|  |
| --- |
| **Data Persistency** |

**Resources and References**

ListView

<http://developer.android.com/guide/topics/ui/layout/listview.html>

SharedPreferences

<http://developer.android.com/guide/topics/data/data-storage.html#pref>

Contents

[**Introducing Data Persistency** 2](#_Toc76196189)

[**Understanding SharedPreferences and Android Activity lifecycle** 2](#_Toc76196190)

[Section A: When to Save 2](#_Toc76196191)

[Design the UI Layout 2](#_Toc76196192)

[Implement the App without Data Persistency 3](#_Toc76196193)

[Section B: Saving and Retrieving Data in the App 7](#_Toc76196194)

[**Applying SharedPreferences in Saving User Input** 10](#_Toc76196195)

[Section C 10](#_Toc76196196)

[Create a New Project for My Profile App 10](#_Toc76196197)

[Design the UI Layout 10](#_Toc76196198)

[Saving and Retrieving Data in the App 10](#_Toc76196199)

[**SQLite** 14](#_Toc76196200)

[Section D: Data modelling 14](#_Toc76196201)

[Section E: Creating the Database 14](#_Toc76196202)

[Section F: Inserting Data 17](#_Toc76196203)

[Section G: Retrieving Data 20](#_Toc76196204)

[Section H: Model Data as Class Object 22](#_Toc76196205)

[Section I: Display using ListView 24](#_Toc76196206)

[Section J: Insert User’s Task 24](#_Toc76196207)

[Section K: Order by Description 25](#_Toc76196208)

# 

# **Introducing Data Persistency**

So far, the apps that we have developed in the 1st half of the semester do not support data persistency, which implies that data entered by user will be lost when the app ceased running in the mobile device.

There are many programming approaches in supporting data persistency in Android, including using SharedPreferences, SQLite Database, or saving the data into the remote cloud server. Today, we will be learning SharedPreferences.

# **Understanding SharedPreferences and Android Activity lifecycle**

Let’s look at how to use SharedPreferences to support data persistency of a simple data structure in the following exercise. You will be creating a simple app that stores and displays a greeting message.

## Section A: When to Save

In the Android Studio, create a new project with the following configuration requirements:

|  |  |
| --- | --- |
| **Project Template** | Empty Activity |
| **Application Name** | Demo Simple Save |
| **Package Name** | sg.edu.rp.c346.id<your student ID>.demosimplesave |
| **Project Location** | D:\C346\Workspace\DemoSimpleSave |
| **Language** | Java |
| **Minimum API Level** | API 16 |

## Design the UI Layout

Overwrite the generated layout file with a linear vertical layout as shown below.

