ENGG2990D Team Airship Design

GETTING STARTED WITH ANDROID DEVELOPMENT

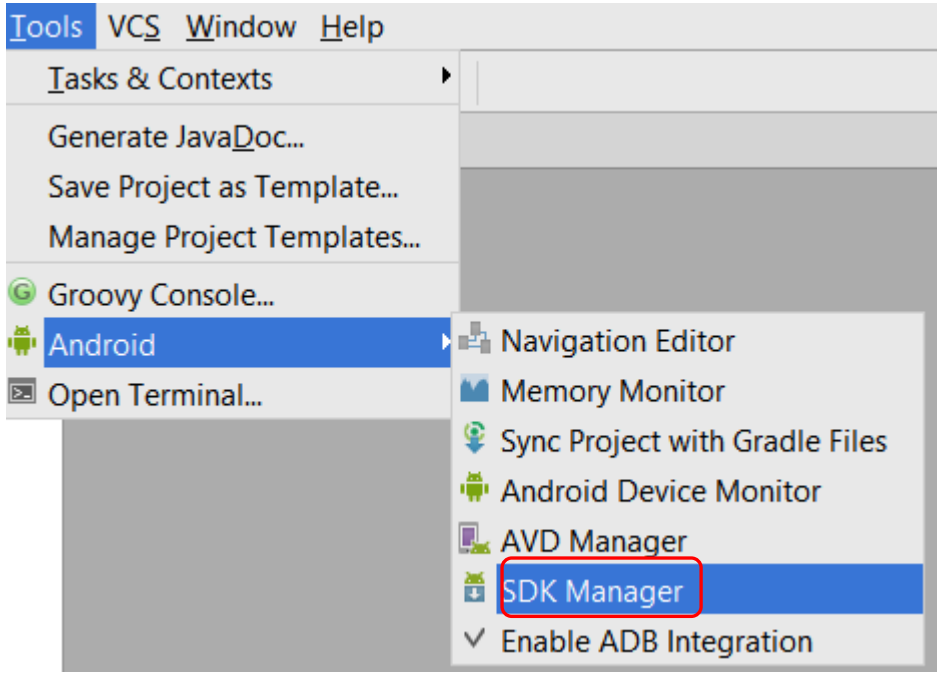
Open up Android Studio

Refer to Android studio installation, or follow guidelines from official website.

Via http://developer.android.com/sdk/index.html

Select workspace directory at your convenience if they ask to do so.

