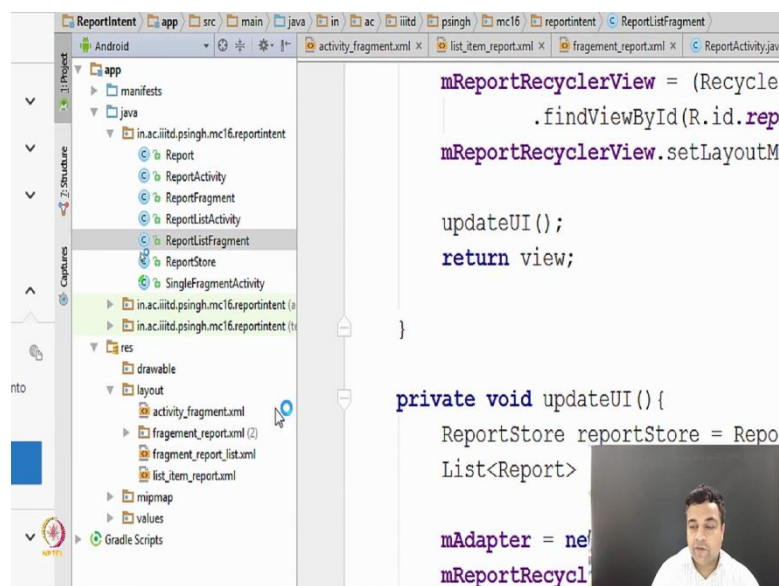


Mobile Computing
Professor Pushendra Singh
Indraprastha Institute of Information Technology Delhi
Lecture 37

Hello, so we will continue with our program and in this series of lecture we will try to come to our original goal of getting a multiple user interface. That is when we click on one item in the list another fragment will start and shows us the detail. So, in this lecture you will learn about how to start an activity from fragment and how to actually implement the multi time display.

(Refer Slide Time: 1:01)

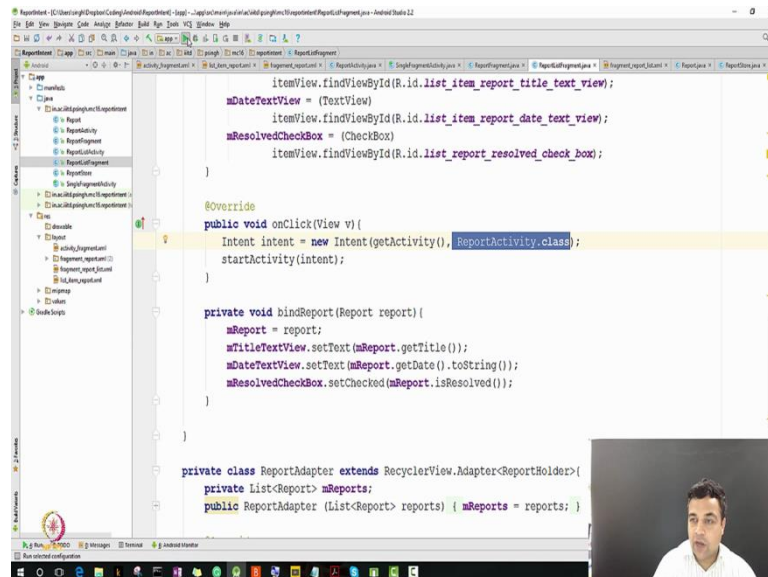


So, let us start programming, I will just do a quick revision we now have following files in our program Report, ReportActivity, ReportFragment, ReportListActivity, ReportListFragment, ReportStore and SingleFragmentActivity. So, these are the files that we currently have and we are now going to start working. So fine so the first thing that we will be doing is that when I user presses an item on the list of reports a new ReportActivity which will host a new fragment will appear and display the details for a particular reference of the report.

So, let us see that how do we start an activity from a fragment. So, starting an activity from a fragment is not very different that we did earlier in the very beginning in the math quiz app, where we started an activity from another activity. Similarly, we can start an activity from a fragment only thing you have to do is that you have to call `Fragment.startActivity` which takes intent as a parameter and then this will start the corresponding activity. So let us start coding, so we will go to report list fragment class, this is our report list fragment class and in

this we will let say update that was earlier doing nothing but just showing a toast the onClick method, but this time now it will send an intent to start an activity.

