CIT 4404 Mobile App Development

Topic4: Data Persistence

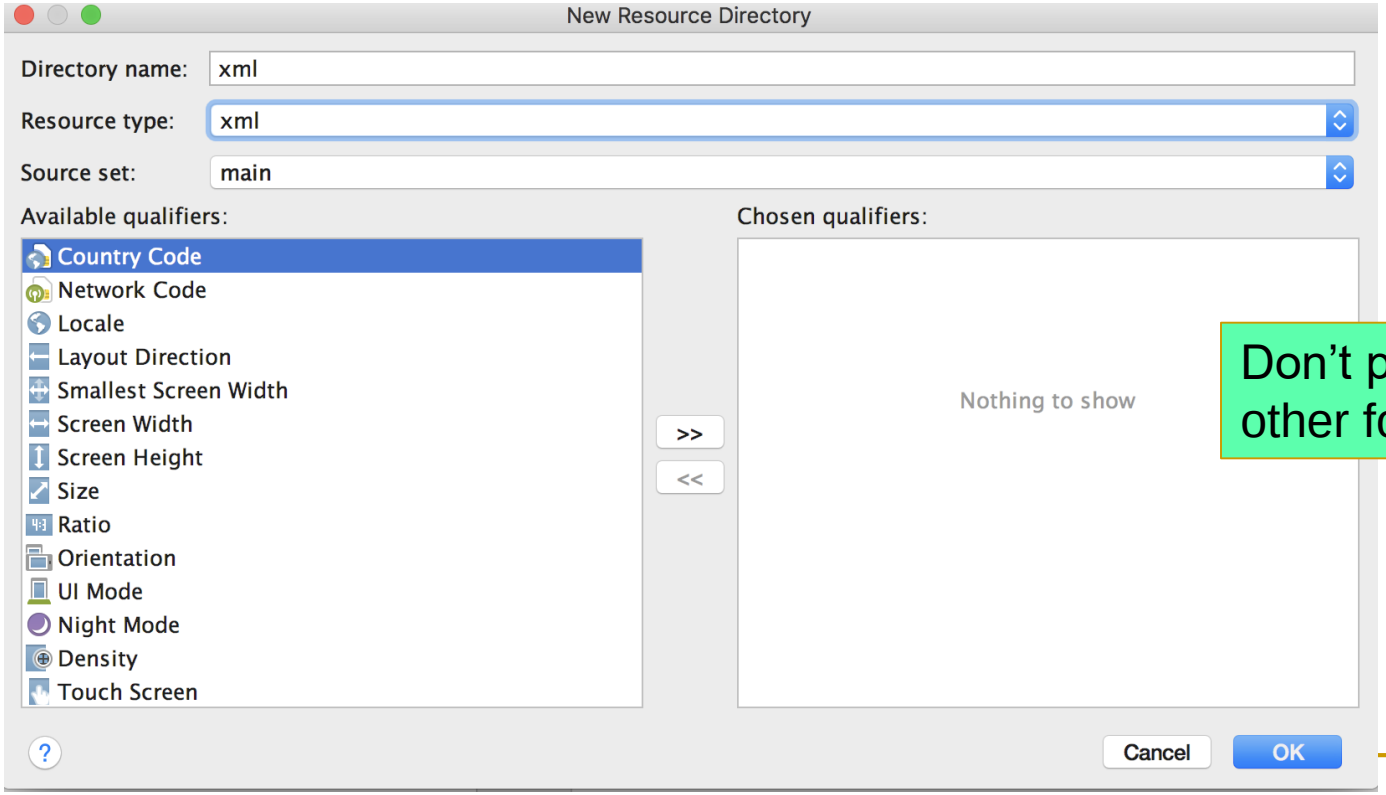
Dr. Fullgence Mwakondo
Institute of Computing and Informatics
Technical University of Mombasa
mwakondo@tum.ac.ke

Data Persistence

- How to save simple data using the SharedPreferences object
- How to enable users to modify preferences using a PreferenceActivity class
- How to write and read files in internal and external storage
- How to create and use a SQLite database

Storing/Accessing Preferences Using an Activity

- Create a new Android project and name it UsingPreferences
- Create a new subdirectory in the res directory and name it **xml**, then create new a xml file and name it myapppreferences.xml