Open SDK manager



Download appropriate SDK packages

Android SDK tool 24.0.2

Android SDK platform tools

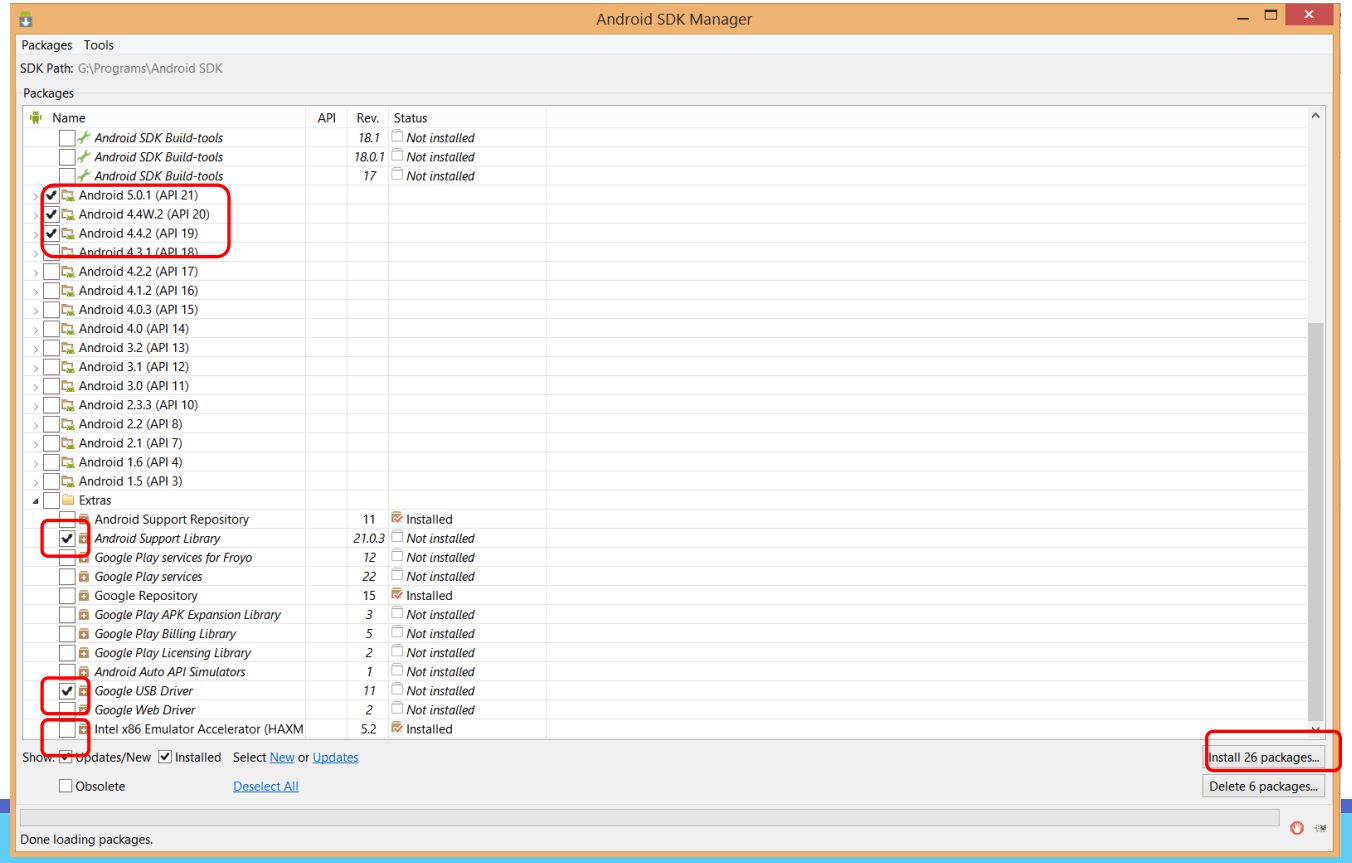
Android SDK build tools

Whole folder for API21

