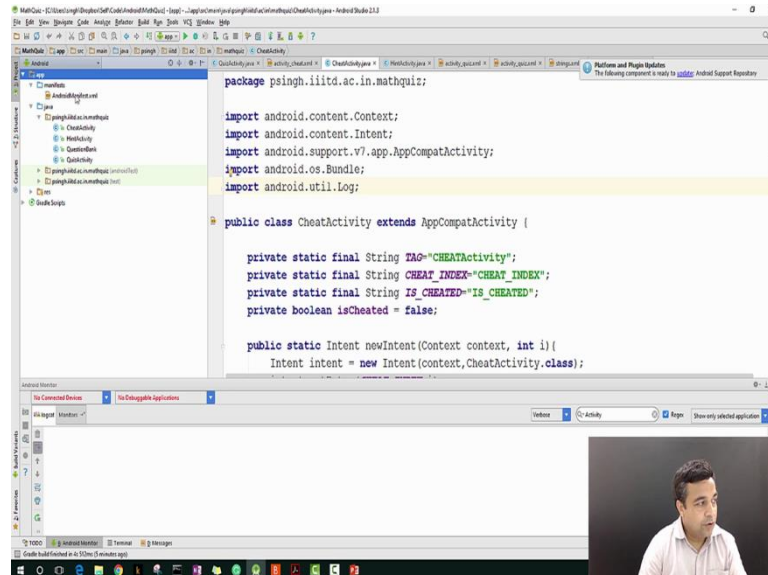


**Mobile Computing**  
**Professor Pushendra Singh**  
**Indraprastha Institute of Information Technology Delhi**  
**Lecture 21**  
**Intents**

Hello, let us continue our lecture, so yesterday we created another activity in our application and we used Intents to transfer data from one activity to another activity. First we transferred the data from the parent activity to the child activity and then we transferred some data from the child activity to the parent activity. We did this using Intent extras and using some function calls that are defined by Android. Today, let us go back to the same application and try to transfer some meaningful data. For example, we will transfer the index number of our question. We will see the answer of that question in our child activity and then we will transfer back, if the person has cheated to the parent activity and based on that we will do our toast that indicates it.

So we are going to make some more changes to our Math Quiz Applications using the knowledge that we learnt yesterday. Let us see. So this is our yesterday's code.

(Refer Slide Time: 1:32)



```
package psingh.liitd.ac.in.mathquiz;

import android.content.Context;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

public class CheatActivity extends AppCompatActivity {

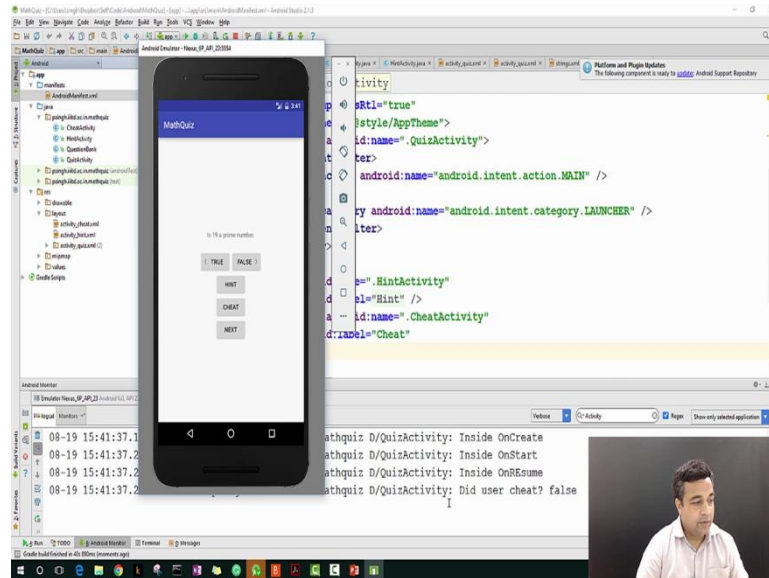
    private static final String TAG="CHEATActivity";
    private static final String CHEAT_INDEX="CHEAT_INDEX";
    private static final String IS_CHEATED="IS_CHEATED";
    private boolean isCheated = false;

    public static Intent newIntent(Context context, int i){
        Intent intent = new Intent(context,CheatActivity.class);
```

Just to quickly revise. We now have 3 activities in our program; you can check it by looking at the Manifest. Let me also add a label for our activity, so I will call it Android, label, label = Cheat that is it, this is fine. Now our new Android activity should display Cheat on the top as a lay. So let us first have a quick revision of what our application was doing. We run our application, we start the emulator. We have added good number of log messages to check the Android application or activity lifecycle. Now let us just quickly have one execution and then

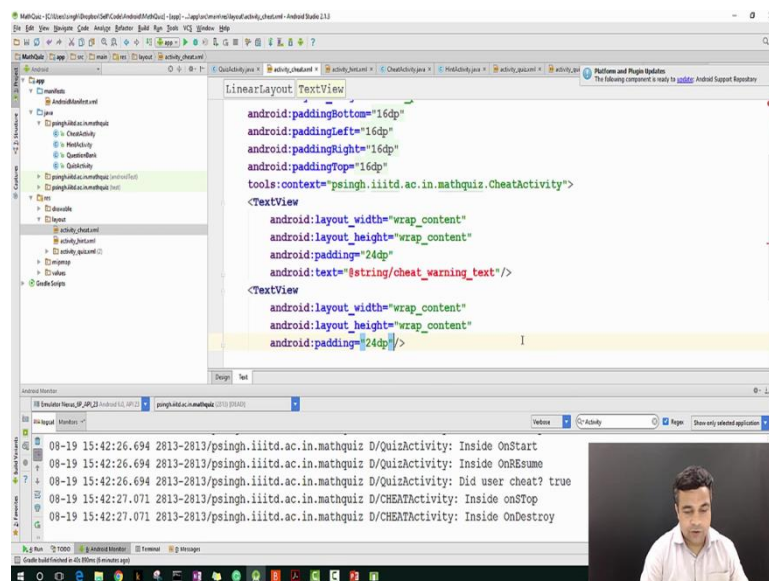
we will move on to do some meaningful changes to make our Math Quiz Applications more interesting. Ok, our application is about to start and now it has started.

(Refer Slide Time: 3:17)



As we can see the log, we are currently in Quiz activity, it gives the Cheat as False, I can answer, I can go to Next, everything is fine and I can go to Hint. You can see the hint and come back; I have still not cheated though I did go in to the hint as shown on the console. Now let me cheat, there is nothing but our label is now changed, which we just did in the manifest. Now we come back and this time we are saying that the user has cheated, that is all very good. Now we are going to actually add two functionalities in our Cheat activity.

