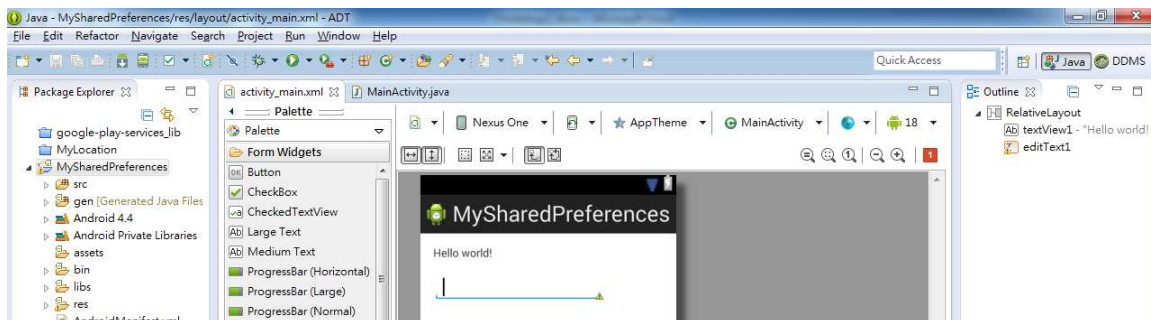


# 1. Shared Preferences

## 1.1 Shared Preferences

1. Create the Android application with the following attributes.
  - Application Name: **MySharedPreferences**
  - Project Name: **MySharedPreferences**
  - Package Name: **com.example.mysharedpreferences**
2. Drag an EditText to the layout.



3. Modify the source file "**MainActivity.java**" as follow:

```
package com.example.mysharedpreferences;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.content.SharedPreferences;
import android.content.Context;
import android.widget.EditText;

public class MainActivity extends Activity {

    private final String PREF_FILE = "com.example.mysharedpreferences.PREF_FILE";
    private final String PREF_KEY = "MY_SAVE_DATA";
    private EditText mEditText;
    private String mDataString;

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

```
// Restore the data from SharedPreferences
ReadSharedPreferences();

// Finds a view that was identified by the ID attribute from the XML
mEditText = (EditText)findViewById(R.id.editText1);

// Display the data on EditText
mEditText.setText(mDataString);
}

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

    // Read the user input text and save
    mDataString = mEditText.getText().toString();

    // Save the data to SharedPreferences
    SaveSharedPreferences();
}

private void ReadSharedPreferences() {
    // Retrieve and hold the contents of the preferences file
    SharedPreferences preferences = getSharedPreferences(PREF_FILE,
        Context.MODE_PRIVATE);

    // Retrieve the String from the preferences
    mDataString = preferences.getString(PREF_KEY, "");
}

private void SaveSharedPreferences() {
    // Retrieve and hold the contents of the preferences file
    SharedPreferences preferences = getSharedPreferences(PREF_FILE,
        Context.MODE_PRIVATE);

    // Create a new Editor for these preferences
    SharedPreferences.Editor editor = preferences.edit();

    // Set the String value in the preferences editor
```

```

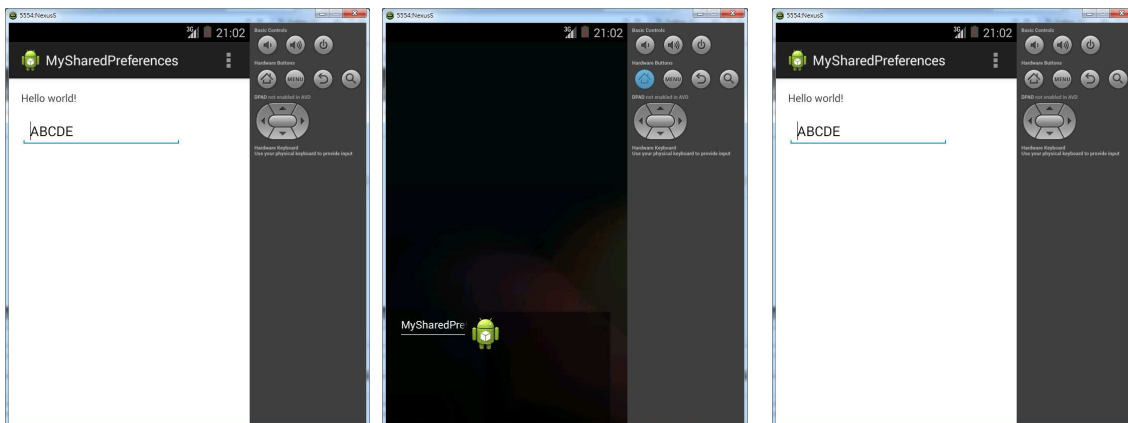
        editor.putString(PREF_KEY, mDataString);

