Mobile Application Development

PROFESSIONAL ELECTIVE-I

BE(CSE) V-Semester

S. Durga Devi ,CSE,CBIT

Unit-III

Topics to be covered

Prerequisites: components of android

- **Intents and Broadcasts**
  - Introducing Intents:
  - Using Intents to Launch Activities.
  - Using Intent to dial a number or to send SMS.
- **Broadcast Receivers**
  - Creating Intent Filters and Broadcast Receivers:
  - Using Intent Filters to Service Implicit Intents.
  - Finding and using Intents received within an Activity.
- **Notifications**
  - Creating and Displaying notifications,
- **Displaying Toasts.**

Android application Components

The following are the components required to build every android application

1. Activity
2. Services
3. Broadcast receivers
4. Content provider
5. Intents
6. Widgets
7. Notifications

1. Activity:

- Every single screen of android application with UI components with which user interacts with device or application is called activity.

2. Services:

- Running back ground processes continuously without interaction of user.

Ex- when you enable the WiFi in mobile and go out side it shows you what are networks available near to you automatically without your interaction.

- media player

3. Broad cast Receiver:

- Broad cast receivers are registered for system announcements.

Ex- when charger connected charger symbol activated on the screen.

- Head phones connected headphones symbol activated.

- click on power button and volume button increased or decreased.

4. Content provider:

- shares data between multiple applications. In android we cannot share the data between two applications to achieve this content provider is used.

Ex- Watapp reads contacts app data as contact app provides the content provider.

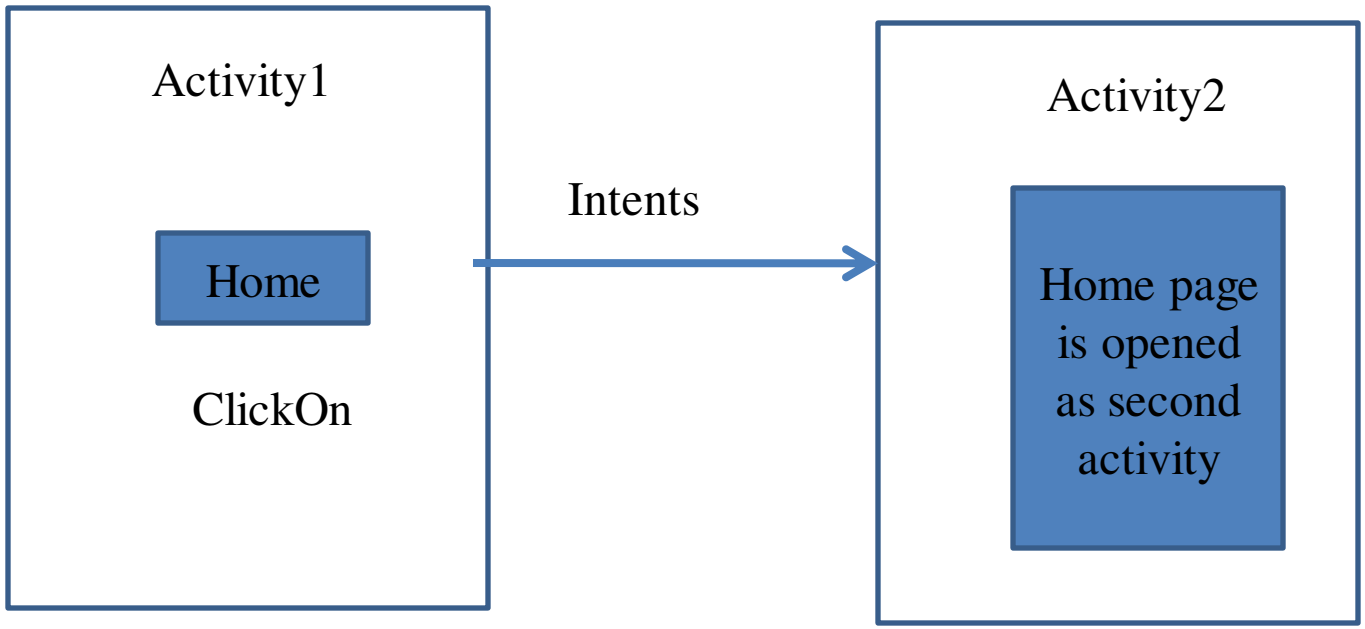
- Content provider provides security, call log , contacts and device media.

