**Birzeit University**
**Faculty of Engineering and Technology**
**Electrical and Computer Engineering Department**
**Advance Computer Systems Engineering Lab ENCS515**

# EXP. No. 1. Introduction to Android Programming

## 1. Objectives

- ❖ Download Android Studio software.
- ❖ Create a “Hello World” Android Application.
- ❖ Design a New Application and understand the various parts of an Android.
- ❖ Install and run the application on Android Emulator and physical device.

## 2. Introduction

The goal of this lab is to learn the fundamentals of developing Android Applications, from project creation to installation on a physical device. More specifically, you should gain the knowledge of how to use basic development tools to support the application development process, as well as the key components of an Android application itself.

## *2.1. Activity Classes*

There are four major types of component classes in any Android application:

- Activities: Much like a Form for a web page, activities display a user interface for performing a single task. An example of an Activity class would be one which displays a Login Screen to the user.
- Services: These differ from Activities in that they have no user interface. Services run in the background to perform some sort of task. An example of a Service class would be one which fetches your email from a web server.
- Broadcast Receivers: The sole purpose of components of this type is to receive and react to broadcast announcements which are either initiated by system code or other applications. If you’ve ever done any work with Java Swing, you can think of these like Event Handlers. For example, a broadcast announcement may be made to signal that a Wi-Fi connection has been established. A Broadcast Receiver for an email application listening for that broadcast may then trigger a Service to fetch your email.
- Content Providers: Components of this type function to provide data from their application to other applications. Components of this type would allow an email application to use the phone’s existing contact list application for looking up and retrieving an email address.

## *2.2. Android Manifest file*

Every project has a file with this exact name in the root directory. It contains all the information about the application that Android will need to run it: item Package name used to identify the application. item List of Activities, Services, Broadcast Receivers, and Content Provider classes and all their necessary information, including permissions. item System Permissions the application must define in order to make use of various system resources, like GPS. item Application defined permissions that other applications must have in order to interact with this application. item Application profiling information. item Libraries and API levels that the application will use.

### *2.3. R.java Class*

This is a special static class that is used for referencing the data contained in your resource files. If you open this file you will see several static inner classes for each of the resource types, as well as static constant integers within them. Notice that the names of the member variables are the same as the names of the values in your resource files. Each value in a resource file is associated with an integer ID, and that ID is stored in a member variable of the same name, within a static class named after its data type.

Android projects has 2 parts: Design and Coding. The components that are taken in design window are given a specific id. This id is given so that during multiple identical components i.e. 2 textview or 2 buttons, they can be distinctly recognized. When you do the coding part, and you want to make a component functionable then you first initialize it using findViewById method. Its identical to the initialization of data variable in C language. This statement just connects designing with the coding on per component basis.

## 3. Procedure

Before starting a new project, you need to be connected to the internet once you create a new project. In this lab you are asked to build a simple application to get the user’s name from text field and then display it on label. You will be creating an Activity class which will allow the user to enter their name into a text field and press a button.

### *3.1. Creating new Android Project in Android Studio*

Before moving on to slightly more advanced topics, now is a good time to validate that all the required development packages are installed and functioning correctly. The best way to achieve this goal is to create an Android application and compile and run it. This section will cover the creation of a simple Android application project using Android Studio. Once the project has been created, the next section will explore the use of the Android emulator environment to perform a test run of the application.