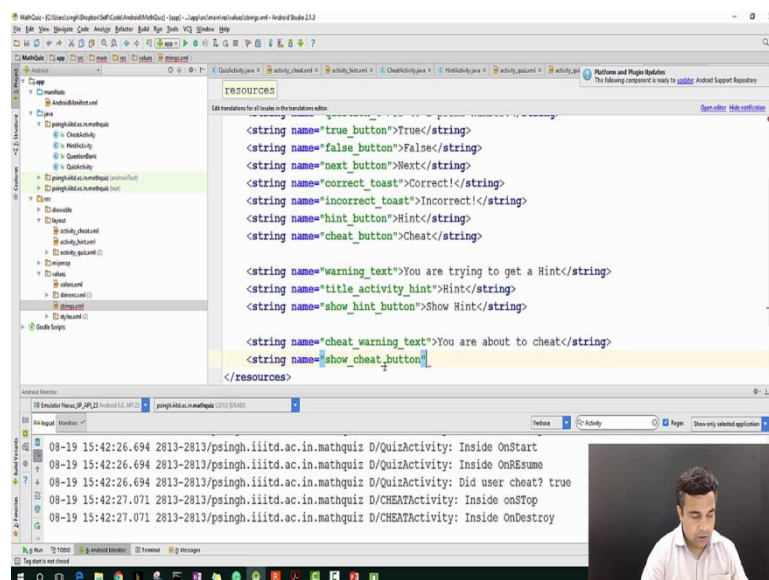
(Refer Slide Time: 4:16)



So Number 1, let us first see where is the XML of our Cheat, some of you have mentioned that you get rendering problems when you are writing your code. I usually press this refresh button and all my language problems go away, just as you saw. So in case you face a rendering problem please try to make use of the rendering, of the refresh button. Ok, so this is our layout of our Cheat, Cheat activity. I will make few changes, number 1 as usual I will make it a linear layout and then it looks all good. Let us start adding some code into the XML file of our Cheat activity. As you can see that now the label is Cheat which we changed in the Manifest. In the text we have a simple text, like I have already converted the relative layout which is done by default to the linear layout and now I am going to add few more things.

Number 1, I want to add a text view so opens a tag, text view. Wrap content is good for me, I will do some padding. Some of you all have also asked a question that how much should be the padding. I usually decide padding depending on how it is, how it looks on a screen, usually 24DP padding is good enough in most of the situations. However, no hard and fast rules are there, you can use a padding which looks good. Now, what we want to do? Number 1, I want to give you a warning that you are going to cheat and this is not a good practice. Cheat warning text, we will create the associated spring resource later on. Let there, let the error be there for the time being. Another text view I will add, and here I would like to show the cheat, i.e. to give the answer that whether the number is Prime or not. Now again I will use the same padding of 24DP, which looks very good on my device.

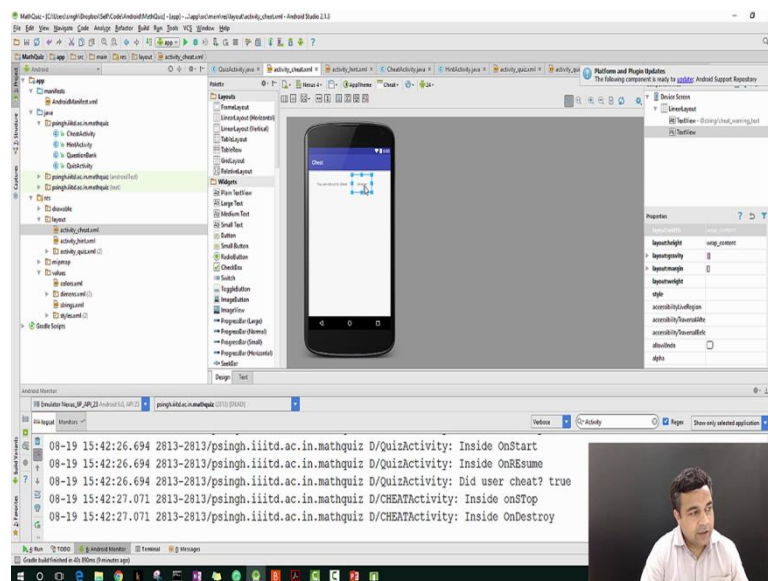
(Refer Slide Time: 9:11)



Now, there is something new that I am going to use here. I am going to use something called tools, text = answer or actually you can say anything. Now the, for using the tools text we

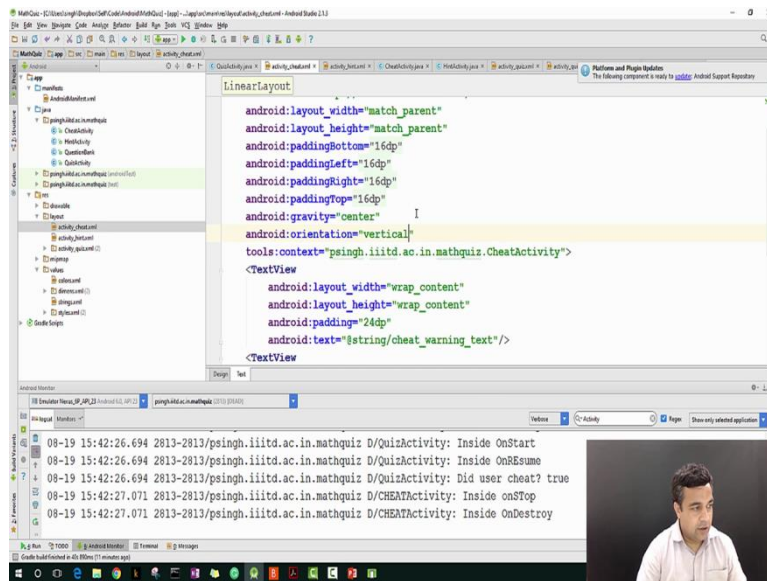
already have the namespace defined, we already have the context defined. So what does the tools text do for you? Let us go into the design and see, again we are having some rendering problem. Ok, I will have to first fix it then only I can show it to you. So, let us see and go to the values, the strings and I would say here, first going to name, cheat warning, text “You are about to cheat”. This is a good enough for me. I will, because I will be adding a button I will add this string resource here while I am working on this. Oops I have a mistake, cheat.