7. Intent:

- intent is used to communicate with one activity to another activity .

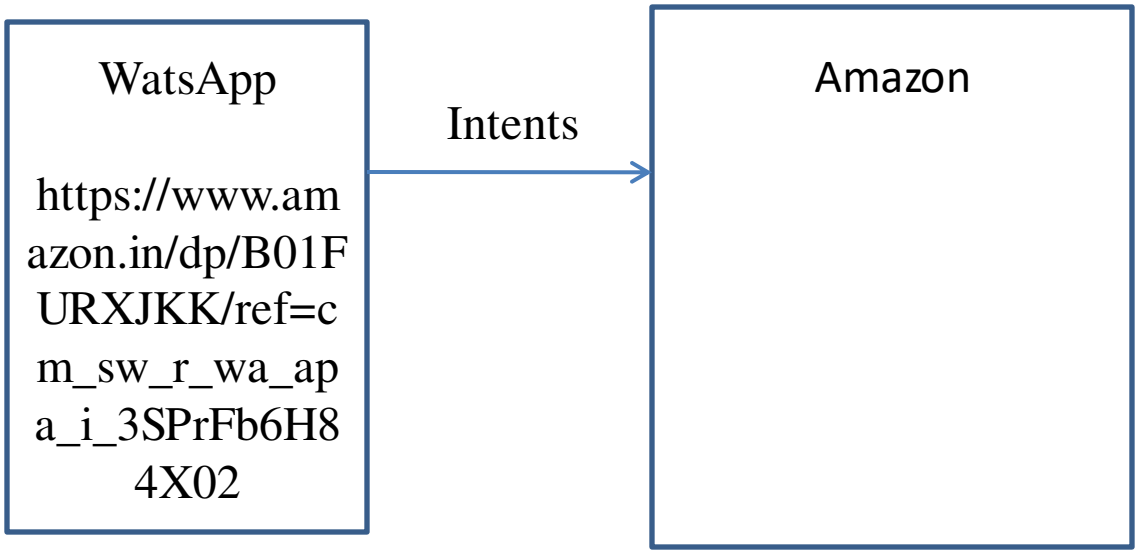
Two types of intents

1. Explicit Intent: communication between two activities inside the same application.



2. Implicit intent:

communication between activities in two different applications is called implicit intent.



8. Widgets:

Visual components that are added to the device home screen.

9. Notifications:

- When application not in use , application will send alert messages to the user
- when application sends notifications to devices display text messages on top of your screen alert you by playing sound, flashing lights , displaying icons.

Intents

- Intent is messaging object used to provide communication between android components such as activities, services, broadcast receivers and Content providers.
- Uses of Intents

1. **Launch an activity :** you can start new instance of an activity by passing intent to startActivitity() method. Intents describes the activity to start and carries necessary information.
2. **Starts services :** service is a component that runs the background operations. You can start a service by passing an intent to startService() method. Intents describes the service and pass the data if necessary.
3. **Delivers a broadcast :** starts the system announcements. You can start broadcast by passing an intent to sendBroadcast() or sendOrderedBroadcast().

Types of intents

- with respect to activity intents are divided into two types

1. Implicit intent
2. Explicit intent

1. Implicit intent: used to call built in activities such as camera, browser, caller, settings , google maps etc.

Syntax

```
// create Intent object
Intent i= new Inent(Intent.ACTION_VIEW);
i.setData(uri.parse(http://www.cbit.ac.in));
startActivity(i);
```

- ACTION-VIEW : specifies some information to be displayed to the user.

2. Explicit intent: calls user defined activities. Or navigate screens in your application.

Syntax

```
Intent i= new Intent(getApplicationContex(),.classfilename);
    startActivity(i);
```

➢Intents carries information that android system uses to information to determine wh component to start.