        // Commit your preferences changes
        editor.commit();
    }

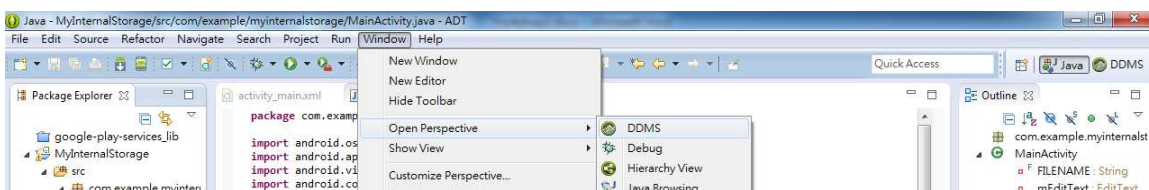
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}

```

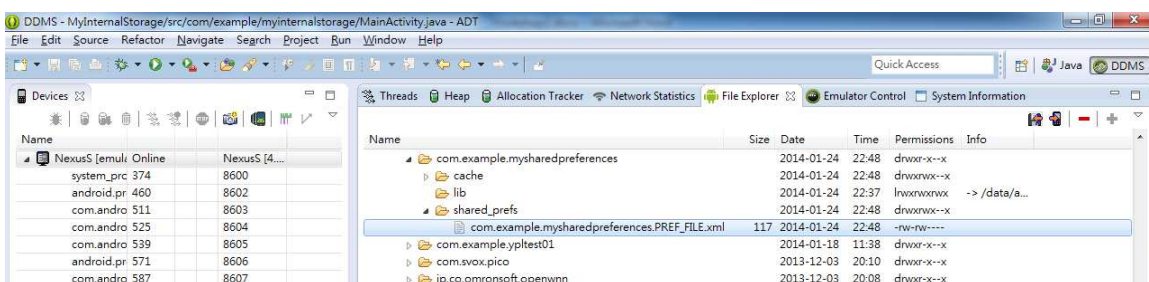
4. Save and execute the app. Input something in text field, and close the AVD. Then execute the app again, can you restore your input value?



5. Select **Windows** → **Open Perspective** → **DDMS**



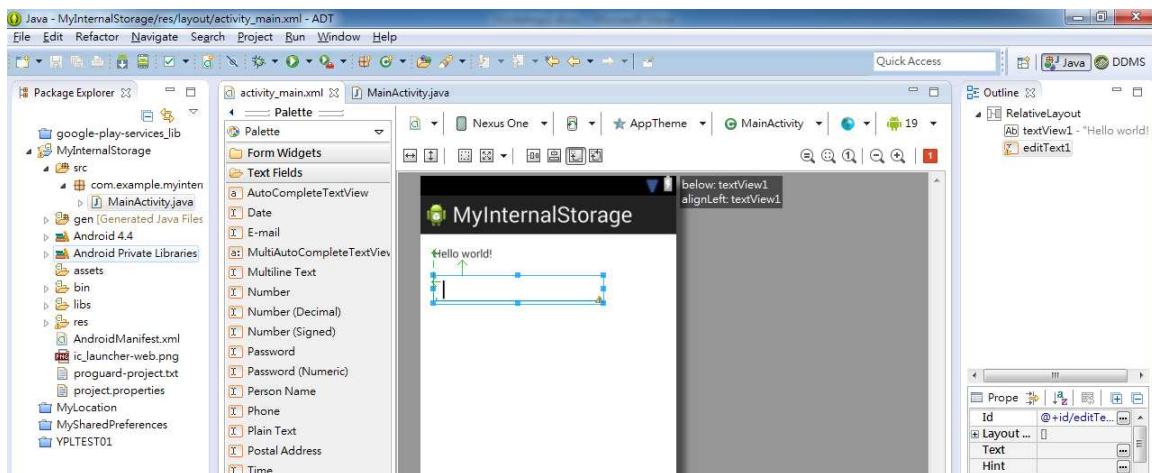
6. You can find your sharedpreference under **data**→**data**→ [your package] → **shared\_prefs**.



## 2. Internal and External Storage

### 2.1 Internal Storage

1. Create the Android application with the following attributes.
  - Application Name: **MyInternalStorage**
  - Project Name: **MyInternalStorage**
  - Package Name: **com.example.myinternalstorage**
2. Drag an EditText into the layout.



3. Then modify the source file "**MainActivity.java**" as follow:

```

package com.example.myinternalstorage;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.content.Context;
import android.widget.EditText;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;

public class MainActivity extends Activity {
    private final String FILENAME = "MyFile";
    private EditText mEditText;
    private String mDataString;
  
```

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

    // Restore the data from SharedPreferences
    ReadInternalStorage();

    // Finds a view that was identified by the ID attribute from the XML
    mEditText = (EditText)findViewById(R.id.editText1);

    // Display the data on EditText
    mEditText.setText(mDataString);
}

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

    // Read the user input text and save
    mDataString = mEditText.getText().toString();

    // Save the data to internal storage
    SaveInternalStorage();
}

// Read from Internal Storage
private void ReadInternalStorage() {
    try {
        // Open a private file for reading
        FileInputStream fos = openFileInput(FILENAME);

        // Constructs a new DataInputStream on the InputStream in
        DataInputStream in = new DataInputStream(fos);

        // Read the content in Unicode format
        mDataString = in.readUTF();

        // Closes this stream
    }
}
```

```
        in.close();

        // Closes the file stream.
        fos.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

// Save to Internal Storage
private void SaveInternalStorage() {
    try {
        // Open a private file for writing
        FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);

        // Constructs a new DataOutputStream on the OutputStream out
        DataOutputStream out = new DataOutputStream(fos);

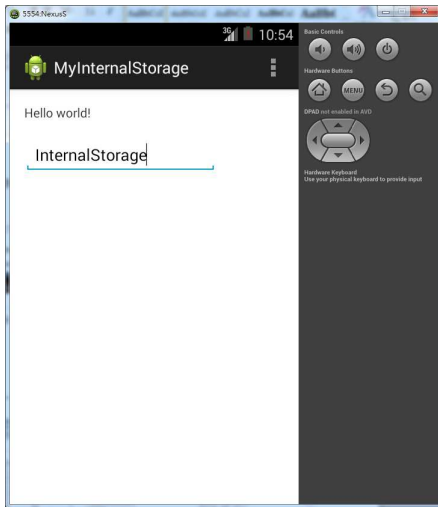
        // Write to file in Unicode format
        out.writeUTF(mDataString);

        // Close file
        out.close();

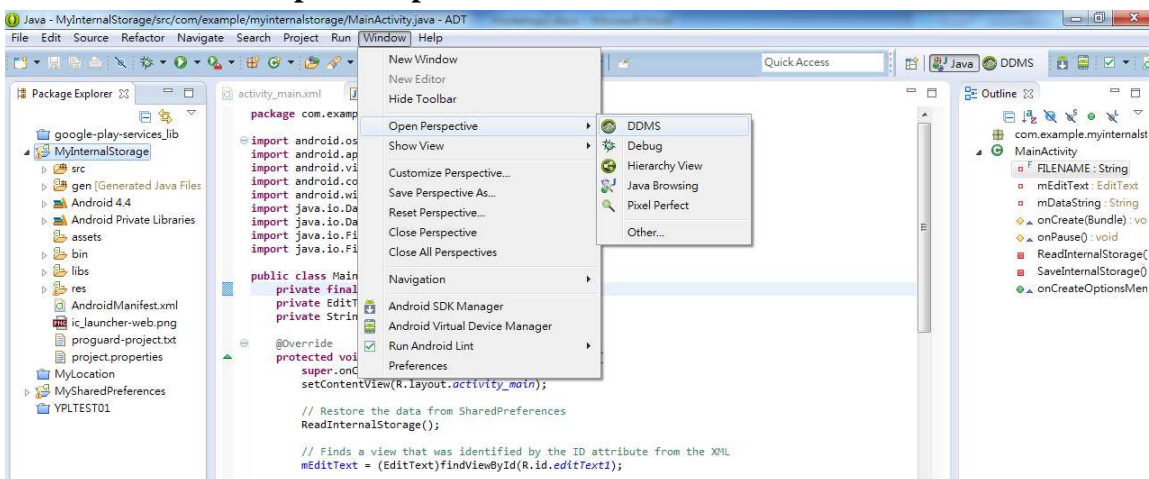
        // Closes the file stream
        fos.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
}
```

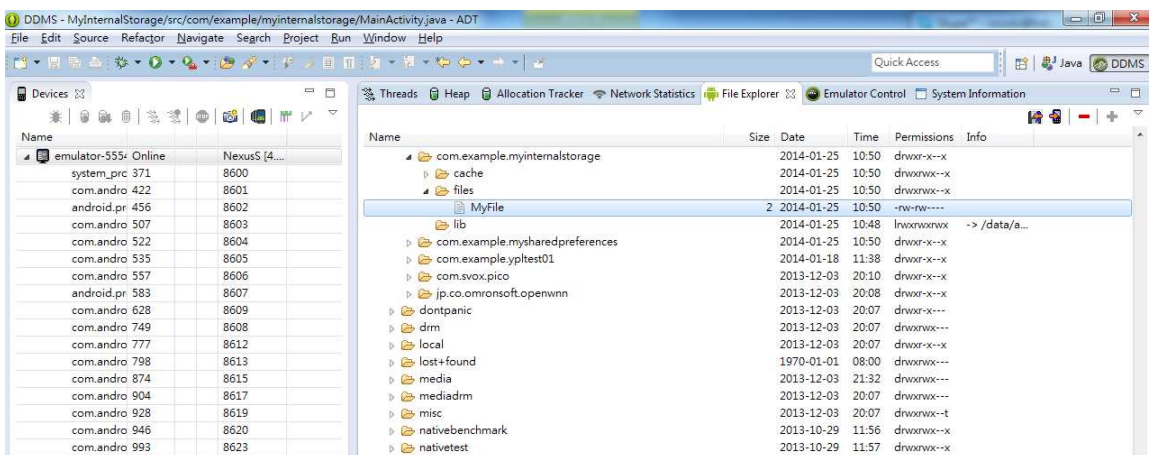
- Save and execute the app. Input something in text field, and close the AVD. Then execute the app again, can you restore your input value?



