Watch this Video

https://www.youtube.com/watch?v=d0ddHJseObw

Unite 3

Files, Saving State, and Preferences: Saving Simple Application Data, Creating and Saving Shared Preferences, Retrieving Shared references, Creating a Settings Activity for the Earthquake Viewer, Introducing the Preference Framework and Preferences Activity, Creating a Standard Preference Activity for the Earthquake Viewer, Including Static Files as Resources, Working with the File System: File Management Tools.

DATABASES AND CONTENT PROVIDERS: Introducing Android Databases, Introducing SQLite, Content Values and Cursers, Working with SQLite Databases, Creating a Content Providers, Using Content Providers, Creating a Searchable Earthquake Content Provider, Native Android Content Providers.

SAVING SIMPLE APPLICATION DATA

SAVING SIMPLE APPLICATION DATA

The data-persistence techniques in Android provide options for balancing speed, efficiency, and robustness.

- **Shared Preferences** — When storing UI state, user preferences, or application settings, you want a lightweight mechanism to store a known set of values. Shared Preferences let you save groups of name/value pairs of primitive data as named preferences.
- **Saved application UI state** — Activities and Fragments include specialized event handlers to record the current UI state when your application is moved to the background.
- **Files** — It's not pretty, but sometimes writing to and reading from fi les is the only way to go. Android lets you create and load fi les on the device's internal or external media, providing support for temporary caches and storing fi les in publicly accessible folders.

CREATING AND SAVING SHARED PREFERENCES

- Using the SharedPreferences class, you can create named maps of name/value pairs that can be persisted across sessions and shared among application components running within the same application sandbox.
- To create or modify a Shared Preference, call getSharedPreferences on the current Context, passing in the name of the Shared Preference to change.

```
SharedPreferences mySharedPreferences = getSharedPreferences(MY_PREFS,
Activity.MODE_PRIVATE);
```

- Shared Preferences are stored within the application's sandbox, so they can be shared between an application's components but aren't available to other applications.
- To modify a Shared Preference, use the SharedPreferences.Editor class. Get the Editor object by calling edit on the Shared Preferences object you want to change.

```
SharedPreferences.Editor editor = mySharedPreferences.edit();
```

Use the put<type> methods to insert or update the values associated with the specifi ed name:

- // Store new primitive types in the shared preferences object.
- editor.putBoolean("isTrue", true);
- editor.putFloat("lastFloat", 1f);
- editor.putInt("wholeNumber", 2);
- editor.putLong("aNumber", 3l);
- editor.putString("textEntryValue", "Not Empty");

To save edits, call apply or commit on the Editor object to save the changes asynchronously or synchronously, respectively.

```
// Commit the changes.
editor.apply();
```

RETRIEVING SHARED PREFERENCES

- Accessing Shared Preferences, like editing and saving them, is done using the getSharedPreferences method.
- Use the type-safe get<type> methods to extract saved values. Each getter takes a key and a default value (used when no value has yet been saved for that key.)

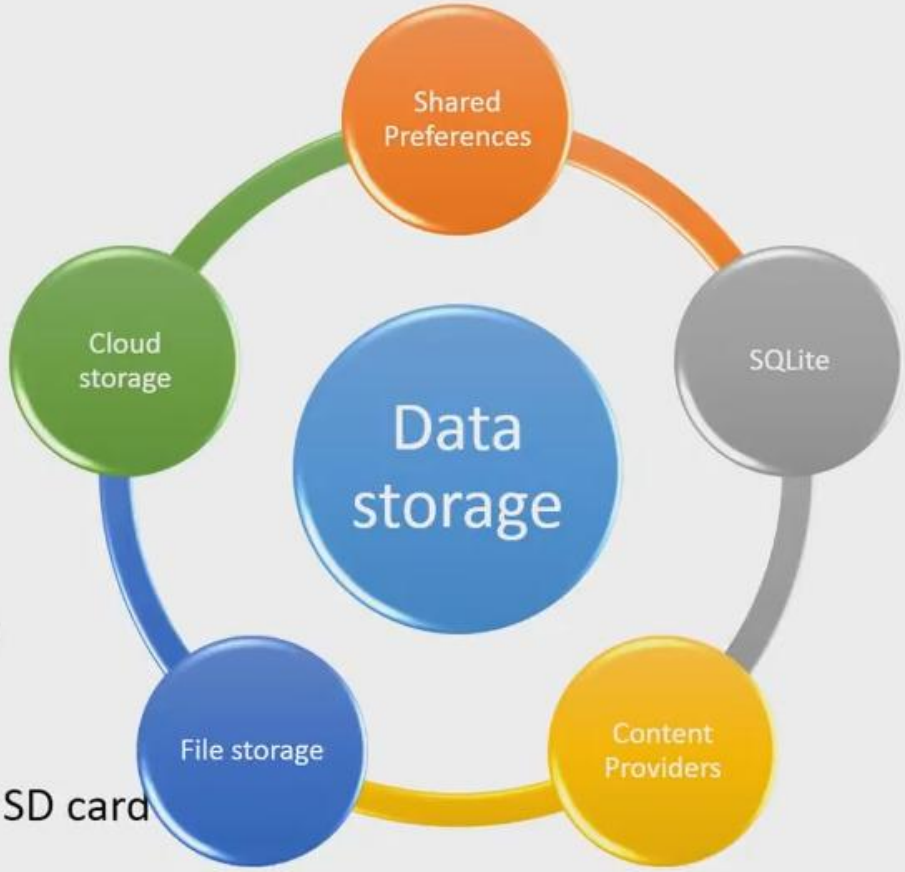
- // Retrieve the saved values.
- boolean isTrue = mySharedPreferences.getBoolean("isTrue", false);
- float lastFloat = mySharedPreferences.getFloat("lastFloat", 0f);
- int wholeNumber = mySharedPreferences.getInt("wholeNumber", 1);
- long aNumber = mySharedPreferences.getLong("aNumber", 0);
- String stringPreference =
- mySharedPreferences.getString("textEntryValue", "");

You can return a map of all the available Shared Preferences keys values by calling getAll, and check for the existence of a particular key by calling the contains method.

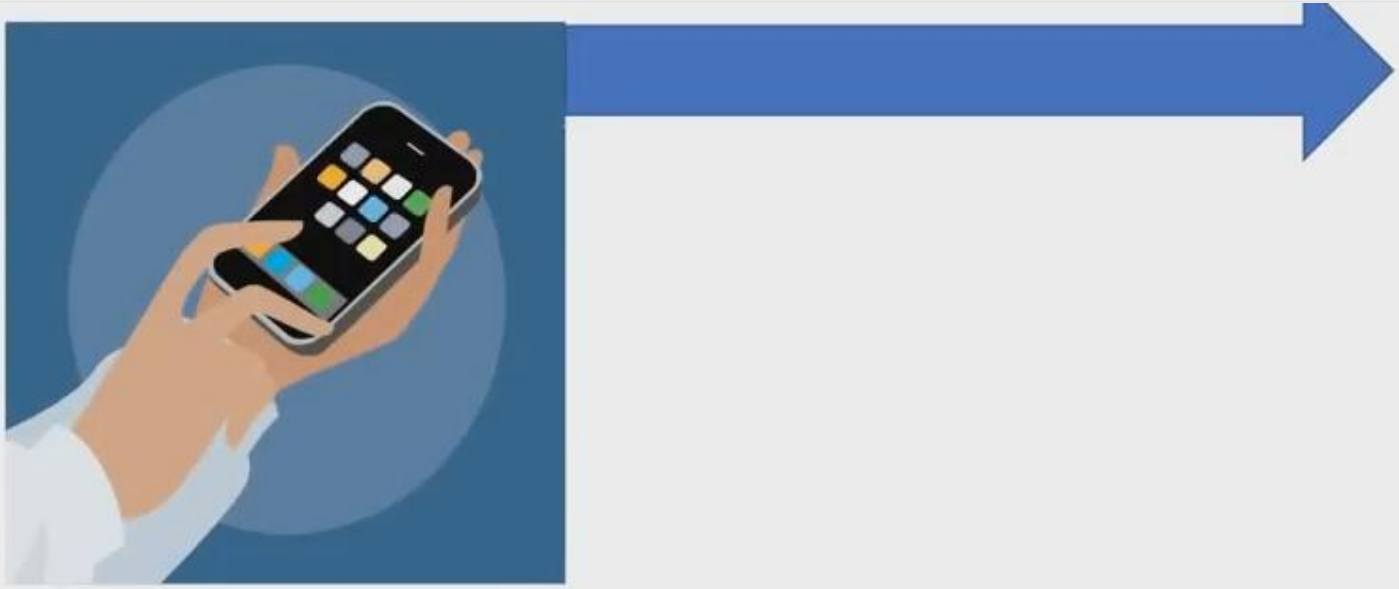
- Map<String, ?> allPreferences = mySharedPreferences.getAll();
- boolean containsLastFloat = mySharedPreferences.contains("lastFloat");

Android Data Storage Options

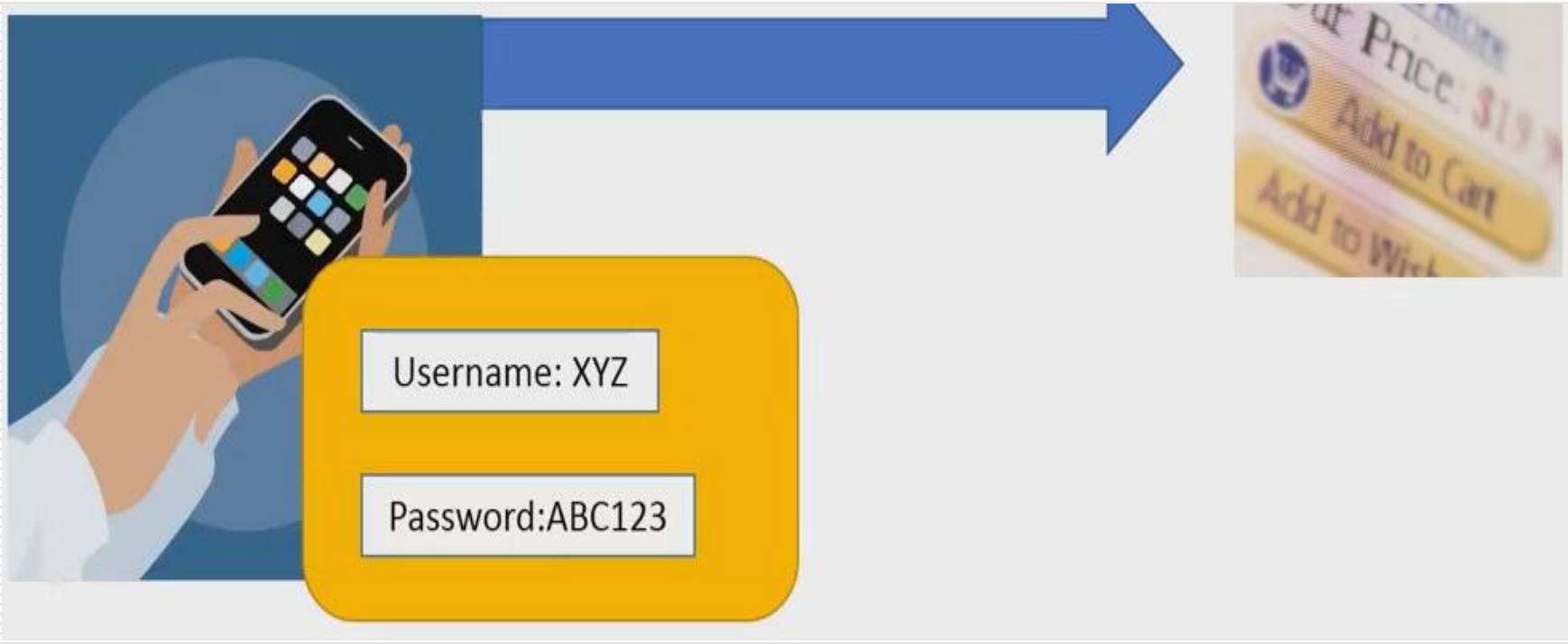
- **SharedPreferences**
  - Store data in key value pairs
- **SQLite**
  - Store structured private app data
- **ContentProvider**
  - Store structured or semi-structured data with user configurable data access
- **File Storage**
  - Store raw files on the phone memory or SD card
- **Cloud storage**
  - Use 3rd party or custom made external data storage APIs



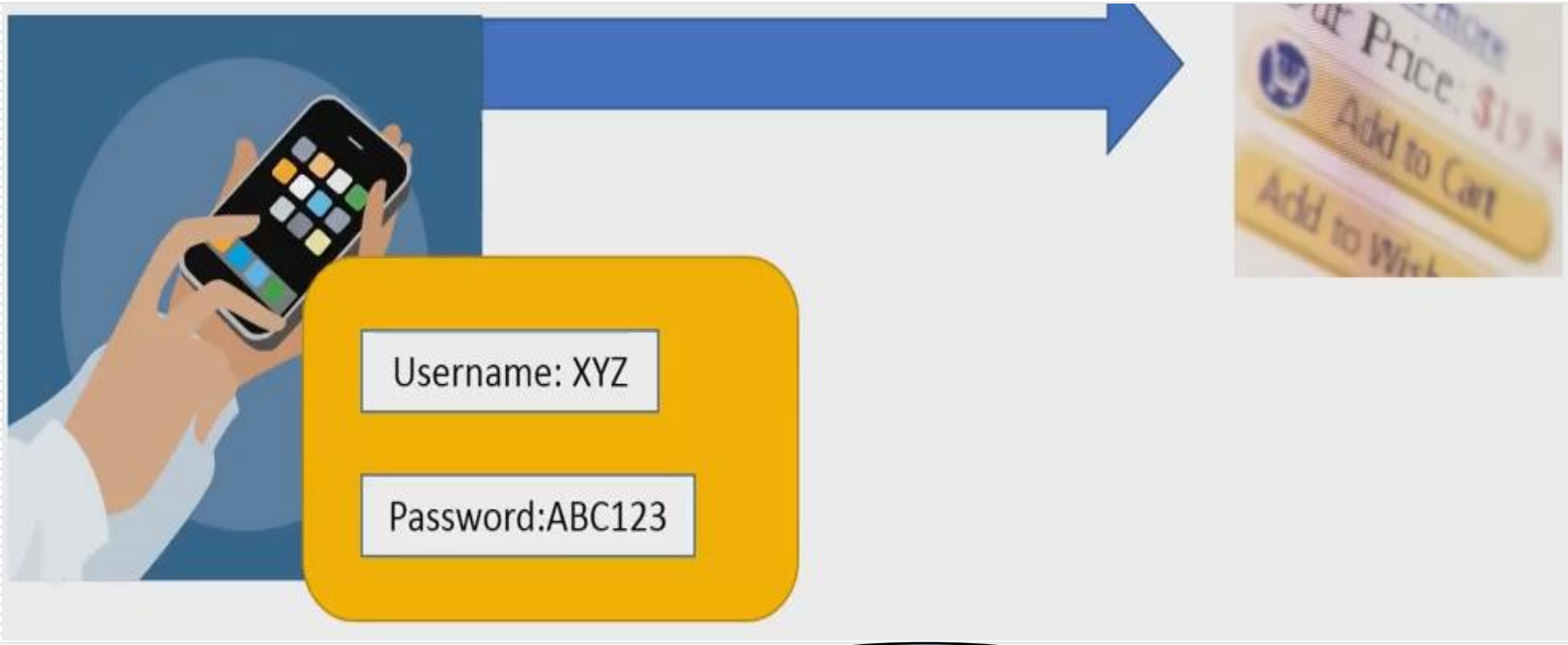
What is SharedPreferences ?



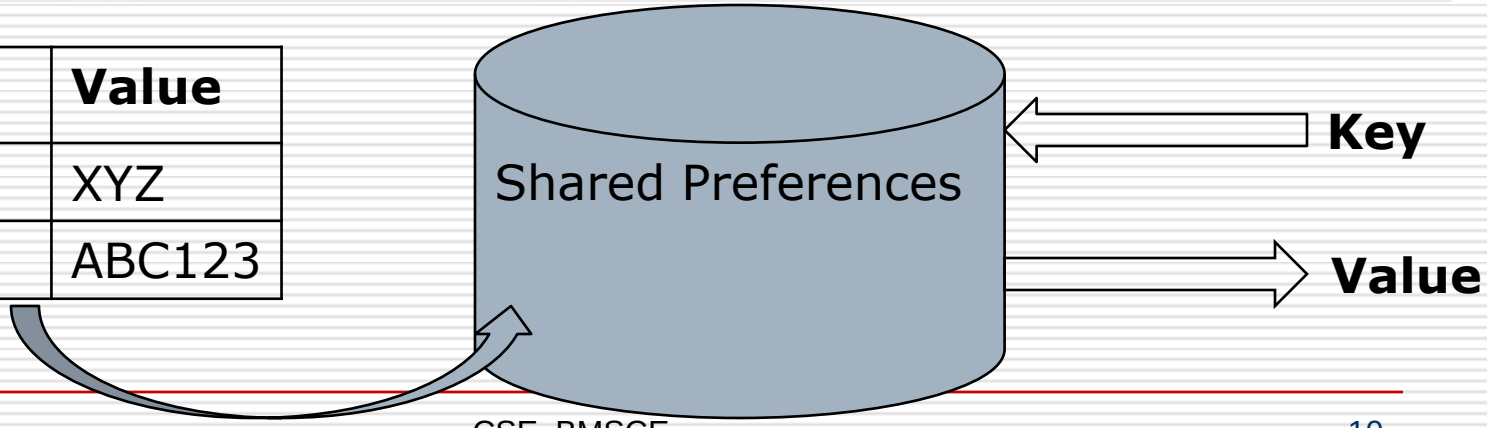
What is SharedPreferences ?



What is SharedPreferences ?