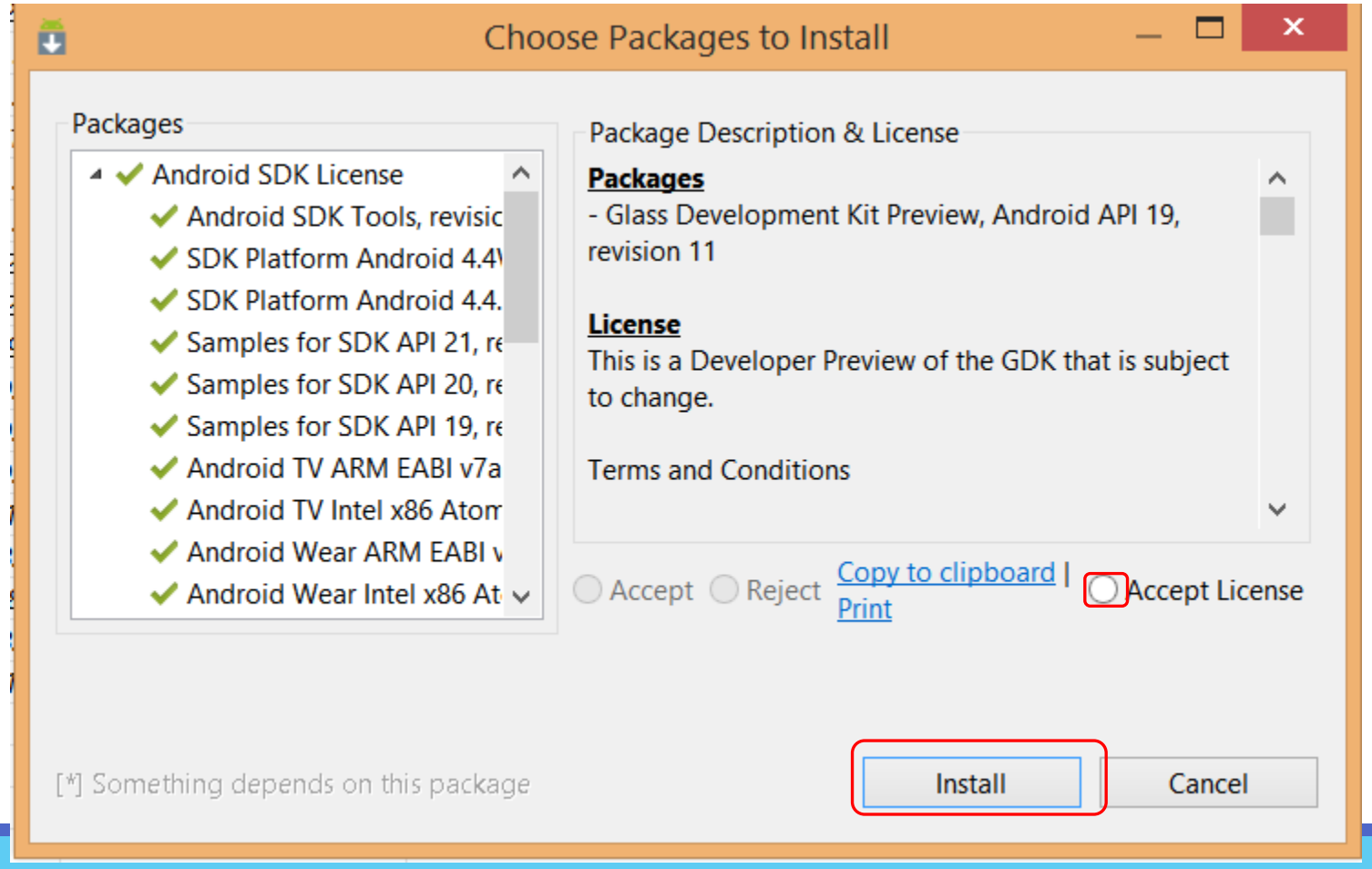
Android support library

Google USB Driver

Intel x86 Emulator Accelerator

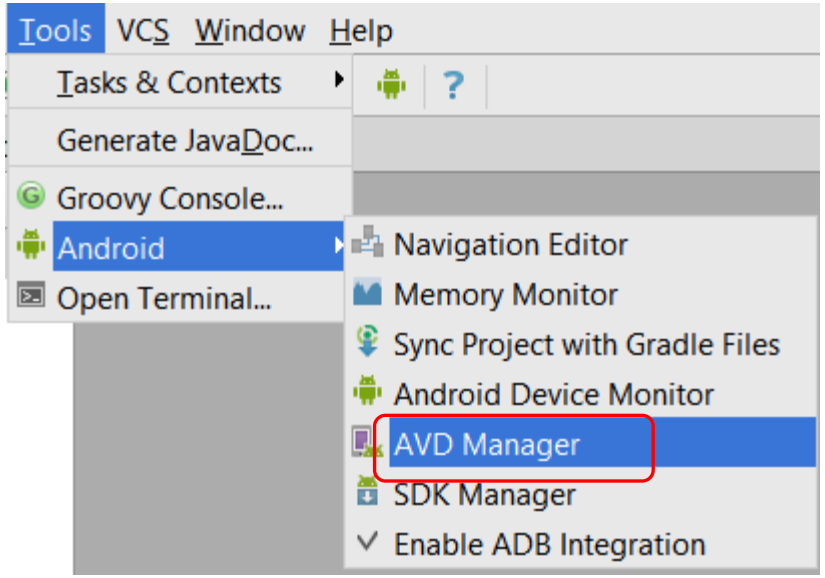


Download appropriate SDK packages