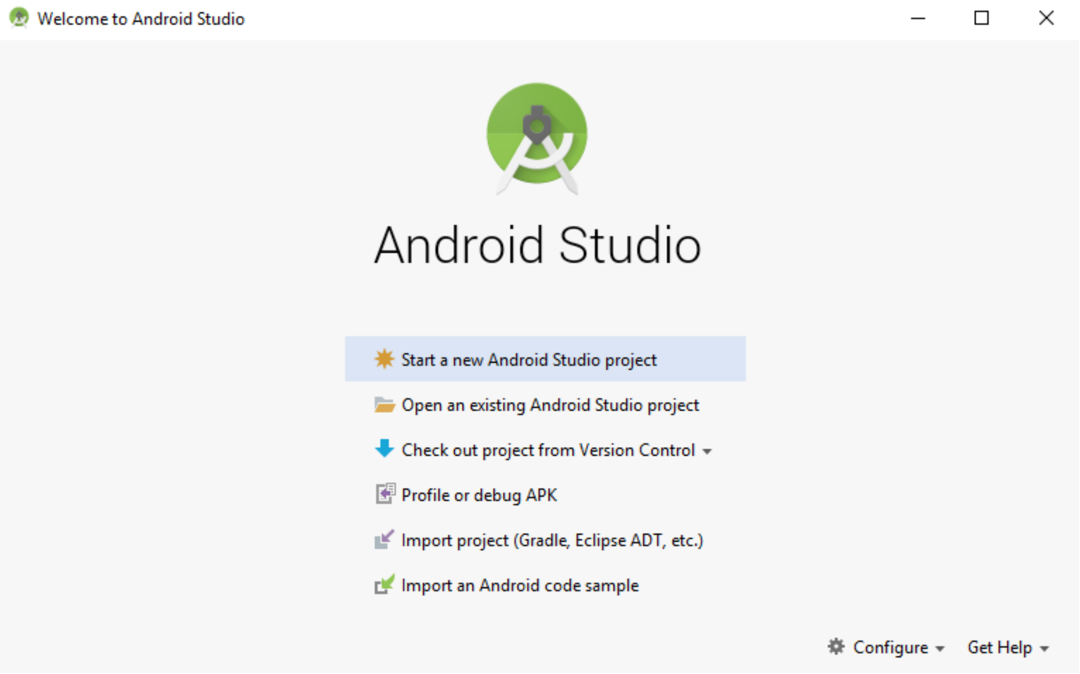
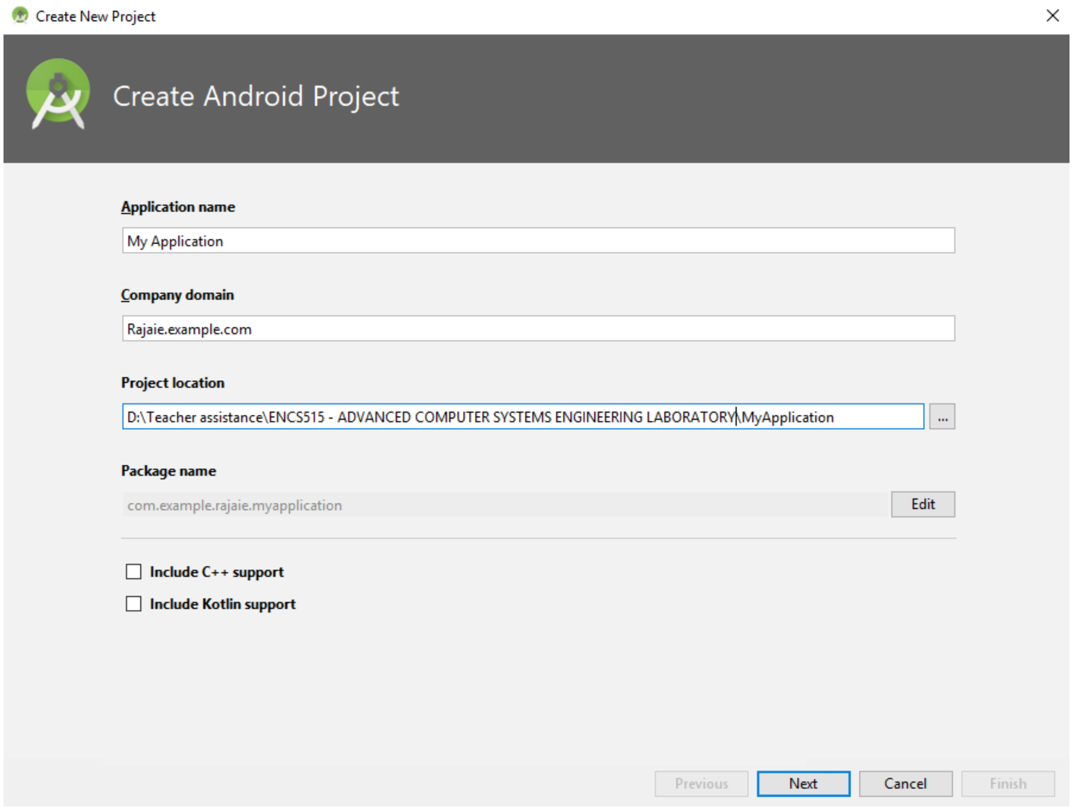

Welcome to Android Studio
Android Studio
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

*Figure 1.1 welcome to Android Studio Screen*



Application name: My Application

Company domain: Rajaie.example.com

Project location: D:\Teacher assistance\ENCS515 - ADVANCED COMPUTER SYSTEMS ENGINEERING LABORATORY\MyApplication