(Refer Slide Time: 9:43)



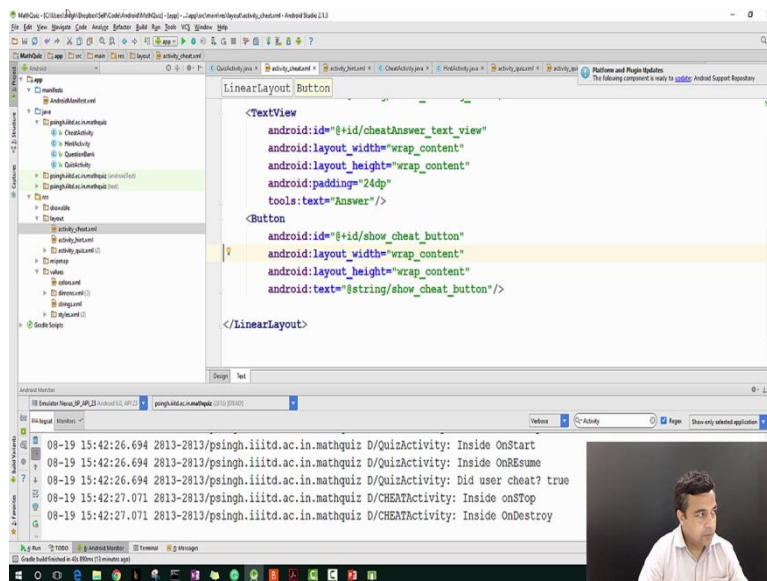
Now I will call it a Show cheat button I will give it value, Show cheat. Ok, so these 2 are sufficient for me, now let us go back to our layout file, my errors are gone, let us see the design, what it is showing is that you are about to cheat and it is showing answer. Now, if I run the application, I will actually not see the answer and that is the whole purpose of the tool of the tools: text field. This just shows me that how it will look without actually displaying it till I run the application. At the time of the application this will go away and ideally my answer will come here. So the value answer is only a place holder. If I had not used tools text then I would be getting a fixed text, something like this, which I cannot change. Ok, or which I would not like to change.

(Refer Slide Time: 10:54)



Fine, let us continue and because we will be changing the text, we need to create an ID for this. So as usual, we will now create an ID, /ID and we will call it a name, Cheat answer text view, that is fine, I was seeing the text on the very top, let me bring it down near to the center, I will have to add a gravity = center. Let me just show you what happens if we add this part. You see that now it has come from the top to the center and while I am added I will also add the orientation to vertical. Let us see what changes it makes, yes so now it is showing my fields in a vertical layout, my view objects in a vertical layout.

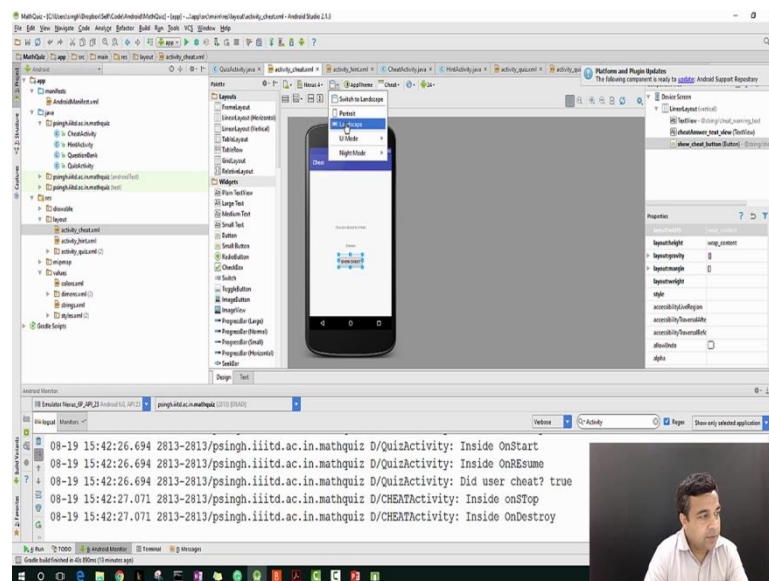
(Refer Slide Time: 12:18)



That is the warning text and that is how my answer will appear, this is all good, except that I would also like to add a button. So let me add a button, in that content is good enough and I

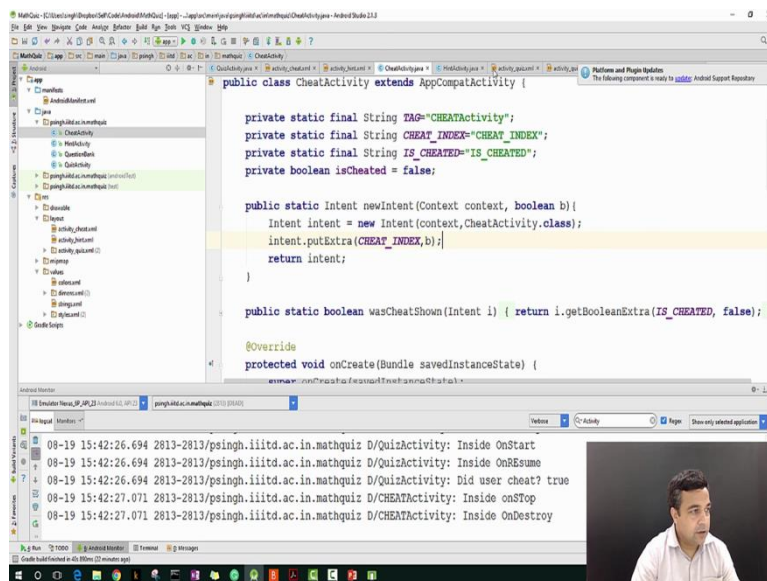
will add a text equal to we already created a string, hopefully we should get cheat button, Show cheat button, ok, so this was our button that we wanted to create, because we will be using this button in our program, we also need to create the ID for this : = Now you must have understood whatever resource we want to use actively in our program, we need to create an ID. Hmm, so what should be the ID, let us call it a show cheat. So we have a show cheat button and we have a string resource that is all fine.

(Refer Slide Time: 14:00)



Now when I click on the button, I should see a button. That is what I did ok; now the layout of our cheat file is complete. I may want to check how it looks in landscape and this is fine with me, so I will not be creating a separate landscape file. Ok, now let us go back and make some changes to pass the values. In our previous lectures testing our concept, we just tried to pass any value. Today we will specially pass the value of this variable because this is the value which is showing our result, right? So where are we using this variable? As you can see that we would be using this value to display our toast. That is when a user was pressing true or false button, we were calling check answer and in check answer we were setting, we were using the value, from this value we were showing the toast.

