Contin……..

UNIT-5

Using an Application's Icon for Navigation:

- The logo or icon displayed in an ActionBar if clicked navigates you to the home of the application.
- By default, the logo or icon displayed in the ActionBar is nonclickable.
- To make the logo or icon clickable, we must call the ActionBar's setHomeButtonEnabled() method, passing the Boolean value true to it as shown here:

  actionBar.setHomeButtonEnabled(true);
- Menu Item clicks are handled by the onOptionsItemSelected handler of our Activity.
- When the logo or icon is clicked, it is considered that a Menu Item with the ID android.R.id.home is clicked.
- In other words, when we click the logo or icon, the onOptionItemSelected() method is called passing the Menu Item with the ID android.R.id.home to it as a parameter as shown here:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case (android.R.id.home) :
Intent intent = new Intent(this,DemoActionBarActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
break;
default:
return super.onOptionsItemSelected(item);
}
return true;
}
```

For navigating to the home Activity, we use an intent flag, FLAG_ACTIVITY_CLEAR_TOP that clears the stack of all activities on top of the home Activity as shown in the preceding code.

Displaying Action Items

- To display menu items in the ActionBar as action items, we need to add an android:showAsAction attribute to the menu items while defining them in the menu file.
- The value of the showAsAction attribute can be any one of the following:
- **always**—Makes the action item appear on the ActionBar.
- **ifRoom**—Makes the action item appear in the ActionBar, but only if there is room available on the ActionBar. If there's not enough room, the item appears in the Overflow Menu.
- **never**—Makes the menu item appear in the Overflow Menu.

Example

Code Written into the Layout File activity_demo_action_bar.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<Button
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/show_action"
android:text="Show ActionBar"/>
<Button
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/hide_action"
android:text="Hide ActionBar"/>
<Button
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/show_title"
android:text="Show Title"/>
<Button
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/hide_title"
android:text="Hide Title"/>
<Button
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/show_logo"
android:text="Show Logo"/>
<Button
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/hide_logo"
android:text="Hide Logo"/>
</LinearLayout>
```

Code Written into the Menu File activity_demo_action_bar.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/create_datab"
android:title="Create"
android:icon="@drawable/create"
android:orderInCategory="0"
android:showAsAction="ifRoom|withText" />
<item android:id="@+id/menu_search"
android:title="Search"
```

```
android:showAsAction="always"
android:actionViewClass="android.widget.SearchView"/>
</menu>
```
Code Written into the Layout File create.xml
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="This is Create Activity"
android:textStyle="bold" />"
</LinearLayout>
```
Code Written into the New Activity File CreateActivity.java
```
package com.androidunleashed.demoactionbar;
import android.app.ActionBar;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
public class CreateActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.create);
ActionBar actionBar = getActionBar();
actionBar.setHomeButtonEnabled(true);
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case (android.R.id.home) :
Intent intent = new Intent(this, DemoActionBarActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
break;
default:
return super.onOptionsItemSelected(item);
}
return true;
}
}
```

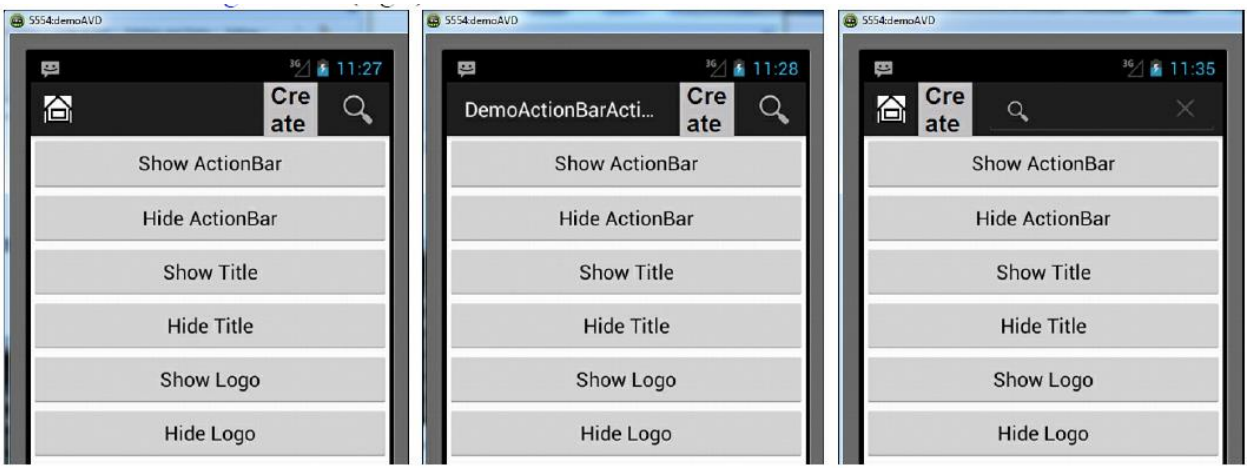
Code Written into the Configuration File AndroidManifest.xml

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.androidunleashed.DemoActionBar"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk
android:minSdkVersion="11"
android:targetSdkVersion="15" />
<application
android:icon="@drawable/home"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".DemoActionBarActivity"
android:label="@string/title_activity_demo_action_bar">
<intent-filter>
<action
android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".CreateActivity"
android:label="@string/app_name" />
</application>
</manifest>
```

Code Written into the Java Activity File DemoActionBarActivity.java

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.app.ActionBar;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.content.Intent;
public class DemoActionBarActivity extends Activity {
Intent intent;
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_demo_action_bar);
final ActionBar actionBar = getActionBar();
Button showAction = (Button) this.findViewById(R.id.show_action);
Button hideAction = (Button) this.findViewById(R.id.hide_action);
Button showTitle = (Button) this.findViewById(R.id.show_title);
```

```
Button hideTitle = (Button) this.findViewById(R.id.hide_title);
Button showLogo = (Button) this.findViewById(R.id.show_logo);
Button hideLogo = (Button) this.findViewById(R.id.hide_logo);
showAction.setOnClickListener(new Button.OnClickListener(){
public void onClick(View v)
{ actionBar.show(); } });
hideAction.setOnClickListener(new Button.OnClickListener(){
public void onClick(View v)
{ actionBar.hide(); } });
showTitle.setOnClickListener(new Button.OnClickListener(){
public void onClick(View v)
{
actionBar.setDisplayShowTitleEnabled(true); } });
hideTitle.setOnClickListener(new Button.OnClickListener(){
public void onClick(View v)
{ actionBar.setDisplayShowTitleEnabled(false); } });
showLogo.setOnClickListener(new Button.OnClickListener()
{
public void onClick(View v)
{ actionBar.setDisplayShowHomeEnabled(true); } });
hideLogo.setOnClickListener(new Button.OnClickListener()
{
public void onClick(View v)
{ actionBar.setDisplayShowHomeEnabled(false);
} });
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.mymenu, menu);
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case R.id.create_datab:intent = new Intent(this, CreateActivity.class);
startActivity(intent);
break;
default:
return super.onOptionsItemSelected(item);
}
return true;
}
}
```



Replacing a Menu with the ActionBar:

Code Written into the Menu File activity_action_bar_app.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/create_datab"
android:title="Create Database"
android:icon="@drawable/create"
android:orderInCategory="0"
android:showAsAction="ifRoom|withText" />
<item android:id="@+id/insert_rows"
android:title="Insert Rows"
android:icon="@drawable/insert"
android:showAsAction="ifRoom" />
<item android:id="@+id/list_rows"
android:title="List Rows"
android:showAsAction="ifRoom" />
<item android:id="@+id/search_row"
android:title="Search"
android:icon="@drawable/search"
android:showAsAction="ifRoom|withText" />
<item android:id="@+id/delete_row"
android:title="Delete"
android:showAsAction="never" />
<item android:id="@+id/update_row"
android:title="Update"
android:icon="@drawable/update"
android:showAsAction="always" />
</menu>
```

Code Written into the Java Activity File ActionBarAppActivity.java

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
public class ActionBarAppActivity extends Activity {
@Override
public void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_action_bar_app);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.activity_action_bar_app, menu);
return true;
}
}
```

Creating a Tabbed ActionBar

- Tabbed and drop-down list ActionBars display menu items in the form of tabs and drop-down lists
- In tabbed and dropdown ActionBars, only one type of navigation can be enabled at a time.
- To display navigation tabs in an ActionBar, its setNavigationMode() method is called, passing the value ActionBar.NAVIGATION_MODE_TABS as a parameter as follows:
  actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
- After determining Navigation mode we add the tabs to the ActionBar by calling its addTab()method:
  actionBar.addTab(actionBar.newTab().setText("Create").setTabListener
- setIcon() method is used to define an image for the tab.
- setContentDescription() method is used to supply more detailed information of the tab.

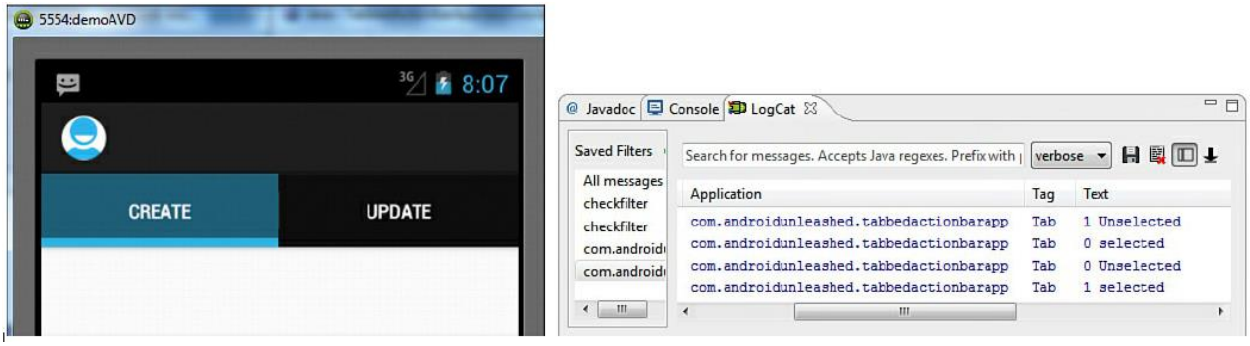
Example:

```
Tab tab1 = actionBar.newTab();
tabOne.setText("Create")
.setIcon(R.drawable.ic_launcher)
.setContentDescription("Creating the Database")
.setTabListener(this));
actionBar.addTab(tab1);
```

Code Written into the Java Activity File TabbedActionBarAppActivity.java

```
package com.androidunleashed.tabbedactionbarapp;
import android.app.Activity;
import android.os.Bundle;
import android.app.ActionBar;
import android.app.ActionBar.Tab;
import android.app.FragmentTransaction;
import android.util.Log;
public class TabbedActionBarAppActivity extends Activity
implements ActionBar.Tab-Listener {
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
final ActionBar actionBar = getActionBar();
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
actionBar.setDisplayShowTitleEnabled(false);
```

```
actionBar.addTab(actionBar.newTab().setText("Create").setTabListener(actionBar.addTab(actio
nBar.newTab().setText("Update").setTabListener(}
@Override
public void onTabReselected(Tab tab, FragmentTransaction ft)
{
Log.d("Tab", String.valueOf(tab.getPosition()) + " reselected");
}
@Override
public void onTabSelected(Tab tab, FragmentTransaction ft) {
Log.d("Tab", String.valueOf(tab.getPosition()) + "selected");
}
@Override
public void onTabUnselected(Tab tab, FragmentTransaction ft)
{
Log.d("Tab", String.valueOf(tab.getPosition()) + "Unselected");
}
}
```



left right

Screen showing two tabs in the tabbed ActionBar (left), and log messages displayedon selecting the action tabs (right).

