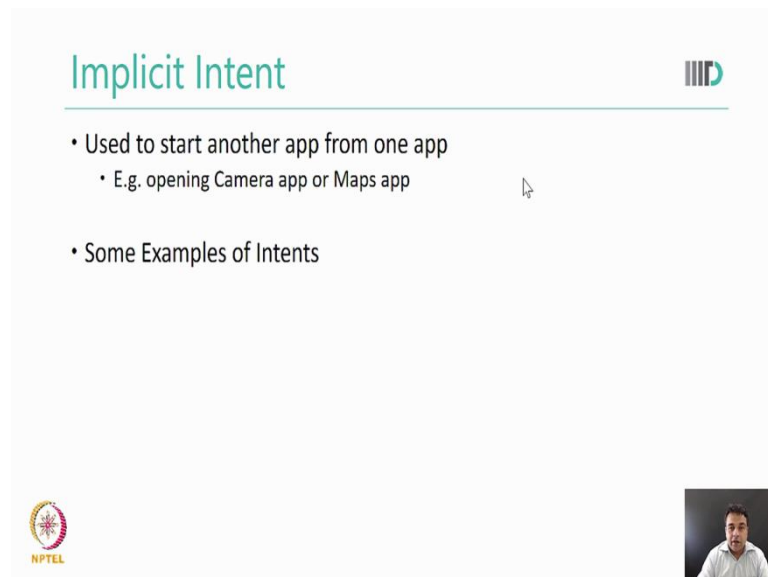


Mobile Computing
Professor Pushendra Singh
Indraprastha Institute of Information Technology Delhi
Lecture 27
Implicit Intent

Hello, welcome everyone today we will learn about implicit intents, we have learned about intents earlier and those intents were known as explicit intents. We use those intents to invoke another activity in the same application. So we had developed a math quiz application in which we had 3 activities and from the main activity we use to invoke either of the other 2 that is hint or the cheat activity. Today we are going to see that how one application can start activity in another application. All the fundamentals are same just some syntax is different so let us get started with it.

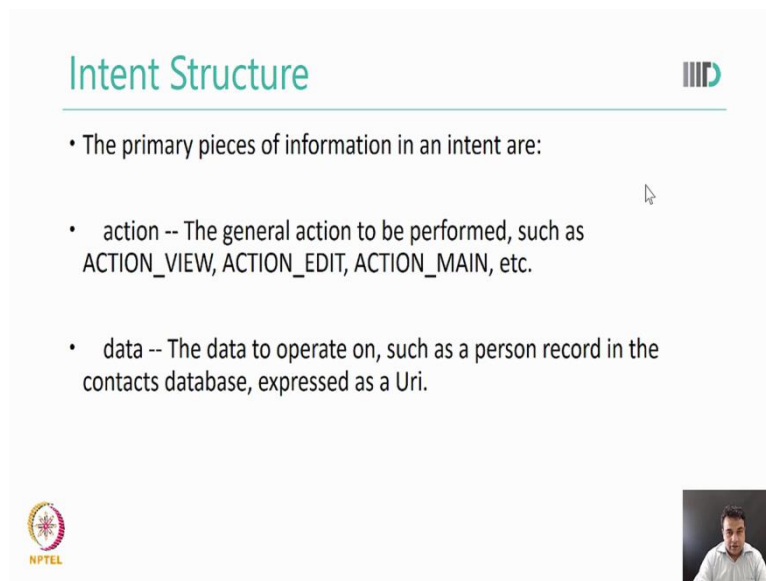
(Refer Slide Time: 1:09)



The slide is titled "Implicit Intent" in a teal font. In the top right corner, there is a logo consisting of three vertical bars of varying heights followed by the letter "D". Below the title, there are two bullet points: the first is "Used to start another app from one app" with a sub-bullet "E.g. opening Camera app or Maps app"; the second is "Some Examples of Intents". In the bottom left corner, there is the NPTEL logo, and in the bottom right corner, there is a small video thumbnail showing a man in a white shirt.

So implicit intents as you see are used to start another app from one app for example, you may have seen that when you are sometimes when you are using Whatsapp you can attach a photo to by taking a picture from your camera and Whatsapp starts your camera app you take a photo it gets added into your Whatsapp which you can then send. This is an example of one app starting activity in another app, today you will learn how to do that yourself. So before we move further let us try to understand what is the intent structure or what does an intent contain.