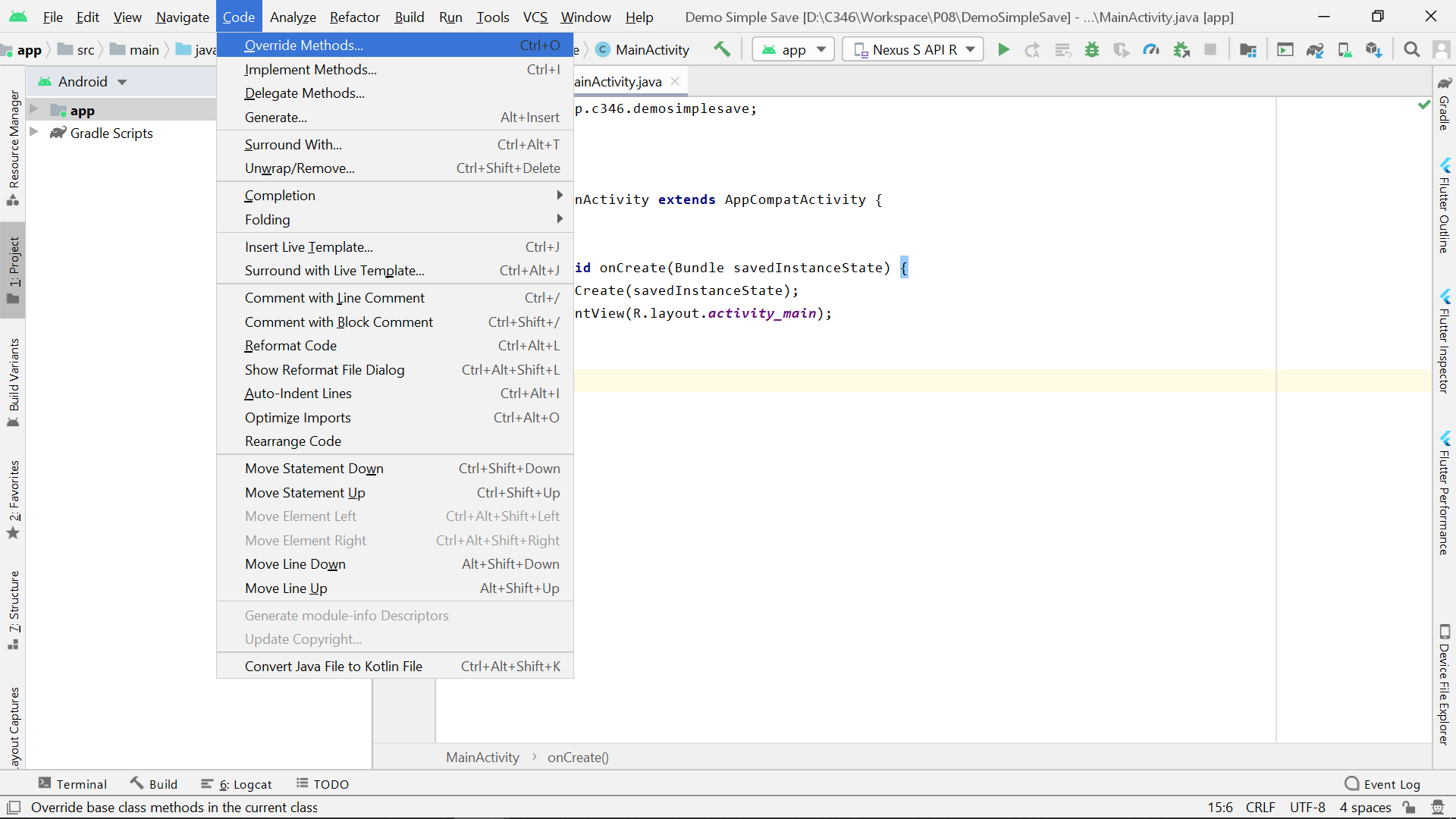
|  |  |
| --- | --- |
|  |  |

## Implement the App without Data Persistency

|  |
| --- |
|  |

1. In P05, we learnt a set of six callback methods in Android Activity lifecycle. Now let’s generate one of them, “onResume()”, in the MainActivity. Since the method is overridden, you can auto-generate it using Android Studio in either of the two ways below:
2. Select a white space in class MainActivity.

Click on the Android Studio menu, “Code 🡪 Override Methods…”



1. Right-click on a white space in class MainActivity.

Select “Generate…” and then “Override Methods…”

|  |  |
| --- | --- |
|  |  |

1. In the pop-up window, search and select “onResume()” as shown below. Click **OK**.

|  |
| --- |
|  |

1. The “onResume()” method should be auto-generated already. Now, add the following code (in red) into the method to display a Toast message when it is called.

|  |
| --- |
|  |

1. Run the app and observe your Emulator.

|  |  |
| --- | --- |
| ? | What message is displayed?  Referring to the Activity lifecycle above, explain why this message is displayed? |
| Message is “No greetings!” and as the application is run from onCreate, to onStart functions, it then runs the onResume() function which displays the message as the user views the Resume state. | |

1. Click the “Overview” button to open the “Recent Apps” window.

|  |
| --- |
|  |

1. Mirroring how an app is closed in the real phone, swipe the app or click the cross button on the top right corner to terminate it.

|  |
| --- |
| C:\Users\denise_quek\Desktop\Screenshot_1593399751.png |

1. Run the app again by launching it from the Emulator menu, and make some observations.

|  |
| --- |
| C:\Users\denise_quek\Desktop\Screenshot_1560503173.png |

|  |  |
| --- | --- |
| ? | What message is displayed this time?  Referring to the Activity lifecycle above, explain why this message is displayed? |
| Message is “No greetings!” as there is only one message that can be displayed when the onResume() function is called as if the application where at either paused or stopped state, when accessing the app again it will have to go through the onResume() function everytime. | |

## Section B: Saving and Retrieving Data in the App

Now let’s use SharedPreferences to save the message in this app.

