LESSON – 9

JSON & GSON

Agenda

- Today we will discuss three things

1. How to work with built-in JSON Parsing, also discussing how to create JSONObject and JSONArray to store and retrieve on File.
2. Third Party Library Googleson(Gson), discussing how to work with Shared Preference to store and retrieve object using Gson.
3. How to store and retrieve list objects into file using Gson.

JSON

- JSON stands for JavaScript Object Notation. It is an independent data exchange format and is the best alternative for XML.
- A modifiable set of name/value mappings. Names are unique, non-null strings. Values may be any mix of JSONObjects, JSONArrays, Strings, Booleans, Integers, Longs, Doubles. Values may not be null.
- It is easy for machines to parse and generate" (*Introducing JSON*. Retrieved on Dec 10, 2015 from www.json.org)
- JSON object represented using { } braces. JSON arrays represented using [ ] braces.

JSON - Elements

An JSON file consist of many components. Here is the table defining the components of an JSON file and their description –

Sr.No	Component & description
1	**Array([)** In a JSON file , square bracket ([) represents a JSON array
2	**Objects({)** In a JSON file, curly bracket ({) represents a JSON object
3	**Key** A JSON object contains a key that is just a string. Pairs of key/value make up a JSON object
4	**Value** Each key has a value that could be string , integer or double e.t.c

XML & JSON Format of Employee record

XML Format

```
<employees>
 <employee>
    <id> 123 </id>
    <name> Renuka </name>
    <desig> AP </desig>
    <dept> CS </dept>
  </employee>
  <employee>
    <id> 125 </id>
    <name> Mohanraj </name>
    <desig> GD </desig>
    <dept> CS </dept>
  </employee>
 </employees>
```

JSON Format

```
{“employees”:[
    {
        “id” : 123,
        “name”: “Renuka”,
        “desig” : “AP”
         “dept”:”CS”
    },
    {
        “id” : 125,
        “name”: “Mohanraj”,
        “desig” : “GD”
         “dept”:”CS”
    }
  ]}
```

XML vs JSON

- Both are used to transfer data from one technology to another technology.
- JSON occupies less space and load faster than XML.
- Information is represented as a collection of key/value pairs, and that each key/value pair is grouped into an ordered list of objects. JSON can provide data type like Integer, String.
  - “id” : 123,
  - “name”: “Renuka”
- JSON parsing is simple. Converting object to JSON and JSON to object.
- Android has a build-in support for JSON parsing.
- Third party libraries are available in the market for paring such as GSON, Retrofit, Jackson & Jettison etc.
- In our course we will discuss, JSON basics and Gson to store the list of objects also to store object on Shared Preferences.

Hands on Example

- Here is the example to store an Employee record in your device external storage in a JSON format and also read the stored data from your device and display as a Toast.

Add this line into AndriodManifest.xml

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

Hands on Example – 1 for built-in JSON

- Design your activity_main.xml as per the screen shot.
- If the user click the WRITE JSON button, Entered data will be stored in your device external storage.
- If the user click the READ JSON button, file data will be displayed on Toast
- Ref : Lesson9\JSON Demo

Create JSON object and Store into File

- **JSON uses JSON Object and JSON Array**
- **Create JSON Object using**

```
JSONObject main = new JSONObject();
JSONObject emp_obj = new JSONObject();
```

- Insert a Key/Value pair

```
emp_obj.put("id", Integer.parseInt(et1.getText().toString()));
emp_obj.put("name", et2.getText().toString());
emp_obj.put("desig", et3.getText().toString());
emp_obj.put("dept", et4.getText().toString());
```

- **Create JSON array using**

```
JSONArray array = new JSONArray();
array.put(emp_obj);
```

- **Finally add emp_obj and array into main JSON object.**

```
main.put("employees", array);
```

- **To Store into a File**

```
String  path = Environment.getExternalStorageDirectory()
                .getAbsolutePath()+"/employe.json";
File f=new File(path);
FileWriter writer=new FileWriter(f);
writer.write(main.toString());
```

Retrieve from File and Store into JSON

```
String path = Environment.getExternalStorageDirectory()
        .getAbsolutePath()+"/employe.json";
          File f=new File(path);
          FileReader reader = new FileReader(f);
          String msg = "";
          int i = reader.read();
              while (i != -1) { // Check the file is empty or not
                  msg = msg + (char) i;
                  i = reader.read();
           }
```