- Select **Windows** → **Open Perspective** → **DDMS**



- You can find your file under **data**→**data**→ [your package] → **file**.

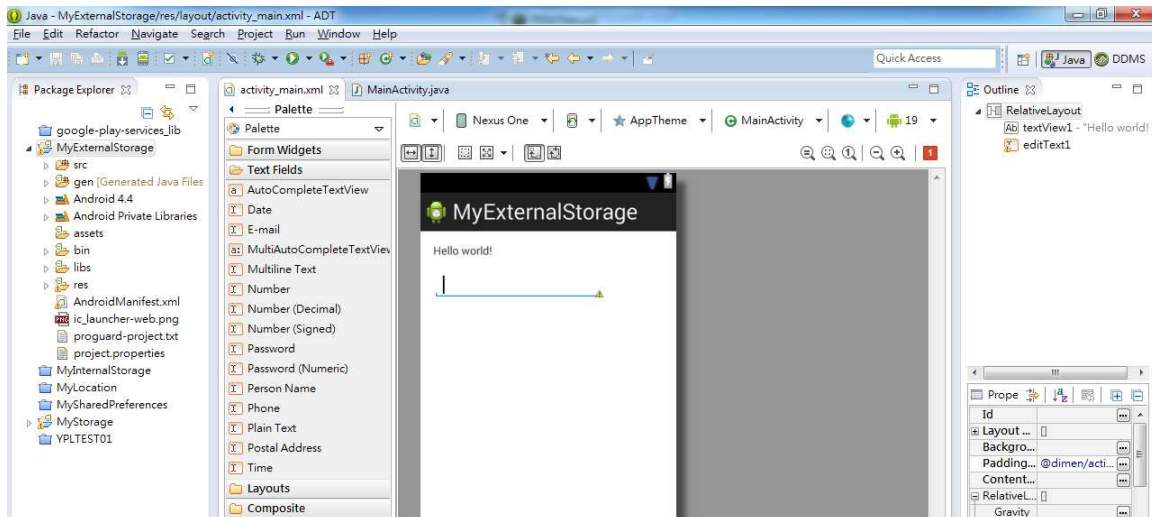




## 2.2 External Storage

### 2.2.1 External Storage – Public Folder

1. Create the Android application with the following attributes.
  - Application Name: **MyExternalStorage**
  - Project Name: **MyExternalStorage**
  - Package Name: **com.example.myexternalstorage**
2. Drag an EditText to the layout.