(Refer Slide Time: 1:56)



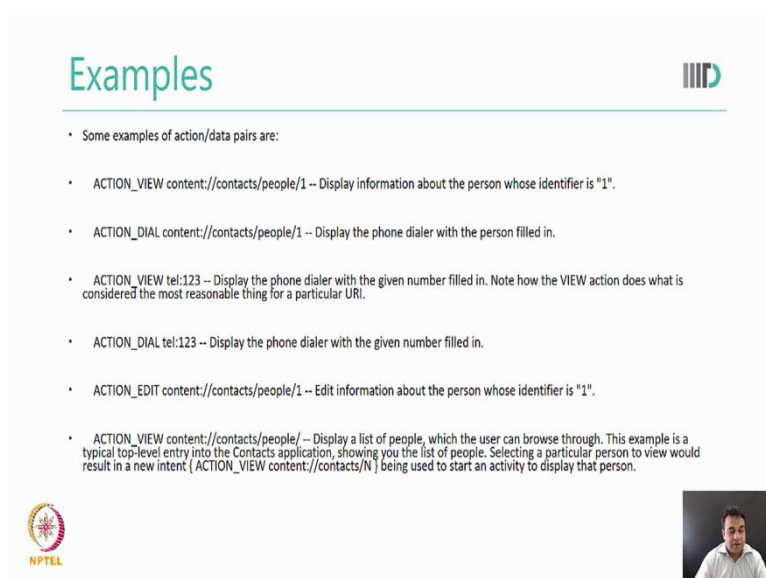
Intent Structure

- The primary pieces of information in an intent are:
 - action -- The general action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.
 - data -- The data to operate on, such as a person record in the contacts database, expressed as a Uri.

NPTEL logo and speaker photo are present at the bottom of the slide.

So intent contains two things. Number one is the action that the intent wants to do and number 2 the data that is the data on which that supposed action needs to be done. So that action could be a simple view action, or an edit action, or any other action and similarly there would be appropriate data for (()) (02:19). Here are some examples of the action and data pairs, let us go through them one by one.

(Refer Slide Time: 2:25)



Examples

- Some examples of action/data pairs are:
 - ACTION_VIEW content://contacts/people/1 -- Display information about the person whose identifier is "1".
 - ACTION_DIAL content://contacts/people/1 -- Display the phone dialer with the person filled in.
 - ACTION_VIEW tel:123 -- Display the phone dialer with the given number filled in. Note how the VIEW action does what is considered the most reasonable thing for a particular URI.
 - ACTION_DIAL tel:123 -- Display the phone dialer with the given number filled in.
 - ACTION_EDIT content://contacts/people/1 -- Edit information about the person whose identifier is "1".
 - ACTION_VIEW content://contacts/people/ -- Display a list of people, which the user can browse through. This example is a typical top-level entry into the Contacts application, showing you the list of people. Selecting a particular person to view would result in a new intent (ACTION_VIEW content://contacts/N) being used to start an activity to display that person.

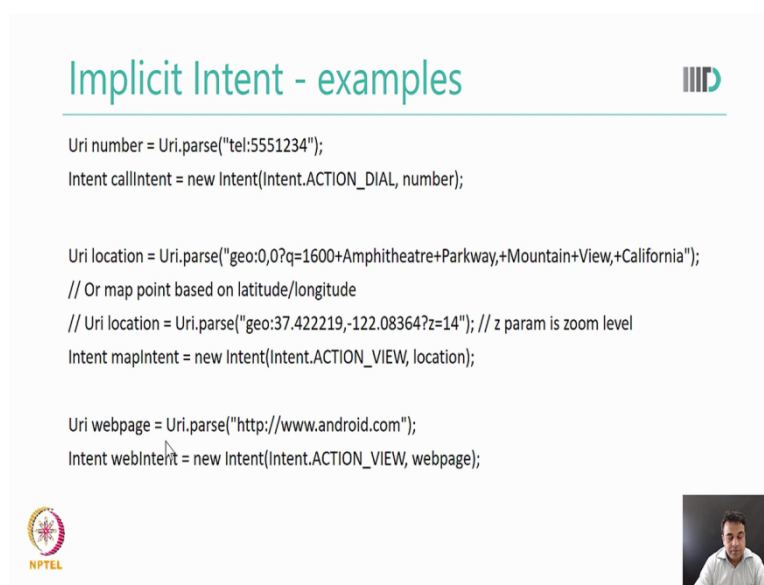
NPTEL logo and speaker photo are present at the bottom of the slide.

So the first example is of an action type view and there is a data which looks like as the data of one of the context in the users device. And this actually is what it looks like, so basically this intent with this act data will display information about the person whose identifier is 1. Similarly, the second example is that not only just displaying the information this time it will

be the phone dialer activity that will get invoked and with the number filled in, with the personal details filled in. And then the third is another example of view action in which we start with a telephone number, so it will displays the phone dialer with the given number filled in. Now you may notice that here we also had a view, here we also had a view so the view action does what is considered the most reasonable thing to do for a particular URI.