Don't put the file in any other folders

Storing/Accessing Preferences Using an Activity

- Populate the myapppreferences.xml as follows
  - Two preference categories for grouping different types of preferences
  - Two check box preferences with keys named checkboxPref and secondEditTextPref
  - A ringtone preference with a key named ringtonePref
  - A preference screen to contain additional preferences

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
   xmlns:android="http://schemas.android.com/apk/res/android">
   <PreferenceCategory android:title="Category 1">
      <CheckBoxPreference
         android:title="Checkbox"
         android:defaultValue="false"
         android:summary="True or False"
         android:key="checkboxPref" />
   </PreferenceCategory>
   <PreferenceCategory android:title="Category 2">
      <EditTextPreference
         android:summary="Enter a string"
         android:defaultValue="[Enter a string here]"
         android:title="Edit Text"
         android:key="editTextPref" />
      <RingtonePreference
         android:summary="Select a ringtone"
         android:title="Ringtones"
         android:key="ringtonePref" />
      <PreferenceScreen
         android:title="Second Preference Screen"
         android:summary=
            "Click here to go to the second Preference Screen"
         android:key="secondPrefScreenPref" >
         <EditTextPreference
            android:summary="Enter a string"
            android:title="Edit Text (second Screen)"
            android:key="secondEditTextPref" />
      </PreferenceScreen>
   </PreferenceCategory>
</PreferenceScreen>
```

The android:key attribute specifies the key that you can programmatically reference in your code to set or retrieve the value of that particular preference

Storing/Accessing Preferences Using an Activity

- Another xml file in the xml directory and name it prefheaders.xml

```
<?xml version="1.0" encoding="utf-8"?>
<preference-headers
   xmlns:android="http://schemas.android.com/apk/res/android">
   <header android:fragment=
      "com.wenbing.usingpreferences.AppPreferenceActivity$PrefFragment"
      android:title="Preferences"
      android:summary="Sample preferences" />
</preference-headers>
```

Storing/Accessing Preferences Using an Activity

- Add a new Java class and name it AppPreferenceActivity, and populate it as following

```
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceFragment;
import android.preference.PreferenceManager;
import java.util.List;

public class AppPreferenceActivity extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
    @Override
    public void onBuildHeaders(List<Header> target) {
        loadHeadersFromResource(R.xml.prefheaders, target);
    }
    @Override
    protected boolean isValidFragment(String fragmentName) {
        return true;
    }
    public static class PrefFragment extends PreferenceFragment {
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            PreferenceManager.setDefaultValues(getActivity(),
                    R.xml.myapppreferences, false);
             // Load the preferences from an XML resource
            addPreferencesFromResource(R.xml.myapppreferences);
        }
    }
}
```

Storing/Accessing Preferences Using an Activity

- In the AndroidManifest.xml file, add the new entry for the AppPreferenceActivity class

```
<activity
   android:name="com.wenbing.usingpreferences.AppPreferenceActivity"
   android:label="@string/app_name" >
   <intent-filter>
      <action android:name="com.wenbing.AppPreferenceActivity" />
      <category android:name="android.intent.category.DEFAULT" />
   </intent-filter>
</activity>
```

Storing/Accessing Preferences Using an Activity

- In the activity_main.xml file, replace it with the following:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.wenbing.usingpreferences.MainActivity">

    <Button
        android:text="Load Preferences Screen"
        android:layout_width="310dp"
        android:layout_height="wrap_content"
        android:id="@+id/btnPreferences"
        app:layout_constraintLeft_toLeftOf="@+id/activity_main"
        android:layout_marginStart="40dp"
        app:layout_constraintTop_toTopOf="@+id/activity_main"
        android:layout_marginTop="16dp"
        app:layout_constraintRight_toRightOf="@+id/activity_main"
        android:layout_marginEnd="16dp"
        app:layout_constraintBottom_toBottomOf="@+id/activity_main"
        android:layout_marginBottom="16dp"
        app:layout_constraintVertical_bias="0.0"
        android:onClick="onClickLoad"/>
```

```
    <Button
        android:text="Display Preferences Values"
        android:layout_width="310dp"
        android:layout_height="wrap_content"
        android:id="@+id/btnDisplayValues"
        app:layout_constraintLeft_toLeftOf="@+id/btnPreferences"
        app:layout_constraintTop_toBottomOf="@+id/btnPreferences"
        android:layout_marginTop="16dp"
        app:layout_constraintRight_toRightOf="@+id/btnPreferences"
        android:onClick="onClickDisplay"/>
    <EditText
        android:layout_width="310dp"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/editText"
        app:layout_constraintLeft_toLeftOf="@+id/btnPreferences"
        app:layout_constraintTop_toBottomOf="@+id/btnDisplayValues"
        android:layout_marginTop="16dp"
        app:layout_constraintRight_toRightOf="@+id/btnPreferences" />
    <Button
        android:text="Modify Preferences Values"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btnModifyValues"
        app:layout_constraintLeft_toLeftOf="@+id/btnDisplayValues"
        app:layout_constraintTop_toBottomOf="@+id/editText"
        android:layout_marginTop="16dp"
        app:layout_constraintRight_toRightOf="@+id/btnDisplayValues"
        android:onClick="onClickModify" />

</android.support.constraint.ConstraintLayout>
```