Creating a Drop-Down List ActionBar:

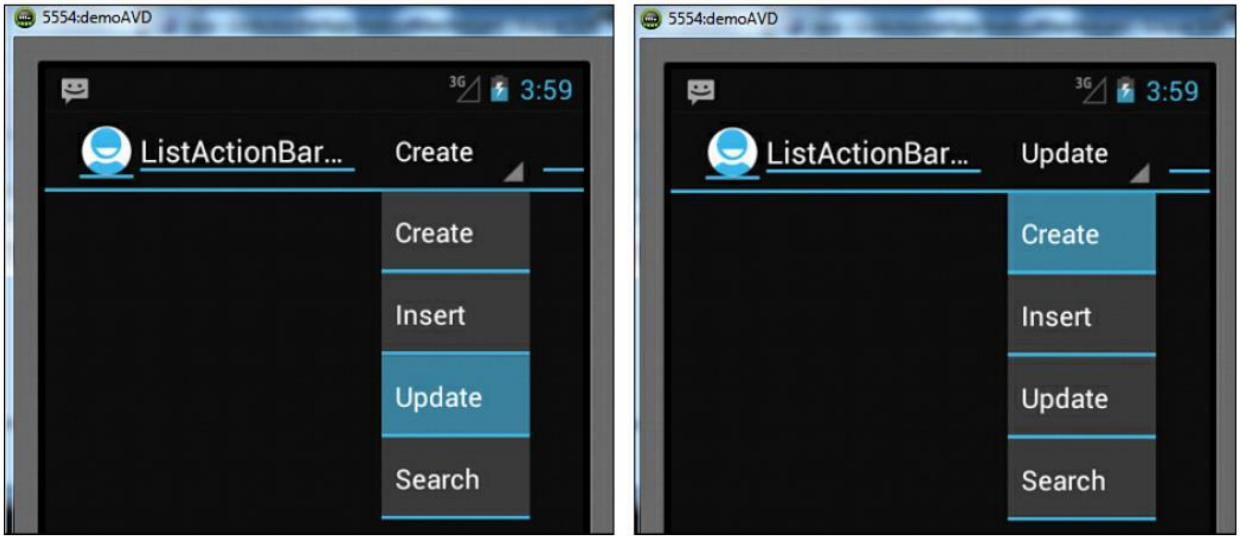
- In a drop-down list ActionBar, the menu items are displayed in the form of a drop-down list. It is popularly used for displaying the content within an Activity on the basis of the selection made by the user.
- To display a drop-down list in an ActionBar, its setNavigationMode() method is called, passing the value ActionBar.NAVIGATION_MODE_LIST as a parameter to it as shown here:

  actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_LIST);
- For displaying options in the drop-down list, we use the Adapter that implements the SpinnerAdapter interface, like an ArrayAdapter or SimpleCursorAdapter.
- We use an ArrayAdapter in this application.

Code Written into the Java Activity File ListActionBarAppActivity.java

```
package com.androidunleashed.listactionbarapp;
import android.app.Activity;
import android.os.Bundle;
import android.app.ActionBar.OnNavigationListener;
import android.app.ActionBar;
import android.widget.ArrayAdapter;
import android.util.Log;
```

```
public class ListActionBarAppActivity extends Activity {
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
String[] items = new String[] { "Create", "Insert","Update", "Search" };
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_dropdown_item, items);
ActionBar actionBar = getActionBar();
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_LIST);
actionBar.setListNavigationCallbacks(adapter, onNavigationItemSelected);
}
OnNavigationListener onNavigationItemSelected = new OnNavigationListener() {
@Override
public boolean onNavigationItemSelected(int itemPosition, long itemId) {
Log.d("Option ", String.valueOf(itemId) + " is selected");
return true;
}
};
}
```



Using Dtabases:

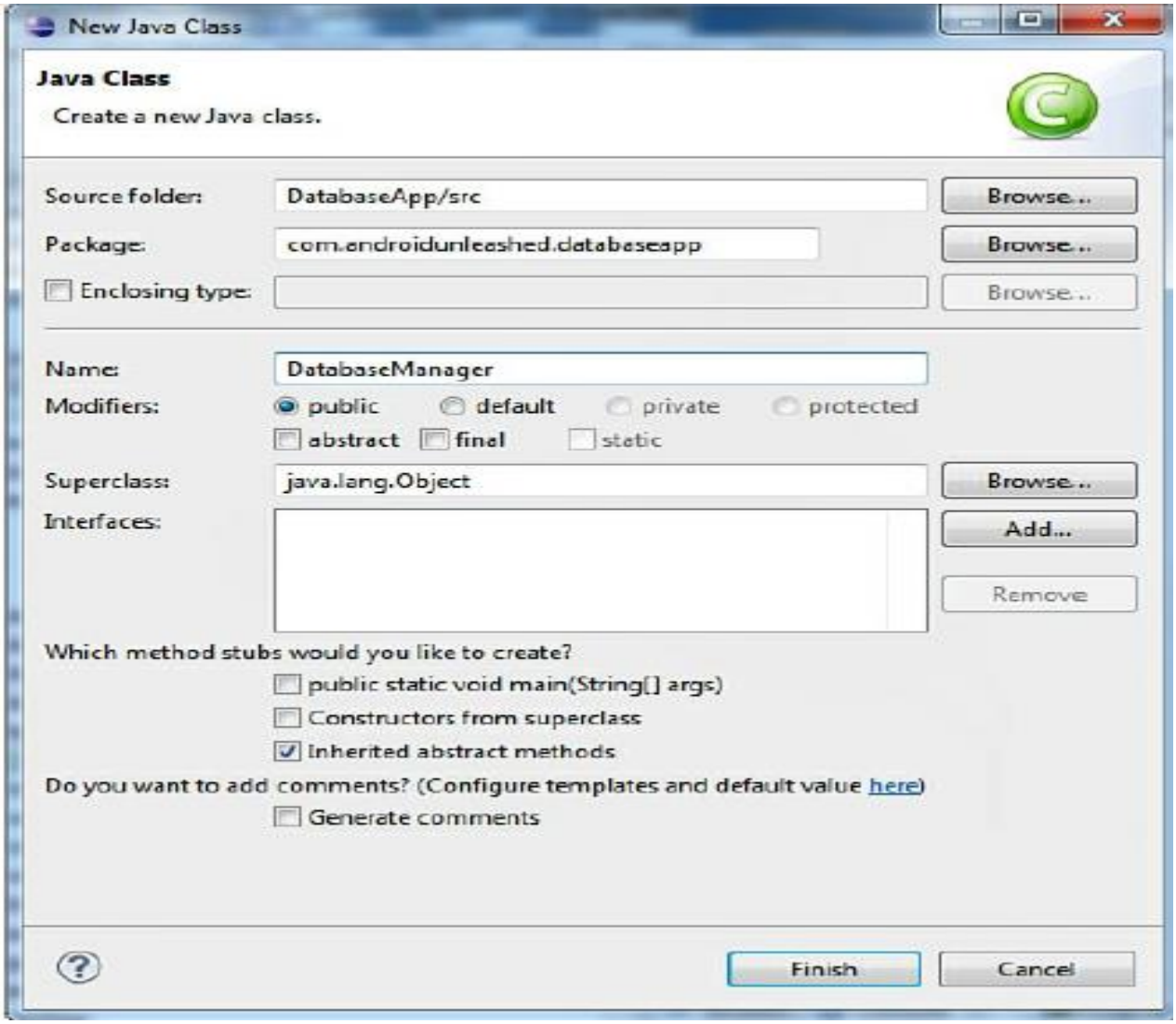
Using the SQLiteOpenHelper Class

- Android databases are stored in the /data/data/<package_name>/databases folder on devices or emulators.
- To create an Android application that stores, accesses, and manipulates user information in a SQLite relational database, we use the SQLiteOpenHelper class.
- The SQLiteOpenHelper class is an abstract class used to create, open, and upgrade databases.
- It provides several methods including getWritableDatabase(), getReadableDatabase(), and close().
- The SQLiteDatabase class also provides methods including insert() and query(), which are used to insert rows in the database table and execute different SQLite queries on the database table.

Building an SQLite Project:

SQLite project can be build by the following steps.

1. Begin Eclipse IDE. Create a new android project with the name DatabaseApplication
2. A database is created with the name Student
3. A table is created with the name details that contains three columns such sid,sname,sdept of the student.
4. Rows are added to the table and are displayed using textview control.
5. Right-click on the link com.androidunleashed.databaseapp package present in the window package explore to add a java class.select new and click on class option.
6. A dialog box opens showing the new java class in which values are required to be entered by a user as shown below



We write code in DatabaseManager.java to perform the following tasks:

- Inherit from the SQLiteOpenHelper class to access and take advantage of the methods defined in it.
- Open and return the writable and readable database file instance to perform writing and reading operations on the database, respectively.
- Create a database table, products, consisting of three columns: code, product_name, and price.

- Add rows to the products table. The information for the new product is supplied by the activity file DatabaseAppActivity.java.
- Fetch rows from the products table and return them to the activity file DatabaseAppActivity.java for displaying on the screen.