(Refer Slide Time: 2:15)



```
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 = new Intent(getActivity(), ReportActivity.class);
    startActivity(intent);
}

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

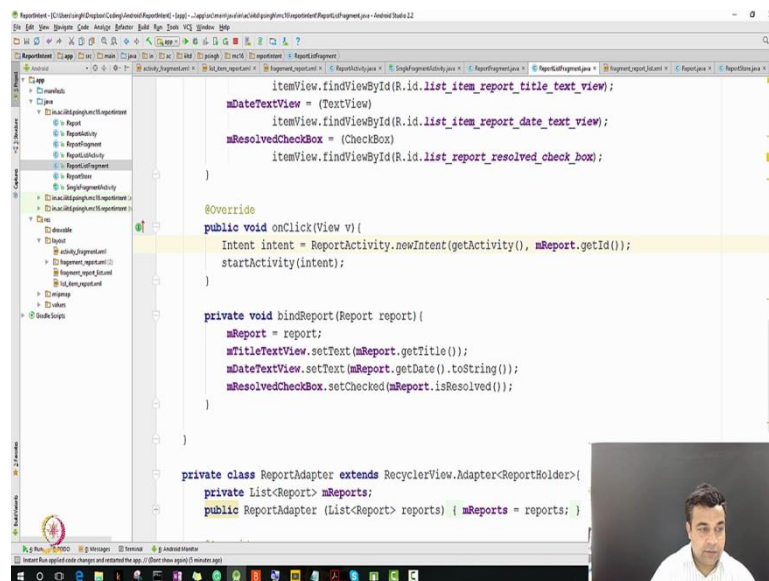
private class ReportAdapter extends RecyclerView.Adapter<ReportHolder> {
    private List<Report> mReports;
    public ReportAdapter(List<Report> reports) { mReports = reports; }
```

So, I will remove this code and I will write intent, because I am going to start an intent = new intent the context and the actual class, so this is an example of explicit intent as you may have guessed. So, I do this then I do start activity intent that is it I do a I do an all int so my error will (()) (03:03). So, now I have started an activity from my own click, so when I do the onClick I take an intent which is nothing but a message object. I give it the actual path, so this is an explicit intent and then I can start that activity, ok. Now, let us try to run our program and to see that what is the impact of these two lines here. So, we start our program, we look at it so the build is still running, yes.

more code so public static final String and (05:30) final and string it will be capital EXTRA_REPORT id = now you would like to give a unique name, so one we have deciding this name is to decide it to just like the way have decided your package name so for example, in my case I will just call it in.ac.in.ac.psing something which just makes it unique for me in.psing.mc16. and then I can just give it the my name of my program.report_id.

And then I create a new intent where I will use extra to pass the extra information. So, I create, so we have done that earlier in the math quiz application as well. So I think you can relate it to that, that whenever we have to do something that is pass an extra we write the code like this V reportId. So here we do nothing but Intent = new Intent context comma ReportActivity.class and then intent.putExtra EXTRA_REPORT_id reported, then I will return this intent, so yes. So, now I am using an extra method is same as we did in themath quiz that we created the new intent method and report an extra and this time we are passing the reportId. So, after creating this explicit intent and calling put extra to pass we can now update our ReportHolder class to use this new intent method.

(Refer Slide Time: 8:41)



```

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 = ReportActivity.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());
}

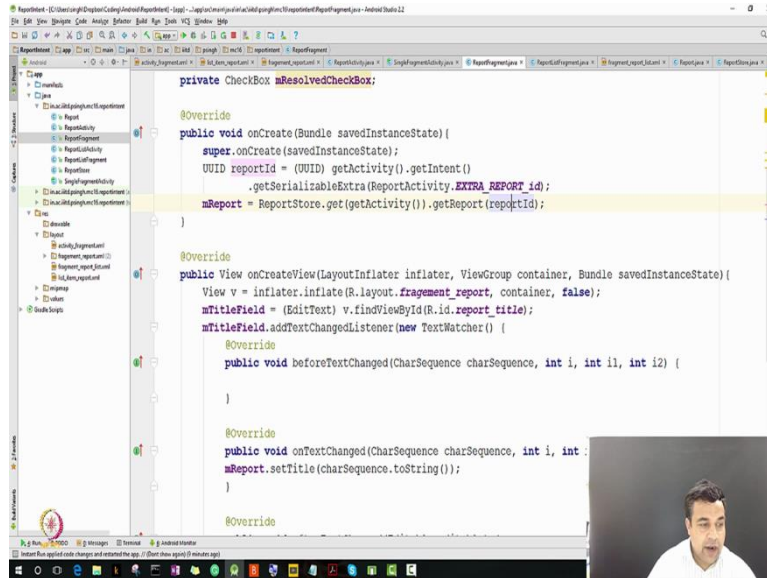
private class ReportAdapter extends RecyclerView.Adapter<ReportHolder> {
    private List<Report> mReports;
    public ReportAdapter (List<Report> reports) { mReports = reports; }
}