Storing/Accessing Preferences Using an Activity

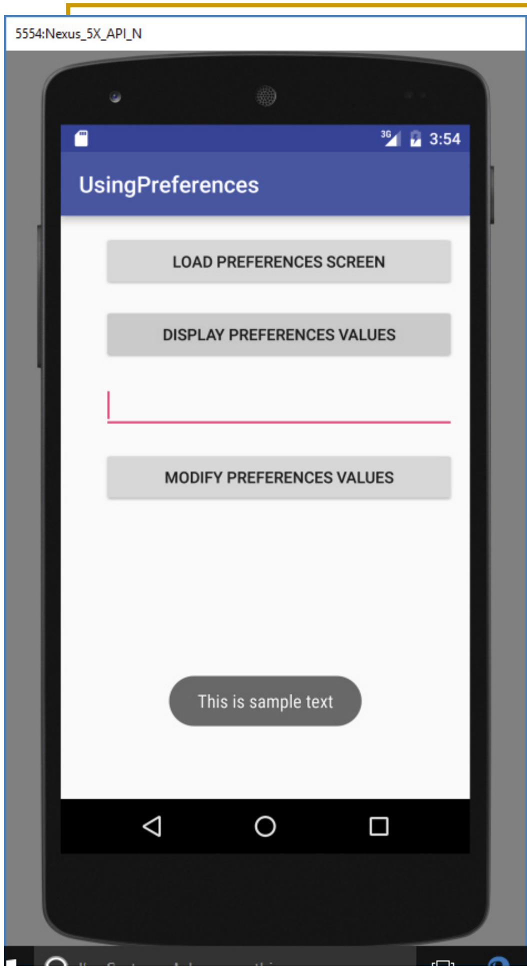
- In the MainActivity.java file, replace it with the following:

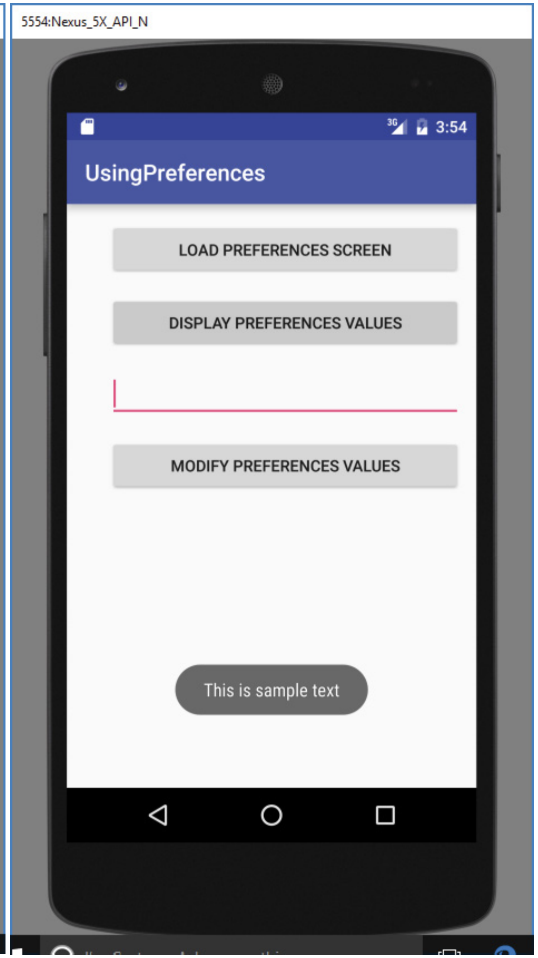
```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClickLoad(View view) {
        Intent i = new Intent("com.wenbing.AppPreferenceActivity");
        startActivity(i);
    }
    public void onClickDisplay(View view) {
        SharedPreferences appPrefs =
                getSharedPreferences("com.wenbing.usingpreferences_preferences", MODE_PRIVATE);
        DisplayText(appPrefs.getString("editTextPref", ""));
    }
    public void onClickModify(View view) {
        SharedPreferences appPrefs =
                getSharedPreferences("com.wenbing.usingpreferences_preferences", MODE_PRIVATE);
        SharedPreferences.Editor prefsEditor = appPrefs.edit();
        prefsEditor.putString("editTextPref", ((EditText) findViewById(R.id.editText)).getText().toString());
        prefsEditor.commit();
    }
    private void DisplayText(String str) {
        Toast.makeText(getBaseContext(), str, Toast.LENGTH_LONG).show();
    }
}
```

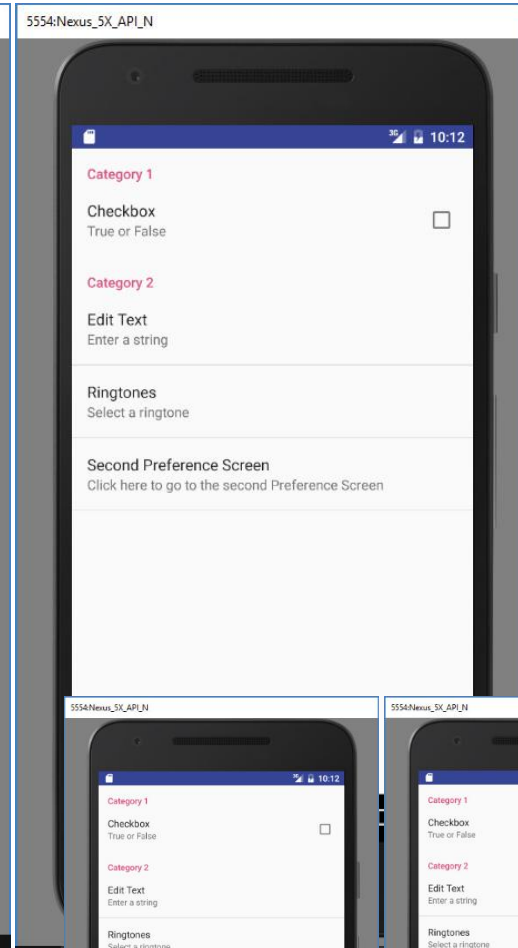
```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.content.Intent;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import android.content.SharedPreferences;
```

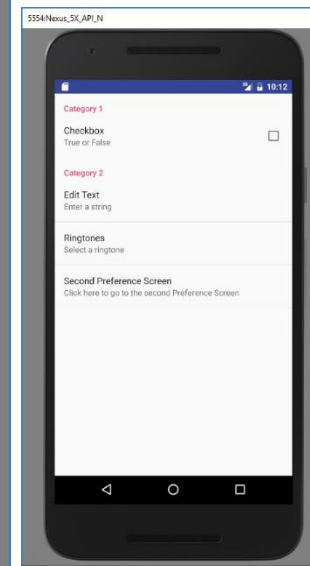
The MODE_PRIVATE constant indicates that the preference file can be opened only by the application that created it