➢The primary information contained in an Intent is the following:

1. Component name : specifies name of component to start
2. Action: specifies generic action to be performed by particular activity such as view or pick

   ACTION_VIEW : Use this action in an intent with startActivity() when you have some information that an activity can show to the user, such as a photo to view in a gallery app, or an address to view in a map app.

   ACTION_SEND : Also known as the *share* intent, you should use this in an intent with startActivity() when you have some data that the user can share through another app, such as an email app.

3. Data : While creating an Intent, you can pass the data and the type of data on which the action is to be performed by the Android System with the help of Intents. Data type may be URI
4. Extras : You can add extra data to an Intent in the form of key-value pairs and this extra information can be passed from one Activity to the other. ***putExtra()*** is used to add some extra data to the Intents and this method accepts two parameters i.e. the key and its corresponding value.

Major methods in Intent class

setAction() // sets what action to be performed

setData() // set the data

setType()// specifies type of data is MIME(Multipurpose Internet Mail Extension)

putExtra()// add additional information

setComponent()// sets the type of the component

getExtra() //current activity retrieves the extra data sent by previous activity

Passing data between activities with intents

- You can use either intent data and intent extras to pass data between the activities.
- The intent *data* can hold only one piece of information. A URI representing the location of the data you want to operate on. That URI could be a web page URL (http://), a telephone number (tel://), a geographic location (geo://) or any other custom URI you define.

Use the intent data field:

- When you only have one piece of information you need to send to the started activity.
- When that information is a data location that can be represented by a URI.

Use the intent extras:

- If you want to pass more than one piece of information to the started activity.
- If any of the information you want to pass is not expressible by a URI.

Add data to intent

```
Intent i= new Intent(this,newactivity.clas);
// set web page
i.setData(Uri.parse(“www.google.com”));
//set sample file URI
i. setData(Uri.fromFile(new File("/sdcard/sample.jpg")));
```

Dial a number using intents

// this example uses Implicit intents to access Dial available in your phone.

- this example should be executed from your phone .

- To dial a number from your phone the following code is to be used

- // Dial any number

- // implicit Intent as dialor is built in app in android phone

- Intent intent=new Intent();
- intent.setAction(Intent.ACTION_DIAL);// access dialor in your phone
- startActivity(intent);

- Dial a particular number
- Intent intent=new Intent(Intent.ACTION_DIAL,Uri.parse("tel:8500022200"));
- startActicity(intent);

activity_main.xml

```
<LinearLayout>
<EditText
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_marginTop="50dp"
   android:id="@+id/et1"/>

<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Dial"
   android:layout_marginLeft="100dp"
   android:layout_marginTop="50dp"
   android:onClick="dial"/>
</LinearLayout>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    EditText editText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }//onCreate
    public void dial(View v)
    {
     Intent intent=new Intent(Intent.ACTION_DIAL);
     editText=(EditText)findViewById(R.id.et1);
     intent.setData(Uri.parse("tel:"+editText.getText().toString()));

     startActivity(intent);
    }//dial
}//MainActivity
```

```
// to access images from gallary in your phone
Publi void gallary(View  v)
{
Intent intent=new Intent();
intent.setAction(Intent.ACTION_GET_CONTENT);
//all the images i want to get
intent.setType("image/*");
startActivity(intent);
}
```

Call Third party applications from user applications

- when you want to call third party application like Whatapp and Skype you must know what is package name of the particular application.
- to get the package name of a particular application we have a app called package name Viewer app .
- Download packageNameViewer app from google play store
- PackageNameViewer app provides list of applications and their package name.

- // call WhatsApp from your application from your application

- **public void** whatsApp(View v)

```
{
 Intent intent=getPackageManager().getLaunchIntentForPackage("com.whatsapp");
 startActivity(intent);
}
```

- PackageManager : this class is used to retrieve various kind of information from the application packages which is being installed on your device.
- PackageManager is created by calling method getPackageManager() which is available in the Context.

- getLaunchIntentForPackage(**"com.whatsapp"**); this method is used to launch a third party application in activity and returns the Intent.