```

So, let us go back to our previous class that we will change it the ReportHolder class. And we should just remove this line and change it with Intent intent = ReportActivity.newIntent method and pass it the activity and getId so that is fine. So, now we are passing the id when we are sending the intent. Now, let us because we are passing an extra we will have to retrieve an extra that is the second part of the work. So, now let us write some code to retrieve the extra. So, now our reportId is within Intent and there are two ways in which a fragment can access data and its activity is Intent because this is passing through the activity

and we need to get the date and the fragment. So, one way is an easy and direct short cut way which we will be using here. And then there is another way which is kind a little bit more complex and use this FragmentArguments which we will learn today in this lecture.

(Refer Slide Time: 10:41)



```
private CheckBox mResolvedCheckBox;

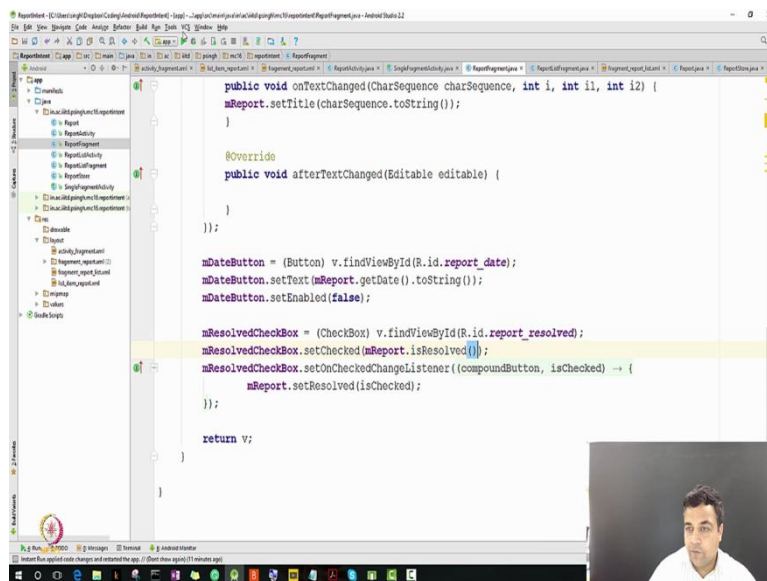
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    UUID reportId = (UUID) getActivity().getIntent()
        .getSerializableExtra(ReportActivity.EXTRA_REPORT_ID);
    mReport = ReportStore.get(getActivity()).getReport(reportId);
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fragment_report, container, false);
    mTitleField = (EditText) v.findViewById(R.id.report_title);
    mTitleField.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {
        }

        @Override
        public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
            mReport.setTitle(charSequence.toString());
        }
    });
}
```

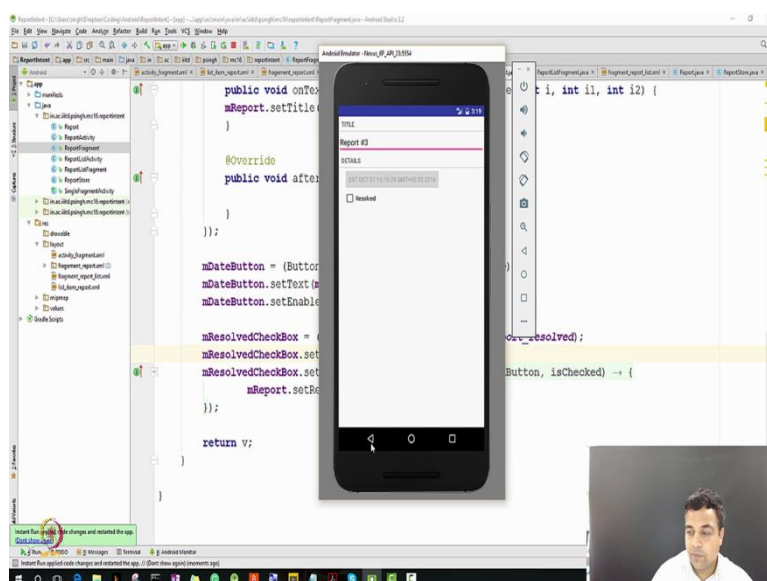
However, first let us start with the easier method and in the easier method we simply use the getActivity method of the fragment and we access the Intent direct and later on we will use the more complex method. So, let us use it in the first, so we will go to our ReportFragment class and we will go to the onCreate method of it where we are passing the bundle. And this is fine and the onCreate and now instead of this we will be adding some more code which is that VUIDreportId = VUIDgetActivity so the fragment is just calling the getActivity method and getting the Intent of the activity that it wants. And, then it is going to get the data so ReportActivity.EXTRAREPORTid, then we the line which we just deleted we add it but in a different way.getgetActivity.getReport and our reportId.