Package name: com.example.rajaie.myapplication

☐ Include C++ support

☐ Include Kotlin support

*Figure 1.2 Create new Project Screen*

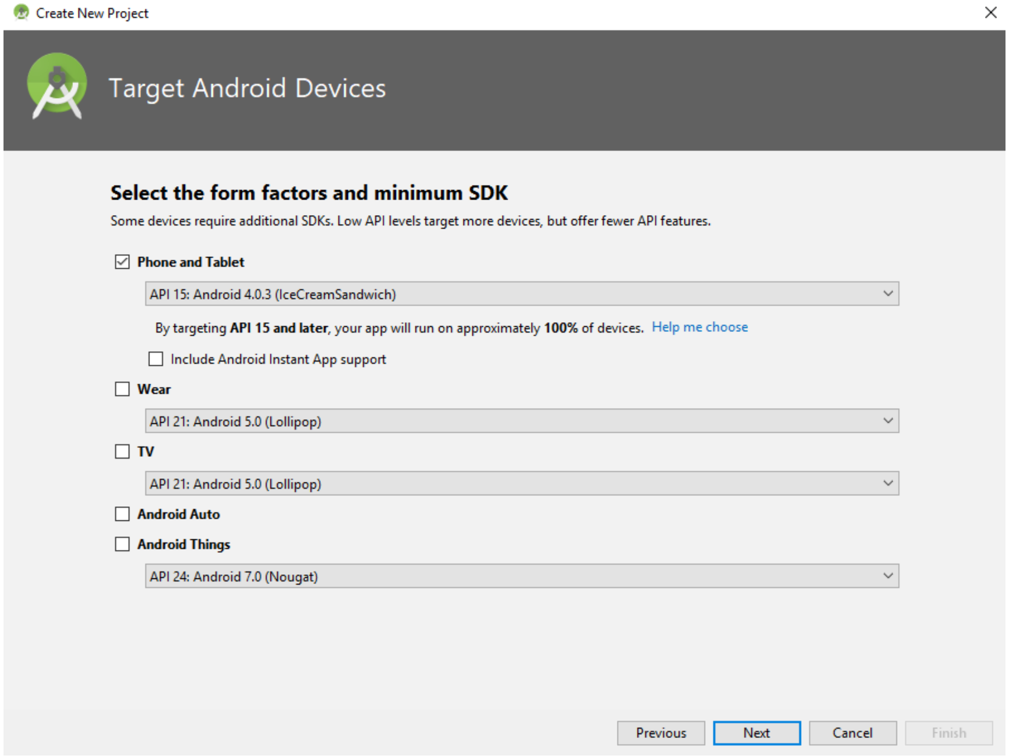
### *3.2. Creating a New Android Project*

The first step in the application development process is to create a new project within the Android Studio environment. Begin, therefore, by launching Android Studio so that the “Welcome to Android Studio” screen appears as illustrated in Figure 1.1.

Once this window appears, Android Studio is ready for a new project to be created. To create the new project, simply click on the Start a new Android Studio project option to display the first screen of the New Project wizard as shown in Figure 1.2.

### *3.3. Defining the Project and SDK Settings*

In the New Project window, set the Application name field to “My Application”. The application name is the name by which the application will be referenced and identified within Android Studio and is also the name that will be used when the completed application goes on sale in the Google Play store. The Package Name is used to uniquely identify the application within the Android application ecosystem. It should be based on the reversed URL of your domain name followed by the name of the application. For example, if your domain is www.mycompany.com, and the application has been named “My Application”, then the package name might be specified as follows: com.mycompany.myapplication If you do not have a domain name, you may also use ebookfrenzy.com for the purposes of testing, though this will need to be changed before an application can be published: com.ebookfrenzy.myapplication The Project location setting will default to a location in the folder named AndroidStudioProjects located in your home directory and may be changed by clicking on the button to the right of the text field containing the current path setting. Click Next to proceed. On Target Android Devices screen as shown in Figure 1.3, enable the Phone and Tablet option and set the minimum SDK setting to API 15: Android 4.0.3 (IceCreamSandwich) and click on Next. The reason for selecting an older SDK release is that this ensures that the finished application will be able to run on the widest possible range of Android devices.



Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☑ **Phone and Tablet**

API 15: Android 4.0.3 (IceCreamSandwich)

By targeting **API 15 and later**, your app will run on approximately **100%** of devices. Help me choose

□ Include Android Instant App support

□ **Wear**

API 21: Android 5.0 (Lollipop)