1. Referring to how we generated the “onResume()” method in Step 1, generate the “onPause()” method in the MainActivity.

|  |
| --- |
|  |

1. To save the persistent data **when the app is no longer (fully) visible**, enter the code below into the “onPause()” method.

*Note: The writing of the value is in a (key, value) pair. The key is the name of the data structure used to “point” to the value that we wish to save in the local storage.*

|  |
| --- |
|  |

|  |  |
| --- | --- |
| ? | Referring to the Activity lifecycle above, explain why we should write the data saving code in the “onPause()” method. |
| When leaving the Resume state it will always go through the onPause() method as | |

1. To retrieve the saved data **when the Activity is resumed (fully visible)**, modify the code in the “onResume()” method as follow.

*Note: The reading of the value is using method getString(key, default value). The key is the name of the preference to retrieve, which points to a value in the SharedPreferences. If the value is not found (e.g., when the program is first installed), it will return the default value, “No greetings!” in this case.*

|  |
| --- |
|  |

1. Re-run the app on your Emulator and observe.

|  |  |
| --- | --- |
| ? | What message is displayed? Explain why. |
|  | |

1. Terminate the app and launch it again by following Step 5~7.

|  |  |
| --- | --- |
| ? | What message do you see this time? Explain why. |
|  | |

|  |  |
| --- | --- |
|  | **Learning Checkpoint 1** |
| *By now, you should be able to*   * Write an Android program that supports data persistency using SharedPreferences. * Apply your understanding of Android Activity lifecycle to implement the code to support data persistency.   **To recap on what we have learnt so far,**  In the above exercise, we created an Android app to save the hardcoded data and retrieve it for display using Toast. We tapped on the callback methods in Android Activity lifecycle for data persistency, using “onPause()” method to save the data when the app dismisses to the background, and using “onResume()” method to retrieve and display the saved data, when the app becomes visible again. | |

# **Applying SharedPreferences in Saving User Input**

## Section C

## Create a New Project for My Profile App

Let’s use SharedPreferences to support persistency of data input from the user. In the next exercise, you will be creating a simple app that stores information on GPA score.

In Android Studio, create a new project with the following configuration requirements:

|  |  |
| --- | --- |
| **Project Template** | Empty Activity |
| **Application Name** | Demo My Profile |
| **Package Name** | sg.edu.rp.c346.id<your student ID>.demomyprofile |
| **Project Location** | D:\C346\Workspace\DemoMyProfile |
| **Language** | Java |
| **Minimum API Level** | API 16 |

GitHub URL

|  |
| --- |
|  |

## Design the UI Layout

Create a layout file with the following content as per the component tree and the device screen in the Android Studio.

|  |  |
| --- | --- |
|  |  |

## Saving and Retrieving Data in the App

1. Declare the following variables in the “MainActivity.java”.

|  |
| --- |
|  |

1. Bind them to the respective UI elements using “findViewById()”.
2. Generate the callback methods “onPause()” and “onResume()” in the MainActivity as we did earlier on.
3. Let’s save the User Name into SharedPreferences first.

Implement the “onPause()” method according to the hints in the comments below, to save the User Name when the app is no longer (fully) visible.

*Note: You may refer to the code in the “Simple Save” project.*

|  |
| --- |
|  |

1. Implement the “onResume()” method according to the hints in the comments below, to retrieve the User Name value when the Activity is resumed (fully visible).

*Note: You may refer to the code in the “Simple Save” project.*

|  |
| --- |
|  |

1. Run and test your app.

User Name should be saved into SharedPreferences, whereas the GPA score is not being persistently stored yet.

1. Now let’s save the GPA score into SharedPreferences.

Usually GPA score is decimal so we should consider declaring it as a float value. With different data type being used, the SharedPreferences method to save/retrieve the key-value pair will be different. Check the following links and complete the tables below.