When it was contact information it was showing displaying information about the person, when it was a telephonic number it was it was displaying the phone dialer. And the forth example is similar to the second example except here instead of getting data from the contact information we are getting data directly from here. And then in the last example we are trying to edit the information of the contact that is given here. And then another example we want to display a list of people because you we were not giving any specific identifier here as such. So it will display list of people which the user can browse through. So these are some good 1, 2, 3, 4, 5, 6 examples which show you an action and data pairs as we use them in intent.

(Refer Slide Time: 4:48)



The slide is titled "Implicit Intent - examples" and features the NPTEL logo in the top right corner. It contains three code snippets demonstrating how to create intents with URIs:

```
Uri number = Uri.parse("tel:5551234");
Intent callIntent = new Intent(Intent.ACTION_DIAL, number);

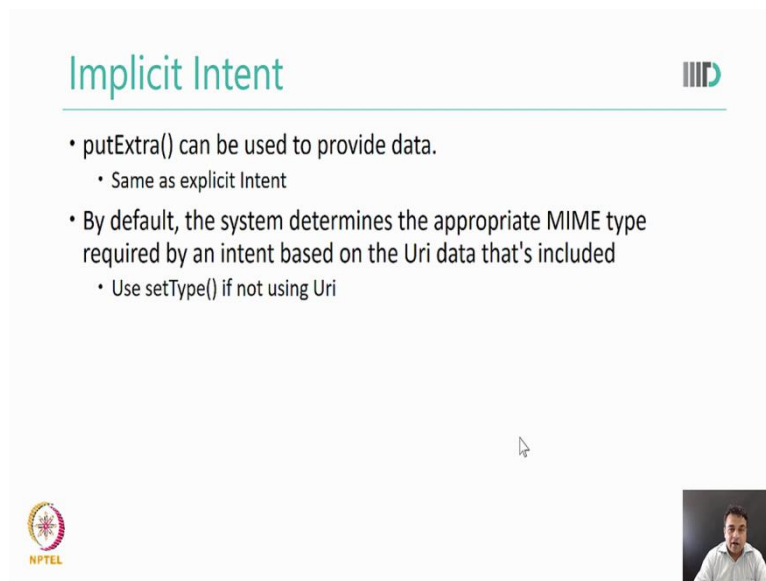
Uri location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California");
// Or map point based on latitude/longitude
// Uri location = Uri.parse("geo:37.422219,-122.08364?z=14"); // z param is zoom level
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);

Uri webpage = Uri.parse("http://www.android.com");
Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);
```

The slide also includes the NPTEL logo in the bottom left and a small portrait of a man in the bottom right.

Now let us move forward, let us look at some of the examples. So in our first example what we are trying to do is we are trying to dial a number which is this number. So we create an Uri object and then we create our intent with the required action and the data. In the second example, we want to show this geo location to the user, so we create another Uri object and then we create the intent which shows an action and the data. Earlier the action was the dial and now the action is the view, then in third example we want to show a web page so we create a Uri object and then we start the intent then we create the intent with the right action which is view and the data.


(Refer Slide Time: 5:54)



Implicit Intent

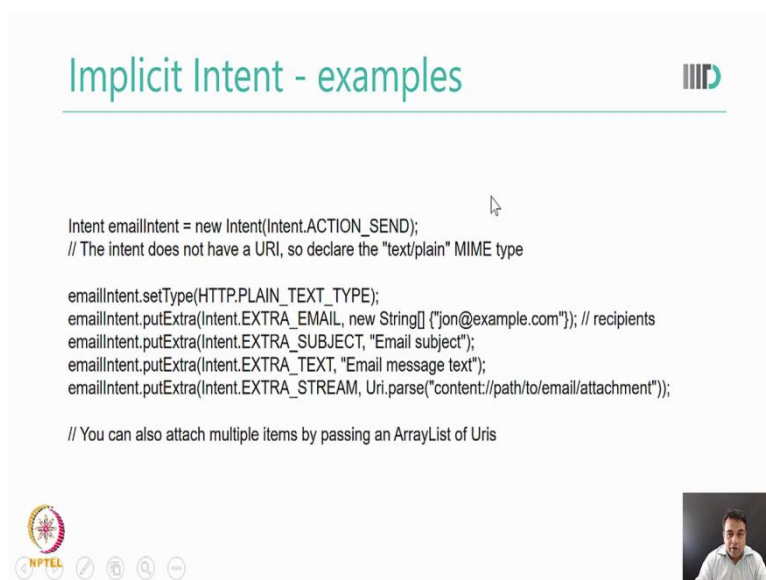
- putExtra() can be used to provide data.
 - Same as explicit Intent
- By default, the system determines the appropriate MIME type required by an intent based on the Uri data that's included
 - Use setType() if not using Uri

NPTEL



Just like you did in the explicit intent that whenever you needed to send extra data u use putExtra() and on the other side we create the data using getExtra. You can do this similar thing in case of implicit intents as well. Now in the previous example you saw that we were using uri. The advantage of using Uri is that if we use the Uri the system will determine the appropriate MIME type by default, ok. However, if you are not using the uri then you can use setType, you will very shortly see another example of this sort.