(Refer Slide Time: 12:54)



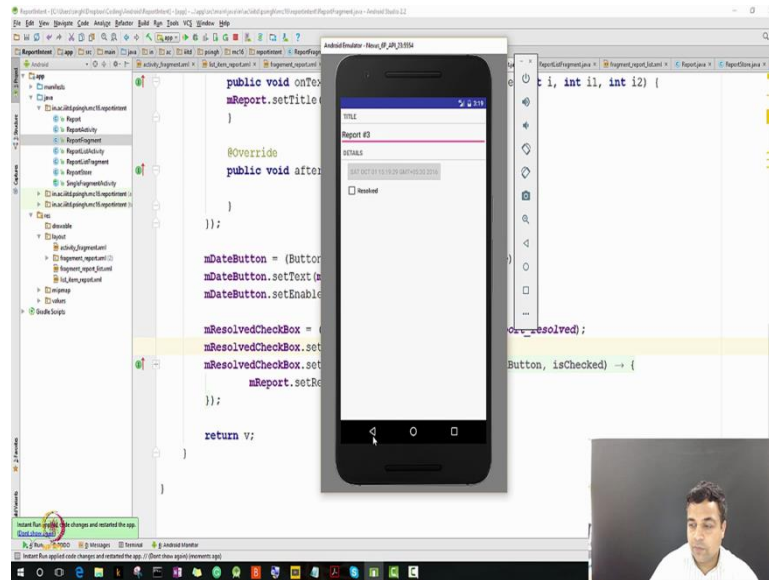
So, Alt Enter, so yeah so now this part is done. So, now we have to also do some update into the way we had written the ReportFragments view. So, let us add some more code but this time in the onCreate View. So, essentially first step is that apart from the TitleField we would I we would also be adding few more details here. So, let us come here and say mTitleField.setTextmReport.getTitle and then we can also do in the ResolvedCheckBox that mResolvedCheckBox.setCheckedmReport.isResolved. So, here I did some code to update the view and now let us watch what difference it has got in. So, earlier if you remember it was only showing an empty detail. Now, you will see what difference it has made, so earlier it was empty detail.

(Refer Slide Time: 13:36)



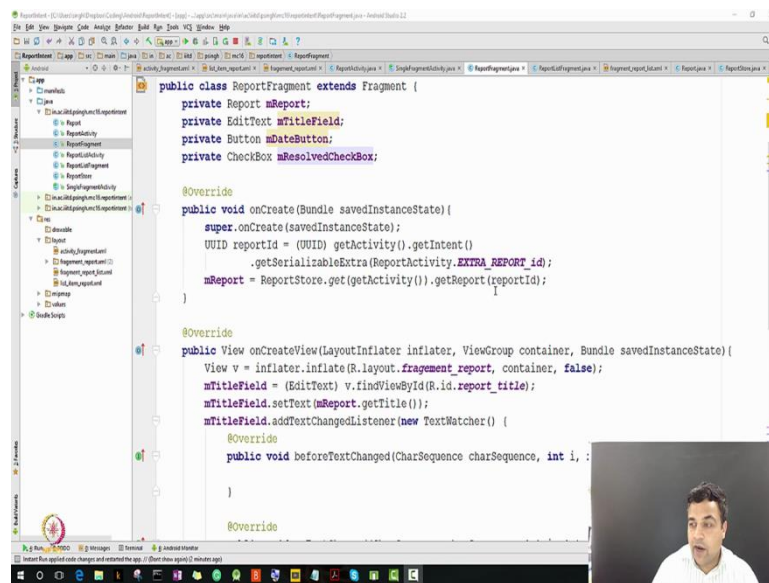
Now, let us see we are clicking on report number 3, now you see that the title is filled in. Let me go back so and this time I will click on report number 4 where we have already put the ResolvedCheckBox.