Send SMS from your Mobile

Sms can be send in two ways

1. Using intents
2. Using SmsManager class

Using Intents

```
Intent  i = new Intent(Intent.ACTION_VIEW);
i.putExtra("address", new String(txtMobile.getText().toString()));
i.putExtra("sms_body",txtMessage.getText().toString());
i.setType("vnd.android-dir/mms-sms");
```

Using SmsManager

```
SmsManager smgr = SmsManager.getDefault();
smgr.sendTextMessage(txtMobile.getText().toString(),null,txtMessage.getText().toString(),null,
null);
```

Manifest file

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

```
public void sendTextMessage (
String destinationAddress,
 String scAddress,
 String text,
 PendingIntent sentIntent,
PendingIntent deliveryIntent)
```

1. DestinationAddress: specifies to whom data should be sent. It cannot be null
2. scAddress: source address can be null
3. Text: what text message you want to send
4. sentIntent : can be null
5. deliveryIntent : can be null

PendingIntent : description of an intent and target action to perform with it.

PendingIntent given to another applications and you are granting it the right to perform operation you have specified as if the other application was your self

// sentIntent specifies whether message is sent or not . Network provider(BSNL,Airtel etc) will send sms. You are initializing your application to send sms through Network provider. Programmatically not possible whether message is sent or not so 4th and 5th parameters should be null.

Send an Email in Android

- Android provides built in Activity to send an Email.
- Use implicit intent to call built in activity

```
Intent intent=new Intent();
//set action to send an email
intent.setAction(Intent.ACTION_SEND);
//specify to address
intent.putExtra(Intent.EXTRA_EMAIL,new String[]{toaddr.getText().toString()});
//specify subject
intent.putExtra(Intent.EXTRA_SUBJECT,subText.getText().toString());
// specify message
intent.putExtra(Intent.EXTRA_TEXT,msgText.getText().toString());
// to send the attachments
intent.putExtra(Intent.EXTRA_STREAM,uri);
// type of the attachment
intent.setType("message/rfc822"):// supports MIME type data any multi media
// choose available applications to send an email
startActivity(intent.createChooser(intent,"select a mail"));
```

```
public void attachFile(View view)
{
    Intent intent=new Intent();
    // get all the list of files
    intent.setAction(Intent.ACTION_GET_CONTENT);
    // all the types of files to get
    intent.setType("*/*");
    startActivityForResult(intent,36);// get results from another activity

}

@Override
// method will be called when user selects file
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    uri=data.getData();

}
```

startActivityForResult() method is used when you want to get result from another activity. Suppose from camera app you captured a photo and receive that captured photo as a result in current activity.

➢**ACTION_SEND:** It is event in android.content.Intent is used to send the data from one activity to another activity and from current activity to outside the application.

➢ createChooser(Intent intent, " Chooser msg") : this method is used to display android sharesheet from which you can choose any one of the application to send the data.

➢ createChooser method will take Intent object as first parameter and second parameter must be title of the chooser dialog.

Intent Filters and Broadcast Recievers

- Broadcast receivers are used when you want to get system announcements
- System announcements are Headset plugin, Charger connected or disconnected, making / receiving a call, Screen off and screen on
- Device will recognize system announcements using Broadcast receivers.

- How to register broadcast receivers for system announcements

1. Create a class extends from android.content.BroadcastReceiver.
   - BroadcastReceiver is an abstract class which has abstract method onReceive()
2. Provide implementation for the onReceive () in BroadcastReceiver class.
3. In MainActivity class define following statements
   - create IntentFilter Object

```
IntentFilter filter= new IntentFilter();
-        filter.addAction(Intent.ACTION_HEAD_SET_PLUGIN);
              filter.addAction(Intent.ACTION_POWER_CONNECTED);
```

