Intesactivity Tools

Intents and filters :-

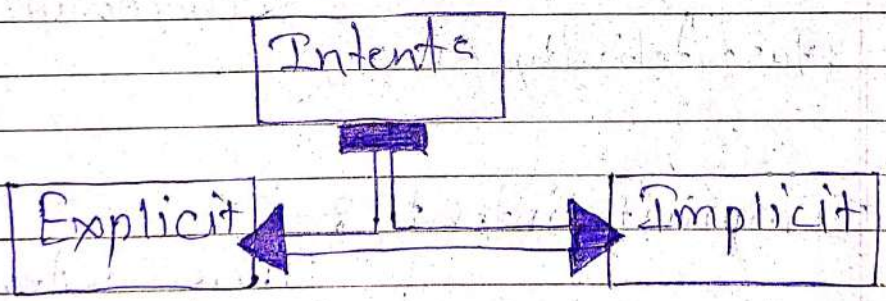
Intent object :-

An Intent object is a bundle of information which is used by the component that receive the intent as well as information used by android

An Intents object can contain following components.

- Action
- Data
- category
- Extra
- flags

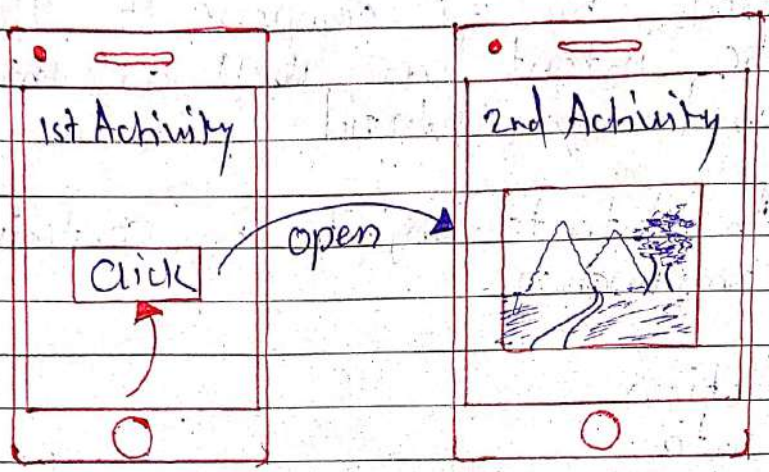
Types of Intents :-



Explicit Intent :-

Explicit intent is going to be connected internal world of application, suppose if

you want to connect one activity to another activity, we can do this quote by explicit intent, below the image is connecting first activity to second activity by clicking button.



These intents designate the target components by it's name and they are tipically used by application

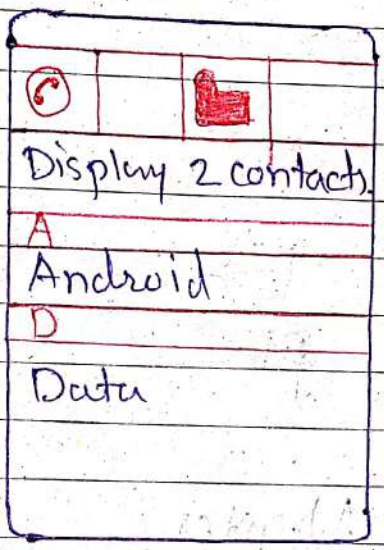
```
Intent i = new Intent(FirstActivity.this,
                      SecondActivity.class);
startActivity(i);
```

Implicit intent :- These intent do not name a target and field for the component name is left blank.