□ **TV**

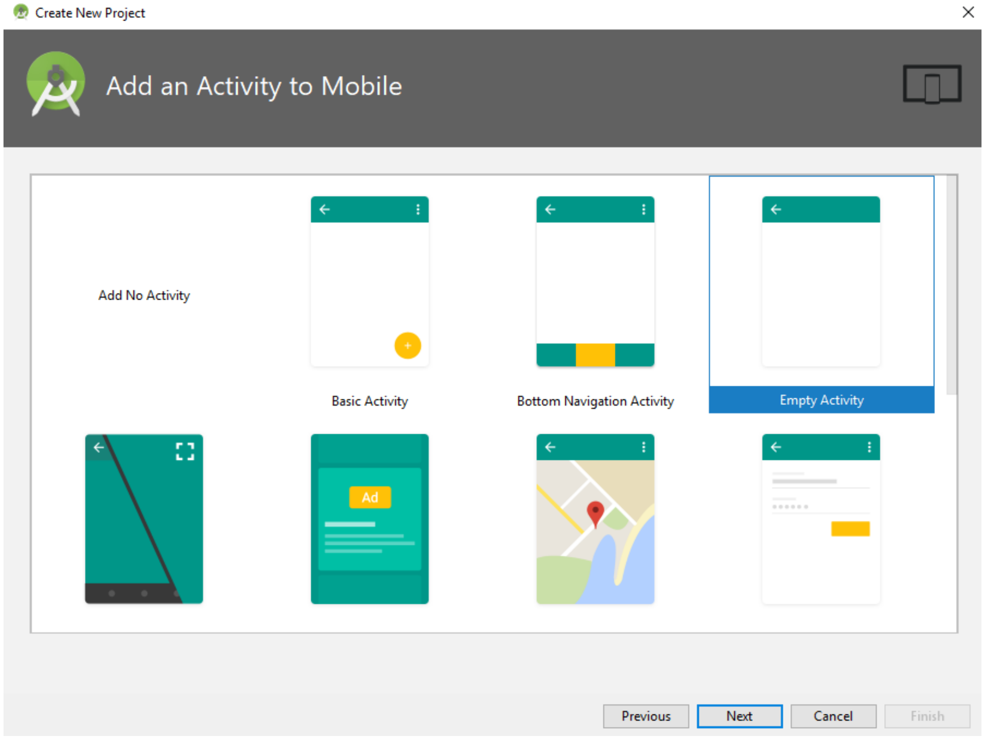
API 21: Android 5.0 (Lollipop)

□ **Android Auto**

□ **Android Things**

API 24: Android 7.0 (Nougat)