Code Written into the Java Class DatabaseManager.java

```
import android.database.sqlite.SQLiteDatabase;
import android.content.Context;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
public class DatabaseManager {
public static final String DB_NAME = "shopping";
public static final String DB_TABLE = "products";
public static final int DB_VERSION = 1;
private static final String CREATE_TABLE = "CREATE TABLE " + DB_TABLE + " (code
INTEGER PRIMARY KEY, product_name TEXT, price FLOAT);";
private SQLHelper helper;
private SQLiteDatabase db;
private Context context;
public DatabaseManager(Context c){
this.context = c;
helper=new SQLHelper(c);
this.db = helper.getWritableDatabase();
}
public DatabaseManager openReadable() throws
android.database.SQLException {
helper=new SQLHelper(context);
db = helper.getReadableDatabase();
return this;
}
public void close(){
helper.close();
}
public void addRow(Integer c, String n, Float p){
ContentValues newProduct = new ContentValues();
newProduct.put("code", c);
newProduct.put("product_name", n);
newProduct.put("price", p);
try{db.insertOrThrow(DB_TABLE, null, newProduct);}
catch(Exception e)
{
Log.e("Error in inserting rows ", e.toString());
e.printStackTrace();
}
}
```

```
public String retrieveRows(){
String[] columns = new String[]{"code", "product_name", "price"};
Cursor cursor = db.query(DB_TABLE, columns, null, null, null, null, null);
String tablerows = "";
cursor.moveToFirst();
while (cursor.isAfterLast() == false) {
tablerows = tablerows + cursor.getInt(0) + ","+cursor.getString(1)+", "+cursor.getFloat(2)+ "\n";
cursor.moveToNext();
}
if (cursor != null && !cursor.isClosed()) {
cursor.close();
}
return tablerows;
}
public class SQLHelper extends SQLiteOpenHelper {
public SQLHelper(Context c){
super(c, DB_NAME, null, DB_VERSION);
}
@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL(CREATE_TABLE);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion,
int newVersion) {
Log.w("Products table","Upgrading database i.e.dropping table and recreating it");
db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);
onCreate(db);
}
}
}
```

Fetching the Desired Rows from Tables:

- To fetch the desired rows from a database table, we execute queries containing a criterion against the given database.
- The queries are executed through the query() method provided by the SQLiteDatabase class.
- The parameters that are passed to the query() method are shown here: db.query(bool_uniq, db_table, array_columns, where_clause, select_arg, group_by, having_clause, order);.

Parameter	Usage
bool_uniq	Optional Boolean value to determine whether the result set should contain only unique values. A True value confirms the unique values in the result set.
db_table	The database table to be queried.
array_columns	A string array containing the list of columns we want in the result set.
where_clause	Defines the criteria for the rows to be returned. For specifying values in the clause, ? wildcard(s) are used, which are then replaced by the values stored in the select_arg parameter.
select_arg	An array of strings that provide values for the ? wildcards used in the where_clause parameter.
group_by	Defines the condition for grouping the returned result set.
having_clause	Defines the conditions to be applied to the groups defined in the group_by clause.
order	Defines the order of the returned rows.

Example:

```
public static final String DB_TABLE = "products";
String[] columns = new String[]{"code", "product_name","price"};
Cursor cursor = db.query(true, DB_TABLE, columns, null, null,null, null, null);
Cursor cursor = db.query(DB_TABLE, columns, null, null, null, null, null);
```

Using Cursors:

Cursors are defined as the pointers that points at the result set of underlying data.These are used for traversing the rows and retrieving the column data of a result set.The cursor class contains many different methods used for setting the pointer at the required position of row and columns of the result set.They are as follows

Method	Usage
moveToFirst	Moves the cursor to the first row in the result set
moveToNext	Moves the cursor to the next row
moveToPrevious	Moves the cursor to the previous row
getCount	Returns the count of the rows in the result set
getColumn-IndexOrThrow	Returns the index for the column with the specified name. It throws an exception if no column exists with that name
getColumnName	Returns the column name with the specified column index
getColumnNames	Returns a string array that contains all the column names in the current cursor
moveToPosition	Moves the cursor to the specified row
getPosition	Returns the current cursor position

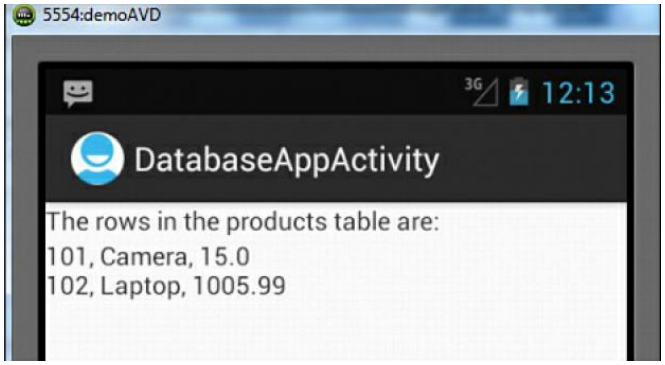
Code Written into the Java Activity File DatabaseAppActivity.java

```
package com.androidunleashed.databaseapp;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class DatabaseAppActivity extends Activity {
```

```
private DatabaseManager mydManager;
private TextView response;
private TextView productRec;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_database_app);
response=(TextView)findViewById(R.id.response);
productRec=(TextView)findViewById(R.id.prodrec);
mydManager = new DatabaseManager(this);
mydManager.addRow(101, "Camera", Float.parseFloat("15"));
mydManager.addRow(102, "Laptop", Float.parseFloat("1005.99"));
mydManager.close();
mydManager = new DatabaseManager(this);
mydManager.openReadable();
String tableContent = mydManager.retrieveRows();
response.setText("The rows in the products table are:");
productRec.setText(tableContent);
mydManager.close();
}
}
```

Code Written into the Layout File activity_database_app.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/response" />
<TextView
android:id="@+id/prodrec"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
</LinearLayout>
```



Accessing databases with ADB:

ADB can be accessed through the windows operating system by following the below steps:

- Open command prompt and enter the path of adb.exe file.
- The default path is C:\Program Files (x86)\Android\android-sdk\platform-tools.

The following are the commands used for interacting with a device or an emulator.

1. **adb Devices:**The list of currently running emulator and devices are shown by device command.

   Example:

   C:\Program Files (x86)\Android\android-sdk\platform-tools>adb
   Devices List of devices attached emulator-5554 offline

2. **adb Shell:** The Shell is activated using adb Shell command.

   Example:

   C:\Program Files (x86)\Android\android-sdk\platform-tools>adb shell#

3. **ls -1:** long listing command shows the list of directories in the emulator/device:

   Example:

   ```
   # ls -l
    ls -l
   drwxrwxrwt  root          root          2011-08-06    09:14
   sqlite_stmt_journals
   drwxrwx--- system        cache         2011-07-22    23:14 cache
   d---rwxrwx system        system        1970-01-01    05:30 sdcard
   ```

4. **ls /data/data:** command shows the list of files and directories in the subdirectory of the data directory. We see the list of all installed applications in the emulator. The installed application's package names are also displayed:

   ```
   # ls /data/data
   ls /data/data
   com.android.mms
   com.android.googlesearch
   com.android.launcher
    ::::::::::::
    ::::::::::::
   com.androidunleashed.databaseapp
   ```

5. **ls command:** we see the list of files and directories in the databases subdirectory. As expected, the name of our shopping database is displayed:

   ```
   # ls
     ls
    shopping
   ```

6. **Sqlite3:** The database is made active by using sqlite3 command.

   Example:

   ```
   # sqlite3 shopping
    sqlite3 shopping
    SQLite version 3.5.9
    Enter ".help" for instructions
   ```

7. **.tables:** command displays the tables that exist in the currently active database:

   Example:

   ```
   sqlite> .tables
   ```

.tables
android_metadata products

8. **.schema:** The structure of all the tables present in the currently active database is given by .schema command.

Example:

sqlite> .schema
.schema
CREATE TABLE android_metadata (locale TEXT);
CREATE TABLE products (code INTEGER PRIMARY KEY, product_name TEXT, price FLOAT);

9. The **SQL SELECT** command can be used to see the number of rows in the products table of our shopping database:

sqlite> select * from products;
select * from products;
101|Camera|15.0
102|Laptop|1005.98999023438

10. We can also issue an **SQL DELETE** command to delete a row. The following command deletes a row from the products table that has a product code equal to 102.

sqlite> delete from products where code=102;
delete from products where code=102;

To confirm that the row was really deleted, we can give a SQL SELECT command as shown here:

sqlite> select * from products;
select * from products;
101|Camera|15.0

11. **SQL UPDATE** command updates the product name to Handy Cam, whose code is equal to 101.

Example:

sqlite> update products set product_name='Handy Cam' where code=101;
update products set product_name='Handy Cam' where code=101;

We can confirm whether the row was successfully updated by issuing an SQL SELECT command:

sqlite> select * from products;
select * from products;
101|Handy Cam|15.0

12. **.exit:** command is used for leaving the sqlite>prompt

Example:

Sqlite>.exit

Creating data entry form:

Code Written into the Layout File activity_database_app.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent" >
<TextView
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/response" />
<TextView
android:id="@+id/prodrec"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
<TableLayout
android:id="@+id/add_table"
android:layout_width="match_parent"
android:layout_height="wrap_content" >
<TableRow >
<TextView
android:text="Product Code:"
android:padding="3dip" />
<EditText
android:id="@+id/prod_code"
android:layout_width="match_parent"
android:layout_height="wrap_content"/>
</TableRow>
<TableRow >
<TextView
android:text="Product Name:"
android:padding="3dip" />
<EditText
android:id="@+id/prod_name"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:minWidth="150dip" />
</TableRow>
<TableRow >
<TextView
android:text="Product Price:"
android:padding="3dip" />
<EditText
android:id="@+id/prod_price"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:minWidth="50dip" />
</TableRow>
<TableRow >
<Button
android:id="@+id/add_button"
android:text="Add Product"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="3dip" />
```

```
<Button
android:id="@+id/cancel_button"
android:text="Cancel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="3dip" />
</TableRow>
</TableLayout>
</LinearLayout>
```

Code Written into the Java Class DatabaseManager.java

```
package com.androidunleashed.databaseapp;
import android.database.sqlite.SQLiteDatabase;
import android.content.Context;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
public class DatabaseManager {
public static final String DB_NAME = "shopping";
public static final String DB_TABLE = "products";
public static final int DB_VERSION = 1;
private static final String CREATE_TABLE = "CREATE TABLE " + DB_TABLE + " (code
INTEGER PRIMARY KEY, product_name TEXT, price FLOAT);";
private SQLHelper helper;
private SQLiteDatabase db;
private Context context;
public DatabaseManager(Context c){
this.context = c;
helper=new SQLHelper(c);
this.db = helper.getWritableDatabase();
}
public DatabaseManager openReadable() throws
android.database.SQLException {
helper=new SQLHelper(context);
db = helper.getReadableDatabase();
return this;
}
public void close(){
helper.close();
}
public boolean addRow(int c, String n, float p){
ContentValues newProduct = new ContentValues();
newProduct.put("code", c);
newProduct.put("product_name", n);
newProduct.put("price", p);
try{db.insertOrThrow(DB_TABLE, null, newProduct);}
catch(Exception e) {
```

```
Log.e("Error in inserting rows ", e.toString());
e.printStackTrace();
return false;
}
db.close();
return true;
}
public String retrieveRows(){
String[] columns = new String[]{"code", "product_name",
"price"};
Cursor cursor = db.query(DB_TABLE, columns, null, null,
null, null, null);
String tablerows = "";
cursor.moveToFirst();
while (cursor.isAfterLast() == false) {
tablerows = tablerows + cursor.getInt(0) + ", "+cursor.getString(1)+", "+ cursor.getFloat(2)+"\n";
cursor.moveToNext();
}
if (cursor != null && !cursor.isClosed()) {
cursor.close();
}
return tablerows;
}
public class SQLHelper extends SQLiteOpenHelper {
public SQLHelper(Context c){
super(c, DB_NAME, null, DB_VERSION);
}
@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL(CREATE_TABLE);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion,
int newVersion) {
Log.w("Products table","Upgrading database i.e.dropping table and recreating it");
db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);
onCreate(db);
}
}
}
```

