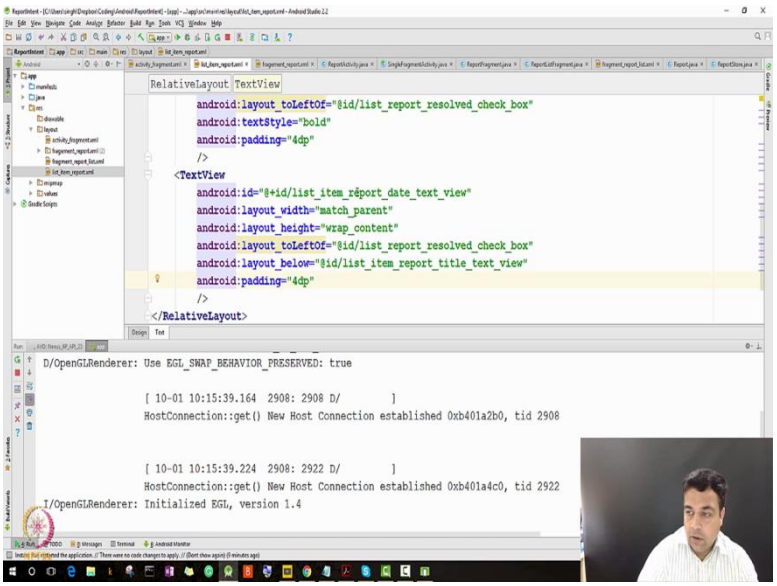

So, let us see now we have got a checkbox, we will have to add also 2 more elements; one is a TextView to show the title. TextView again we will do the width we would like to have it as matchParent, the height is good enough to the wrap_content. Then we will add, so we would like to add it to the left of our checkbox and that is the reason why we are choosing a RelativeLayout. Android layout_toLeftOf and here we will simply give the id of the item we want our current item to be left off. And we also want that is text is bold, so text style = let us say bold. We could have chosen italic as you can see and another option and we will also do some padding here may be of same amount, so that it looks good and I will just say in case, well this is not really essential so we can delete Report Title we can remove that, it is not necessary.

(Refer Slide Time: 32:01)




So, let us have this important item is also again the id we will do id. Android id = again @+id/list/item_report_title because this is the title and this is the textview. So, this is complete, let us see the design as you can see that here our title will come and here our checkBox will come. So that is all right we need to also add one more textview just underneath, so we will copy it and just changed the required values.

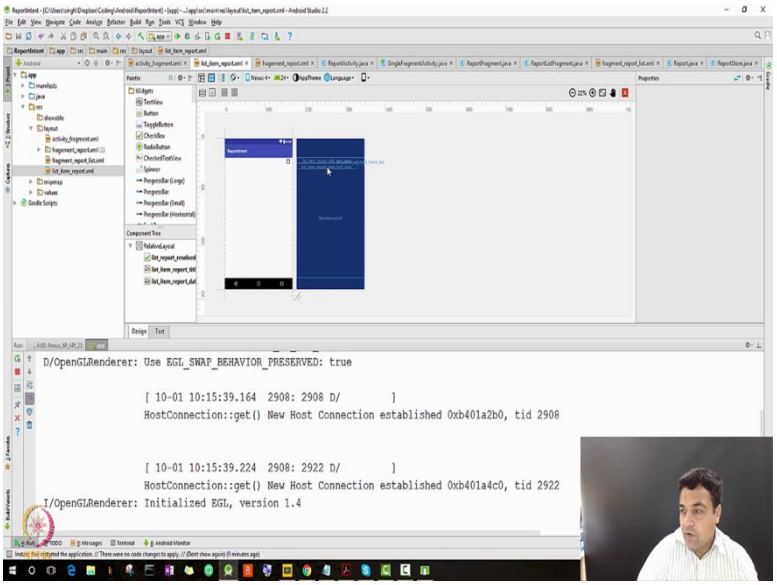
(Refer Slide Time: 32:27)