(Refer Slide Time: 6:37)




Implicit Intent - examples

```
Intent emailIntent = new Intent(Intent.ACTION_SEND);
// The intent does not have a URI, so declare the "text/plain" MIME type

emailIntent.setType(HTTP.PLAIN_TEXT_TYPE);
emailIntent.putExtra(Intent.EXTRA_EMAIL, new String[] {"jon@example.com"}); // recipients
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Email subject");
emailIntent.putExtra(Intent.EXTRA_TEXT, "Email message text");
emailIntent.putExtra(Intent.EXTRA_STREAM, Uri.parse("content://path/to/email/attachment"));

// You can also attach multiple items by passing an ArrayList of Uris
```

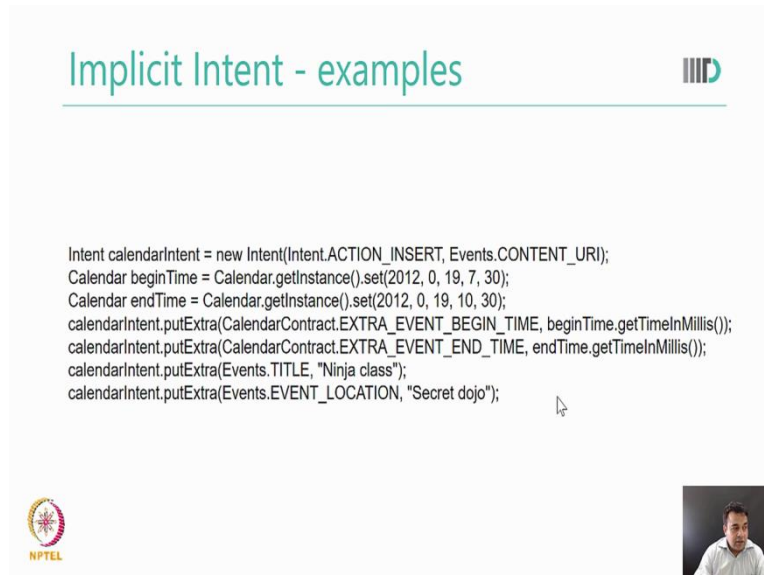
NPTEL



So yes here is another example, this example is of a type in which I look like to send an email. So I create intent, where my action is a send action and then I want to set some data which will be required by it. So number one I set the data type, here my data type is

HTTP.PLAIN_TEXT_TYPE and then I use putExtras() to had additional data so that my email can be send. So I use putExtra() to get the address of the recipient, to get the subject of the email, to set the text of the email and then also to attach something to the email, let us look at another example.

(Refer Slide Time: 7:29)



The slide is titled "Implicit Intent - examples" and features a small logo in the top right corner. It contains a block of Java code for creating an implicit intent to insert a calendar event. The code sets the action to Intent.ACTION_INSERT and the type to Events.CONTENT_URI. It then sets the begin and end times for the event using Calendar.getInstance().set(). Finally, it uses putExtra() to add event details: start and end times (using CalendarContract.EXTRA_EVENT_BEGIN_TIME and EXTRA_EVENT_END_TIME), the title "Ninja class", and the location "Secret dojo".

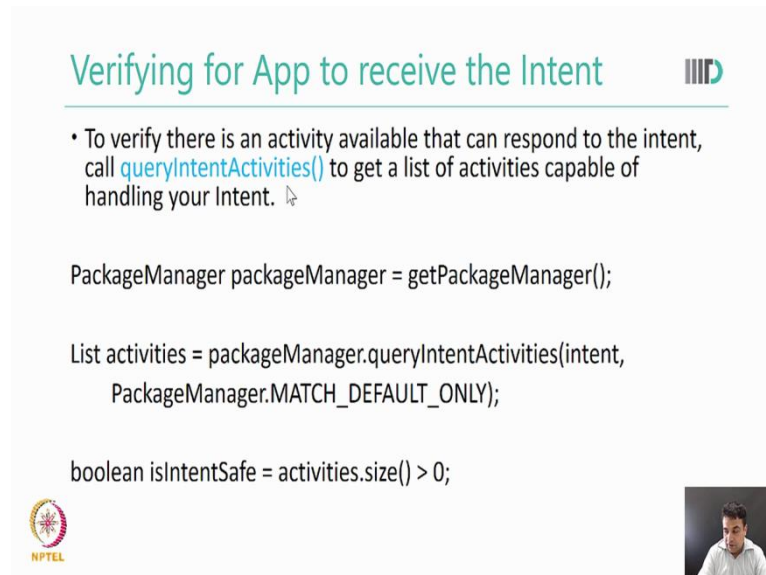
```
Intent calendarIntent = new Intent(Intent.ACTION_INSERT, Events.CONTENT_URI);
Calendar beginTime = Calendar.getInstance().set(2012, 0, 19, 7, 30);
Calendar endTime = Calendar.getInstance().set(2012, 0, 19, 10, 30);
calendarIntent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, beginTime.getTimeInMillis());
calendarIntent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME, endTime.getTimeInMillis());
calendarIntent.putExtra(Events.TITLE, "Ninja class");
calendarIntent.putExtra(Events.EVENT_LOCATION, "Secret dojo");
```