<https://developer.android.com/reference/android/content/SharedPreferences.Editor.html>

|  |  |
| --- | --- |
| **Type of data to save** | **SharedPreferences method** |
| String | putString(String key, String value) |
| int | putInt(String key, int value) |
| float | putFloat(String key, float value) |
| boolean | putBoolean(String key, boolean value) |

<https://developer.android.com/reference/android/content/SharedPreferences>

|  |  |
| --- | --- |
| **Type of data to retrieve** | **SharedPreferences method** |
| String | [getString](https://developer.android.com/reference/android/content/SharedPreferences.html#getString(java.lang.String,%20java.lang.String))([String](https://developer.android.com/reference/java/lang/String.html) key, [String](https://developer.android.com/reference/java/lang/String.html) defValue) |
| int | [getInt](https://developer.android.com/reference/android/content/SharedPreferences.html#getInt(java.lang.String,%20int))([String](https://developer.android.com/reference/java/lang/String.html) key, int defValue) |
| float | [getFloat](https://developer.android.com/reference/android/content/SharedPreferences.html#getFloat(java.lang.String,%20float))([String](https://developer.android.com/reference/java/lang/String.html) key, float defValue) |
| boolean | [getBoolean](https://developer.android.com/reference/android/content/SharedPreferences.html#getBoolean(java.lang.String,%20boolean))([String](https://developer.android.com/reference/java/lang/String.html) key, boolean defValue) |

Modify the “onPause()” and “onResume()” using the correct SharedPreferences methods:

* To save the GPA score when the app is no longer (fully) visible
* To retrieve the GPA score when the Activity is resumed (fully visible)

*Note: You need to use Float.parseFloat() to convert the GPA score input by the user to the “float” data type before saving it into the SharedPreferences object.*

1. Did you manage to write similar code in “onPause()” and “onResume()” as shown in the box below? It is still missing the code to save and retrieve the GPA, you would need to add the code similar to how it is done for the name but do note GPA is a float.

|  |
| --- |
| @Override **protected void** onPause() {  **super**.onPause();   *// Step 1a: Get the user input from the EditText and store it in a variable* String strName = **etName**.getText().toString();  **float** gpa= Float.*parseFloat*(**etGPA**.getText().toString());   *// Step 1b: Obtain an instance of the SharedPreferences* SharedPreferences prefs = *getPreferences*(**MODE\_PRIVATE**);   *// Step 1c: Obtain an instance of the SharedPreference Editor for update later* SharedPreferences.Editor prefEdit = prefs.edit();   *// Step 1d: Add the key-value pair* prefEdit.putString(**"name"**, strName); ***// How to save GPA?***  *// Step 1e: Call commit() to save the changes into SharedPreferences* prefEdit.commit(); }  @Override **protected void** onResume() {  **super**.onResume();   *// Step 2a: Obtain an instance of the SharedPreferences* SharedPreferences prefs = *getPreferences*(**MODE\_PRIVATE**);   *// Step 2b: Retrieve the saved data from the SharedPreferences object* String strName = prefs.getString(**"name"**, **"John"**);  ***// How to retrieve GPA?***  *// Step 2c: Update the UI element with the value* **etName**.setText(strName);  **etGPA**.setText(gpa + **""**); } |

|  |  |
| --- | --- |
|  | **Learning Checkpoint 2** |
| *By now, you should be able to*   * Write an Android program that supports persistency of user input data. * Reinforce your understanding of Android Activity lifecycle to implement the code to support data persistency.   **To recap on what we have learnt so far,**  In the above exercise, you created an Android app to save and retrieve the user input data. To save or retrieve different types of data, we need to use different SharedPreferences methods. | |

# **SQLite**

## Section D: Data modelling

1. Given a Java class below, below is the SQL statement to create the corresponding SQLite table that is able to store the info in the Java objects.

|  |
| --- |
| Task |
| -\_id: int  -description: String  -date: String |
| +Task(\_id: int, description: String, date: String)  +getID(): int  +getDescription(): String  +getDate(): String  +toString(): String |

## 

|  |
| --- |
| CREATE TABLE Task(  \_id INTEGER PRIMARY KEY AUTOINCREMENT,  description TEXT,  date TEXT) |

