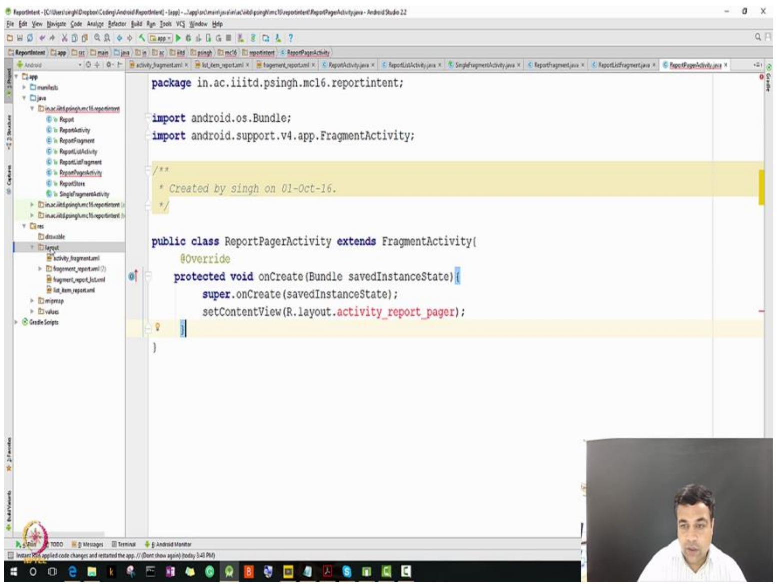
Mobile Computing
Professor Pushpendra Singh
Indraprastha Institute of Information Technology Delhi
Lecture 39

Hello, so In this lecture we will use some things what is known as a view pager. So, that we can swipe from one page to another page to watch the details of different reports that we are creating. So, a view pager will provide us much better interface and how much better user experience you may have already seen it is use some of the application that you use where you can just swipe left or right to move from one page to the another of an application and today we will learn how to developed that functionality into your own application.

So, in this lecture we will add a new activity you will call it a pager activity and then we will define a view therapy that consists of view pager and then we will wire up this new pager and its adapter in our new activity that we describe and then we will modify our onclick method in the report holder class, so that we can start this pager activity instead of our regular report activity that we are currently starting. So, let us start with the developing our new activity that we will call it report pager activity.

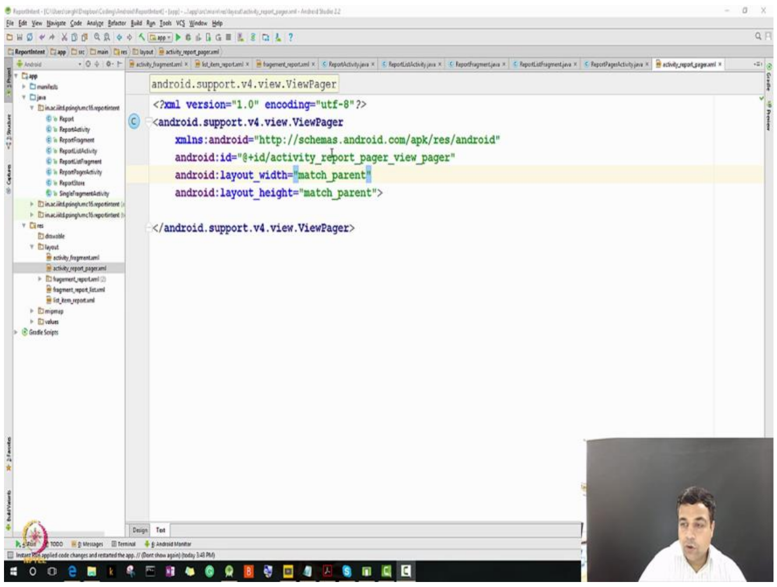
(Refer Slide Time: 2:04)



```java
package in.ac.iiitd.psingh.mc16.reportintent;

import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

/**
 * Created by singh on 01-Oct-16.
 */

public class ReportPagerActivity extends FragmentActivity{
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report_pager);
    }
}
```

So, this will be a sub class of the fragment activity. So, let us get a started, so I am going to new java class, I will call it report report pager activity, I add it, I now extends fragment activity as you can see that our library is down folded move on and I just override the oncreate method, protected void oncreate bundle saved instance state and in this we just call the super.oncreate saved instance state, then we set the view to a layout that we have not it created but we will soon be creating. So, I will call this layout as activity_report_pager. So, we are getting an error this is there because we have not created this layout. So, in the next step let us create this layout of the typed report pager that we wants to do, so let us a start working with it, it start creating this layout okay.