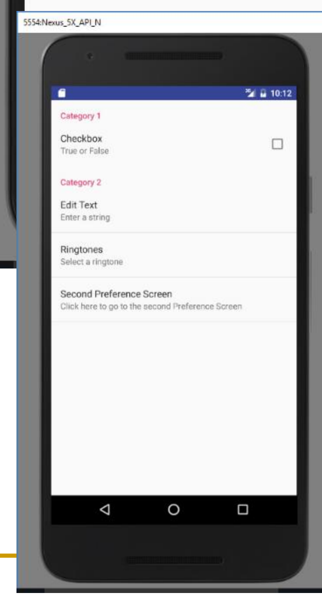
The android:key

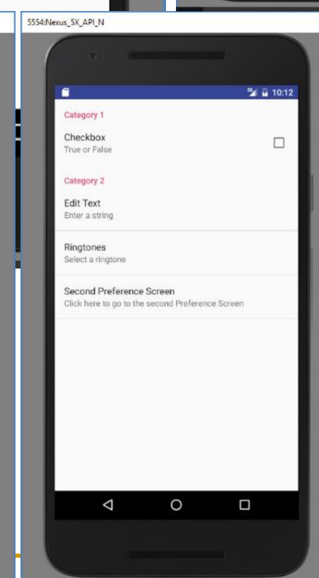

5554:Nexus_5X_API_N
3:54
UsingPreferences
LOAD PREFERENCES SCREEN
DISPLAY PREFERENCES VALUES
MODIFY PREFERENCES VALUES
This is sample text


5554:Nexus_5X_API_N
3:54
UsingPreferences
LOAD PREFERENCES SCREEN
DISPLAY PREFERENCES VALUES
MODIFY PREFERENCES VALUES
This is sample text


5554:Nexus_5X_API_N
10:12
Category 1
Checkbox
True or False
Category 2
Edit Text
Enter a string
Ringtones
Select a ringtone
Second Preference Screen
Click here to go to the second Preference Screen


5554:Nexus_5X_API_N
10:12
Category 1
Checkbox
True or False
Category 2
Edit Text
Enter a string
Ringtones
Select a ringtone
Second Preference Screen
Click here to go to the second Preference Screen


5554:Nexus_5X_API_N
10:12
Category 1
Checkbox
True or False
Category 2
Edit Text
Enter a string
Ringtones
Select a ringtone
Second Preference Screen
Click here to go to the second Preference Screen


5554:Nexus_5X_API_N
10:12
Category 1
Checkbox
True or False
Category 2
Edit Text
Enter a string
Ringtones
Select a ringtone
Second Preference Screen
Click here to go to the second Preference Screen

Persisting Data to Files

- Internal Storage: Store private data on the device memory
  - By default, files saved to the internal storage are private to your application and other applications cannot access them (nor can the user)
  - When the user uninstalls your application, these files are removed
- External Storage: Store public data on the shared external storage
  - Issue: not all phone has external storage
- Saving files that can be shared with other apps
  - Should be saved to a "public" location on the device, such as Music/, Pictures/, and Ringtones/, where other apps can access them and the user can easily copy them from the device

Saving to Internal Storage

- Create a project and name it Files
- Modify activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.wenbing.files.MainActivity">

    <TextView
        android:text="Please enter some text."
        android:layout_width="245dp"
        android:layout_height="wrap_content"
        android:id="@+id/textView"
        app:layout_constraintLeft_toLeftOf="@+id/activity_main"
        app:layout_constraintTop_toTopOf="@+id/activity_main"
        android:layout_marginTop="16dp"
        app:layout_constraintRight_toRightOf="@+id/activity_main"
        app:layout_constraintBottom_toTopOf="@+id/editText"
        android:layout_marginBottom="8dp"
        app:layout_constraintVertical_bias="0.28" />
    <EditText
        android:layout_width="241dp"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:ems="10"
        tools:layout_editor_absoluteY="82dp"
        android:id="@+id/editText”
        app:layout_constraintLeft_toLeftOf="@+id/activity_main"
        app:layout_constraintRight_toRightOf="@+id/activity_main"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        android:layout_marginTop="136dp"/>
```

Saving to Internal Storage

- Modify activity_main.xml:

```
<Button
        android:text="Save"
        android:layout_width="240dp"
        android:layout_height="wrap_content"
        android:id="@+id/btnSave"
        app:layout_constraintLeft_toLeftOf="@+id/activity_main"
        android:layout_marginStart="16dp"
        app:layout_constraintTop_toBottomOf="@+id/editText"
        android:layout_marginTop="136dp"
        app:layout_constraintRight_toRightOf="@+id/activity_main"
        android:layout_marginEnd="16dp"
        android:onClick="onClickSave" />
    <Button
        android:text="Load"
        android:layout_width="241dp"
        android:layout_height="wrap_content"
        android:id="@+id/btnLoad"
        app:layout_constraintLeft_toLeftOf="@+id/activity_main"
        android:layout_marginStart="16dp"
        app:layout_constraintTop_toBottomOf="@+id/editText"
        android:layout_marginTop="48dp"
        app:layout_constraintRight_toRightOf="@+id/activity_main"
        android:layout_marginEnd="16dp"
        android:onClick="onClickLoad" />
</android.support.constraint.ConstraintLayout>
```

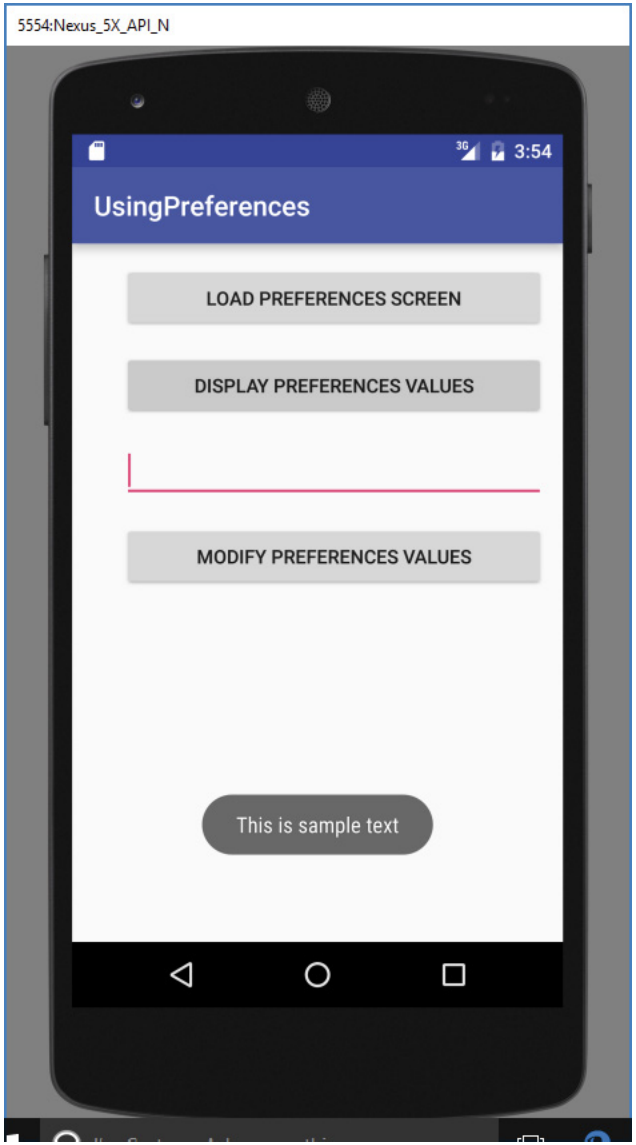
Saving to Internal Storage

- Modify MainActivity.java:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
```

```
public class MainActivity extends AppCompatActivity {
    EditText textBox;
    static final int READ_BLOCK_SIZE = 100;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textBox = (EditText) findViewById(R.id.editText);
    }

    public void onClickSave(View view) {
        String str = textBox.getText().toString();
        try {
            FileOutputStream fOut = openFileOutput("textfile.txt", MODE_PRIVATE);
            OutputStreamWriter osw = new OutputStreamWriter(fOut);
            try {
                osw.write(str);
            } catch (IOException e) {
                e.printStackTrace();
            }
            osw.flush();
            osw.close();
            Toast.makeText(getBaseContext(),
                    "File saved successfully!", Toast.LENGTH_SHORT).show();
             //---clears the EditText---
            textBox.setText("");
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
```

```
public void onClickLoad(View view) {
      try {
         FileInputStream fIn = openFileInput("textfile.txt");
         InputStreamReader isr = new InputStreamReader(fIn);
         char[] inputBuffer = new char[READ_BLOCK_SIZE];
         String s = "";
         int charRead;
         while ((charRead = isr.read(inputBuffer)) > 0) {
            //---convert the chars to a String---
            String readString =
                  String.copyValueOf(inputBuffer, 0, charRead);
            s += readString;
            inputBuffer = new char[READ_BLOCK_SIZE];
         }
         //---set the EditText to the text that has been read---
         textBox.setText(s);
         Toast.makeText(getBaseContext(), "File loaded successfully!",
               Toast.LENGTH_SHORT).show();
      } catch (IOException ioe) {
         ioe.printStackTrace();
      }
   }
}
```