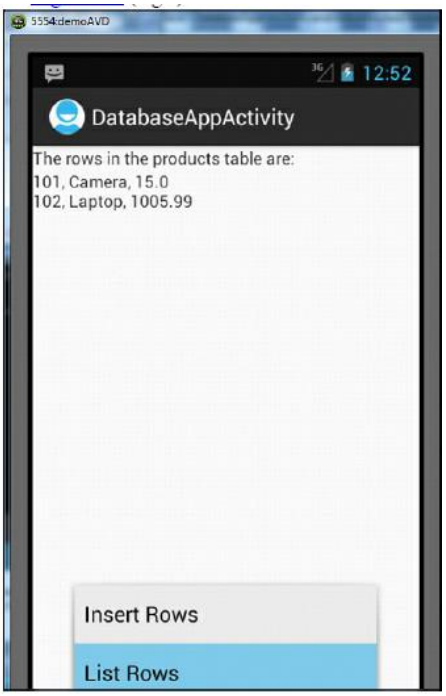
Code Written into the DatabaseAppActivity.java Java Activity File

```
package com.androidunleashed.databaseapp;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.TextView;
import android.view.Menu;
```

```
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TableLayout;
import android.view.View.OnClickListener;
import android.view.inputmethod.InputMethodManager;
public class DatabaseAppActivity extends Activity {
private DatabaseManager mydManager;
private TextView response;
private TextView productRec;
EditText pcode, pname, price;
Button addButton;
private TableLayout addLayout;
private boolean recInserted;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_database_app);
response=(TextView)findViewById(R.id.response);
productRec=(TextView)findViewById(R.id.prodrec);
addLayout=(TableLayout)findViewById(R.id.add_table);
addLayout.setVisibility(View.GONE);
response.setText("Press MENU button to display menu");
Button addButton = (Button)
findViewById(R.id.add_button);
addButton.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
mydManager = new
DatabaseManager(DatabaseAppActivity.this);
pcode=(EditText)findViewById(R.id.prod_code);
pname=(EditText)findViewById(R.id.prod_name);
price=(EditText)findViewById(R.id.prod_price);
recInserted=mydManager.addRow(Integer.parseInt(pcode.toString()),
pname.getText().toString(),
Float.parseFloat (price.getText().toString()));
addLayout.setVisibility(View.GONE);
if(recInserted)
response.setText("The row in the products table is inserted");
else
response.setText("Sorry, some errors occurred while inserting the row in the products table");
InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
imm.hideSoftInputFromWindow(price.getWindowToken(),
InputMethodManager.HIDE_NOT_ALWAYS);
mydManager.close();
```

```
pcode.setText("");
pname.setText("");
price.setText("");
productRec.setText("");
}
});
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.activity_database_app, menu);
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case R.id.insert_rows: addLayout.setVisibility(View.VISIBLE);
response.setText("Enter information of the new product");
productRec.setText("");
break;
case R.id.list_rows: showRec();
break;
}
return true;
}
public boolean showRec(){
addLayout.setVisibility(View.GONE);
mydManager = new DatabaseManager(this);
mydManager.openReadable();
String tableContent = mydManager.retrieveRows();
response.setText("The rows in the products table are:");
productRec.setText(tableContent);
mydManager.close();
return true
}
}
```


5554:demoAVD
12:42
DatabaseAppActivity
Press MENU button to display menu


5554:demoAVD
12:52
DatabaseAppActivity
The rows in the products table are:
101, Camera, 15.0
102, Laptop, 1005.99
Insert Rows
List Rows


5554:demoAVD
12:54
DatabaseAppActivity
Enter information of the new product
Product Code: 103
Product Name: CellPhone
Product Price: 24.50
Add Product
Cancel


5554:demoAVD
12:55
DatabaseAppActivity
The row in the products table is inserted


5554:demoAVD
12:55
DatabaseAppActivity
The rows in the products table are:
101, Camera, 15.0
102, Laptop, 1005.99
103, CellPhone, 24.5
Insert Rows
List Rows


5554:demoAVD
5:10
DatabaseAppActivity
The rows in the products table are:
101, Camera, 15.0
102, Laptop, 1005.99
103, CellPhone, 24.5

Displaying Table Rows Via ListView:

Code Written into the Layout File activity_database_app.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/response" />
<ListView
android:id="@+id/prodrec"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:drawSelectorOnTop="false"/>
<TableLayout
android:id="@+id/add_table"
android:layout_width="match_parent"
android:layout_height="wrap_content" >
<TableRow >
<TextView
android:text="Product Code:"
android:padding="3dip" />
<EditText
android:id="@+id/prod_code"
android:layout_width="match_parent"
android:layout_height="wrap_content"/>
</TableRow>
<TableRow >
<TextView
android:text="Product Name:"
android:padding="3dip" />
<EditText
android:id="@+id/prod_name"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:minWidth="150dip" />
</TableRow>
<TableRow >
<TextView
android:text="Product Price:"
android:padding="3dip" />
<EditText
android:id="@+id/prod_price"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:minWidth="50dip"/>
</TableRow>
<TableRow >
<Button
android:id="@+id/add_button"
android:text="Add Product"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="3dip"/>
<Button
android:id="@+id/cancel_button"
android:text="Cancel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="3dip" />
</TableRow>
</TableLayout>
</LinearLayout>
```

Code Written into the Java Class DatabaseManager.java

```
package com.androidunleashed.databaseapp;
import android.database.sqlite.SQLiteDatabase;
import android.content.Context;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import java.util.ArrayList;
public class DatabaseManager {
public static final String DB_NAME = "shopping";
public static final String DB_TABLE = "products";
public static final int DB_VERSION = 1;
private static final String CREATE_TABLE = "CREATE TABLE " + DB_TABLE + " (code
INTEGER PRIMARY KEY, product_name TEXT, price FLOAT);";
private SQLHelper helper;
private SQLiteDatabase db;
private Context context;
public DatabaseManager(Context c){
this.context = c;
helper=new SQLHelper(c);
this.db = helper.getWritableDatabase();
}
public DatabaseManager openReadable() throws
android.database.SQLException {
helper=new SQLHelper(context);
db = helper.getReadableDatabase();
return this;
```

```
}
public void close(){
helper.close();
}
public boolean addRow(int c, String n, float p){
ContentValues newProduct = new ContentValues();
newProduct.put("code", c);
newProduct.put("product_name", n);
newProduct.put("price", p);
try{db.insertOrThrow(DB_TABLE, null, newProduct);}
catch(Exception e)
{
Log.e("Error in inserting rows ", e.toString());
e.printStackTrace();
return false;
}
db.close();
return true;
}
public ArrayList<String> retrieveRows(){
ArrayList<String> productRows=new ArrayList<String>();
String[] columns = new String[]{"code", "product_name", "price"};
Cursor cursor = db.query(DB_TABLE, columns, null, null, null, null, null);
cursor.moveToFirst();
while (cursor.isAfterLast() == false) {
productRows.add(Integer.toString(cursor.getInt(0))      +      ",      "+cursor.get-      String(1)+",
"+Float.toString(cursor.getFloat(2)));
cursor.moveToNext();
}
if (cursor != null && !cursor.isClosed()) {
cursor.close();
}
return productRows;
}
public class SQLHelper extends SQLiteOpenHelper {
public SQLHelper(Context c){
super(c, DB_NAME, null, DB_VERSION);
}
@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL(CREATE_TABLE);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion,
int newVersion) {
Log.w("Products table","Upgrading database i.e. dropping table and recreating it");
db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);
```

```
onCreate(db);
}
}
}
```

Code Written into the DatabaseAppActivity.java Java Activity File

```
package com.androidunleashed.databaseapp;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.TextView;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TableLayout;
import android.view.View.OnClickListener;
import android.view.inputmethod.InputMethodManager;
import android.widget.ListView;
import android.widget.ArrayAdapter;
import java.util.ArrayList;
public class DatabaseAppActivity extends Activity {
private DatabaseManager mydManager;
private TextView response;
private ListView productRec;
EditText pcode, pname, price;
Button addButton;
private TableLayout addLayout;
private Boolean recInserted;
ArrayList<String> tableContent;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_database_app);
response=(TextView)findViewById(R.id.response);
productRec=(ListView)findViewById(R.id.prodrec);
addLayout=(TableLayout)findViewById(R.id.add_table);
addLayout.setVisibility(View.GONE);
response.setText("Press MENU button to display menu");
Button add_button = (Button) findViewById(R.id.add_button);
add_button.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
mydManager = new DatabaseManager(DatabaseAppActivity.this);
pcode=(EditText)findViewById(R.id.prod_code);
pname=(EditText)findViewById(R.id.prod_name);
price=(EditText)findViewById(R.id.prod_price);
```

```
recInserted=mydManager.addRow(Integer.parseInt(pcode.get-Text().toString()),
pname.getText().toString(),Float.parseFloat(price.getText().
toString()));
addLayout.setVisibility(View.GONE);
if(recInserted)
response.setText("The row in the products table is inserted");
else
response.setText("Sorry, some errors occurred while inserting the row in the products table");
InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
imm.hideSoftInputFromWindow(price.getWindowToken(),
InputMethodManager.
HIDE_NOT_ALWAYS);
mydManager.close();
pcode.setText("");
pname.setText("");
price.setText("");
productRec.setVisibility(View.GONE);
}
});
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.activity_database_app, menu);
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case R.id.insert_rows:
addLayout.setVisibility(View.VISIBLE);
response.setText("Enter information of the new product");
productRec.setVisibility(View.GONE);
break;
case R.id.list_rows: showRec();
break;
}
return true;
}
public boolean showRec(){
addLayout.setVisibility(View.GONE);
mydManager = new DatabaseManager(this);
mydManager.openReadable();
tableContent = mydManager.retrieveRows();
response.setText("The rows in the products table are:");
productRec = (ListView)findViewById(R.id.prodrec);
```

