S.N.J.B.'s
SHRI H. H. J. B. POLYTECHNIC, CHANDWAD
CLASS TEST I / II (201 - 201)

Course & Semester :	Roll No.:	Date : / /201
Name of Subject :		Marks obtained :
Sign. of Supervisor :	Sign. of Sub. Examiner :	

Q. No. 1	Q. No. 2	Q. No. 3	Total Marks

Chapter 3

Android Appln Component

It is essential building block of Android Appln These component loosely coupled by application Manifest file that describes each component of the appln & how they interact.

1) **Activities :-**
They dictate UI & handle user interaction to smartphone screen

2) **Services :-**
They handle background processing associated with appln

3) **Broadcast Receivers :-**
They handle comm" bet" Android o.s & Appl"

4) **Content Providers :-**
They handle data & db mgt issues

Additional component :-

5) **Fragments :-**
Represents portion of user interface in Activ

6) **Views :-**
UI elements that are drawn on screen including button, list form etc

7) **Intents :-**
msg wiring component together

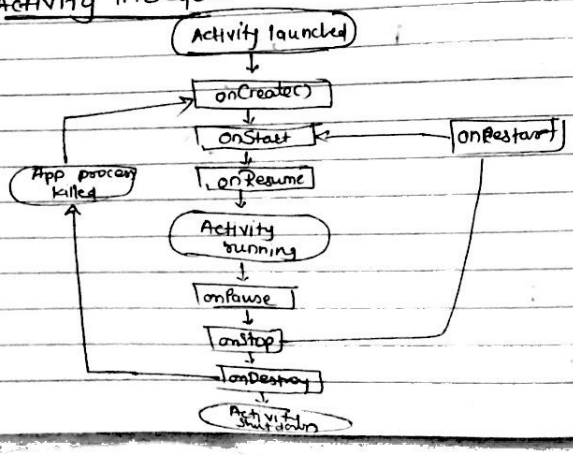
8) Resources :-
external elements such as string, constant, & drawable picture

9) Layouts :-
view hierarchies that control screen format & appearance of the view

10) Manifest
configuration file for appln

1) Activities
- Single screen with user interface
- It performs action on the screen
- activity is the subclass of contextth emeWeappee class
- Andeoid system initiates its program with in activity starting with a call on onCreate() callback method

Activity life cycle



eg:- email appln < list of new mail
< compose mail
< reading mail

Callbacks

1) onCreate()
It is called when the activity is first created

2) onStart()
It is called when the activity become visible to user

3) onResume()
It is called when the user starts interacting with the appln

4) onPause()
The pause activity doesn't user i/p & cannot execute any code & called when the current activity is being paused & previous activity is being resumed

5) onStop()
It is called when activity is no longer visible

6) onDestroy()
It is called before the activity is destroyed by the system

7) onRestart()
It is called when the activity restar or after stopping it

eg:-
public class MainActivity extends Activity
{
}

eg - play music in background while the user in diff appln

2) Services

It is component that runs in background to perform long running operations without needing to interact with the user & its works even if appln is destroyed.

A service can essentially take two states

a) Started

A service is started when appln component such as activity, starts it by calling startService(). once started, a service can run in the background indefinitely, even if the component that started it is destroyed

b) Bound

A service is bound when an appln component binds to it by calling bindService()

A bound service offers a client-server interface that allows components to interact with service, send request, get results & even do so across processes with interprocess comm^n

eg:- public class MyService extends Service
{

}

3) BroadCast Receivers

simply respond to broadcast messages from other appln or from the system itself. These msgs are sometimes called events or intents

There are two imp steps to make BroadcastReceiver works for system broadcasted intents

- Creating
- Registering

a) Creating the BroadcastReceiver

It is implemented as subclass of BroadcastReceiver class & overriding the onReceive() method where each message is received as Intent object parameter.

b) Registering Broadcast Receiver

An appln listens for specific broadcast intents by registering a broadcast receiver in AndroidManifest.xml file.

Android UI controls

- TextView
- EditText
- AutoCompleteTextView
- Button
- ImageButton
- CheckBox
- ToggleButton
- RadioButton
- RadioGroup
- ProgressBar
- Spinner
- TimePicker
- DatePicker

S.N.J.B.'s
SHRI H. H. J. B. POLYTECHNIC, CHANDWAD

CLASS TEST I / II (201 - 201)

Course & Semester :	Roll No.:	Date : / /201
Name of Subject :		Marks obtained :
Sign. of Supervisor :	Sign. of Sub. Examiner :	

Q. No. 1	Q. No. 2	Q. No. 3	Total Marks

Organize resource in Android studio

```
MyProject/
    app/
        manifest/
            AndroidManifest.xml
        java/
            MyActivity.java
        res/
            drawable/
                icon.png
            layout/
                activity-main.xml
                info.xml
            value/
                strings.xml
```

Directory & Resource type

① anim/

XML files that define property animation. They are save in res/anim/ folder & accessed from R.anim class.

② color/

XML files that define state list of colors. They are saved in res/color & access from the R.color class

③ drawable/
image file like .jpg, .gif, .png or XML files that are compiled into bitmaps, state lists, shapes, animati ons. drawable. They are saved in res/drawable & accessed from the R.drawable class.