Retrieve from File and Store into JSON

```
// Pass the String object and retrieve as a JSONObejct
        JSONObject object = new JSONObject(msg);
// Pass the Key name which is used on writing into JSON object
        JSONArray array = object.getJSONArray("employees");
        for(int j=0;j<array.length();j++){
        // Read the individual object
  JSONObject emp_obj=array.getJSONObject(j);
   Toast.makeText(getApplicationContext(),
        emp_obj.getInt("id")+"\n"+
              emp_obj.getString("name")+"\n"+
              emp_obj.getString("desig")+"\n"+
              emp_obj.getString("dept")
        ,Toast.LENGTH_LONG).show();
}
```

Google **Gson (**Gson)

- Gson is a Java library that can be used to convert Java Objects into their JSON representation.
- It can also be used to convert a JSON string to an equivalent Java object.
- It has extensive support for java generics.
- Provide simple toJson() and fromJson() methods to convert Java objects to JSON and vice-versa.
- To use Gson in Android add the below dependencies

```
dependencies { compile 'com.google.code.gson:gson:2.8.2' }
```

- Square’s Retrofit, including its Gson-based converter code, for retrieving JSON data from Web services.

Add Gson dependencies

- Step 1 : Right click app-> Open Module Settings

GsonDemo - [E:\ReMo\Andriod_Coding\Lesson9\GsonDemo] - [app] - ...\app\src\main\java\com\example\rmohanraj\gsontest\Employee.java - Android Studio 3.0.1

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

GsonDemo > app > src > main > java > com > example > rmohanraj > gsontest > Employee

activity_main.xml | MainActivity.java | Employee.java | gradle.properties

Employee dept

```
package com.example.rmohanraj.gsontest;

/**
 * Created by rMohanraj on 4/16/2018.
 */

public class Employee {
    private int id;
    private String name;
    private String desig;
    private String dept;

}
```

New
Link C++ Project with Gradle
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy as Plain Text
Paste Ctrl+V
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Local History
Synchronize 'app'
Show in Explorer
Directory Path Ctrl+Alt+F12
Compare With Ctrl+D
Open Module Settings
Create Gist...

Terminal | 0: Messages | 6: Logcat | TODO | Event Log | Gradle Console

Gradle build finished in 23s 420ms (27 minutes ago)