(Refer Slide Time: 13:49)



So, now you see it comes as resolved while by press on report number three it is not. So, essentially using the Intent and the extra we are passing 2 values one is the Title and one the status of this CheckBox. So, and then we are able to successfully pass it and the fragment would call the getActivity method and could get access to the Intent data and close it. So, this was fairly easy, now this was fairly easy as did fast but there is a down side to it. Number one, that we always insisted the fragment should be dependent of the activity which are hosting in so that they can be used across different activities.

(Refer Slide Time: 14:39)



```
public class ReportFragment extends Fragment {
    private Report mReport;
    private EditText mTitleField;
    private Button mActionButton;
    private CheckBox mResolvedCheckBox;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        UUID reportId = (UUID) getActivity().getIntent()
            .getSerializableExtra(ReportActivity.EXTRA_REPORT_ID);
        mReport = ReportStore.get(getActivity()).getReport(reportId);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_report, container, false);
        mTitleField = (EditText) v.findViewById(R.id.report_title);
        mTitleField.setText(mReport.getTitle());
        mTitleField.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, .
            }
        });
    }
}
```

However, in this case if you see the way we have written the code our fragment is dependent on an activity which is having this variable called EXTRAREPORTID. So, this actually limits the use of our fragments, reuse of our fragments which may be ok in our particular case but is not a good approach. A good approach or a better solution would be that we that we keep this id somewhere else in a space which belongs to the ReportFragment but then rather than keeping it in the reportActivities for small space. And then report fragment can access this data whenever it wants without line on the reportActivities so currently it is relying on the reportActivity which is not good.

We want to separate the both process out and this functionality is achieve what we call the fragment arguments. So, let us see that how to use fragment arguments in our program. So, each fragment instance can have a bundle as you can see can have a bundle object associated with it. And this bundle can have value key value pairs of the data just like we saw earlier in case of activities. And we can use some of these to store the information that we want.

So, this key value pair inside the bundle is known as argument and that is what we are going to use it. So, in order to create a fragment argument to that to use the bundle object than do it. So, let us get started by first creating the bundle objects and then attaching the arguments to it which the fragment can use later on. So, in order to use these arguments we will have to first create a bundle and then attach the arguments bundle to the fragment by calling a method called set arguments fragment.setarguments bundle uhh.

And this attachment must be done before the fragment is associated with an activity. ok, so usual method in android programming is to add a static method name new instance the static method is name new instance to the fragment class. So, that this method can create the fragment instance and bundles up and sets this update. So, when the hosting activity needs an instance of a fragment, we will have to call the new instance method rather than calling the constructive directory. So, essentially what we are saying is that instead of a constructor of a report fragment, we will be creating a new instance method of the ReportFragment. And, when the activity needs to use ReportFragment it will have to call that (()) (17:20). So, let us get started with programming.

(Refer Slide Time: 17:33)

```

package in.ac.iiitd.psingh.mcl6.reportintenc;

import ...

/**
 * Created by singh on 24-Sep-16.
 */

public class ReportFragment extends Fragment {
    private static final String ARG_REPORT_ID = "report_id";

    private Report mReport;
    private EditText mTextField;
    private Button mDateButton;
    private CheckBox mResolvedCheckBox;

    public static ReportFragment newInstance(UUID reportId) {
        Bundle args = new Bundle();
        args.putSerializable(ARG_REPORT_ID, reportId);

        ReportFragment fragment = new ReportFragment();
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {

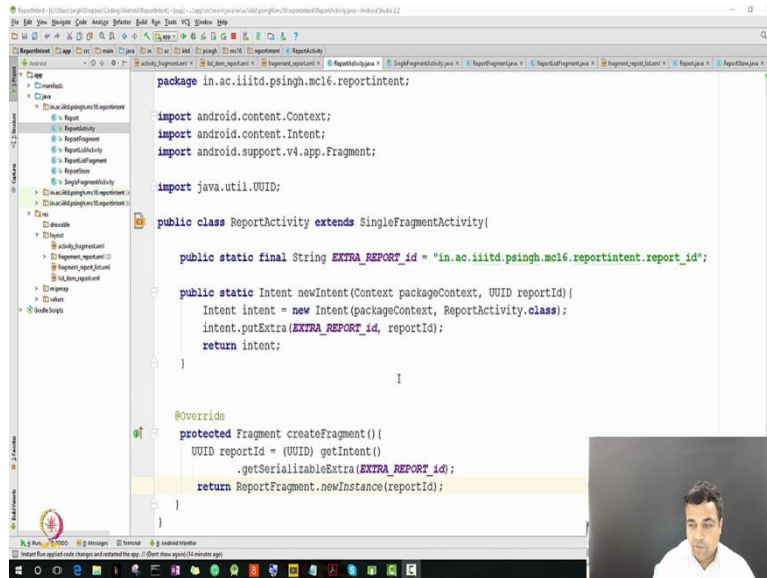
```

