Tutorial 3 Install Android Studio 3.5 on Windows

Contents

Background for the Exercise:

- ## What is JAVA?

Java is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another.

- ## What is XML?

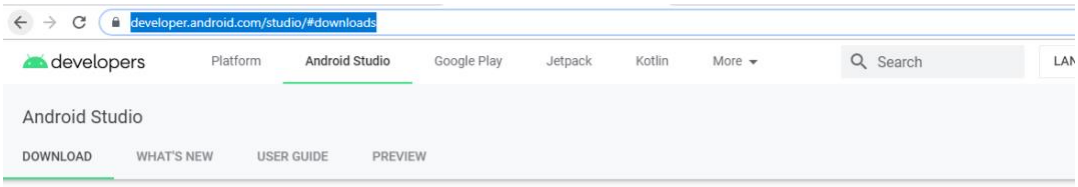
XML (i.e., Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The World Wide Web Consortium's XML 1.0 Specification of 1998 and several other related specifications—all of them free open standards—define XML. The basic building block of an XML document is an element, defined by tags. An element has a beginning and an ending tag. All elements in an XML document are contained in an outermost element known as the root element. XML can also support nested elements, or elements within elements. This ability allows XML to support hierarchical structures. Element names describe the content of the element, and the structure describes the relationship between the elements.

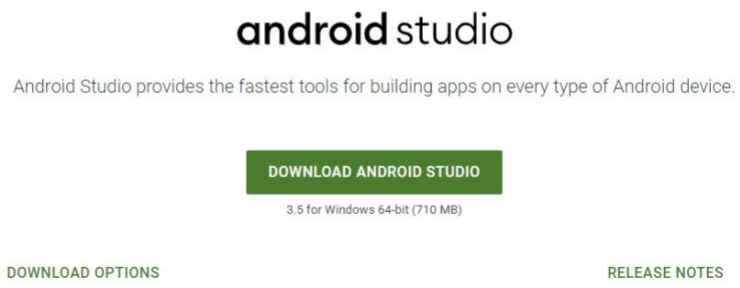
- ## What is Android?

Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially developed by Android, Inc., which Google backed financially and later purchased in 2005. Android is open source and Google releases the code under the Apache License. This open source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. The current version of the Android SDK is found in Android Studio 3.5.

Exercise

1. Set up an Android Development Environment





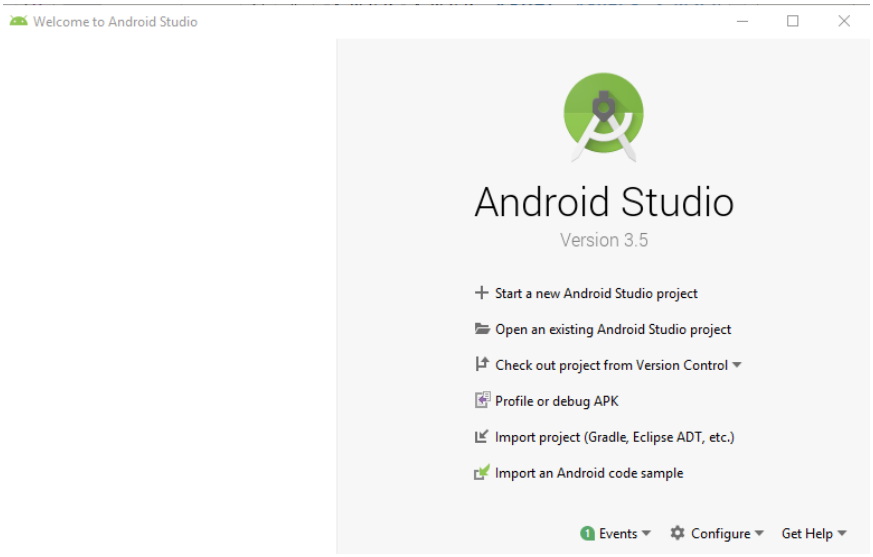
Android SDK

- Download Android Studio 3.5 here: https://developer.android.com/studio/#downloads
- Extend the **DOWNLOAD FOR OTHER PLATFORMS** on the screen

Platform	Android Studio package	Size	SHA-256 checksum
Windows (64-bit)	android-studio-ide-191.5791312-windows.exe Recommended	710 MB	75c538069f2171343032cfcfbf1bb2bbba28cc7bc75bf5a4b9f90cc1a4c1217c
	android-studio-ide-191.5791312-windows.zip No .exe installer	723 MB	201328aa7b2593ad538f0cc5d2d98237435d56c89d2bb08f072391de5faaf9f1
Windows (32-bit)	android-studio-ide-191.5791312-windows32.zip No .exe installer	722 MB	38431518936af0425f3949f8516dd9b5be8e6f1aa29b0ff8579dd425ffa0fa39
Mac (64-bit)	android-studio-ide-191.5791312-mac.dmg	724 MB	be3a0b809f4c3e6c588d4c3019f0a9062882ffab8440942bd23ebce7effa4989
Linux (64-bit)	android-studio-ide-191.5791312-linux.tar.gz	729 MB	5794fd6edca6e4daa31f5ed710670d0b425534549a6c4aa2e3e9753ae116736f
Chrome OS	android-studio-ide-191.5791312-cros.deb	614 MB	3a665cbce1dd94eb9dd9a6e2fc253b992c872395b575855eb4a836745d55197e

- Choose the platform you wish to install. And click **Next**
- Respond to installation prompts until Android Studio 3.5 is installed