Date: / /

Implicit intent are often used to activate components in other applications.

```
Intent seeall = new Intent();
seeall.setAction(Candroid.content.
                 Intent.ACTION_VIEW);
seeall.setData(ContactsContract.Contacts.
                 CONTENT_URI);
startActivity(seeall);
```



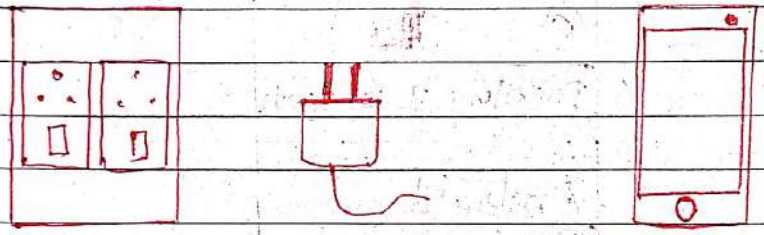
The targetcomponents which receives the intent can use the getExtra() method to get the extra data sent by the source component.

```
Bundle extras = getIntent().getExtras();
String v1 = extras.getString("key1");
String v2 = extras.getString("key2");
```

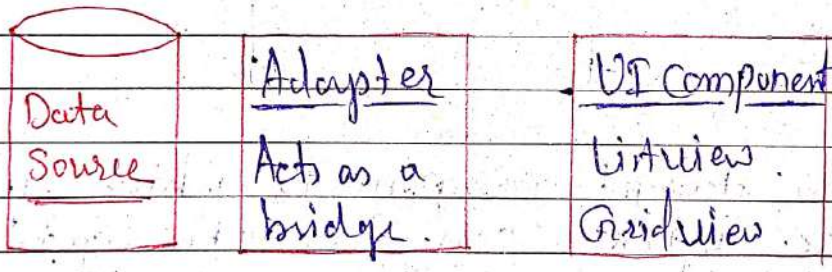
Page No.
Date: / /

Adapters :-

In Android adapter is a bridge between UI component and data source that helps us to fill data in UI components. It holds the data and send the data to an Adapter view then view can take the data from the adapter view and show the data on different view.



Socket Pin Mobile



Adapter in Android

There are some commonly used Adapter in android used to fill the data in the UI components.

- BaseAdapter
- ArrayAdapter
- CustomArrayAdapter
- SimpleAdapter
- CustomSimpleAdapter

BaseAdapter

```
public class Custom extends BaseAdapter
{

}
```

ArrayAdapter

```
ArrayAdapter (context context, int
        resource, int textViewResourceId
            T[] objects)
```

Adapter Example

- Create a new project and name it
- main.xml file in that we implement the user interface code.

```
<Relativelayout>

<ListView>

</ListView>

</Relativelayout>
```

- open the package and file that name is MainActivity.java file

```
public class MainActivity extends
        AppCompactActivity {

ListView simpleListview;

String fruitName[] = {"Apple, Mango"};

int[] fruitImage = {R.drawable.apple,
                    R.drawable.Mango};

void onCreate()
{

simpleListview = (ListView) findViewById
        (R.id.simpleview);

ArrayList<HashMap<String,String>>
    arrayList = new ArrayList<>();

for(int i=0; i<fruitName.length; i++)
{
 HashMap<String, String> hashmap =
    new HashMap<>();
 hashmap.put("name", fruitName[i]);
 hashmap.put("image", fruitImage[i]+"");
 arrayList.add(hashmap);
}
```

String[] from = {"name", "image"};

int[] to = {R.id.textview, R.id.imageView};

SimpleAdapter SA = new SimpleAdapter

(this, arrayList, R.layout.list_view_items,

from, to);

~~SimpleListview~~

~~SA.setAdapter(SA);~~

SimpleListview.setAdapter(SA);

}

- Add the listview item in .xml file.
- Run the android application.

Dialogs :-

Android alert dialog can be used to display the dialog message with OK and cancel buttons. It can be used to interrupt and the user about her choice or to continue.

Android AlertDialog is the subclass of Dialog class.