(Refer Slide Time: 15:30)



```
public class CheatActivity extends AppCompatActivity {

    private static final String TAG="CHEATActivity";
    private static final String CHEAT_INDEX="CHEAT_INDEX";
    private static final String IS_CHEATED="IS_CHEATED";
    private boolean isCheat = false;

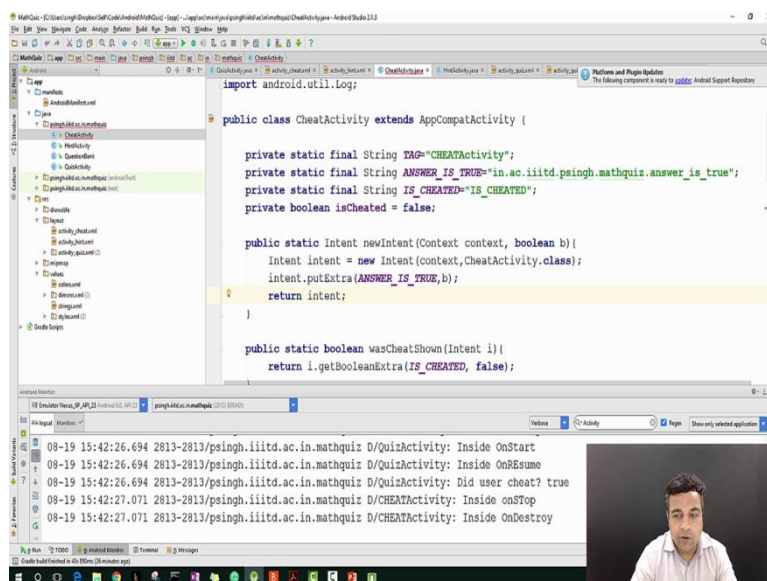
    public static Intent newIntent(Context context, boolean b){
        Intent intent = new Intent(context,CheatActivity.class);
        intent.putExtra(CHEAT_INDEX,b);
        return intent;
    }

    public static boolean wasCheatShown(Intent i) { return i.getBooleanExtra(IS_CHEATED, false); }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

Now let us pass this value. Let me copy it, control c, let me go to the Quiz activity where we were trying to create the Intent, so we did the Intent when we were at the cheat activity. So number 1 thing is that the cheat activity extra created a int, so I will now remove this int and I will call it a Boolean b ok, make it b that is fine. I will now come back to Quiz activity and I will just have a Boolean b = isTrueAnswer and I will pass this b here ok, so now I have successfully passed the Boolean, let me remove the statements, because now we already understand the program very well.

(Refer Slide Time: 17:11)



```
import android.util.Log;

public class CheatActivity extends AppCompatActivity {

    private static final String TAG="CHEATActivity";
    private static final String ANSWER_IS_TRUE="in.ac.iiitd.psingh.mathquiz.answer_is_true";
    private static final String IS_CHEATED="IS_CHEATED";
    private boolean isCheat = false;

    public static Intent newIntent(Context context, boolean b){
        Intent intent = new Intent(context,CheatActivity.class);
        intent.putExtra(ANSWER_IS_TRUE,b);
        return intent;
    }

    public static boolean wasCheatShown(Intent i){
        return i.getBooleanExtra(IS_CHEATED, false);
    }
}
```

So, now we are passing the Boolean variable. Now let us go to the cheat activity and try to see what changes we need. One change which was needed earlier we have already done. So

the another change that we need to do is to change this string index to reflect what we are putting now, because we are no longer putting the cheat index. So let me change it to, let us say answer is true. When you are developing an app just for yourself, you need not worry about it, however if you want to deploy app on the Play store you need to make sure that the values that can be used are unique. Why? Because, as we know in Android that any application can start activity from another application. There is a good possibility that another application may want to store some values and may want to use a commonly defined name.

Now hint, cheat these are very common names. So let us try to make it slightly complicated so that we can ensure that it is more likely to be used. One way of doing it is to use the package definition here. So I would just call it whatever is my package that is in.ac.iiit.psingh.mathQuiz.answer is true. Now I am fairly sure that nobody will like to guess and end up with the same string. So this is one way to ensure that you are using unique key values. Now let us just use our new key value. So one required change we have done, that is now we are passing the extra which is a Boolean and we are accessing that extra here.

(Refer Slide Time: 19:14)

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cheat);
    Log.d(TAG, "Inside onCreate()");
    isCheat = getIntent().getBooleanExtra(ANSWER_IS_TRUE, false);

    setAnswerResult(isCheat);
}

private void setAnswerResult(boolean b) {
    Intent i = new Intent();
    i.putExtra(IS_CHEATED, b);
    setResult(RESULT_OK, i);
}

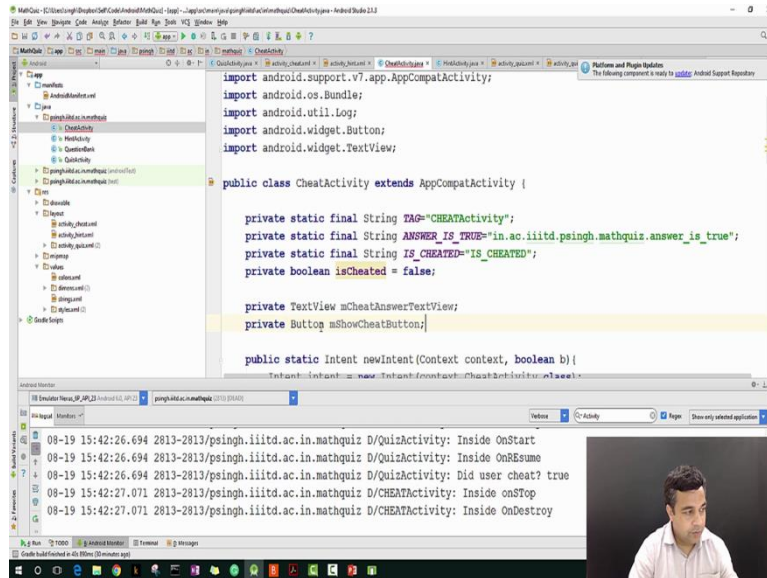
@Override
public void onStop() {
    // ...
}
```

Now let us look at some other changes just because we have made that changes, we also need to make changes in the onCreate function. So because last time we were accessing an int using this, now we are accessing a Boolean. So let me do get Boolean extra and we know our string, answer is true, let us say that the default value is false. We will have to assign it to a Boolean b or the better thing is to assign to this variable isCheat directly, so now we have a isCheat. So yes, and we will have to also remove this message. So we have done some



basic required changes. Now we have to make more changes to display the result when a person presses the cheat button.