```
ArrayAdapter<String> arrayAdpt=new ArrayAdapter<String>
(this,
android.R.layout.simple_list_item_1, tableContent);
productRec.setAdapter(arrayAdpt);
productRec.setVisibility(View.VISIBLE);
mydManager.close();
return true;
}
}
```

Communicating SMS and Emails:

Understanding Broadcast Receivers:

A broadcast receiver is a component that responds to different messages that are broadcast by the system or other applications. Messages such as new mail, low battery, captured photo, or completed download require attention. Such messages are broadcast as an Intent object that is waiting for broadcast receivers to respond.

Broadcasting an Intent

To broadcast an Intent, we first create an Intent object, assign a specific action to it, attach data or a message to the broadcast receiver, and finally broadcast it.

The methods that are used to broadcast an intent are as follows:

Method	Description
`putExtra()`	Used to add data or a message to the `Intent` that we want to send to the broadcast receiver. Syntax: `putExtra(String name, String value)` Where `name` is the key or name of the `value` that we want to pass along with the `Intent`. The `name` is used to identify the `value`.
`setAction()`	Used to set the action to perform on the data or message being sent with the `Intent`. Syntax: `setAction(String action)` The broadcast receiver uses the `getAction()` method to retrieve the action to be performed on the received data.
`sendBroadcast()`	Available on the `Context` class, this method is used to send the broadcast `Intent` to all the registered `Intent` receivers. Syntax: `void sendBroadcast(intent_to_broadcast)` Where the `intent_to_broadcast` parameter represents the `Intent` that we want to broadcast.

Example:

```
public static String BROADCAST_STRING = "com.androidunleashed.testingbroadcast";
Intent broadcastIntent = new Intent();
broadcastIntent.putExtra("message", "New Email arrived");
broadcastIntent.setAction(BROADCAST_STRING);
sendBroadcast(broadcastIntent);
```

A broadcast receiver is a class that extends the BroadcastReceiver. It also needs to be registered as a receiver in an Android application via the AndroidManifest.xml file or through code at runtime. The broadcast receiver class needs to implement the onReceive() method. The following is sample code of an onReceive() method:

```
public void onReceive(Context context, Intent intent) {
```

```
String actionName = intent.getAction();
if(actionName != null &&
actionName.equals("com.androidunleashed.testingbroadcast"))
{
String msg = intent.getStringExtra("message");
Log.d("Received Message: ",msg);
}
}
```

The getAction() and getStringExtra() methods used here need some explanation:

getAction()—Retrieves the action to be performed from the Intent object. It is the action that indicates what task has to be performed on the data passed along with the Intent.

Syntax:

```
getAction()
```

getStringExtra()—Retrieves the extended data from the Intent.

Syntax:

```
getStringExtra(String name)
```

where name represents the key or the name assigned to the value while adding data to the Intent through the putExtra() method.

Code Written into activity_broadcast_app.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<Button
android:id="@+id/broadcast_button"
android:text="Send Broadcast"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center" />
</LinearLayout>
```

Code Written into BroadcastAppActivity.java

```
package com.androidunleashed.broadcastapp;
import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.widget.Button;
import android.view.View;
public class BroadcastAppActivity extends Activity {
public static String BROADCAST_STRING ="com.androidunleashed.testingbroadcast";
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_broadcast_app);
Button broadcastButton = (Button) this.findViewById(R.id.broadcast_button);
broadcastButton.setOnClickListener(new Button.OnClickListener(){
public void onClick(View v) {
```

```
Intent broadcastIntent = new Intent();
broadcastIntent.putExtra("message", "New Email arrived");
broadcastIntent.setAction(BROADCAST_STRING);
sendBroadcast(broadcastIntent);
}
});
}
}
```

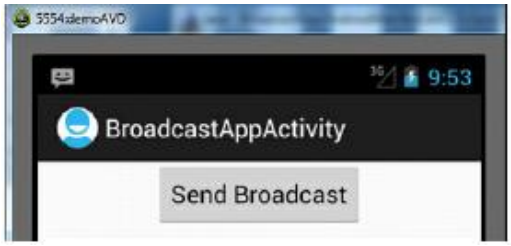
Code Written into ReceiveBroadcastActivity.java

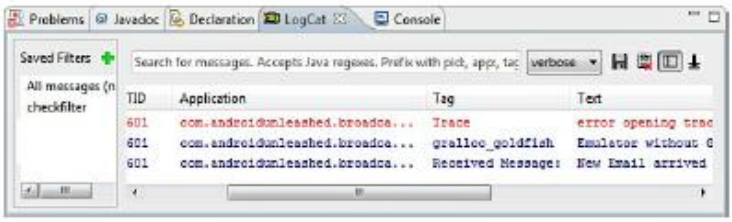
```
package com.androidunleashed.broadcastapp;
import android.content.BroadcastReceiver;
import android.content.Intent;
import android.content.Context;
import android.util.Log;
public class ReceiveBroadcastActivity extends
BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent) {
String actionName = intent.getAction();
if(actionName != null && actionName.equals("com.androidunleashed.testingbroadcast"))
{
String msg = intent.getStringExtra("message");
Log.d("Received Message: ",msg);
}
}
}
```

Code in the AndroidManifest.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.androidunleashed.broadcastapp"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk android:minSdkVersion="8"
android:targetSdkVersion="15" />
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".BroadcastAppActivity"
android:label="@string/title_activity_broadcast_app"
>
<intent-filter>
<action
android:name="android.intent.action.MAIN" />
<category
```

```
android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<receiver android:name=".ReceiveBroadcastActivity">
<intent-filter>
<action
android:name="com.androidunleashed.testingbroadcast"></
action>
</intent-filter>
</receiver>
</application>
</manifest>
```





Using the Notification System:

- The Android notification system provides us with several ways of alerting users.Notification via the Status Bar
- The simplest type of notification is status, which appears in the status bar as an icon along with some optional ticker text.
- Users can pull down the status bar to see the notification list and clear the notification by clicking the Clear button.
- For creating notifications, the following two classes are used:
- **Notification**—The object that defines the information to be displayed, which can be text to be displayed on the status/expanded status bar, an icon displayed with the text, the number of times the notification is triggered, and so on.
- **NotificationManager**—The base object with which notifications are handled. It displays the information encapsulated in the Notification object, which is displayed via the notify() method.

Creating Notifications

The first step is to create a Notification object and configure it by defining notification properties.

The following code shows how to do so:

```
Notification notification = new Notification();
notification.icon = R.drawable.ic_launcher;
notification.tickerText = "There is a new notification";
notification.when = System.currentTimeMillis();
notification.flags |= Notification.FLAG_AUTO_CANCEL;
```

Here, we see that a Notification object called notification is created and thereafter its public members are used to configure it:

• **icon**—Assigns the notification icon.

• **tickerText**—Assigns the small notification text.

• **when**—Assigns the time when the notification occurred. We use the system time to specify the

time the notification occurred.

• **flag—**Assigns the constant that determines the subsequent action when the notification is selected from the notification window.

- We usually assign the FLAG_AUTO_CANCEL constant to this public variable, which specifies that the notification be automatically canceled after it is selected from the notifications.
- We can also assign a notification icon, ticker text, and time of occurrence through the Notification object constructor, as shown here:

```
Notification notification = new
Notification(R.drawable.ic_launcher, "There is a
new notification", System.currentTimeMillis());
```

Creating PendingIntent:

- After receiving the notification, we may choose to take a necessary action. We use the PendingIntent class to switch to the desired Intent when the notification is tapped.
- The PendingIntent class enables us to create Intents that can be triggered by our application when an event occurs.
- The following code creates a PendingIntent called pendIntent:

```
Intent intent = new Intent(getBaseContext(),
TargetActivity.class);
PendingIntent pendIntent =
PendingIntent.getActivity(getBaseContext(), 0, intent, 0);
```

Method	Description
`setSmallIcon()`	Used to supply the small icon resource that is displayed to represent the notification in the status bar. Syntax: `setSmallIcon(int icon)` where the `icon` parameter represents the resource ID of the drawable to be used as the icon of the notification.
`setAutoCancel()`	Used to determine whether we want to make the notification invisible when it is tapped. The Boolean value `true` is supplied to this method to make the notification invisible. Syntax: `setAutoCancel(boolean autoCancel)`
`setTicker()`	Used to supply the ticker text that is displayed in the status bar when the notification arrives. Syntax: `setTicker(CharSequence textMessage)`
`setWhen()`	Used to supply the time of occurrence of the notification. Syntax:`setWhen(long timeOfOccurrence)`
`setContentTitle()`	Used to supply the title of the notification when the status bar is expanded. Syntax: `setContentTitle(CharSequence title)`
`setContentText()`	Used to supply the text of the notification. Syntax: `setContentText(CharSequence text)`
`setContentIntent()`	Used to supply a `PendingIntent` to be sent when the notification is tapped. Syntax: `setContentIntent(PendingIntent intent)`

Code Written into activity_notification_app.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<Button
android:id="@+id/createbutton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Create Notification"
android:layout_gravity="center" />
</LinearLayout>
```

Code Written into target.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<TextView
android:id="@+id/messageview"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="This is target activity"
android:layout_gravity="center" />
</LinearLayout>
```

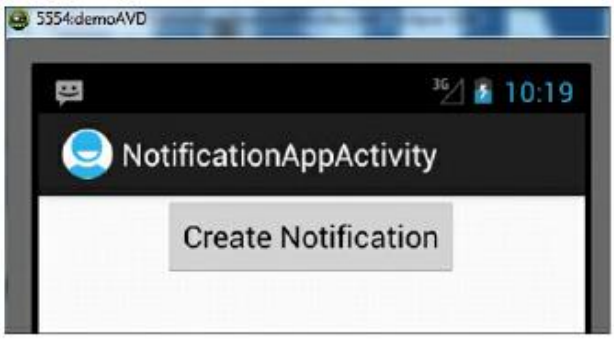
Code Written into TargetActivity.java

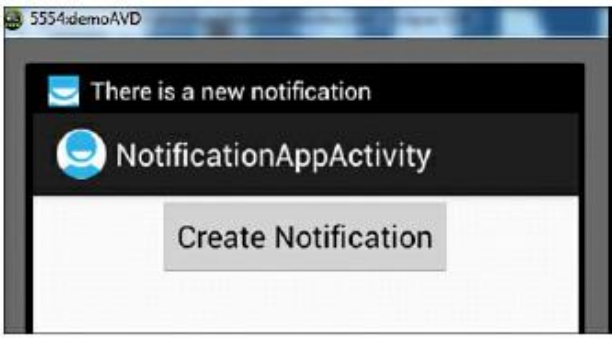
```
package com.androidunleashed.notificationapp;
import android.app.Activity;
import android.os.Bundle;
public class TargetActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.target);
```

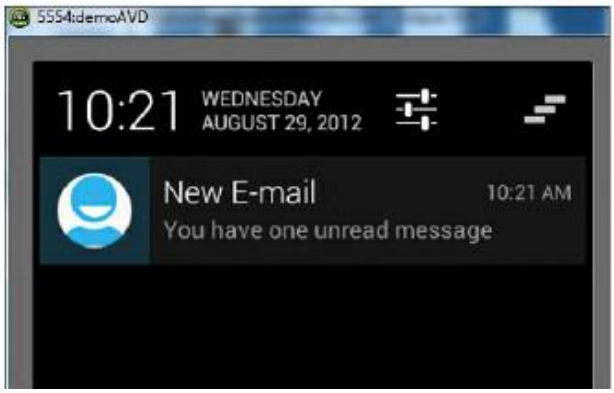
Code Written into NotificationAppActivity.java

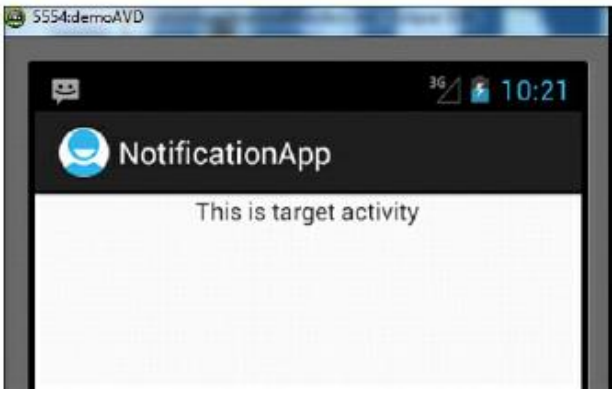
```
package com.androidunleashed.notificationapp;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.app.Notification;
import android.widget.Button;
import android.view.View.OnClickListener;
```