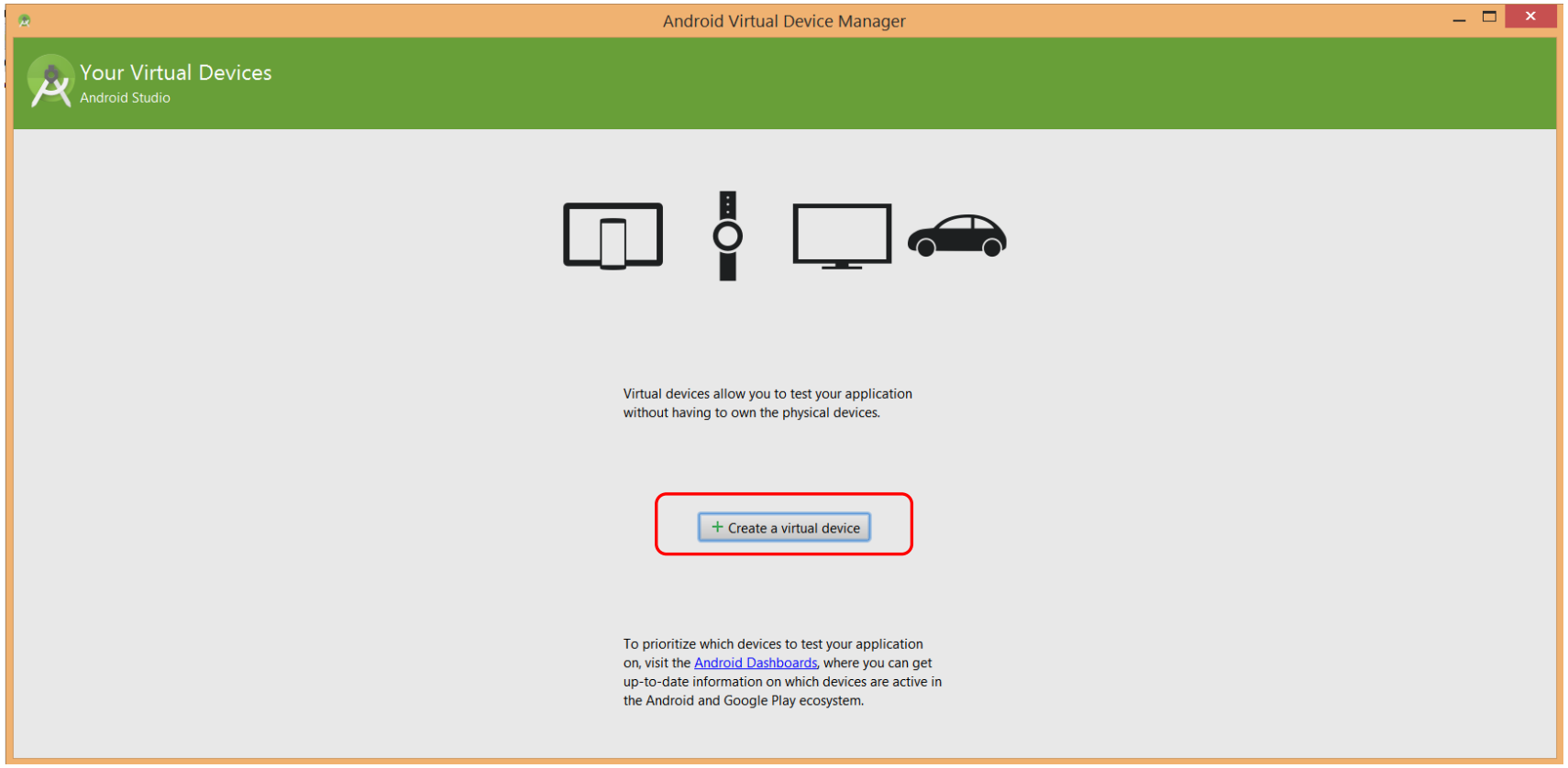


Create Android Virtual Device

Leave SDK manager opened, and then go to AVD manager.
It usually takes a long time to download.