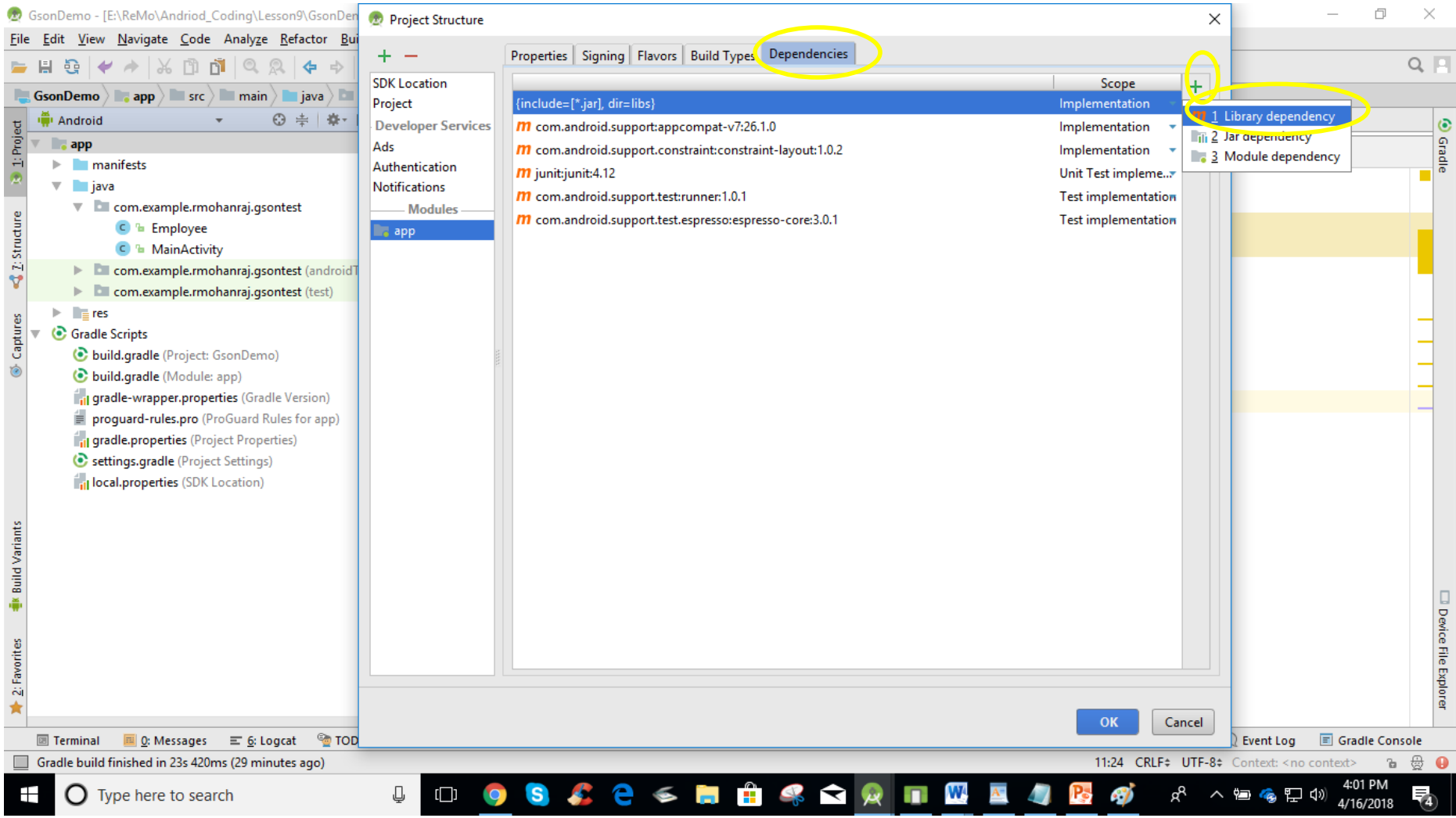
11:24 CRLF UTF-8 Context: <no context>

Type here to search

3:59 PM 4/16/2018

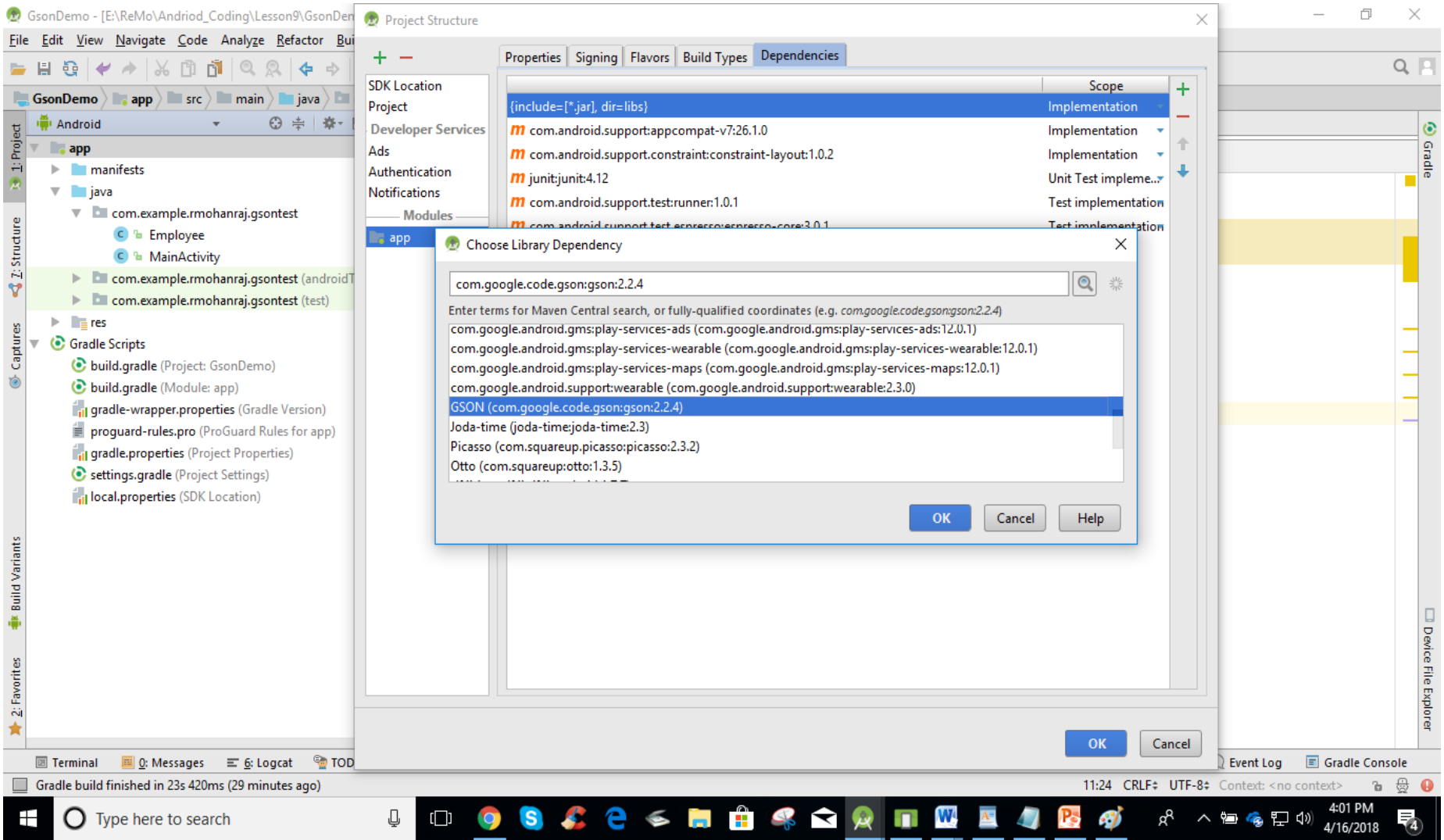
Add Gson dependencies

- Step 2 : Choose the Dependencies Tab and click Library Dependency



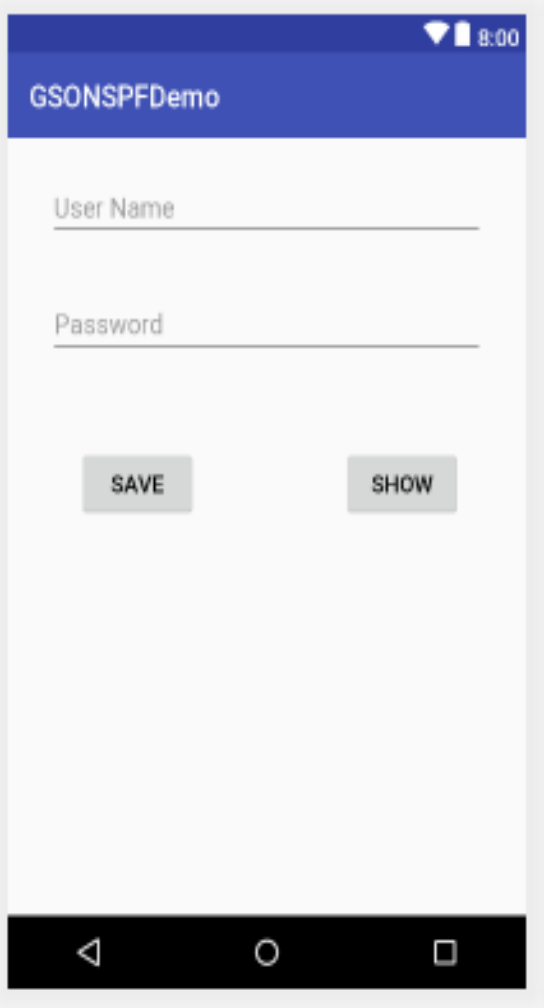
Add Gson dependencies

- Step 3 : Choose the Gson Dependencies and Click OK



Hands on Example – 2 using Gson

- Here User Name and Password stored as a User Object.
- Using Gson, object is stored and retrieved on Shared Preferences.
- Once the user select the SAVE button, data will be stored into Shared Preference.
- Once the user select the SHOW button, data will be retrieved from Shared Preference.
- Refer : Lesson9\GSONSPF



User.java

```
public class User {
    String uname;
    String password;

    public User(String uname, String password) {
        this.uname = uname;
        this.password = password;
    }
}
// Need to include Getters and Setters
```

MainActivity.java

- *// This demonstrates to insert User object into Shared Preferences using GSON*

```
public class MainActivity extends AppCompatActivity {
   EditText uname,pwd;
    @Override
    protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        uname = (EditText) findViewById(R.id.et1);
        pwd = (EditText) findViewById(R.id.et2);
    }
```

