CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test 2 – August. 2022

Sub:	Mobile Applications							Sub Code:	20MCA263
Date:	1/9//2022	Duration:	90 min's	Max Marks:	50	Sem:	II	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

		MARKS	OBE	
			CO	RBT
	PART I			
1	Illustrate the different methods for getting location data? Explain	[10]	CO4	L1
	OR			
2	How do you publish android applications? List out steps briefly	[10]	CO5	L2
	PART II			
3	Explain the concept of sending SMS and write the procedure for sending an SMS through Android application with a code segment.	[10]	CO4	L2
	OR			
4a)	Write a program on Date and Time picker views.	[5]	CO4	L3
b)	Develop a standard calculator application to perform basic calculations like addition, subtraction, multiplication and division	[5]	CO2	L3

CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test 2 – August. 2022

Sub:	Mobile Applications							Sub Code:	20MCA263
Date:	1/9/2022	Duration:	90 min's	Max Marks:	50	Sem:	II	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

		MARKS	OBE	
			CO	RBT
	PART I			
1	Illustrate the different methods for getting location data? Explain	[10]	CO4	L1
	OR			
2	How do you publish android applications? List out steps briefly	[10]	CO5	L2
	PART II			
3	Explain the concept of sending SMS and write the procedure for sending an SMS through Android application with a code segment.	[10]	CO4	L2
	OR			
4a)	Write a program on Date and Time picker views.	[5]	CO4	L3
b)	Develop a standard calculator application to perform basic calculations like addition, subtraction, multiplication and division.	[5]	CO2	L3

	PART III			
5	Differentiate the basics of android UI design and location based services. **OR**	[10]	CO4	L3
6	Write a program in android for sending Email.	[10]	CO4	L2
	PART IV			
7	What are notifications? Explain various types of notifications.	[10]	CO4	L2
	OR			
8	Write a program to demonstrate autocompleteview with a list of places	[10]	CO5	L4
	PARTV			
9	Write in details how to display Google maps in your own android applications. **OR**	[10]	CO2	L2
10	Write a program to demonstrate the progress bar along with the status of completion	[10]	CO5	L4

	PART III			
5	Differentiate the basics of android UI design and location based services. **OR**	[10]	CO4	L3
6	Write a program in android for sending Email.	[10]	CO4	L4
	PART IV			
7	What are notifications? Explain various types of notifications.	[10]	CO4	L2
	OR			
8	Write a program to demonstrate autocompleteview with a list of places	[10]	CO5	L4
	PARTV			
9	Write in details how to display Google maps in your own android applications **OR**	[10]	CO2	L2
10	Write a program to demonstrate the progress bar along with the status of completion	[10]	CO5	L4

1.Illustrate the different methods for getting location data? Explain
Nowadays, mobile devices are commonly equipped with **GPS receivers.**
o Because of the many satellites orbiting the earth, you can use a GPS receiver to find your location easily. However, GPS requires a clear sky to work and hence does not always work indoors or where satellites can't penetrate (such as a tunnel through a mountain).
 Another effective way to locate your position is through ***cell tower triangulation***.
o When a mobile phone is switched on, it is constantly in contact with base stations surrounding it.
o By knowing the identity of cell towers, it is possible to translate this information into a physical location through the use of various databases containing the cell towers' identities and their exact geographical locations.
o The advantage of cell tower triangulation is that it works indoors, without the need to obtain information from satellites.
o It is not as precise as GPS because its accuracy depends on overlapping signal coverage, which varies quite a bit.
o Cell tower triangulation works best in densely populated areas where the cell towers are closely located.
 A third method of locating your position is to rely on **Wi-Fi triangulation.**
o Rather than connect to cell towers, the device connects to a Wi-Fi network and checks the service provider against databases to determine the location serviced by the provider
On the Android, the SDK provides the LocationManager class to help your device determine the user's physical location.
lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListener);
This method takes four parameters:
1. Provider -The name of the provider with which you register. In this case, you are using GPS to obtain your geographical location data.
2. minTime - The minimum time interval for notifications, in milliseconds.
3. minDistance - The minimum distance interval for notifications, in meters.
4. Listener - An object whose onLocationChanged() method will be called for each location update