(Refer Slide Time: 4:51)

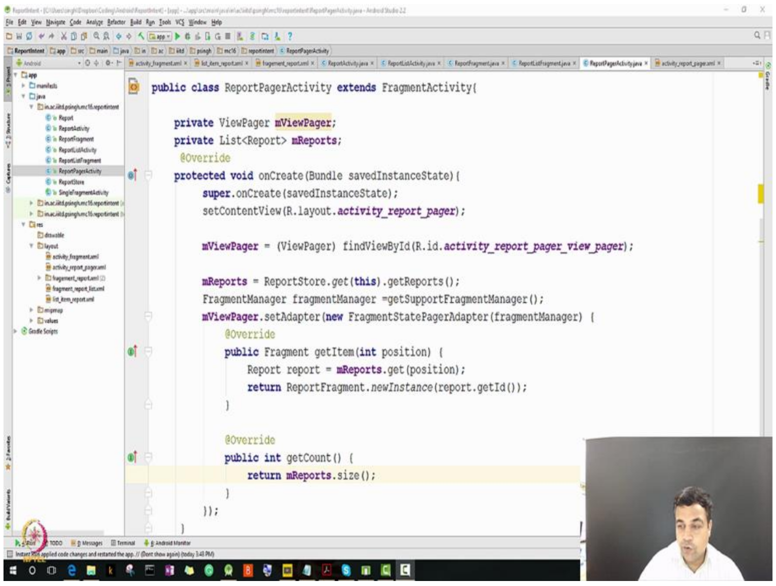


```
android.support.v4.view.ViewPager
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_report_pager_view_pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</android.support.v4.view.ViewPager>
```

In the layout file I go and I get a new layout resource file and the I am just giving it the name the name which we just give that as the activity report pager, activity_report_pager, when I look in this I hope to find something useful and this is my root element which is currently written as in a layout but I will call it view pager. So, as you can see that I can access this, so view pager and then I press okay. So, if you go to the tags we have got this code automatically added and we will now be making some changes to this code. The one change that we definitely want is to add an id because we will be making the use of it. So, as usual B2 @ +id/let us call it activity_report_pager_view_pager.

So, you may have already noticed that I always put the suffices of the type creating and that is usually a very good practice and it is also the (())(5:27) so, we created it rest we will leave to match_parent, match_parent and that is how we look like, which is fine for our purpose. So, this looks all right now we will go on and we will write some other code. So, the view pager is very similar to the recycle view that we are using that is that a view pager will work in the same manner that are recycle of view works with its adapter, the view pager has a pager adaptor and the recycle you had an adaptor however, the interaction between the view pager and the pager adaptor which there it is a uses is much more involved. So, we will now we creating that part of the code in our program so that we can make use of the repager

(Refer Slide Time: 6:39)



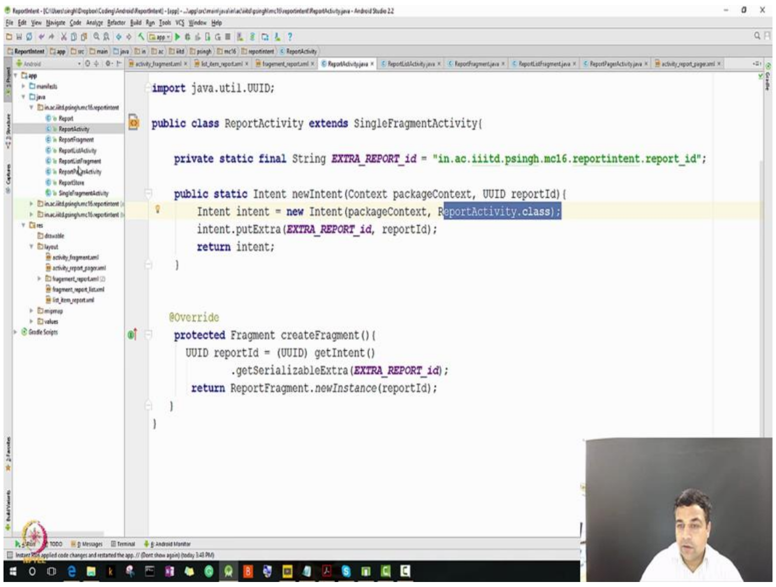
```java
public class ReportPagerActivity extends FragmentActivity{

    private ViewPager mViewPager;
    private List<Report> mReports;
     @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report_pager);

        mViewPager = (ViewPager) findViewById(R.id.activity_report_pager_view_pager);

        mReports = ReportStore.get(this).getReports();
        FragmentManager fragmentManager =getSupportFragmentManager();
        mViewPager.setAdapter(new FragmentStatePagerAdapter(fragmentManager) {
            @Override
            public Fragment getItem(int position) {
                Report report = mReports.get(position);
                return ReportFragment.newInstance(report.getId());
            }

            @Override
            public int getCount() {
                return mReports.size();
            }
        });
    }
```