   - registerReceiver(BroadcastReceiver object, IntentFilter object);

```
class MyBroadcastReceiver extends BroadcastReceiver{
@Override
   public void onReceive(Context context, Intent intent) {
      if(intent.getAction()==(Intent.ACTION_HEADSET_PLUG))
         textView.setText("head set plugged in ");
      else if(intent.getAction()==(Intent.ACTION_POWER_CONNECTED))
         textView.setText("power connected ");
      else if(intent.getAction()==(Intent.ACTION_POWER_DISCONNECTED))
         textView.setText("power disconnectd");
      else if(intent.getAction()==(Intent.ACTION_SCREEN_ON))
         textView.setText("screen off ");
      else if(intent.getAction()==(Intent.ACTION_SCREEN_OFF))
         textView.setText("screen off ");
   }
}
```

Intent filters

- Intent filters tells to the android which activity can handle which action.
- In explicit intent , we can specify directly which intent directed to which component using following statement

```
Intent intent= new Intent(getApplicationContext(),secondActivity.class);
        startActivity(intent);
```

- in implicit intent, android figure out which activities handle which actions by using intent-filter tag in AndroidManifest.xml file

- Intent filter specifies what type of intent each component can receive.

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
        <data android:mimeType="image/*"/>
    </intent-filter>
</activity>
```

This tells Android the activity can handle ACTION_SEND.

The intent filter must include a category of DEFAULT or it won't be able to receive implicit intents.

These are the types of data the activity can handle.

Category : provides extra information about the activity such as whether it can starts by an web browser or main entry point of the app.

There are several standard values for the category

1.CATEGORY_LAUNCHER: should be displayed as top level launcher

2.CATEGORY_DEFAULT:- set if activity is default option

3.CATEGORY_BROWSABLE:- activity invoked through browser

4.CATEGORY_HOME:- This is the home activity, that is the first activity that is displayed when the device boots.

For more refer following

https://developer.android.com/reference/android/content/Intent#CATEGORY_LAUNCHER

Intent defined in activity class

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT, "Hello");
```

```
<activity android:name="SendActivity">
<intent-filter>
<action android:name="android.intent.action.SEND"/>
<category android:name="android.intent.category.DEFAULT"/>
<data android:mimeType="*/*"/>
</intent-filter>
</activity>
```

How android resolves intent filters

- when implicit intents are used , android compares the information given in intent with the intent filters specified in AndroidManifest.xml.

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

Android will only consider activities that specify an intent filter with an action of android.intent.action.SEND like this:

```
<intent-filter>
<action android:name="android.intent.action.SEND"/>
            ...
</intent-filter>
```

Similarly, if the intent MIME type is set to “text/plain” using

```
intent.setType("text/plain");
<intent-filter>
<data android:mimeType="text/plain"/>
...
</intent-filter>
```

Once Android has finished comparing the intent to the component intent filters, it sees how many matches it finds. If Android finds a single match, Android starts the component (in our case, the activity) and passes it the intent. If it finds multiple matches, it asks the user to pick one.

Intent intent = new Intent(Intent.ACTION_SEND); intent.setType("text/plain"); intent.putExtra(Intent.EXTRA_TEXT, "Hello");	**<activity android:name="SendActivity"> <intent-filter> <action android:name="android.intent.action.SEND"/> <category android:name="android.intent.category.DEFAULT"/> <data android:mimeType="*/*"/> </intent-filter> </activity>**
	<activity android:name="SendActivity"> <intent-filter> <action android:name="android.intent.action.SEND"/> <category android:name="android.intent.category.MAIN"/> <data android:mimeType="text/plain"/> </intent-filter> </activity>

Play an audio using android

activity_main.xml

```
<Button
   android:id="@+id/button1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:onClick="start"
   android:text="START"
   android:textStyle="bold“/>
```

```
<Button
   android:id="@+id/button2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:onClick="stop"
   android:text=“STOP"
   android:textStyle="bold“/>
```

```
<Button
   android:id="@+id/button1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:onClick=“pause"
   android:text=“PAUSE"
   android:textStyle="bold“/>
```

```
public class MainActivity extends AppCompatActivity {
    // create object for MediaPlayer
    MediaPlayer mediaPlayer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mediaPlayer=MediaPlayer.create(this,R.raw.telugu1);
    }
    public void start(View view)
    {
        mediaPlayer.start();

    }//start
    public void stop(View view)
    {
        mediaPlayer.stop();
    }
    public void pause(View view)
    {
        mediaPlayer.pause();
```

Services in android

Service: Service is one of the android application component. Runs back ground processes without interaction of the user. Examples wifi, music player.