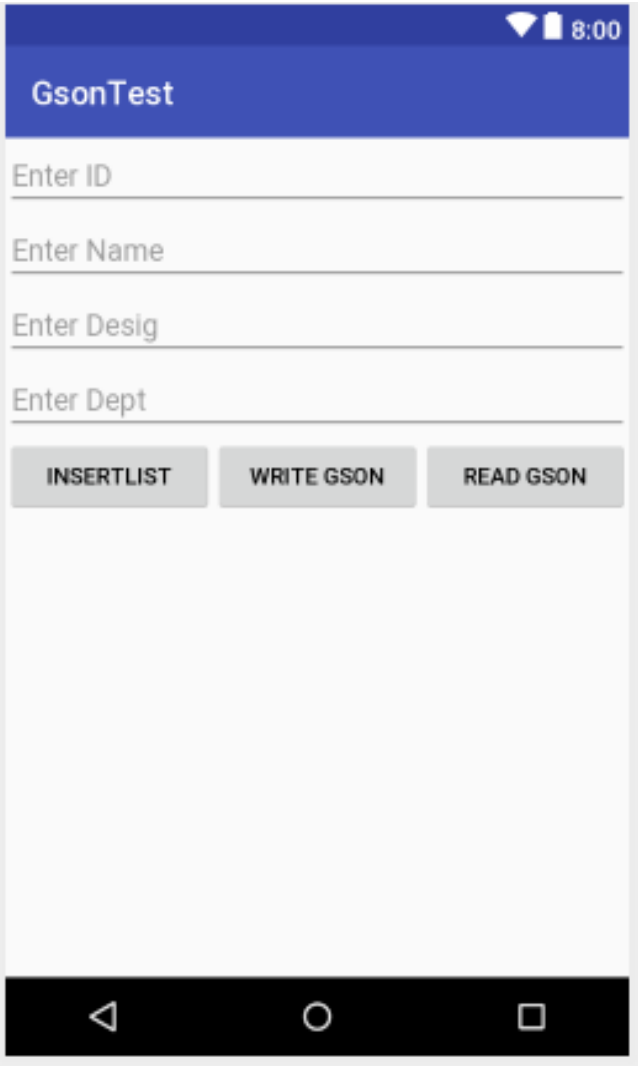
```java
public void save(View view) {
    // Create Gson obeject to store into Shared Preferences
    Gson gson = new Gson();
     String name = uname.getText().toString();
     String pass = pwd.getText().toString();
    // Get the Input from the Edit text and make an User Object
    User ob = new User(name,pass);
    // Convert User object into String object using toJson() method
         String first = gson.toJson(ob);
        // Store the retrieved Sting into Shared Preferences
        SharedPreferences spf = getSharedPreferences("user", 0);
        SharedPreferences.Editor edit = spf.edit();
        edit.putString("data",first);
        edit.commit();
}
```

```
public void show(View view) {
   // Create Gson object to retrieve data from Shared Preferences
    Gson gson = new Gson();
    SharedPreferences spf = getSharedPreferences("user", 0);
    String res = spf.getString("data","");
    // Convert the String object into User object using fromJson() method, return a type of User
    User opt = gson.fromJson(res,User.class);
   uname.setText(opt.getUname());
   pwd.setText(opt.getPassword());
}
```

Hands on Example-3 to store Multiple objects using Gson

File Reading and Writing JSON format using Gson

- If the user click the INSERT LIST button, Entered employee data will be stored into ArrayList<Employee>.
- If the user click the WRITE GSON button, Employee List will be stored on File.
- If the user click the READ GSON button, employee data will be read from file and displayed one by one on Toast.
- Ref : Lesson9\GSON Demo



POJO Classes

- POJO, or *Plain Old Java Object*, is a normal Java object class (that is, not a JavaBean, Entity Bean etc.)
- To work with Example – 3, you have to create two POJO classes.

  1.Employee(Structure of Individual Employee)

  2. Employees (Structure to store Multiple Employees)
- Data member of these classes annotated with *@SerializedName indicates this member should be serialized to JSON with the provided name value as its field name.*

Employee class

```
public class Employee {
    /* @SerializedName annotation indicates this member should be serialized to
     JSON    with the provided name value as its field name.
     It will be serialized with that key, if Serialized {"id":10}*/

    @SerializedName("id")
    private int id;
    @SerializedName("name")
    private String name;
    @SerializedName("desig")
    private String desig;
    @SerializedName("dept")
    private String dept;
}
// Need to provide Getters and Setters
```