2. How do you publish android applications? List out steps briefly

Once you have signed your APK fi les, you need a way to get them onto your users' devices. Three methods are here:
 Deploying manually using the adb.exe tool
 Hosting the application on a web server
 Publishing through the Android Market
Besides the above methods, you can install your applications on users' devices through emails, SD card, etc. As long as you can transfer the APK file onto the user's device, you

can install the application.
Using the adb.exe Tool: Once your Android application is signed, you can deploy it to emulators and devices using the adb.exe (Android Debug Bridge) tool (located in the platform-tools folder of the Android SDK). Using the command prompt in Windows, navigate to the "<Android_SDK>\platform-tools" folder. To install the application to an emulator/device (assuming the emulator is currently up and running or a device is currently connected), issue the following command:
adb install "C:\Users\Wei-Meng Lee\Desktop\LBS.apk"
(Note that, here, LBS is name of the project)
Besides using the adb.exe tool to install applications, you can also use it to remove an installed application. To do so, you can use the shell option to remove an application from

its installed folder:
adb shell rm /data/app/net.learn2develop.LBS.apk
Another way to deploy an application is to use the DDMS tool in Eclipse. With an emulator (or device) selected, use the File Explorer in DDMS to go to the /data/app folder and use the "Push a file onto the device" button to copy the APK file onto the device.
Using a Web Server: If you wish to host your application on your own, you can use a web server to do that. This is ideal if you have your own web hosting services and want to provide the application free of charge to your users or you can restrict access to certain groups of people. Following are the steps involved:
 Copy the signed LBS.apk fi le to c:\inetpub\wwwroot\.
<html>
<title>Where Am I application</title>
<body>
Download the Where Am I application here
</body>
</html>
 On your web server, you may need to register a new MIME type for the APK file. The MIME type for the .apk extension is application/vnd.android.packagearchive.
 From the Application settings menu, check the "Unknown sources" item. You will be prompted with a warning message. Click OK. Checking this item will allow the Emulator/device to install applications from other non-Market sources (such as from a web server).
 To install the LBS.apk application from the IIS web server running on your computer, launch the Browser application on the Android Emulator/device and navigate to the URL pointing to the APK file. To refer to the computer running the emulator, you should use the special IP address of 10.0.2.2.
 Alternatively, you can also use the IP address of the host computer. Clicking the "here" link will download the APK file onto your device. Drag the notification bar down to reveal the download status. To install the downloaded application, simply tap on it and it will show the permission(s) required by this application.
 Click the Install button to proceed with the installation. When the application is installed, you can launch it by clicking the Open button.
Besides using a web server, you can also e-mail your application to users as an attachment; when the users receive the e-mail they can download the attachment and install the application directly onto their device.
Publishing on Android Market: It is always better to host your application on Android market (Google Playstore). Steps involved in doing so, are explained hereunder:
 Creating a Developer Profile:
o Create a developer profile at http://market.android.com/publish/Home using a Google account.
o Pay one-time registration fees.
o Agree Android Market Developer Distribution Agreement
 Submitting Your Apps: If you intend to charge for your application, click the Setup Merchant Account link located at the bottom of the screen. Here you enter additional information such as bank account and tax ID. You will be asked to supply some

details for your application. Following are the compulsory details to be provided:
o The application in APK format
o At least two screenshots. You can use the DDMS perspective in Eclipse to capture screenshots of your application running on the Emulator or real device.
A high-resolution application icon. This size of this image must be 512×512 pixels.
o Provide the title of your application, its description and recent update details.

o Indicate whether your application employs copy protection, and specify a content rating.

When all these setup is done, click Publish to publish your application on the Android Market.

3. Explain the concept of sending SMS and write the procedure for sending an SMS through Android application with a code segment.

SMS messaging is one of the main *killer applications* on a mobile phone today — for some users as necessary as the phone itself. Any mobile phone you buy today should have at least SMS messaging capabilities, and nearly all users of any age know how to send and receive such messages. Android comes with a built-in SMS application that enables you to send and receive SMS messages. However, in some cases you might want to integrate SMS capabilities into your own android application. For example, you might want to write an application that automatically sends a SMS message at regular time intervals. For example, this would be useful if you wanted to track the location of your kids — simply give them an Android device that sends out an SMS message containing its geographical location every 30 minutes.