*Figure 1.3 Target Android Devices Screen*



Create New Project

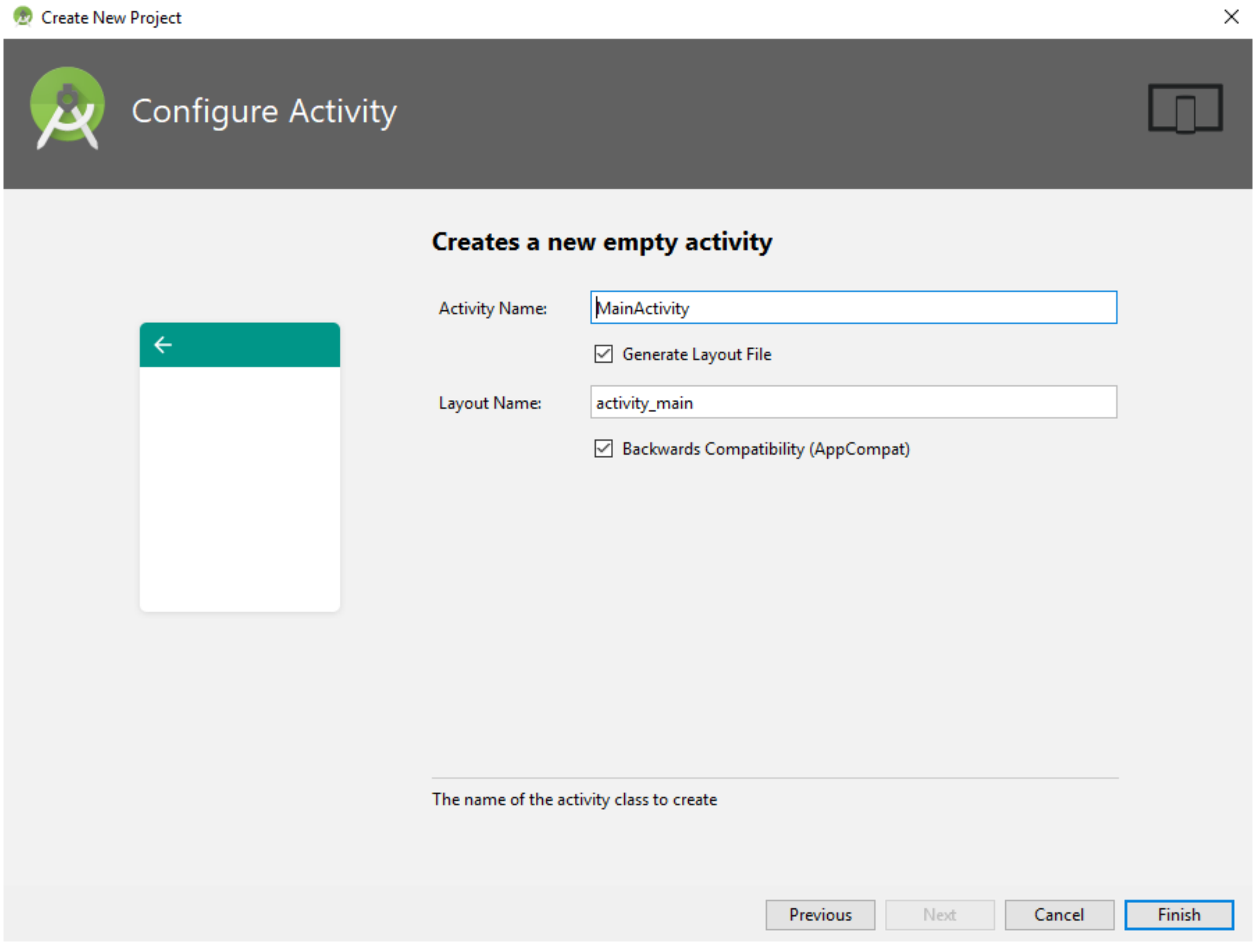
Add an Activity to Mobile

*Figure 1.4 Add an Activity to Mobile Screen*

### *3.4. Creating an Activity*

The next step is to define the type of initial activity that is to be created for the application. A range of different activity types is available when developing Android applications. For the purposes of this example, however, simply select the option to create an Empty Activity as shown in Figure 1.4.

With the Empty Activity option selected, click Next. On the final screen as shown in Figure 1.5 name the activity and title MainActivity. The activity will consist of a single user interface screen layout which, for the purposes of this example, should be named activity_main shown in Figure 1.5. Finally, click on Finish to initiate the project creation process.



**Creates a new empty activity**

Activity Name: MainActivity

☑ Generate Layout File

Layout Name: activity_main

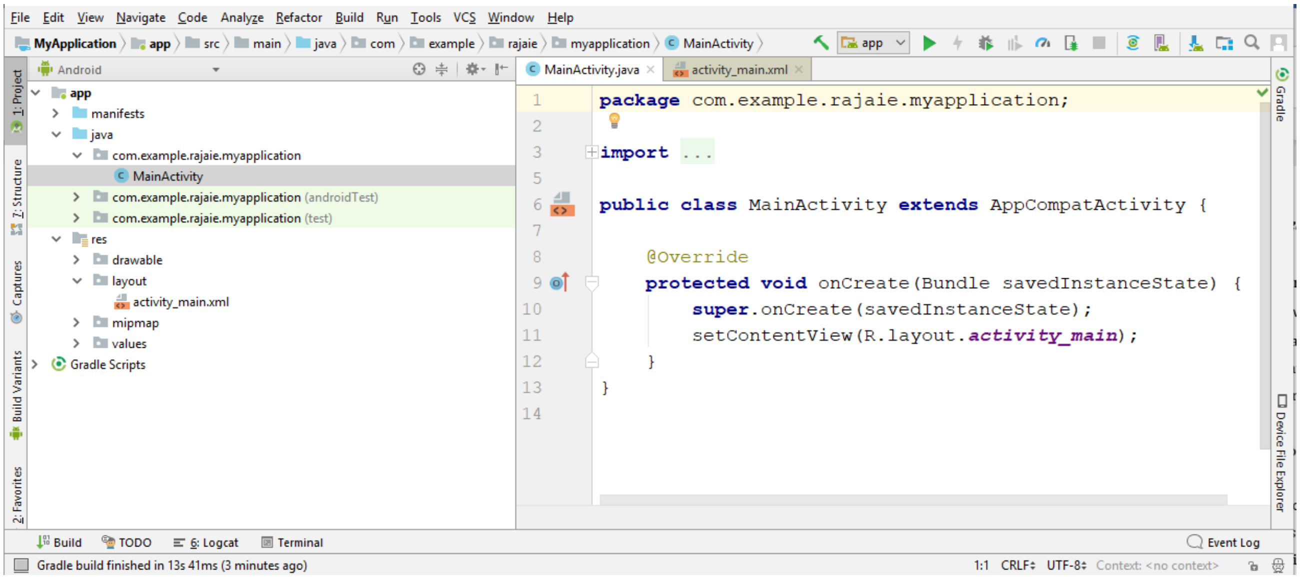
☑ Backwards Compatibility (AppCompat)