Private static final String ARGREPORTID = report_id and now, let us go and put some public static report ReportFragmentnewInstance that is the method we want to create UUIDreportId. And inside this static method we will create our bundle bundle and then we will put few things into it Id.reportId and then we will return the fragment returnfragment so yes. This is now this is the method that we will now be using instead of the constructor of the ReportFragment directory. So, now my reportActivity class should call this method when it needs to create the ReportFragment and not the not the constructor. And when it calls it will pass in the VUID it receives from its extra. And then return to the reportActivity and then create fragment retrieve the extra from the reportActivities Intent and pass it in the ReportFragment new instance method.

So, that is how we will, so this is a slightly complicated loop that first we create an instance and the reportActivity passes it and then it retrieves, then the earlier method where we will be

just calling that activity. But in this method our fragment will be independent of the report activity, so that is the best.

(Refer Slide Time: 21:22)



```
package in.ac.iiitd.psingh.mc16.reportintenc;

import android.content.Context;
import android.content.Intent;
import android.support.v4.app.Fragment;

import java.util.UUID;

public class ReportActivity extends SingleFragmentActivity{

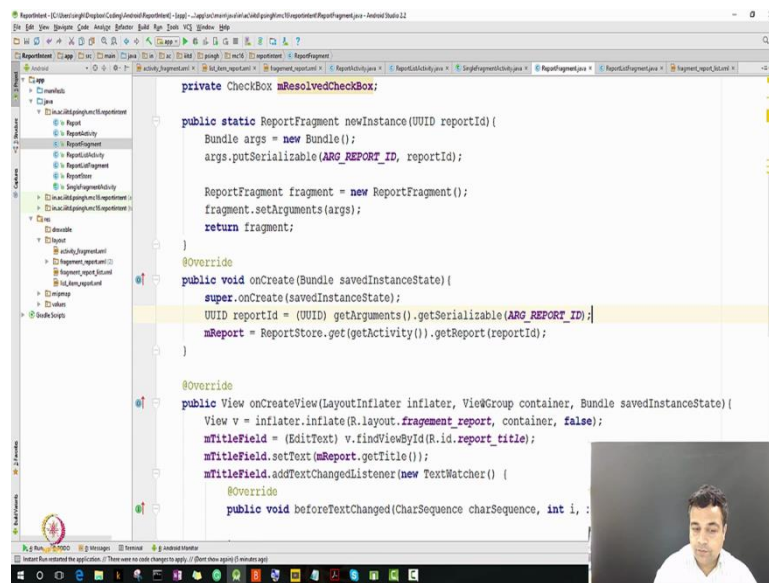
    public static final String EXTRA_REPORT_id = "in.ac.iiitd.psingh.mc16.reportintenc.report_id";

    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, let us see how do we achieve it. Let us go to the reportActivity now first and instead of instead of this line let us use something else here, which is to change the create fragment with not just this VUID reportId = VUID getIntent.getSerializableExtra EXTRAREPORTid and then return ReportFragment.newInstance.reportId. So, yes so now we have added good amount of code here which can help help our reportActivity to call the fragment using the new method. Now, here you will notice that the activity has to still depend on the fragment, because that is what the need of that application usually asked. The activities have to know about the fragments that they are using but there is no need for the fragments to know which activity is going to use in that.

(Refer Slide Time: 23:59)



```
private CheckBox mResolvedCheckBox;

public static ReportFragment newInstance(UUID reportId) {
    Bundle args = new Bundle();
    args.putSerializable(ARG_REPORT_ID, reportId);