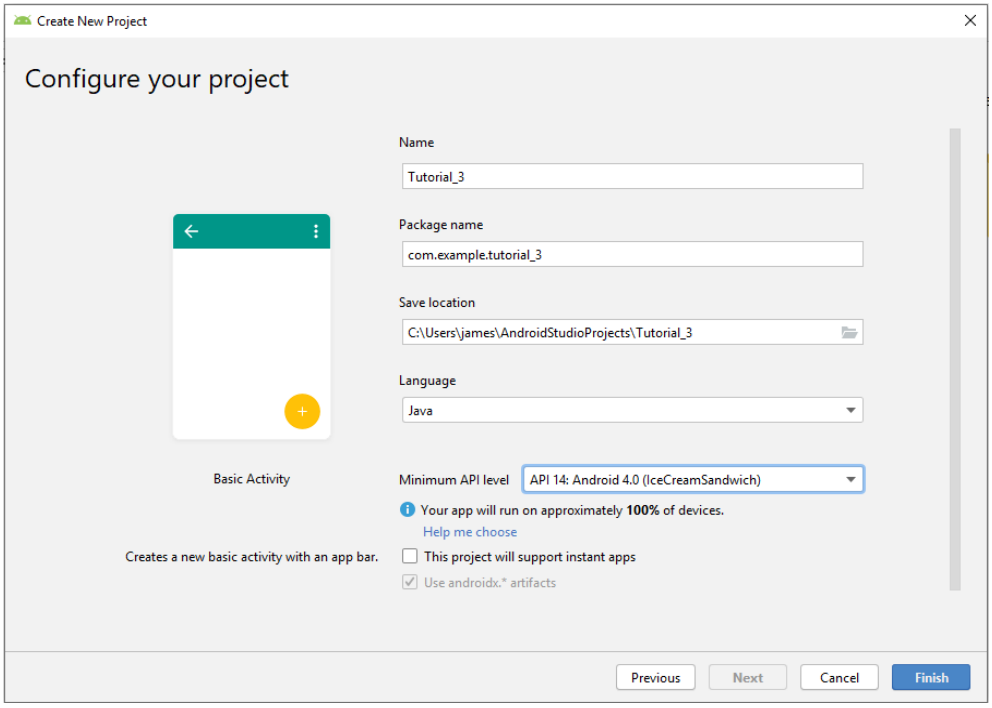
- Open Android Studio v3.5



- Start a new project named Tutorial_3YourName. Replace YourName with your name.

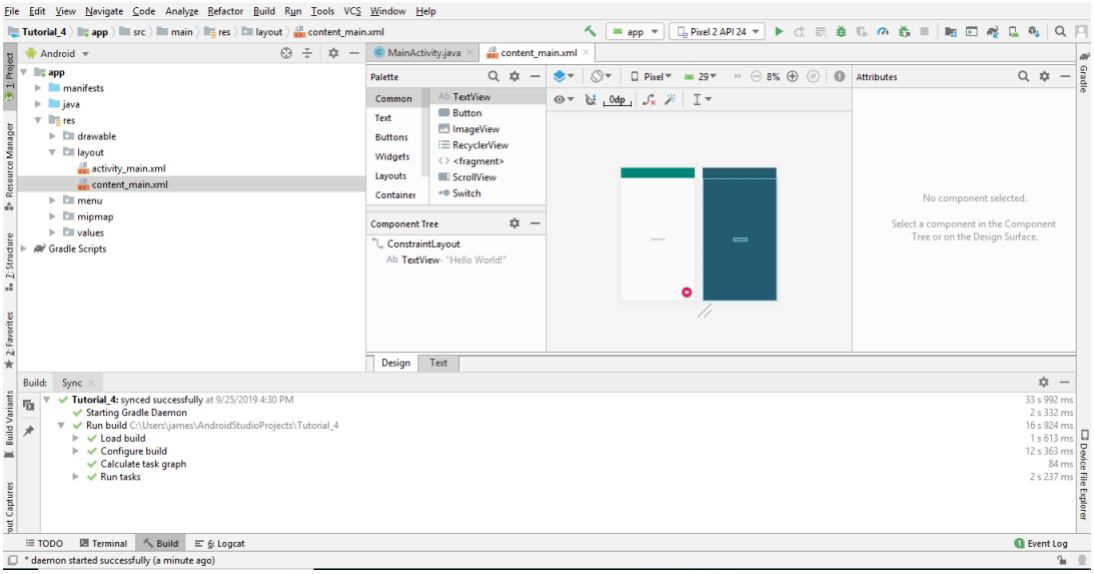
> ***Notice:*** Please replace **YourName** with **your real name** so that you can have an app with your name on it and we can know it`s yours! Do the same for the rest two below.

- Be sure to choose Java as the language.



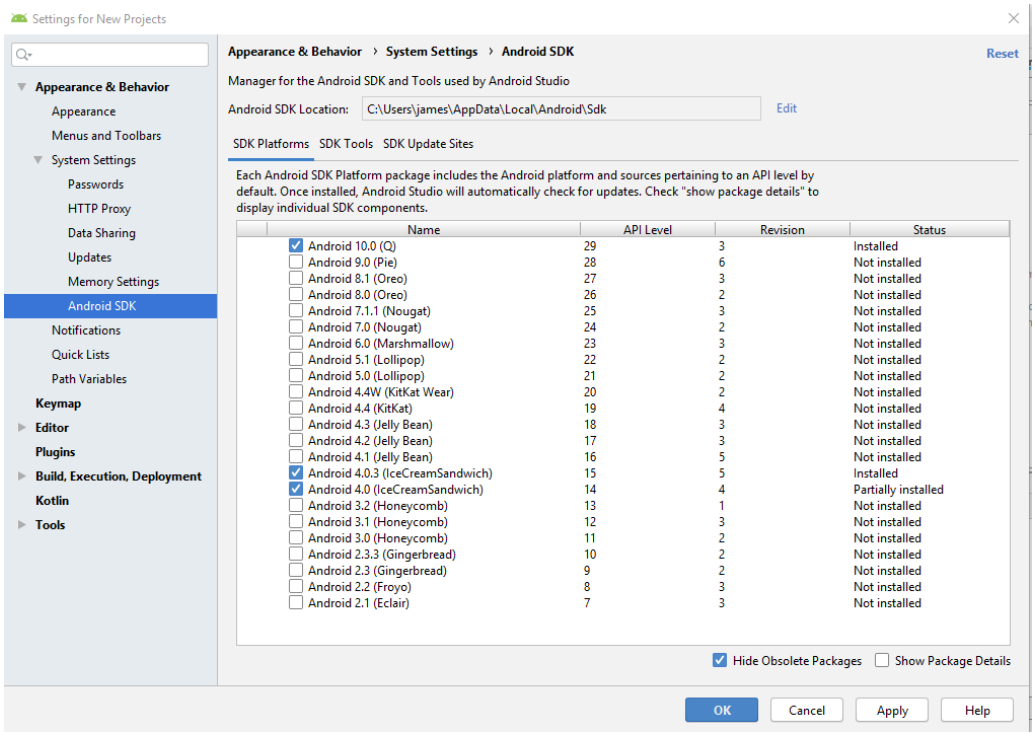
> ***Notice:*** Please replace **YourName** with **your real name** so that you can have an app with your name on it and we can know it`s yours! Do the same for the rest two below.