- activity_main.xml

```
<RelativeLayout>
<Button
android:id="@+id/button"
/>
</RelativeLayout>
```

- String.xml

```
<resources>
<String name="name">Alert Dialog
</string>
<string name="Message">Welcome
</string>
<String name="Title">Alert</string>
</resources>
```

- Activity class

MainActivity.java

```
public class MainActivity extends
        AppCompatActivity
{
    Button bt;
    AlertDialog.Builder builder;
```

```
void onCreate()
{
  bt = (Button) findViewById(R.id.button);
  builder = new AlertDialog.Builder(this);
  bt.setOnClickListener(new View.OnClickListener
  {
    void onClick(View v) {
  builder.setMessage(R.string.dialog_
                Message).setTitle
    (R.string.dialog_title);
  builder.setMessage("Do you want to
        close the application");
  .setCancelable(false)
  .setPositiveButton("yes", new DialogInterface.OnClickListener() {
    onClick()
    {
      finish();
    // Toast use
    }
  }
  AlertDialog alert = builder.create();
  alert.setTitle("AlertDialogExample");
  alert.show();
```

Page No.
Date: / /

Menu :-

There are 3 types menu in android

1) Option Menu
2) Context Menu
3) Pop-up Menu

Option Menu :-

The option menu is the primary collection of menu item for an activity.

Context Menu :-

A context menu is floating menu that appears when the user perform a long click on an element. It provide the action that affect the selected content or context.

Pop-up Menu :-

A popup menu display a list of item in a vertical list that is anchored (sticked) to the view that invoked the menu

How to Create Menu :-

Android provide the standard XML format to define menu item

Page No.
Date: / /

Using the Menu resources it's good practice for few reasons.

- It's easy to visualize the menu structure in XML
- It separates the content for the menu from your application behavior code
- It allow to create alternative menu configuration for diff platform.

How to create menu in XML file

- Right click on res --> new --> then click on Android Resource Directory.
- New window appear. Here choose the menu from resource type drop down list click on ok button.
- After Right click on Menu --> New --> Menu Resource file
- Assign the name and click on ok

Menu file. XML

```
<Menu>
  <item "android:id=@id/a" />
</Menu>
```

Page No.
Date: / /

Making option Menu.

```
public boolean onCreateOptionMenu
                    (Menu menu)
{
MenuInflater inflater = getMenuInflater
                                ();
 inflater.inflate(R.menu.menu_file,
                    menu);
 return true;
}
```

Handling click event :-

onoptionItemselected() Method is used for Handling event.

```
public boolean onOptionItemSelected (
                MenuItem item ){
 switch (item.getItem()) {
   case R.id.P1 :
     return true;
 }
```

Context Menu :-

Register the view to which the context

menu should be associated by calling RegisterForContextMenu()

```
void onCreateContextMenu (ContextMenu
                          View v,
                          ContextMenuInfo menuInfo)
{
super.onCreateContextMenu (menu, v,
                           menuInfo)
MenuInflater in = getMenuInflater();
in.inflate (R.menu.menufile, menu);
}
```

Pop-up Menu :-

If you have define menu file xml in xml.

- Make the object of pop-up Menu whose constructor take current application context
- use the Inflater inflate your menu into the Menu object return by PopupMenu.getMenu()
- call PopupMenu.show()

```
Void Pop(View v) {
PopupMenu p = new PopupMenu(this, v);
MenuInflater in = getMenuInflater();
```

inflate.inflate(R.menu.menu_file,
popup.getMenu());
popup.show();
}

Notification :-

Notification is a kind of message, status of application that is visible / available in android

Different types of notification as follow.

- Status bar Notification
- Notification drawer notification
- Heads up Notification
- Lock screen Notification

+ Create a New project

2 Working with activity.xml file

<RelativeLayout>

<Button android:id="@+id/bt"
/>

</RelativeLayout>

Page No.
Date: / /

2 Working with MainActivity.java file.

notificationManager = getsystemservices
(Context.NOTIFICATION_SERVICE) as
NotificationManager

notificationManager.notify(12, builder.
build())

Android Notification Properties.

setSmallIcon()
SetContentTitle()
SetContentText()
setAutoCancel()
SetPriority().

Notification.compat.Builder B =
new Notification.compat.Builder(this)

This ~~Bu~~ Class is used for creating
Notification in android.