The name of the activity class to create

*Figure 1.5 Configure Activity Screen*

## 3.5. *Modifying the Created Application*

At this point, Android Studio has created a minimal example application project and opened the main window as shown in Figure 1.6.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MyApplication app src main java com example rajaie myapplication MainActivity
app
Android
MainActivity.java
activity_main.xml
app
manifests
java
com.example.rajaie.myapplication
MainActivity
com.example.rajaie.myapplication (androidTest)
com.example.rajaie.myapplication (test)
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
1: Project
7: Structure
Captures
Build Variants
2: Favorites
Gradle
Device File Explorer
Build
TODO
6: Logcat
Terminal
Event Log
1:1 CRLF UTF-8 Context: <no context>

```
package com.example.rajaie.myapplication;

import ...

public class MainActivity extends AppCompatActivity {

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

Gradle build finished in 13s 41ms (3 minutes ago)

*Figure 1.6 Android Studio Application screen*

The newly created project and references to associated files are listed in the Project tool window located on the left-hand side of the main project window. The Project tool window has several modes in which information can be displayed. By default, this panel will be in Android mode. This setting is controlled by the drop-down menu at the top of the panel. If the panel is not currently in Android mode, click on this menu and switch to Android mode.

when finished to proceed to the HelloWorld greeting Activity. The example project created for us when you selected the option to create an activity consists of a user interface containing a label that will read “Hello World” when the application is executed.