Android Studio 3.5 IDE

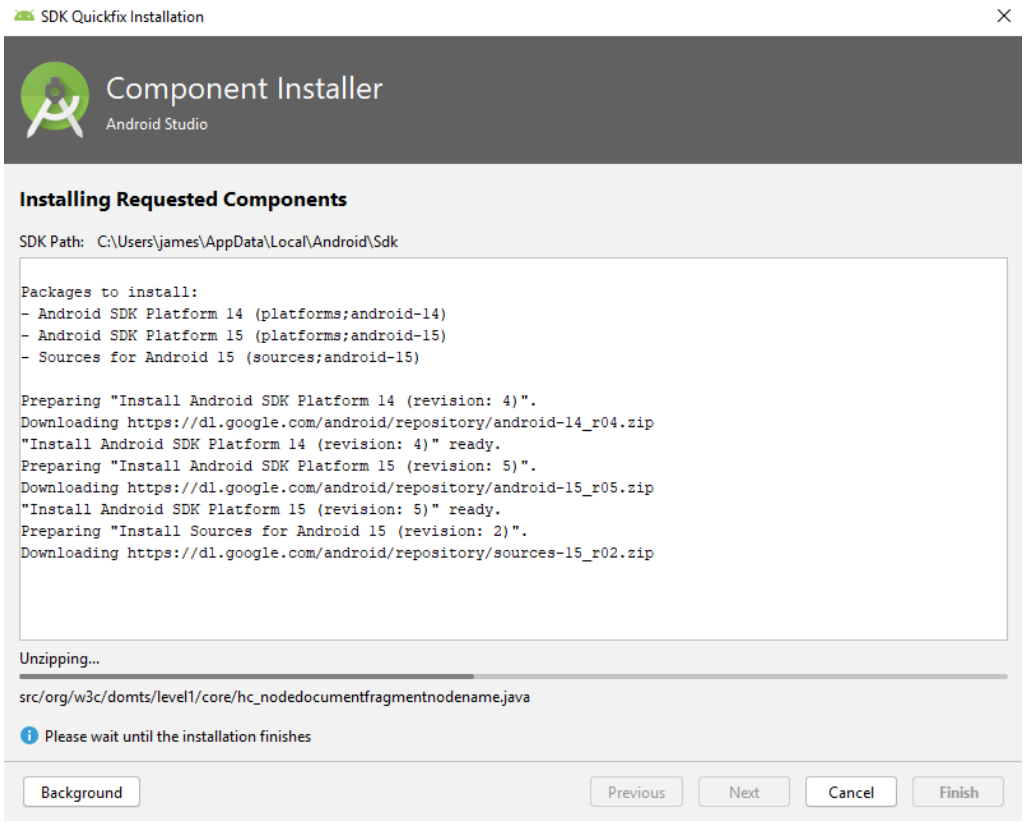


Choose and Download Android SDK

- Select **Tools -> SDK Manager**

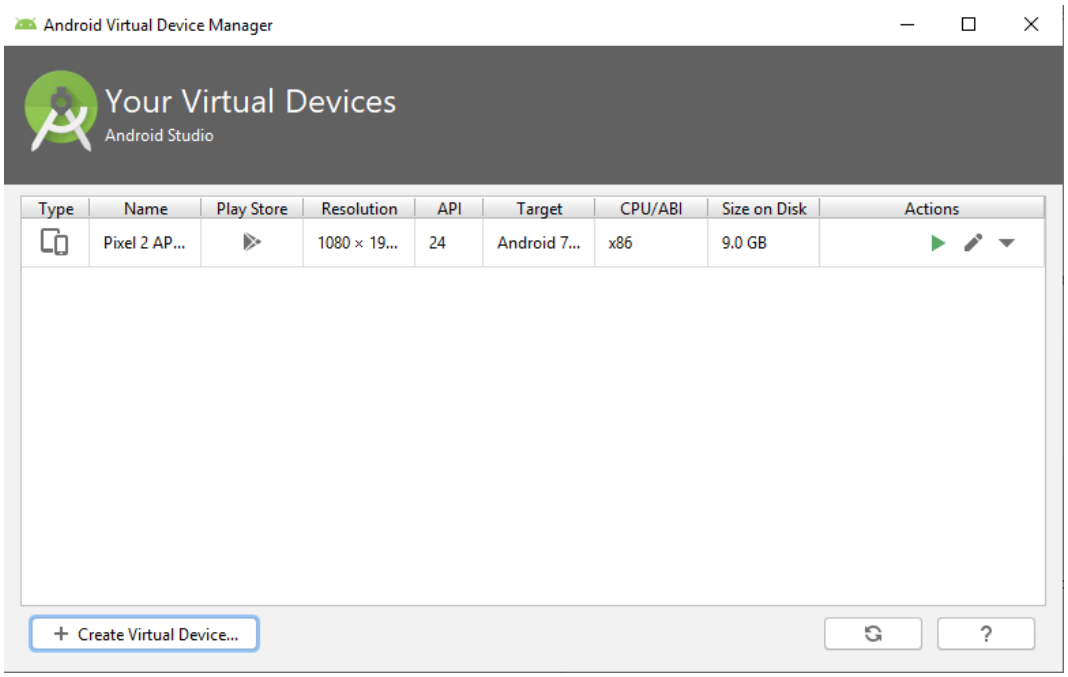


- Choose the Android SDKs to download