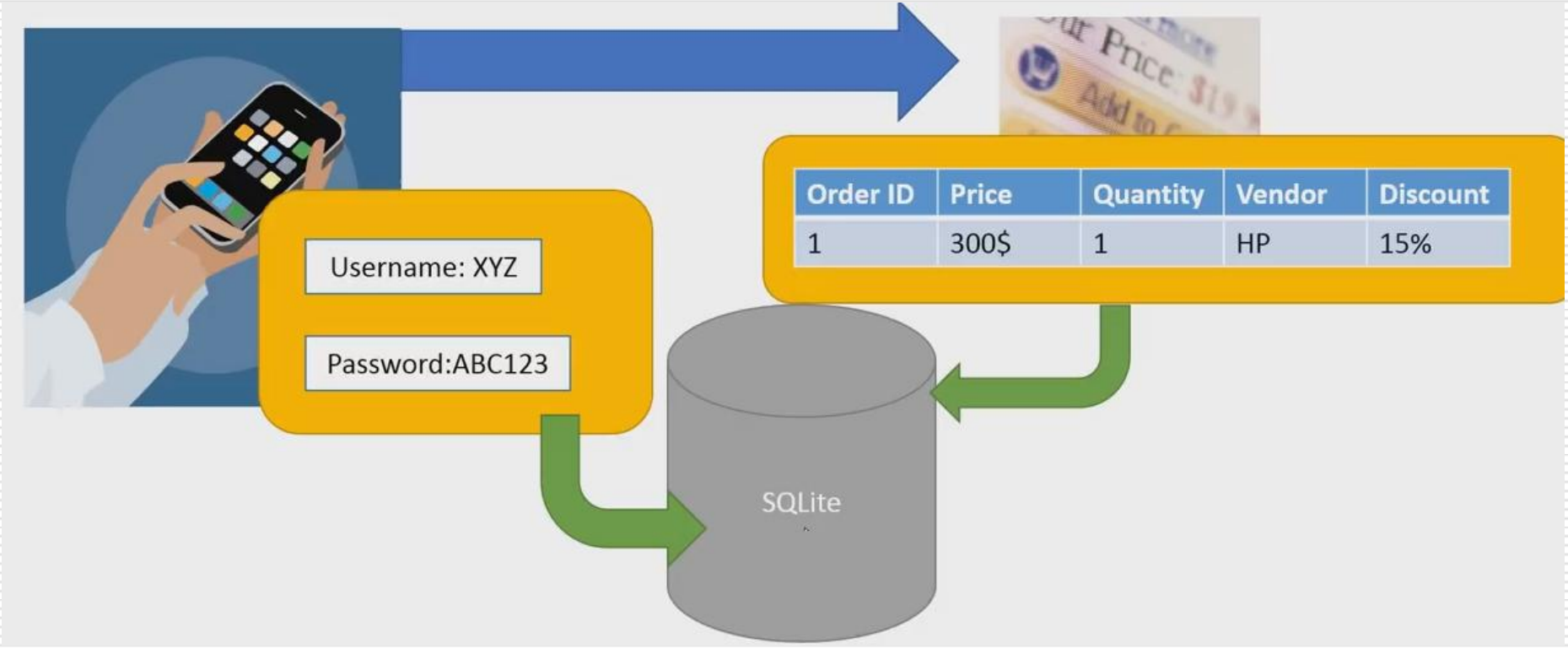


Key	Value
Username	XYZ
Password	ABC123



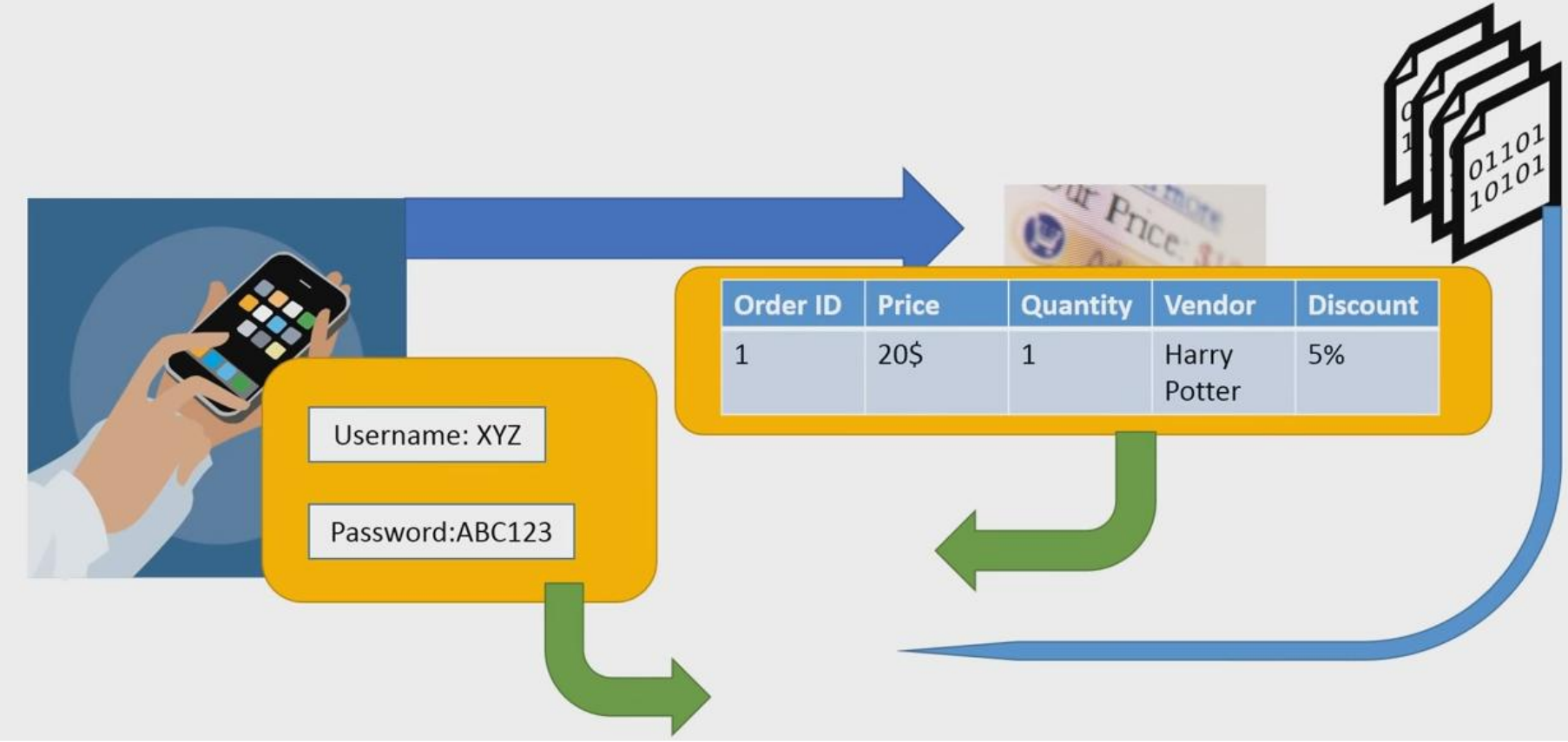
SQLite storage

SQLite storage

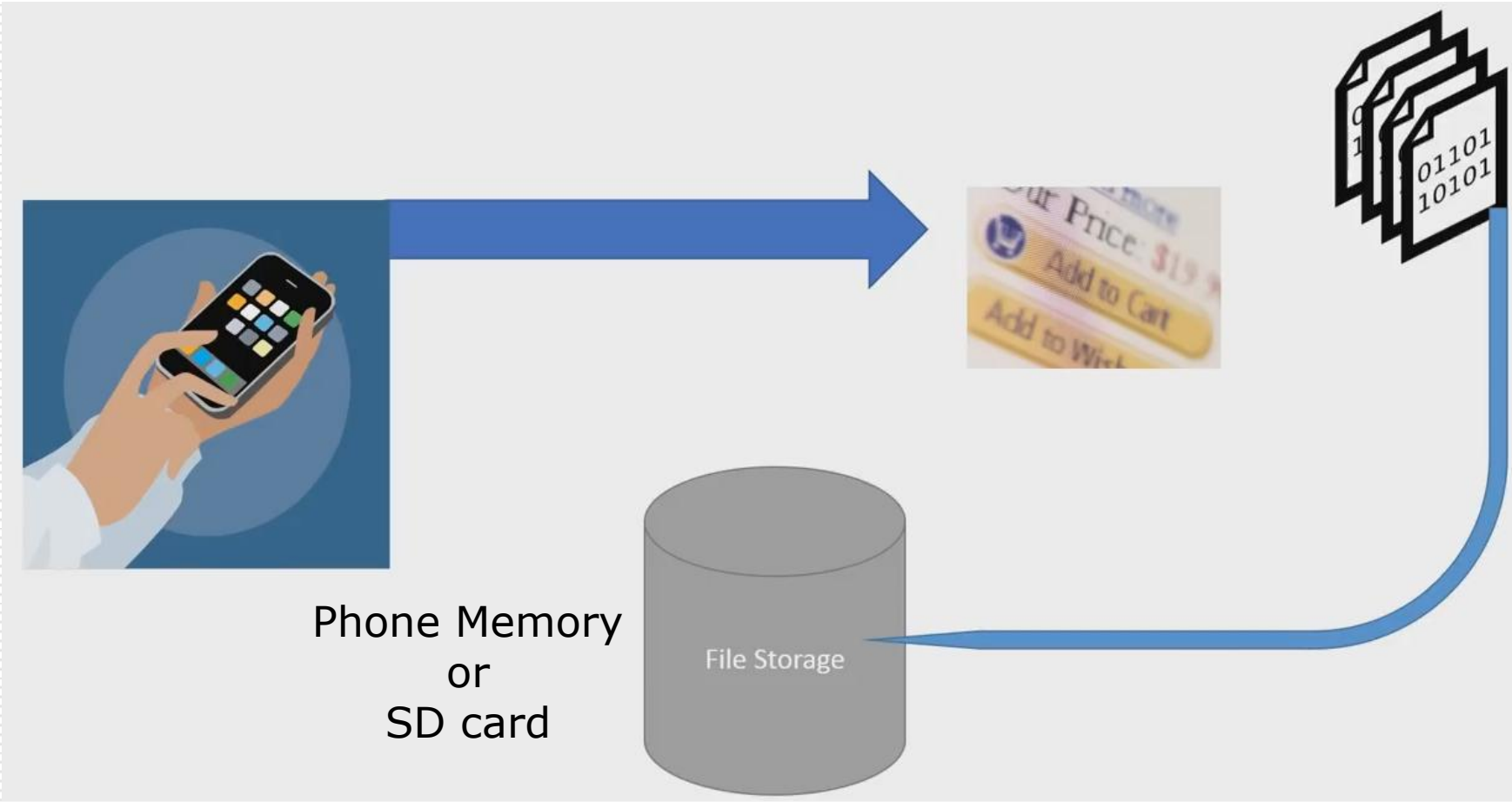


Database

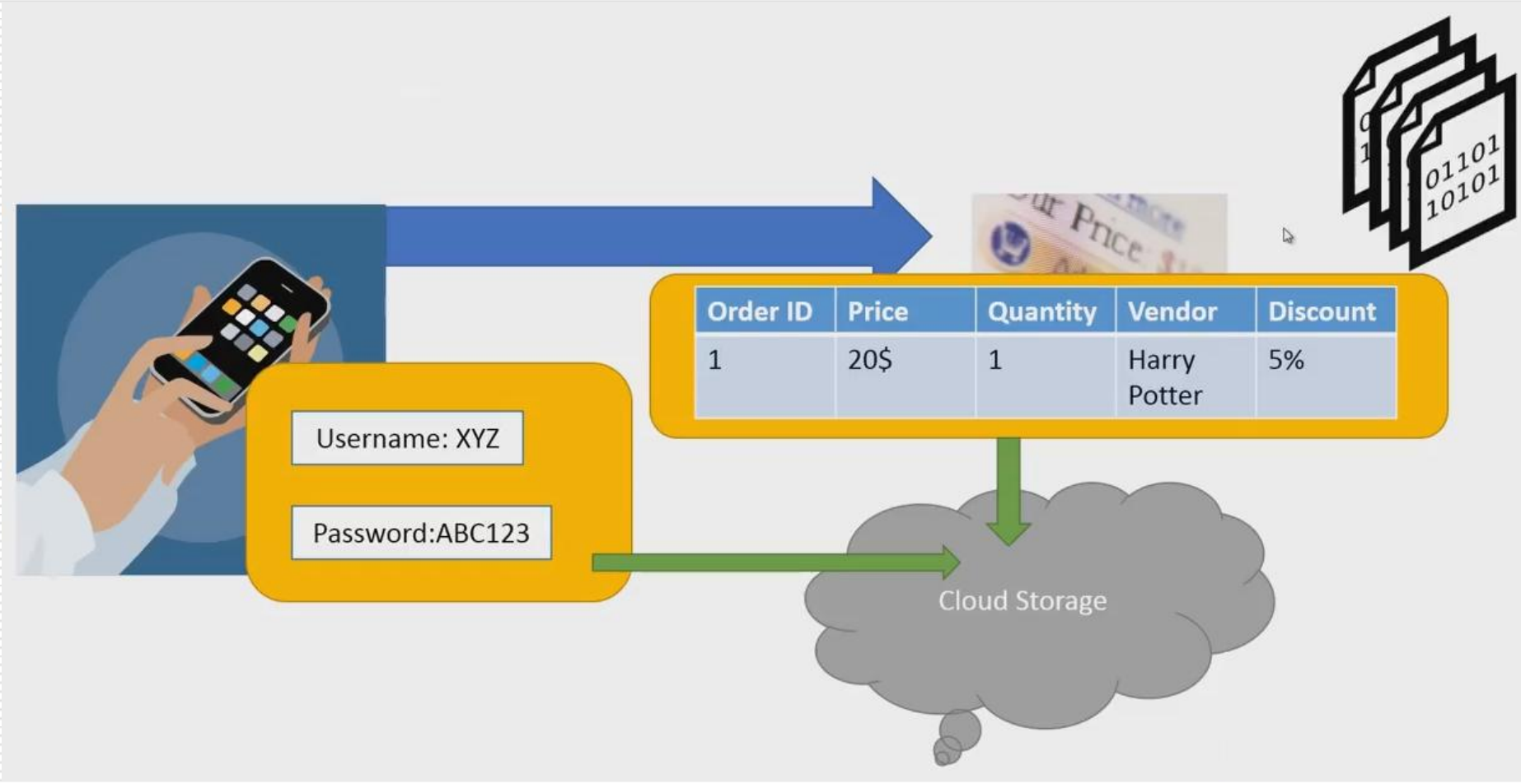
Content Providers



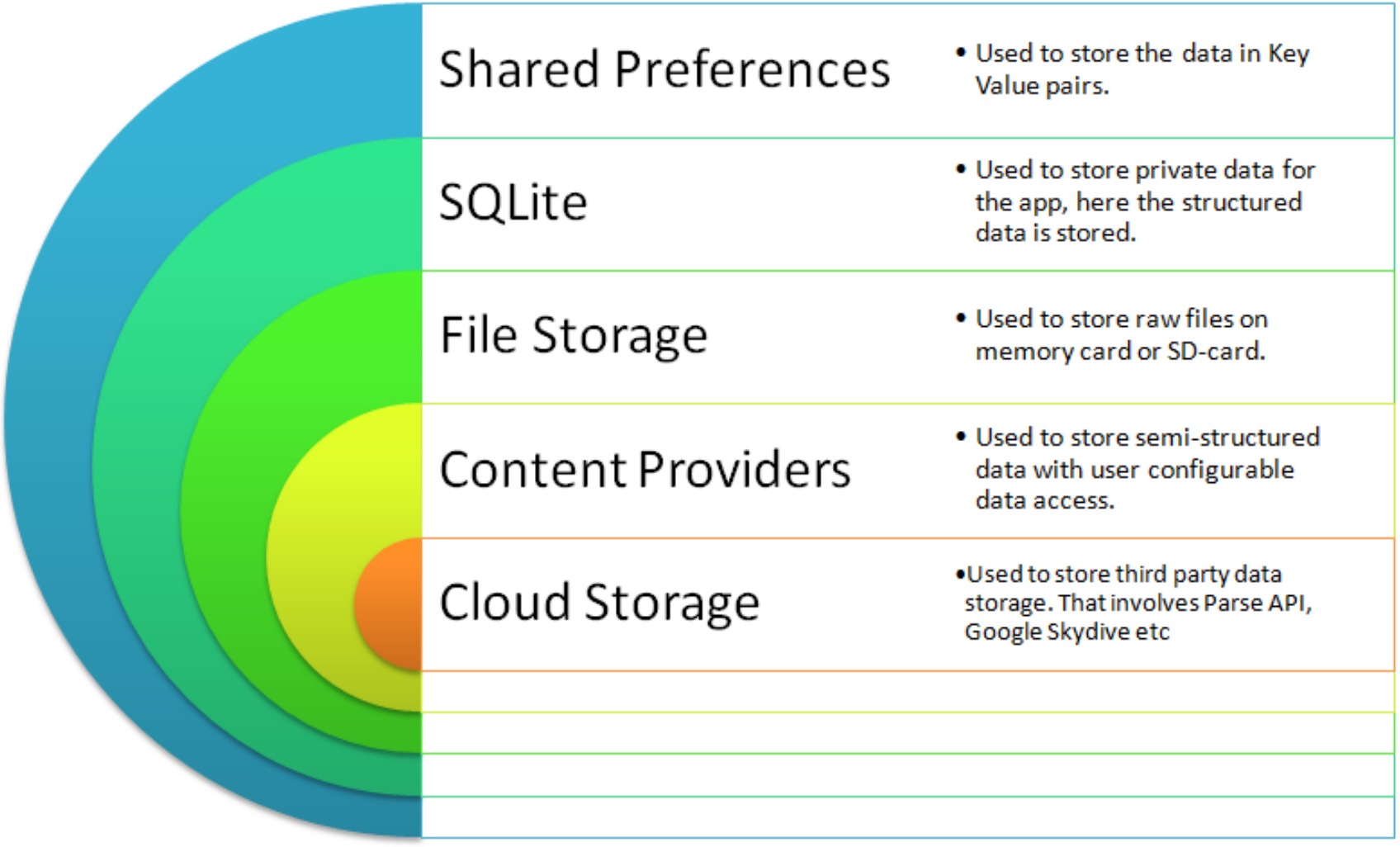
File Storage



Cloud Storage



Different Storage options in Android



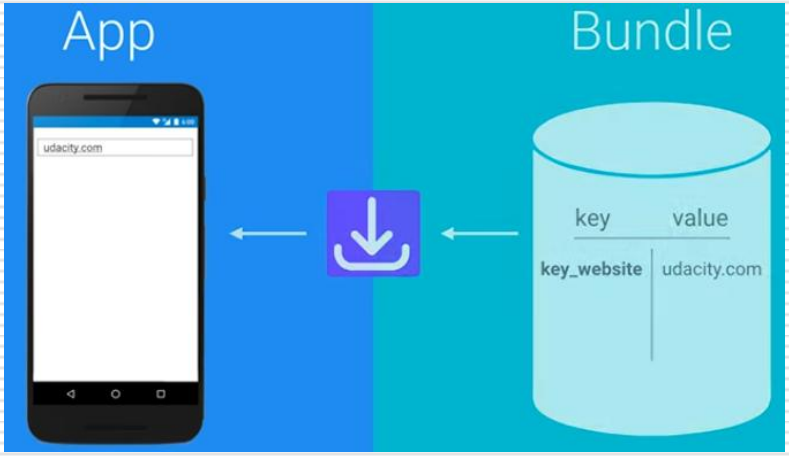
Fur further details on different data storage options in Android

- Visit the following URL

http://stackoverflow.com/questions/9986734/which-android-data-storage-technique-to-use

onSavedInstanceState

- To save app state during phone rotation "onSavedInstanceState" can be used which will store data in key/value pair



Data Persistence Types

Persistence Option	Type of data saved	Length of time saved
onSavedInstanceState	key/value (complex values)	While app is open
SharedPreferences	key/value (primitive values)	Between app and phone restarts
SQLite Database	Organized, more complicated text/numeric/boolean data	Between app and phone restarts
Internal/External storage	Multimedia or larger data	Between app and phone restarts
Server (ex. Firebase)	Data that multiple phones will access	Between app and phone restarts, deleting the app, using a different phone, etc.

What are SharedPreferences ?

- SharedPreferences is used by apps to save data in **Name-Value Pairs (NVP)**
- Data is stored in **XML file** in the directory data/data/<package-name>/shared-prefs folder
- Store data such as username, password, theme settings, other application settings
- SahredPreferences only allows you to save primitive data types (that is boolean, float, long, int and string)

Example

Key	**Value**
Username	XYZ
Password	ABC123

How to access SharedPreferences?

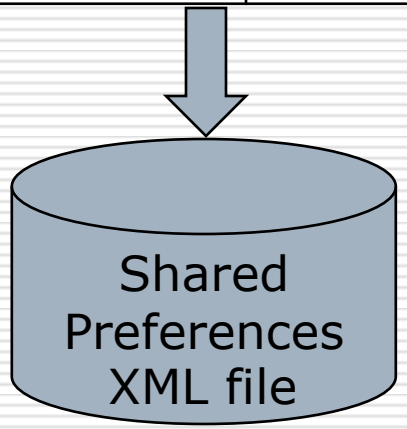
- If you have only **1 preference file**, call **getPreferences**(int mode)
- If you have **several files** call **getSharedPreferences**(String name, int mode)
- **MODE_PRIVATE**: Only your app can access the file
- **MODE_MULTI_PROCESS**: Multiple processes can modify the same shared preference file

How to use SharedPreferences?

To Store data

1. Get a reference to the SharedPreferences object
   1. For a single file, call **getPreferences(int mode)**
   2. For several files, call **getSharedPreferences(String name, int mode)**
2. Call the editor
3. Use the editor to add the data with a key
4. **Commit editor changes**

Key	Value
Username	XYZ
Password	ABC123

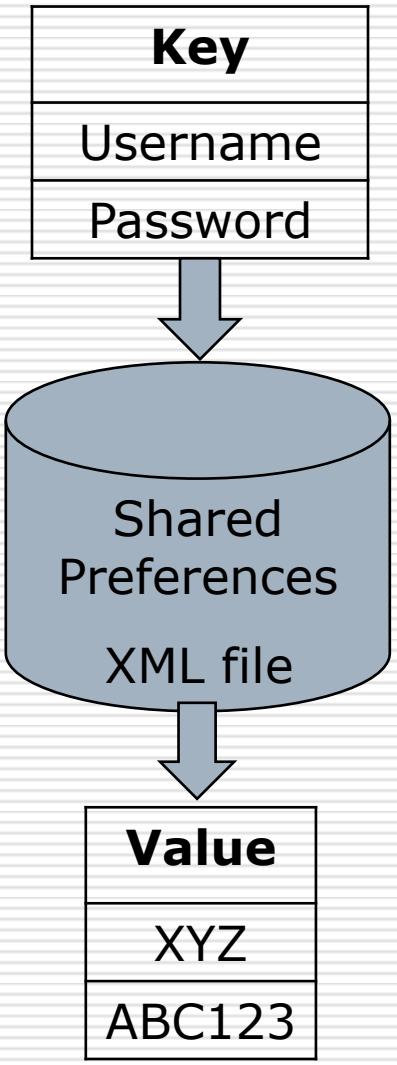



```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="USERNAME">XYZ</string>
<string name="PASSWORD">ABC123</string
</map>
```

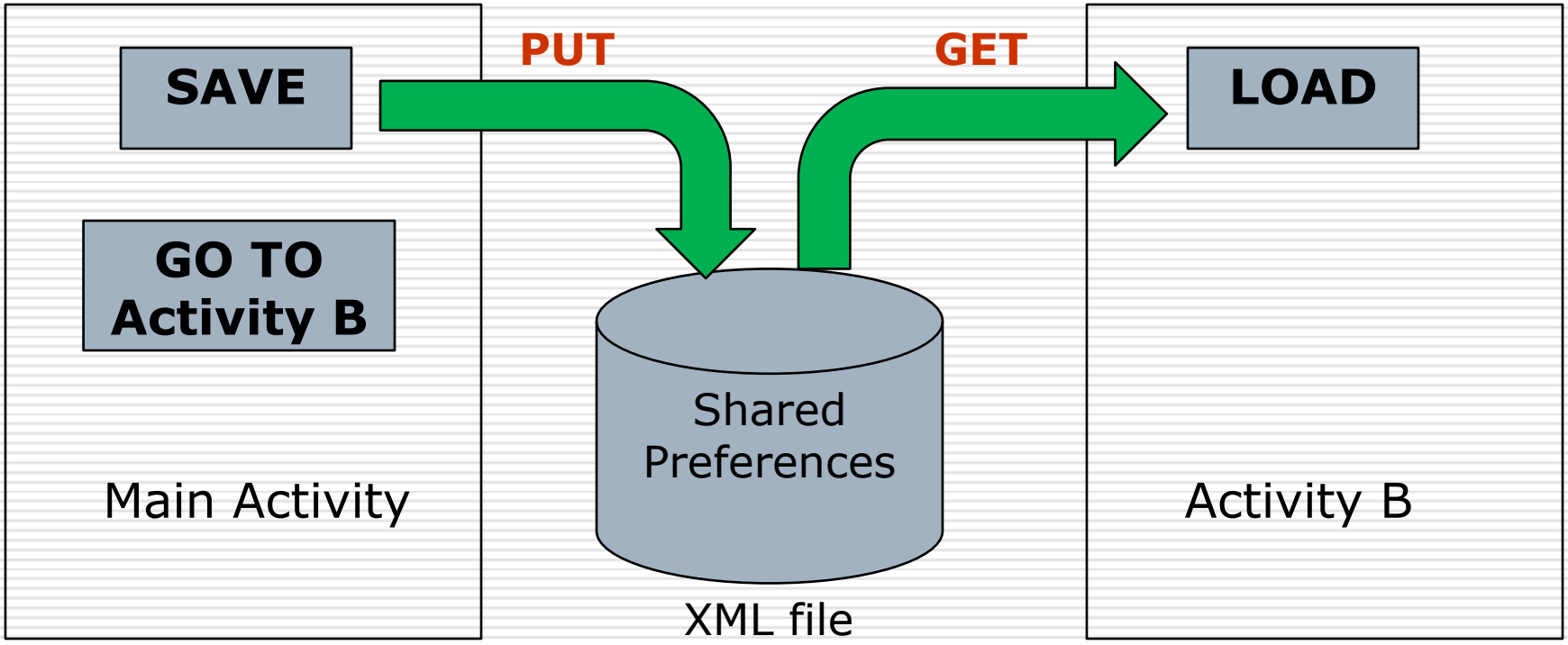
How to use SharedPreferences?