3. Modify the source file "**MainActivity.java**" as follow:

```
package com.example.myexternalstorage;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.EditText;
import android.os.Environment;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;

public class MainActivity extends Activity {
    private final String FOLDERNAME = "MyFolder";
    private final String FILENAME = "MyFile";
    private File myFolder, myFile;
```



```
private EditText mEditText;
private String mDataString;

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

    // Get the public storage folder and file name
    myFolder = getExternalFilesDir(FOLDERNAME);
    myFile = new File(myFolder, FILENAME);

    // Restore the data from external storage
    ReadExternalStorage();

    // Finds a view that was identified by the ID attribute from the XML
    mEditText = (EditText)findViewById(R.id.editText1);

    // Display the data on EditText
    mEditText.setText(mDataString);
}

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

    // Read the user input text and save
    mDataString = mEditText.getText().toString();

    // check if external storage is available and not read only
    if (isExternalStorageWritable() && isExternalStorageReadable()) {
        // Save the data to external storage
        SaveExternalStorage();
    }
}

public File getExternalFilesDir(String FolderName) {
    // Get the directory for the user's public directory (DOWNLOAD)
    File folder = new File(Environment.getExternalStoragePublicDirectory(
```

```
        Environment.DIRECTORY_DOWNLOADS), FolderName);

    // Create the folder
    folder.mkdirs();

    // Return the folder name
    return folder;
}

// Checks if external storage is available for read and write
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

// Checks if external storage is available to at least read
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}

private void SaveExternalStorage() {
    try {
        // Open file
        FileOutputStream fos = new FileOutputStream(myFile);

        // Write to file in binary format
        fos.write(mDataString.getBytes());

        // Close file
        fos.close();
    } catch (Exception e) {
```

```
        e.printStackTrace();
    }
}

// Read from External Storage
private void ReadExternalStorage() {
    String strLine;

    try {
        // Open file
        FileInputStream fis = new FileInputStream(myFile);

        // Open input stream
        DataInputStream in = new DataInputStream(fis);

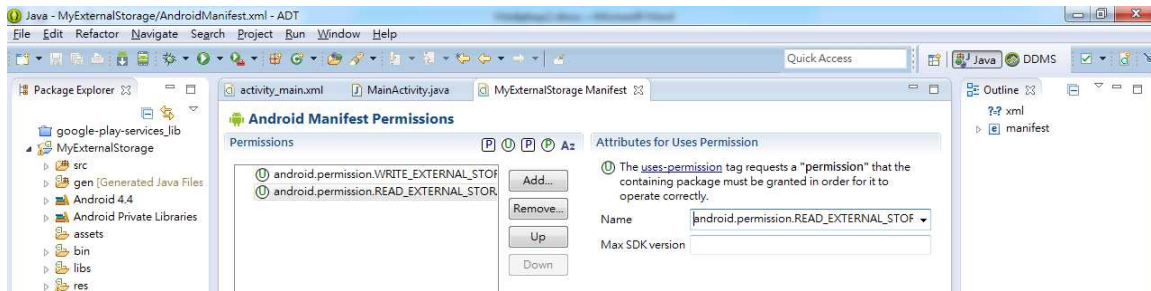
        // Read the file content into Buffer
        BufferedReader br = new BufferedReader(new InputStreamReader(in));

        // Store the buffer into variables
        mDataString = "";
        while ((strLine = br.readLine()) != null) {
            mDataString = mDataString + strLine;
        }

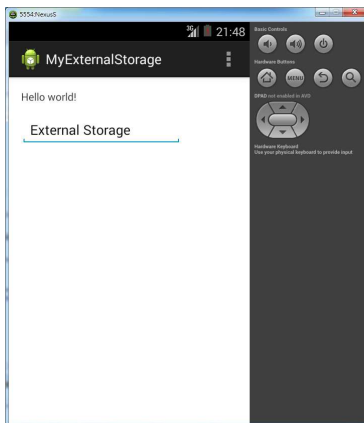
        // Close file
        in.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
}
```

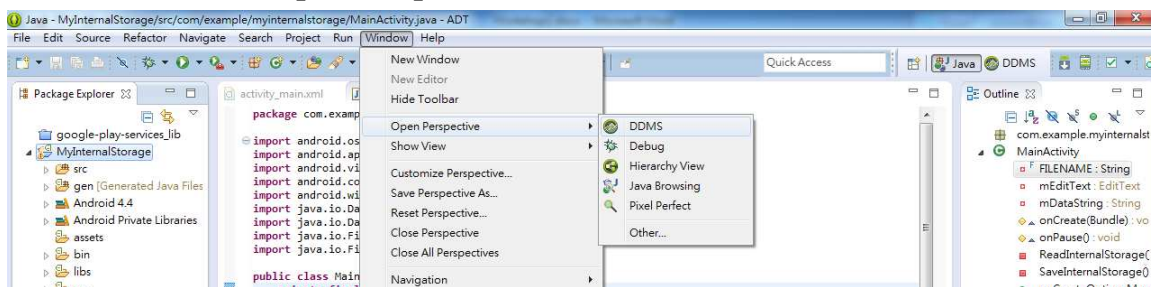
4. Add the following user permission in "**AndroidManifest.xml**":
  - android.permission.WRITE\_EXTERNAL\_STORAGE
  - android.permission.READ\_EXTERNAL\_STORAGE



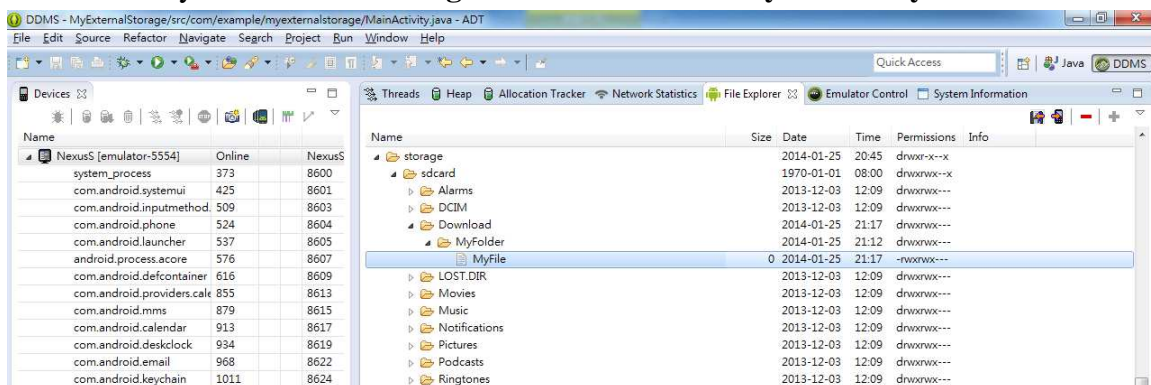
5. Save and execute the app. Input something in text field, and close the AVD. Then execute the app again, can you restore your input value?