- Click Apply

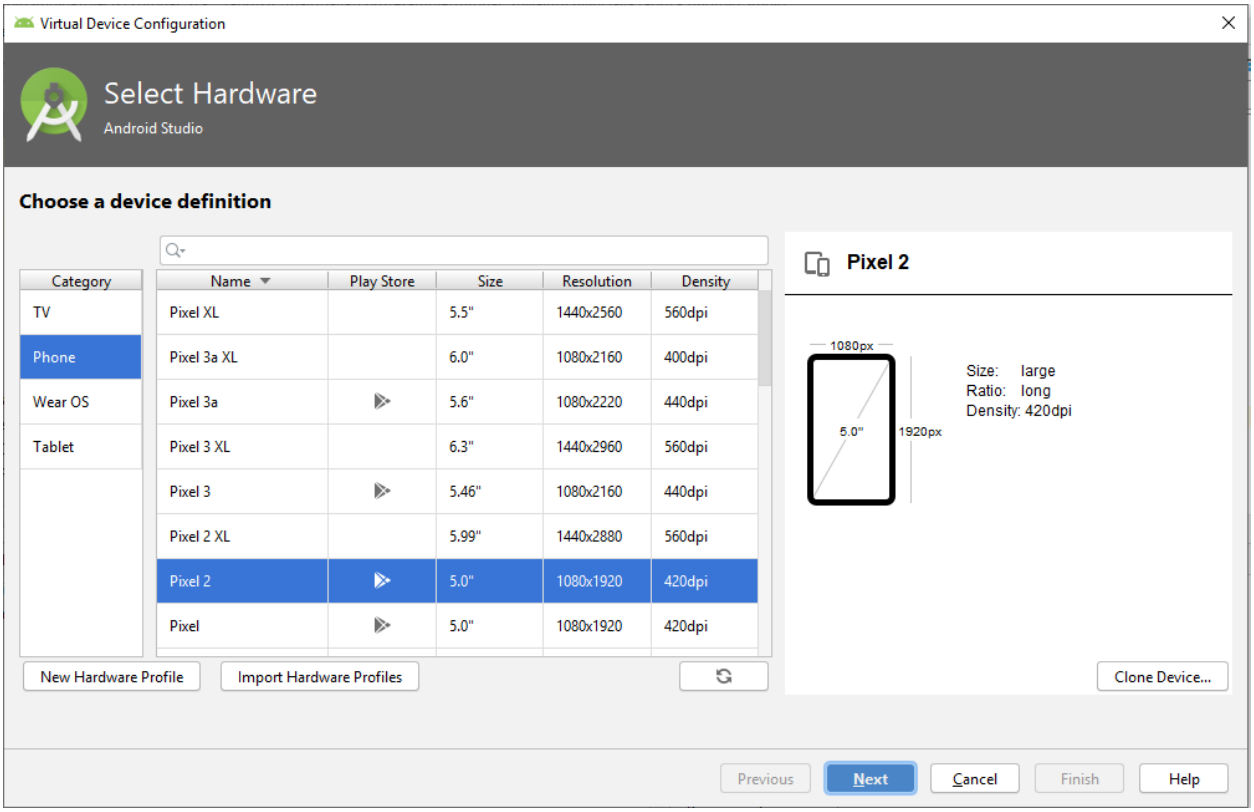


Create Android Virtual Device (AVD) Manager

- Select **Tools** -> **AVD Manager**, and then manage your Android Virtual Devices.