To retrieve data

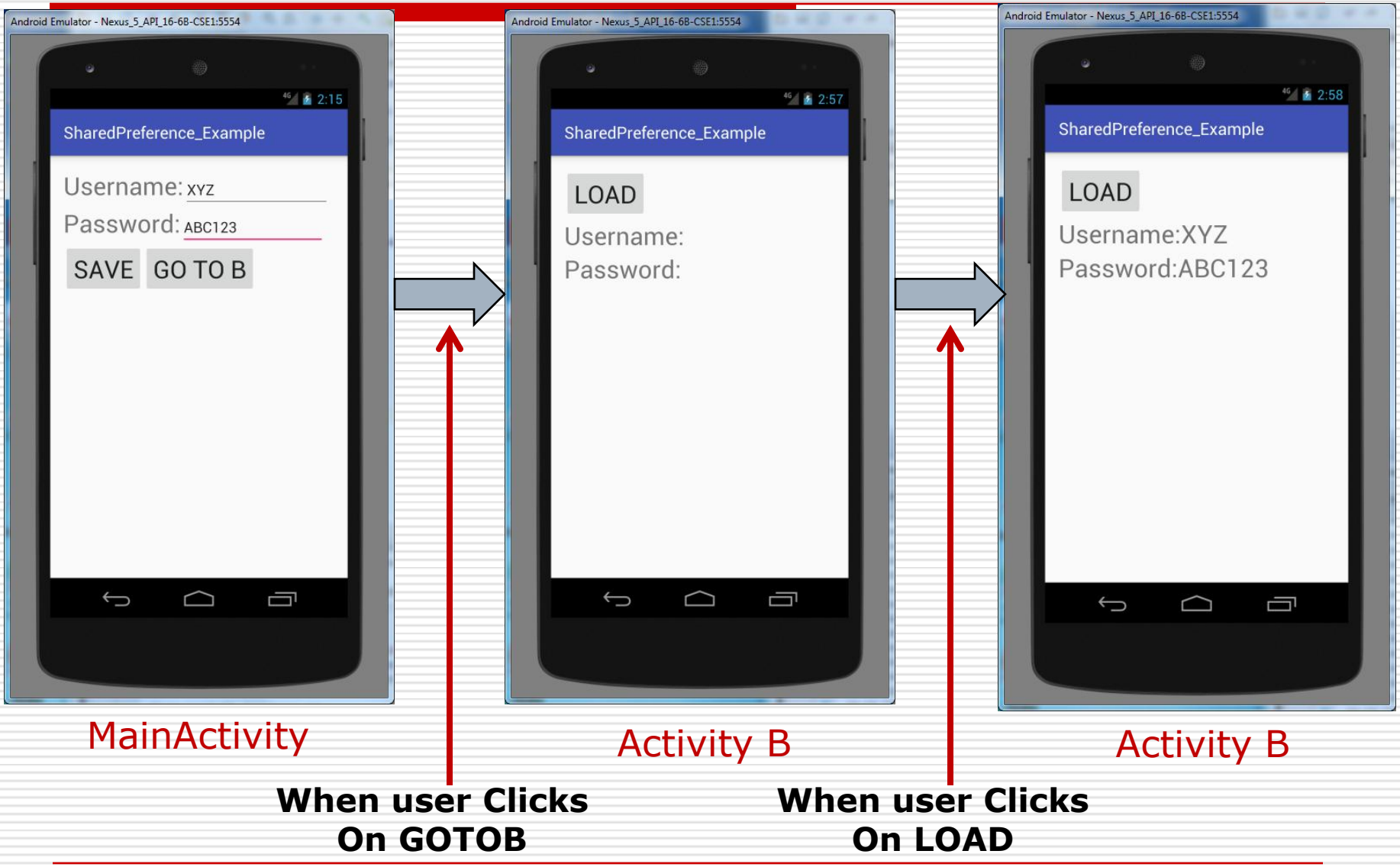
1. Get a reference to the SharedPreferences object
   1. For a single file, call **getPreferences(int mode)**
   2. For several files, call **getSharedPreferences(String name, int mode)**
- Use the key provided earlier to get data



Example: SharedPreferences



Example: SharedPreferences

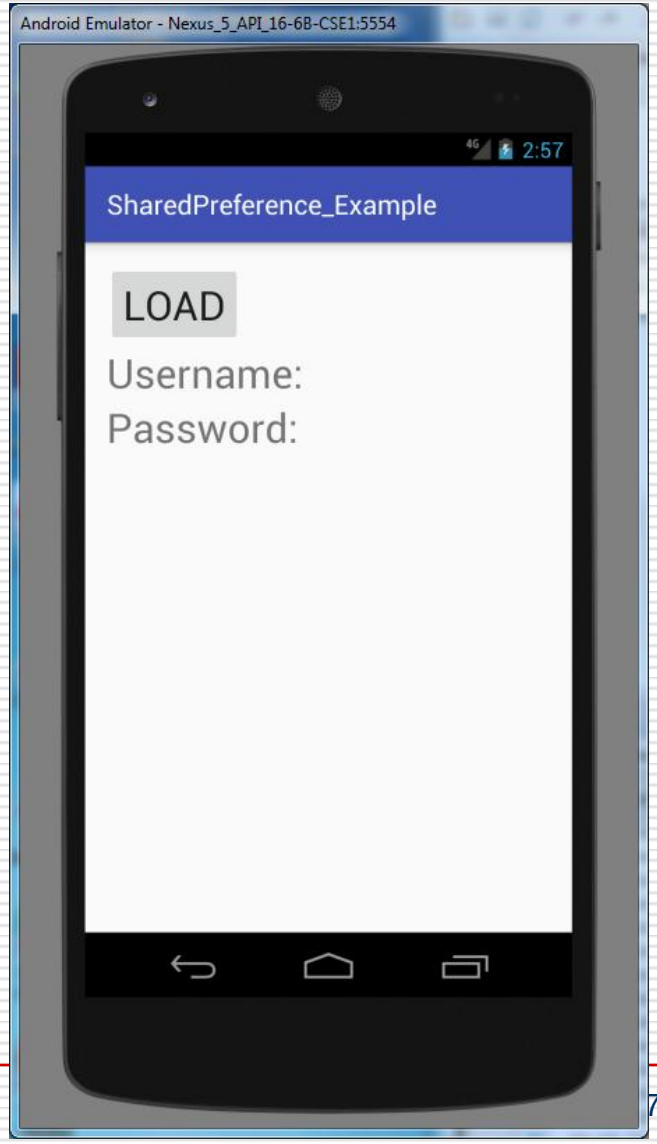


actvity_main.xml

```
<?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_main"
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"
android:orientation="vertical"
tools:context="umadevi.example.com.sharedpreference_example.MainActivity">

<LinearLayout
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="horizontal">
   <TextView
      android:id="@+id/textview"
      android:text="Username:"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"/>
   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/editText1"
      android:hint="Enter your Username"/>
</LinearLayout>
<LinearLayout
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="horizontal">
   <TextView
      android:id="@+id/textview2"
      android:text="Password:"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"/>
   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:hint="Enter your Password"
      android:id="@+id/editText2"/>
</LinearLayout>
   <LinearLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:orientation="horizontal">
<Button
   android:id="@+id/btnSave"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:textSize="30sp"
   android:text="Save"
   android:onClick="saveData"/>
      <Button
         android:id="@+id/btnB"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:textSize="30sp"
         android:text="Go to B"
         android:onClick="GoToB"/>
   </LinearLayout>
</LinearLayout>
```

activity_layout_b.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_main"
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"
android:orientation="vertical"
tools:context="umadevi.example.com.sharedpreference_example.ActivityB">
    <Button
       android:id="@+id/btnSave"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textSize="30sp"
       android:text="Load"
       android:onClick="load"/>
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
       android:id="@+id/textview"
       android:text="Username:"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textSize="30sp"/>
    <TextView
       android:id="@+id/textviewUsername"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textSize="30sp"/>
</LinearLayout>
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
       android:id="@+id/textview2"
       android:text="Password:"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textSize="30sp"/>
    <TextView
       android:id="@+id/textviewPasswd"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textSize="30sp"/>
</LinearLayout>
</LinearLayout>
```



MainActivity.java

```
public class MainActivity extends AppCompatActivity {
EditText userName, passWord;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
}
public void saveData(View v)    {
userName=(EditText)findViewById(R.id.editText1);
passWord=(EditText)findViewById(R.id.editText2);
SharedPreferences mySharedPref=getSharedPreferences("myData",
Context.MODE_PRIVATE);



}
public void GoToB(View v){

}}
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
EditText userName, passWord;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
}
public void saveData(View v)    {
userName=(EditText)findViewById(R.id.editText1);
passWord=(EditText)findViewById(R.id.editText2);
SharedPreferences mySharedPref=getSharedPreferences("myData",
Context.MODE_PRIVATE);
SharedPreferences.Editor editor=mySharedPref.edit();




}
public void GoToB(View v){


}}
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
EditText userName, passWord;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
}
public void saveData(View v)    {
userName=(EditText)findViewById(R.id.editText1);
passWord=(EditText)findViewById(R.id.editText2);
SharedPreferences mySharedPref=getSharedPreferences("myData",
Context.MODE_PRIVATE);
SharedPreferences.Editor editor=mySharedPref.edit();
editor.putString("USERNAME",userName.getText().toString());
editor.putString("PASSWORD",passWord.getText().toString());

}
public void GoToB(View v){

}}
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
EditText userName, passWord;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
}
public void saveData(View v)    {
userName=(EditText)findViewById(R.id.editText1);
passWord=(EditText)findViewById(R.id.editText2);
SharedPreferences mySharedPref=getSharedPreferences("myData",
Context.MODE_PRIVATE);
SharedPreferences.Editor editor=mySharedPref.edit();
editor.putString("USERNAME",userName.getText().toString());
editor.putString("PASSWORD",passWord.getText().toString());
editor.commit();
}
public void GoToB(View v){


}}
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
EditText userName, passWord;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
}
public void saveData(View v)    {
userName=(EditText)findViewById(R.id.editText1);
passWord=(EditText)findViewById(R.id.editText2);
SharedPreferences mySharedPref=getSharedPreferences("myData",
Context.MODE_PRIVATE);
SharedPreferences.Editor editor=mySharedPref.edit();
editor.putString("USERNAME",userName.getText().toString());
editor.putString("PASSWORD",passWord.getText().toString());
editor.commit();
}
public void GoToB(View v){
 Intent i=new Intent(MainActivity.this,ActivityB.class);
startActivity(i); }}
```

ActivityB.java

```
public class ActivityB extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_layout_b);
    }
public void load(View v)    {
 TextView user,pwd;
 user=(TextView)findViewById(R.id.textviewUsername);
 pwd=(TextView)findViewById(R.id.textviewPasswd);
SharedPreferences mySharedPref=getSharedPreferences("myData",
                                        Context.MODE_PRIVATE);




}}
```

ActivityB.java

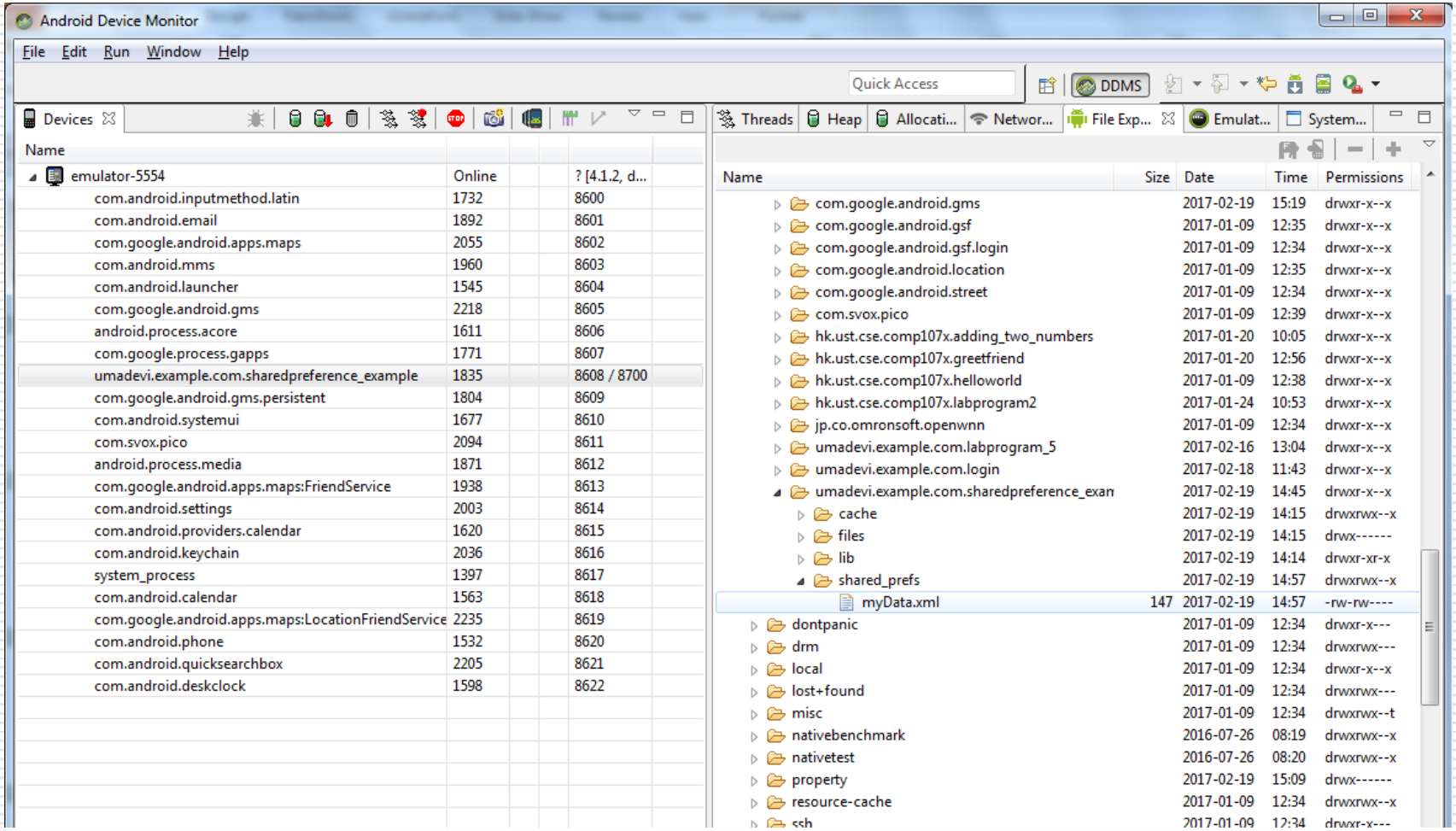
```
public class ActivityB extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_layout_b);
    }
public void load(View v)    {
 TextView user,pwd;
 user=(TextView)findViewById(R.id.textviewUsername);
 pwd=(TextView)findViewById(R.id.textviewPasswd);
SharedPreferences mySharedPref=getSharedPreferences("myData",
                                                Context.MODE_PRIVATE);
String username=mySharedPref.getString("USERNAME","");
String password=mySharedPref.getString("PASSWORD","");



}}
```

ActivityB.java

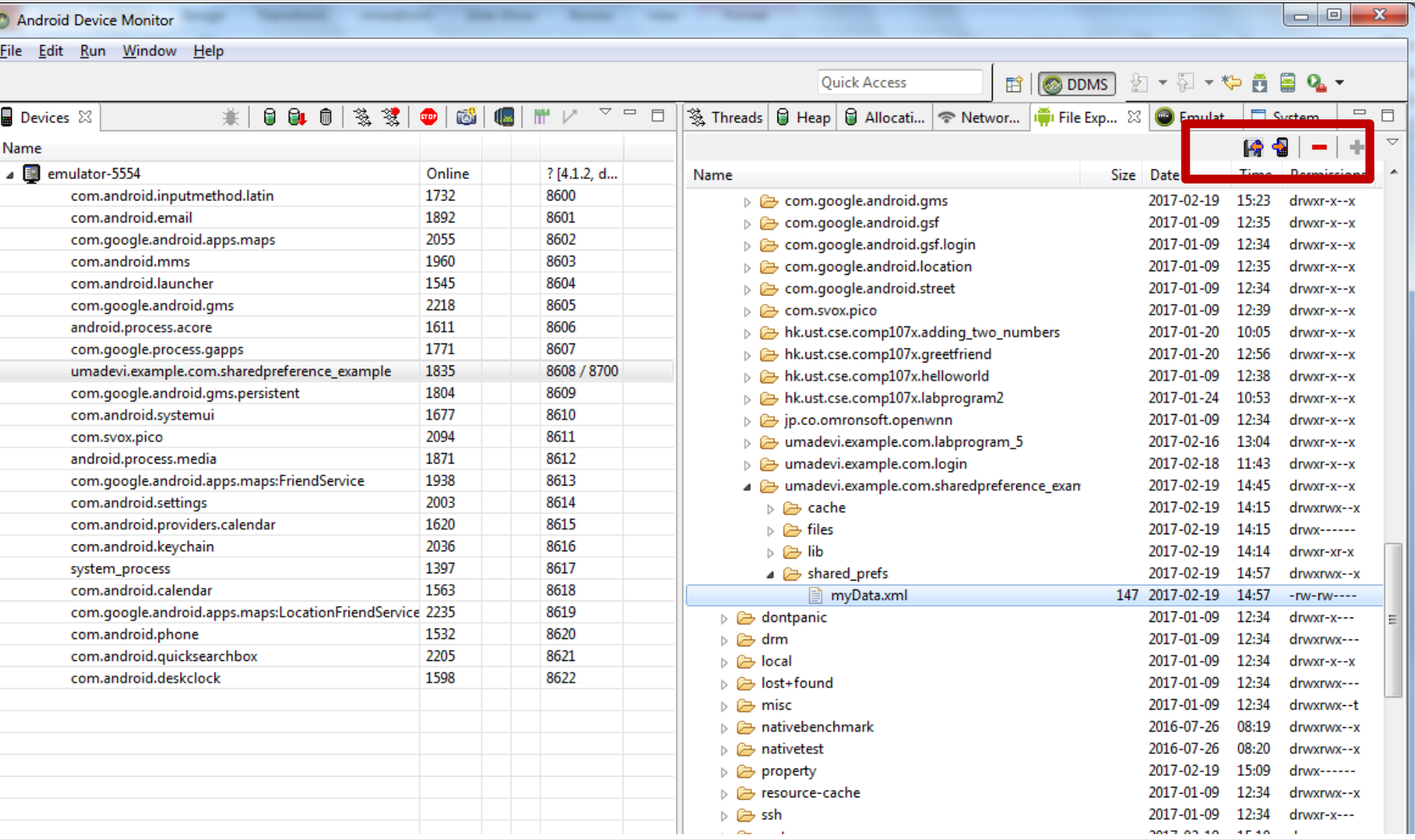
```
public class ActivityB extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_layout_b);
    }
public void load(View v)    {
 TextView user,pwd;
 user=(TextView)findViewById(R.id.textviewUsername);
 pwd=(TextView)findViewById(R.id.textviewPasswd);
SharedPreferences mySharedPref=getSharedPreferences("myData",
                                        Context.MODE_PRIVATE);
String username=mySharedPref.getString("USERNAME","");
String password=mySharedPref.getString("PASSWORD","");
user.setText(username);
pwd.setText(password);    }}
```

View SharedPreference XML file in Android Studio

- tools -> Android Device Monitor -> clicking on the emulator in the left panel -> file explorer -> data -> data -> com.project-name



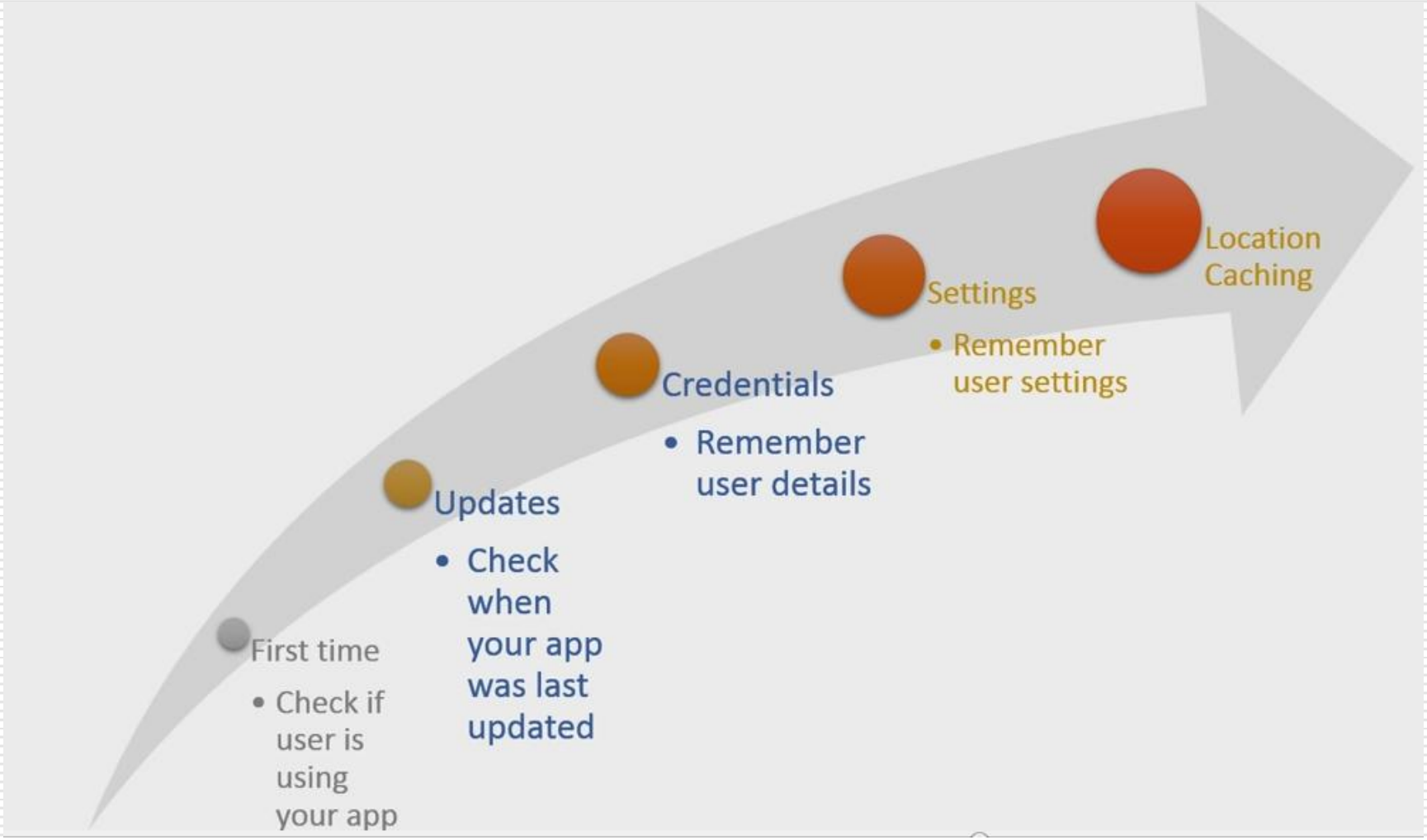
Inside “Android Device Monitor” window click on the right corner to pull SharedPreference XML file



SahredPreference XML file **myData.xml**

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="USERNAME">XYZ</string>
<string name="PASSWORD">ABC123</string>
</map>
```

Some uses of SharedPreferences



INTRODUCING THE PREFERENCE FRAMEWORK AND THE PREFERENCE ACTIVITY

Preference API's

- During the development of mobile applications, a common requirement is of storing app related data to the phone.
- In Android we perform this task with the help of **Preference APIs**.
- We may also need settings that allow users to modify preferences in app. Thankfully Android provides a powerful **framework** to **manage user preferences**.

Preference Framework

Preference Framework we have four components to cover, they are:

- **Preference Activity or Fragment** – these host the Preference Screen, displaying your settings. We may use any one (Activity or Fragment) as per our requirement.
- **Preference XML** – a xml file defining your settings items.
- **Preference Headers** – these are lists of subscreens. An xml file defines the Preference Fragments used for the Headers subscreens.
- **Shared Preference Change Listener** – listens for any changes in the Shared Preference values.

Defining a Preference Screen Layout in XML

- Each preference layout is defi ned as a hierarchy, beginning with a single PreferenceScreen

element:

<?xml version=”1.0” encoding=”utf-8”?>

<PreferenceScreen

xmlns:android=”http://schemas.android.com/apk/res/android”>

</PreferenceScreen>

- Preference Category elements, as shown in the following snippet, are used to break each Preference Screen into subcategories using a title bar separator:

<PreferenceCategory

android:title=”My Preference Category”/>

Preference Control Attributes

- android:key — The Shared Preference key against which the selected value will be recorded.
- android:title — The text displayed to represent the preference.
- android:summary — The longer text description displayed in a smaller font below the title text.
- android:defaultValue — The default value that will be displayed (and elected) if no preference value has been assigned to the associated preference key.

Native Preference Controls

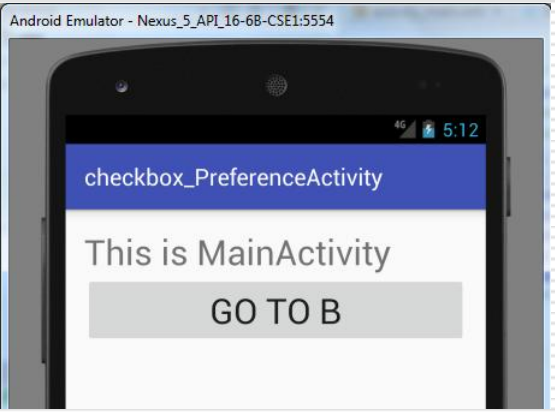
- **CheckBoxPreference** — A standard preference check box control used to set preferences to true or false.
- **EditTextPreference** — Allows users to enter a string value as a preference. Selecting the preference text at run time will display a text-entry dialog.
- **ListPreference** — The preference equivalent of a spinner. Selecting this preference will display a dialog box containing a list of values from which to select. You can specify different arrays to contain the display text and selection values.
- **MultiSelectListPreference** — Introduced in Android 3.0 (API level 11), this is the preference equivalent of a check box list.
- **RingtonePreference** — A specialized List Preference that presents the list of available ringtones for user selection. This is particularly useful when you're constructing a screen to configure notification settings.

