getResources().getStringArray(R.array.arr); - get variable from main folders

<uses-permission android:name="android.permission.READ\_CONTACTS"></uses-permission> - set permissions

<android:permission="android.permission.READ\_Contacts"> - set permissions in activity

You can create interface in frame, then implement it in main activity and override methods. Then override method **onAttach** in frame and create there example of that interface. If you use functions in this variable they will process in main activity

**.setVisibility(View.GONE)** – don’t display

Write strings in special file

TextView view = findViewById(R.id.second); - get variable

Log.d("title", "message"); - show logs

**Toast.makeText(getActivity(),"**string**",Toast.LENGTH\_SHORT).show();** - alert on the bottom of the screen

**Actions**

To create action on button you have to create new activity **new->create activity->empty activity** set function name in tool window (onclick) and create this action in class, where button is placed, then write in function:

**In manifest:**

<category android:name="android.intent.category.LAUNCHER" />

In main:

public void main2(View view){ - create function and get this parameter  
 Intent intent = new Intent(this, Main2Activity.class); - create intent, second variable is name of action class + Action   
 startActivity(intent); - set new window   
}

to handle long click events implement View.onLongClickListener

Exchange info between actions

EditText text = findViewById(R.id.ft); - get variable by id  
String str = text.getText().toString(); - get text  
  
Intent intent = new Intent(this, Main2Activity.class); - get intent  
intent.putExtra(“var1”, str); - record as map  
startActivity(intent); - start activity

Second class

Intent intent = getIntent(); - get current intent   
  
String str = intent.getStringExtra(“var1”); - get str by key  
TextView view = findViewById(R.id.second); - get text variable by id (second)  
view.setText(str); - set text

Create shortcut to the selected class

In file AndroidManifest in needed action add parameter android:parentActivityName=”.name of class”, this will add an arrow on the top bar, press it and you will go to the action of the pasted class

**Preferences**

They are being saved even if application is closed

Create main activity and class which will process information. Import **android.content.SharedPreferences;**

SharedPreferences sharedPreferences = context.getSharedPreferences("name", Context.MODE\_PRIVATE); - create preference, to get context use:

getApplicationContext()

sharedPreferences.edit().putBoolean("key", value).commit(); - set variable (int, Boolean, String, …)

sharedPreferences.getBoolean("name", default); - get valuable or default one

**Run one application from another**

**Creating application, which can be used by another**

In manifest in needed activity add:

<intent-filter>  
 <action android:name="android.intent.action.SEND"></action> - set kind of action  
 <data android:mimeType="text/plain"></data>  
 <category android:name="android.intent.category.DEFAULT"></category>  
</intent-filter>

In another application call activity

They have to have the same permissions (for example permission to read contacts)

Intent intent = new Intent();  
intent.setAction(Intent.ACTION\_SEND); - set action as well as in needed application   
intent.setType("text/plain");  
intent.putExtra("mes","second message"); - add parameters  
startActivity(intent); - start activity (it will show window with available programs)

**Check network and its type**

In manifest:

<uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>

In main:

**ConnectivityManager** connectivityManager**;**

Set in onCreate function:  
 connectivityManager = (ConnectivityManager)getSystemService(CONNECTIVITY\_SERVICE);

Check internet:  
 NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();  
 if (networkInfo!=null&&networkInfo.isConnected()){ - check for network   
  
Check type:  
 NetworkInfo wifi = connectivityManager.getNetworkInfo(ConnectivityManager.TYPE\_WIFI);  
 NetworkInfo mob = connectivityManager.getNetworkInfo(ConnectivityManager.TYPE\_MOBILE);  
 if (wifi.isConnected()){ - check if it is wifi  
 }else if(mob.isConnected()){ - check if it is mobile internet

**Make color of bar with notifications invisible**

if (Build.VERSION.SDK\_INT >= 19){  
 getWindow().addFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_STATUS);  
}else{  
 getWindow().clearFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_STATUS);  
}