④ layout/
XML files that defines a UI layout They are saved in res/layout & acce ssed by from R.layout class

⑤ Menu/
XML files that defines appln menus such as option, context or sub menu They are saved in res/menu/ accessed from R.menu class

⑥ raw/
Arbitrary files to save in their raw form. You need to call Resources. openRawResource() with resourceID, which is R.raw.filename to open such raw files.

⑦ Values/
XML files that contain simple values such as string, integers. & color
1) arrays.xml
for resource arrays & accessed from R.array class

2) bools.xml for resource boolean & accessed from R.bool class
3) integers.xml for resource integers & accessed from R.integer class.
4) colors.xml for color values & accessed from R.color class
5) dimens.xml for dimension value & accessed from R.dimen class
6) strings.xml for string values & accessed from R.string class
7) styles.xml for styles & accessed from R.style class

⑧ xml/
Arbitary XML files that can be read at runtime by calling Resources.getXML() you can save various configuration files here which will be used at run time

Andeoid Manifest.xml
which describes the fundamental characteristics of the app & defines each of its components

Build.gradle
This is auto generated file which contains compileSdk version, buildToolsVersion, application id, minsdk version, target sdk version, version code & version name

Types of layout
1. Linear
2. Absolute
3. Table
4. Frame
5. Relative

1) Linear layout → horizontal, vertical — it can arrange ie view in single column or row

Attributes
1) android:id → which uniquely identifies layout

2) andeoid:orientation → This specifies the direction of arrangement
By default → horizontal
horizontal → row
vertical → column

3) andeoid:baselineAligned → This must be boolean value prevents the layout from aligning its childrens baselin

4) andeoid:divider → This is drawable to use as a vertical divider between but you use a color value, in form of "#rgb",

5) android:gravity → This specifies how an object should position its content, on both x & y axis. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc

6) andeoid:weightsum → sum up of child weight

② Relative layout

enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent

Attribute

1) andeoid : id
2) andeoid : gravity
3) andeoid : ignoreGravity
   This indicates what view should not be affected by gravity

By default, all child views are drawn at the top left of the layout. So you must define the position of each view.

1) andeoid : layout_above
   "@[+][package:] type : name"
   eg:-
   andeoid : layout_above "@ + id/name"
   andeoid id

2) andeoid : layout_below

:layout_alignBottom
:layout_alignLeft
:layout_alignParentBottom
:layout_alignParentEnd
:layout_alignParentLeft
——=—— Right
——=—— Start
——=—— Top

, Right, Start, Top, below

True or false value

andeoid : layout_below
andeoid : layout_centerHorizontal
——=—— InParent
——=—— Vertical
} true or false

andeoid : layout_toLeftOf
——=—— Right
——=—— Start
}

③ Absolute Layout

specify exact locations (x/y co-ordinates) of its children.

It is less flexible & harder to maintain than other types of layouts without absolute positioning

Attributes

andeoid : id
——=—— : layout_x
——=—— : layout_y
} specifies x/y coordinate of the view

Constructors

AbsoluteLayout(Context context)

④ Table Layout

arranged groups of views into rows & columns to use <TableRow> element to build a row in table. each row has zero or more cells, each cell can hold one view object

Table layout container don't display borders lines for their rows, columns or cells

Attributes

① andeoid: id

② andeoid: collapseColumns

zero based index of the columns to collapse. The column indicies must be separated by comma

③ andeoid: ShrinkColumns

④ andeoid: StretchColumns

<TableLayout>

Row 1		
Row 2 column1	Row 2 column2	Row2 column3
Row3 column1	Row3 column2	

</TableLayout>

⑤ Frame layout

It is placeholder on screen that you can use to display a single view. ~~It can be declared~~

It is designed to block out area on the screen to display single item

it is used to hold a single child view bcz it can be difficult to organize child views in way that's scalable to different screen sizes without the children overlapping each other

Add multiple children to frame layout & control their position within the frame layout by assigning gravity to each child using the andeoid: layout_gravity

Attributes

1) android: id

2) android: foreground

This defines the drawable to draw over the content & possible values may be color value

3) andeoid: foregroundGravity

define the gravity to apply the foreground drawable. The gravity defaults to fill. Possible values: top, bottom, left, right etc

4) andeoid: measureAllChildren

determines whether to measure all children or just those in the visible or invisible state when measuring. defaults to false

* setContentView(R.layout.activity_main);
  - give info about our layout resource
Here, our layout resources are defined in activity_main.xml file

* <TextView
    android:text="@string/hello_world" />
  provides info about textview, msg. The value of attributes hello world is defined in String.xml file

string.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="hello_world">Hello World!</string>
</resources>