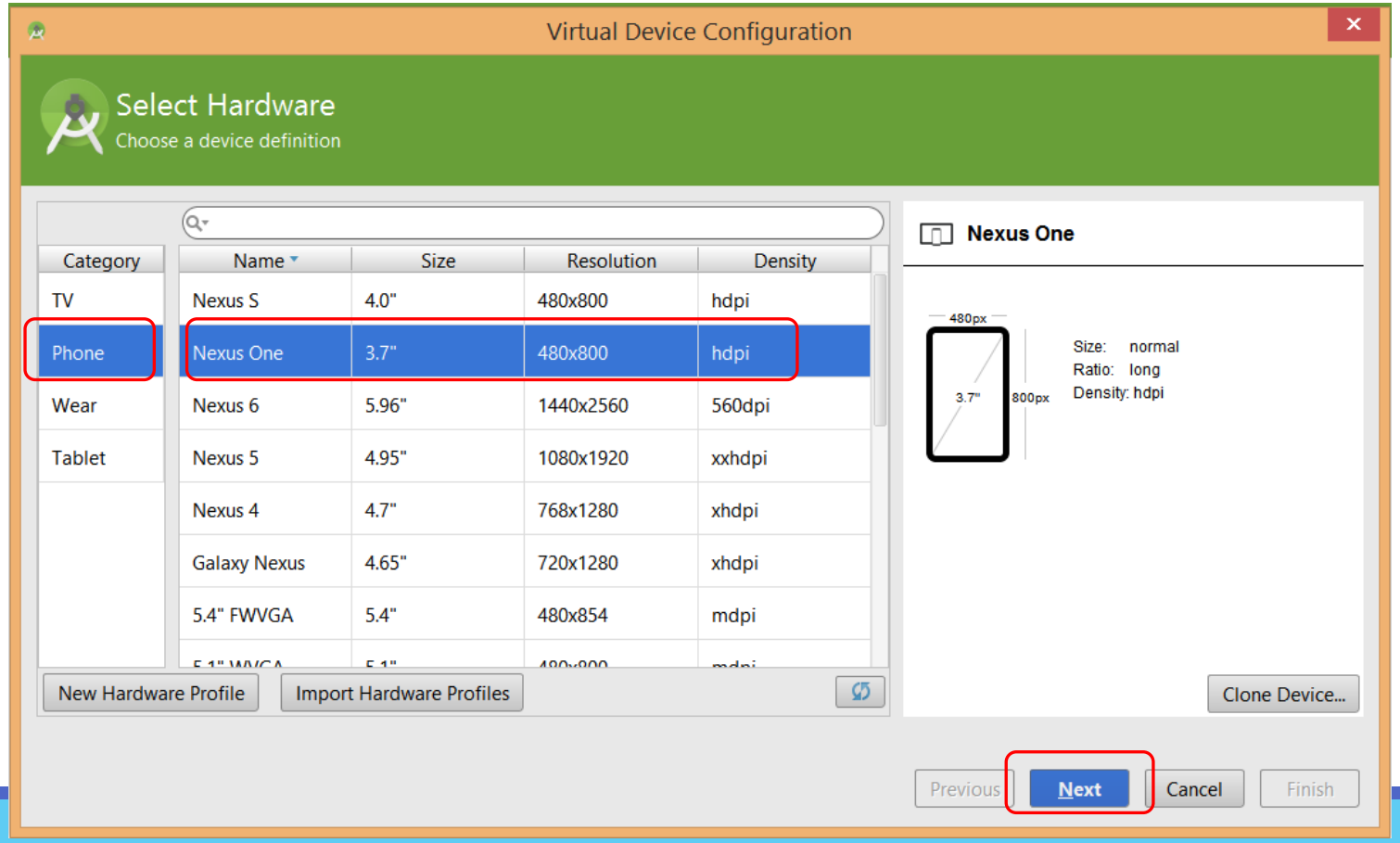


Create Android Virtual Device



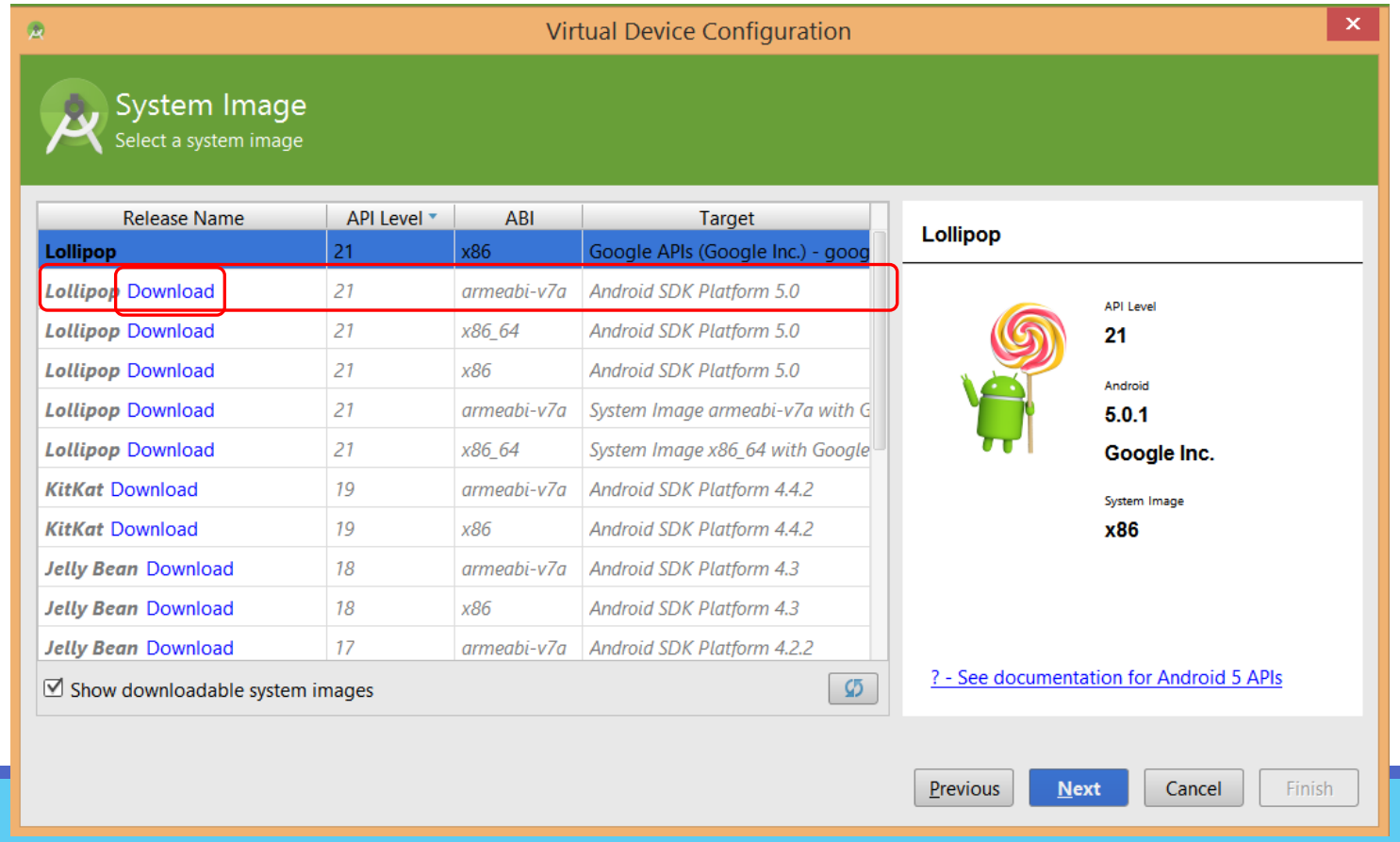
Create Android Virtual Device

Feel free to select different profile