- Service does not provide any UI , it should be managed by activity using intents.

How do you create services in android?

1. Create a sub class of type android.app.Service

- Service is an abstract class which as abstract method called as onBind()

-Methods in Service class

onCreate()

onStartCommand()

onDestroy()

-When you want to start a service which is not available onCreate() and onStartCommand() methods are invoked. If service is already available invokes onStartCommand() method.

- onDestroy() method invoked when you stop service.

2. Intent intent= new Intent();//create intent object

intent.setComponent(new ComponentName(this,class_name));// specify which component to navigate

startService(intent);// start service with intent object

3. stopService(intent);// stop the service with intent object

4. For every Service class we must configure AndroidManifest.xml file

<service android:name="pkg_name.class_name"/>

Activity_main.xml

activity_main.xml

```
<Button
   android:id="@+id/button1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:onClick="start"
   android:text="START"
   android:textStyle="bold“/>


<Button
   android:id="@+id/button2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:onClick="stop"
   android:text=“STOP"
   android:textStyle="bold“/>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }
    public void start(View view)
    {
          Intent i=new Intent();
          i.setComponent(new ComponentName(this,AudioService.class));
          startService(i);
    }//start
    public void stop(View view)
    {
        Intent i= new Intent();
        i.setComponent(new ComponentName(this,AudioService.class));
        stopService(i);
    }
}
```

AudioService.java

```
public class AudioService extends Service
{
    MediaPlayer mediaPlayer;
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        mediaPlayer = MediaPlayer.create(this, R.raw.telugu1);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        mediaPlayer.start();
        return super.onStartCommand(intent, flags, startId);

    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        mediaPlayer.stop();
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.audioplayex">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".AudioService"/>
    </application>

</manifest>
```

Notifications

- notifications are messages that are appear on top of the screen of your device. We can see those notifications by dragging finger from the top of the screen to down.
- Several built in services provided by an android such as alarm service, location service, download service, notification service etc.
- Notifications are displayed on the notification bar using notification service which is built in service provided by android.
- Android 8.0(Oreo), (API level 26), and above Oreo version onward all notifications must be assigned to a channel.
- Steps to create notifications

1. Create NotificationManager object to manage notification services

```
NotificationManager
nm=(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
```

2. Display notifications on notification bar , create NotificationCompat.Builder object

```
NotificationCompat.Builder n=new NotificationCompat.Builder(MainActivity.this,"my notification");
```

3. Set an icon for your notification

```
n.setSmallIcon("R.drawable.pic");
```

4. Make icon big when you drag notification in notification bar

```
Bitmap bitmap= BitmapFactory.decodeResource(getResources(),R.drawable.pic);
n.setLargeIcon(bitmap);
```

5. Set title

```
n.setContentTitle("testing notification");
```

6. Set text

```
n.setContentText("hi");
```

7. Set subtext

```
n.setSubText("goo morning");
```

8. When you select notification we need to all one activity is called pending intent(sub type of intent which is called after some time). If you call any activity immediately is called intent.

```
Intent intent= new Intent();
intent.setComponent(new
ComponentName(getApplicationContext(),MainActivity.class);
PendingIntent pi=PendingIntent. getActivity(getApplicationContext(),0,intent,0);
n.setContentIntent(pi);
```

9. `n.setAutoCancel(true);// remove from notification bar when I select this`

10. `// get the notification`

```
nm.notify(1,n.build());
```

- If your android sdk version is oreo or above we need to assign notifications to notification channel.
- Check whether sdk version is above or equal to oreo. If yes create NotificationChannel