So let us get a started into our report pager activity that will earlier create it. So, earlier we had only used the oncreate now if we as going to augment it with more code, so that we can do more point. The first thing is that I will create a view pager, m view pager Alt Enter, I will do in code class say yes and then the list we list items that you want to see, so we need to have a list of report we list reports report m reports yes Alt Enter, so this is also done in everything right and then we are going to augment the oncreate method further to get our desired functionality. So, I start with m view pager which will create. So, view pager then we find view by ID, R.id.activity report pager pager that is just the layout that we created than I am reports = report store.get this.get reports.

Now something which we have already done use of a fragment manager, I just say call it a fragment manager and then we do get support fragment manager and the pager.set adaptor new fragment state pager adaptor that is the adaptor that I was talking to you about, when we discussed the similarity between the view pager and recycle the view. Now it is overriding 2 methods one is the get item method and one is the get count method. So, in the position we need to write some more code. So, report report = m reports.get because we need to give the write report whose detail that you want to see and that we want we return the null, but we will returned the report fragment.new instance.

So, if you remember we have created this new instance method to (read) to create our fragment. So, we will no longer working with the constructive of the fragment, but the new instance method and here we just say the report.get ID and in the get count also we need not return 0, we can return m report.size that is it quickly look into what we have done here. So, we what we have done is that we have created a view pager and then we have given it the list of report that we want to work with and then we set the then we used the fragment manager and set the adaptor to the correct fragment, so we get that from this p support and here so essentially what we are showing is that we are creating fragments and we are making sure that we keep a track of the fragments that we have created two this play the right details by using some of these variables.

(Refer Slide Time: 11:49)



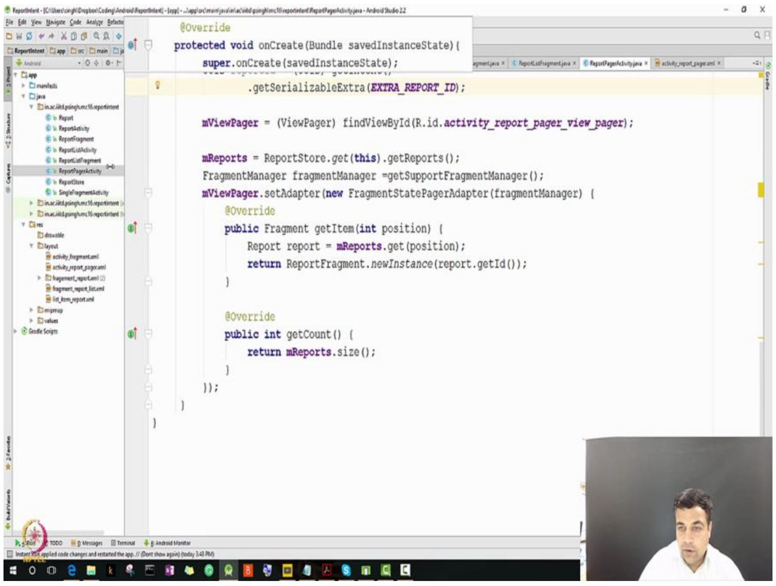
```
import java.util.UUID;

public class ReportActivity extends SingleFragmentActivity{

    private static final String EXTRA_REPORT_id = "in.ac.iiitd.psingh.mc16.reportintent.report_id";

    public static Intent newIntent(Context packageContext, UUID reportId){
        Intent intent = new Intent(packageContext, ReportActivity.class);
        intent.putExtra(EXTRA_REPORT_id, reportId);
        return intent;
    }


   @Override
    protected Fragment createFragment(){
     UUID reportId = (UUID) getIntent()
             .getSerializableExtra(EXTRA_REPORT_id);
      return ReportFragment.newInstance(reportId);
    }
}
```