(Refer Slide Time: 20:48)



```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.Button;
import android.widget.TextView;

public class CheatActivity extends AppCompatActivity {

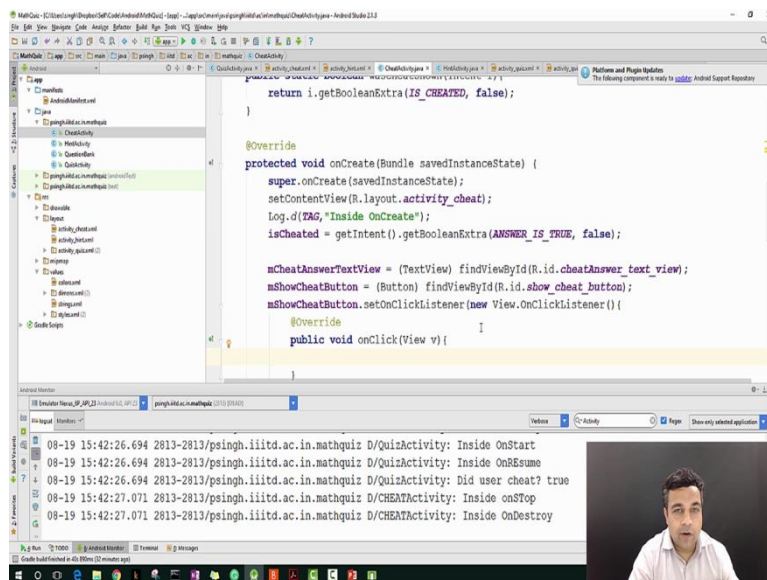
    private static final String TAG="CHEATActivity";
    private static final String ANSWER_IS_TRUE="in.ac.iiitd.psingh.mathquiz.answer_is_true";
    private static final String IS_CHEATED="IS_CHEATED";
    private boolean isCheat = false;

    private TextView mCheatAnswerTextView;
    private Button mShowCheatButton;

    public static Intent newIntent(Context context, boolean b){
        Intent intent = new Intent(context, CheatActivity.class);
        intent.putExtra(ANSWER_IS_TRUE, b);
        return intent;
    }
}
```

So, let us go back to onCreate, we are here, we have to enable the text view. Oh well, first we will have to describe a variable to hold the text view. So let us say private Textview mAnswer or mCheatAnswerTextView. And similarly private Button mShowCheatButton. I will press Alt Enter to make sure that I am reporting correct libraries.

(Refer Slide Time: 21:42)



```
return i.getBooleanExtra(IS_CHEATED, false);
}

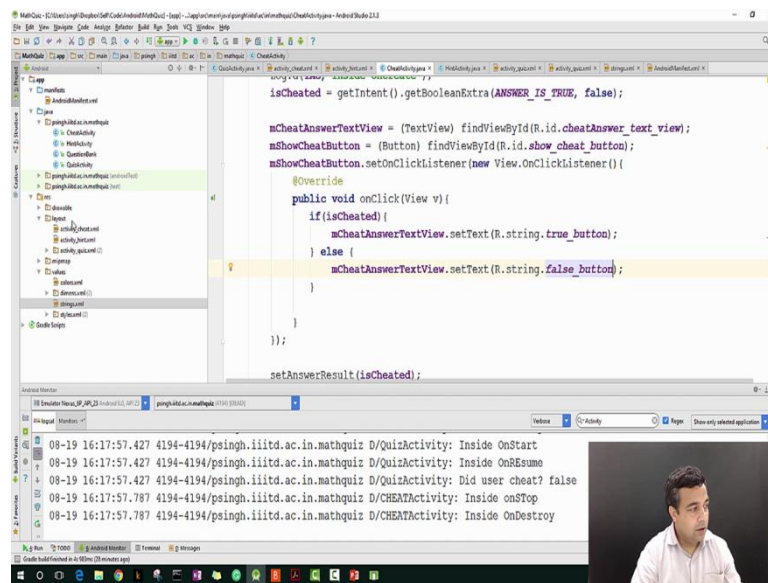
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cheat);
    Log.d(TAG, "Inside onCreate");
    isCheat = getIntent().getBooleanExtra(ANSWER_IS_TRUE, false);

    mCheatAnswerTextView = (TextView) findViewById(R.id.cheatAnswer_text_view);
    mShowCheatButton = (Button) findViewById(R.id.show_cheat_button);
    mShowCheatButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // TODO: Implement this method
        }
    });
}
```

Now, we will go down and I will say mCheatAnswerTextview = TextView, then we can use our familiar method, findViewById R.Id.cheatAnswer\_text\_view. Similarly we can show for

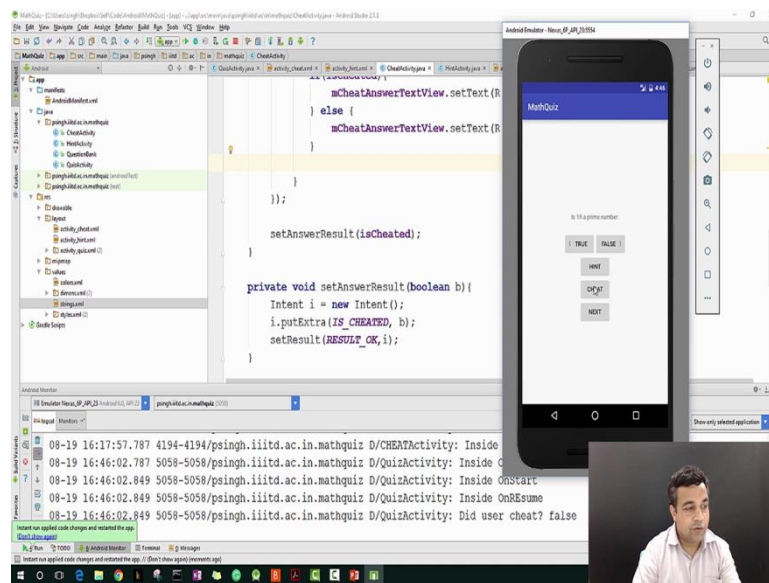
our button `mShowCheatButton = Button, findViewById, R.Id.showCheatButton` that is fine. Now, the next step is as you all know to set a listener `mShowCheatButton.setOnClickListener, new View.OnClickListener`, start a new method and do a override and then override the public void `onClick View v` method okay. Ok, so this is the method which will invoke when we press the Cheat button. Now, what do you want to show when the user presses the Cheat button is, number 1 we want to show is the value, the value that we obtained here Answer is true, Answer is true from here.

(Refer Slide Time: 23:55)



Now, let us see ok now let us add some functionality to the onClick, I would say, if `isCheated`. That is, if `isCheated` is true, my text view which is `mCheatAnswer.TextView` should display true otherwise it should display false. Now we already have created string resources with respect to true and false when we were creating our buttons, so let us just use them `R.string.isTruebutton` and else `mCheatAnswer.SetText R.string.falsebutton`. You can check the values of true button and false button, so we will get True and False. Now, let us come back. We seems to have done enough to display True or false when the cheat button is pressed, let us quickly see an execution.

