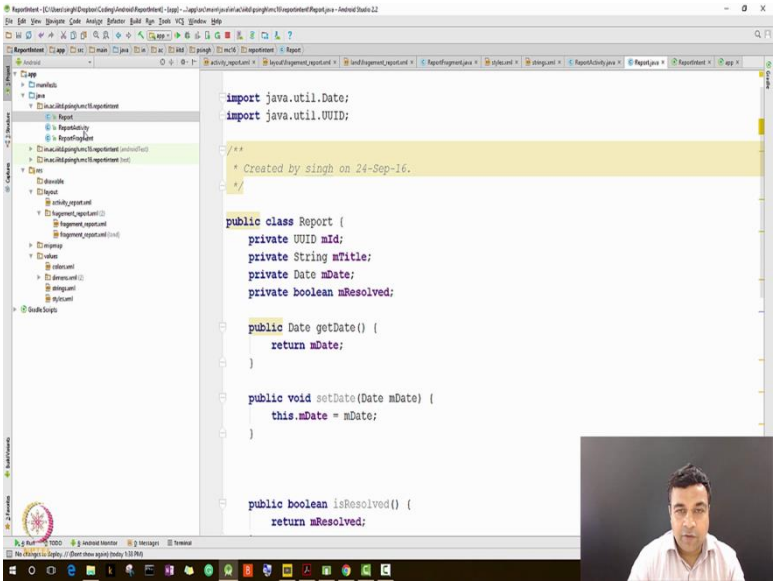
Mobile Computing
Professor Pushpendra Singh
Indraprastha Institute of Information Technology Delhi
Lecture 35

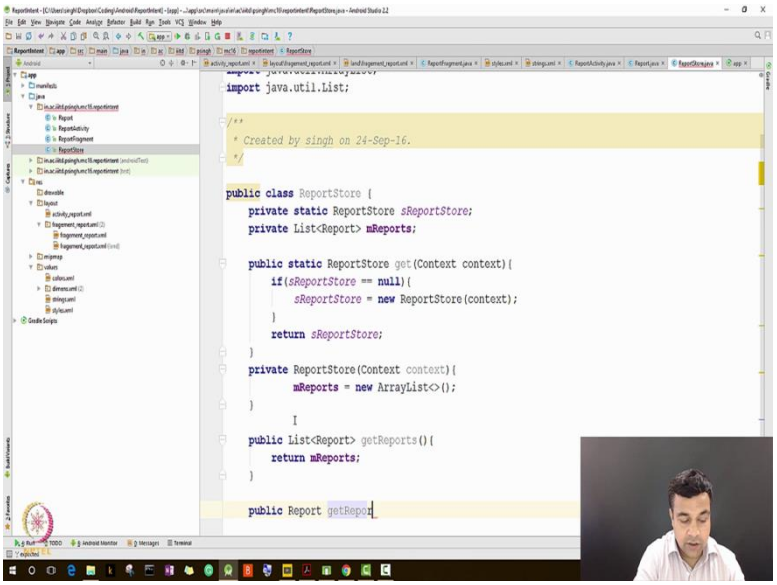
Hello, in this lecture we will continue to extend our program. So far our program can handle only one report, now we will extend it into make sure that it handles more than one report. So let us first have a quick over view of what we have done so far and then let us move ahead. So we have created one activity, one fragment. We have created two layouts for the fragment, in the activity we are calling our fragment using the FragmentManager and then we have some other files which we need to do our task.

(Refer Slide Time: 1:00)



Our ask currently is to show a report list, currently we have only one element but today we will increase it to contain more than one element and then when an item is clicked then show the details, we are yet to do that. So yes let us get started. Number one we are going to do is to add another class here and let us call it report list or we can call it ReportStore. So ReportStore will be a store of multiple reports. Now let us get started and develop and start developing this class. There will be something in this class which I have not done before that is I will be using the single turn pattern and I will explain you why we need a single turn pattern and how to use a single turn pattern. Those of you who have already read about single turn, this could be good revision those of you who have never used single turn please pay attention.