So, now the task that is left for assist to integrate the activity and this code, so now let us go back to our report pager activity and add some more code here first is again because we need to pass some information and we are so the way we are preparing the report with the activity is to that it will do the work what earlier report activity was. So, if you remember the report activity was passing a report ID and then you creating intent and then we was all of creating a fragment related to back. Now we want to do similar activity here because will replace report activity with report pager activity.

So, we are starting with string extra_report_ID = in this create the same name t.psingh. where is my M16.report intent.report_id and this is fine and the second thing we I need to do is to create the new intent method similar to the one that we had created in the report activity class, so public static intent new intent on text that which context we could have actually simply copied that method and get some changes but let us write it again. So, intent intent = new intent and then we will do practice from text and repeat. So, instead of the report a activity now will do report pager activity.class and then we will do some extra, just like we were doing last time. So, extra report ID and the report ID, then we will do the returned intent.

So, you see that this method is very similar to the method which we had earlier created in the report activity the only difference is here rest everything you set because you want to eventually replace report activity with the report pager activity.

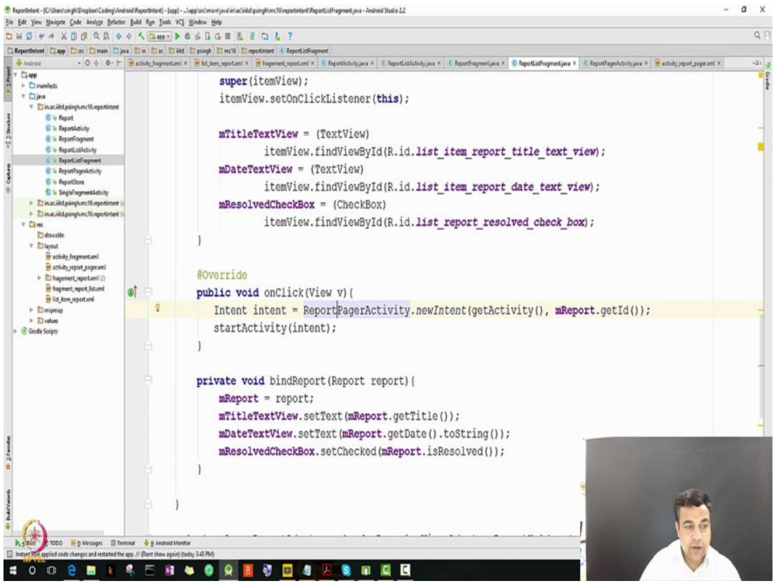
(Refer Slide Time: 15:18)



```
@Override
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
            .getSerializableExtra(EXTRA_REPORT_ID);

    mViewPager = (ViewPager) findViewById(R.id.activity_report_pager_view_pager);

    mReports = ReportStore.get(this).getReports();
    FragmentManager fragmentManager =getSupportFragmentManager();
    mViewPager.setAdapter(new FragmentStatePagerAdapter(fragmentManager) {
        @Override
        public Fragment getItem(int position) {
            Report report = mReports.get(position);
            return ReportFragment.newInstance(report.getId());
        }

        @Override
        public int getCount() {
            return mReports.size();
        }
    });
  }
}
```

Now just some another change in the oncreate why will give a doing the, so if you go here then we had something like this. So, we will have to also enable this thing here. So, we will go to the report pager activity and had some code may be here will UUID report ID = UUID get intent.gets you realizable extra report ID. So, yes and this looks fine for the time being let us have you reliably adding some more code to make this program work at view pager.