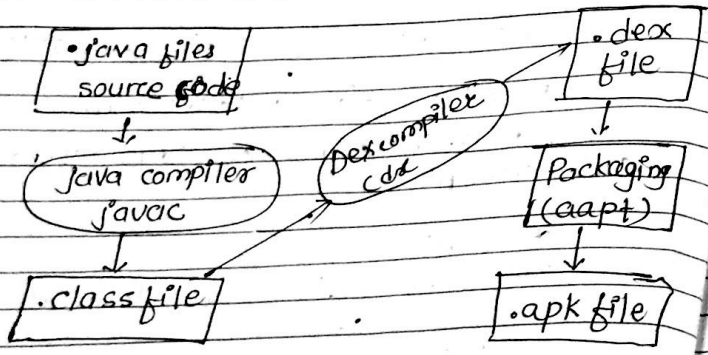
* Generated R.java file
  It is auto generated file that contains ID's for all the resources of res directory It is generated by aapt (Android asset packaging tool).
  Whenever you create any component on activity_main, a corresponding ID is created

In R.java file which can be used in Java source file later

* DVM (Dalvik Virtual Machine)

Dalvik - name of town in Iceland

dex compiler converts class files into .dex file that run on the DVM multiple class files are converted into one dex file.



Android Hide title Bar & Full screen

```
MainActivity class
{
    onCreate()
{ super.onCreate(saveInstanceState);
  requestWindowFeature(Window.FEATURE_
① -NO_TITLE);
                    //will hide the title
② getSupportActionBar().hide(); // hide
                    the title bar
③ this.getWindow().setFlags(WindowManag
er.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULL
SCREEN);  //enable full screen

  setContentView(R.layout.activity_main);
}
}
```

① method of Activity must be called to hide the title, but is must be coded before setContentView method

② It is used to retrieve the instance of ActionBar class calling the hide() method of ActionBar class hides title bar

③ getFlags() method of window class is used to display content in fullscreen mode

Andeoid screen orientation

- It is attribute of activity element
- It can be portrait, landscape, sensor, unspecified etc
- You need to define it in Andeoid Manifest.xml file

eg:-

```
<activity andeoid:name=" example.MainActivity"
android:screenOrientation="portrait">
</activity>
```

① unspecified → default value system chooses the orientation

② portrait → tallee not widee

③ landscape → widee not tallee

④ sensor → determined by device orienation sensor

AndroidManifest.xml

```
<activity android:name =" example.mainActivity"
android: screenOrientation ="portrait">
</activity>
<activity android:name =" example.SecondActivity"
android:screenOrientation =" landscape">
</Activity>
```

activity class

```
{ onCreate()
Button b1 = (Button) findViewById( R.id.button1);
{
onClick(View v)
{
Intent i = new Intent(MainActivity.this, SecondActivity.class);
startActivity(i);
}
}
```

second activity class

```
setContentView (R.layout.activity_second);
```

Date :

Page No. 120

① Linear Layout

Empty activity

xml file

< android.support.constraint.ConstraintLayout

by default LinearLayout (replace) / RelativeLayout

horizontal

android:orientation = "vertical"

android:weightSum="sum of child"

v → Left & right

h → bottom

to button

"bottom"

android:layout_gravity = "center"

android:layout_width & height = "wrap_content"

match_parent. full size

small size

android:weight = "3"

"1"

(button size)

</LinearLayout>

② Relative Layout

button/TextView

android:textColor = "@color/colorPrimaryDark" or "#ffff"

textStyle = "Italic" bold, normal

textSize = "30dp"

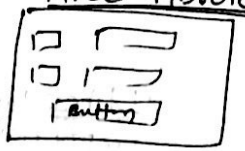
TextView → _centerVertical = "true"

_layout_below = "@source id";

MainActivity.java

```
public class MainActivity extends Activity
{   Button b;
    TextView t;
        onCreate()
    {   b=(Button) findViewById(R.id.button);
        t=(TextView) findViewById(R.id.TextView);
        b.setOnClickListener(new View.onClick
            Listener())
        {
            public void onClick(View v)
            {
                t.setText("This is relative layout");
            }
        }
    }
```

Ride NDroid



MainActivity.java

```
class
{
    Button b;
    EditText e1,e2;
    TextView t1,t2;
    onCreate()
    {  b=(Button) findViewById(R.id.button);
       e1=(EditText) ——————— editText1);
       e2=(TextView ——————— editText1);
       t1=(TextView ——————— TextView);
       t2=( ——————— TextView);
       b.setOnClickListener(View v)
       {     onClick(View v)
             {
                if(e1.getText().toString.equals(
                  "admin")&& e2.getText().
                toString.equals("123"))
                {
                   Toast.makeText(getAppli:
                   cation(context(), text:"
                SUCCESS", Toast.LENTH_LONG).show();
                }
                else
                {
                "unsuccess"
                }
             }
       }
}
```

EditText → Hint properties

Date :

Page No. 121

⑤ TableLayout

<TableLayout → stretchcolumn="1" → column को attribute जोड़ा होता

<TableRow

<Button → android:layout_span="2" (by default 1 — 2 column ki size आयेगी)

/Button>

<Button

/Button>

</TableRow>

-layout_column="2" → यहाँ से start करना है

④ Absolute Layout

x-axis

y-axis

layout_x="200px"

layout_y="400px"

⑤ frame Layout → view को effect नहीं होता.

<frameLayout

<Button>

<Button>

-marginTop="100dp"

-marginLeft="100dp"