In this example I am trying to set up a calendar event. So I create an intent here my intent action is insert action and then I want to pass it some data which it will use. So let us see that what kind of data we want to pass. So we would like to pass the begin time and the end time of the event. So we create a begin time, end time, objects and we added those that information using putExtra and then we would also like to add the title of the event and then we would also like to add the location where this event is taking place. So this is all needed to create a calendar event, many times you may have seen applications which display some information and then ask you would you like to add it to your calendar and at that time if you press yes your calendar application pops up with some information filled in and you just save it. Now you can see how the call works.

Now we just gave few very interesting examples. Let us consider a scenario in which let us say your phone does not have a camera or your phone does not have a browser. Now if it does not have a camera and another activity starts an intent, which ask the photo to be taken it has the risk of the app getting crash, similarly for the browser or any other action which your device or another device cannot support. As a developer you should write your code such that your application always works and is never shut down unexpectedly. So if you are developing our application such that it is starting another activity we should not just assume

that there would be some activity available at all the user devices that may be using your application.

(Refer Slide Time: 10:14)



The slide is titled "Verifying for App to receive the Intent" and features a list of instructions, code snippets, and a small video inset of a speaker. The instructions describe how to use `PackageManager.queryIntentActivities()` to check for available activities. The code shows the steps: getting the `PackageManager`, querying for activities, and checking if the list is non-empty.

```
PackageManager packageManager = getPackageManager();

List activities = packageManager.queryIntentActivities(intent,
    PackageManager.MATCH_DEFAULT_ONLY);

boolean isIntentSafe = activities.size() > 0;
```

So what we need to do is before we launch our activity or before we launch the target activity we should check that if such an activity exist or not that is there an activity which can handle my intent or not, so how do we do that. So we do that using a method call `queryIntentActivities()` this method returns you a list of activities capable of handling your intent and the use of this method is very easy. So for example if I want to use this method all I need to do is have these three lines, let us see what these lines are. So I start with a `PackageManager`, I get a reference to it and then I create a list activities and I call my method on the `PackageManager` reference. So `packageManager.queryIntentActivities` then I pass it the right kind of intents for which I am querying the activities.

The second parameter is not very important here, but you may want to view its description and the developer website. So here we are only saying that give us only the one which `MATCH_DEFAULT`. This call will fill the list of the activities that can handle this type of intent. Now all we are interested is, in knowing that is there even a single activity which can do that. So next I check by simply by the size and if the size is 0, I say yes there is at least one activity which can handle it. So if I want to display a web page I know there is at least one activity which can show the web page and that may be good enough for me.




(Refer Slide Time: 11:43)

Example

```
// Build the intent
Uri location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California");
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);

// Verify it resolves
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities = packageManager.queryIntentActivities(mapIntent, 0);
boolean isIntentSafe = activities.size() > 0;

// Start an activity if it's safe
if (isIntentSafe) {
    startActivity(mapIntent);
}
```



Here is the more complete example of the same thing I would like to show a geo location so I create a Uri, I create an intent with ACTION_VIEW and the location. Now I need some activities on the user device which can show this location. So instead of just starting my intent directly I will first do the check, I will verify that it was (()) (12:10). So I start with the packageManager and then I go into the list I call my method queryIntentActivities and I get the size of the activities into a Boolean. So my Boolean sets true if and only if the size is greater than 0. If the size is less than 0 or equivalent to 0, well it will never be less than 0 it will be equivalent to 0 the Boolean sets to pops.

Now when I want to start my activity by when I want to launch my intent or start my activity all I do is I first make a check. So this one line, two lines and this third line make sure that my intent will only try to invoke an activity if such an activity exists on the target device, this is really the right way of writing your code.