So first is that, we want it to be the date, so we do a date, now first error is gone, match_parent, wrap_content is fine. Id again we do list item report so the laptop. So this is not this is to the left of the solved checkbox, but it is also going to be below the well let us just type it. It is also going to be below the title so layout below and @id/list TitleTextView,

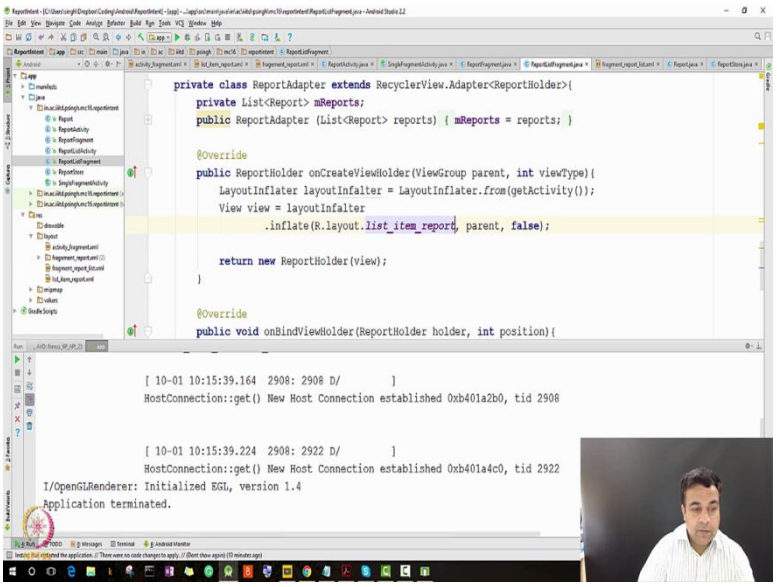
so it is to the left of the checkbox and below the title, the padding is fine we can remove the text style and this is ok.

(Refer Slide Time: 33:35)



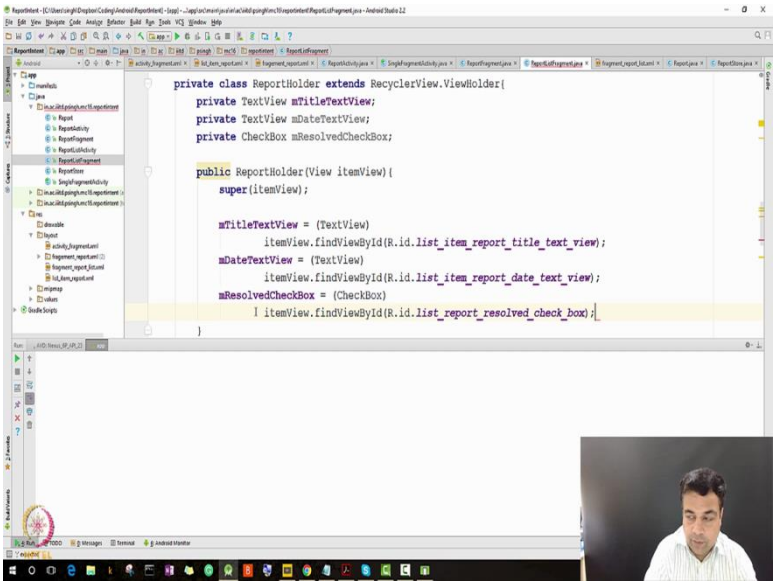

As you can see, here you can see the title here the title will come, here the date will come, here the checkbox will come. So that seems to make our layout complete now need now we need to go back into our code and use this layout instead of the default android layout that we were using. So we were using a simple list layout now we will just change that we will just change that to our layout, so this was a layout that we were using we need not to use it anymore.