## Section E: Creating the Database

1. Create a new project named **Demo Database** with a suitable package name and the rest of the options as default.

|  |  |
| --- | --- |
| Project Name | Demo Database |
| Package | com.myapplicationdev.android.demodatabase |
| Activity Name | MainActivity |
| Layout Name | activity\_main.xml |
| Min SDK | API 16 |

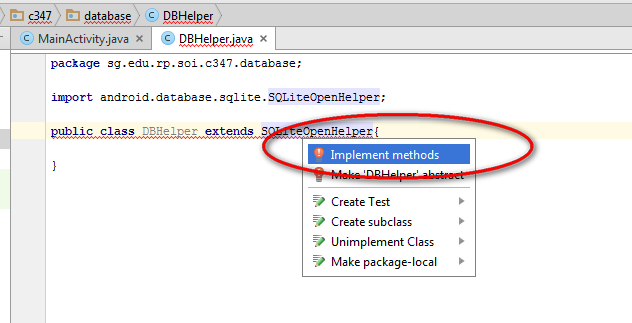
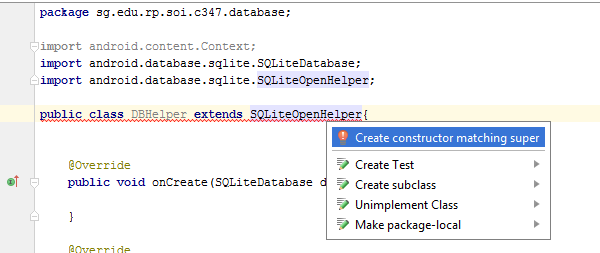
\*Import into GitHub

GitHub URL

|  |
| --- |
|  |

1. Create a new class and name it **DBHelper**. Make DBHelper **extends SQLiteOpenHelper** as below.

|  |  |
| --- | --- |
| 1  2 | public class DBHelper extends SQLiteOpenHelper {  } |

1. The class name DBHelper will be marked as having error. Place your mouse cursor on the line and hit “Alt+Enter”. You’ll need to add **unimplemented methods**.  
     
   
2. The quick fix would add 2 methods, namely **onCreate()** and **onUpgrade()**. This is the methods required for Android to manage SQLite database in an Android App.
3. After that, the DBHelper will still have an error as a constructor is required to the implemented. Place your mouse cursor on the line and hit “Alt+Enter”. You need to create constructor for it. Choose to add the first suggested constructor.   
     
   
4. The constructor would be added for you. This is the constructor to be invoked to create an instance of DBHelper later.
5. The constructor will need a few information. The vital information will be the **database name** and the **version number**. Typically, version number is used to represent any subsequent revisions to the database structure.
6. For the purpose of ease of managing the database, we could create some constants to be used in this class. The constants will be used in the constructor.
7. Let’s change the constructor to have clearer parameter name and also remove the unnecessary parameters as below. We change the constructor to take in only the Context object as we would use *DATABASE\_NAME* and *DATABASE\_VER* defined in the class for the name and version of the database.
8. Change DBHelper to the following.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public class DBHelper extends SQLiteOpenHelper {  // Start version with 1  // increment by 1 whenever db schema changes.  private static final int *DATABASE\_VER* = 1;  // Filename of the database  private static final String *DATABASE\_NAME* = "tasks.db";  Modify the list of arguments  public DBHelper(Context context) {  super(context, *DATABASE\_NAME*, null, *DATABASE\_VER*);  }  …  } |

1. Whenever we create a DBHelper object, it would check the database version passed to the super() constructor.
2. It would also automatically check if the database (tasks.db) already exists. It calls onCreate() to create the database if the database is not created yet.

For this to happen, the SQL statement to create the table needs to be executed in onCreate().

|  |
| --- |
| Task |
| \_id INTEGER PRIMARY KEY AUTOINCREMENT  description TEXT  date TEXT |

The above shows the structure of the table to be created. You need to insert the SQL statement for creating the table.