(Refer Slide Time: 2:25)



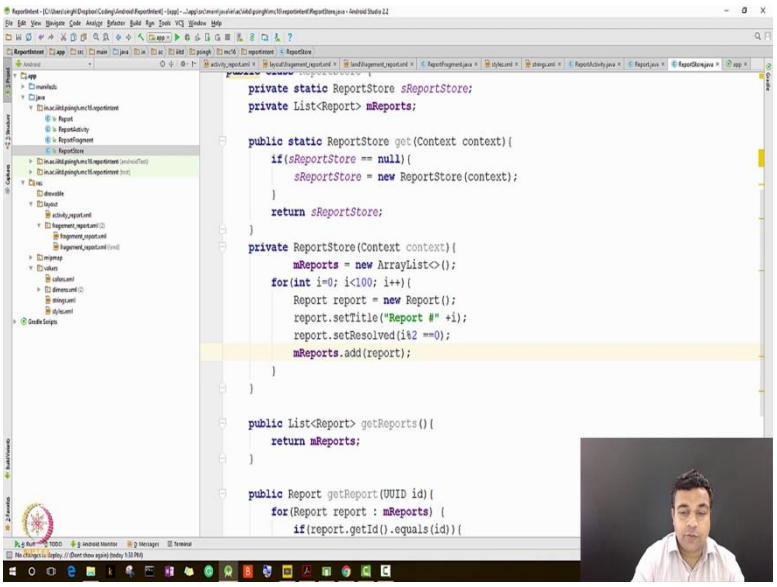
So we start with a static private variable of the type ReportStore only let me call it sReportStore, you will notice that I have put a small ‘s’ that is to use the Android convention to show that this particular variable is a static variable. Now we go we make a public method, this method is also static and we are saying that this method returns a ReportStore, the method name is simply get. So what does this method do it does nothing it just checks that if sReportStore is = null then sReportStore = new ReportStore context and then return sReportStore. One thing that we are missing is a constructor here and please pay attention to what I am doing, so I am creating a constructor. Now my basic class is ready, this class is using a single turn pattern, which basically means that I can create only one object of app report store at any given time.

So in the single turn as we see we can create only one object. Now, if some of you have used eclipse earlier and once you start eclipse you cannot start another eclipse and that is how the single turn pattern is used. Single turn pattern is very useful and is used in many situations, now let us see that how do we use it, so the major difference that you see here at here the constructor is private and because the constructor is private it cannot be called from outside.

Now if it cannot be called from outside then how do we get the reference of the object? So for getting reference of the object we will have to use this method, the get method. Now get method the first thing it checks is that whether there is already an existing object or not. If that if nothing exists then it uses the constructor to create it and then it returns the reference otherwise it just returns the reference of the existing object.

Which basically means that once an object is there you cannot create objects. This is single turn pattern; later on we will see more use of it, so let us move ahead. Now the first thing that we need is something which can store multiple reports. So I will create a list of reports here. So I create private list so list which will store the report and let us give it the name mReports. So we have created a list in which we will store other reports. Now let us start creating some more functions so we move forward, first that in the constructor you say mReports and we will initialize our list report list that is it. Then we would like to have a getter, so public list Report getReports return mReports, public Report now I would like to get a very very specific reports.