(Refer Slide Time: 34:22)



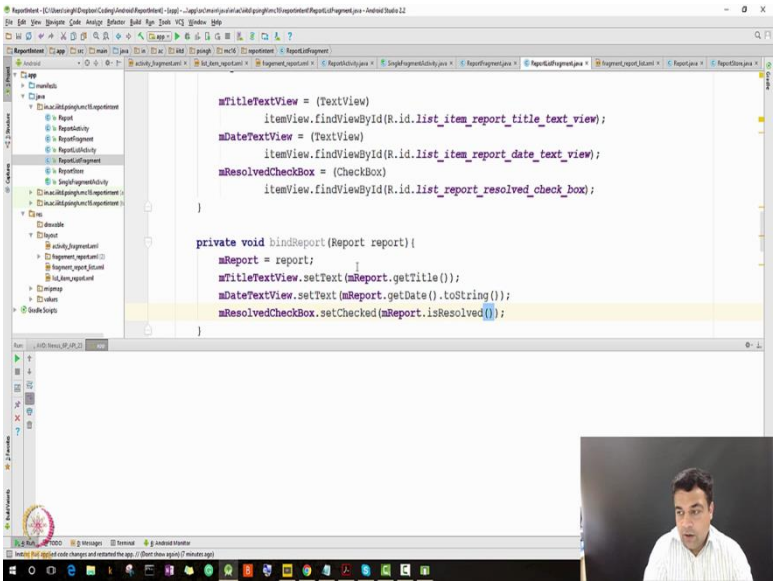
We can just remove it and we can say R.layout.listitemreport, now this is fine. So let us go to the file ReportHolder which was the part of the code list fragment yes, so this was our private ReportHolder. Now we are going to make some changes into it. Number one thing is that because we have changed our layout, the program will not run as it is, so we will have to make some more changes, so let me remove this line and we will be adding few more.

(Refer Slide Time: 35:08)




Private TextView so now we have p textview mTitleTextView we have another private TextView for the date mDate mDateTextView and we have one CheckBox private checkbox mResolvedCheckBox. Public ReportHolder so for itemview and that is fine but we will have to now remove this line and put mTitleTextView = TextView but we will have to now do it in a proper manner so R.id.listtitletextview. And the same thing for and actually it would be good if we put an android here that makes more clear mDateTextView = similar line of code and then it will list of itemView.findViewbyidR.id.listdatetextview and then further CheckBoxmResolvedCheckBox=CheckBox itemView.findViewbyIdR.id.listreportresolvedcheckbox.

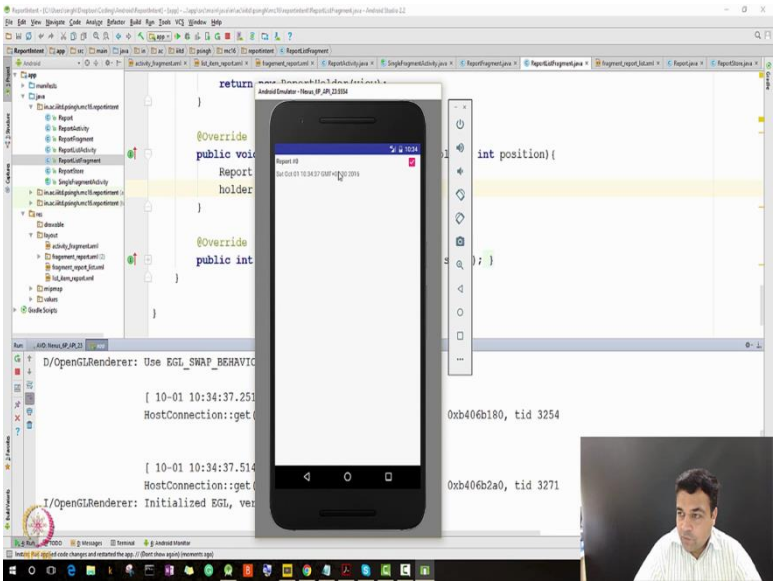
(Refer Slide Time: 38:17)




So now our ReportHolder has what we have recently created. Now, let us make some more changes into our code especially in the ReportHolder class, the same class that we have been changing. And we will just implement few more methods into it, so this is our point I will just make sure that have a private Report mreport. Then I add a method private void bind Report takes a Report report and so it is few things mTitleTextView.setText.mReport.getTitle mDateTextView.setText.mReport.getDate.toString and then mResolvedCheckBox.setCheckedmReport.isResolved.

