**ToDoList App**

In *MainActivity.java*, you should have code something like the below:

package com.cs260class.todolist; // This will refelect your package name

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

}

On line 11, you set the view of this activity to R.layout.activity\_main, which points to a file called *activity\_main.xml* in the */res/layout* directory of the project. A view controls layout of the Android interface and looks like this:

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context="com.aziflaj.todolist.MainActivity">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World!" />

</RelativeLayout>

In the main view, you will add a ListView, which will contain a ToDo item in each row. To do this, replace the TextView element with the code below:

<ListView

android:id="@+id/list\_todo"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

Now you will define a list item, which will represent a task in the interface. Create a new layout file in the */res/layout* folder called *item\_todo.xml*. You will add two elements to this file, a TextView to show the task, and a “Done” Button to delete the task. Add this code to *item\_todo.xml*, replacing anything that is already there.

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_gravity="center\_vertical">

<TextView

android:id="@+id/task\_title"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true"

android:text="Hello"

android:textSize="20sp" />

<Button

android:id="@+id/task\_delete"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentEnd="true"

android:layout\_alignParentRight="true"

android:text="Done" />

</RelativeLayout>

The app needs a menu item to allow user to add more tasks. Add a *main\_menu.xml* file in the */res/menu* directory with the following code:

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

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto">

<item

android:id="@+id/action\_add\_task"

android:icon="@drawable/ic\_menu\_add"

android:title="Add Task"

app:showAsAction="always" />

</menu>

Add the code below to the *MainActivity.java* file, after the onCreate method:

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main\_menu, menu);

return super.onCreateOptionsMenu(menu);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.action\_add\_task:

Log.d(TAG, "Add a new task");

return true;

default:

return super.onOptionsItemSelected(item);

}

}

Android developers frequently create a TAG constant with the name of the class for logging. Add this to the beginning of the MainActivity class:

public class MainActivity extends AppCompatActivity {

private static final String TAG = "MainActivity";

...

The onCreateOptionsMenu() method inflates (renders) the menu in the main activity, and uses the onOptionsItemSelected() method to react to different user interactions with the menu item(s). If you run the application, it should look something like this:

If you click the add button, you will see something like this in the Android Studio log:

D/MainActivity: Add a new task

Next, you will add an AlertDialog to get the task from the user when the add item button is clicked. You already know where to add the code to react to the user, so replace the logging statement with this:

final EditText taskEditText = new EditText(this);

AlertDialog dialog = new AlertDialog.Builder(this)

.setTitle("Add a new task")

.setMessage("What do you want to do next?")

.setView(taskEditText)

.setPositiveButton("Add", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

String task = String.valueOf(taskEditText.getText());

Log.d(TAG, "Task to add: " + task);

}

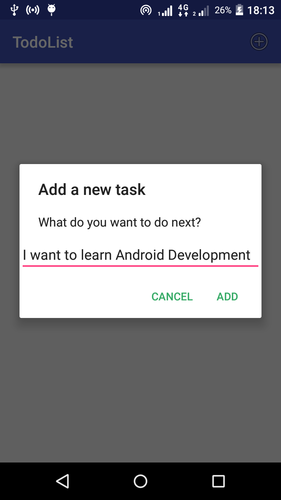
})

.setNegativeButton("Cancel", null)

.create();

dialog.show();

Now, clicking the plus button gives you this:



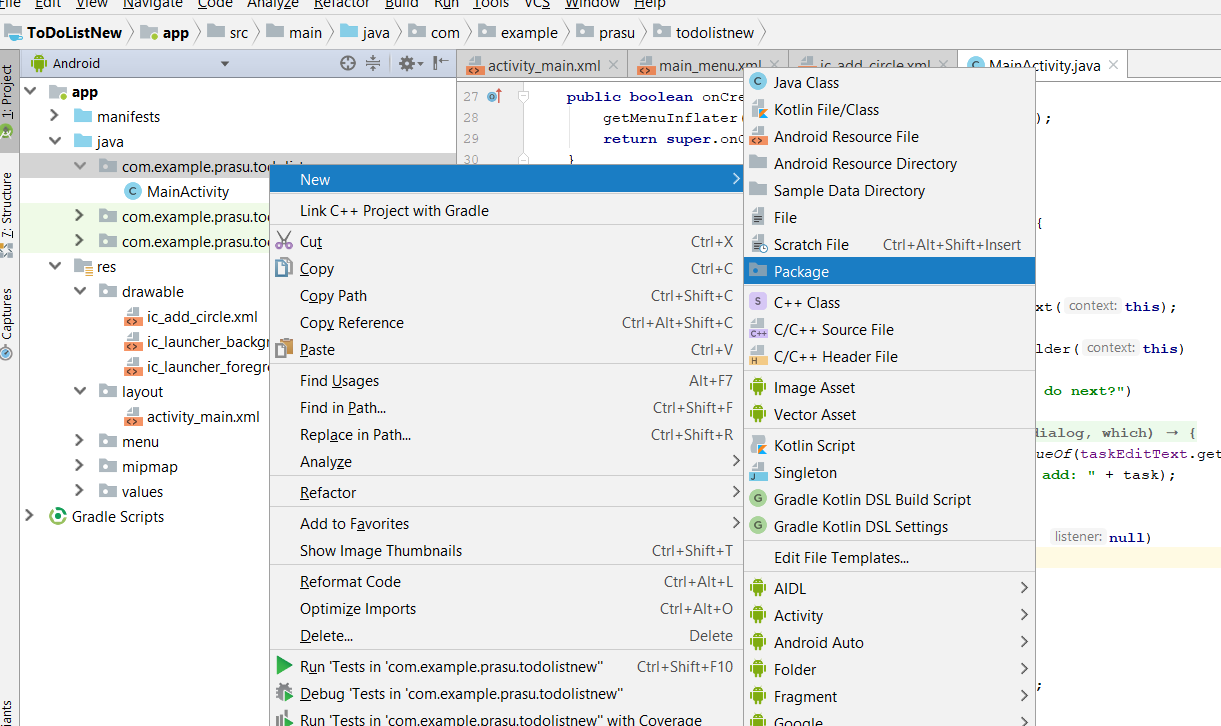
Enter some text and when you click the *add* button, the Android Studio log (“logcat”) will show something like this:

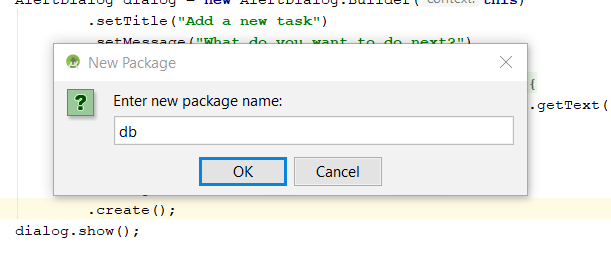
D/MainActivity: Task to add: I want to learn Android Development

**Storing and Retrieving Data**

Android ships with an embedded [SQLite](https://www.sqlite.org/) database. The database needs a table before it can store any tasks, called “TaskTable”. Create a new *db* folder in the same location as *MainActivity.java*. Then create a new class called TaskContract with the file name *TaskContract.java*:

To create a new folder db : Right click on the main folder under java -> New -> package. Give a name to the package (here: db) and click on ok.





The TaskContract class defines constants which used to access the data in the database. Add this code to *TaskContract.java*, changing the two package names appropriately.

package com.cs260class.todolist.db;

import android.provider.BaseColumns;

public class TaskContract {

public static final String DB\_NAME = "com.cs260class.todolist.db";

public static final int DB\_VERSION = 1;

public class TaskEntry implements BaseColumns {

public static final String TABLE = "tasks";

public static final String COL\_TASK\_TITLE = "title";

}

}

You also need a helper class called TaskDbHelper to open the database. Create this class in the db package and add the following code:

package com.cs260class.todolist.db;

import android.content.Context;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

public class TaskDbHelper extends SQLiteOpenHelper {

public TaskDbHelper(Context context) {

super(context, TaskContract.DB\_NAME, null, TaskContract.DB\_VERSION);

}

@Override

public void onCreate(SQLiteDatabase db) {

String createTable = "CREATE TABLE " + TaskContract.TaskEntry.TABLE + " ( " +

TaskContract.TaskEntry.\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +

TaskContract.TaskEntry.COL\_TASK\_TITLE + " TEXT NOT NULL);";

db.execSQL(createTable);

}

@Override

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

db.execSQL("DROP TABLE IF EXISTS " + TaskContract.TaskEntry.TABLE);

onCreate(db);

}

}

On lines 15 to 17 is this SQL query:

CREATE TABLE tasks (

\_id INTEGER PRIMARY KEY AUTOINCREMENT,

title TEXT NOT NULL

);

Now you need to adapt MainActivity to store data in the database. Add this code where you defined the DialogInterface.OnClickListener() for the AlertDialog‘s add button, replacing:

String task = String.valueOf(taskEditText.getText());

Log.d(TAG, "Task to add: " + task);

with:

String task = String.valueOf(taskEditText.getText());

SQLiteDatabase db = mHelper.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(TaskContract.TaskEntry.COL\_TASK\_TITLE, task);

db.insertWithOnConflict(TaskContract.TaskEntry.TABLE,

null,

values,

SQLiteDatabase.CONFLICT\_REPLACE);

db.close();

This makes the whole onOptionsItemSelected() method look like:

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.action\_add\_task:

final EditText taskEditText = new EditText(this);

AlertDialog dialog = new AlertDialog.Builder(this)

.setTitle("Add a new task")

.setMessage("What do you want to do next?")