4.1.1 Sending SMS Messages Programmatically

To create an application that can send SMS, following are the steps to be followed:

Create a new android application.

Add the following statements in to the main.xml file:

```
<Button
android:id=”@+id/btnSendSMS”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Send SMS” />
```

In the AndroidManifest.xml file, add the following statements:

```
<uses-sdkandroid:minSdkVersion=”8” />
<uses-permission
android:name=”android.permission.SEND_SMS”>
</uses-permission>
```

Add the following statements to the MainActivity.java file:

```
import android.app.PendingIntent;
import android.content.Intent;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity
{
Button btnSendSMS;
/** Called when the activity is first created.
@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
btnSendSMS.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
```

4.a. Write a program on Date and Time picker views.

4.b. Develop a standard calculator application to perform basic calculations like addition, subtraction, multiplication and division.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/linearLayout1"
      android:layout_marginLeft="10pt"
      android:layout_marginRight="10pt"
      android:layout_marginTop="3pt">
      <EditText
         android:layout_weight="1"
         android:layout_height="wrap_content"
         android:layout_marginRight="5pt"
         android:id="@+id/etNum1"
         android:layout_width="match_parent"
         android:inputType="numberDecimal">
      </EditText>
      <EditText
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:layout_marginLeft="5pt"
         android:id="@+id/etNum2"
         android:layout_width="match_parent"
         android:inputType="numberDecimal">
      </EditText>
   </LinearLayout>
   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/linearLayout2"
      android:layout_marginTop="3pt"
      android:layout_marginLeft="5pt"
      android:layout_marginRight="5pt">
      <Button
         android:layout_height="wrap_content"
         android:layout_width="match_parent"
         android:layout_weight="1"
         android:text="+"
         android:textSize="8pt"
         android:id="@+id/btnAdd">
      </Button>
      <Button
         android:layout_height="wrap_content"
         android:layout_width="match_parent"
         android:layout_weight="1"
         android:text="-"
         android:textSize="8pt"
         android:id="@+id/btnSub">
      </Button>
      <Button
         android:layout_height="wrap_content"
         android:layout_width="match_parent"
```

```
            android:layout_weight="1"
            android:text="*"
            android:textSize="8pt"
            android:id="@+id/btnMult">
        </Button>
        <Button
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:layout_weight="1"
            android:text="/"
            android:textSize="8pt"
            android:id="@+id/btnDiv">
        </Button>
    </LinearLayout>
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:layout_marginLeft="5pt"
        android:layout_marginRight="5pt"
        android:textSize="12pt"
        android:layout_marginTop="3pt"
        android:id="@+id/tvResult"
        android:gravity="center_horizontal">
    </TextView>
</LinearLayout>
```

MainActivity.java

```
public class MainActivity extends Activity implements OnClickListener {

  EditTextetNum1;
  EditTextetNum2;

  Button btnAdd;
  Button btnSub;
  Button btnMult;
  Button btnDiv;

  TextViewtvResult;

  String oper = "";

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // find the elements
    etNum1 = (EditText) findViewById(R.id.etNum1);
    etNum2 = (EditText) findViewById(R.id.etNum2);

    btnAdd = (Button) findViewById(R.id.btnAdd);
    btnSub = (Button) findViewById(R.id.btnSub);
    btnMult = (Button) findViewById(R.id.btnMult);
```

```
  btnDiv = (Button) findViewById(R.id.btnDiv);

  tvResult = (TextView) findViewById(R.id.tvResult);

  // set a listener
  btnAdd.setOnClickListener(this);
  btnSub.setOnClickListener(this);
  btnMult.setOnClickListener(this);
  btnDiv.setOnClickListener(this);

}

@Override
public void onClick(View v) {
  // TODO Auto-generated method stub
  float num1 = 0;
  float num2 = 0;
  float result = 0;

  // check if the fields are empty
  if (TextUtils.isEmpty(etNum1.getText().toString())
      || TextUtils.isEmpty(etNum2.getText().toString())) {
    return;
  }