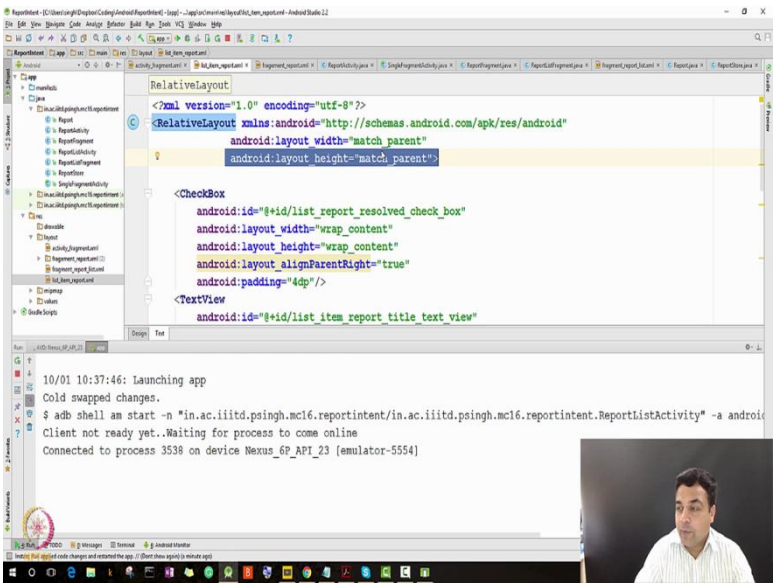
Ok, so we have taken a report and then set our viewItems accordingly, now let us go to our adaptor and try to use this new method that we have just created. So, we will go to our adaptor and in this onBindViewHolder earlier we were simply using our mTitleTextView which is no longer valid because now we are displaying lot more, then we will just simply go to bindReport report that is it. So, we have changed a substantial amount of code from just starting just by displaying title to now just playing lot more than the title. And, now let us run our program to see that how it works.

(Refer Slide Time: 40:42)



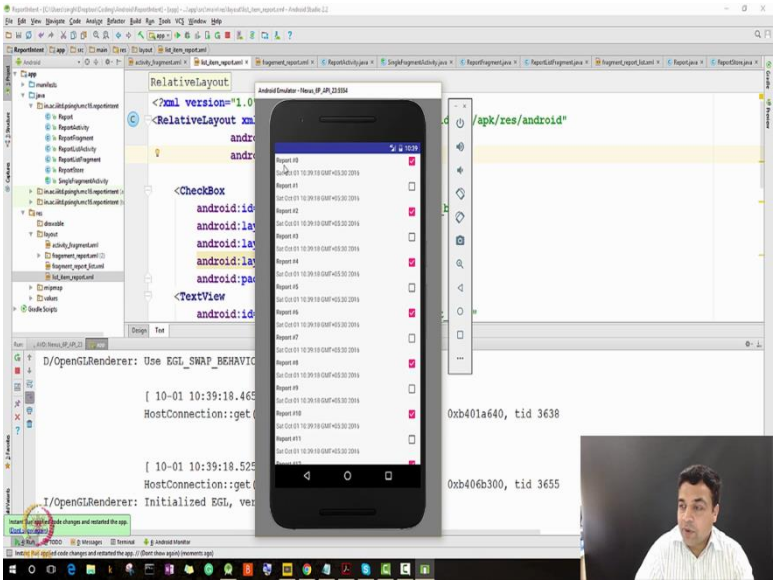
As you can see that now we see more information, but we would like to see where the other reports have gone. So if you run the, if you scroll down then you will see that we have all the other reports, somehow each report is occupying one string. So let us try to see that why it is doing that and why we are not able to see the list that we were earlier seeing. So, let us stop our program, so what is your guess where do you think we may have made an error so that we are now seeing the multiple reports but each report is occupying one string which is not correct. So my guess is that we made an error when we created the new layout. So let us look into our xml file to check whether we indeed made an error. So the layout that we are currently using is the list item report.