6. Select **Windows** → **Open Perspective** → **DDMS**



7. You can find your file under **storage/sdcard/Download/MyFolder/MyFile**.



## 2.2.2 External Storage – Private Storage

1. Modify the source file "**MainActivity.java**" as follow:

```
package com.example.myexternalstorage;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.EditText;
import android.os.Environment;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;

public class MainActivity extends Activity {

    private final String FOLDERNAME = "MyFolder";
    private final String FILENAME = "MyFile";
    private File myFolder, myFile;
    private EditText mEditText;
    private String mDataString;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Get the public storage folder and file name
        myFolder = getExternalFilesDir(FOLDERNAME);
        myFile = new File(myFolder, FILENAME);

        // Restore the data from external storage
        ReadExternalStorage();

        // Finds a view that was identified by the ID attribute from the XML
        mEditText = (EditText)findViewById(R.id.editText1);

        // Display the data on EditText
```

```
mEditText.setText(mDataString);
}

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

    // Read the user input text and save
    mDataString = mEditText.getText().toString();

    // check if external storage is available and not read only
    if (isExternalStorageWritable() && isExternalStorageReadable()) {
        // Save the data to external storage
        SaveExternalStorage();
    }
}

public File getExternalFilesDir(String FolderName) {
    // Get the private directory
    File folder = new File(Environment.getExternalStorageDirectory(),
        FolderName);

    // Create the folder
    folder.mkdirs();

    // Return the folder name
    return folder;
}

// Checks if external storage is available for read and write
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

// Checks if external storage is available to at least read
```

```
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}

private void SaveExternalStorage() {
    try {
        // Open file
        FileOutputStream fos = new FileOutputStream(myFile);

        // Write to file in binary format
        fos.write(mDataString.getBytes());

        // Close file
        fos.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

// Read from External Storage
private void ReadExternalStorage() {
    String strLine;
    try {
        // Open file
        FileInputStream fis = new FileInputStream(myFile);

        // Open input stream
        DataInputStream in = new DataInputStream(fis);

        // Read the file content into Buffer
        BufferedReader br = new BufferedReader(new InputStreamReader(in));

        // Store the buffer into variables
        mDataString = "";
    }
}
```



```

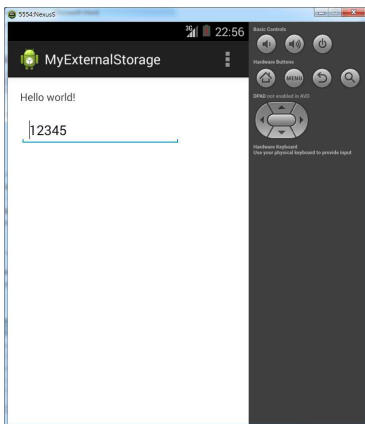
        while ((strLine = br.readLine()) != null) {
            mDataString = mDataString + strLine;
        }

        // Close file
        in.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

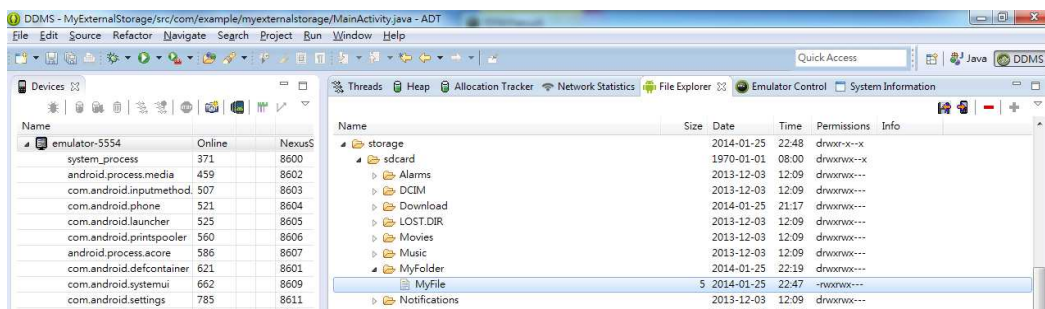
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
}

```

2. Save and execute the app. Input something in text field, and close the AVD. Then execute the app again, can you restore your input value?



3. Select **Windows** → **Open Perspective** → **DDMS** . You can find your file under **storage/sdcard/MyFolder/MyFile**.