(Refer Slide Time: 9:57)

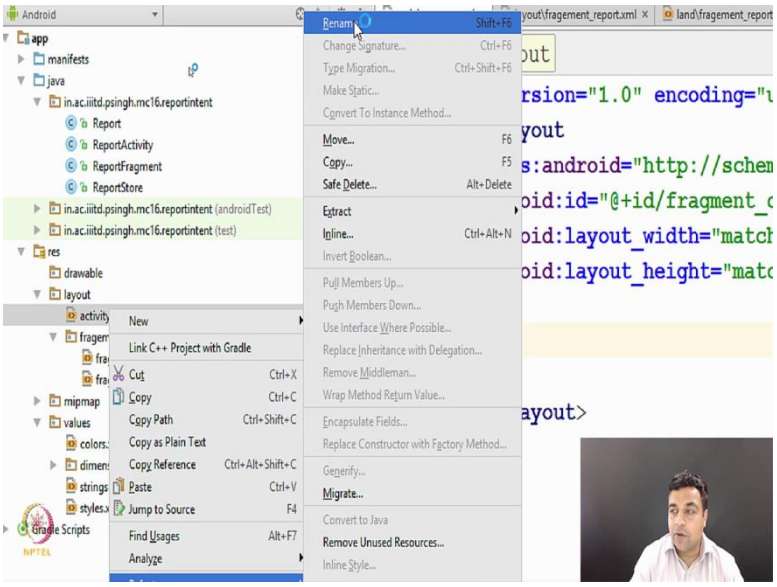


So first I am getting the list in the previous method now I am getting a very specific report, report mReports if report.getId.equals the id that we are supplying then return the report otherwise return null. So let us look at the functions I have created a simple getter which will later may the complete list then I have created a get where I get a very specific report and we are using this Id of the specific report to retrieve it from the ReportStore. Now for the time being let us generate our reports so that we can see the functionality of our program. So I go back again into the constructor and I will try to write a for loop which will generate five such reports for us i is less than 100 and i++ Report report = new Report() report.setTitle() Report number +i report.setResolved so we would like to set this variable alternatively for each report.

So I will just use some piece of code which will return me the right value. As you can see that this piece of code depending on the i value will send half of the reports as resolved and half

of the reports as unresolved. Then I say mReports.addreport so that is it, this is the code that will. Now add some reports to our program 100 reports to be precise ok. So now our basic work in ReportStore is complete. Let us come back to our earlier created layout file activity_report.xml as you see that this file there is nothing specific, but it is a very very generic in fragment container file. So there is nothing specific with this report infact you can use this code for any fragment.

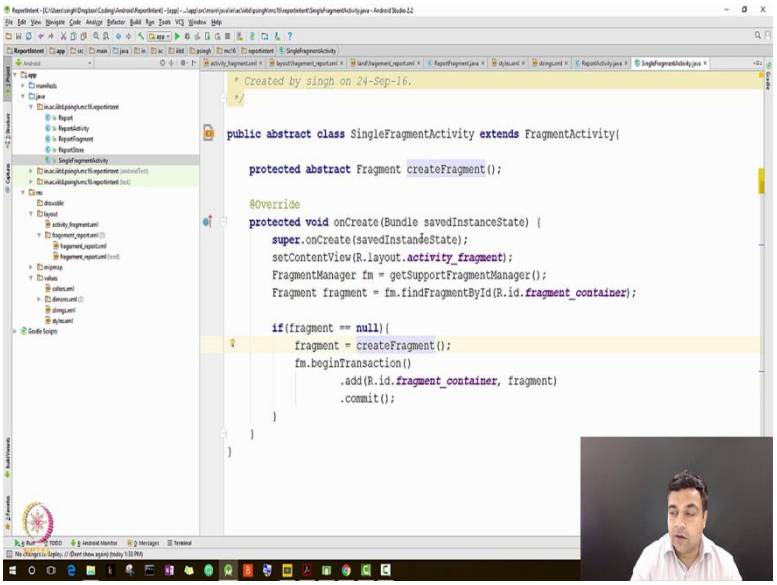
(Refer Slide Time: 12:49)



So let us do one thing, let us first because it is a very generic then let us first change its name and call it let us say activity_fragment because we were want to use it for many other things instead of just for the report here. So how do we change the name of this file, the right way of changing the name is to right click and go to an option called Refactor, in the Refactor you will get an option called Rename and that is what we have to choose. Let me start the magnifier, so I right click, I go to Refactor and from the Refactor I go to Rename. In the Rename I change the activity_report to activity_fragment, so that the name reflects the generic nature of this file I press Refactor and the name is changed.