To Do

Open your Mobile Phones
Click on Settings, do few settings, Close
Open once again Settings

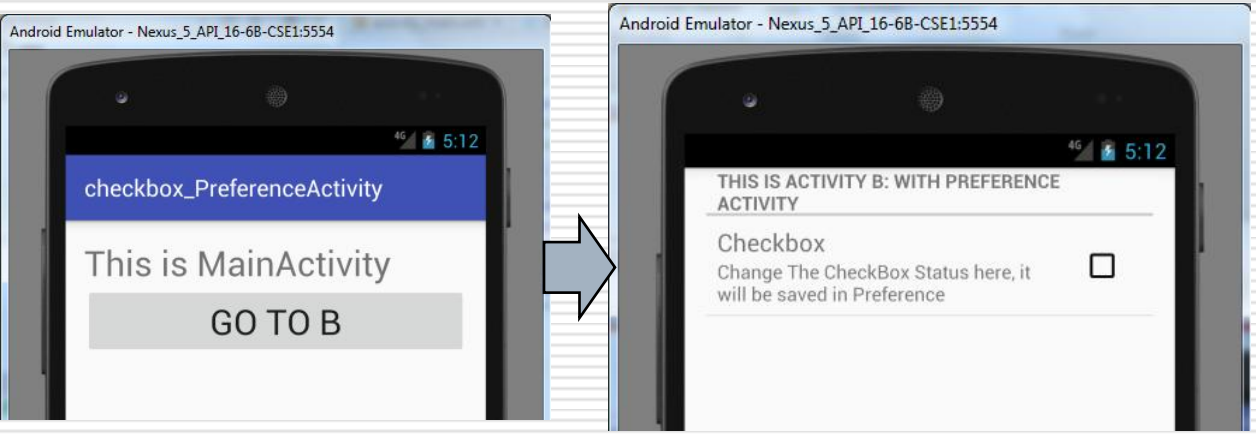
Output with PreferenceActivity

Click on
"Go To B"



Output with PreferenceActivity

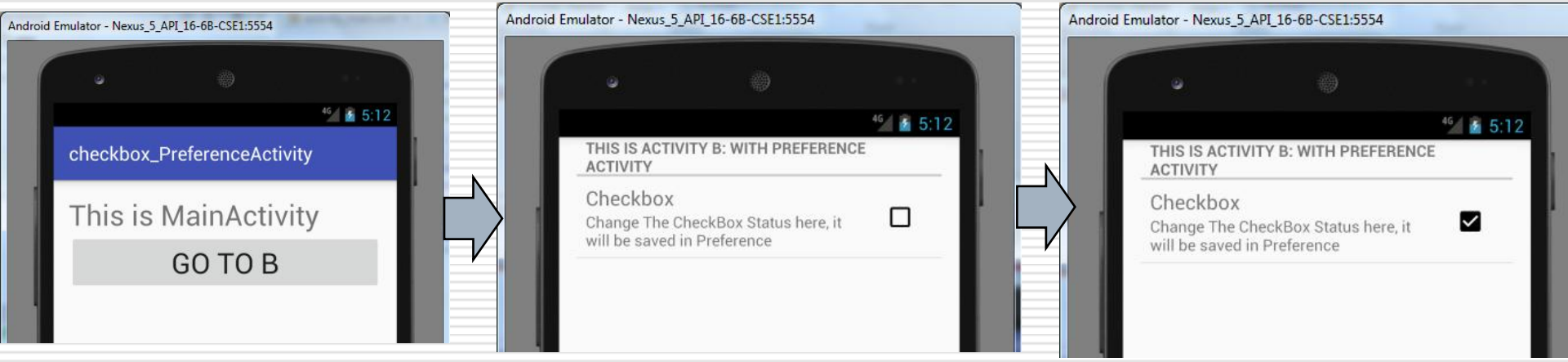
Click on
"Go To B"



Output with PreferenceActivity

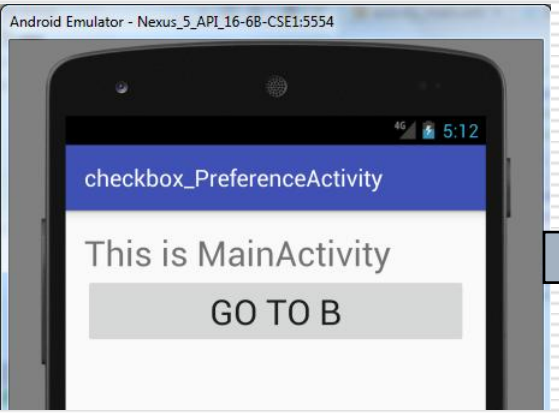
Click on "Go To B"

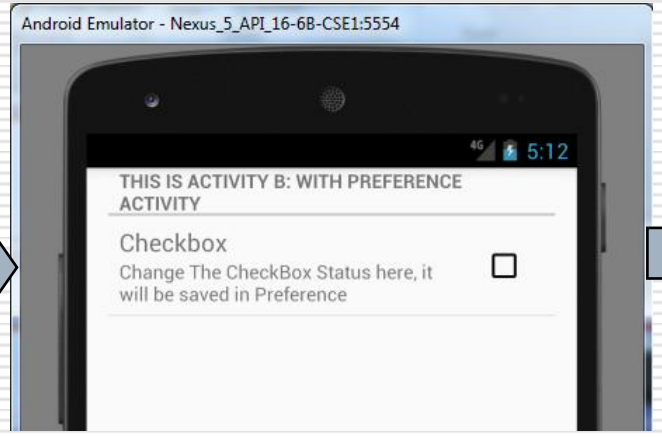
Check the Checkbox



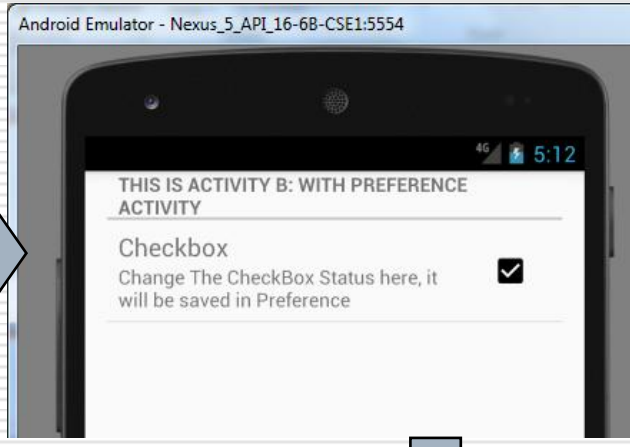
Output with PreferenceActivity

Click on "Go To B"

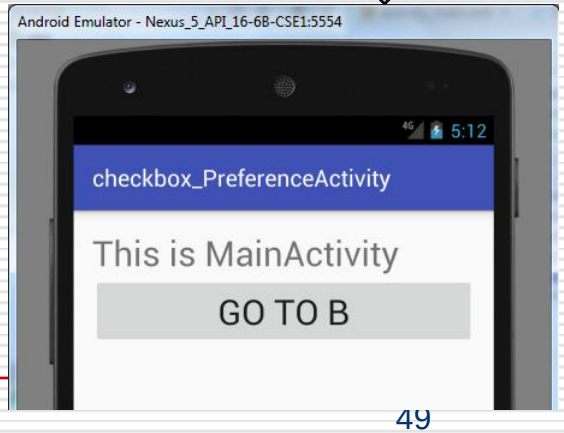




Check the Checkbox

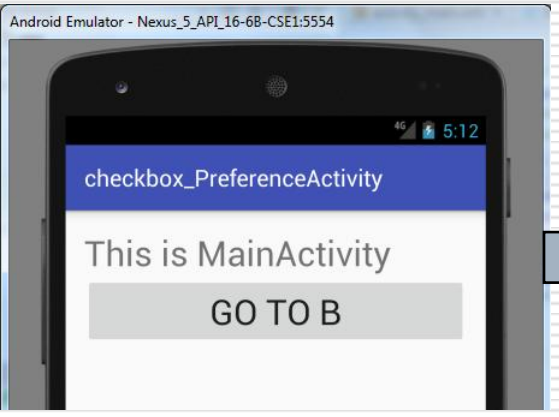


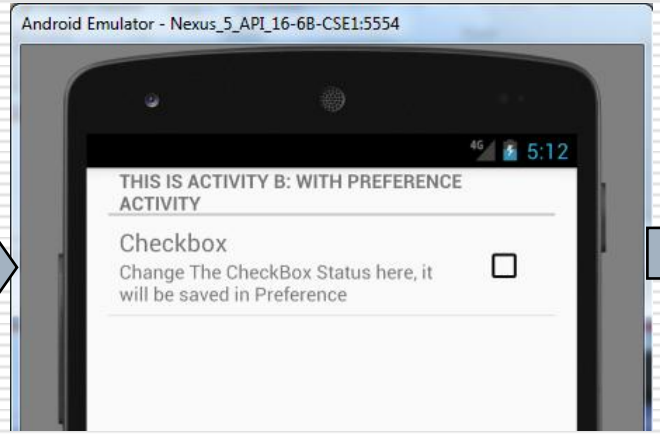
Pressback Button



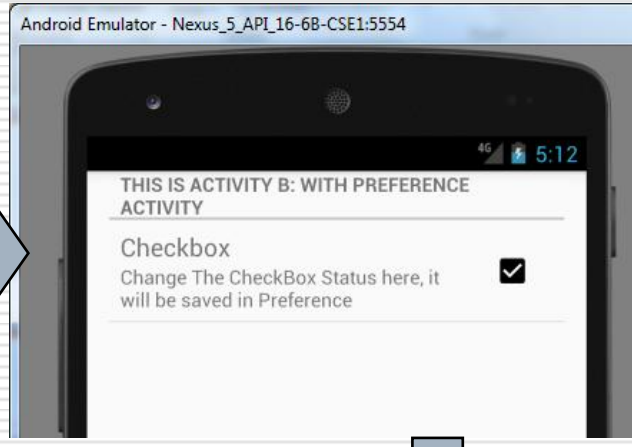
Output with PreferenceActivity

Click on "Go To B"

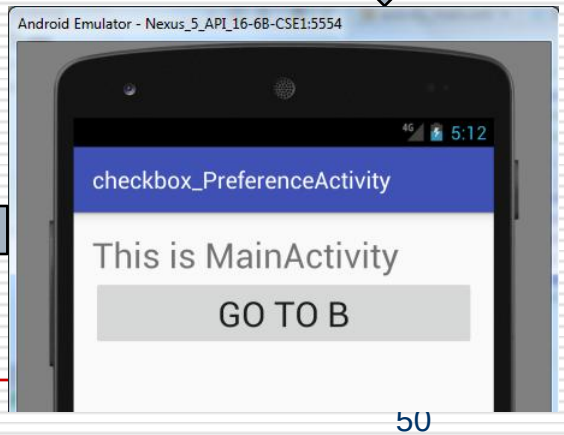




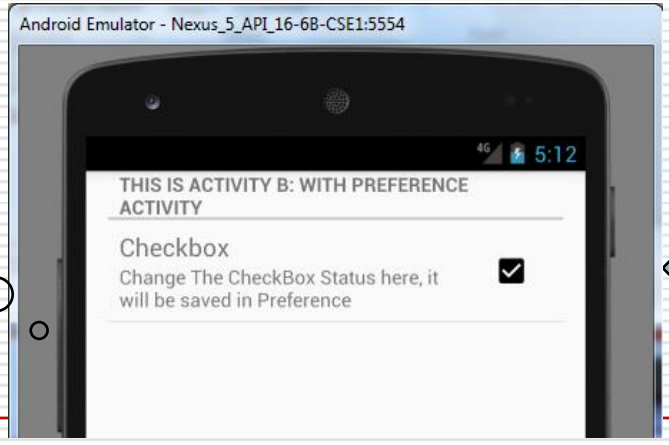
Check the Checkbox



Pressback Button



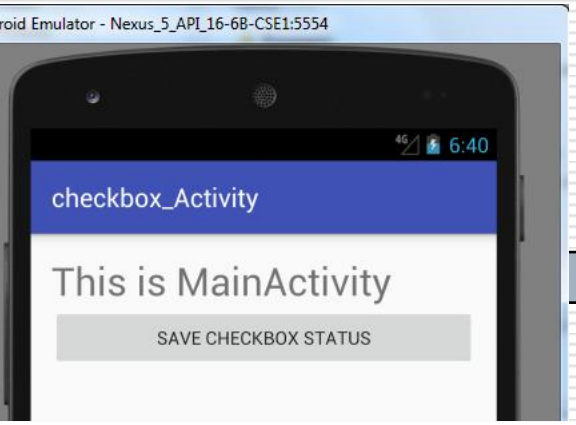
Click on "Go To B"

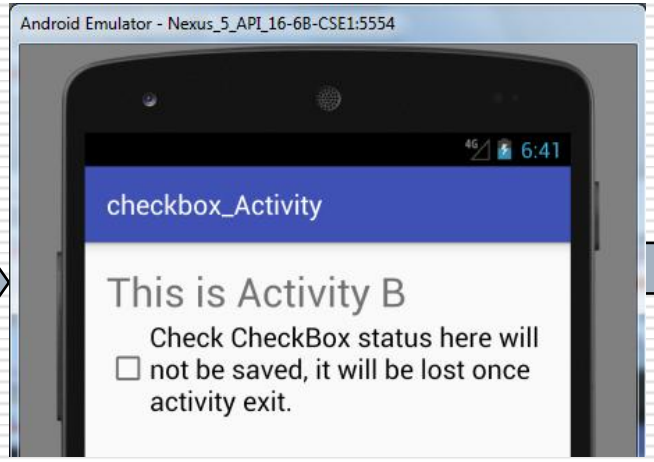


Checkbox status has been saved

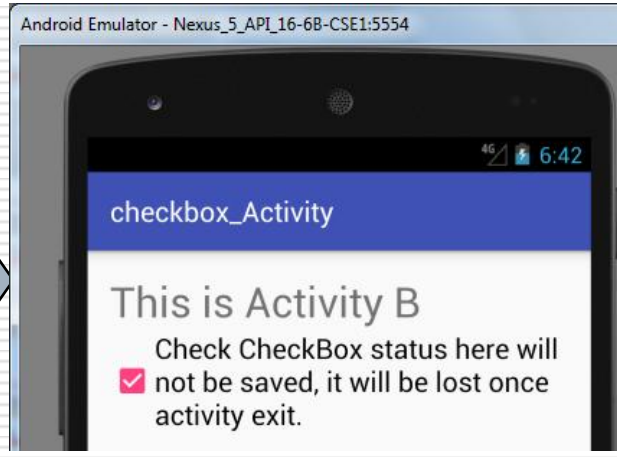
Output without PreferenceActivity

Click on"Go To B
(Save CheckBoxStatus)"

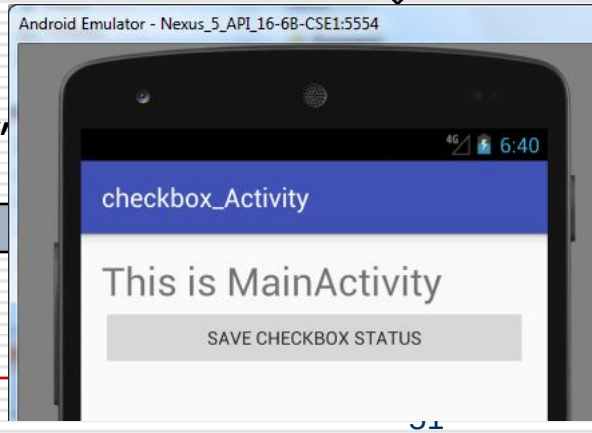




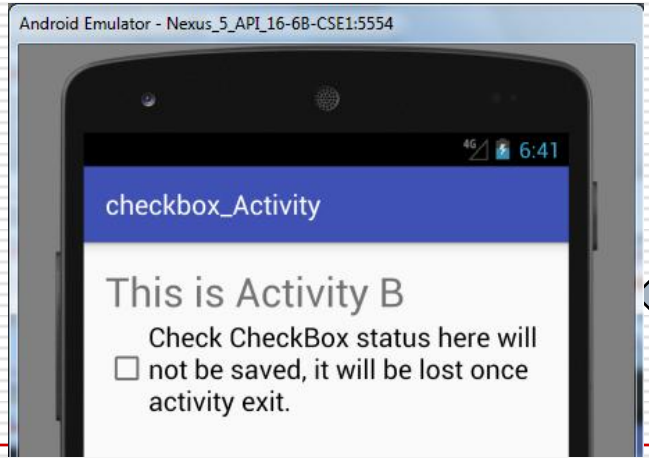
Check the
Checkbox



Pressback
Button

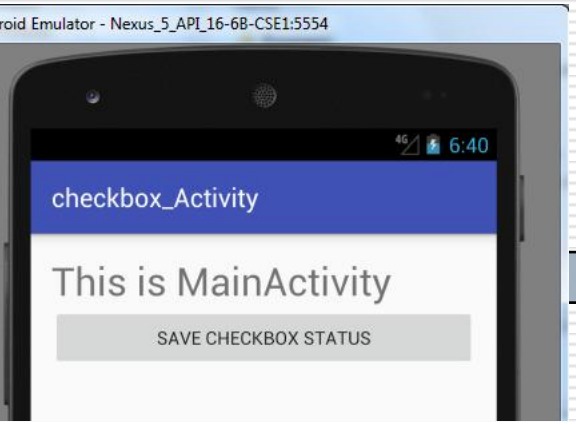


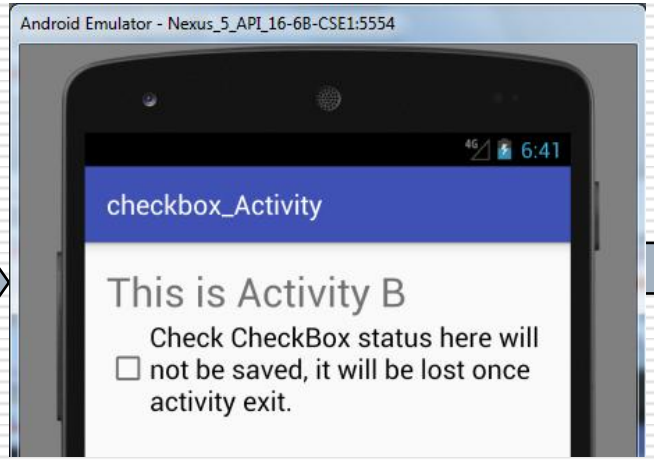
Click on
"Go To B"



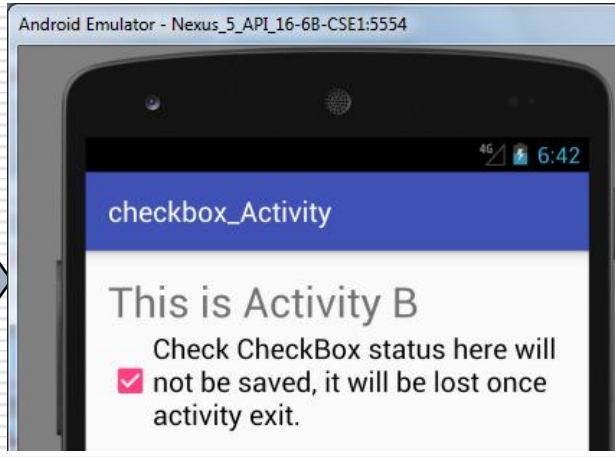
Output without PreferenceActivity

Click on"Go To B (Save CheckBoxStatus)"

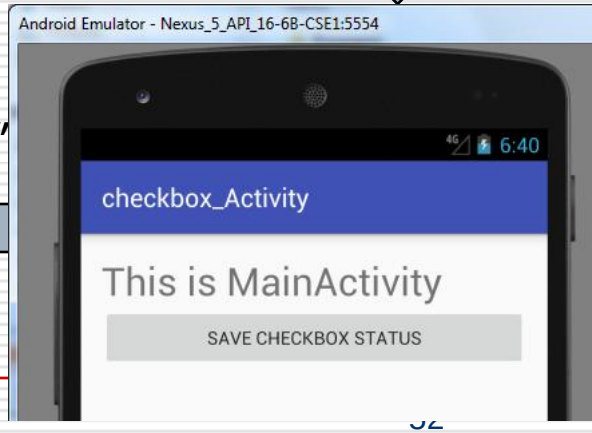




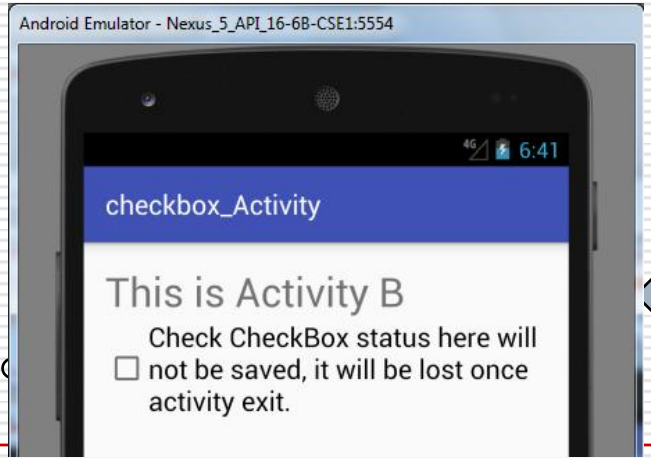
Check the Checkbox



Pressback Button



Click on "Go To B"



Checkbox status **has not** been saved

Saving checkbox status without Preference Activity

activity_layout_b.xml

```
<LinearLaoyout……..
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Activity B"
        android:textSize="30sp"/>
    <CheckBox
        android:id="@+id/checkbox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:defaultValue="false"
        android:text="Check CheckBox status here will not be saved, it will be lost once activity exit."
        android:textSize="20sp"/>
</LinearLayout>
```

ActivityB.java

```
public class ActivityB extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_layout_b);
    }
}
```

activity_main.xml

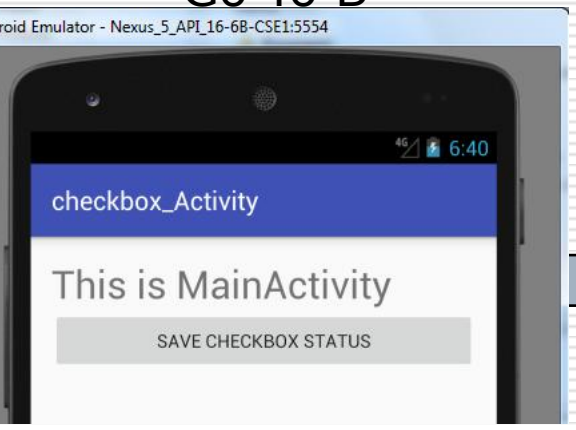
```
<LinearLayout ……
 <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is MainActivity"
      android:textSize="30sp"/>
   <Button
      android:id="@+id/setpreference"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Go To B/ Save Checkbox Status"
      android:onClick="loadActivityB"/>
</LinearLayout>
```