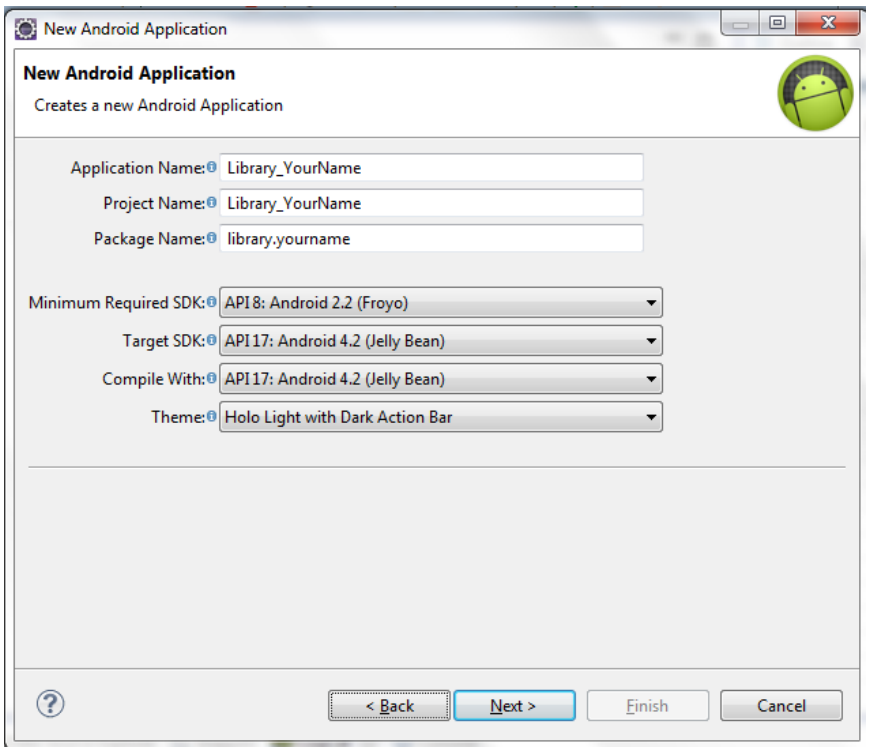


- Choose existing AVD or click + Create Virtual Device


Virtual Device Configuration
Select Hardware
Android Studio
Choose a device definition
Category
TV
Phone
Wear OS
Tablet
Name
Play Store
Size
Resolution
Density
Pixel XL 5.5" 1440x2560 560dpi
Pixel 3a XL 6.0" 1080x2160 400dpi
Pixel 3a 5.6" 1080x2220 440dpi
Pixel 3 XL 6.3" 1440x2960 560dpi
Pixel 3 5.46" 1080x2160 440dpi
Pixel 2 XL 5.99" 1440x2880 560dpi
Pixel 2 5.0" 1080x1920 420dpi
Pixel 5.0" 1080x1920 420dpi
New Hardware Profile
Import Hardware Profiles
Pixel 2
1080px
5.0"
1920px
Size: large
Ratio: long
Density: 420dpi
Clone Device...
Previous
Next
Cancel
Finish
Help

- Create an Android Project

1. Select **File** -> **New** > **Project**.
2. Select **Android** > **Android Application Project**, and click **Next**.



Create Database in Android Project

There are two ways to create database in an Android application:

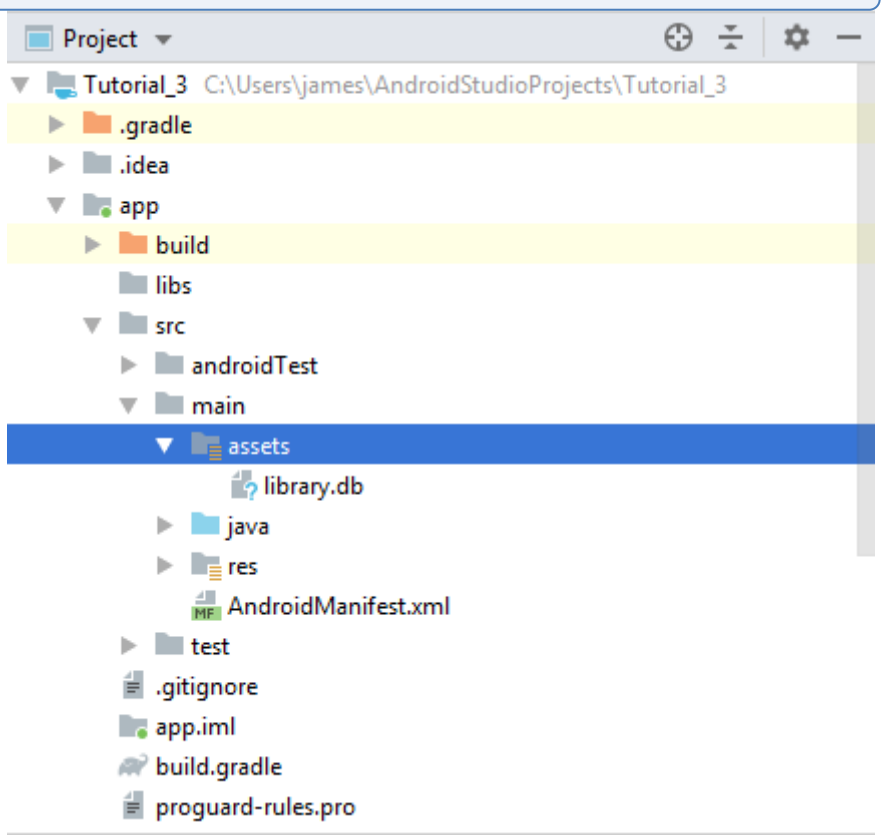
1) Create a database directly,
2) Read data from an existing SQLite file.

Since we`ve already knew how to create database using SQLite, let`s see how to use established file in Android. You can deploy the same program architecture in other Android projects.

1. Prepare database file

- ✓ From the LibDBCreateSQL.txt script build and save library.db,
- ✓ In the project window of Android Studio, right-click over app then **New->Folder->Assets Folder**
- ✓ Place **library.db** file under **assets** folder.

> ***Tips***: you can simply drag an external file into the folder you want.