Saving files to public folders

- To get a File representing the appropriate public directory, call getExternalStoragePublicDirectory(), passing it the type of directory you want, such as DIRECTORY_MUSIC, DIRECTORY_PICTURES, DIRECTORY_RINGTONES
- Exercise (**required**): Modify the Files app by saving the file to the DIRECTORY_PICTURES directory so that you can retrieve the file using your computer: create a subdirectory under DIRECTORY_PICTURES

```
public File getAlbumStorageDir(String albumName) {
    // Get the directory for the user's public pictures directory.
    File file = new File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

Saving files to public folders

- Add permission in Manifest

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

```
File dir = getAlbumStorageDir("gimbaldata");
//System.out.println("saving to dir "+dir.getPath());
String filePath = dir.getPath().toString()+"/log.csv";
File outputFile = new File(filePath);
if(!outputFile.exists()) {
    try {
        outputFile.createNewFile();
    } catch (IOException e) {
        e.printStackTrace();
        //System.out.println("cannot create new file: "+filePath);
    }

}
```

Creating and Using Databases

- Android uses the SQLite database system
- The SQLite database that you create programmatically in an application is stored in the /data/data/<package_name>/databases folder
- Create an app and named it Databases
  - Create an adapter for data structure
  - Insert, read, update, delete operations

Creating and Using Databases

- Create a new Java file and name it DBAdapter:

```
mport android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DBAdapter {
    static final String KEY_ROWID = "_id";
    static final String KEY_NAME = "name";
    static final String KEY_EMAIL = "email";
    static final String TAG = "DBAdapter";
    static final String DATABASE_NAME = "MyDB";
    static final String DATABASE_TABLE = "contacts";
    static final int DATABASE_VERSION = 1;
    static final String DATABASE_CREATE =
          "create table contacts (_id integer primary key autoincrement, "
              + "name text not null, email text not null);";
    final Context context;
    DatabaseHelper DBHelper;
    SQLiteDatabase db;
    public DBAdapter(Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }
```

```
private static class DatabaseHelper extends SQLiteOpenHelper
{
    DatabaseHelper(Context context)
    {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db)
    {
        try {  db.execSQL(DATABASE_CREATE);
        } catch (SQLException e) { e.printStackTrace(); }
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
    {
        Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
                + newVersion + ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS contacts");
        onCreate(db);
    }
}
//---opens the database---
public DBAdapter open() throws SQLException
{
    db = DBHelper.getWritableDatabase();
    return this;
}
//---closes the database---
public void close()
{
    DBHelper.close();
}
```