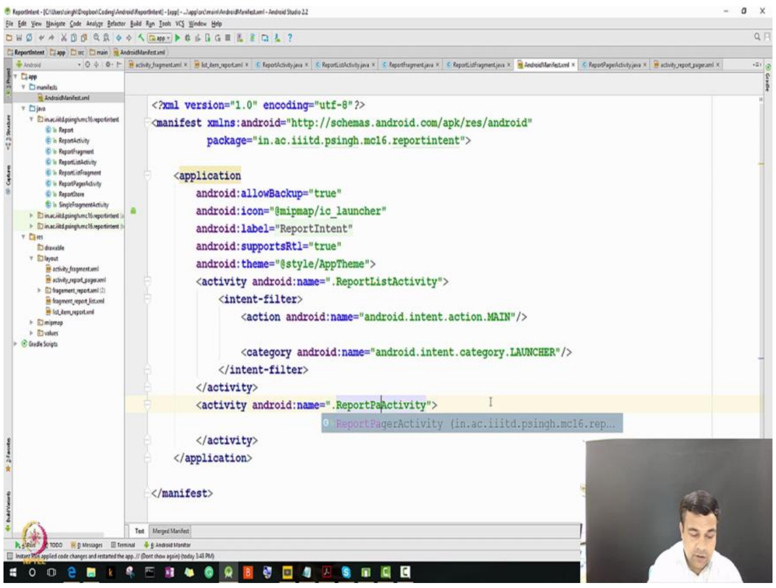
(Refer Slide Time: 16:42)



```
        super(itemView);
        itemView.setOnClickListener(this);

        mTitleTextView = (TextView)
                itemView.findViewById(R.id.list_item_report_title_text_view);
        mDateTextView = (TextView)
                itemView.findViewById(R.id.list_item_report_date_text_view);
        mResolvedCheckBox = (CheckBox)
                itemView.findViewById(R.id.list_report_resolved_check_box);
    }

    @Override
    public void onClick(View v){
      Intent intent = ReportPagerActivity.newIntent(getActivity(), mReport.getId());
      startActivity(intent);
    }

    private void bindReport(Report report){
        mReport = report;
        mTitleTextView.setText(mReport.getTitle());
        mDateTextView.setText(mReport.getDate().toString());
        mResolvedCheckBox.setChecked(mReport.isResolved());
    }

}
```

Now let us go to the report list fragment class which will now start our instead of starting the report activity should now start report pager activity. So, yes so let us see where we were starting report activity. So, in the onclick we will start report activity now we would be doing the report pager activity. So, let me just add pager I hope everything is fine.

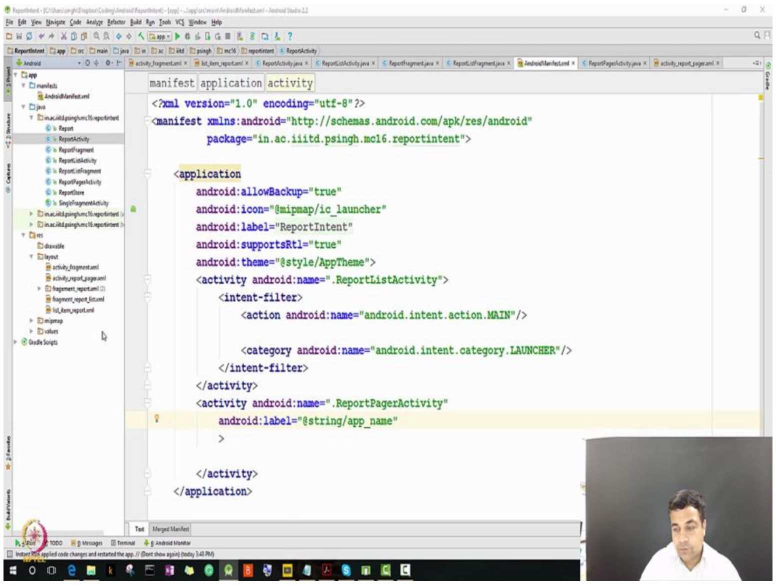
(Refer Slide Time: 17:10)



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="in.ac.iiitd.psingh.mc16.reportintent">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ReportIntent"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".ReportListActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:name=".ReportPaActivity">
                          ReportPagerActivity (in.ac.iiitd.psingh.mc16.rep...
        </activity>
    </application>