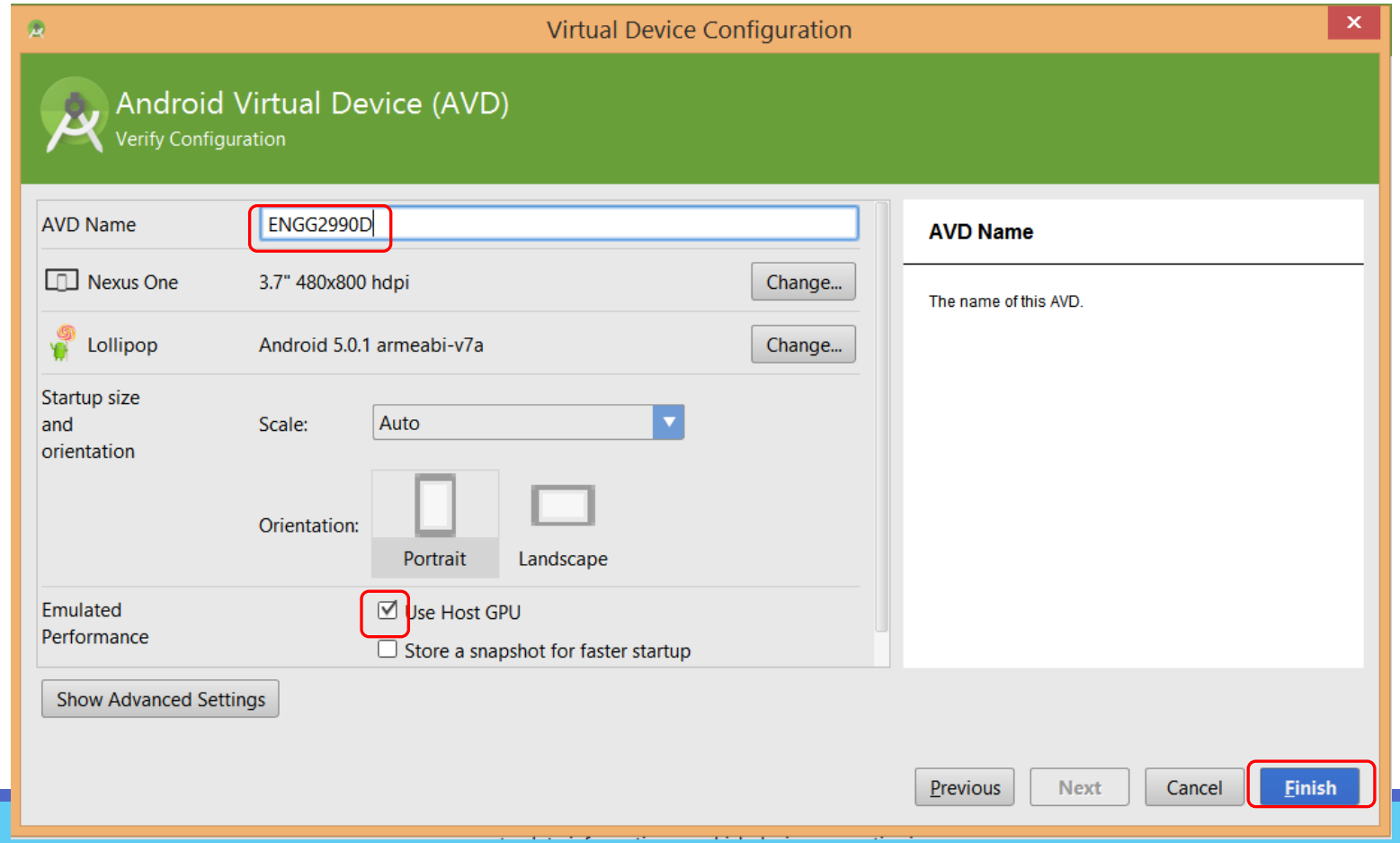
Create Android Virtual Device

Select API 21, armeabi-v7a. DO NOT use x86 as ABI. If you wish to do so, you would have to download corresponding drivers. After that, click next.