```
public class NotificationAppActivity extends Activity {
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_notification_app);
Button createButton = (Button)
findViewById(R.id.createbutton);
createButton.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View arg0) {
Intent intent = new Intent(getBaseContext(), TargetActivity.class);
PendingIntent pendIntent = PendingIntent. getActivity(getBaseContext(), 0, intent, 0);
NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
Notification notification = new Notification();
Notification.Builder builder = new Notification. Builder(getBaseContext())
.setSmallIcon(R.drawable.ic_launcher)
.setAutoCancel(true)
.setTicker("There is a new notification")
.setWhen(System.currentTimeMillis())
.setContentTitle("New E-mail")
.setContentText("You have one unread message")
.setContentIntent(pendIntent);
notification = builder.getNotification();
notificationManager.notify(0, notification);
}
});
}
}
```







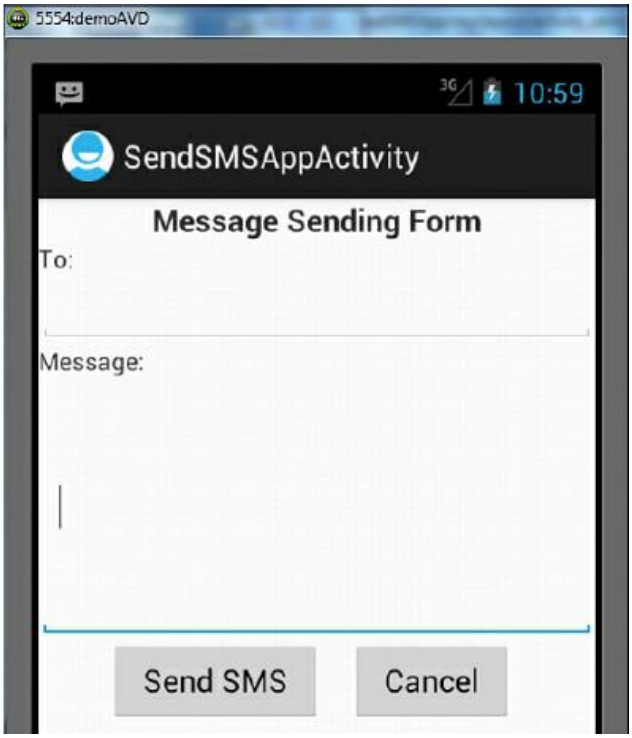


Sending SMS Messages with Java Code:

Code Written into activity_send_smsapp.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<TextView
android:layout_height="wrap_content"
android:text="Message Sending Form"
android:textStyle="bold"
android:textSize="18sp"
android:layout_width="match_parent"
android:gravity="center_horizontal"/>
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="To:" />
<EditText
android:id="@+id/recvr_no"
android:layout_height="wrap_content"
android:layout_width="match_parent" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Message:" />
<EditText
android:id="@+id/txt_msg"
android:layout_width="match_parent"
android:layout_height="150dp" />
<RelativeLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="horizontal">
<Button
android:id="@+id/send_button"
android:text="Send SMS"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="40dip"
android:paddingLeft="20dip"
android:paddingRight="20dip" />
<Button
android:id="@+id/cancel_button"
android:text="Cancel"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:layout_toRightOf="@id/send_button"
android:layout_marginLeft="15dip"
android:paddingLeft="20dip"
android:paddingRight="20dip" />
</RelativeLayout>
</LinearLayout>
```



Getting Permission to Send SMS Messages

To send and receive SMS messages in an application, we need to add permissions to the <manifest> element of our AndroidManifest.xmlfile:

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

Code Written into AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.androidunleashed.sendsmsapp"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk android:minSdkVersion="8"
android:targetSdkVersion="15" />
<uses-permission
android:name="android.permission.SEND_SMS"/>
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".SendSMSAppActivity"
android:label="@string/title_activity_send_smsapp" >
<intent-filter>
```

<action
android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>

Writing Java Code

To add an action to the send_button and cancel_button, we need to write Java code into the SendSMSAppActivity.java activity file.

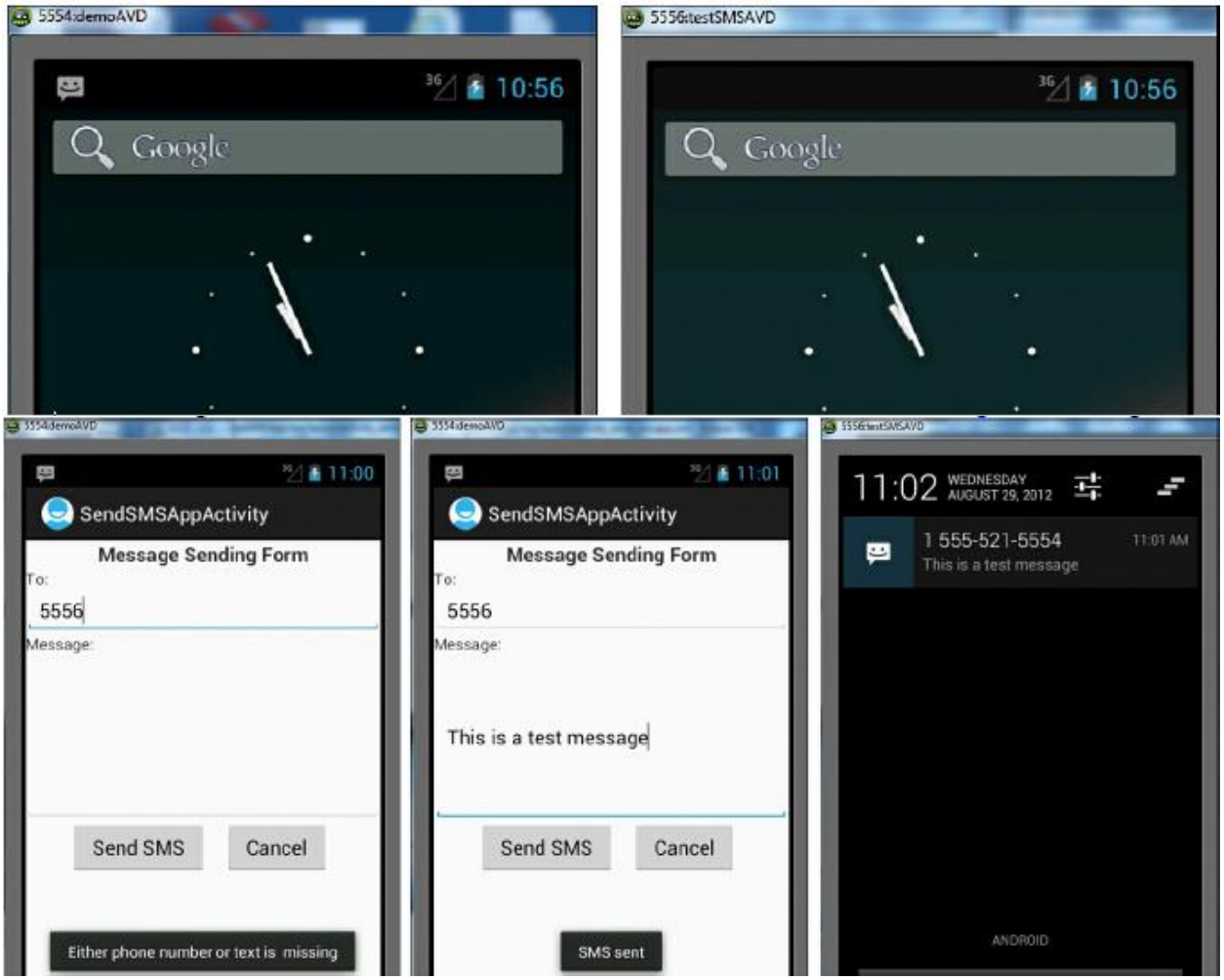
Code Written into SendSMSAppActivity.java

```
package com.androidunleashed.sendsmsapp;
import android.app.Activity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.widget.Button;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Intent;
import android.content.Context;
import android.content.IntentFilter;
public class SendSMSAppActivity extends Activity {
EditText phoneNumber, message;
BroadcastReceiver sentReceiver, deliveredReceiver;
String SENT = "SMS_SENT";
String DELIVERED = "SMS_DELIVERED";
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_send_smsapp);
final PendingIntent sentPendIntent = PendingIntent.getBroadcast(this, 0, new Intent(SENT), 0);
final PendingIntent delivered_pendintnet =PendingIntent.getBroadcast(this, 0, new Intent(DELIVERED), 0);
sentReceiver = new BroadcastReceiver(){
@Override
public void onReceive(Context arg0, Intent arg1) {
switch (getResultCode()) {
case Activity.RESULT_OK:
Toast.makeText(getBaseContext(), "SMS sent", Toast.LENGTH_SHORT).show();
break;
case
SmsManager.RESULT_ERROR_GENERIC_FAILURE:Toast.makeText(getBaseContext(),
"Generic failure", Toast. LENGTH_SHORT).show();
```