(Refer Slide Time: 13:21)



Show an App Chooser

- Letting user choose when there is more than one app to handle the intent e.g. mail/share
- To show the chooser, create an Intent using `createChooser()` and pass it to `startActivity()`.

```
Intent intent = new Intent(Intent.ACTION_SEND);
...

// Always use string resources for UI text.
// This says something like "Share this photo with"
String title = getResources().getString(R.string.chooser_title);
// Create intent to show chooser
Intent chooser = Intent.createChooser(intent, title);

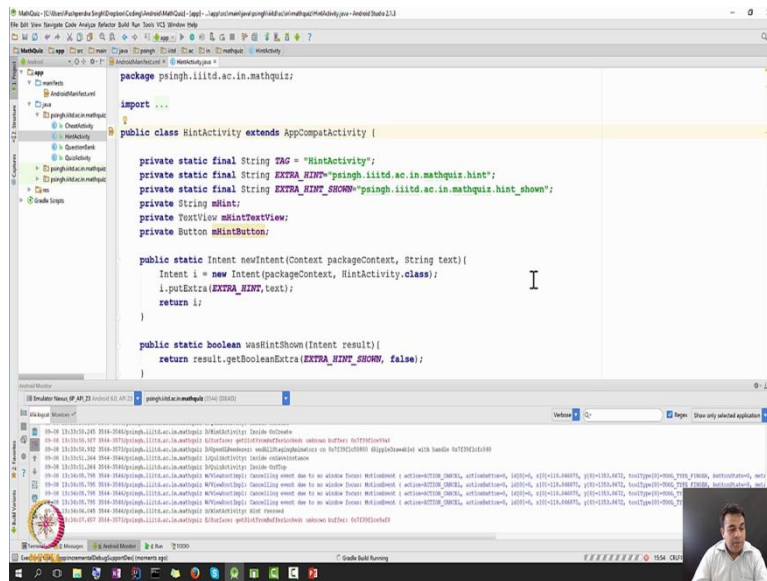
// Verify the intent will resolve to at least one activity
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(chooser);
}
```



Now at times you may have also shown you may have also seen android giving you the choice of application to choose from for example, if there is an information need to be displayed android may want to show you the different browsers that you have to stopped, so how does android do it? So again we have a method called `createChooser()` which does it. How do we use the `createChooser()` again the use of `createChooser()` is very easy for any given intent for any given intent I start it by calling `intent.createChooser`. So `createChooser()` I am calling on the intent and I pass down my intent and my title.

Now I again do some verification by saying that if there is a activity and but this time because I am using chooser here I was not using anything I was just starting activity because I only want it to start with any activity, here I want the user to choose this will display a user a choice of the activities that can handle my intent and user can choose a suitable activity then that activity will be lost.

(Refer Slide Time: 15:13)



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

import ...

public class HintActivity extends AppCompatActivity {

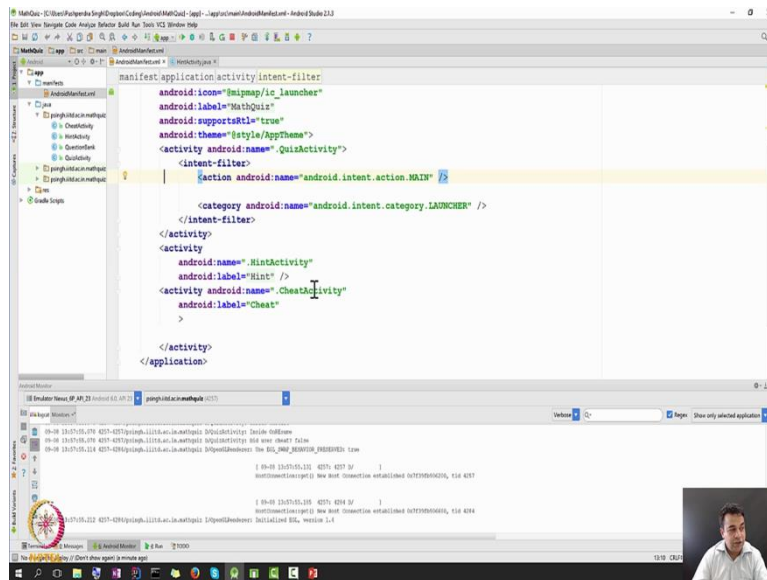
    private static final String TAG = "HintActivity";
    private static final String EXTRA_HINT = "psingh.iitd.ac.in.mathquiz.hint";
    private static final String EXTRA_SHOW = "psingh.iitd.ac.in.mathquiz.hint_shown";
    private String mHint;
    private TextView mHintTextView;
    private Button mHintButton;

    public static Intent newInstance(Context packageContext, String text){
        Intent i = new Intent(packageContext, HintActivity.class);
        i.putExtra(EXTRA_HINT, text);
        return i;
    }