The advantage that you see is for example if I go to the report or report but ReportActivity the name is automatically changed there. If I had simply done a Rename here without going Refactory, I would have to fix it then. And in bigger program this is one of the problems so always use the Refactor option for Rename, etc, so yes that is fine. Now similarly, just because we made a very generic layout we will also be making an abstract activity class. Now our activity_fragment.xml is a generic layout class for hosting any fragment in may another situation. So we have created it so that we can use it in many other programs.

(Refer Slide Time: 14:24)



Let us go back to our ReportActivity, if you look at the code of the ReportActivity then you will realize that except this part rest of the code of the ReportActivity is good enough to host any given fragment, which means that even the ReportActivity class is very generic in nature and we can use it any other program wherever one activity needs to host one fragment. So let us create an abstract class which has this generic code and then from that abstract class we can have classes which are very specific so I am going to add another class here which I will call as SingleFragmentActivity and in this class I will actually use all the code that I have been using in the code activity.

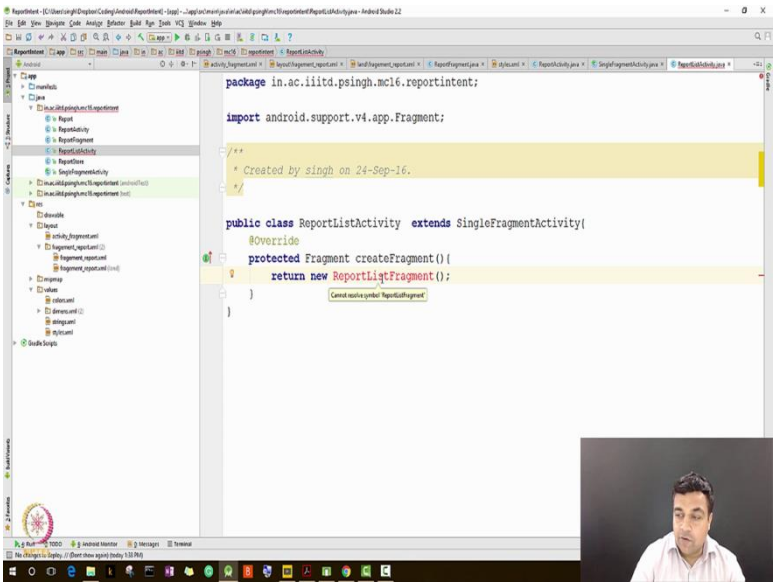
I will call it extends FragmentActivity the only thing that we need to is we will have to take care of this report fragment and we will have to also add an abstract viewer here. To take care of this report fragment we will create a method called createFragment will all the subclasses will inherit. So in order to take care of this report fragment I am creating a abstract method createFragment, and then I will remove this line with the following. So this is fine, now you see that our new class SingleFragmentActivity class is a very generic class and this class can be used wherever we want one activity to host one fragment.

So we have created a very generic layout class and we have created a very generic java class for hosting a single fragment. Now in order to make use of this we will have to go back in our ReportActivity and what we will do is that we will first thing that instead of the FragmentActivity we will call it SingleFragmentActivity and then what we need to do is that we will remove all that and only thing which we will have to do is the override our

createFragment method protected Fragment createFragment method and in this case our createFragment method will just call ReportFragment and that is it.