2. Create packages

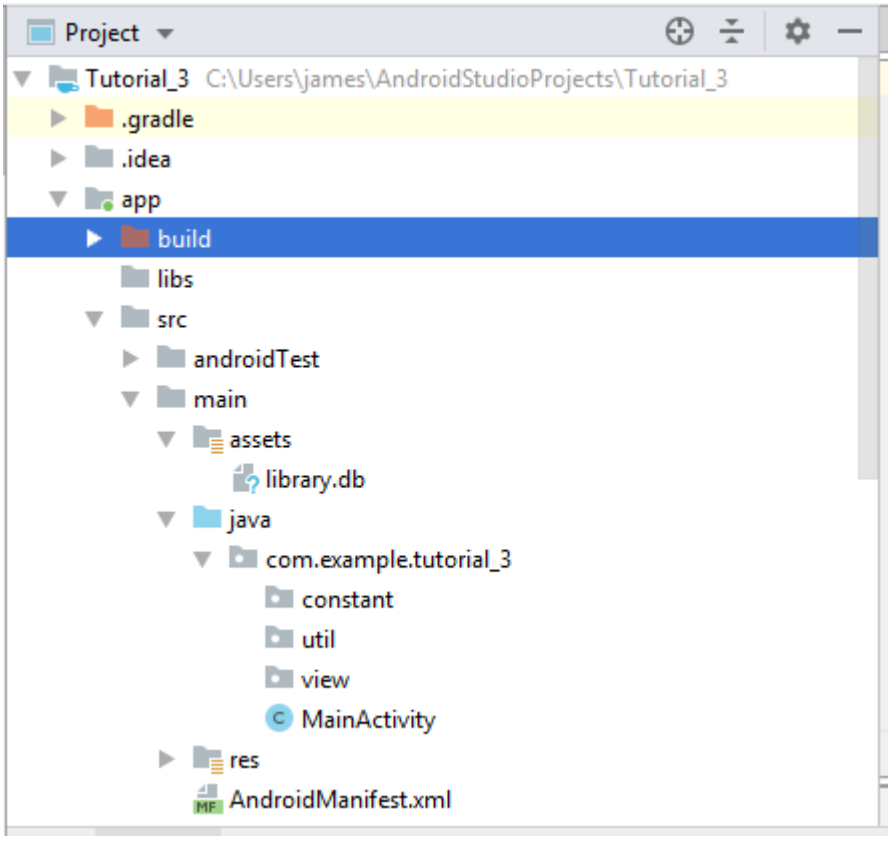
In this project, you need to create three new packages:

- ➢ **constant** – stores database constants, such as SQL and database file paths.

- **util** – contains utility classes.
- **view** – contains extensions of the existing views in the Android SDK.

Let`s create a **view** package first:

- ✓ Right click on com.example.tutorial_3 in Project Explorer -> **New** -> **Package**.
- ✓ Name the new package as "view" and **Finish**.
- ✓ Create constant package and util package on your own.
- ✓



3. Create Database classes

We need to create five classes:

- In **util** package:
  - **DBOperator** - encapsulates database operations such as query and update
  - **DBOpenHelper** – used to open database file.

  Find sample codes in Appendix 1 and copy the codes to corresponding classes.

> ***Note***: You must replace dd [yourname] in sample codes.

- In **constant** package:
  - **DBConstant** - contains all constants related to database, such as the database filenames and file paths.
  - **SQLCommand** - contains all SQL commands (select, update, delete, insert) that you may use in this project.

  Find sample codes in Appendix 2 and copy the codes to corresponding classes.

> ***Note***: You must replace dd [yourname] in sample codes.

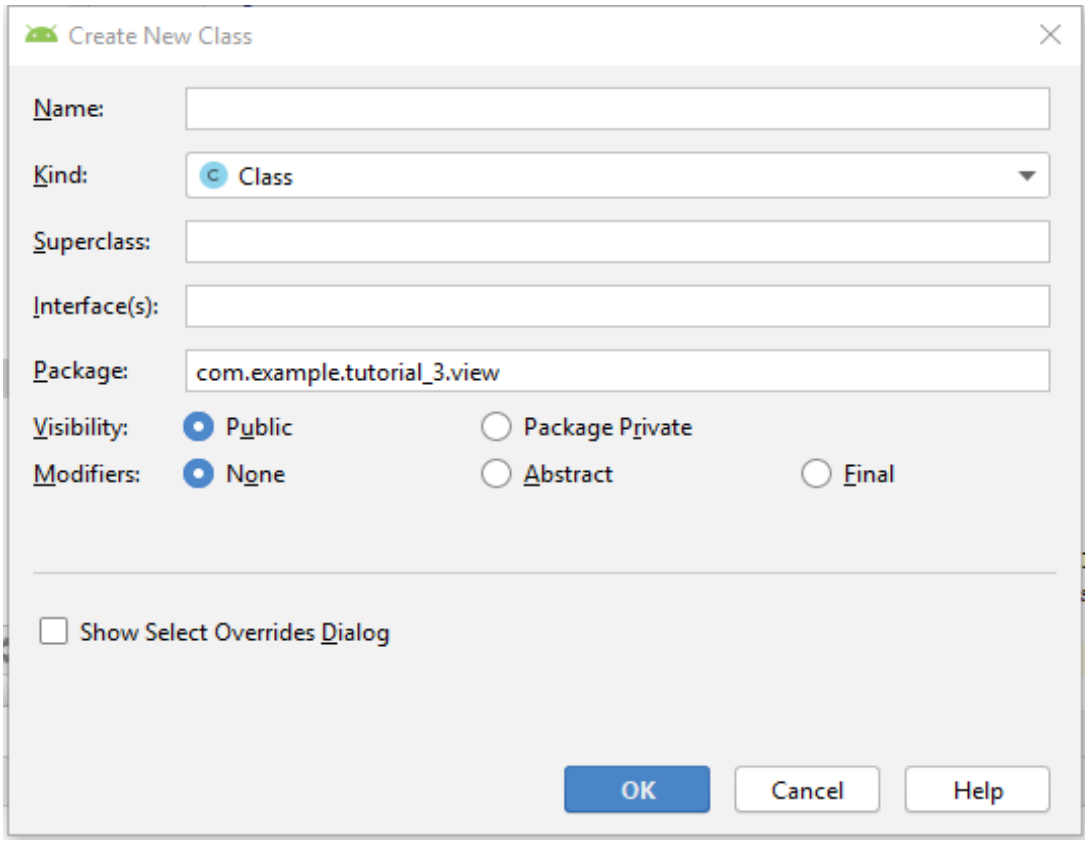
- In **view** package:
  - **TableView** - used to show all data in the result set of a SQL query.

  Find sample codes in Appendix 3 and copy the codes to your class.

> ***Note***: You must replace dd [yourname] in sample codes.

Let`s create a **TableView** class first.