```
break;
case SmsManager.RESULT_ERROR_NO_SERVICE: Toast.makeText(getBaseContext(), "No service", Toast.LENGTH_SHORT).show();
break;
case SmsManager.RESULT_ERROR_NULL_PDU: Toast.makeText(getBaseContext(), "Null PDU", Toast.LENGTH_ SHORT).show();
break;
case SmsManager.RESULT_ERROR_RADIO_OFF: Toast.makeText(getBaseContext(), "Radiooff", Toast.LENGTH_ SHORT).show();
break;
}
}
};
deliveredReceiver = new BroadcastReceiver(){
@Override
public void onReceive(Context arg0, Intent arg1) {
switch (getResultCode()) {
case Activity.RESULT_OK: Toast.makeText(getBaseContext(), "SMS successfully delivered", Toast.LENGTH_SHORT).show();
break;
case Activity.RESULT_CANCELED:
Toast.makeText(getBaseContext(), "Failure—SMS not delivered", Toast.LENGTH_SHORT).show();
break;
}
}
};
registerReceiver(sentReceiver, new IntentFilter(SENT));
registerReceiver(deliveredReceiver, new
IntentFilter(DELIVERED));
Button sendBtn = (Button)
this.findViewById(R.id.send_button);
sendBtn.setOnClickListener(new Button.OnClickListener(){
public void onClick(View v) {
phoneNumber = (EditText)
findViewById(R.id.recvr_no);
message = (EditText) findViewById(R.id.txt_msg);
if(phoneNumber.getText().toString().trim().length()
>0 && message.
getText().toString().trim().length() >0) {
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(phoneNumber.getText().toString(),null,
message.getText().toString(),
sentPendIntent, delivered_pendintnet);
}
else {
```

```
Toast.makeText(SendSMSAppActivity.this,"Either phone number or text is missing", Toast.LENGTH_SHORT).show();
}
}
});
Button cancelBtn = (Button)
this.findViewById(R.id.cancel_button);
cancelBtn.setOnClickListener(new Button.OnClickListener(){
public void onClick(View v) {
phoneNumber.setText("");
message.setText("");
}
});
}
}
```

To run the application, two AVD's are required .One AVD is needed to send the message and other AVD is needed to receive the message.



Receiving SMS Messages:

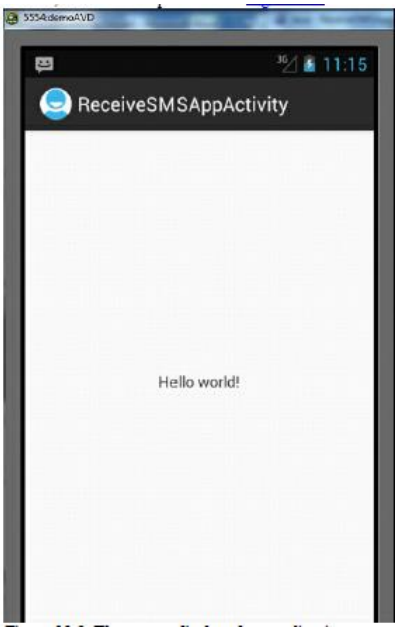
Code Written into ReceiverSMS.java

```
package com.androidunleashed.receivesmsapp;
import android.content.BroadcastReceiver;
import android.content.Intent;
import android.content.Context;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;
public class ReceiverSMS extends BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent) {
Bundle bundle = intent.getExtras();
SmsMessage[] msg = null;
String str = "";
if (bundle != null) {
Object[] pdus = (Object[]) bundle.get("pdus");
msg = new SmsMessage[pdus.length];
for (int i=0; i<msg.length; i++){msg[i] =SmsMessage.createFromPdu((byte[])pdus[i]);
 str +="SMS Received from: " + msg[i].getOriginatingAddress();
str += ":";
str += msg[i].getMessageBody().toString();
str += "\n";
}
Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
}
}
}
```

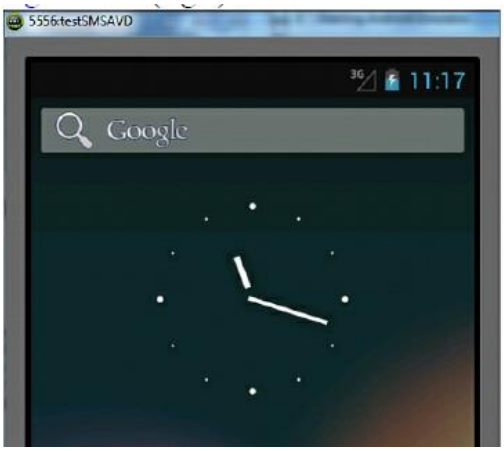
Code Written into AndroidManifest.xml

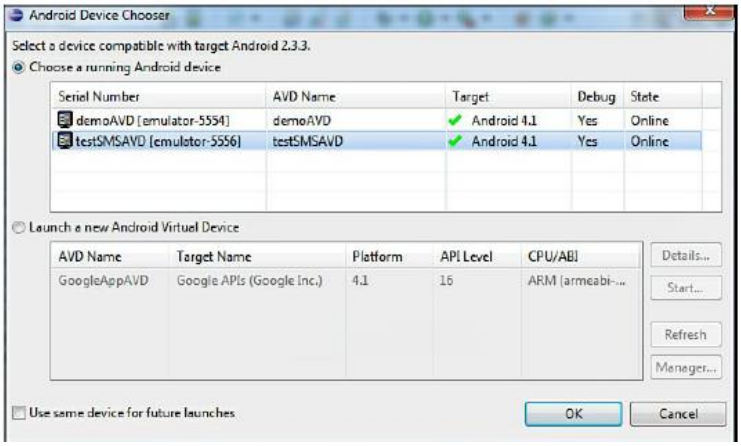
```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.androidunleashed.receivesmsapp"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk android:minSdkVersion="8"
android:targetSdkVersion="15" />
<uses-permission
android:name="android.permission.RECEIVE_SMS" />
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".ReceiveSMSAppActivity"
```

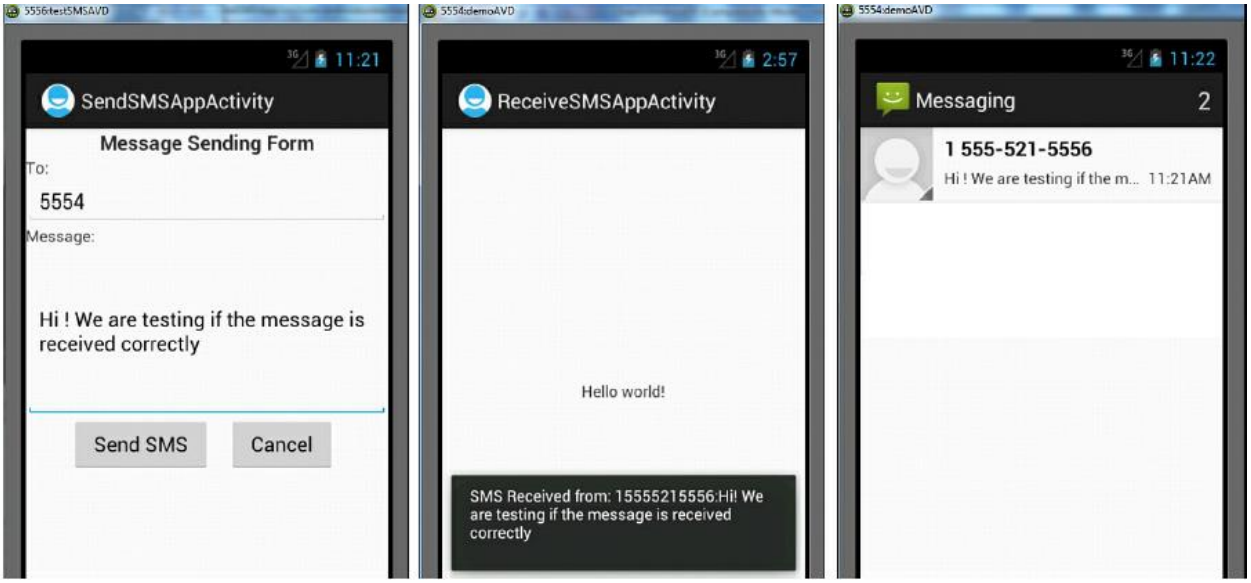
```
android:label="@string/title_activity_receive_smsapp"
>
<intent-filter>
<action
android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<receiver android:name=".ReceiverSMS">
<intent-filter>
<action
android:name="android.provider.Telephony.SMS_RECEIVED" />
</intent-filter>
</receiver>
</application>
</manifest>
```



A new AVD must be used to see how the message is received. And then select the AVD from the list in Android Device chooser dialog box to run the above code.







Sending Email:

- To send an email with Android, we use the following Intent: Intent.ACTION_SEND.
- The Intent.ACTION_SEND calls an existing email client to send an email. So, for sending email through the Android emulator, we need to first configure the email client. If the email client is not configured, the emulator does not respond to the Intent.
- For example, the following statement declares that the text data type will be sent through the Intent:

  emailIntent.setType("plain/text");
- When a user choose the email client to handle the Intent, we call startActivity() with the createChooser() method.

Example:

```
startActivity(Intent.createChooser(emailIntent, "Sending
Email"));
```

To supply data for the email message fields, we set certain standard extras for the Intent. For example, we can set the following:

• EXTRA_EMAIL—Sets the To: address (email address of the receiver)
• EXTRA_CC—Sets the Cc: address (email address of the carbon copy receiver)
• EXTRA_BCC—Sets the Bcc: address (email address of the blind carbon copy receiver)
• EXTRA_SUBJECT—Sets the Subject of the email
• EXTRA_TEXT—Sets the body of the email

To configure the email client of the Android emulator in these steps:

- Start the emulator and then click on the Menu button.
- Click on the System settings option.
- From the Accounts section, click on the Add account button
- From the two options, Corporate and Email, click the Email option.
- we need to enter our existing Email ID and password followed by clicking the Next button.
- Select the required check boxes in Account settings
- Click the Next button to finish
- configuring the email client.

Code Written into activity_send_email_app.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent" >
<TextView
android:id="@+id/email_form"
android:text = "Email Form"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:typeface="serif"
android:textSize="18sp"
android:textStyle="bold"
android:padding="10dip"
android:layout_centerHorizontal="true"/>
<TextView
android:id="@+id/to_addressview"
android:text = "To:"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_margin="10dip"
android:layout_below="@id/email_form" />
<EditText
android:id="@+id/toaddresses"
android:layout_height="wrap_content"
android:layout_width="match_parent"
android:layout_below="@id/email_form"
android:layout_toRightOf="@id/to_addressview"
android:singleLine="true" />
<TextView
android:id="@+id/cc_addressview"
android:text = "Cc:"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/to_addressview"
android:layout_margin="10dip" />
<EditText
android:id="@+id/ccaddresses"
android:layout_height="wrap_content"
android:layout_width="match_parent"
android:singleLine="true"
android:layout_below="@id/toaddresses"
android:layout_toRightOf="@id/cc_addressview" />
<TextView
android:id="@+id/bcc_addressview"
android:text = "Bcc:"
android:layout_width="wrap_content"
```