1. Add the variables and amend the onCreate() method as follows to create the table.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | public class DBHelper extends SQLiteOpenHelper {  private static final int *DATABASE\_VER* = 1;  private static final String *DATABASE\_NAME* = "tasks.db";   private static final String *TABLE\_TASK* = "task";  private static final String *COLUMN\_ID* = "\_id";  private static final String *COLUMN\_DESCRIPTION* = "description";  private static final String *COLUMN\_DATE* = "date";    public DBHelper(Context context) {  super(context, *DATABASE\_NAME*, null, *DATABASE\_VER*);  }   @Override  public void onCreate(SQLiteDatabase db) {  String createTableSql = "CREATE TABLE " + *TABLE\_TASK* + "("  + *COLUMN\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT,"  + *COLUMN\_DATE* + " TEXT,"  + *COLUMN\_DESCRIPTION* + " TEXT )";  db.execSQL(createTableSql);  Log.*i*("info" ,"created tables");  }  @Override  public void onUpgrade(SQLiteDatabase db, int oldVersion, int  newVersion) {  }  } |

1. If the database version had changed, it would call onUpgrade() and it is our job to write the code to do the upgrade. For simplicity, we may just make it delete the existing database and create a new one.

|  |  |
| --- | --- |
| 25  26  27  28  29  30  31  32  33 | @Override  public void onUpgrade(SQLiteDatabase db, int oldVersion, int  newVersion) {  *// Drop older table if existed*  db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_TASK*);  *// Create table(s) again*  onCreate(db);  }  } |

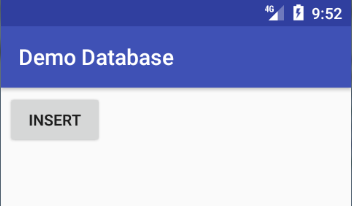
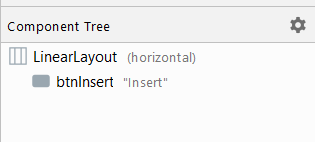
1. We are done with the bare minimum functions of DBHelper – it now creates and upgrades the database when necessary. But it is hardly useful as we have not implemented any code to do any of the four operations of database, namely, Create, Retrieve, Update and Delete (CRUD).

## Section F: Inserting Data

1. Let’s implement the code to insert a new note to the table. It is up to you to create a meaningful method name for this operation. For now, let’s create our method as **insertTask()**.
2. Add the following method to DBHelper class.

|  |  |
| --- | --- |
| 1  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | public class DBHelper extends SQLiteOpenHelper{  …  …  …  public void insertTask(String description, String date){  *// Get an instance of the database for writing*  SQLiteDatabase db = this.getWritableDatabase();  *// We use ContentValues object to store the values for  // the db operation*  ContentValues values = new ContentValues();  *// Store the column name as key and the description as value*  values.put(*COLUMN\_DESCRIPTION*, description);  *// Store the column name as key and the date as value*  values.put(*COLUMN\_DATE*, date);  *// Insert the row into the TABLE\_TASK*  db.insert(*TABLE\_TASK*, null, values);  *// Close the database connection*  db.close();  }  } |

1. Let’s create a button to test the database operation. Open **activity\_main.xml** and create a button as below with the ID as shown.

1. Create the **OnClickListener()** for the button as below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | public class MainActivity extends AppCompatActivity {   Button btnInsert;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);   btnInsert = findViewById(R.id.*btnInsert*);  btnInsert.setOnClickListener(new View.OnClickListener(){  @Override  public void onClick(View v) {  *// Create the DBHelper object, passing in the*  *// activity's Context* DBHelper db = new DBHelper(MainActivity.this);   *// Insert a task* db.insertTask("Submit RJ", "25 Apr 2021");    }  });  } } |

1. Run the app on an emulator.
2. Click on the **INSERT** button and open the **Device File Explorer** in Android Studio by following the steps below.

|  |
| --- |
|  |

A **db** file should be generated as shown below and you can view the details of your database by using tool *SQLiteBrowser* which needs to be installed.