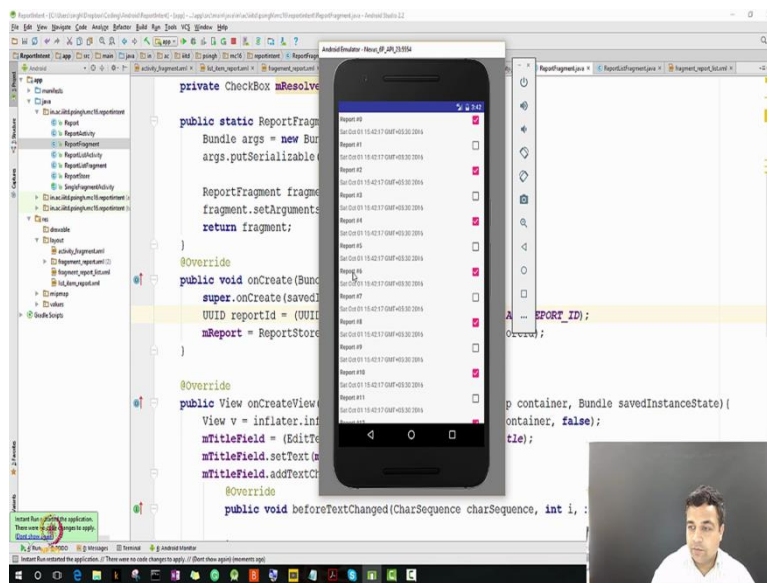
    ReportFragment fragment = new ReportFragment();
    fragment.setArguments(args);
    return fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    UUID reportId = (UUID) getArguments().getSerializable(ARG_REPORT_ID);
    mReport = ReportStore.get(getActivity()).getReport(reportId);
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fragment_report, container, false);
    mTitleField = (EditText) v.findViewById(R.id.report_title);
    mTitleField.setText(mReport.getTitle());
    mTitleField.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence charSequence, int i, :
    });
}
```

So, the independence is not your `()` (22:27). Now, let us focus on the next step which is to retrieve the arguments, so when a fragment needs to access its argument, it needs to call a method call `getArguments` then one of the type specific get method of bundle to get the specific value. So, let us write some code and try to see that how it is done, we will go back to the `ReportFragment` class and in that we will go back to the `onCreate` method and we will try to see what we can do here. So, ok so one more thing we now can make it a private and it will be fine. So, it will be fine it will not affect it program uhh. So we can make it as private and now let us go back and as you see that this is the older method. So, we will now remove it and we will change it to the new `UUIDgetArguments.getserialgetserializable` then the `REPORTID`.

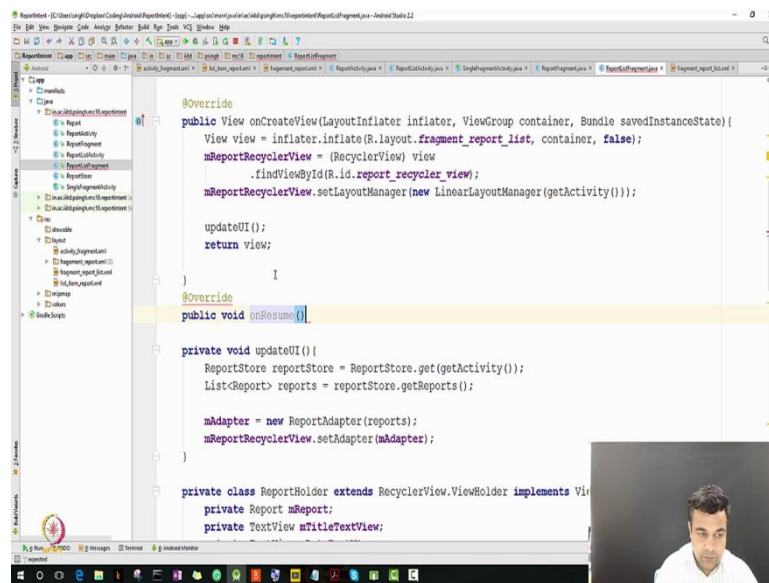
(Refer Slide Time: 24:10)



So, that is it now let us try to run our program and see if it is still working and doing the same. So, now we can see that it is still doing the same thing but our fragment no longer depends on the activity and our fragment is now totally independent. So, this is a better way of doing passing the data using fragment arguments, so now let us see our our application one more time.

So, I am clicking on Report1, then I am clicking back. Now, currently the Report1 is unchecked, let me put a check here and come back, ok so it is still unchecked. So what is happening is that I am modifying my report detail but and these changes are saved in my program but when I am returning my RecyclerView it is still unchanged and what I need to do is that I need to inform the adaptor of my RecyclerView that the data has changed so that it can refresh the data and reload the list. As we have solved this problem earlier we will try to solve it by writing some code. Essentially we will have to modify the onResume method which is called when an activity is destroyed by pressing the back button and then we create it. So, let us see what we write so let us go to the report list fragment and so far we had not overridden the onResume but now there is a need for it.

(Refer Slide Time: 26:02)