  // read EditText and fill variables with numbers
  num1 = Float.parseFloat(etNum1.getText().toString());
  num2 = Float.parseFloat(etNum2.getText().toString());

  // defines the button that has been clicked and performs the corresponding operation
  // write operation into oper, we will use it later for output
  switch (v.getId()) {
  case R.id.btnAdd:
    oper = "+";
    result = num1 + num2;
    break;
  case R.id.btnSub:
    oper = "-";
    result = num1 - num2;
    break;
  case R.id.btnMult:
    oper = "*";
    result = num1 * num2;
    break;
  case R.id.btnDiv:
    oper = "/";
    result = num1 / num2;
    break;
  default:
    break;
  }

  // form the output line
  tvResult.setText(num1 + " " + oper + " " + num2 + " = " + result);
}
```

}

5.Differentiate the basics of android UI design and location based services.

Android UI design	**Location based services**
Uses all view components	Uses only map view
Displays all components and retrieves data	Provides customized services and search
Data is retrieved by UI components	Data is retrieved as latitude and longitude
No need of additional jars	Needs map.jar
No need of any translation	Geodecoder is used

6.Write a program in android for sending Email.

One can set email through Android program. Following are the steps involved:

Create a new android application.

Add the following statements in to the main.xml file:

```
<Button
android:id=”@+id/btnSendEmail”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Send Email” />
```

Add the following statements in bold to the MainActivity.java file:

```
import android.content.Intent;
import android.net.Uri;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity
{
Button btnSendEmail;
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
btnSendEmail = (Button) findViewById(R.id.btnSendEmail);
btnSendEmail.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
String[] to = {“weimenglee@learn2develop.net”,
“weimenglee@gmail.com”};
String[] cc = {“course@learn2develop.net”};
sendEmail(to,cc,“Hello”, “Hello my friends!”);
}
});
}
//---sends an SMS message to another device---
private void sendEmail(String[] emailAddresses, String[]
carbonCopies, String subject, String message)
{
Intent emailIntent = new Intent(Intent.ACTION_SEND);
emailIntent.setData(Uri.parse(“mailto:”));
String[] to = emailAddresses;
String[] cc = carbonCopies;
emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
```

```
emailIntent.putExtra(Intent.EXTRA_CC, cc);
emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
emailIntent.putExtra(Intent.EXTRA_TEXT, message);
emailIntent.setType("message/rfc822");
startActivity(Intent.createChooser(emailIntent,"Email"));
}
}
```

7. What are notifications? Explain various types of notifications.

Notification is a user interface element that will display outside of any other app's normal UI to indicate that an event has occurred. Users can choose to view the notification while using other apps and respond to it when it's convenient for them.

Android notification will be displayed in the Notification area and to see the details regarding the notification, the user can expand it in by open the Notification drawer.

Following are the three types of android notifications,

1. Toast Notification – Shows message that fades away after a few seconds. (Background type also)
2. Status Notification – Shows notification message and displayed till user action. (Background type also)
3. Dialog Notification – Comes out of an active Activity.

(Background type) – is result of some background Service event that may not be related to current activity. That is, we can use this notification type in Service also, added to Activity.

1. Toast NotificationThis type of notification will be used when there is no need of user interaction on seeing this message. This message occupies a rectangular box which will fade in and fade out after some time. 

The size of the box depends on the message content.

For example, when user creates an event using calendar application it will notify the user as “Event Created” after the create action is completed. Refer the image.

Toast notification is best suited for one way information to the use where we don’t expect any response. Toast message does not stop or disturb the current activity, just the message is shown in parallel.

Example for Android Toast Notification

Toast notification can be created from an Activity or Service. Toast is the class to be used as below,