</manifest>
```

So, yes everything is fine and now we also need to add the report pager activity to the manifest file so that the OS can start it and we are in the manifest file we will be having ,so we had this activity called report activity now I will remove report activity and that code pager activity but how what will do is I will just add this. So, now report activity is gone and report pager activity has come and what I am saying is essentially that now there is no activity in the ‘I’ program called report activity because we you do not need it and let me just give it a label label = @string so this is all fine. I save my manifest file, I have made some changes in the code and now I do not need report activity anymore so, I do not need report activity I will most likely delete it.

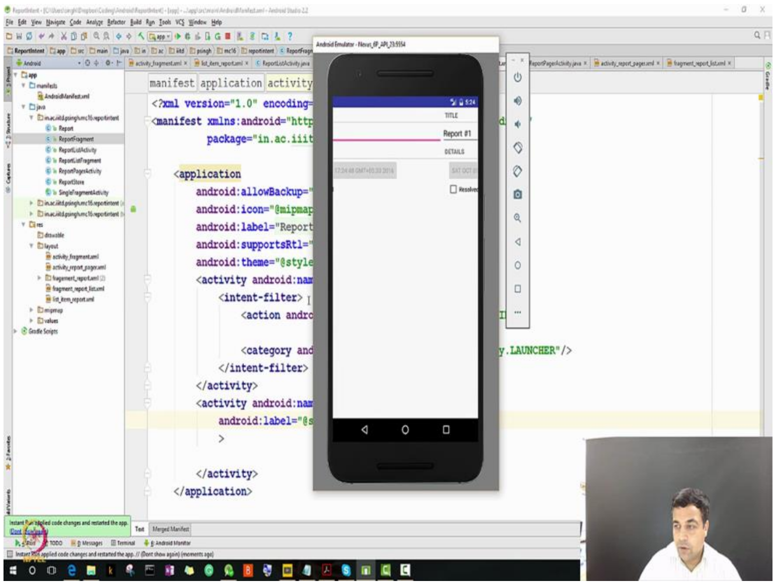
(Refer Slide Time: 18:03)



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="in.ac.iiitd.psingh.mc16.reportintent">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ReportIntent"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".ReportListActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:name=".ReportPagerActivity"
            android:label="@string/app_name"
            >

        </activity>
    </application>
```

So, I right click and I delete and I press okay. So, now let us run our program and see what changes that happened let us run our program let us click on report 2.

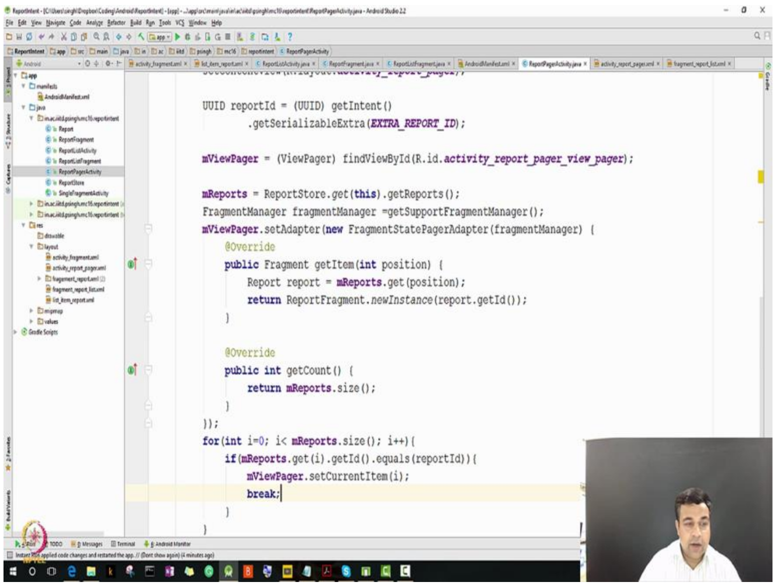
(Refer Slide Time: 18:33)



So, now you see that I can swipe report 3, report 4, report 5, I can swap left and obviously this is a much better UI and gives a much better user experience. So, in this very small is span of time you just saw how to use view pager and how to integrate it and you can also see that by using fragments, fragment argument recycle view and view pager how you can make applications which have multiple interfaces and how easy it is to make such applications.

Now have we done everything here may be let us try it again. So, I am currently here I go back and let say I want to see the report 6 how but it always the start with report 0 is that, so that it try one more time I click on report 10 oh no, it is starting with report 0 again. So, we need to do some more fixing of the code because by default we see that it is always starting with report 0 and this is happening because the view pager always source the first item in the pager adaptor and because the first item is report 0 that is why it is shown here.