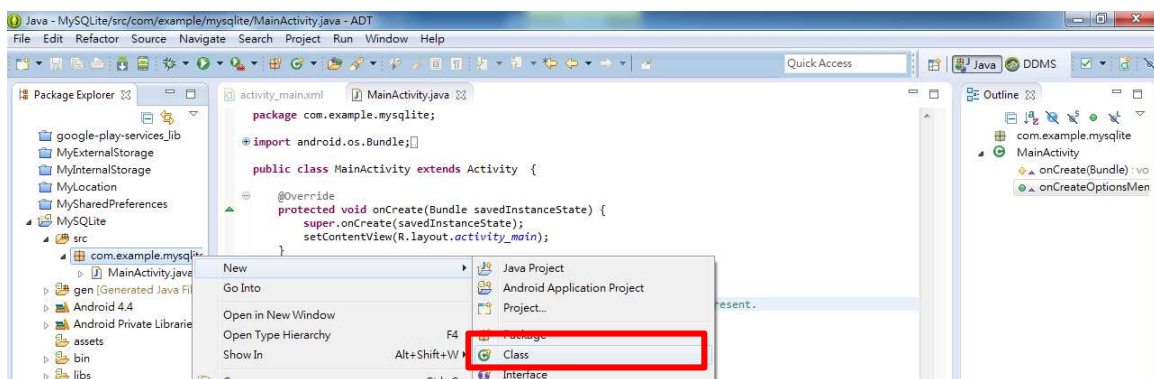


## 3. SQLite

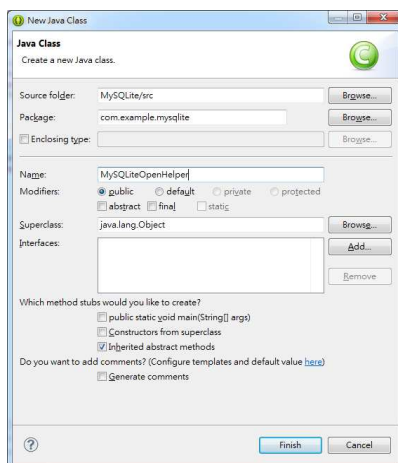
### 3.1 Simple SQLite Database

#### 3.1.1 Create Table

1. Create the Android application with the following attributes.
  - Application Name: **MySQLite**
  - Project Name: **MySQLite**
  - Package Name: **com.example.mysqlite**
2. The recommended method to create a new SQLite database is to create a subclass of *SQLiteOpenHelper* and override the *onCreate()* method, in which you can execute a SQLite command to create tables in the database. Right click the package, and select **New** → **Class**.



3. Input the class name **MySQLiteOpenHelper** and press **[Finish]**.



4. Modify the **MySQLiteOpenHelper.java** as follow:

```
package com.example.mysqlite;

import android.content.ContentValues;
import android.content.Context;
```

```
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import java.util.ArrayList;

// This class will create a Database "MyDatabase" with one table "myTable"
public class MySQLiteOpenHelper extends SQLiteOpenHelper {
    // If you change the database schema, you must increment the database version.
    public static final int DATABASE_VERSION = 1;

    // Database Name
    public static final String DATABASE_NAME = "MyDatabase";

    // Table name
    private static final String TABLE_NAME = "myTable";

    // Table Columns names
    private static final String COL_ID = "_id";
    private static final String COL_STUDENT = "student";
    private static final String COL_MARK = "mark";

    public MySQLiteOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // SQL statement to create table
        String SQL_CREATE_ENTRIES = "CREATE TABLE " + TABLE_NAME + " ( " +
            "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "student TEXT, "+
            "mark INTEGER )";

        // Create table
        db.execSQL(SQL_CREATE_ENTRIES);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

```
// This database is only a cache for online data, so its upgrade policy is  
// to simply to discard the data and start over. Drop older table if existed  
db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);  
  
// create fresh table  
onCreate(db);  
}  
  
// Insert single record  
public long insertRecord(String student, int mark){  
    // 1. get reference to writable DB  
    SQLiteDatabase db = this.getWritableDatabase();  
  
    // 2. create ContentValues to add key "column"/value  
    ContentValues values = new ContentValues();  
    values.put(COL_STUDENT, student);  
    values.put(COL_MARK, mark);  
  
    // 3. Insert record  
    long flag = db.insert(TABLE_NAME, null, values);  
  
    // 4. close database connection  
    db.close();  
  
    // 5. Return the newly inserted row ID, or -1 if an error occurred  
    return flag;  
}  
  
// Update single record  
public int updateRecord(int id, String student, int mark) {  
    // 1. get reference to writable DB  
    SQLiteDatabase db = this.getWritableDatabase();  
  
    // 2. create ContentValues to add key "column"/value  
    ContentValues values = new ContentValues();  
    values.put(COL_STUDENT, student);  
    values.put(COL_MARK, mark);  
  