.setView(taskEditText)

.setPositiveButton("Add", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

String task = String.valueOf(taskEditText.getText());

SQLiteDatabase db = mHelper.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(TaskContract.TaskEntry.COL\_TASK\_TITLE, task);

db.insertWithOnConflict(TaskContract.TaskEntry.TABLE,

null,

values,

SQLiteDatabase.CONFLICT\_REPLACE);

db.close();

}

})

.setNegativeButton("Cancel", null)

.create();

dialog.show();

return true;

default:

return super.onOptionsItemSelected(item);

}

}

Add a private instance of TaskDbHelper in the MainActivity class:

private TaskDbHelper mHelper;

And initialize it in the onCreate() method:

mHelper = new TaskDbHelper(this);

Now you need to fetch all the data from the database and show it in the main view.

Replace your MainActivity.onCreate() method with this:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mHelper = new TaskDbHelper(this);

SQLiteDatabase db = mHelper.getReadableDatabase();

Cursor cursor = db.query(TaskContract.TaskEntry.TABLE,

new String[]{TaskContract.TaskEntry.\_ID, TaskContract.TaskEntry.COL\_TASK\_TITLE},

null, null, null, null, null);

while(cursor.moveToNext()) {

int idx = cursor.getColumnIndex(TaskContract.TaskEntry.COL\_TASK\_TITLE);

Log.d(TAG, "Task: " + cursor.getString(idx));

}

cursor.close();

db.close();

}

When you run the application, LogCat will show a list of all the tasks stored in the database. Next you will display data in the main view using an Adapter.

Get a reference to the ListView created in *activity\_main.xml* file by adding an instance of the ListView:

private ListView mTaskListView;

Initialize the reference by adding this line of code to the onCreate() method, right after creating mHelper:

mTaskListView = (ListView) findViewById(R.id.list\_todo);

Move the code (with some changes) that was logging the tasks into a private method called updateUI():

private void updateUI() {

ArrayList<String> taskList = new ArrayList<>();

SQLiteDatabase db = mHelper.getReadableDatabase();

Cursor cursor = db.query(TaskContract.TaskEntry.TABLE,

new String[]{TaskContract.TaskEntry.\_ID, TaskContract.TaskEntry.COL\_TASK\_TITLE},

null, null, null, null, null);

while (cursor.moveToNext()) {

int idx = cursor.getColumnIndex(TaskContract.TaskEntry.COL\_TASK\_TITLE);

taskList.add(cursor.getString(idx));

}

if (mAdapter == null) {

mAdapter = new ArrayAdapter<>(this,

R.layout.item\_todo,

R.id.task\_title,

taskList);

mTaskListView.setAdapter(mAdapter);

} else {

mAdapter.clear();

mAdapter.addAll(taskList);

mAdapter.notifyDataSetChanged();

}

cursor.close();

db.close();

}

Add this private field to the MainActivity class:

private ArrayAdapter<String> mAdapter;

This ArrayAdapter will help populate the ListView with the data.

If you don’t understand the updateUI() method, that’s OK. Instead of logging the tasks, add them into an ArrayList of Strings. Then check if mAdapter is created or not. If it isn’t, and mAdapter is null, create and set it as the adapter of the ListView:

mAdapter = new ArrayAdapter<>(this,

R.layout.item\_todo, // what view to use for the items

R.id.task\_title, // where to put the String of data

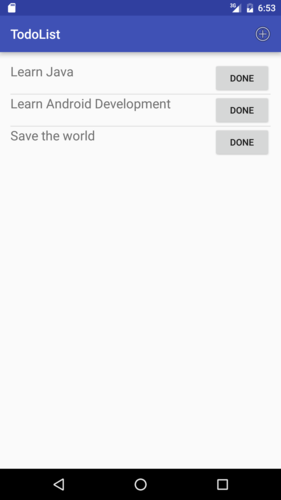
taskList); // where to get all the data

mTaskListView.setAdapter(mAdapter); // set it as the adapter of the ListView instance

If the adapter is already created (which implies that it’s assigned to the ListView), clear it, re-populate it and notify the view that the data has changed. This means that the view will repaint on the screen with the new data.

To see the updated data, you need to call the updateUI() method every time the underlying data of the app changes. So, add it in two places:

* In the onCreate() method, that initially shows all the data
* After adding a new task using the AlertDialog



**Deleting Tasks**

After finishing a task, it should be deleted from the list.

Open the *item\_todo.xml* layout and add this line to the Button tag:

android:onClick="deleteTask"

When the button is clicked, it calls this method deleteTask() in the MainActivityclass:

public void deleteTask(View view) {

View parent = (View) view.getParent();

TextView taskTextView = (TextView) parent.findViewById(R.id.task\_title);

String task = String.valueOf(taskTextView.getText());

SQLiteDatabase db = mHelper.getWritableDatabase();

db.delete(TaskContract.TaskEntry.TABLE,

TaskContract.TaskEntry.COL\_TASK\_TITLE + " = ?",

new String[]{task});

db.close();

updateUI();

}

Now, clicking the Done button will delete the task from the list and the SQLite database.