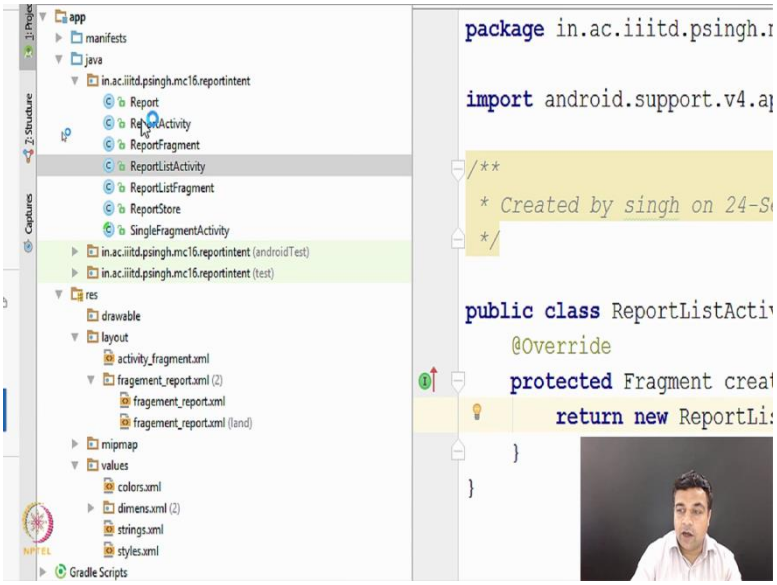
Now we can also remove this, so now I hope you are understanding what I am trying to do I have created a very generic class where there is one activity which host one fragment and then for my specific purpose I am just using that generic classes as super class and for my very specific fragment I am generating it here. So now the advantage is that tomorrow if I am writing some other program I can use SingleFragmentActivity class directly. There I will not have to write the code for it.

(Refer Slide Time: 20:01)



Now let us add some controllers in our program, so for that we will add other two classes, one class we will call as ReportListActivity report because now we are working with a list of reports and not just one report. So apart from ReportActivity we will have to have a class called report list activity I will just extend this class with my SingleFragmentActivity class, it gives me a warning that it must implement the method, I am going to implement it @override protected fragment createFragment and so far I am just saying new ReportListFragment. We do not yet have a ReportListFragment but we are very soon going to create it.

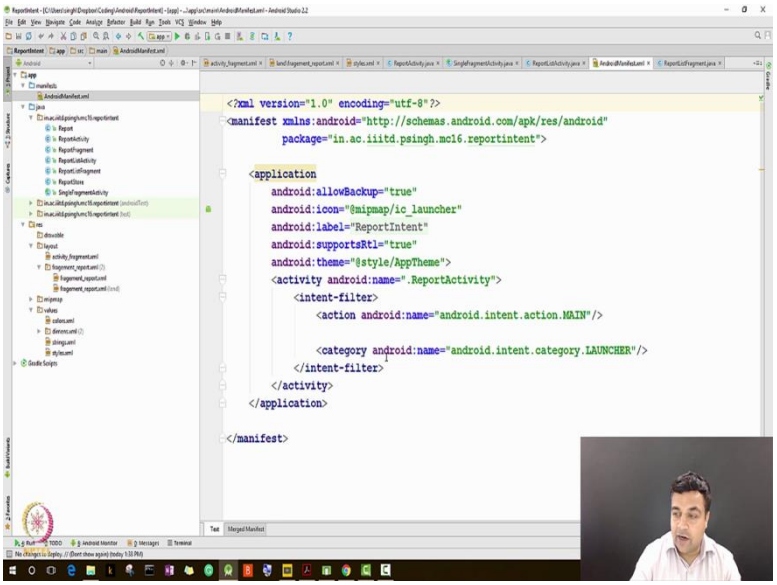
(Refer Slide Time: 21:50)



Let us start working on it, another java class called ReportListFragment press ok. This will actually be a fragment, so I will just extend Fragment and that is it I will not do anything, I save a ReportListActivity error is now gone and that is fine. Now let us first because we have done a lot of things here let us first have a very quick look on how the program looks like. So we start with this currently now I have so many files, as you will see that my SingleFragmentActivity is an abstract class which only holds a generic code and then I have my ReportActivity for individual reports. I have my ReportListActivity for list of reports, I have my Fragment for a single fragment and I have my ListFragment, then I have a ReportStore where different reports are stored and then a single report, class where I have some generic information about each report such as its Title, its Date, etc, etc.