    public static boolean wasHintShown(Intent result){
        return result.getBooleanExtra(EXTRA_SHOW, false);
    }
}
```

Now we come to the last topic allowing other apps to start your activity. For this let us go back to our code. I am again displaying you the code of our math quiz application, those of you who may have forgotten our math quiz application here is a simple revision our math quiz application had 3 activities and we were using explicit intent to go from one activity to another activity.

(Refer Slide Time: 15:45)



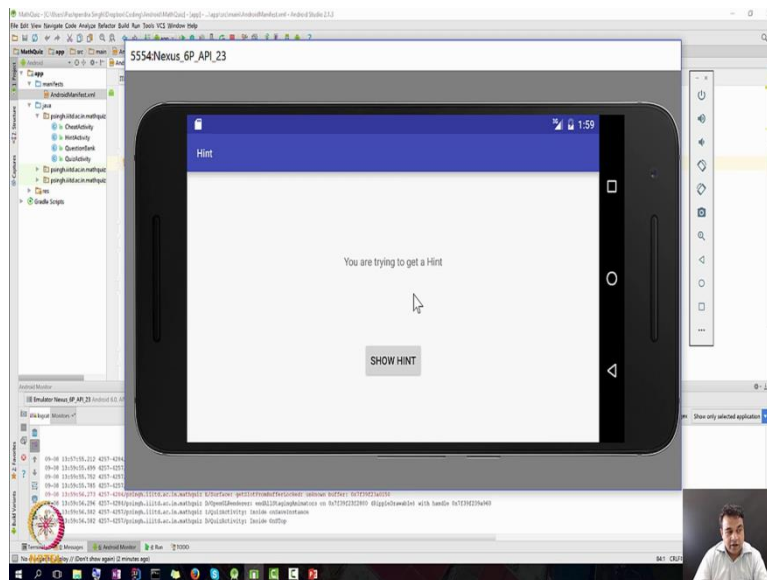
```
manifest:application,activity,intent-filter
android:icon="@mipmap/ic_launcher"
android:label="MathQuiz"
android:supportRtl="true"
android:theme="@style/AppTheme"
<activity android:name=".QuizActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".HintActivity"
    android:label="Hint" />
<activity android:name=".CheatActivity"
    android:label="Cheat"
/>
</activity>
</application>
```

Now I am going into the manifest file as you know that all android apps have a manifest file. So I am going to the manifest file and I am now looking at it more closely. So you will see that all my three activities are mentioned here the quiz activity, the hint activity and the cheat activity. However my main activity which is the quiz activity here also has another field

called intent filter. And earlier we have known that android programs do not have a main function because any application can start an activity, but they do have something called an action main, which determines the first activity that needs to be launched for a default app. So now you can see that if I want another application to start my activity I will have to set the correct intent filters in my manifest file.

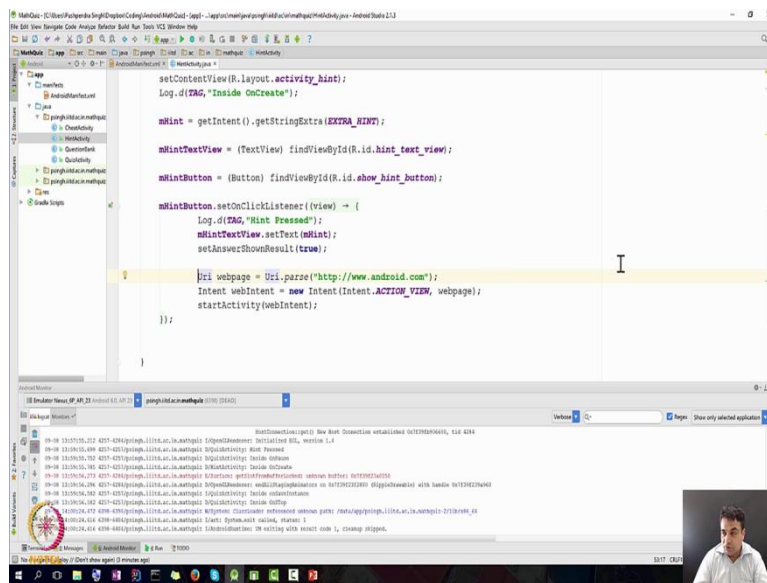
So with this our knowledge about intent is complete. Now we will very quickly see an example of using an implicit intent within our math quiz application only. The purpose of the example is to show you how the implicit intent cannot change any functionality of the math quiz application. But with that example done hopefully you would learn everything that you need to learn about the intent and you can then take self study to learn further.

(Refer Slide Time: 17:24)



So let us go back to our program and let us add some code to just display a webpage. So if you remember our math quiz application or simple we figure to hint we were starting another activity and if then if you were pressing this button it was showing the second. Now I want to make some changes so that when I press this button it not only does what it was doing earlier, but it also launches another activity in another application and in this case I would just like the new activity to display a webpage, ok. So let us stop this execution, go back and add some code.