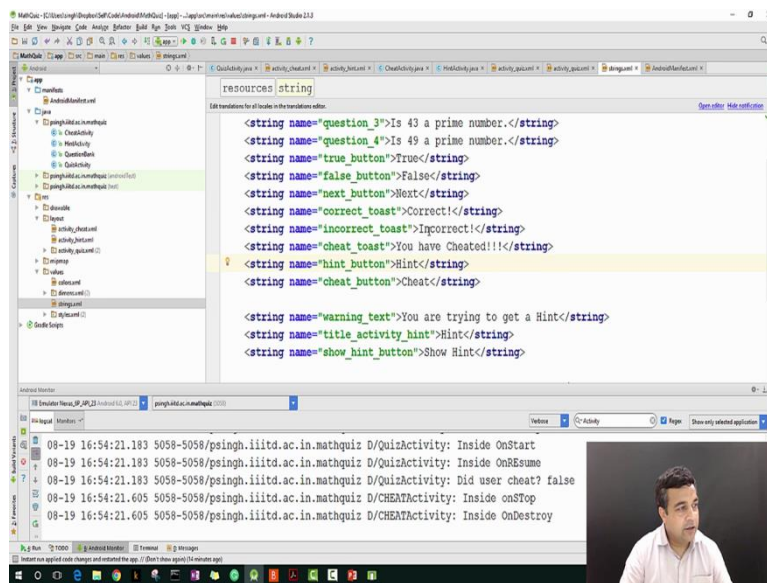
(Refer Slide Time: 25:18)



Our emulator is launched; we know the answer of this question. Let us press the Cheat button it shows our message, this space is empty because we were using tools: text. Now I will press the show cheat button and the answer is displayed. We go back and we know that we have cheated, now let us go to next next next question to the last question. We again press the cheat button and the correct answer is displayed. So we have completed a good number of functionality, as far as going from parent activity to cheat activity. We are now going to fix the reverse path that is, now when the user has cheated we are going to display a different toast, earlier we were displaying toast which was saying only correct or incorrect. Now we will display a toast that will say that the user has cheated.

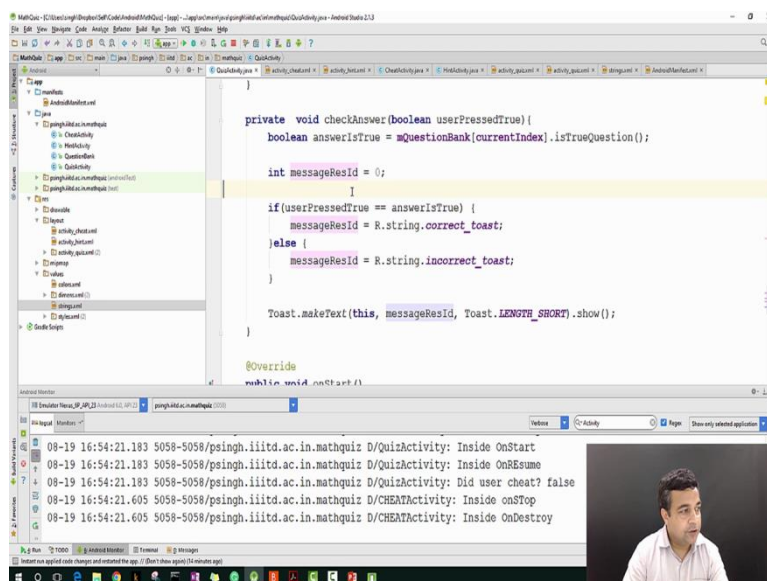
Ok, now let us fix the reverse path, earlier we were calling this method setAnswerResult in this we are doing this. Number 1 we will call this method only when isCheated is pressed and instead of taking that value we will say isCheated, because the user pressed the button, we know that he has cheated and that is why we are passing the value True and we are passing this value true and we are putting isCheated and the isCheated and the static value static function that this activity will call wasCheatShown will be either taking from here that is true otherwise if this is not set, this is not set then it will be taking as false, that will solve our purpose, right.

(Refer Slide Time: 27:48)



Now let us go into the Quiz activity and change our toast. First let us go to our String resources, initially we were showing the toast as correct or incorrect, let me add the other toast which says, this string, I will call it cheat toast and I will say “You have cheated, Try to solve it yourself next time” or may be just “You have cheated”. So this is our cheat toast, we created the resource, we will go to Quiz activity and now when the user presses the true or false button, that is when the user presses the true or false button here we want to show either true or false or the cheat toast.

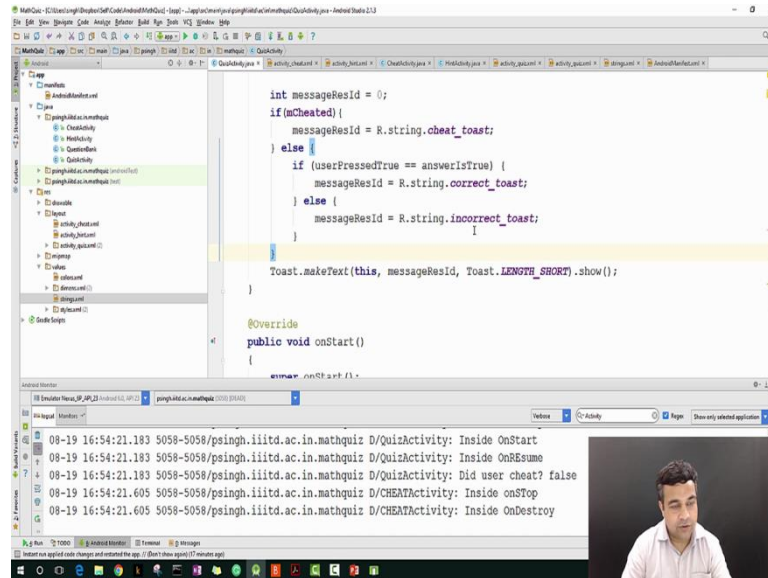
(Refer Slide Time: 29:23)



Let us do our checkAnswer, so in our checkAnswer we check this variable and we show the correct or incorrect toast. The only thing we need to change in the checkAnswer is that if the

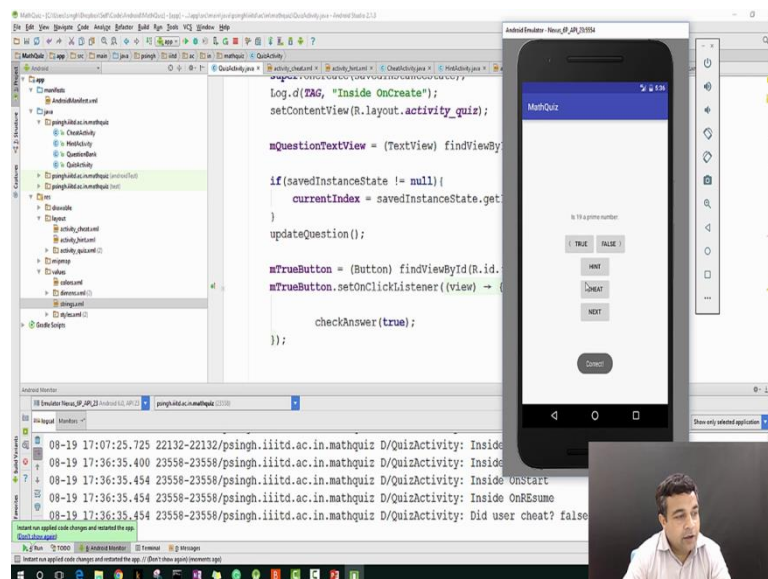
user is doing a cheating then the messageResId is set to the text of the cheat toast. So in order to do this I will first go to the method where we are receiving the result, onActivityResult.

