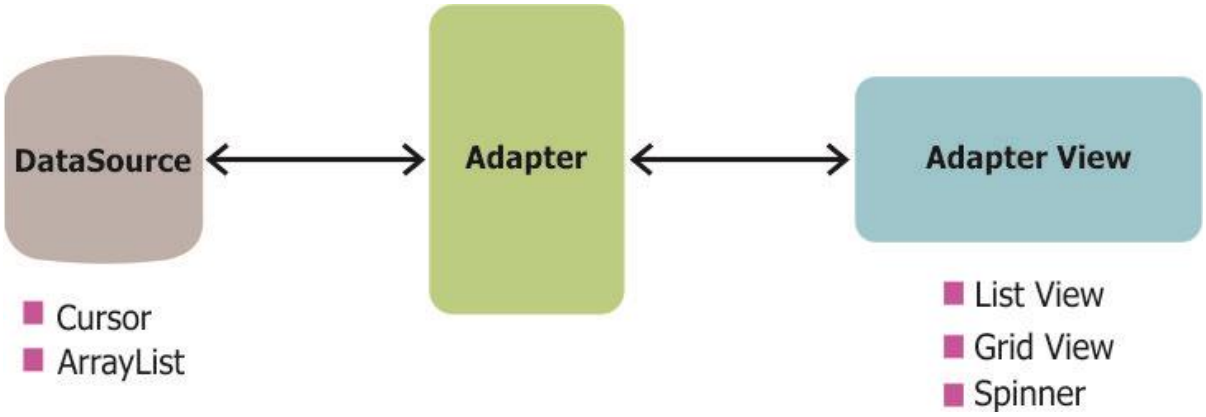
List View

In Android development, any time we want to show a vertical list of scrollable items we will use a `ListView` which has data populated using an `Adapter`. The simplest adapter to use is called an `ArrayAdapter` because the adapter converts an `ArrayList` of objects into `View` items loaded into the `ListView` container.



The `ArrayAdapter` fits in between an `ArrayList` (data source) and the `ListView` (visual representation) and configures two aspects:

- Which array to use as the data source for the list
- How to convert any given item in the array into a corresponding View object

To use a basic `ArrayAdapter`, you just need to initialize the adapter and attach the adapter to the ListView.

Initializing the adapter *(Using constructor with three parameters)*

```
ArrayAdapter<String> itemsAdapter =
    new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
items);
```

- The `ArrayAdapter` requires a declaration of the type of the item to be converted to a `View` (a `String` in this case)
- Accepts three arguments: `context` (activity instance), XML item layout (Layout for the row) , and the array of data.
- we've chosen simple_list_item_1.xml which is a simple `TextView` as the layout for each of the items

OR we can use adapter which takes 4 parameters:

- First parameter - Context

- Second parameter - Layout for the row
- Third parameter - ID of the TextView to which the data is written
- Forth - the Array of data

ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, android.R.id.text1, items);

Attaching the adapter to the ListView

```
ListView listView = (ListView) findViewById(R.id.listViewItems);
listView.setAdapter(itemsAdapter);
```

Handling On Click Events using SetOnItemClickListener

listView.setOnItemClickListener(new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {

//Handle event here

}});

Parameters	
`parent`	`AdapterView`: The AdapterView where the click happened.
`view`	`View`: The view within the AdapterView that was clicked (this will be a view provided by the adapter)
`position`	`int`: The position of the view in the adapter.
`id`	`long`: The row id of the item that was clicked.

`getItemAtPosition(int position)` can be used to get the item value associated with the specified position in the list.

Example: Display the list of geometric shapes in a list view and On clicking the each item in the list, display the position and value using a toast message.

Steps:
1. Create GUI for the application (drag and drop List View) and assign unique Id for the view. (Refer *Activity_List_Items.xml)*
2. *Modify Java file (Refer* listItems.java) and do the following:
    - Access the Id of List View created in UI using findViewById () method ,
    - Create a String array and assign it with different shape names(like circle, square etc.)

- create an array adapter of String Type,with above created String variable.
- Attach the adapter to the ListView object using setAdapter() method
- Handle On Click event using setOnItemClickListener method and Display the value and position of clicked item.

Activity_List_Items.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_list_items"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.student.listviewgenerate.listItems">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="155dp"
        android:id="@+id/shapesList" />

</LinearLayout>
```

listItems.java

```
package com.example.student.listviewgenerate;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class listItems extends AppCompatActivity {
ListView shapeslist;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_items);
        shapeslist=(ListView) findViewById(R.id.shapesList);
        String[] shapes=new
String[]{"Circle","Square","RectAngle","Triangle","SemiCircle","pentagon"};
        ArrayAdapter<String> arrayShapes=new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,shapes);
        shapeslist.setAdapter(arrayShapes);
        shapeslist.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
```

```
                int itemposition=position;
                String itemVal=(String)shapeslist.getItemAtPosition(itemposition);
                Toast.makeText(getApplicationContext(),itemVal +" is clicked at
position" +itemposition ,Toast.LENGTH_SHORT).show();
              }
        });


    }
}
```

Alert Dialog

It is small window that prompts the user to a decision or enter additional information.
To create an alert dialog, Make an object of AlertDialogBuilder which is an inner class of AlertDialog.

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);

We can set the positive (yes) or negative (no) button using the object of the AlertDialogBuilder class and also on click events for those buttons can be handled.(Refer code below)

After creating and setting the dialog builder , you will create an alert dialog by calling the create() method of the builder class
AlertDialog alertDialog = alertDialogBuilder.create();
alertDialog.show(); // this will show the alert box on the screen

Example: Create an application that has a button with text "**Show Alert Box**" and On clicking that button, display alert box titled "Alert Box Display" with message "**"Click yes to exit"**". On clicking "Yes" , It must close the app; On Clicking "No" it must close the alert dialog.(Users should not be allowed to cancel the dialog box when touched outside the window's bounds.)

The methods used are:

setTitle()
setMessage()
setCancelable()

activity_alertbox_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_alertbox_dialog"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.student.alertbox.alertboxDialog">

    <Button
        android:text="Show Alert Box"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="203dp"
        android:id="@+id/bt_alert"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_marginStart="94dp" />
</RelativeLayout>
```

alertboxDialog.Java

```
package com.example.student.alertbox;

import android.content.Context;
import android.content.DialogInterface;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.*;

public class alertboxDialog extends AppCompatActivity {
Button bAlert;
    final Context context=this;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_alertbox_dialog);
        bAlert=(Button) findViewById(R.id.bt_alert);
        bAlert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AlertDialog.Builder alert =new AlertDialog.Builder(context);
                alert.setTitle("Alert Box Display").setMessage("Click yes to
exit").setCancelable(false).setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
```

```
                        alertboxDialog.this.finish(); // close the application
                    }
                }).setNegativeButton("No", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();//close the dialog box
                    }
                });
                AlertDialog alertdia=alert.create();
                alertdia.show();
            }
        });

    }
}
```