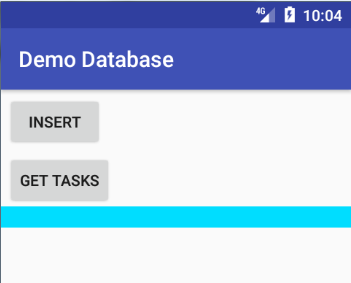
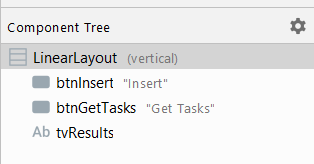
(Viewing databases from Android Studio): <https://stackoverflow.com/questions/17529766/view-contents-of-database-file-in-android-studio/51697711>

## Section G: Retrieving Data

1. With **DBHelper** class opened, add a method to get all the tasks’ content, into an **ArrayList<String>** object. Add the following getTaskContent() method to DBHelper class.

|  |  |
| --- | --- |
| 1  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83 | public class DBHelper extends SQLiteOpenHelper{  …  public ArrayList<String> getTaskContent() {  *// Create an ArrayList that holds String objects*  ArrayList<String> tasks = new ArrayList<String>();  *// Select all the tasks' description*  String selectQuery = "SELECT " + *COLUMN\_DESCRIPTION* + " FROM " + *TABLE\_TASK*;   *// Get the instance of database to read*  SQLiteDatabase db = this.getReadableDatabase();  *// Run the SQL query and get back the Cursor object*  Cursor cursor = db.rawQuery(selectQuery, null);   *// moveToFirst() moves to first row, null if no records*  if (cursor.moveToFirst()) {  *// Loop while moveToNext() points to next row  // and returns true; moveToNext() returns false  // when no more next row to move to* do {  *// Add the task content to the ArrayList object  // getString(0) retrieves first column data  // getString(1) return second column data  // getInt(0) if data is an integer value* tasks.add(cursor.getString(0));  } while (cursor.moveToNext());  }  *// Close connection*  cursor.close();  db.close();   return tasks;  }  } |

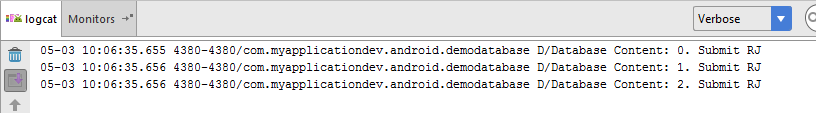
1. With **activity\_main.xml** opened, add a button **Get Tasks** and a **TextView** as below.

1. Set the **OnClickListener** for the button to get all the tasks as below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  .  .  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | public class MainActivity extends AppCompatActivity {   Button btnInsert, btnGetTasks;  TextView tvResults;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);   btnInsert = findViewById(R.id.*btnInsert*);  btnGetTasks = findViewById(R.id.*btnGetTasks*);  tvResults = findViewById(R.id.*tvResults*);   btnInsert.setOnClickListener(new View.OnClickListener(){  …  );   btnGetTasks.setOnClickListener(new View.OnClickListener(){  @Override  public void onClick(View v) {  *// Create the DBHelper object, passing in the*  *// activity's Context* DBHelper db = new DBHelper(MainActivity.this);   *// Insert a task* ArrayList<String> data = db.getTaskContent();  db.close();   String txt = "";  for (int i = 0; i < data.size(); i++) {  Debug using LogCat  Log.*d*("Database Content", i +". "+data.get(i));  txt += i + ". " + data.get(i) + "\n";  }  tvResults.setText(txt);  }  });  } } |

1. **Line 41** shows that the task content would be logged via **Log.d()** under the tag “Database Content”.
2. Run the app and click **Get Tasks** button, you’ll see the log as below? This is located at the lower window in Android Studio.



## Section H: Model Data as Class Object

1. Recap that the **task** table in database consists of three columns, namely \_id, description and date. This corresponds to the Java class that we have visited [earlier](#_Section_A:_Data).

|  |
| --- |
| Task |
| \_id INTEGER PRIMARY KEY AUTOINCREMENT  description TEXT  date TEXT |

|  |
| --- |
| Task |
| -\_id: int  -description: String  -date: String |
| +Task(\_id: int, description: String, date: String)  +getID(): int  +getDescription(): String  +getDate(): String  +toString(): String |

1. Create a Java class from the class diagram above. Also, create the constructor and the getters methods.
2. Create the getters for the fields and also a 3 parameter constructor to initialize the fields.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | public class Task {  private int id;  private String description;  private String date;   public Task(int id, String description, String date) {  this.id = id;  this.description = description;  this.date = date;  }   public int getId() { return id; }   public String getDescription() { return description; }   public String getDate() { return date;}  } |