(Refer Slide Time: 30:26)



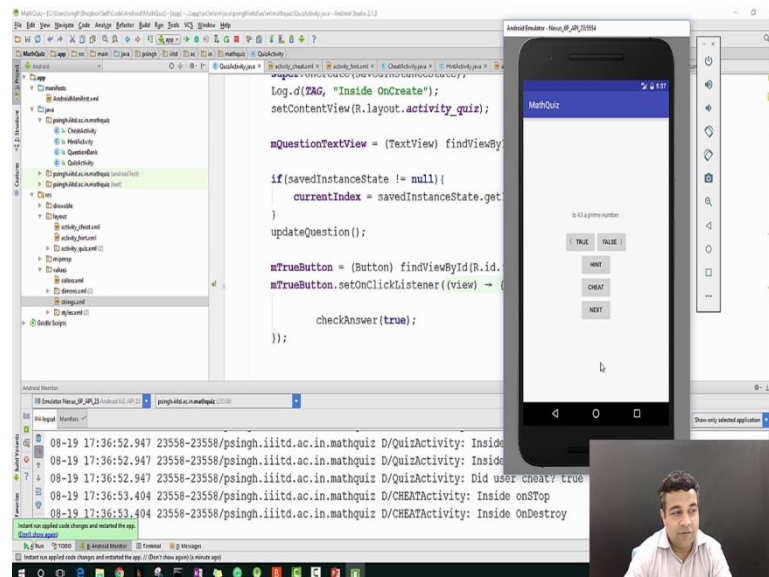
Earlier we had set this code inside the result and we can now remove this message, we are already setting the mCheated variable, so this time we will just say that in the checkAnswer it will say if mCheated then messageResId equals to the cheat toast else, else the remaining code and then I make the toast. We hope that we have done everything correctly and now if we run the application, unless we cheat, it should display the correct or incorrect toast or else it should display us the cheat toast, let us run our application. We are running our application and we will see how it behaves now.

(Refer Slide Time: 31:33)



It is 19 and I press True without cheating and I get the answer, go to next, I cheat, I get the cheated answer, I go back I press it and it says “You have Cheated”. Do you think we have done everything correct? Think about it. Ok, let us now go to the next question. This time we will not cheat and try to answer it.

(Refer Slide Time: 32:09)




Oops. We are still seeing the wrong message. Which is that You have cheated, while we have not cheated now. So yes, we have left one small detail in our program, let us fix it. What all we need to do is, that when we press next button, we reset the mCheated to false. Now let us run our program again. So, we have not cheated, it is correct, we cheat, come back, it is correct. Go to next, we do not cheat, it is again correct. Now one more thing is left to test, we go to next, we cheat, but we do not press the Show Cheat button, we come back and we give the answer and it is correct. So now our application is working correctly, we have added one activity in our lectures, I have already added one more activity, so this is an application which has two activities besides the main activity.

(Refer Slide Time: 33:59)

## Intents

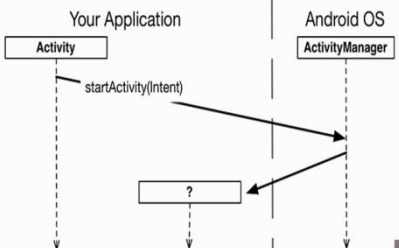
- A message object that a component can use to communicate with the OS
  - Component
    - Activity
    - Background receivers
    - Service
    - Content provider
- Intents are multi-purpose communication tools
  - The Intent class provides different constructors which you may use depending on your need.




So in this lecture, through this program you have learned how to use Intents to communicate between two activities. Now, let us go back to our basics and run through this presentation to cement what we have learned. You have already learnt all the concepts, so let me revise Intent is a message object that a component can use to communicate with the OS, which may have components activity, background receivers, services and content providers and Intents are used mainly as a multi-purpose communication tool. Like we saw, we use an Intent to pass the data to the child activity and then we use the Intent to pass the result from the child activity to the parent activity.

(Refer Slide Time: 34:35)

## Intent and Activity

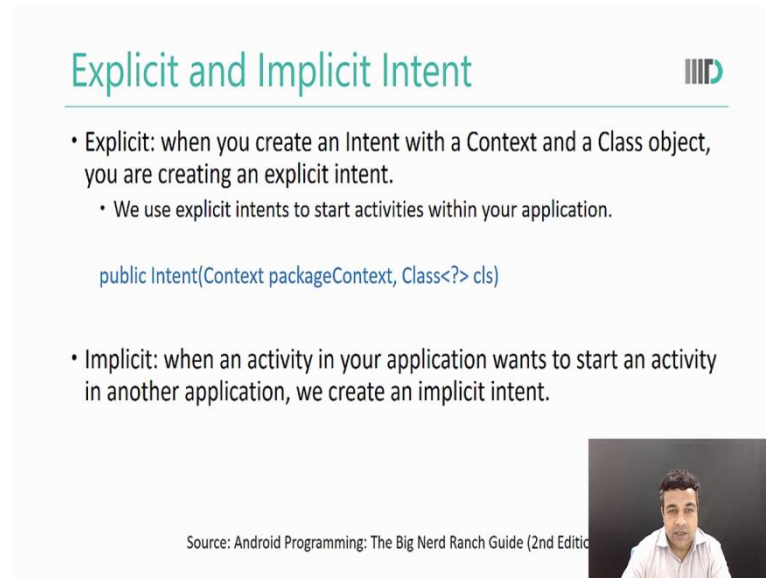


Source: Android Programming: The Big Nerd Ranch Guide (2nd Edition)



This is how Intent and activity work roughly your application Android OS, your application calls a start activity, it goes to the Android OS activity manager, which then starts another activity. For our particular case, something like this was happening, we were doing Quiz activity, we were calling start activity, we were passing an Intent of the component Cheat activity, Android OS manager was then starting our Cheat activity.