All the notifications posted from app are targeting Build.VERSION_CODES.

```
if(Build.VERSION.SDK_INT>=Build.VERSION_CODES.O)
{
   NotificationChannel notificationChannel=new NotificationChannel("my notification","notification",
NotificationManager.IMPORTANCE_DEFAULT);
   NotificationManager
notificationManager=(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
   notificationManager.createNotificationChannel(notificationChannel);
}
```

To vibrate the phone when notification received use following code in manifest.xml file

<uses-permission android:name="android.permission.VIBRATE"/>

- Call method setVibrate() in MainActivity.java

Displaying Toast

- Toast is a class in android used to display message to the user for short period of time and current activity remains visible and interactive.
- Toast class is in package named as android.app.Toast.
- Example when you send a message or email you may see message like sending message or message delivery successfully.
- Toast is something like pop up message appears on the screen for some time and disappears after timeout.
- There are two constants

1. public static final int LENGTH_LONG: displays view for long period of time
2. public static final int **LENGTH_SHORT: displays view for short period of time.**

- **Methods in Toast class**

public static Toast makeText(Context context, CharSequence text, int duration)	makes the toast containing text and duration.
public void show()	displays toast.
public void setMargin (float horizontalMargin, float verticalMargin)	changes the horizontal and vertical margin difference.

Position toast on the screen

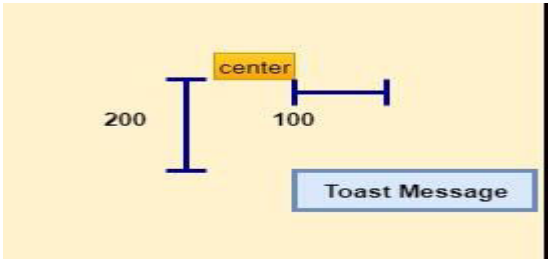
- default toast message appears centre bottom of the screen.
- Use setGravity() method to change position of the toast on the screen
- setGravity(int gravity, int x,int y);

1. Int gravity: uses predefined values like

   Gravity.TOP,BOTTOM,LEFT,RIGHT,CENTER,CENTER_HORIZONTAL,CENTER_VERTICAL

2. Int x: specifies horizontal distance
3. Int y: specifies vertical distance

toast.setGravity(Gravity.CENTER,100,200);



```
Toast t = Toast.makeText(getApplicationContext(),
        "This a positioned toast message",
        Toast.LENGTH_LONG);
t.setGravity(Gravity.CENTER, 100, 300);
t.show();
```

Create own layout

// create below layout save in res/layout folder with name custom_toast.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/toast_root_view"
   android:orientation="vertical" android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@android:color/background_dark"
   android:padding="16dp"
   >

   <TextView
      android:id="@+id/toast_header"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:textColor="@android:color/holo_green_dark"
      android:textSize="20dp" />

   <TextView
      android:id="@+id/toast_body"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:textColor="@android:color/holo_orange_dark" />

</LinearLayout>
```

http://tutorials.jenkov.com/android/toast.html#creating-a-toast

Create customized Toast

```
// converts XML into the VIEW object
LayoutInflater inflater = getLayoutInflater();

View toastLayout = inflater.inflate(R.layout.custom_toast,
        (ViewGroup) findViewById(R.id.toast_root_view));

TextView header = (TextView) toastLayout.findViewById(R.id.toast_header);
header.setText("Message for you:");

TextView body = (TextView) toastLayout.findViewById(R.id.toast_body);
body.setText("You have got mail!");

Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(toastLayout);
toast.show();
```

References

https://codelabs.developers.google.com/codelabs/android-training-create-an-activity/index.html?index=..%2F..android-training#1

https://www.tutlane.com/tutorial/android/android-phone-calls-with-examples

https://blog.mindorks.com/what-are-intents-in-android

https://data-flair.training/blogs/send-sms-in-android/

http://www.click2code.in/category/android-studio/

https://mkyong.com/android/how-to-send-sms-message-in-android/

https://guides.codepath.com/android/Sharing-Content-with-Intents

https://developer.android.com/reference/android/app/NotificationChannel

http://tutorials.jenkov.com/android/toast.html

http://tutorials.jenkov.com/android/toast.html#creating-a-toast