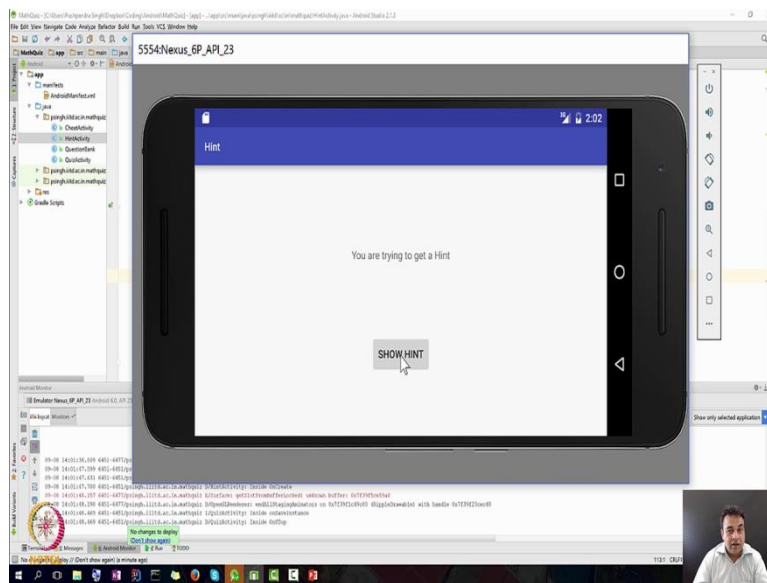
(Refer Slide Time: 18:06)



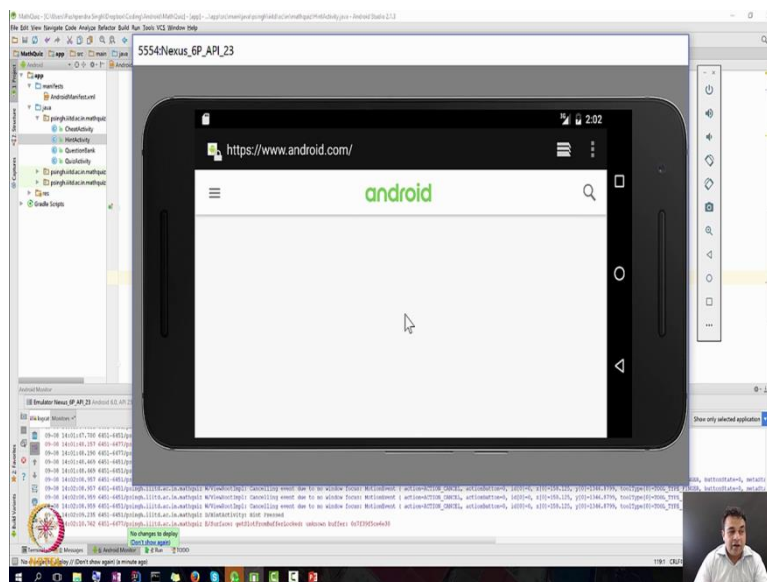
So I will go to the set on click listener on hint button and as you see that I have already added some code, this code is very easy we saw this code as an example. You can use the same code to see it an action. So I am starting with the Uri object, Uri webpage, Uri parse and then I am creating my intent which is of type ACTION_VIEW. So our action item is the view action only and then we are passing the data. Our data is complete so we need not to use putExtra() and then we start to then we call the start activity method. As you know that whenever we use intent we launch that intent in a way by calling the start activity method or start activity for result, but because here we do not want anything to come back to us we are only doing start activity.

Now let us start our application again and let us see what happens if we press the show hint button. So our application is starting, so our application has started press the hint button we go to the hint activity in a way we start an explicit intent. We move to another activity within our application and now we will be starting an implicit intent and we will moving to an activity in another application and not just our application, so let us press the button and see what happens.

(Refer Slide Time: 19:34)



(Refer Slide Time: 19:41)



So I press the button and here we go our new activity has been launched and the webpage has been displayed that is this code `www.android.com` has come into the place. So in this example you saw both explicit intent and implicit intent in action. Let us stop it, so now you know everything about using intents to move from one activity to another activity. Some of you may be tempted specially in case of the explicit intent to short circuit your program and instead of intent try to do the same activity using let us say static variables. This is not the right approach because as you saw that when we want to use the activities in another application we can we do not have access to the code. So intent is a very cleaner way of

starting other activities. Soon we will be learning about permissions and there we will know another benefit of using intent, thank you very much happy learning.