1. Remember we will be using the simple layout **android.R.layout.simple\_list\_item\_1** when we create a new ArrayAdapter like this:   
   ArrayAdapter(MainActivity.this,**android.R.layout.simple\_list\_item\_1**,al) . This simple layout contains only a TextView and the ArrayAdapter will call the method toString() to get the String for this one TextView. We must make the three attributes, id, description and date, into a String in the method toString(). Override the method toString() and add the code as follows.

|  |  |
| --- | --- |
| 1  18  19  20  21  22  23 | public class Task {  ...  @NonNull  @Override  public String toString() {  return id + "\n" + description + "\n" + date;  } } |

1. Create a method called **getTasks()** in DBHelper that returns **ArrayList<Task>**.

|  |  |
| --- | --- |
| 1  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108 | public class DBHelper extends SQLiteOpenHelper{  …  …  public ArrayList<Task> getTasks() {  ArrayList<Task> tasks = new ArrayList<Task>();  String selectQuery = "SELECT " + *COLUMN\_ID* + ", "  + *COLUMN\_DESCRIPTION* + ", "  + *COLUMN\_DATE* + " FROM " + *TABLE\_TASK*;   SQLiteDatabase db = this.getReadableDatabase();  Cursor cursor = db.rawQuery(selectQuery, null);   if (cursor.moveToFirst()) {  do {  int id = cursor.getInt(0);  String description = cursor.getString(1);  String date = cursor.getString(2);  Task obj = new Task(id, description, date);  tasks.add(obj);  } while (cursor.moveToNext());  }  cursor.close();  db.close();  return tasks;  }  } |

1. This method will return the database records in the form of Task objects.

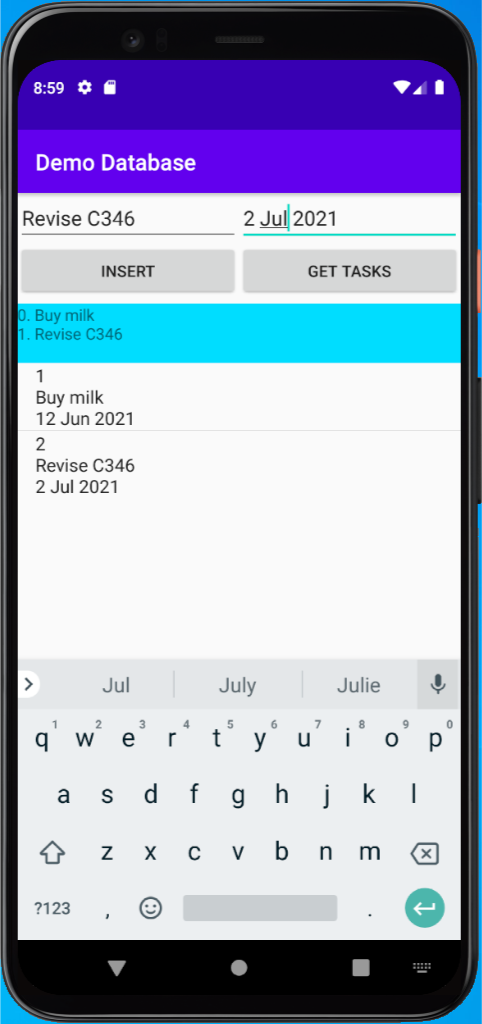
## Section I: Display using ListView

1. Instead of using TextView to display the records, we could also use a ListView.
2. Typically, the following steps would be required
   1. Create/modify an Activity to house a ListView
   2. Creates an ArrayAdapter object and binds it to the ListView object in the Activity.
3. Look at P07 on ListView creation.

|  |  |
| --- | --- |
| activity\_main.xml |  |

## Section J: Insert User’s Task

1. We have been adding the hardcoded task of “Submit RJ 25” on Apr 2021.Will you be able to add two EditTexts to the layout for the user to enter a task and the corresponding date? Insert the new task when the INSERT button is clicked as below.



## Section K: Order by Description

1. How do you change the SQL statement in the method getTasks() to retrieve the tasks sorted by the description in ascending (like the dictonary) or descending (like the dictonary in reverse) order?
2. How do you implement the feature that will alternate between retrieving the tasks in ascending and descending order when you press the GET TASKS button?