    // 3. Define 'where' part of query. You may include ?s in the where clause,
```

```
//    which will be replaced by the values from whereArgs.
//    The values will be bound as Strings.
String selection = COL_ID + " = ?";

// 4. Specify arguments in placeholder order
String[] selectionArgs = new String[] { String.valueOf(id) };

// 5. Updating row
int rowAffected = db.update(TABLE_NAME, values, selection, selectionArgs);

// 6. close connect
db.close();

// 7. Return the number of rows deleted.
return rowAffected;
}

// Delete single record
public int deleteRecord(int id) {
    // 1. get reference to writable DB
    SQLiteDatabase db = this.getWritableDatabase();

    // 2. Define 'where' part of query.
    //    To remove all rows, pass "1" as the whereClause.
    String selection = COL_ID + " = ?";

    // 3. Specify arguments in placeholder order
    String[] selectionArgs = new String[] { String.valueOf(id) };

    // 4. Delete the row by execute the SQL statement
    int rowAffected = db.delete(TABLE_NAME, selection, selectionArgs);

    // 5. close connect
    db.close();

    // 6. Return the number of rows deleted
    return rowAffected;
}
```

```
// List all records

public ArrayList<String> ListRecord() {

    // 1. get reference to writable DB

    SQLiteDatabase db = this.getWritableDatabase();

    // 2. Define a projection that specifies which columns from the database
    // you will actually use after this query.

    String[] projection = { COL_ID, COL_STUDENT, COL_MARK };

    // 3. Define the sorting order for the results

    String sortOrder = COL_ID + " DESC";

    // 4. Define 'where' part of query.
    //    Passing null will return all rows for the given table.

    String selection = null;

    // 5. Specify arguments in placeholder order

    String[] selectionArgs = new String[] { };

    // 6. Declaring how to group rows, formatted as an SQL GROUP BY clause
    //    Passing null will cause the rows to not be grouped.

    String groupBy = null;

    // 7. Declare which row groups to include in the cursor, if row grouping is
    //    being used, formatted as an SQL HAVING clause
    //    Passing null will cause all row groups to be included, and is required
    //    when row grouping is not being used.

    String having = null;

    // 8. Query the table and obtain the result in cursor

    Cursor cursor = db.query(

        TABLE_NAME,        // The table to query
        projection,         // The columns to return
        selection,          // The columns for the WHERE clause
        selectionArgs,      // The values for the WHERE clause
        groupBy,           // The grouping of rows
        having,             // The filter of row groups
        sortOrder );       // The sort order
}
```

```
// 9. go over each row, build record and add it to list
ArrayList<String> results = new ArrayList<String>();
if (cursor.moveToFirst()) {
    do {
        int id = cursor.getInt(0);
        String student = cursor.getString(1);
        int mark = cursor.getInt(2);
        results.add(id + " : " + student + " = " + mark);
    } while (cursor.moveToNext());
}

// 11. Close cursor
cursor.close();

// 12. Close database connection
db.close();

// 13. return the list
return results;
}
}
```

5. Modify the source file "**MainActivity.java**" as follow:

```
package com.example.mysqlite;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.AdapterView;
import android.widget.Toast;
import android.app.ListActivity;
import java.util.ArrayList;

public class MainActivity extends ListActivity {
    public ArrayList<String> ListItems;
    public ArrayAdapter<String> adapter;
    long newRowID;
    int rowAffected;
}
```



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

    // To access your database, instantiate your subclass of SQLiteOpenHelper
    MySQLiteOpenHelper db = new MySQLiteOpenHelper(this);

    // Insert several record to table
    newRowID = db.insertRecord("John", 85);
    Toast.makeText(this, "Insert record to row:" + newRowID,
        Toast.LENGTH_LONG).show();

    newRowID = db.insertRecord("Mary", 90);
    Toast.makeText(this, "Insert record to row:" + newRowID,
        Toast.LENGTH_LONG).show();

    newRowID = db.insertRecord("David", 70);
    Toast.makeText(this, "Insert record to row:" + newRowID,
        Toast.LENGTH_LONG).show();

    // Update the first record
    rowAffected = db.updateRecord(1, "New David", 100);
    Toast.makeText(this, rowAffected + " rows updated",
        Toast.LENGTH_LONG).show();

    // Delete the third record
    rowAffected = db.deleteRecord(3);
    Toast.makeText(this, rowAffected + " rows deleted",
        Toast.LENGTH_LONG).show();

    // Get the list of record
    ListItems = db.ListRecord();

    // Define a new Adapter
    adapter = new ArrayAdapter<String>
        (this, android.R.layout.simple_list_item_1, ListItems);

    // Assign adapter to ListView
```

```

        setListAdapter(adapter);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}

```

6. Save and execute the app, the database record is retrieved and displayed on the screen.



7. Select **Windows** → **Open Perspective** → **DDMS**. You can find your file under **data/data/com.example.mysqlite/database/myDatabase**.