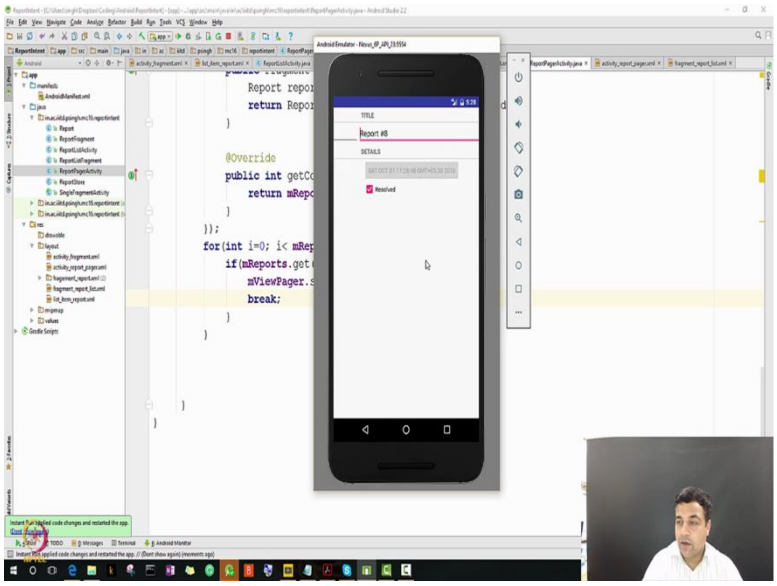
(Refer Slide Time: 20:23)



```java
UUID reportId = (UUID) getIntent()
        .getSerializableExtra(EXTRA_REPORT_ID);

mViewPager = (ViewPager) findViewById(R.id.activity_report_pager_view_pager);

mReports = ReportStore.get(this).getReports();
FragmentManager fragmentManager =getSupportFragmentManager();
mViewPager.setAdapter(new FragmentStatePagerAdapter(fragmentManager) {
    @Override
    public Fragment getItem(int position) {
        Report report = mReports.get(position);
        return ReportFragment.newInstance(report.getId());
    }

    @Override
    public int getCount() {
        return mReports.size();
    }
});
for(int i=0; i< mReports.size(); i++){
    if(mReports.get(i).getId().equals(reportId)){
        mViewPager.setCurrentItem(i);
        break;
    }
}
```

So, now let us add some more code and we will be adding this code to the report pager activity and towards the very end in the oncreate method let us add some code. so, this is for report pager activity, so just after everything I am saying after setting the adaptor let me add a simple do, I<m reports.size I this ++ if m reports.get i.get ID.equals and it should be = the report ID, then set my view pager current item to I and just read out of the loop. So, now let save it and start running the program again to see what happens.

(Refer Slide Time: 21:59)



```
        Report repor
        return Repor
    }

    @Override
    public int getCo
        return mRepo
    }
});
for(int i=0; i< mRep
    if(mReports.get(
        mViewPager.s
        break;
    }
}
```

So, now we run so now if I click on the report 7, the report 7 comes in and I can go to 8 and I can go to 6, so that small bug is now removed. Now if you see that here we were using something called fragment state pager adaptor, there is another pager adaptor that we would have used which is just called fragment pager adaptor, both works pretty much same accept that there is one crucial difference that when we are using fragment state pager adaptor and we are not using the fragment then that fragment is destroyed.

So, the idea of the fragment state pager adaptor is that the as soon as the fragment is not used it is destroyed, but if we you are using fragment pager adaptor, then the fragment pager adaptor dose not destroy the fragment, but it just removes the fragment and it destroyed only the fragment view, so the fragment instance remains alive but its view is destroyed. So, as you can imagine that fragment is pager adaptor is more frugal with the memory, it manages the memory very well and if each of your fragment is heavy for example, if each you fragment includes photograph and then it may be a better choice.

So, whether you use the fragment state pager adaptor or whether you use the fragment pager adaptor, it depends on the application that you are using and the type of fragments that you are creating obviously one is good with the memory while the other one will possibly keep you less delay, so that is your choice. So, in this video we learned about using view pager and we learned about how to use in pager to scroll to create a UI, where you can restore instead of scroll you can do swap left and right and you can see new fragments viewed in your applications, so that us for it Thank you.