✓ Right click **com.example.tutorial_3.view** package in Project Explorer -> **New** ->**Class**.

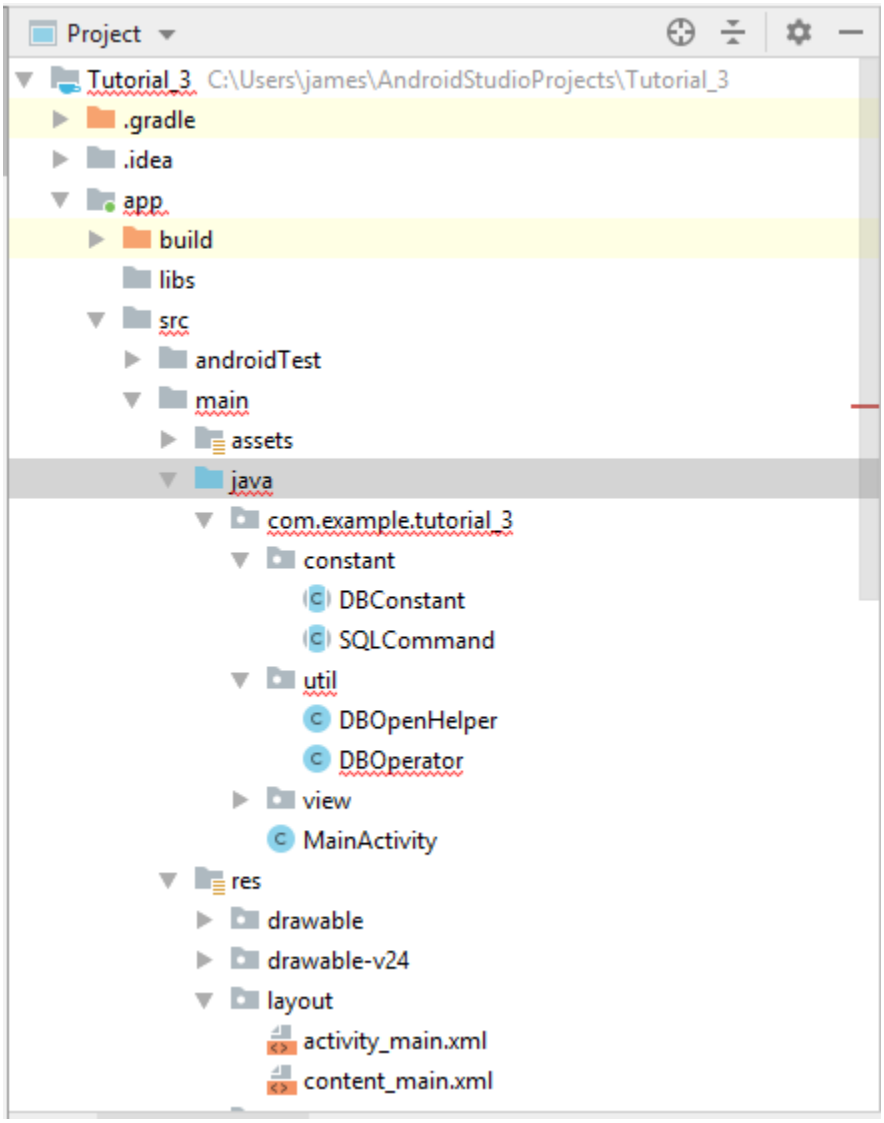


✓ Use **TableView** as the class name and **Finish**.

> ***Note***: you can choose an existing class as the superclass for this class.

4. Do it yourself

✓ Create other four classes (**DBConstant**, **SQLCommand, DBOperator**, and **DBOpenHelper**) in corresponding packages on your own.



5. XML File

Although we`ll learn how to build friendly user interface in detail next week, let`s experience how GUI works first.

- **XML Files activity_main.xml, context_main.xml**

- ✓ Expand **com.example.tutorial_3** project-> expand **res** folder-> expand **layout** folder.
- ✓ Right click on **activity_main.xml** and you will be able to view its content

- **Modify content_main.xml**
  - ✓ Delete the 'Hello World' text.
  - ✓ Modify **layout_width** and **layout_height** attribute of **ScrollView** to "match_parent".

The updated main.xml to add is shown below.

```
        <LinearLayout
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"
         tools:context=".YourNameActivity" >

          <ScrollView
           android:id="@+id/scrollView1"
           android:layout_width="match_parent"
           android:layout_height="match_parent" >
          </ScrollView>

        </LinearLayout>
```

6. **Finish main activity**

   Copy the codes in Appendix 4 to the Main Java class under com.example.tutorial_3. This class completes two tasks: copy database file from assets folder to Android application; query all records in Student table and show them on screen.

7. **Run Application**

   Click the **Run**

Note: You must replace dd [yourname] in sample codes.

Tips: If you failed to load the app, you can remove the app and restart the device to load the app again. In the app view, choose **Settings** -> **Apps**-> **Library_YourName** -> **Uninstall & Clear data**. Then restart the device by closing the device window and run the app again.

What to submit

One Screenshot

Please capture the screenshots of your completed Android Database Project and paste them into a Word document.

The screenshot should look like:

All deliverables should be submitted via Canvas assignment manager as a <u>single Word or PDF</u> document by the due date.