```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_report_list, container, false);
    mReportRecyclerView = (RecyclerView) view
        .findViewById(R.id.report_recycler_view);
    mReportRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));

    updateUI();
    return view;
}

@Override
public void onResume() {
    updateUI();
}

private void updateUI() {
    ReportStore reportStore = ReportStore.get(getActivity());
    List<Report> reports = reportStore.getReports();

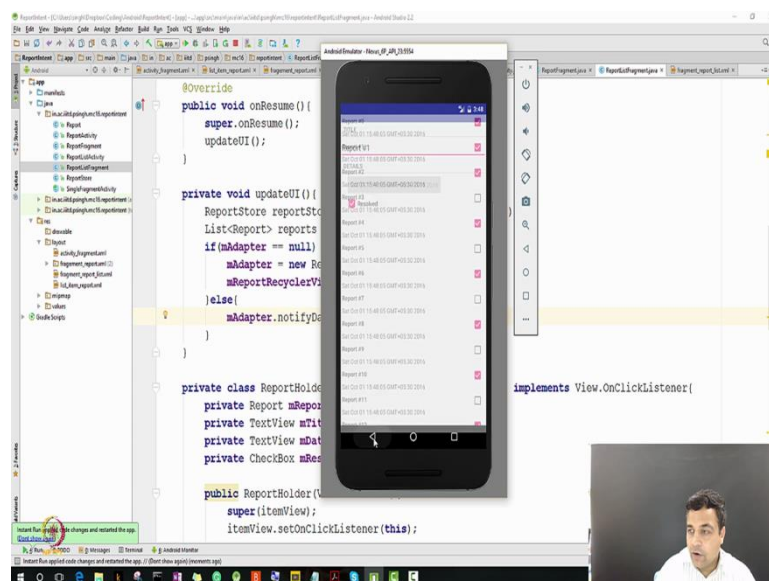
    mAdapter = new ReportAdapter(reports);
    mReportRecyclerView.setAdapter(mAdapter);
}

private class ReportHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
    private Report mReport;
    private TextView mTitleTextView;
```

So, we override super.onResume and all we have to do is to call this update UI method because this is what needs to be called. So, update UI method, so we have called it. Now, another thing which we need to do is we need to do a smaller change in it because now we are here calling it in the onResume as well. So just a little bit of housekeeping code which says if mAdapter = null than do the same thing we have been doing so far.

But, else just do call a method which is called NotifyDataSetChanged. So, this method is there to make sure that the adaptor reloads it and now let us try to see that what happens when we run our program now. So, we made just very small change by onResume which we did and just to take care of the change.

(Refer Slide Time: 27:41)



```
@Override
public void onResume() {
    super.onResume();
    updateUI();
}

private void updateUI() {
    ReportStore reportStore = ReportStore.get(getActivity());
    List<Report> reports = reportStore.getReports();
    if (mAdapter == null) {
        mAdapter = new ReportAdapter(reports);
        mReportRecyclerView.setAdapter(mAdapter);
    } else {
        mAdapter.notifyDataSetChanged();
    }
}

private class ReportHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
    private Report mReport;
    private TextView mTitleTextView;
    private TextView mDateTextView;
    private CheckBox mDeleteCheckBox;

    public ReportHolder(View itemView) {
        super(itemView);
        itemView.setOnClickListener(this);
    }
}
```

implements View.OnClickListener {

So, now we click program here we go so as you see Report1 is unchecked here click I check it go back and now it is checked. So, now our program is a complete program where we have going from the list of details to the details coming back everything is in sink and everything is working. Now, in this case if you remember when we were discussing activities and Intent then we said you know what if you want a result back then you have to start an activities differently. But, if you do not want the result back then you have to start an activity differently.

Same thing here, here we did not need to get results, while we were working. But if we were working it then we would have to start our activity differently. Essentially, calling the method start activity for result and then modifying the onActivity result method to retrieve the result value. This thing we did not need here, but you may need in your other programs, the fundamentals are same as we studied in activity for the fragment as well. So, thank you and let us do some more coding in another class, where we will be adding more functionalities to this program, so thank you.