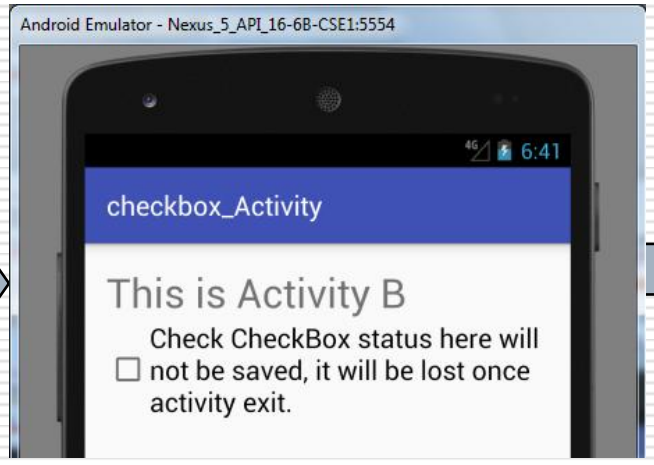
MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void loadActivityB(View v){
        Intent i=new Intent(MainActivity.this, ActivityB.class);
        startActivity(i);
    }
}
```

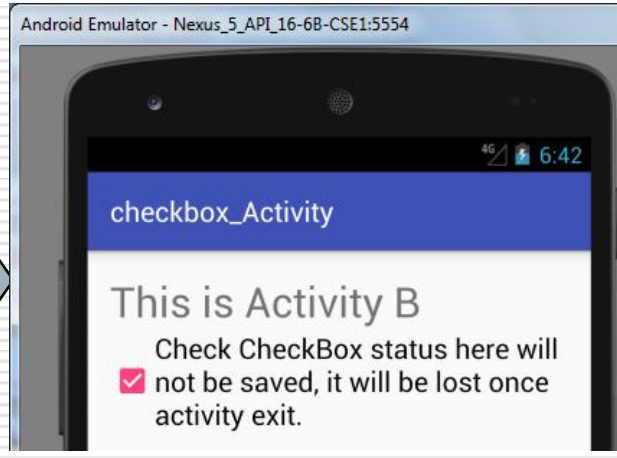
Output

Click on "Go To B"

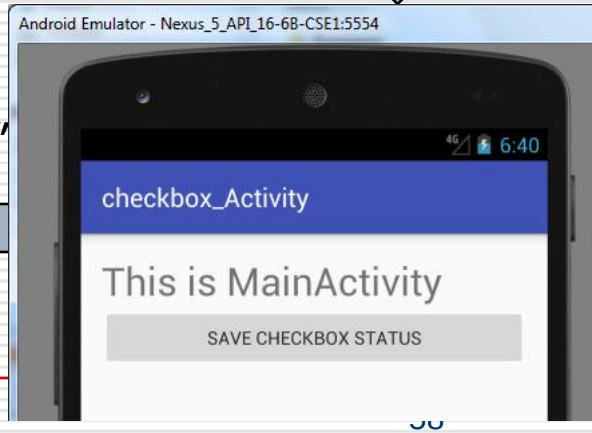




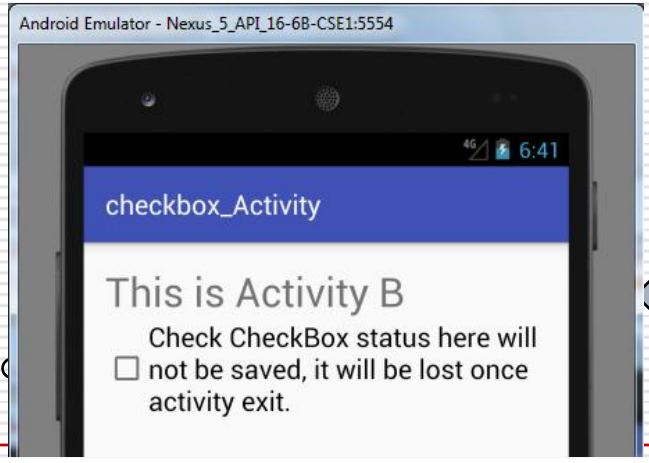
Check the Checkbox



Pressback Button



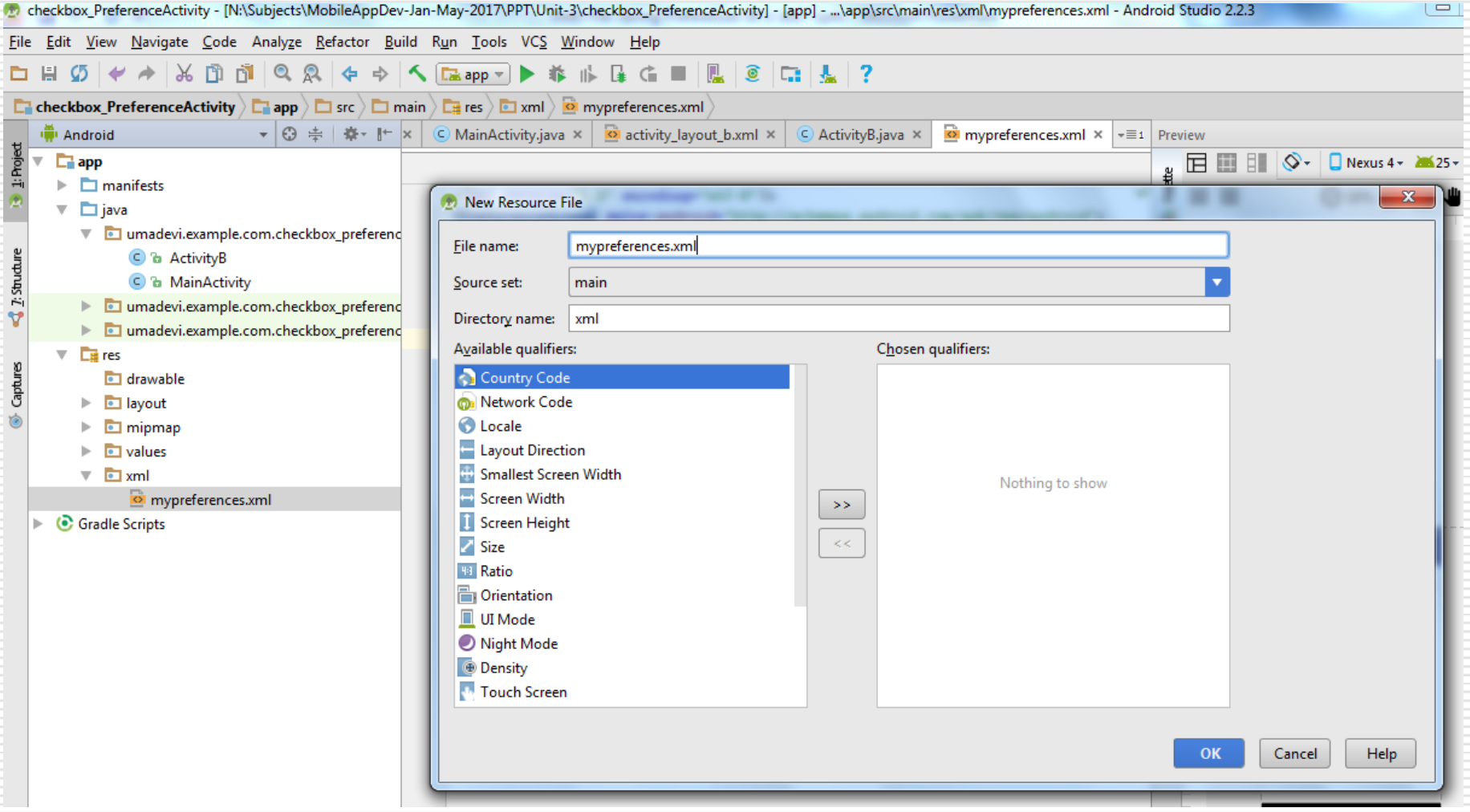
Click on "Go To B"



Checkbox status **has not** been saved

Saving checkbox status using **Preference Activity**

Create “xml” folder under res/ directory and create xml file “mypreferences.xml”



mypreference.xml

```
<PreferenceScreen
   xmlns:android="http://schemas.android.com/apk/res/android">
   <CheckBoxPreference
      android:key="mypreference_checkbox"
      android:title="Checkbox"
      android:defaultValue="false"
      android:summary="Change The CheckBox Status here, it will be
saved in Preference"/>
</PreferenceScreen>
```

activity_layout_b.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_layout_b"
    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"
    android:orientation="vertical"
tools:context="umadevi.example.com.checkbox_preferenceactivity.Activity
B">

</LinearLayout>
```

ActivityB.java

```
public class ActivityB extends PreferenceActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_layout_b);
        addPreferencesFromResource(R.xml.mypreferences);
    }
}
```

activity_main.xml

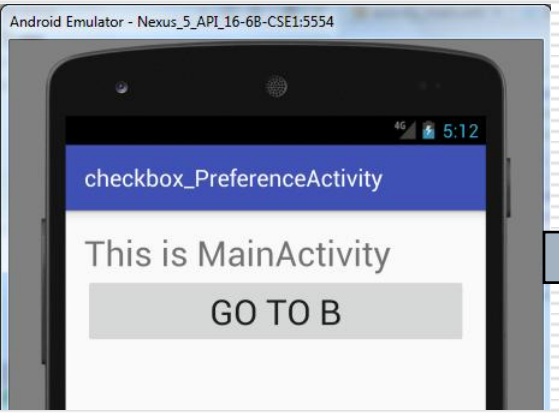
```
<LinearLayout……
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is MainActivity"
        android:textSize="30sp"/>
    <Button
        android:id="@+id/setpreference"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Go To B"
        android:textSize="30sp"
        android:onClick="loadActivityB"/>
</LinearLayout>
```

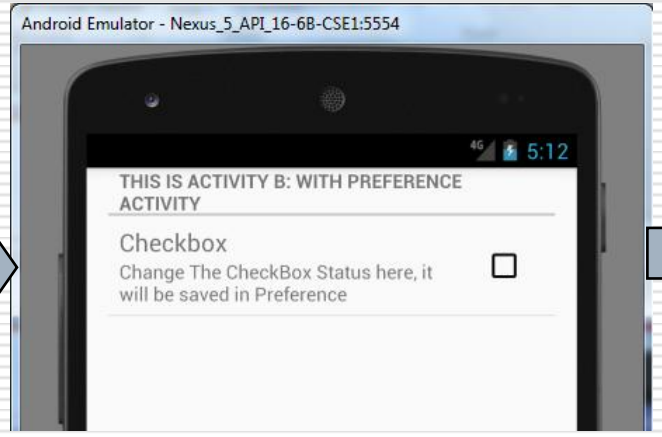
MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void loadActivityB(View v){
        Intent i=new Intent(MainActivity.this, ActivityB.class);
        startActivity(i);
    }
}
```

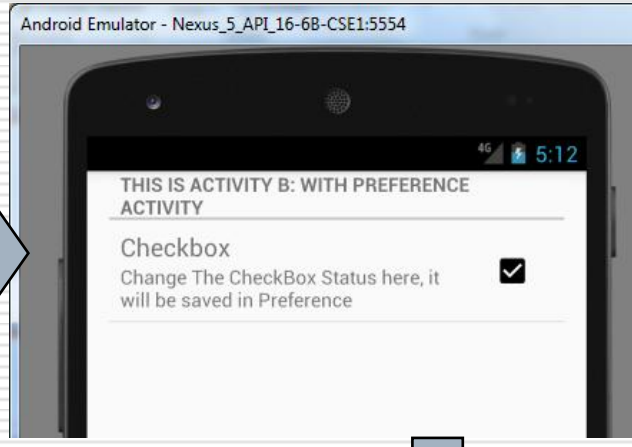
Output

Click on "Go To B"

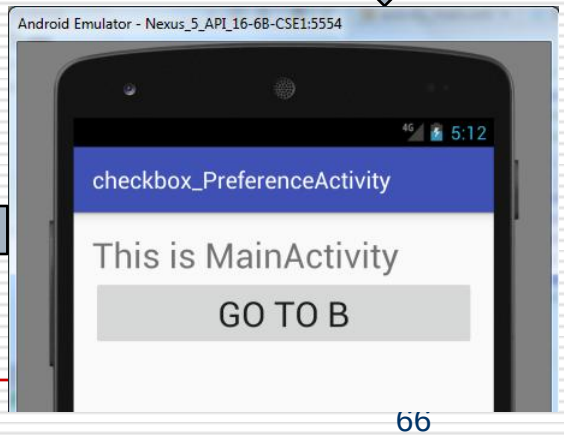




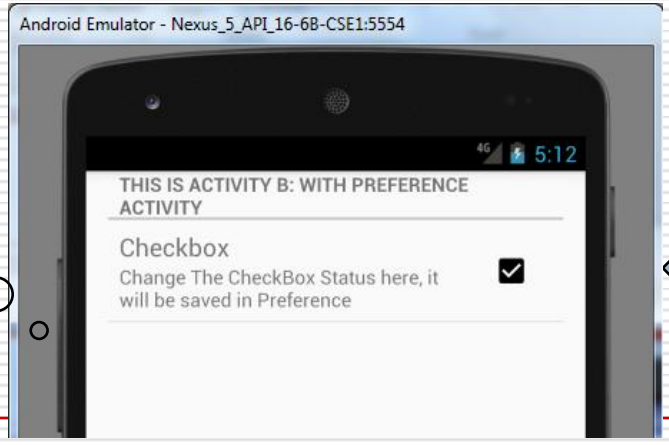
Check the Checkbox



Pressback Button



Click on "Go To B"



Checkbox status has been saved

Difference

```
public class ActivityB extends PreferenceActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
addPreferencesFromResource(R.xml.mypreferences);
    }
}
```

```
public class ActivityB extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_layout_b);
    }
}
```

Difference

mypreferences.xml

```
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
   <CheckBoxPreference
      android:key="mypreference_checkbox"
      android:title="Checkbox"
      android:defaultValue="false"
      android:summary="Change The CheckBox Status here, it will be
saved in Preference"/>
</PreferenceScreen>
```

```
<LinearLaoyout……..
<CheckBox
      android:id="@+id/checkbox"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:defaultValue="false"
      android:text="Check CheckBox status here will not be saved, it will be
lost once activity exit."
      android:textSize="20sp"/>
</LinearLayout>
```

umadevi.example.com.user_preference_settings_preferences.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<boolean name="mypreference_checkbox"
value="true" />
</map>
```

Preference Fragment

- PreferenceFragment class has been used to host the preference screens defined by Preferences Screen resources. To create a new Preference Fragment, extend the PreferenceFragment class, as follows:

```
public class MyPreferenceFragment extends PreferenceFragment
```

- To inflate the preferences, override the onCreate handler and call addPreferencesFromResource,as shown here:

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
addPreferencesFromResource(R.xml.userpreferences);
}
```

- Your application can include several different Preference Fragments, which will be grouped according to the Preference Header hierarchy and displayed within a Preference Activity.

Finding and Using the Shared Preferences Set by Preference Screens

- The Shared Preference values recorded for the options presented in a Preference Activity are stored within the application's sandbox. This lets any application component, including Activities, Services, and Broadcast Receivers, access the values, as shown in the following snippet:

```
Context context = getApplicationContext();
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(context);
// TODO Retrieve values using get<type> methods.
```

- The onSharedPreferenceChangeListener can be implemented to invoke a callback whenever a particular Shared Preference value is added, removed, or modified.
- Register your On Shared Preference Change Listeners using the Shared Preference you want to monitor:

```
public class MyActivity extends Activity implements
OnSharedPreferenceChangeListener {
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
// Register this OnSharedPreferenceChangeListener
SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(this);
prefs.registerOnSharedPreferenceChangeListener(this);
}
public void onSharedPreferenceChanged(SharedPreferences prefs,
String key) {
// TODO Check the shared preference and key parameters
// and change UI or behavior as appropriate.
} }
```

INCLUDING STATIC FILES AS RESOURCES

- If your application requires external file resources, you can include them in your distribution package by placing them in the res/raw folder of your project hierarchy.
- To access these read-only file resources, call the openRawResource method from your application's Resource object to receive an InputStream based on the specified file. Pass in the filename (without the extension) as the variable name from the R.raw class, as shown in the following skeleton code:
- Resources myResources = getResources();
- InputStream myFile = myResources.openRawResource(R.raw.myfilename);

Using Application-Specifi c Folders to Store Files

- Many applications will create or download fi les that are specific to the application. There are two options for storing these application-specific files: internally or externally.
- Android offers two corresponding methods via the application Context, getDir and getExternalFilesDir, both of which return a File object that contains the path to the internal and external application file storage directory, respectively.

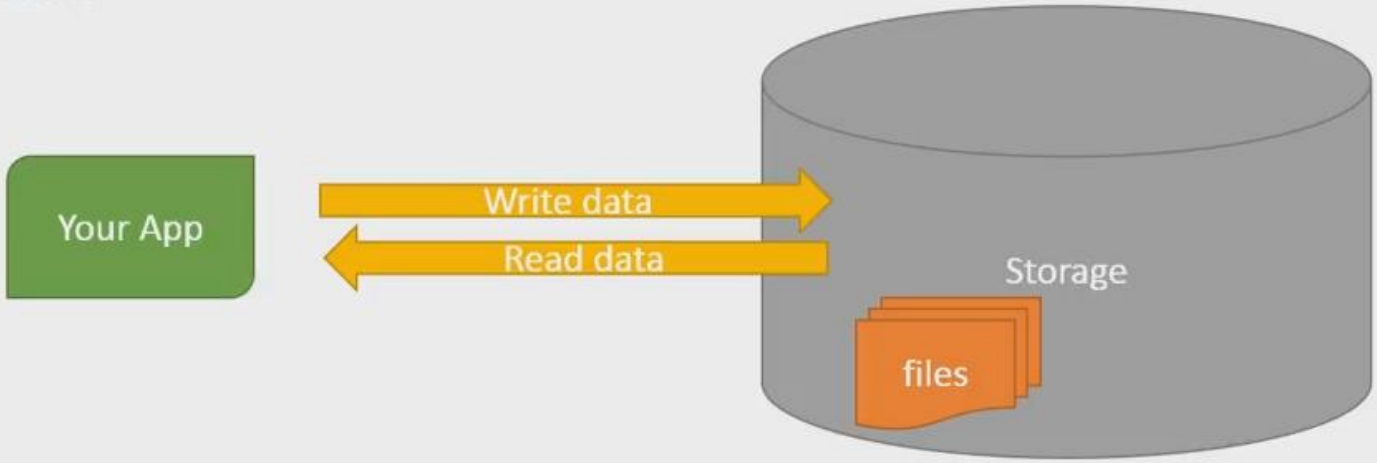
To Do

- Open your Mobile Phones and check
- What are the different memory storage options are available ? What are default storage folders available ?

File Storage on Primary memory (Internal)

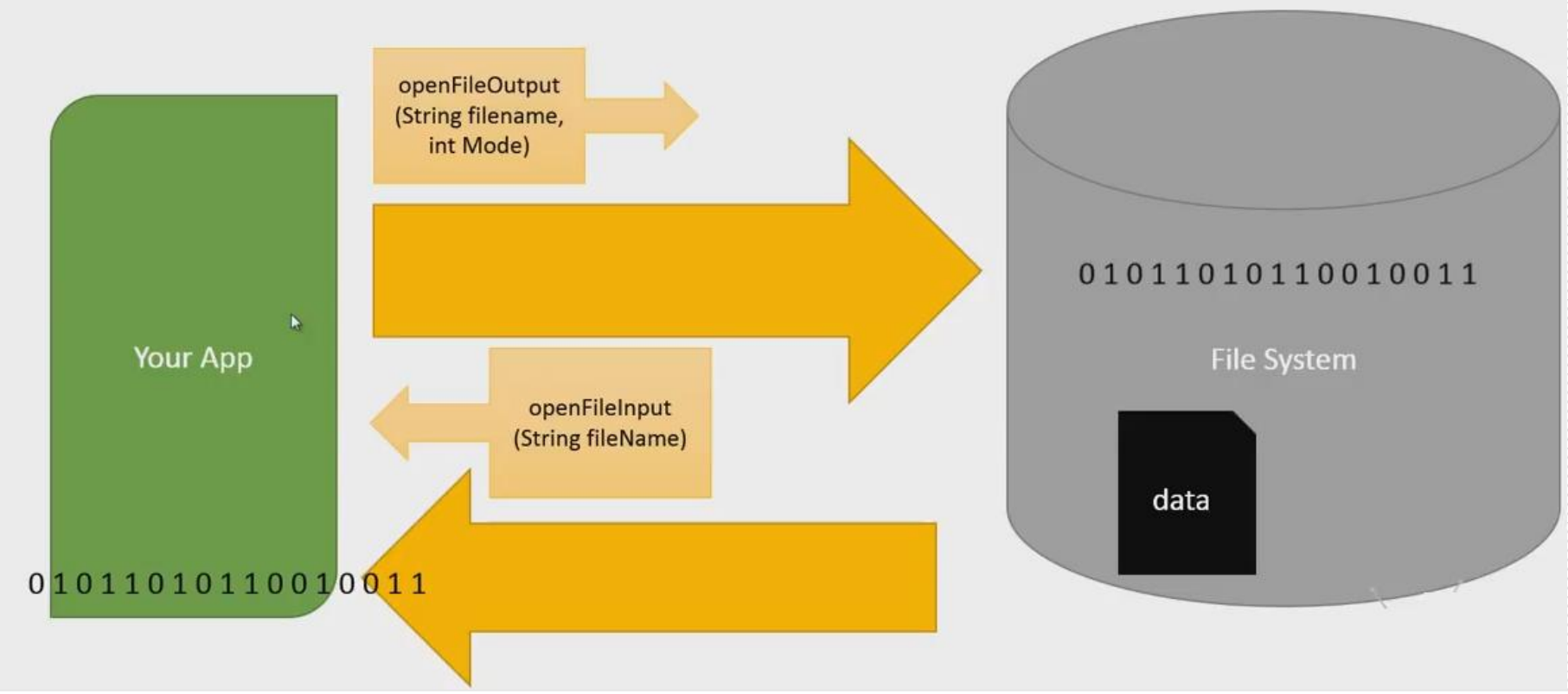
Internal Storage

- Internal allows you to read and write to files that are associated with each application's internal memory.
- These files can only be accessed by the application and cannot be accessed by other applications or users.
- When the application is uninstalled, these files are automatically removed as well.



Internal Storage

- Context.MODE_PRIVATE: cannot be read by others
- Context.MODE_APPEND: keep appending data to the existing contents
- Context.MODE_WORLD_READABLE or Context.MODE_WORLD_WRITEABLE when creating the output fi le, to make it available in other applications



Internal Storage (Contd….)

Writing file

- In order to use internal storage to write some data in the file, call the openFileOutput() method with the name of the file and the mode. The mode could be Context.MODE_PRIVATE, Context.MODE_APPEND, Context.MODE_WORLD_READABLE or Context.MODE_WORLD_WRITEABLE e.t.c. Its syntax is given below –

FileOutputStream fOut = openFileOutput("file name here",Mode);

The method openFileOutput() returns an instance of FileOutputStream. So you receive it in the object of FileInputStream. After that you can call write method to write data on the file. Its syntax is given below –

String str = "data"; fOut.write(str.getBytes()); fOut.close();

Reading file

- In order to read from the file , call the openFileInput() method with the name of the file. It returns an instance of FileInputStream. Its syntax is given below –

FileInputStream fin = openFileInput(file);

After that, you can call read method to read one character at a time from the file and then you can print it. Its syntax is given below –

int c; String temp="";

while((c = fin.read()) != -1){

temp = temp + Character.toString((char)c); }

fin.close();

Internal Storage (Contd….)

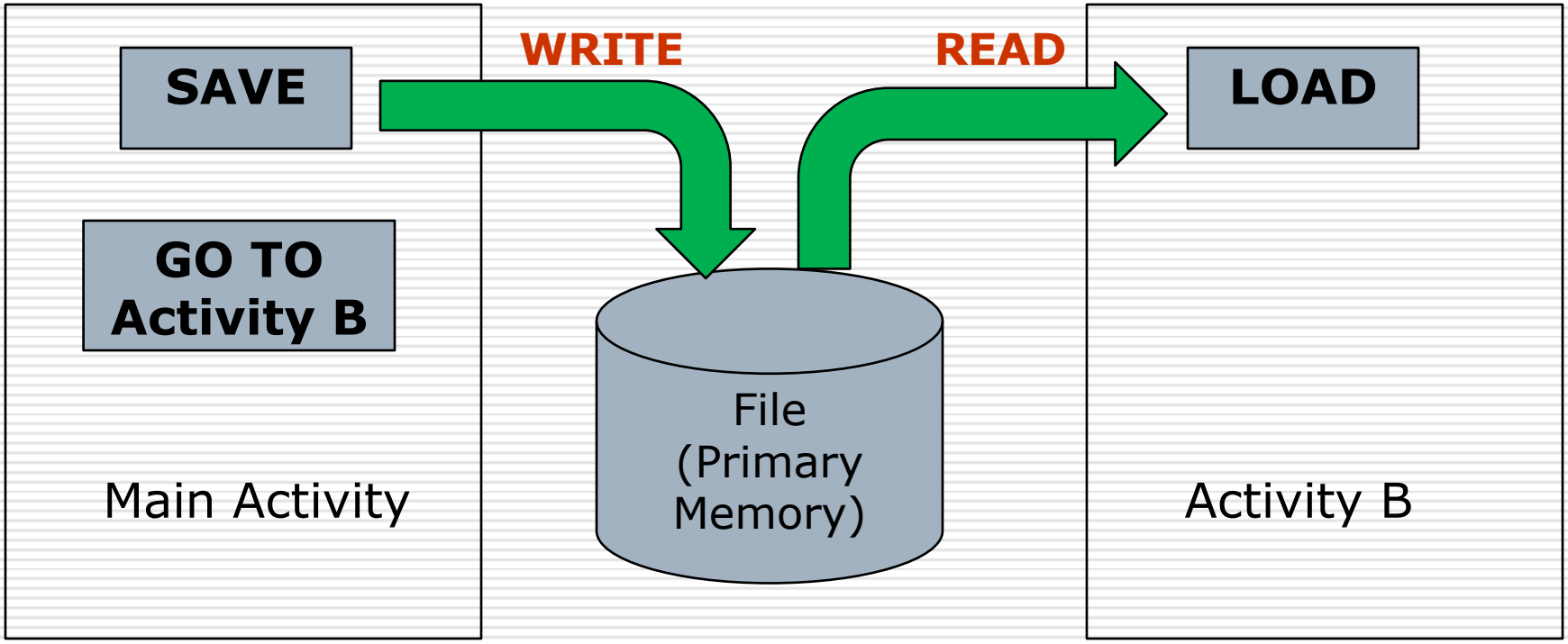
- You can find the location of files stored by calling getFilesDir. This method will return the absolute path to the fi les created using openFileOutput:

```
File file = getFilesDir();
Log.d("OUTPUT_PATH_", file.getAbsolutePath());
```

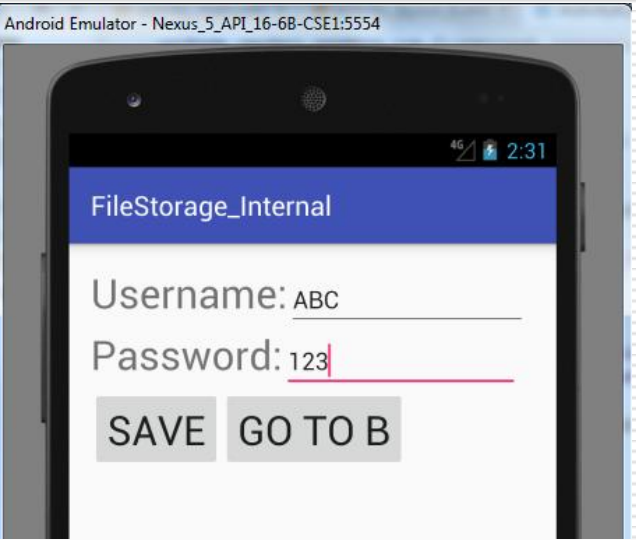
Internal Storage: Writing into File

```
String FILENAME = “mytext.txt";
String string = "hello world!";


FileOutputStream fos = openFileOutput(FILENAME,
Context.MODE_PRIVATE);
byte[] buf=string.getBytes();
fos.write(buf);
fos.close();
```

Example: File Storage Internal

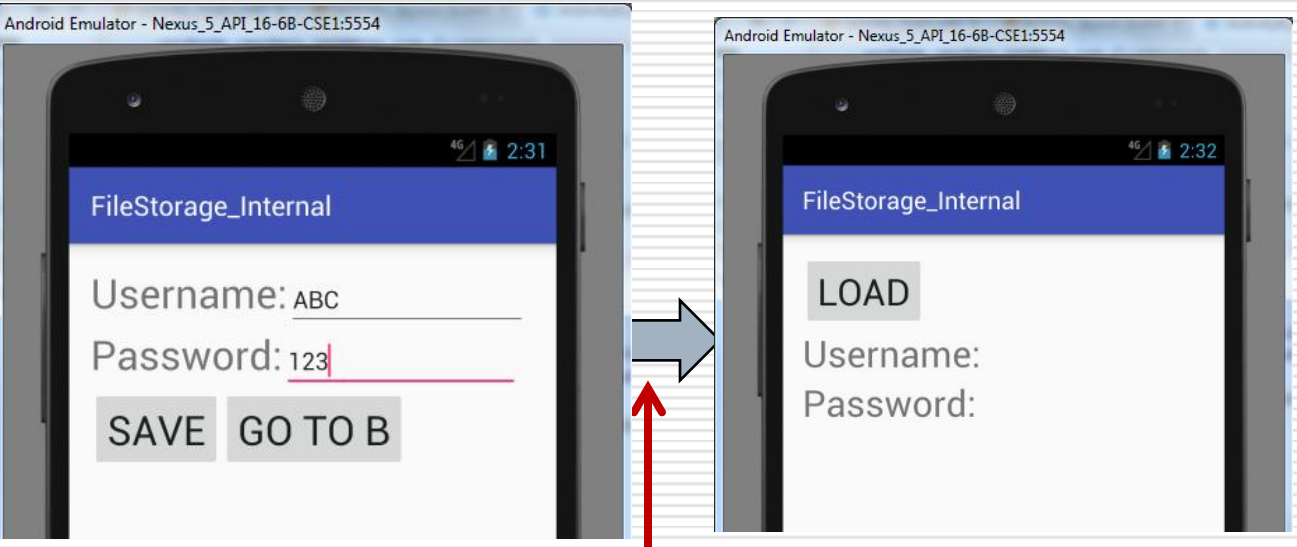


Example: File Storage Internal



MainActivity

Example: File Storage Internal

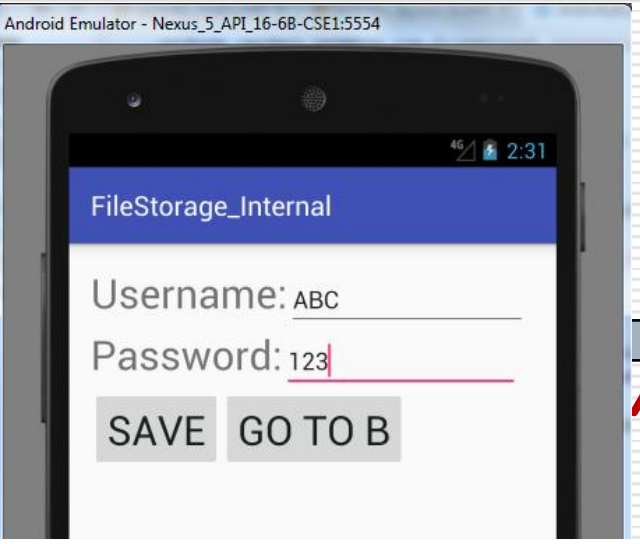


MainActivity

Activity B

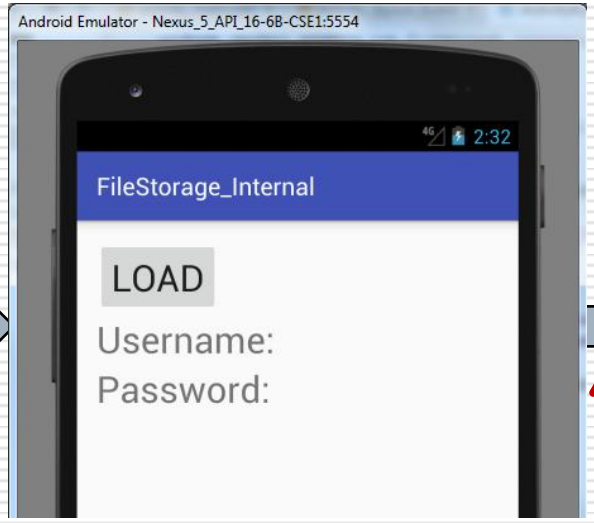
When user Clicks On GOTOB

Example: File Storage Internal



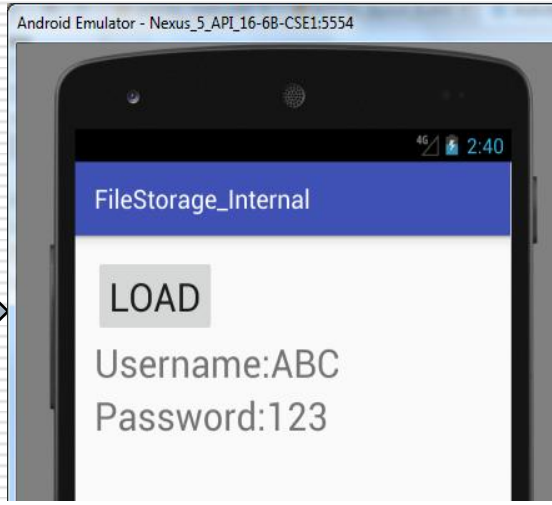
MainActivity

When user Clicks On GOTOB



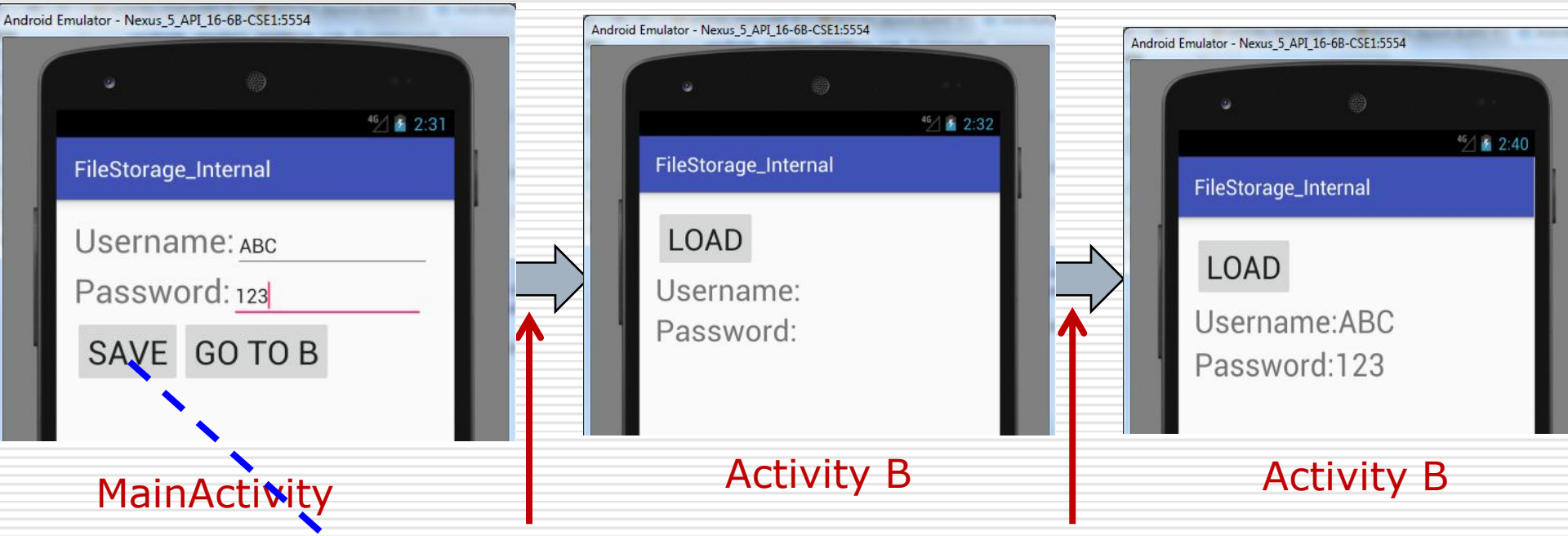
Activity B

When user Clicks On LOAD



Activity B

Example: File Storage Internal



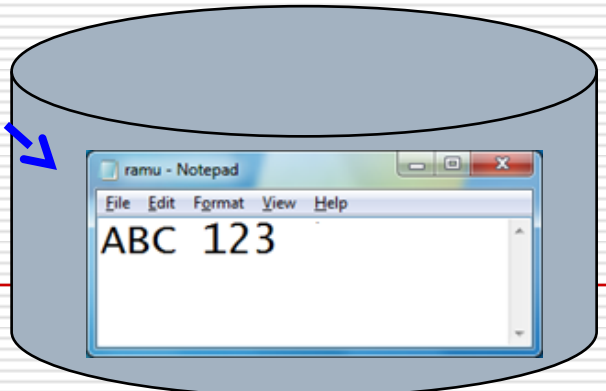
MainActivity

Activity B

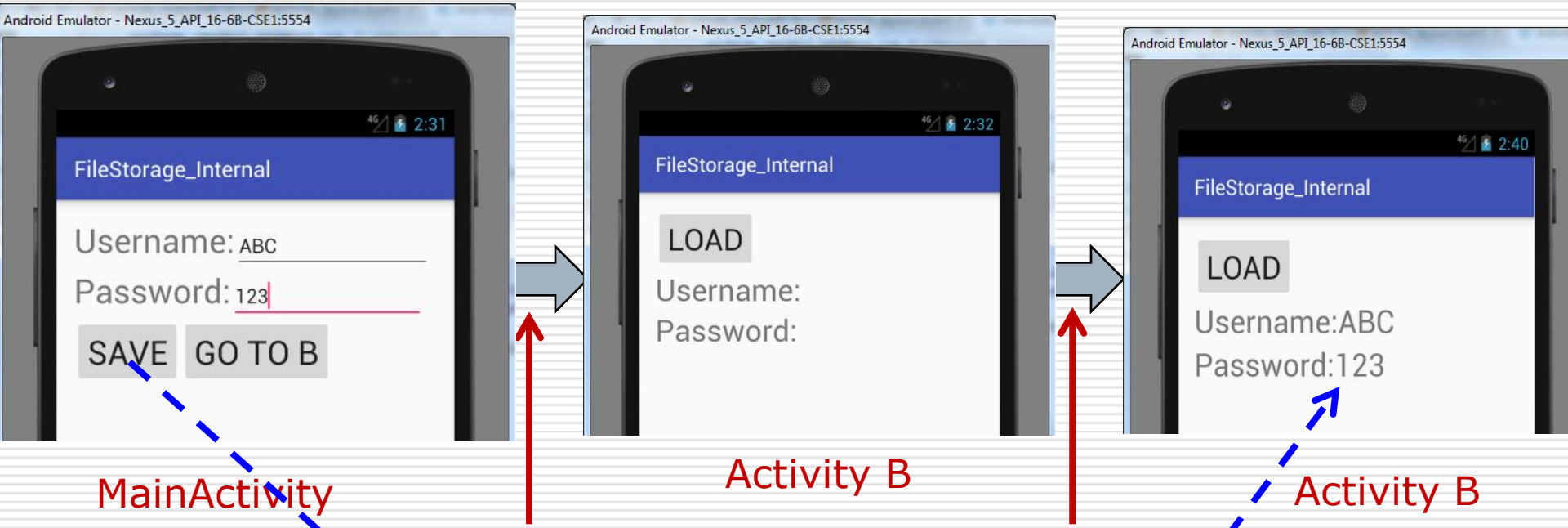
Activity B

When user Clicks On GOTOB

When user Clicks On LOAD



Example: File Storage Internal



activity_main.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_main"
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"
android:orientation="vertical"
tools:context="umadevi.example.com.filestorage_internal.MainActivity">

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/textview"
        android:text="Username:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"/>
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText1"
        android:hint="Enter your Username"/>
</LinearLayout>
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/textview2"
        android:text="Password:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"/>
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Enter your Password"
        android:id="@+id/editText2"/>
</LinearLayout>
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <Button
        android:id="@+id/btnSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:text="Save"
        android:onClick="saveData"/>
    <Button
        android:id="@+id/btnB"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:text="Go to B"
        android:onClick="GoToB"/>
</LinearLayout>
</LinearLayout>
```

activity_layout_b.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_main"
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"
android:orientation="vertical"
tools:context="umadevi.example.com.filestorage_internal.ActivityB">
<Button
   android:id="@+id/btnSave"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:textSize="30sp"
   android:text="Load"
   android:onClick="load"/>
<LinearLayout
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="horizontal">
   <TextView
      android:id="@+id/textview"
      android:text="Username:"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"/>
   <TextView
      android:id="@+id/textviewUsername"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"/>
</LinearLayout>
<LinearLayout
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="horizontal">
   <TextView
      android:id="@+id/textview2"
      android:text="Password:"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"/>
   <TextView
      android:id="@+id/textviewPasswd"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"/>
</LinearLayout>
</LinearLayout>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    EditText userName, passWord;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void saveData(View v) throws IOException {
        userName=(EditText)findViewById(R.id.editText1);
        passWord=(EditText)findViewById(R.id.editText2);
        String uName=userName.getText().toString();
        String pwd=passWord.getText().toString();









}
    public void GoToB(View v)
    {
                        } }
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    EditText userName, passWord;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void saveData(View v) throws IOException {
        userName=(EditText)findViewById(R.id.editText1);
        passWord=(EditText)findViewById(R.id.editText2);
        String uName=userName.getText().toString();
        String pwd=passWord.getText().toString();
        String txt=uName+" "+pwd+"-"; //Hypehen - to find the end of file




}
    public void GoToB(View v)
    {
                    } }
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    EditText userName, passWord;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void saveData(View v) throws IOException {
        userName=(EditText)findViewById(R.id.editText1);
        passWord=(EditText)findViewById(R.id.editText2);
        String uName=userName.getText().toString();
        String pwd=passWord.getText().toString();
        String txt=uName+" "+pwd+"-"; //Hypehen - to find the end of file
        FileOutputStream fileHandler=null;
        fileHandler=openFileOutput("ramu.txt", Context.MODE_PRIVATE);



}
    public void GoToB(View v)
    {
            } }
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    EditText userName, passWord;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void saveData(View v) throws IOException {
        userName=(EditText)findViewById(R.id.editText1);
        passWord=(EditText)findViewById(R.id.editText2);
        String uName=userName.getText().toString();
        String pwd=passWord.getText().toString();
        String txt=uName+" "+pwd+"-"; //Hypehen - to find the end of file
        FileOutputStream fileHandler=null;
        fileHandler=openFileOutput("ramu.txt", Context.MODE_PRIVATE);
        fileHandler.write(txt.getBytes());
        fileHandler.close();
    }
    public void GoToB(View v)
    {
                        } }
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    EditText userName, passWord;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void saveData(View v) throws IOException {
        userName=(EditText)findViewById(R.id.editText1);
        passWord=(EditText)findViewById(R.id.editText2);
        String uName=userName.getText().toString();
        String pwd=passWord.getText().toString();
        String txt=uName+" "+pwd+"-"; //Hypehen - to find the end of file
        FileOutputStream fileHandler=null;
        fileHandler=openFileOutput("ramu.txt", Context.MODE_PRIVATE);
        fileHandler.write(txt.getBytes());
        fileHandler.close();
    }
    public void GoToB(View v)
    {  Intent i=new Intent(MainActivity.this,ActivityB.class);
        startActivity(i);     } }
```

ActivityB.java

```
public class ActivityB extends AppCompatActivity {
……..
public void load(View v) throws IOException {
        TextView uName, pwd;
        uName=(TextView)findViewById(R.id.textviewUsername);
        pwd=(TextView)findViewById(R.id.textviewPasswd);
```

```
} }
```

ActivityB.java

```
public class ActivityB extends AppCompatActivity {
……..
public void load(View v) throws IOException {
        TextView uName, pwd;
        uName=(TextView)findViewById(R.id.textviewUsername);
        pwd=(TextView)findViewById(R.id.textviewPasswd);
        FileInputStream fileHandler=openFileInput("ramu.txt");
```

```
} }
```

ActivityB.java

```
public class ActivityB extends AppCompatActivity {
……..
public void load(View v) throws IOException {
        TextView uName, pwd;
        uName=(TextView)findViewById(R.id.textviewUsername);
        pwd=(TextView)findViewById(R.id.textviewPasswd);
        FileInputStream fileHandler=openFileInput("ramu.txt");
        int read;
        StringBuffer buffer=new StringBuffer();
        while((read=fileHandler.read())!='-'){
            buffer.append((char)read);
        }
```

```
} }
```

ActivityB.java

```
public class ActivityB extends AppCompatActivity {
……..
public void load(View v) throws IOException {
        TextView uName, pwd;
        uName=(TextView)findViewById(R.id.textviewUsername);
        pwd=(TextView)findViewById(R.id.textviewPasswd);
        FileInputStream fileHandler=openFileInput("ramu.txt");
        int read;
        StringBuffer buffer=new StringBuffer();
        while((read=fileHandler.read())!='-'){
            buffer.append((char)read);
        }

        String text1=buffer.substring(0,buffer.indexOf(" "));

        String text2=buffer.substring(buffer.indexOf(" ")+1);



    } }
```

ActivityB.java

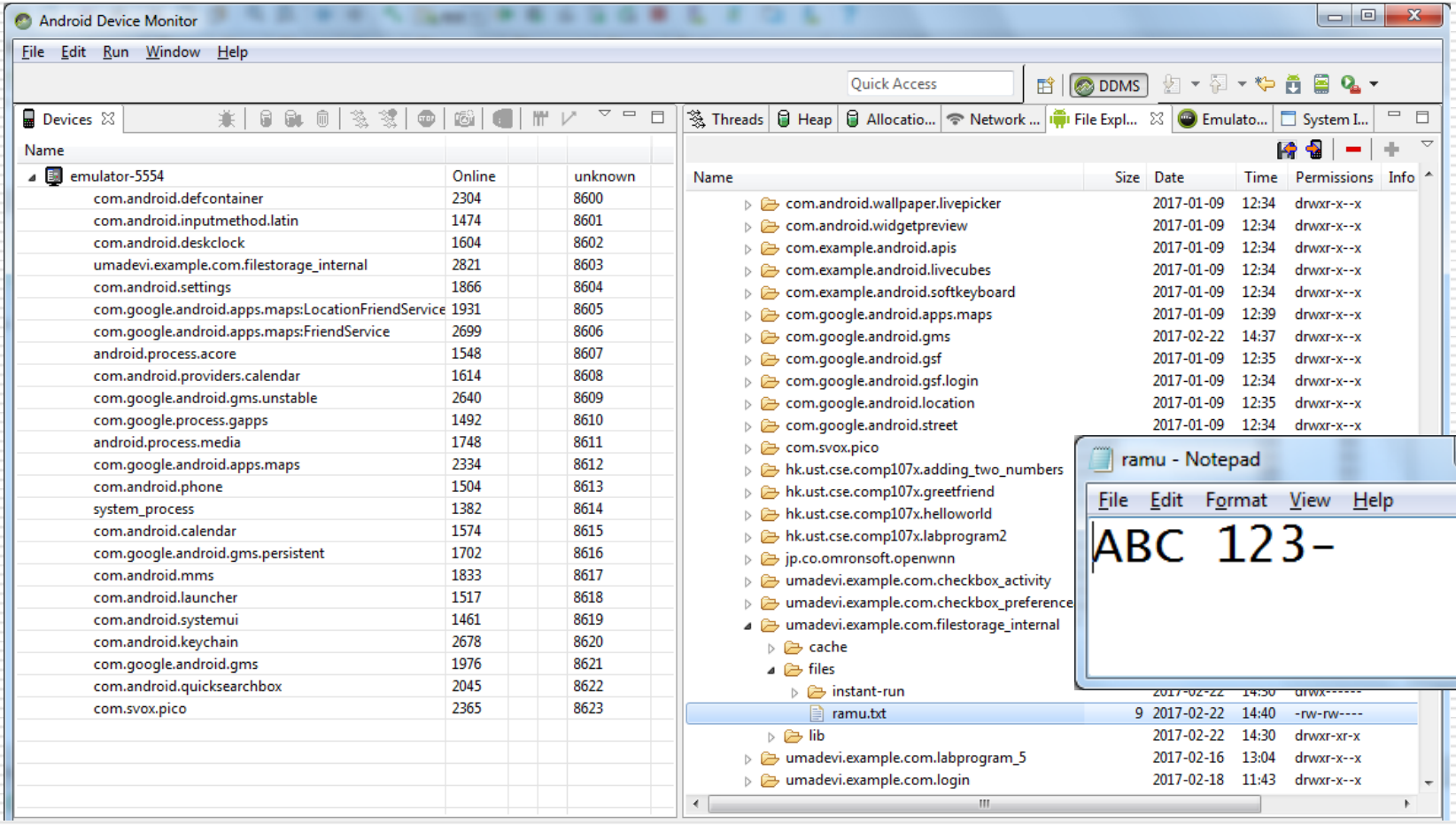
```
public class ActivityB extends AppCompatActivity {
……..
public void load(View v) throws IOException {
        TextView uName, pwd;
        uName=(TextView)findViewById(R.id.textviewUsername);
        pwd=(TextView)findViewById(R.id.textviewPasswd);
        FileInputStream fileHandler=openFileInput("ramu.txt");
        int read;
        StringBuffer buffer=new StringBuffer();
        while((read=fileHandler.read())!='-'){
            buffer.append((char)read);
        }
        //Extract Username
        String text1=buffer.substring(0,buffer.indexOf(" "));
        //Extract Password
        String text2=buffer.substring(buffer.indexOf(" ")+1);