```
ContextappContext=getApplicationContext();
ToastmailMessage=Toast.makeText(appContext,“EmailReceived.”,Toast. LENGTH_LONG);
mailMessage.setGravity(Gravity.TOP,0,0);//optional
mailMessage.show();
```

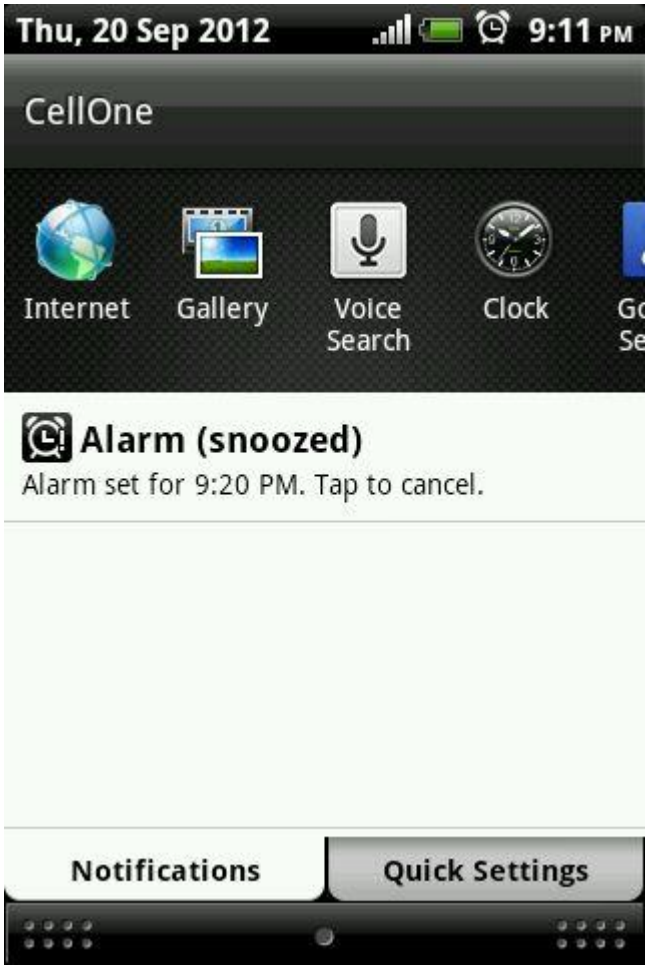
- duration – can be either LENGTH_SHORT or LENGTH_LONG
- setGravity – is used to position the message in screen. By default it shows at bottom centered. First parameter is Gravity a constant identifying location in container broadly like TOP | BOTTOM | LEFT … , second and third parameters are x, y-offset.

2. Status Notification

Status notification is used to display rich notification information especially from a (background) Service where user can interact. It will be shown as an icon with an alert in the status bar. When the user pulls down the status bar, the list of notification will be in the notification window.

For example when a SMS message is received a message icon is shown in the status bar. On pull down, the list of unread messages will be shown in the notification window.

Example shown in image: On snoozing the alarm, corresponding notification will be will be sent to the status bar with notification icon. A ticker message will be shown next to the icon for some time. In the image the clock icon represents the notification about the snooze event and the ticker message is shown next to the clock icon.



1. Create a simple notification with an icon alert. Alert can be a ticker text message or sound or vibration or flashlight.
2. Associate notification message with details shown on message expansion to activity/intent. Notification message can be a list and it is identified using a unique identifier. Existing messages can be updated too.
3. Register the notification message with notification manager. NotificationManager is a system service that manages all the notifications.

Example for Android Status Notification

```
//part 1 – notification icon alert

int icon =R.drawable.notification_icon;
```

```
// a ticker text message or sound or vibration or flashlight can be used for alert

CharSequence ticker =“Hi”;

longshowAt=System.currenttimeMillis();//immediately

Notificationnotification=newNotification(icon, ticker,showAt);

//part 2 – associate notification message with details shown on message expansion to activity/intent

CharSequencenotificationTitle="Notification:";

CharSequencenotificationMessage="SMS Received.";

Intentintent=newIntent(this,Activity.class);

PendingIntentobjPendingIntent=PendingIntent.getActivity(this,0, intent,0);

Contextctx=getApplicationContext();

notification.setLatestEventInfo(ctx,notificationTitle,notificationMessage,objPendingIntent);

//part 3 – register the notification message with notification manager

privatestaticfinalintnotificationIdentifier=101;//an unique number set by developer to identify a notification, using this notification can be updated/replaced

NotificationManagernotificationManager=(NotificationManager)

getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.notify(notificationIdentifier,objNotification);
```

Notification Alerts

Sound:

```
notification.defaults|=Notification.DEFAULT_SOUND;