(Refer Slide Time: 35:08)



The slide is titled "Explicit and Implicit Intent" in green text. It features a blue logo in the top right corner. The content includes two bullet points: "Explicit: when you create an Intent with a Context and a Class object, you are creating an explicit intent." and "Implicit: when an activity in your application wants to start an activity in another application, we create an implicit intent." A code snippet is shown: `public Intent(Context packageContext, Class<?> cls)`. At the bottom left, the source is cited as "Source: Android Programming: The Big Nerd Ranch Guide (2nd Edition)". A small video inset of a man is in the bottom right corner.

## Explicit and Implicit Intent

- Explicit: when you create an Intent with a Context and a Class object, you are creating an explicit intent.
  - We use explicit intents to start activities within your application.

```
public Intent(Context packageContext, Class<?> cls)
```

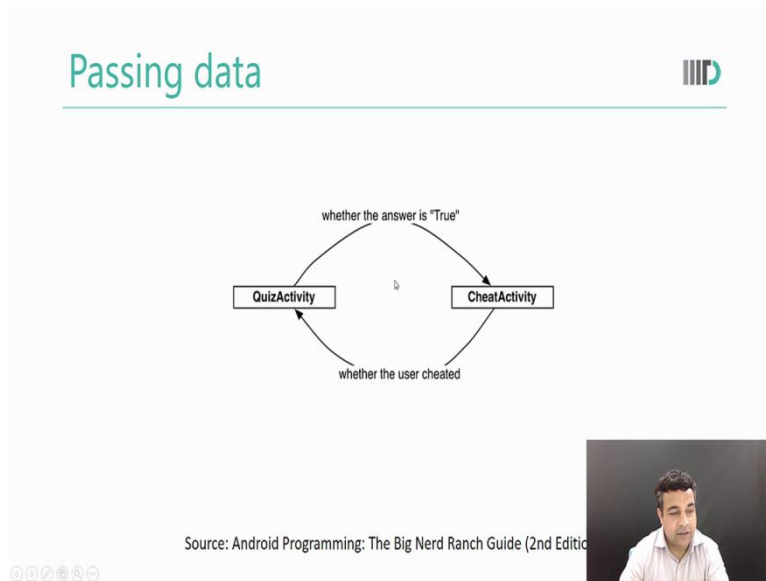
- Implicit: when an activity in your application wants to start an activity in another application, we create an implicit intent.

Source: Android Programming: The Big Nerd Ranch Guide (2nd Edition)

Now, there are two types of Intents, one is an Explicit Intent and one is an Implicit Intent. An Explicit Intent is, when you create Intent with a context and a class object like the way that we created in our program. We gave a context which was most likely Quiz activity.this and then we gave a class which was the class file of the new activity that we wanted to start. There are also Implicit Intents which are used when an activity in your application wants to start an activity in another application. So far we have not covered this part, so we will leave the discussion for our future lectures, whatever we have been using was Explicit.

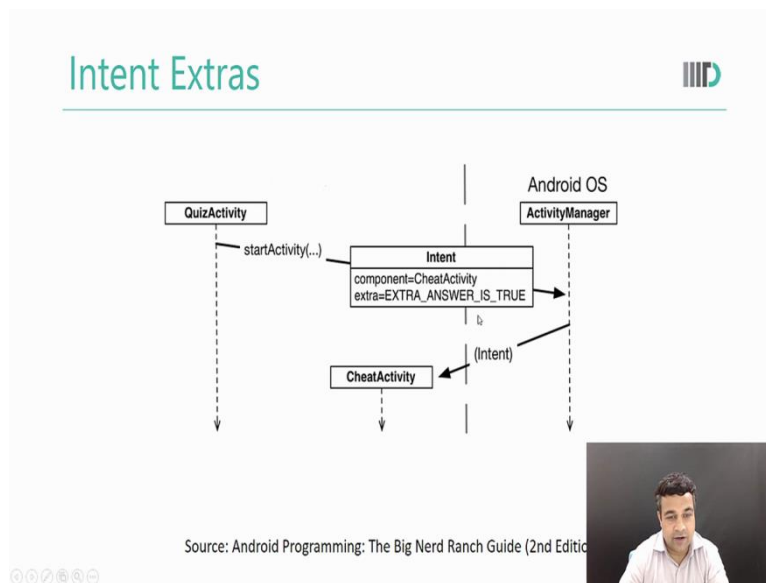


(Refer Slide Time: 35:58)



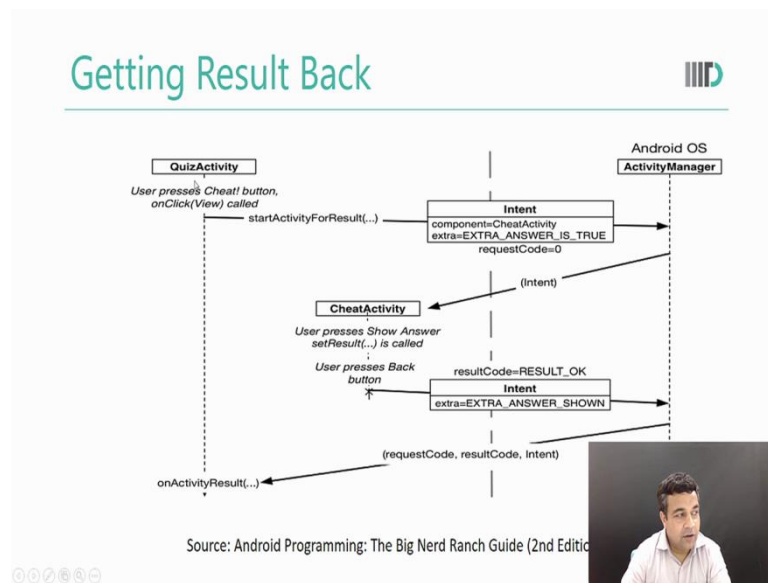
So this is what normally happens when you pass data between two Intents.

(Refer Slide Time: 36:07)



Now, we also used Intent extras for passing data from one activity to another activity. That is, when we do beyond, just starting another activity. As you saw that the extras were key value pairs and we used them using methods called GetExtra and PutExtra.

(Refer Slide Time: 36:28)



So yes, initially here is a good flow chart. Quiz activity starts activity for result pass an Intent with extra to the Android OS manager, which uses the Intent to start a new activity, the new activity then creates another Intent and uses an extra calls Android OS, which then calls this to pass the result back to the Quiz activity, the parent activity and parent activity can access this result by calling a method onActivityResult, this is all for your lecture. In these 2 lectures we have learnt how to create multiple activities in an application and how do we use Intents to communicate between these two activities.

Thank you.