    } }
```

ActivityB.java

```
public class ActivityB extends AppCompatActivity {
……..
public void load(View v) throws IOException {
        TextView uName, pwd;
        uName=(TextView)findViewById(R.id.textviewUsername);
        pwd=(TextView)findViewById(R.id.textviewPasswd);
        FileInputStream fileHandler=openFileInput("ramu.txt");
        int read;
        StringBuffer buffer=new StringBuffer();
        while((read=fileHandler.read())!='-'){
            buffer.append((char)read);
        }
        //Extract Username
        String text1=buffer.substring(0,buffer.indexOf(" "));
        //Extract Password
        String text2=buffer.substring(buffer.indexOf(" ")+1);
        uName.setText(text1);
        pwd.setText(text2);
        fileHandler.close();
    } }
```

Accessing the file “ramu.txt”

- tools -> Android Device Monitor -> clicking on the emulator in the left panel -> file explorer -> data -> data -> com.project-name -> files



Take home assignment

- Consider an android application with following has been created
  - Main activity with the layout having two buttons “SAVE (id button1)” and “Go To Activity B (id button2)”
  - ActivityB with the layout have one button “LOAD (id button3)” and a TextView (id textView1)

You should write the code to do the following

- When user clicks on the button “SAVE” a text “**UTSAV 2017**” should be written into the file “bmsce.txt”
- When user clicks on “Go To Activity B” control should be transferred to “Activity B”
- When user clicks on “LOAD” the data from the file “bmsce.txt” should be read and display the read text onto the TextView

Observe the Difference between Read and Write

```
public class MainActivity extends AppCompatActivity {
    EditText userName, passWord;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void saveData(View v) throws IOException {
        userName=(EditText)findViewById(R.id.editText1);
        passWord=(EditText)findViewById(R.id.editText2);
        String uName=userName.getText().toString();
        String pwd=passWord.getText().toString();
        String txt=uName+" "+pwd+"-"; //Hypehen - to find the end of file
        FileOutputStream fileHandler=null;
        fileHandler=openFileOutput("ramu.txt", Context.MODE_PRIVATE);
        fileHandler.write(txt.getBytes());
        fileHandler.close();
    }
    public void GoToB(View v)
    {  Intent i=new Intent(MainActivity.this,ActivityB.class);
        startActivity(i);     } }
```

Observe the Difference between Read and Write

```
public class ActivityB extends AppCompatActivity {
……..
public void load(View v) throws IOException {
        TextView uName, pwd;
        uName=(TextView)findViewById(R.id.textviewUsername);
        pwd=(TextView)findViewById(R.id.textviewPasswd);
        FileInputStream fileHandler=openFileInput("ramu.txt");
        int read;
        StringBuffer buffer=new StringBuffer();
        while((read=fileHandler.read())!='-'){
            buffer.append((char)read);
        }
        //Extract Username
        String text1=buffer.substring(0,buffer.indexOf(" "));
        //Extract Password
        String text2=buffer.substring(buffer.indexOf(" ")+1);
        uName.setText(text1);
        pwd.setText(text2);
        fileHandler.close();
    } }
```

SharedPreferences vs Internal/External Storage

- **SharedPrefences**
- Storing data internally in XML files.
- More efficient due to less read/write overhead.
- Only primitive types and at most, arrays of strings can be stored

- **Internal/External storage**
- Storing data and files to the phone's external **Secure Digital (SD)** card.
- Less Efficient
- Store complex data, media, images

File Storage on SD card (External)

Internal vs External Storage

- **Internal Storage**
- Less memory available [many MBs , few GBs]
- Access speed is more or less the same
- Only your app can access internal storage data and its deleted when the app is uninstalled

- **External storage**
- More memory available [many GBs]
- Anyone can read the external files and they are not destroyed unless they are located inside the directory returned by **getExternalFilesDir()**

How to write to an External Storage ?

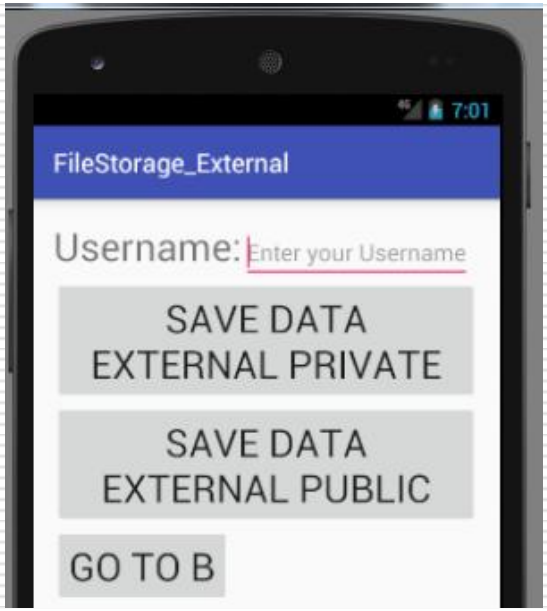
- Add into your manifest file the WRITE_EXTERNAL_STORAGE permission.
- Check the external storage to see if its MOUNTED, READ_ONLY, or failure using **getExternalStorageState()** method
- Get the files directory by calling **getExternalStorageFilesDir(String type)** or **getExternalStoragePublicDirectory(String type)** or **getExternalStorageDirectory()**
- Use a FileInputStream and FileOutputStream to do the required operations

Private vs Public Files

- Files that rightfully belong to your app and should be deleted when the user uninstalls your app. They don't provide value to the user outside your app.
- Example: Other resources downloaded by your app.
- getExternalFilesDir() to store private files on External storage
- Deleted when user uninstalls the app

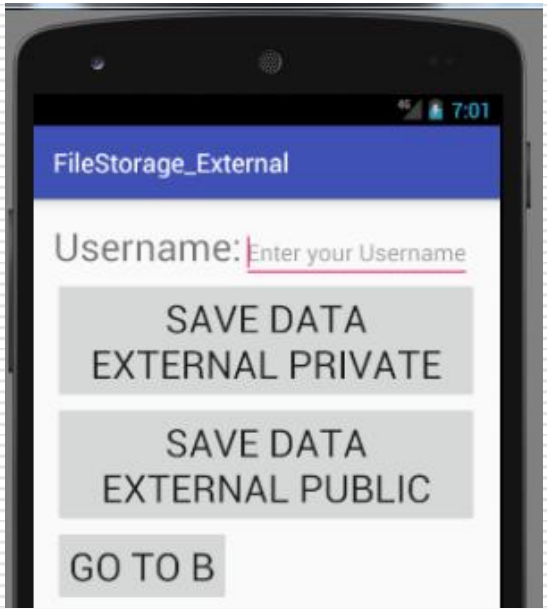
- Files that should be freely available to other apps and to the user.
- Example: photos captured by your app or other downloaded files.
- getExternalStoragePublicDirectory() to store public files on External Storage
- Not deleted

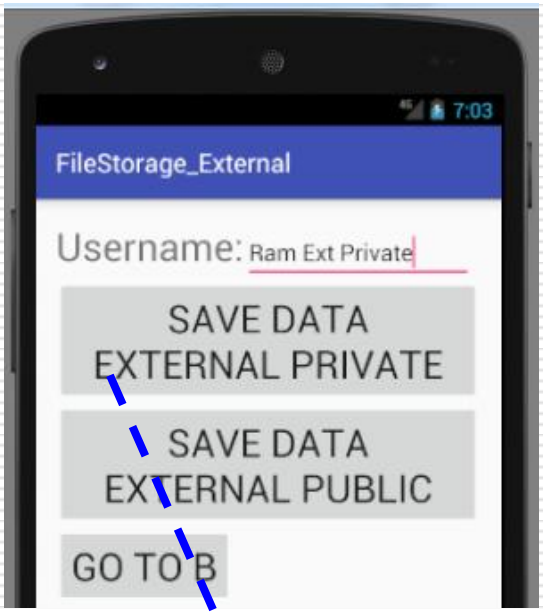
Example: External SD Card - Reading and Writing Files



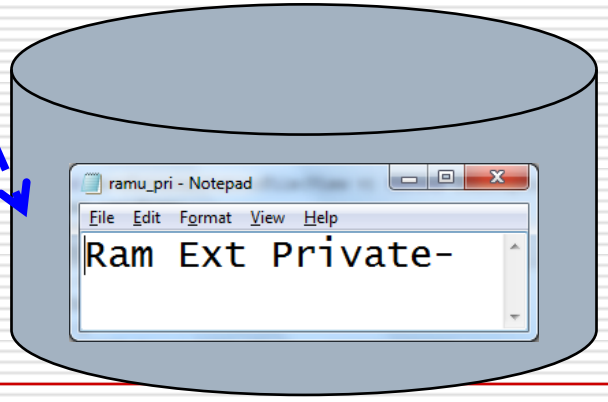
MainActivity

Example: External SD Card - Reading and Writing Files



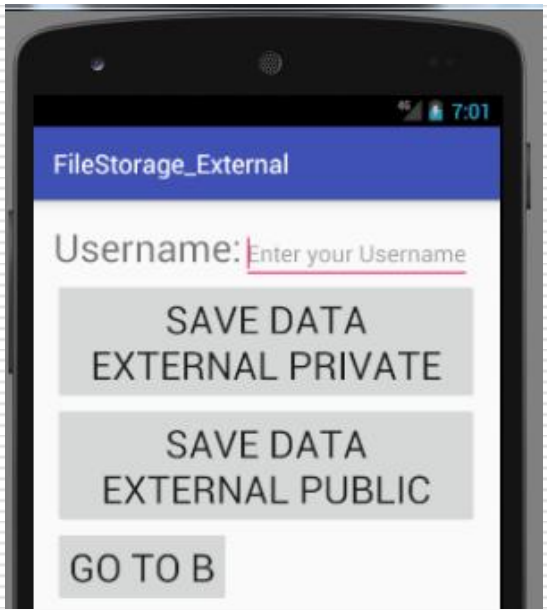


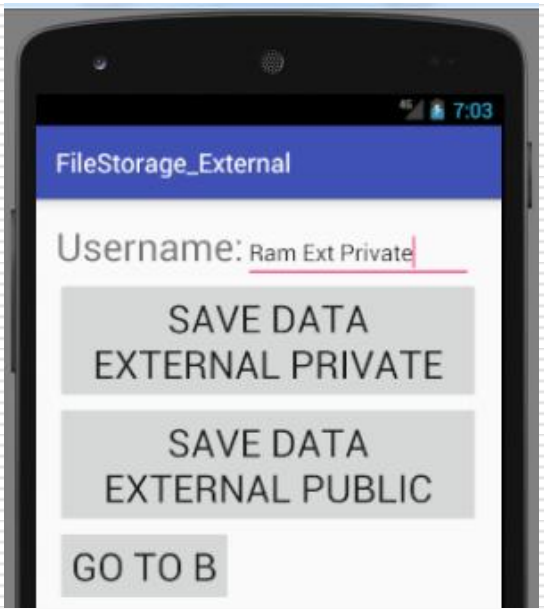
MainActivity

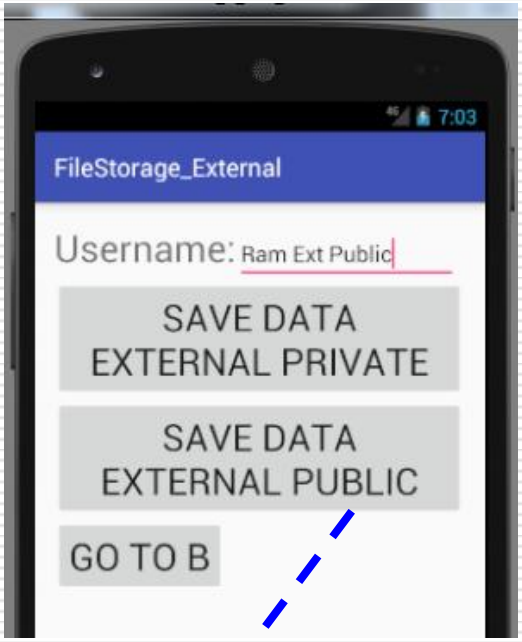


/mnt/sdcard/BMSCE/**ramu_pri.txt**

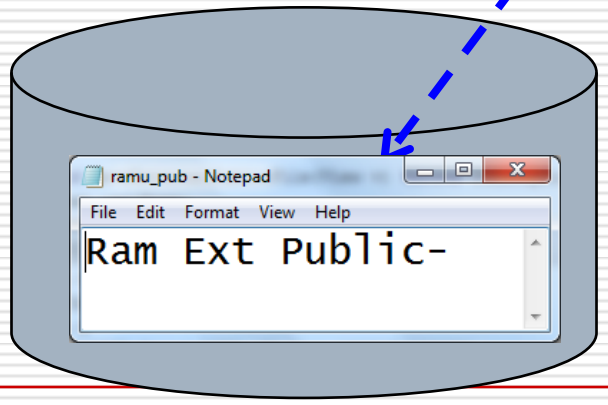
Example: External SD Card - Reading and Writing Files



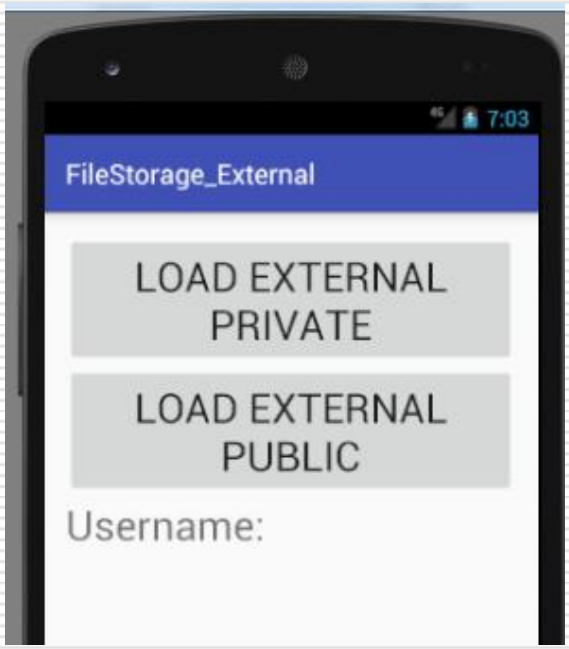




/mnt/sdcard/BMSCE/**ramu_pub.txt**

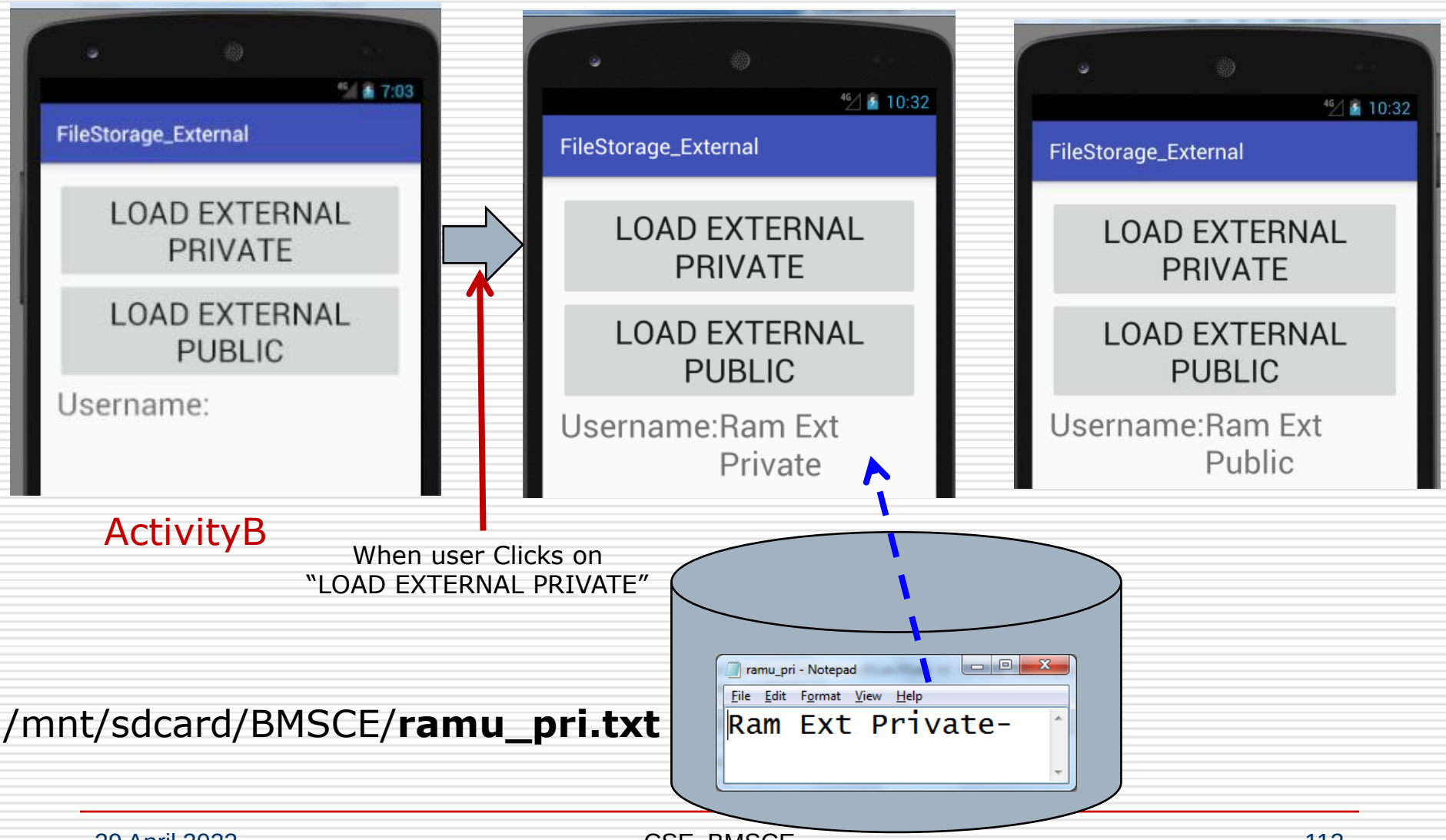


Example: External SD Card - Reading and Writing Files



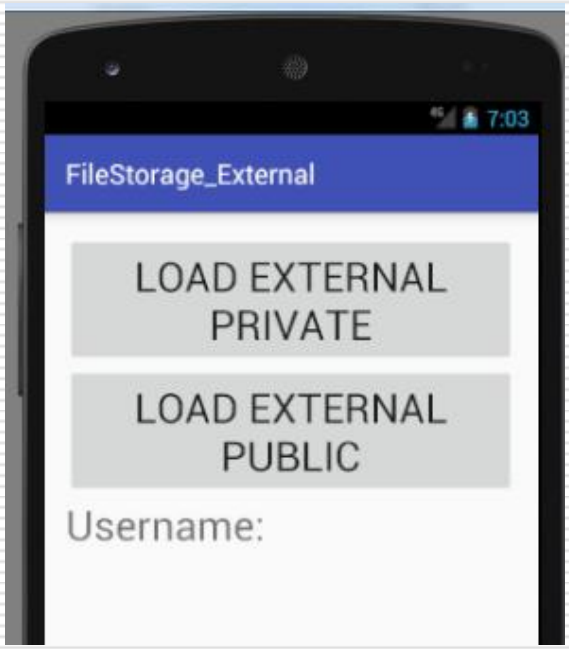
ActivityB

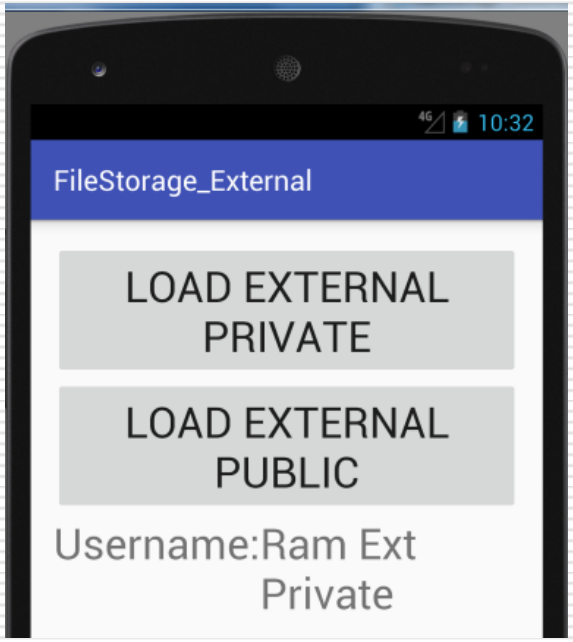
Example: External SD Card - Reading and Writing Files

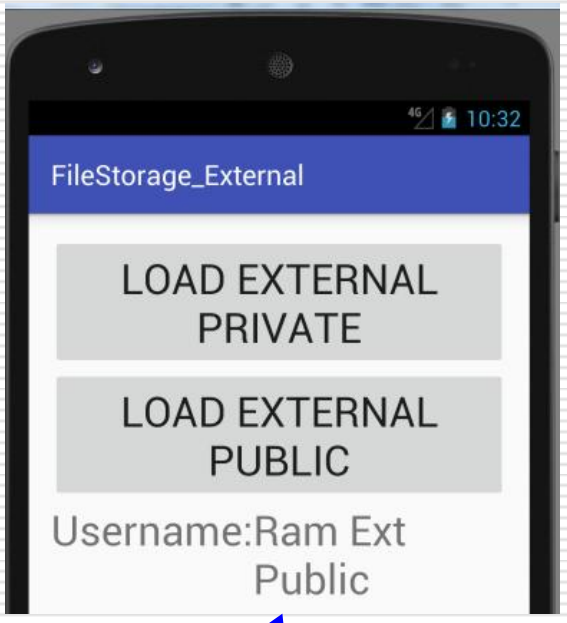


/mnt/sdcard/BMSCE/**ramu_pri.txt**

Example: External SD Card - Reading and Writing Files

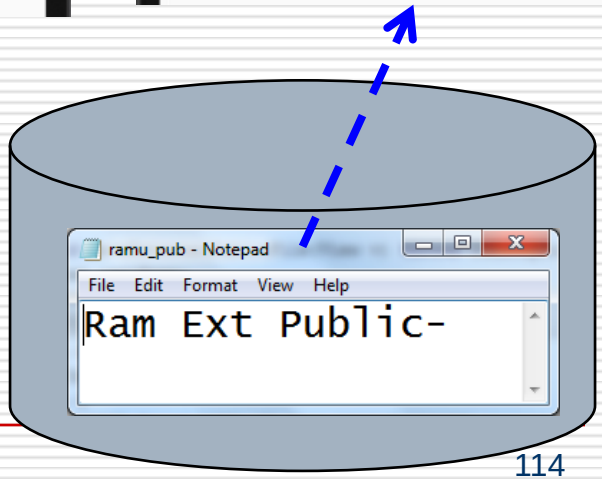






ActivityB

/mnt/sdcard/BMSCE/**ramu_pub.txt**

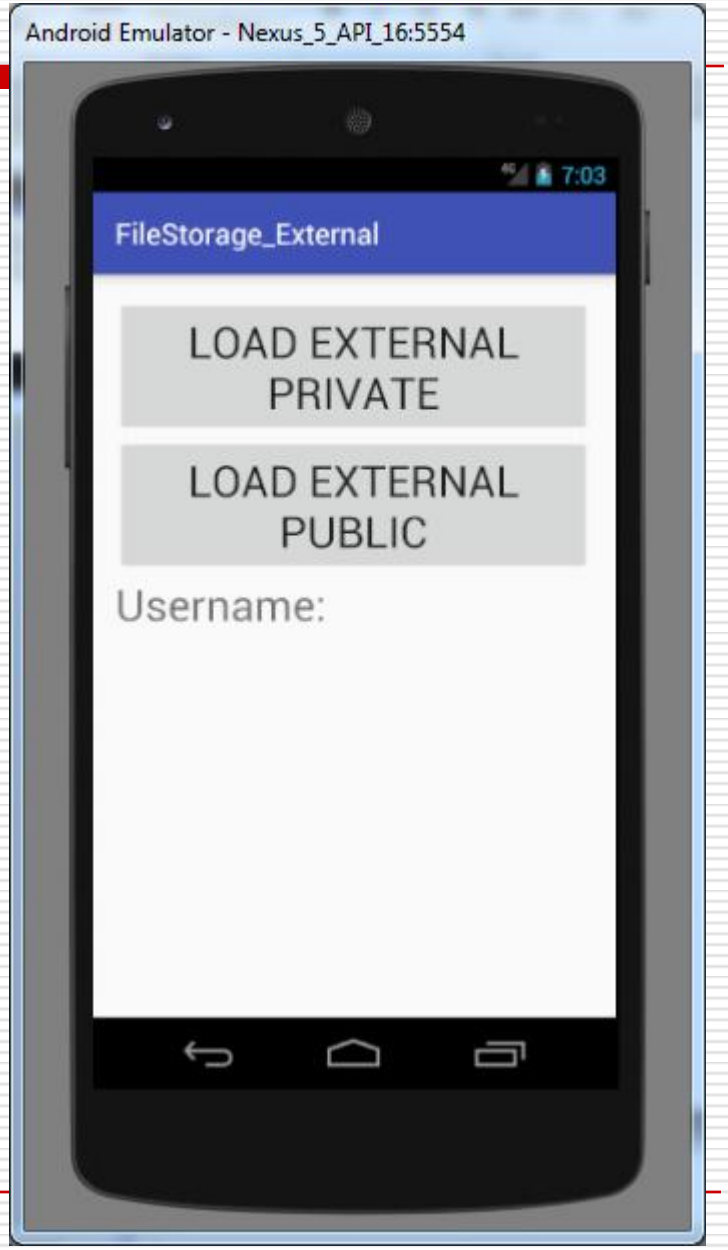


activity_main.xml