The user interface design for our activity is stored in a file located under app, res, layout in the project file hierarchy. Using the Project tool window, locate this file as shown in Figure 1.7.

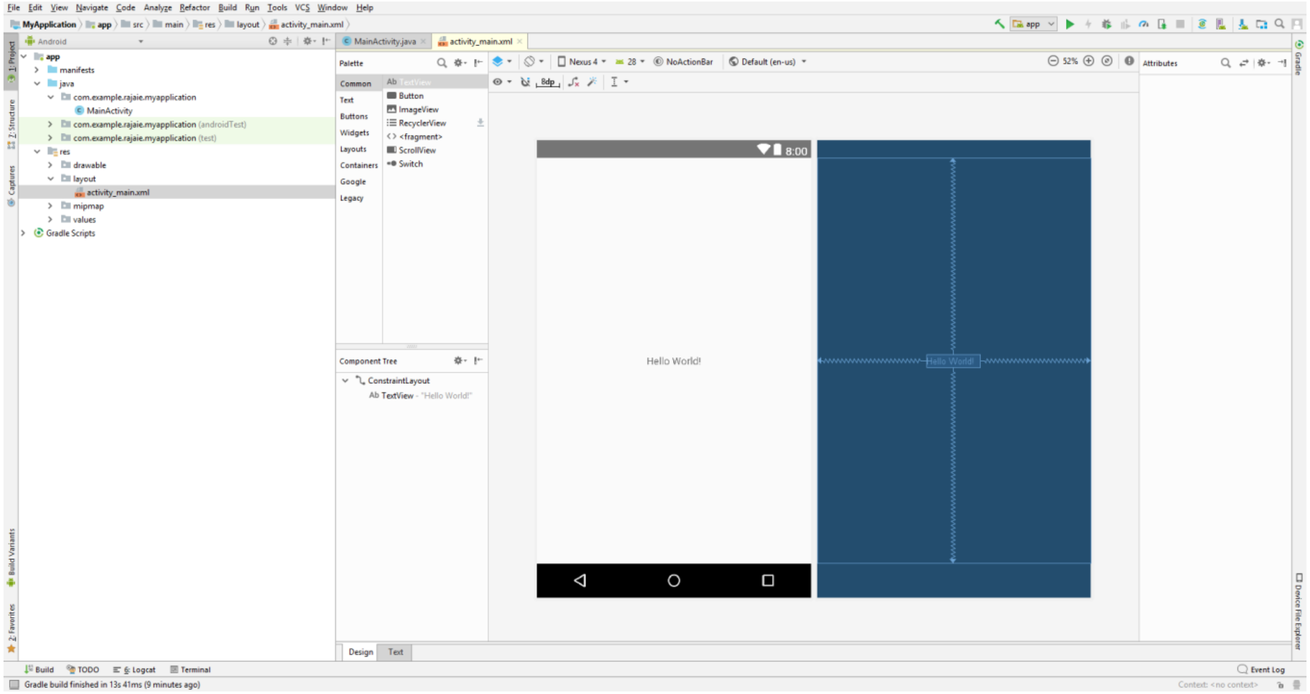


*Figure 1.7 Layout Screen*

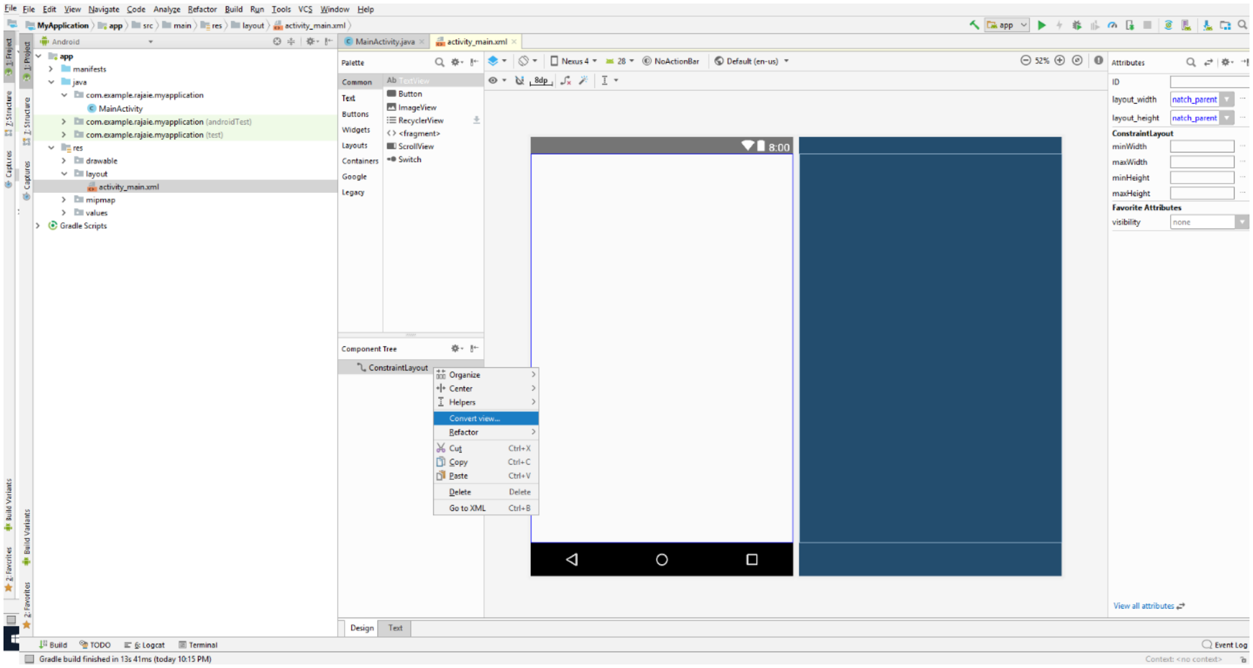
Once located, double click on the file to load it into the User Interface Designer tool which will appear in the center panel of the Android Studio main window. In the toolbar across the top of Designer window is a menu currently set to Nexus 4 which is reflected in the visual representation of the device within the Designer panel. A wide range of other device options are available for selection by clicking on this menu.

As can be seen in the device screen, the layout already includes a label that displays a Hello World! message. Running down the left-hand side of the panel is a palette containing different categories of user interface components that may be used to construct a user interface, such as buttons, labels and text fields.

- The first step in modifying the application is to delete the “Hello World!” TextView component from the design. Begin by clicking on the TextView object within the user interface view so that it appears with a blue border around it. Once selected, press the Delete key on the keyboard to remove the object from the layout.
- The Android Studio Designer tool also provides an alternative to dragging and dropping components from the palette on to the design layout. Components may also be added by

selecting the required object from the palette and then simply clicking on the layout at the location where the component is to be placed

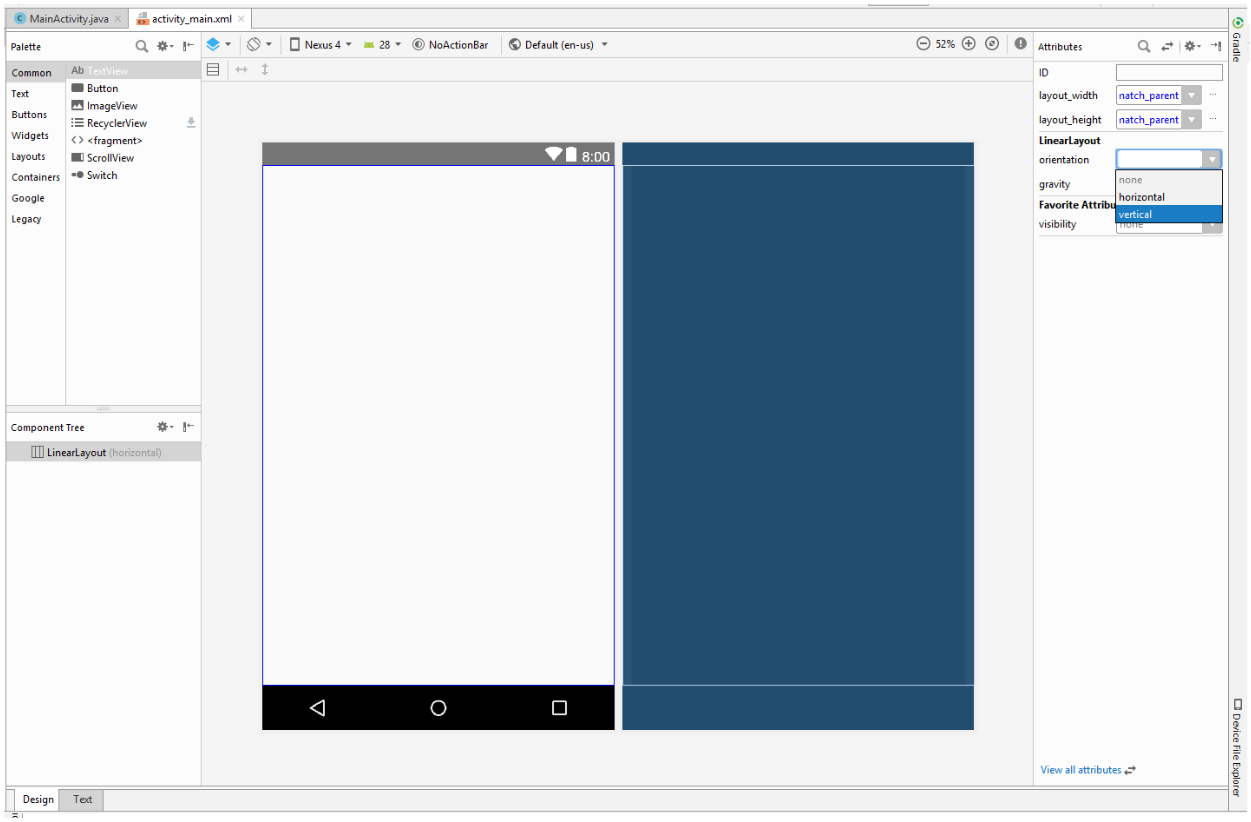
- Start by pressing right click on the constraint layout (which is considered as the main layout) and click on convert layout to convert it to linear Layout as shown in Figure 1.8.