Appendix 1

DBOperator

```
package com..util;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import library.[yourname].constant.DBConstant;

import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

/**
 * Class to manipulate tables & data
 * Uses singleton pattern to create single instance
 */
public class DBOperator
{
	private static DBOperator instance = null;
	private SQLiteDatabase db;

	private DBOperator()
	{
		//path of database file
		String path = DBConstant.DATABASE_PATH + "/" + DBConstant.DATABASE_FILE;
		db = SQLiteDatabase.openDatabase(path, null,
SQLiteDatabase.OPEN_READWRITE);
	}
	/*
	 * Singleton Pattern
	 * Why should we avoid multiple instances here?
	 */
	public static DBOperator getInstance()
	{
		if (instance==null) instance = new DBOperator();
		return instance;
	}
	/**
	 * Copy database file
	 * From assets folder (in the project) to android folder (on device)
	 */
	public static void copyDB(Context context) throws
IOException,FileNotFoundException{
		String path = DBConstant.DATABASE_PATH + "/" + DBConstant.DATABASE_FILE;
		File file = new File(path);
		if (!file.exists()){
```

```
                    DBOpenHelper dbhelper = new DBOpenHelper(context, path ,1);
                    dbhelper.getWritableDatabase();
                    InputStream is =
context.getAssets().open(DBConstant.DATABASE_FILE);
                    OutputStream os = new FileOutputStream(file);
                    byte[] buffer = new byte[1024];
                    int length;
                    while ((length = is.read(buffer))>0){
                          os.write(buffer, 0, length);
                    }
                    is.close();
                    os.flush();
                    os.close();
             }
      }

      /**
       * execute sql without returning data, such as alter
       * @param sql
       */
      public void execSQL(String sql) throws SQLException
      {
             db.execSQL(sql);
      }
      /**
       * execute sql such as update/delete/insert
       * @param sql
       * @param args
       * @throws SQLException
       */
      public void execSQL(String sql, Object[] args)
      {
             db.execSQL(sql, args);
      }
      /**
       * execute sql query
       * @param sql
       * @param selectionArgs
       * @return cursor
       * @throws SQLException
       */
      public Cursor execQuery(String sql,String[] selectionArgs) {
             return db.rawQuery(sql, selectionArgs);
      }
      /**
       * execute query without arguments
       * @param sql
       * @return
       * @throws SQLException
       */
      public Cursor execQuery(String sql)    {
             return this.execQuery(sql, null);
      }
      /**
       * close database
```

```
         */
        public void closeDB()
        {
                if (db!=null) db.close();
        }
}
```

DBOpenHelper

```
package com..util;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/**
 * Class used to open database file
 */
public class DBOpenHelper extends SQLiteOpenHelper {

        public DBOpenHelper(Context context, String path, int version){
                super(context, path, null, version);
        }
        @Override
        public void onCreate(SQLiteDatabase db) {
        }
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        }

}
```

Appendix 2

DBConstant

```
package com….constant;

/**
 * Constants related to database file
 * Such as file path and file name
 */
public abstract class DBConstant
{
	//database file directory
	public static String DATABASE_PATH =
"/data/data/library.[yourname]/databases";
	//database file name
	public static String DATABASE_FILE = "library.db";
	//database version
	public static int DATABASE_VERSION = 1;
}
```

SQLCommand

```
/**
 * SQL commands
 * Including select/delete/update/insert
 */
package com….constant;

public abstract class SQLCommand
{
	//query all students
	public static String QUERY_STUDENT = "select stid, stname from Student";
}
```

Appendix 3

TableView

```
package com…..view;

import com…..util.DBOperator;
import android.content.Context;
import android.database.Cursor;
import android.view.View;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
/**
 * This class is used to show data in database
 * It is an extension of TableLayout
 */
public class TableView extends TableLayout
{
	public TableView(Context context, String tableName)
	{
		super(context);
		String sql = "select * from " + tableName + ";";
		DBOperator op = DBOperator.getInstance();
		Cursor cursor = op.execQuery(sql, null);
		this.extractData(context, cursor);
	}
	public TableView(Context context, Cursor cursor)
	{
		super(context);
		this.extractData(context, cursor);
	}
	/*
	 * fill data in table view with a cursor
	 */
	private void extractData(Context context,Cursor cursor)
	{
		TextView textView;
		TableRow row;
		boolean first = true;
		while (cursor.moveToNext())
		{
			/*
			 * Before displaying the first row,
			 * display column names as a header
			 */
			if (first){
				textView = new TextView(context);
				String[] columnNames = cursor.getColumnNames();
				StringBuilder strBuilder = new StringBuilder();
				for (int i=0;i<columnNames.length;i++){
					if (i>0) strBuilder.append("|");
					strBuilder.append(columnNames[i]);
				}
```

```
                        textView.setText(strBuilder);
                        this.addView(textView);
                        //show separation line
                        View line = new View(context);
                        line.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,2));
                        line.setBackgroundColor(0xFF909090);
                        this.addView(line);
                        first = false;
                    }
                    //show values in a row
                    row = new TableRow(context);
                    int length = cursor.getColumnCount();
                    for (int i=0;i<length;i++)
                    {
                        if (i>0){
                            textView = new TextView(context);
                            textView.setText("|");
                            row.addView(textView);
                        }
                        textView = new TextView(context);
                        textView.setText(cursor.getString(i));
                        row.addView(textView);
                    }
                    this.addView(row);
                }
                /*
                 * Do not forget to close the cursor!
                 * Otherwise database exceptions will be thrown
                 */
                cursor.close();
        }
}
```

Appendix 4

MainActivity

```
package com…;

import com….constant.SQLCommand;
import com…view.TableView;
import com….util.DBOperator;
import android.app.Activity;
import android.database.Cursor;
import android.os.Bundle;
import android.widget.ScrollView;

public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_yourname_layout);
             //copy database file
             try{
                   DBOperator.copyDB(getBaseContext());
             }catch(Exception e){
                   e.printStackTrace();
             }
             //implement SQL query and get cursor of resultset
             Cursor cursor =
DBOperator.getInstance().execQuery(SQLCommand.QUERY_STUDENT);
             TableView tableView = new TableView(this, cursor);
             //show data in tableview
             ScrollView scrollView = (ScrollView)this.findViewById(R.id.scrollView1);
             scrollView.addView(tableView);
    }
}
```