Employee s class

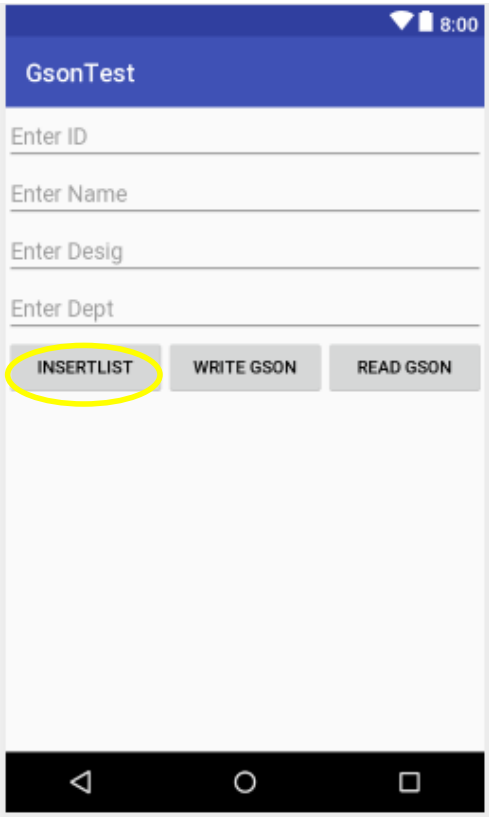
```
// This class helps to hold multiple Employee object
public class Employees {
    @SerializedName("employees")
    private ArrayList<Employee> employees;

    public ArrayList<Employee> getEmployees() {
        return employees;
    }

    public void setEmployees(ArrayList<Employee> employees) {
        this.employees = employees;
    }
}
// Need to provide Getters and Setters
```

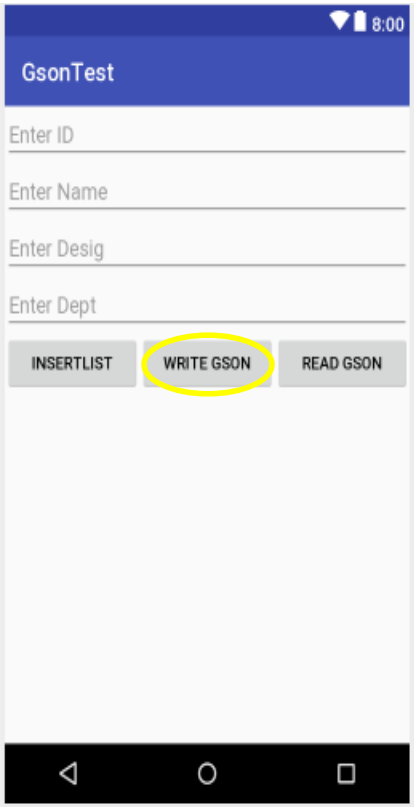
INSERT LIST BUTTON CLICK LOGIC

```
// This class helps to hold multiple Employee object
ArrayList<Employee> list = new ArrayList<>();
public  void add(View v){
    // Create an Employee object using the EditText Inputs
    Employee e = new Employee();
    e.setId(Integer.parseInt(et1.getText().toString()));
    e.setName(et2.getText().toString());
    e.setDesig(et3.getText().toString());
    e.setDept(et4.getText().toString());
    // Add an Employee into the list
   list.add(e);
}
```



WRITE GSON BUTTON CLICK LOGIC

```
public  void write(View v){
    // Set the list of Employees
        Employees emps = new Employees();
       emps.setEmployees(list);
  // Conversion Employees list object to JSON using Gson
       Gson gson = new Gson();
       String response = gson.toJson(emps);
  // Writing the converted data into File using FileWriter in Your device external storage
       String path = Environment.getExternalStorageDirectory()
              .getAbsolutePath() + "/emps_gson.json";
       try {
           FileWriter writer = new FileWriter(path);
           writer.write(response);
           writer.flush();
           writer.close();
       } catch (IOException e1) {
           e1.printStackTrace();
       }    } }
```



READ GSON BUTTON CLICK LOGIC

```
public  void read(View v){
   // Read the File using FileReader
   String  path = Environment.getExternalStorageDirectory()
          .getAbsolutePath()+"/emps_gson.json";
   try {
      FileReader reader = new FileReader(path);
      Gson gson = new Gson();
      Employees emps = gson.fromJson(reader, Employees.class
      ArrayList<Employee> list = emps.getEmployees();
      if (list.size() > 0) {
         for (Employee e : list) {
            Toast.makeText(getApplicationContext(), e.toString(),
Toast.LENGTH_LONG).show();
         }
      }
      else
         Toast.makeText(getApplicationContext(), "No data",
Toast.LENGTH_LONG).show();
      }  catch(FileNotFoundException e){
         e.printStackTrace();
      }
}
```