- Set the orientation to Vertical by selecting the Linear layout on the left-hand of the panel and changing the orientation to Vertical from the right-side of the panel as in Figure 1.9.
- Find and drag Plain Text which exists in Text fields on the left-side of the panel as shown in Figure 1.10. Then find and drag Button and TextView Form Widgets panel on the left-side of the panel. You can see the layout for the application in Figure 1.11.



ID
layout_width natch_parent
layout_height natch_parent
ConstraintLayout
minWidth
maxWidth
minHeight
maxHeight
Favorite Attributes
visibility none

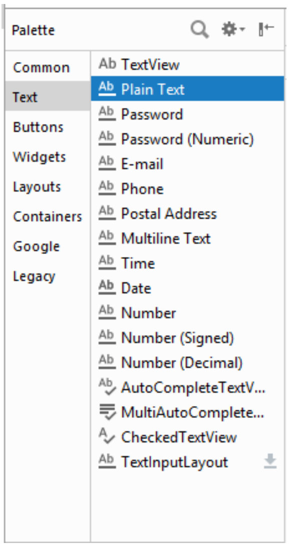
Gradle build finished in 13s 41ms (today 10:15 PM)

*Figure 1.8 Converting the Constraint layout Screen*



ID
layout_width natch_parent
layout_height natch_parent
LinearLayout
orientation vertical
gravity
visibility none

*Figure 1.9 Changing the Layout to Vertical Layout Screen*



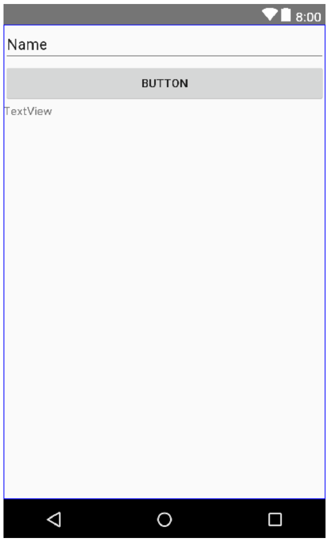


*Figure 1.10 Adding Plain Text Screen*

*Figure 1.11 Application Layout Design*

- Open MainActivity.java file. Add the following code inside the onCreate method in the MainActivity Class (don’t remove the first two lines in the onCreate method) (the setContentView is for linking the MainActivity java file to the activity_main layout using the R.java file).

```
Button button = (Button)findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        EditText editText=(EditText)findViewById(R.id.editText);
        TextView textView=(TextView) findViewById(R.id.textView);
        textView.setText(editText.getText().toString());
    }
});
```

- Your application is ready now, you can build the application and test it by running it on a real device or on a virtual device.

## 4. Todo

Ask the Instructor or the Teacher Assistant for the todo (it will be specified at the end of the lab) you should perform the todo on your own machine and show it next lab.