(Refer Slide Time: 41:57)



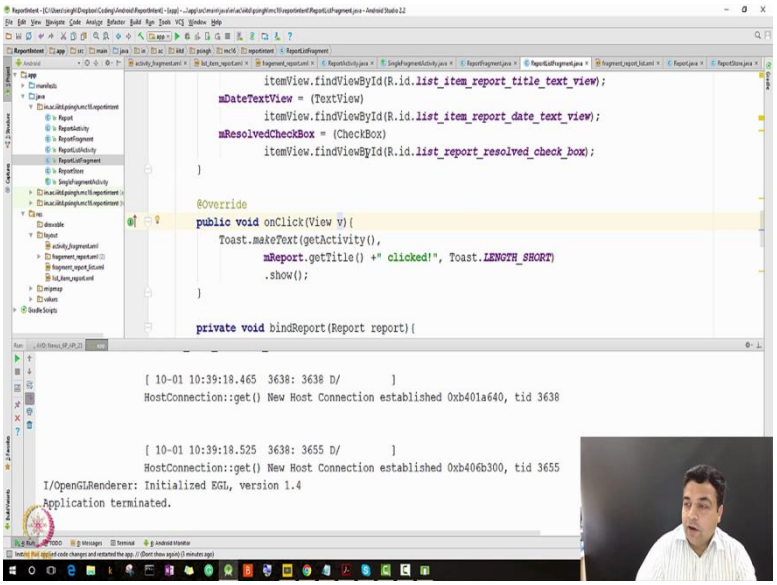
The list item report is here we described it using a RelativeLayout and that here we added the TextView TextView and CheckBox, ok. So, here is our error we are setting the height to match parent and that is why each item is occupying the whole screen. Let us change it to wrap content so currently you are seeing one report occupying whole screen let us change it to wrap content and see what happens, so let us restart our program after changing to wrap content and see what happens.

(Refer Slide Time: 42:30)



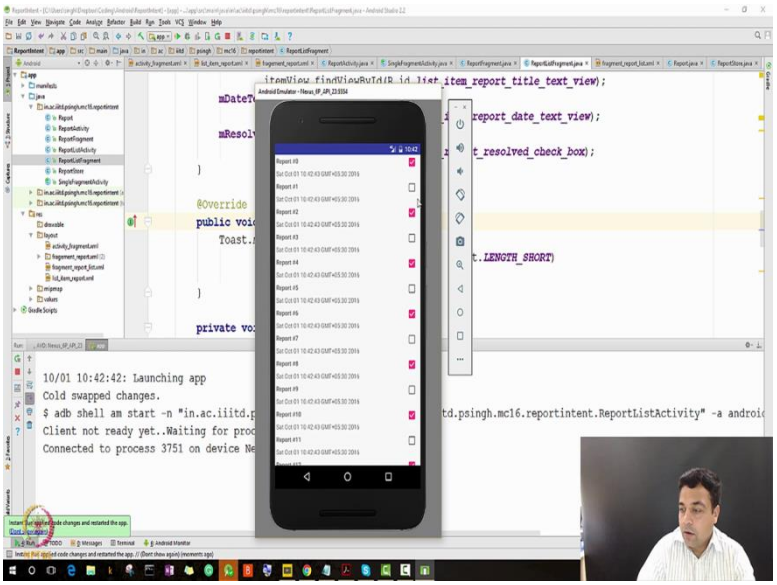

Yes, so now we can see our list again each this is the title, this is the date, this is the CheckBox corresponding to each report. Currently nothing is happening if we are doing the CheckBox, so let us add some more code so that when we do a CheckBox then we can see something happening in our code, so all we have to do is write a listener that is it. Let us stop our program and go back to coding, so let us go back to our same file where we have been making most of our changes that is the ReportHolder ReportHolder because here we would like to set a listener to this.