```
<LinearLayout ……..
<LinearLayout
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="horizontal">
   <TextView
      android:id="@+id/textview"
      android:text="Username:"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"/>
   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/editText1"
      android:hint="Enter your Username"/>
</LinearLayout>
   <Button
      android:id="@+id/btnSavePrivate"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"
      android:text="Save Data External Private"
      android:onClick="saveExternalPrivateFile"/>
   <Button
      android:id="@+id/btnSavePublic"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"
      android:text="Save Data External Public"
      android:onClick="saveExternalPublicFile"/>
   <Button
      android:id="@+id/btnB"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"
      android:text="Go to B"
      android:onClick="GoToB"/>
</LinearLayout>
```

activity_layout_b.xml

```
<LinearLayout ……..>
<Button
   android:id="@+id/btnLoadPrivate"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:textSize="30sp"
   android:text="Load External Private"
   android:onClick="loadExternalPrivateFile"/>
   <Button
      android:id="@+id/btnLoadPublic"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"
      android:text="Load External Public"
      android:onClick="loadExternalPublicFile"/>
<LinearLayout
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="horizontal">
   <TextView
      android:id="@+id/textview"
      android:text="Username:"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"/>
   <TextView
      android:id="@+id/textviewUsername"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="30sp"/>
</LinearLayout>
</LinearLayout>
```



AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="umadevi.example.com.filestorage_external">
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"
android:maxSdkVersion="18" />

<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
           <action android:name="android.intent.action.MAIN" />
           <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ActivityB"></activity>
    </application>
</manifest>
```

MainActivity.java (External Private)

```
public void saveExternalPrivateFile(View v) throws IOException {
EditText userName;
userName = (EditText) findViewById(R.id.editText1);
String uName = userName.getText().toString();
uName=uName+'-'; //-Hyphen to find the end of file




}

public void writeData(File myFile, String data) throws IOException {



}

public void GoToB(View v)
{
}
```

MainActivity.java (External Private)

```
public void saveExternalPrivateFile(View v) throws IOException {
EditText userName;
userName = (EditText) findViewById(R.id.editText1);
String uName = userName.getText().toString();
uName=uName+'-'; //-Hyphen to find the end of file
File sdCard=Environment.getExternalStorageDirectory();



}

public void writeData(File myFile, String data) throws IOException {




}

public void GoToB(View v)
{
}
```

MainActivity.java (External Private)

```
public void saveExternalPrivateFile(View v) throws IOException {
EditText userName;
userName = (EditText) findViewById(R.id.editText1);
String uName = userName.getText().toString();
uName=uName+'-'; //-Hyphen to find the end of file
File sdCard=Environment.getExternalStorageDirectory();
File directory=new File(sdCard.getAbsolutePath()+"/BMSCE");



}

public void writeData(File myFile, String data) throws IOException {



}

public void GoToB(View v)
{
}
```

MainActivity.java (External Private)

```
public void saveExternalPrivateFile(View v) throws IOException {
EditText userName;
userName = (EditText) findViewById(R.id.editText1);
String uName = userName.getText().toString();
uName=uName+'-'; //-Hyphen to find the end of file
File sdCard=Environment.getExternalStorageDirectory();
File directory=new File(sdCard.getAbsolutePath()+"/BMSCE");
directory.mkdirs();

}

public void writeData(File myFile, String data) throws IOException {



}

public void GoToB(View v)
{

}
```

MainActivity.java (External Private)

```
public void saveExternalPrivateFile(View v) throws IOException {
EditText userName;
userName = (EditText) findViewById(R.id.editText1);
String uName = userName.getText().toString();
uName=uName+'-'; //-Hyphen to find the end of file
File sdCard=Environment.getExternalStorageDirectory();
File directory=new File(sdCard.getAbsolutePath()+"/BMSCE");
directory.mkdirs();
File myFile=new File(directory,"ramu_pri.txt");
writeData(myFile, uName); }

public void writeData(File myFile, String data) throws IOException {



}

public void GoToB(View v)
{

}
```

MainActivity.java (External Private)

```
public void saveExternalPrivateFile(View v) throws IOException {
EditText userName;
userName = (EditText) findViewById(R.id.editText1);
String uName = userName.getText().toString();
uName=uName+'-'; //-Hyphen to find the end of file
File sdCard=Environment.getExternalStorageDirectory();
File directory=new File(sdCard.getAbsolutePath()+"/BMSCE");
directory.mkdirs();
File myFile=new File(directory,"ramu_pri.txt");
writeData(myFile, uName); }

public void writeData(File myFile, String data) throws IOException {
    FileOutputStream fileHandler=null;
    fileHandler=new FileOutputStream(myFile);
    fileHandler.write(data.getBytes());
    fileHandler.close();
}

public void GoToB(View v)
{  Intent i=new Intent(MainActivity.this,ActivityB.class);
    startActivity(i); }
```

Storing Publicly Readable Files

- Environment.**getExternalStoragePublicDirectory**, that can be used to find a path in which to store your application files. The returned location is where users will typically place and manage their own files of each type.

The **getExternalStoragePublicDirectory** method accepts a String parameter that determines which subdirectory you want to access using a series of Environment static constants:

- DIRECTORY_ALARMS — Audio files that should be available as user-selectable alarm sounds
- DIRECTORY_DCIM — Pictures and videos taken by the device
- DIRECTORY_DOWNLOADS — Files downloaded by the user
- DIRECTORY_MOVIES — Movies
- DIRECTORY_MUSIC — Audio files that represent music
- DIRECTORY_NOTIFICATIONS — Audio files that should be available as user-selectable notification sounds
- DIRECTORY_PICTURES — Pictures
- DIRECTORY_PODCASTS — Audio files that represent podcasts
- DIRECTORY_RINGTONES — Audio files that should be available as user-selectable ringtones

MainActivity.java (External Public)

```
public void saveExternalPublicFile(View v) throws IOException {
    EditText userName;
    userName = (EditText) findViewById(R.id.editText1);
    String uName = userName.getText().toString();
    uName=uName+'-'; //-Hyphen to find the end of file
File folder =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File myFile = new File(folder, "ramu_pub.txt");
writeData(myFile, uName);
}

public void writeData(File myFile, String data) throws IOException {
    FileOutputStream fileHandler=null;
    fileHandler=new FileOutputStream(myFile);
    fileHandler.write(data.getBytes());
    fileHandler.close();
}
```

ActivityB.java (Load External Private)

```
public void loadExternalPrivateFile(View v) throws IOException
{
File sdCard = Environment.getExternalStorageDirectory();




}
public void readData(File myFile) throws IOException {




}
```

ActivityB.java (Load External Private)

```
public void loadExternalPrivateFile(View v) throws IOException
{
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath() +
"/BMSCE");



}
public void readData(File myFile) throws IOException {




}
```

ActivityB.java (Load External Private)

```
public void loadExternalPrivateFile(View v) throws IOException
{
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath() +
"/BMSCE");
File myFile = new File(directory, "ramu_pri.txt");



}
public void readData(File myFile) throws IOException {




}
```

ActivityB.java (Load External Private)

```
public void loadExternalPrivateFile(View v) throws IOException
{
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath() +
"/BMSCE");
File myFile = new File(directory, "ramu_pri.txt");
readData(myFile);
}
public void readData(File myFile) throws IOException {









}
```

ActivityB.java (Load External Private)

```
public void loadExternalPrivateFile(View v) throws IOException
{
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath() +
"/BMSCE");
File myFile = new File(directory, "ramu_pri.txt");
readData(myFile);
}
public void readData(File myFile) throws IOException {
TextView uName;
uName=(TextView)findViewById(R.id.textviewUsername);









}
```

ActivityB.java (Load External Private)

```
public void loadExternalPrivateFile(View v) throws IOException
{
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath() +
"/BMSCE");
File myFile = new File(directory, "ramu_pri.txt");
readData(myFile);
}
public void readData(File myFile) throws IOException {
TextView uName;
uName=(TextView)findViewById(R.id.textviewUsername);
FileInputStream fileHandler=new FileInputStream(myFile);
int read;

  StringBuffer buffer=new StringBuffer();
  while((read=fileHandler.read())!= '-'){
       buffer.append((char)read);
    }
.....................}
```

ActivityB.java (Load External Private)

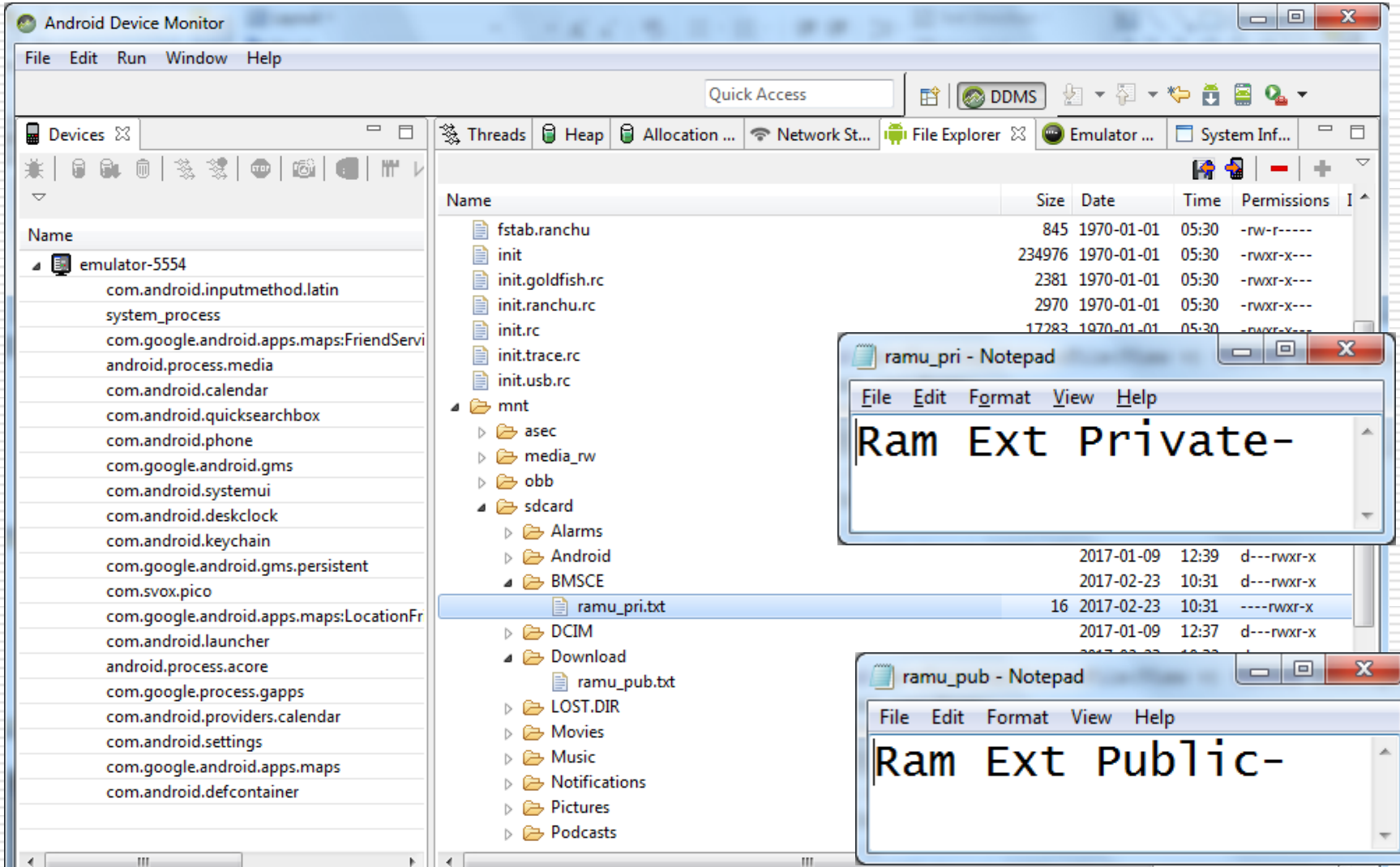
```
public void loadExternalPrivateFile(View v) throws IOException
{
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath() +
"/BMSCE");
File myFile = new File(directory, "ramu_pri.txt");
readData(myFile);
}
public void readData(File myFile) throws IOException {
TextView uName;
uName=(TextView)findViewById(R.id.textviewUsername);
FileInputStream fileHandler=new FileInputStream(myFile);
int read;

  StringBuffer buffer=new StringBuffer();
  while((read=fileHandler.read())!= '-'){
       buffer.append((char)read);
    }
uName.setText(buffer);  fileHandler.close(); }
```

ActivityB.java (Load External Public)

```
public void loadExternalPublicFile(View v) throws IOException {
File sdCard =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTO
RY_DOWNLOADS);
File directory = new File(sdCard.getAbsolutePath());
File myFile = new File(directory, "ramu_pub.txt");
readData(myFile);
}
public void readData(File myFile) throws IOException {
TextView uName;
uName=(TextView)findViewById(R.id.textviewUsername);
FileInputStream fileHandler=new FileInputStream(myFile);
int read;

  StringBuffer buffer=new StringBuffer();
  while((read=fileHandler.read())!= '-'){
       buffer.append((char)read);
    }
uName.setText(buffer);
fileHandler.close(); }
```

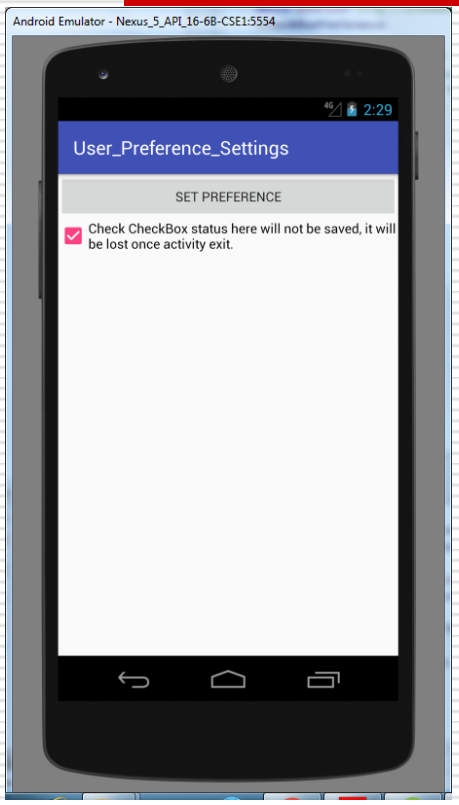
Accessing the files

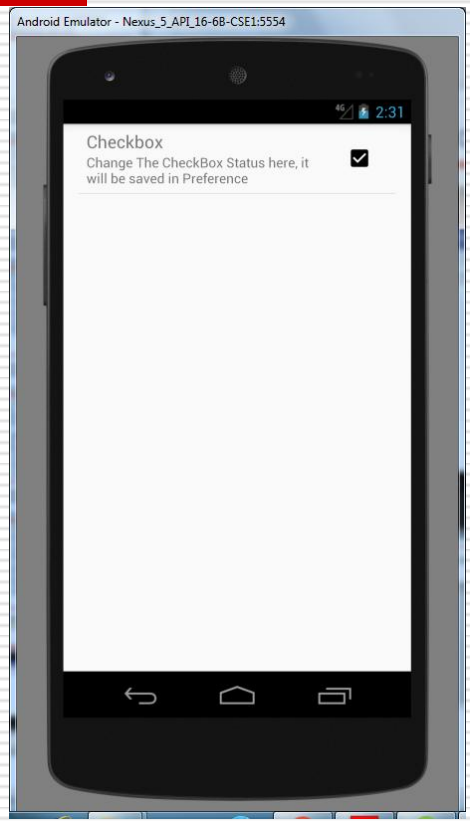
tools -> Android Device Monitor -> clicking on the emulator in the left panel -> file explorer -> mnt -> sdcard

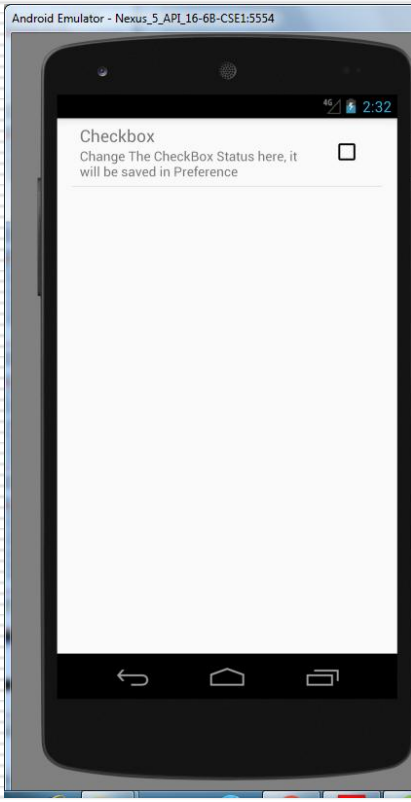


Thanks for Listening

Example: PrefereceActivity w.r.t Checkbox





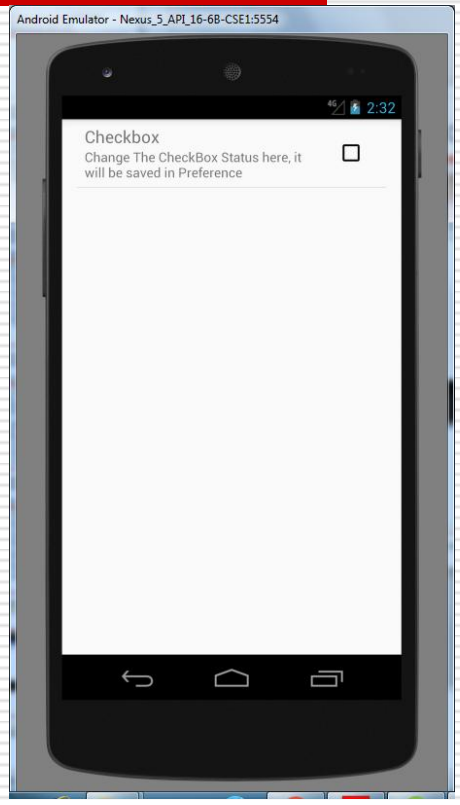


Uncheck the Checkbox and Click on "SET PREFERENCE"

Uncheck the Checkbox and Close the app

Example: PrefereceActivity w.r.t Checkbox

After app restart
MainActivity checkbox
Status will not be remeberd



But after app restart
"SetPreference" activity
Checkbox status
Will be remembered

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"    >
      <Button
       android:id="@+id/setpreference"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:text="Set Preference"  />
   <CheckBox
       android:id="@+id/checkbox"
       android:layout_width="match_parent"
       android:layout_height="100dp"
       android:text="Check CheckBox status here will not be saved, it will be lost once activity exit."
       android:textSize="16sp"/>
</LinearLayout>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    CheckBox checkBox;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button buttonSetPreference = (Button)findViewById(R.id.setpreference);
        checkBox = (CheckBox)findViewById(R.id.checkbox);

        buttonSetPreference.setOnClickListener(new Button.OnClickListener(){

            @Override
            public void onClick(View arg0) {
                // TODO Auto-generated method stub
                startActivity(new Intent(MainActivity.this, SetPreference.class));
            }});
    }

    @Override
    protected void onResume() {
      super.onResume();

SharedPreferences myPreference=
PreferenceManager.getDefaultSharedPreferences(MainActivity.this);
checkBox.setChecked(myPreference.getBoolean("checkbox", true));
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="umadevi.example.com.user_preference_settings">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <activity android:name=".MainActivity">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <activity android:name=".SetPreference"></activity>
   </application>

</manifest>
```

mypreference.xml

```
<PreferenceScreen
   xmlns:android="http://schemas.android.com/apk/res/android">
   <CheckBoxPreference
      android:key="preference_checkbox"
      android:title="Checkbox"
      android:defaultValue="true"
      android:summary="Change The CheckBox Status here, it will be
saved in Preference"/>
</PreferenceScreen>
```

Activity_layout_set_preference.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_layout_set_preference"
    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="umadevi.example.com.user_preference_settings.SetPreference">

</RelativeLayout>
```

SetPreference.java

```
public class SetPreference extends PreferenceActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.mypreference);
    }
}
```

umadevi.example.com.user_preference_settings_preferences.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<boolean name="preference_checkbox"
value="false" />
</map>
```