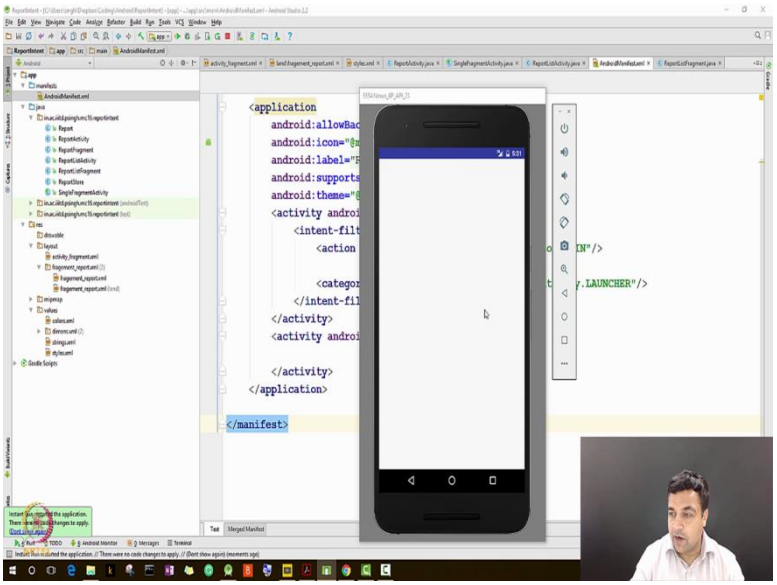
So our program is now getting bigger and more complex and hopefully we are also adding enough functionality to make it a really interesting program, so far in this program we have learned about Activity and Fragments which we have discussed very soon you will be learning about many other concepts in practice, we have already learned them in the lectures now we will see them in practice.

(Refer Slide Time: 23:24)



So let us go back to ReportListActivity first thing that I want now to do is that I want to make this class as the main class to start the program. So in order to do that I will go into the manifest file and as you can see that my manifest file is showing me currently the activity the original activity call ReportActivity and it has an intent filter of type launcher, I would like to have another activity here another activity which I am calling as ReportListActivity and I would like to make that ReportListActivity as the activity which starts the code. So I copy this part, I paste it, I change it to ReportListActivity, I then keep the same intent filter of main and category launcher, but I remove it from here. I have made some changes in the manifest, which basically means that my ReportListActivity is now the LauncherActivity and my ReportActivity is not, so now let us try to run our program.

(Refer Slide Time: 24:57)



So when we run our program you see so now we have again got an empty, so earlier we were seeing a one item with respect to the report and now because we have changed it to ReportListActivity and let us see that what ReportListActivity is doing that is only showing us the fragment container, and the fragment container has nothing that is why we are not seeing anything here.

Now this is all good, but we actually wanted to see a list of reports, so we will have to do some more work. So essentially for showing this list of report we will be using something what we call as RecyclerView. The RecyclerView is the subclass of the view group and it displays a list of child view objects. Now one for each item in your list of items in our case these items are reports and depending on the complexity of what we need to display, these child views can be complex or they can be very simple, but anyway the RecyclerView is the right choice for displaying a list of child view objects.

The reason why we use the RecyclerView is that in case of let us say 100 reports like in our example we would like to display a list of 100 reports. Now if we go to each report you can see that on the screen that each report has a TextView, a button, a TextBox and many other things. Now if I want to display 100 such reports, do I need to create 100 such view objects, that is too much of the work and plus that will consume resources.

And this what if the report has 1000 elements, so on the other hand on a mobile screen or on a tablet screen or any smaller device screen only some of these will be displayed at any given time. So why do not we only create the ones which are needed at any point of time and then

when the user scrolls down or scrolls up, we reuse the elements. And this is something which is provided by RecyclerView as we will soon see in the program however, for today we have learned enough and we stop right now. In the next class we will start with the RecyclerView and adaptors to learn more and to see how to extend our program, thank you.