(Refer Slide Time: 43:26)



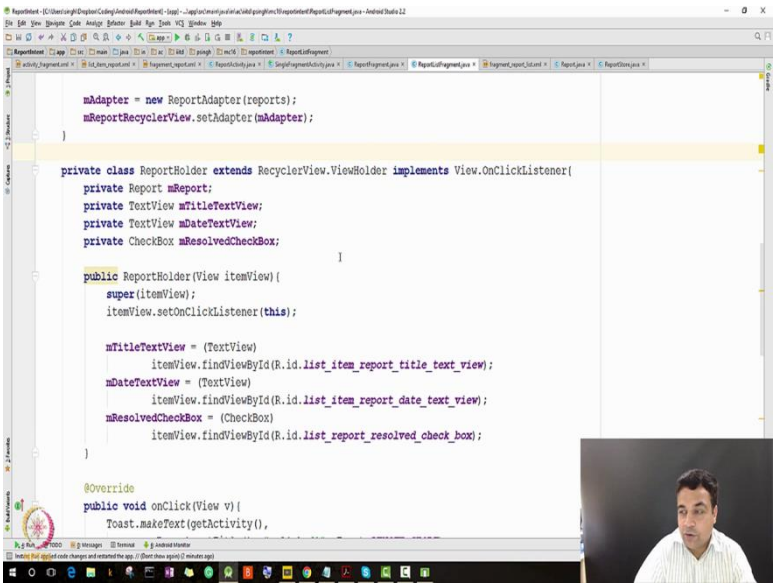

So, we did the ItemView and then we will do as we will just make that itemView.setOnClickListener this, and in order to make it work we will have to implement view.OnClickListener, this is giving me an error because I have not implemented method oh sorry so that method is called the OnClick, So I will soon override that method and everything will be fine. Let us see public void onClick v and we will do nothing but we will just display our toast as we usually do when we want to just do a small activity. So, mReport.getTitle+ let us say clicked and then this is a Toast of, obviously short length Toast of short length. And, then we show, so yes. So, now we have created a listener as well so now let us run our program again and see what happens.

(Refer Slide Time: 45:40)



You see that we are seeing that the report clicked report by clicked our listener was general listener to the reports so it is corresponding to checkbox to the report because we added the listener here to the itemView, so it is referring to the case. So that is all sounds very good, what we did in this what we did in the program so far is that we have been able to add a lot of code. And that code has helped us to show a better view that what we were earlier having. In this class we learned about something new, something called a RecyclerView and we also learned about how these adaptors and ViewHolder in RecyclerView to create a view that is very useful for displaying a list of item.

(Refer Slide Time: 47:54)