//use the above default or set custom valuse as below

notification.sound=Uri.parse("file:///sdcard/notification/robo_da.mp3");
```

Vibration:

```
notification.defaults|=Notification.DEFAULT_VIBRATE;

//use the above default or set custom valuse as below
```

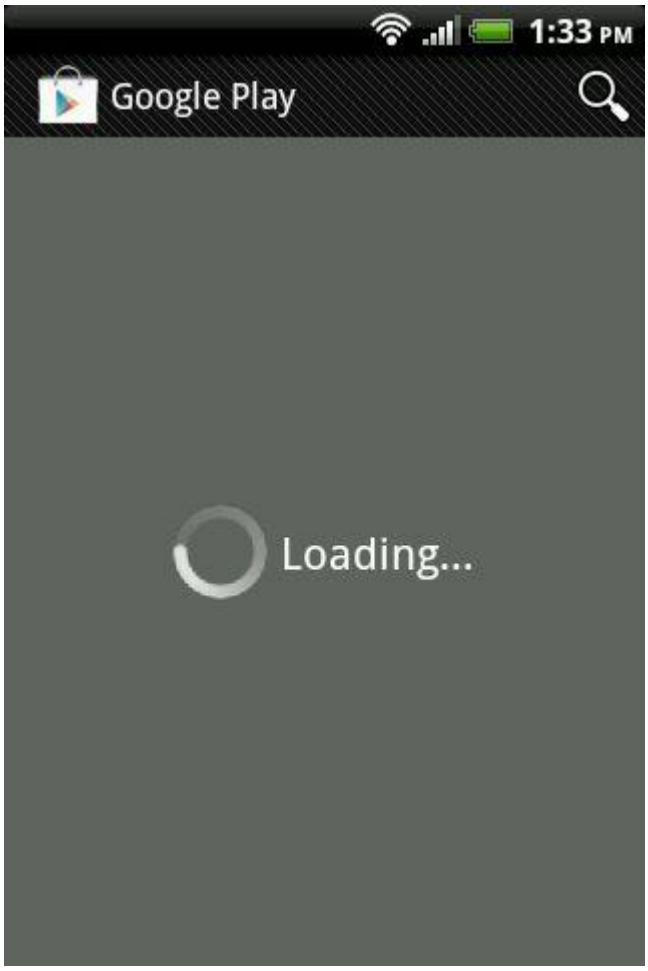
```
long[] vibrate ={0,200,100,200};

notification.vibrate= vibrate;
```

Flash Light:

```
notification.defaults|=Notification.DEFAULT_LIGHTS;

//use the above default or set custom valuse as below

notification.ledARGB=0xffff0000;//red color

notification.ledOnMS=400;

notification.ledOffMS=500;

notification.flags|=Notification.FLAG_SHOW_LIGHTS;
```

3. Dialog Notification

Dialog notification is not an exact type of notification. Dialog is common in window based UIs. A small panel that appears on top of an active window and user will not be able to do any other activity other than acting on the dialog. This is same here too. From an android Activity a dialog will be launched and the Activity loses focus. User should give input and work on the dialog. Once the user action is completed the dialog is closed. Dialog has many uses and one among them is notification to user.



For example we can show a progress bar which is a notification to user. We can ask for confirmation 'yes' or 'no' from user and this is a type of notification. For all these purposes dialog notification is used. There are many types of dialogs available such as,

- AlertDialog
- ProgressDialog
- DatePickerDialog
- TimePickerDialog

8. Write a program to demonstrate autocompleteview with a list of places.

```
package com.example.autocomplete;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;


import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
public class MainActivity extends Activity {
 String[] presidents = {
 "Dwight D. Eisenhower",
```

```
"John F. Kennedy",
"Lyndon B. Johnson",
"Richard Nixon",
"Gerald Ford",
"Jimmy Carter",
"Ronald Reagan",
"George H. W. Bush",
"Bill Clinton",
"George W. Bush",
"Barack Obama"
};