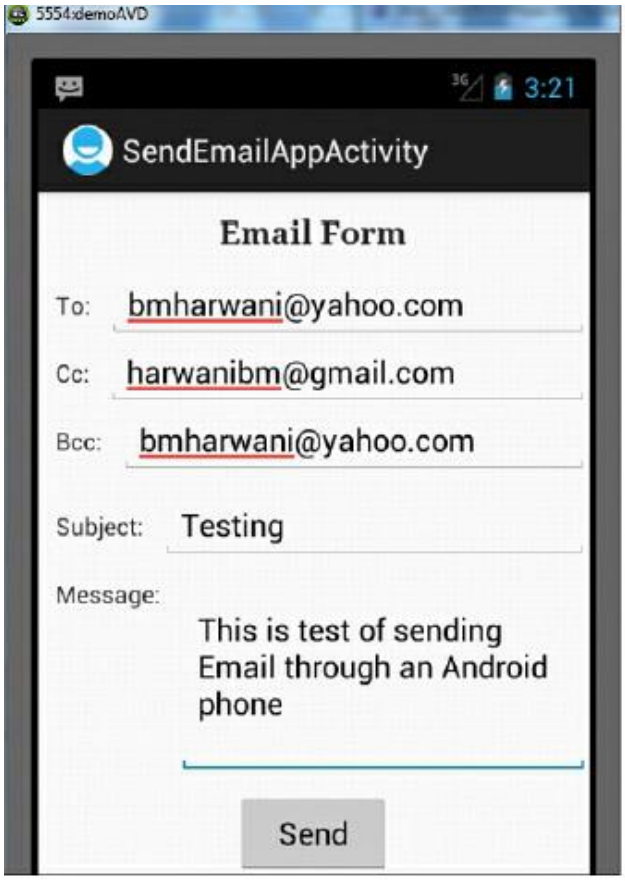
```
android:layout_height="wrap_content"
android:layout_below="@id/cc_addressview"
android:layout_margin="10dip" />
<EditText
android:id="@+id/bccaddresses"
android:layout_height="wrap_content"
android:layout_width="match_parent"
android:singleLine="true"
android:layout_below="@id/ccaddresses"
android:layout_toRightOf="@id/bcc_addressview" />
<TextView
android:id="@+id/subjectview"
android:text = "Subject:"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/bcc_addressview"
android:layout_margin="10dip"
android:paddingTop="10dip"/>
<EditText
android:id="@+id/emailsubject"
android:layout_height="wrap_content"
android:layout_width="match_parent"
android:singleLine="true"
android:layout_below="@id/bccaddresses"
android:layout_toRightOf="@id/subjectview"
android:layout_marginTop="10dip" />
<TextView
android:id="@+id/emailtextview"
android:text = "Message:"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/subjectview"
android:layout_margin="10dip" />
<EditText
android:id="@+id/emailtext"
android:layout_height="wrap_content"
android:layout_width="match_parent"
android:lines="5"
android:layout_below="@id/emailsubject"
android:layout_toRightOf="@id/emailtextview" />
<Button
android:id="@+id/send_button"
android:text="Send"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:paddingLeft="25dip"
```

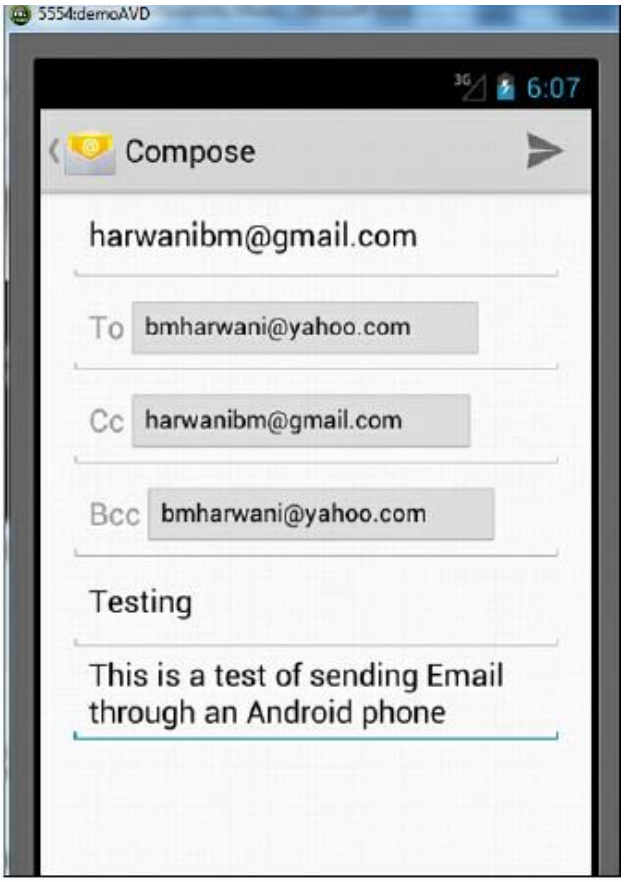
```
android:paddingRight="25dip"
android:layout_marginTop="10dip"
android:layout_below="@id/emailtext" />
</RelativeLayout>
```

Code Written into SendEmailAppActivity.java

```
package com.androidunleashed.sendemailapp;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.content.Intent;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class SendEmailAppActivity extends Activity {
Button sendbutton;
EditText toAddress, ccAddress, bccAddress, subject, emailMessage;
String toAdds, ccAdds, bccAdds;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_send_email_app);
sendbutton=(Button) findViewById(R.id.send_button);
toAddress=(EditText) findViewById(R.id.toaddresses);
ccAddress=(EditText) findViewById(R.id.ccaddresses);
bccAddress=(EditText) findViewById(R.id.bccaddresses);
subject=(EditText) findViewById(R.id.emailsubject);
emailMessage=(EditText) findViewById(R.id.emailtext);
sendbutton.setOnClickListener(new OnClickListener(){
@Override
public void onClick(View v) {
final Intent emailIntent = new
Intent(Intent.ACTION_SEND);
if(toAddress.getText().length() >0) {
toAdds =""+toAddress.getText().toString()+"";
emailIntent.putExtra(Intent.EXTRA_EMAIL, newString[]{ toAdds});
}
if(ccAddress.getText().length() >0) {
ccAdds =""+ccAddress.getText().toString()+"";
emailIntent.putExtra(Intent.EXTRA_CC, new String[]{ ccAdds});
}
if(bccAddress.getText().length() >0) {
bccAdds = ""+bccAddress.getText().toString()+"";
emailIntent.putExtra(Intent.EXTRA_BCC, new String[]{ bccAdds});
}
emailIntent.putExtra(Intent.EXTRA_SUBJECT,subject.getText().
toString());
emailIntent.putExtra(Intent.EXTRA_TEXT,emailMessage.getText());
```

```
emailIntent.setType("plain/text");
startActivity(Intent.createChooser(emailIntent,"Sending Email"));
}
});
}
}
```





Working with the Telephony Manager:

The Android telephony APIs include the Telephony Manager that accesses the telephony services on the device and enables us to

- Provide a user interface for entering or modifying the phone number to dial.
- Implement call handling in the application.
- Register and monitor telephony state changes.
- Get subscriber information.

Making the Outgoing Call

The simplest way to make an outgoing call is to invoke the Dialer application by using the Intent.ACTION_CALL action.

The sample code for doing so follows:

```
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:1111122222"));
startActivity(callIntent);
```

To use this action, the application must request the CALL_PHONE uses-permission by adding the following

statement to the manifest file:

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

Listening for Phone State Changes:

To listen for phone state changes, that is, to see when the phone state is changed from idle to ringing, off-hook, and so on, we have to implement a broadcast receiver on android.intent.action.PHONE_STATE. That is, we need to implement a PhoneStateListener and call the listen() method of the TelephonyManager to receive notification whenever there is a change in the phone state. When a phone state changes, the onCallStateChanged() method of PhoneStateListener is called with the new phone state.

The phone state is represented by the
constants shown here:

• CALL_STATE_IDLE—The phone is in an idle state.
• CALL_STATE_RINGING—A phone call has arrived.
• CALL_STATE_OFFHOOK—The phone is off-hook.

To access the phone state information, the application must have permission for doing so. To ask permission, add following statement to the manifest file:

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

Code Written into activity_phone_call_app.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<Button
android:id="@+id/callbutton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Make Phone Call"
android:layout_gravity="center" />
<TextView
android:id="@+id/messageview"
android:layout_width="wrap_content"
android:layout_height="match_parent"
android:layout_gravity="center" />
</LinearLayout>
```

Code Written into PhoneCallAppActivity.java

```
package com.androidunleashed.phonecallapp;
import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
import android.view.View;
import android.widget.TextView;
import android.widget.Button;
```

```
import android.view.View.OnClickListener;
import android.util.Log;
public class PhoneCallAppActivity extends Activity {
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_phone_call_app);
MyPhoneCallListener phoneListener = new
MyPhoneCallListener();
TelephonyManager telephonyManager = (TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE);
telephonyManager.listen(phoneListener,PhoneStateListener.LISTEN_Button callButton =
(Button) findViewById(R.id.callbutton);
callButton.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View arg0) {
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:1111122222"));
startActivity(callIntent);
}
});
}
class MyPhoneCallListener extends PhoneStateListener {
TextView messageview = (TextView)findViewById(R.id.messageview);
String msg;
@Override
public void onCallStateChanged(int state, String incomingNumber) {
super.onCallStateChanged(state, incomingNumber);
switch(state){
case TelephonyManager.CALL_STATE_IDLE:
msg= "Call state is idle";
Log.d("idle", msg);
break;
case TelephonyManager.CALL_STATE_RINGING:
msg = "Call state is Ringing. Number is "+ incomingNumber;
Log.d("ringing", msg);
break;
case TelephonyManager.CALL_STATE_OFFHOOK:
msg = "Call state is OFFHOOK";
Log.d("offhook", msg);
break;
default:
msg = "Call state is" + state + ". Number is" + incomingNumber;
Log.d("state", msg);
break;
}
messageview.setText(msg);
```

```
}
}
}
```

Code in AndroidManifest.xml

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.androidunleashed.phonecallapp"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk android:minSdkVersion="8"
android:targetSdkVersion="15" />
<uses-permission
android:name="android.permission.CALL_PHONE" />
<uses-permission
android:name="android.permission.READ_PHONE_STATE" />
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".PhoneCallAppActivity"
android:label="@string/title_activity_phone_call_app"
>
<intent-filter>
<action
android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```