```
    public long insertContact(String name, String email)
    {
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_NAME, name);
        initialValues.put(KEY_EMAIL, email);
        return db.insert(DATABASE_TABLE, null, initialValues);
    }
    public boolean deleteContact(long rowId)
    {
        return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
    }
    public Cursor getAllContacts()
    {
        return db.query(DATABASE_TABLE, new String[] {KEY_ROWID, KEY_NAME,
                KEY_EMAIL}, null, null, null, null, null);
    }
    public Cursor getContact(long rowId) throws SQLException
    {
        Cursor mCursor = db.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
                            KEY_NAME, KEY_EMAIL}, KEY_ROWID + "=" + rowId, null, null, null, null, null);
        if (mCursor != null) { mCursor.moveToFirst(); }
        return mCursor;
    }
    public boolean updateContact(long rowId, String name, String email)
    {
        ContentValues args = new ContentValues();
        args.put(KEY_NAME, name);
        args.put(KEY_EMAIL, email);
        return db.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId, null) > 0;
    }
}
```

Creating and Using Databases

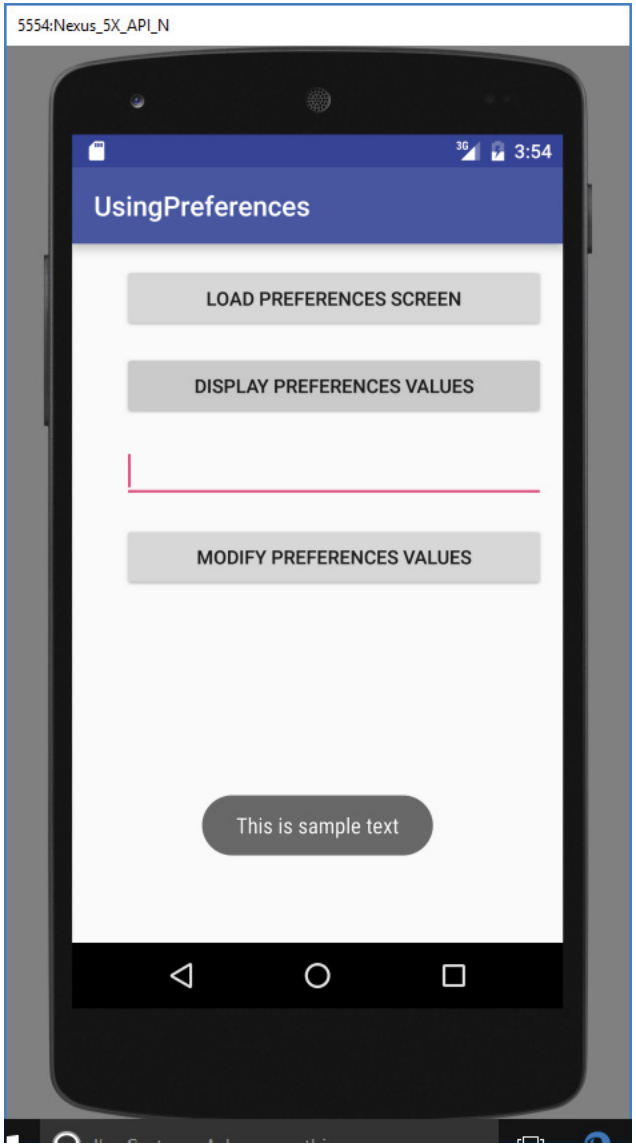
- Modify MainActivity.java:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.database.Cursor;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

        DBAdapter db = new DBAdapter(this);
        //---add a contact---
        db.open();
        long id = db.insertContact("Jennifer Ann",
                "jenniferann@jfdimarzio.com");
        id = db.insertContact("Oscar Diggs", "oscar@oscardiggs.com");
        db.close();

        db.open();
        Cursor c = db.getAllContacts();
        if (c.moveToFirst())
        {
            do {
                DisplayContact(c);
            } while (c.moveToNext());
        }
```

```
        c = db.getContact(2);
        if (c.moveToFirst())
            DisplayContact(c);
        else
            Toast.makeText(this, "No contact found", Toast.LENGTH_LONG).show();
        db.close();

        //---update contact---
        db.open();
        if (db.updateContact(1, "Oscar Diggs", "oscar@oscardiggs.com"))
            Toast.makeText(this, "Update successful.", Toast.LENGTH_LONG).show();
        else
            Toast.makeText(this, "Update failed.", Toast.LENGTH_LONG).show();
        db.close();

        //---delete a contact---
        db.open();
        if (db.deleteContact(1))
            Toast.makeText(this, "Delete successful.", Toast.LENGTH_LONG).show();
        else
            Toast.makeText(this, "Delete failed.", Toast.LENGTH_LONG).show();
        db.close();
    }

    public void DisplayContact(Cursor c)
    {
        Toast.makeText(this,
                "id: " + c.getString(0) + "\n" +
                        "Name: " + c.getString(1) + "\n" +
                        "Email: " + c.getString(2),
                Toast.LENGTH_LONG).show();
    }
}
```

Using menus with views

- Exercise (required):
  - Add several buttons to control the following operations (one for each operation): insert, read all records, read one record, update one record, delete one record
  - Remove the default TextView



Challenge Task

- For the contact manager app, add persistency using a database