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_dropdown_item_1line, presidents);
AutoCompleteTextView textView = (AutoCompleteTextView)
findViewById(R.id.txtCountries);
textView.setThreshold(3);
textView.setAdapter(adapter);
}
}
```

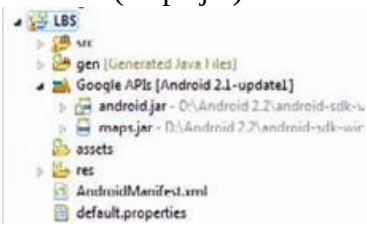
9. Write in details how to display Google maps in your own android applications.

Google Maps is one of the many applications bundled with the Android platform. In addition to simply using the Maps application, you can also embed it into your own applications and make it do some very cool things. This section describes how to use Google Maps in your Android applications and programmatically perform the following:

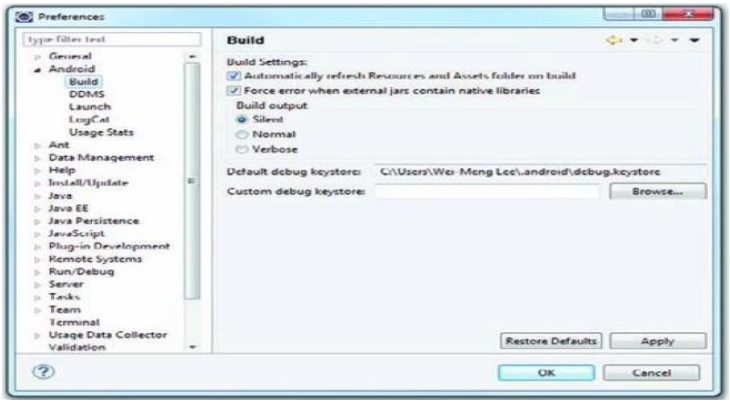
- Change the views of Google Maps.
- Obtain the latitude and longitude of locations in Google Maps.
- Perform geocoding and reverse geocoding (translating an address to latitude and longitude and vice versa).
- Add markers to Google Maps.

We will discuss how to build a project using maps.

Creating the Project: Create a new android project. In order to use Google Maps in your Android application, you need to ensure that you check the Google APIs as your build target. Google Maps is not part of the standard Android SDK, so you need to find it in the Google APIs add-on. If LBS is the name of your project, then you can see the additional JAR file (maps.jar) located under the Google APIs folder as below–



Obtaining the Maps API Key: Beginning with the Android SDK release v1.0, you need to apply for a free Google Maps API key before you can integrate Google Maps into your Android application. When you apply for the key, you must also agree to Google's terms of use, so be sure to read them carefully.



First, if you are testing the application on the Android Emulator or an Android device directly connected to your development machine, locate the SDK debug certificate located in the default folder (C:\Users\<username>\.android for Windows 7 users). You can verify the existence of the debug certificate by going to Eclipse and selecting Window ➪Preferences. Expand the Android item and select Build (as shown in figure above). On the right side of the window, you will be able to see the debug certificate's location. The filename of the debug keystore is debug.keystore. This is the certificate that Eclipse uses to sign your application so that it may be run on the Android Emulator or devices.

10. Write a program to demonstrate the progress bar along with the status of completion

```
package com.example.progressbar1;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.os.Handler;
import android.widget.ProgressBar;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.os.Handler;
import android.widget.ProgressBar;

public class MainActivity extends Activity {

        private static int progress;
        private ProgressBar progressBar;
        private int progressStatus = 0;
        private Handler handler = new Handler();
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        progress = 0;
        progressBar = (ProgressBar) findViewById(R.id.progressbar);
       progressBar.setMax(200);
        //---do some work in background thread---
```

```
 new Thread(new Runnable()
 {
 public void run()
 {
 //---do some work here---
 while (progressStatus < 100)
 {
 progressStatus = doSomeWork();
 handler.post(new Runnable()
 {
          public void run()
          {
                  progressBar.setProgress(progressStatus);
          }
 });
 }
 //---hides the progress bar---
 handler.post(new Runnable()
 {
 public void run()
{
 //---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
progressBar.setVisibility(View.GONE);
 }
 });
 }
 //---do some long lasting work here---
 private int doSomeWork()
 {
 try {
 //---simulate doing some work---
Thread.sleep(500);
 } catch (InterruptedException e)
 {
 e.printStackTrace();
 }
 return ++progress;
 }
 }).start();
 }
}
```