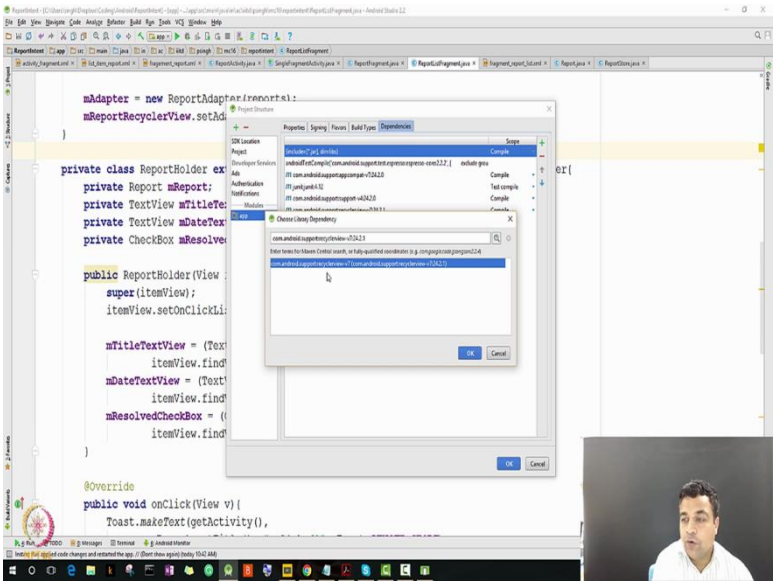

Earlier, we were using LinearLayouts, RelativeLayouts, etc, which are very good where we know that how many items we are going to display, but RecyclerView is very good that we want to display a lot many items and we do not know their number either in advance or they are just too many for example here we had 100 items and we may have had 1000. Now earlier android used to have something called his list view and grid view and these were these are still there these are the list view and grid view are still present. But, earlier till the android 5.0 these were the preferred based create list on grids of items.

And the API of those is also very similar to what is RecyclerView, but now in the never implementation of android especially after android 5.0 RecyclerView is used by default instead of the list view or grid view because the listview and gridview were much more complex and RecyclerView is much simpler. As you saw that RecyclerView beyond the view in ViewHolder and an adaptor and then we could define even a custom layout for each of the

item that we were displaying,sSo this was all very easy to do in the RecyclerView. another thing is that the RecyclerView also provides an animation of items in the list.

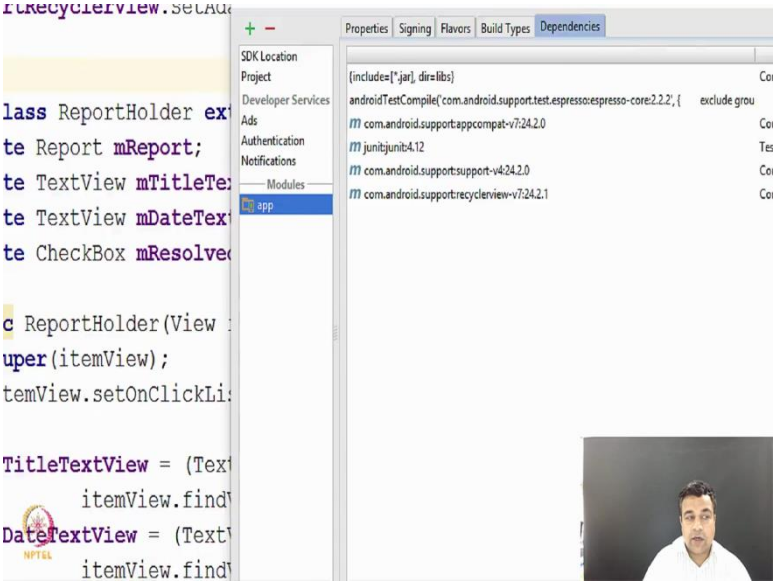
Uhh so you can do that later on in the programs this was something just very difficult in list view and grid view but android but RecyclerView makes it very easy. So, it has some few built in animations which make pattern (())(48:28). So that is all for this line of lectures, so in this series of lectures we have learned about RecyclerView single terms and some other things. We will continue modifying our program to add more functionality to this.

(Refer Slide Time: 49:19)




So one thing which I did not show very clearly was how to include the support library for RecyclerView, we included just the same way we included library for the fragment. So, the procedure is same you click on file go to project structure, go to your app, go to your dependencies, go to the + sign, go to library dependency and here you can type in the search bar Recycler and hopefully it will give you the right, so it will give you the right recycler, the library that will be include and you can just press ok and get included because I have already included it.

(Refer Slide Time: 49:40)




It will show up here, I can show you in the more detail view. So that is what you need in order to compile this project. So that is it we will continue developing this work but in this series of lecture by now you have learned about Recycler, thank you.