Create Android Virtual Device

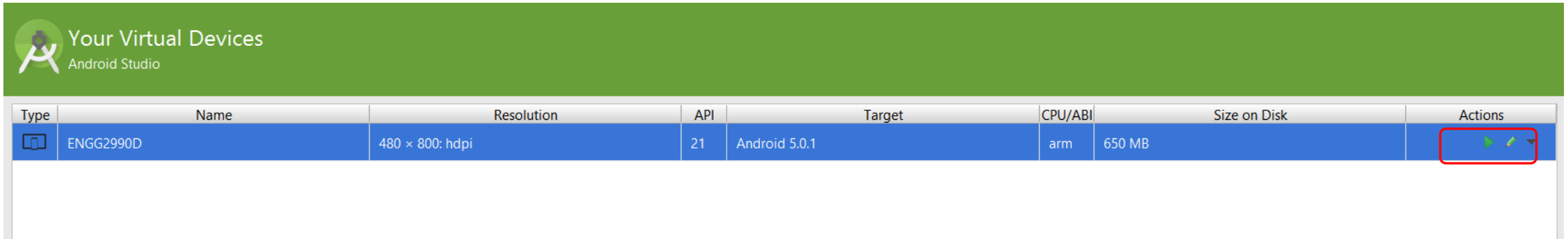
Set AVD name, select “use host GPU” if you have proper GPU driver. Then, click finish.



Create Android Virtual Device

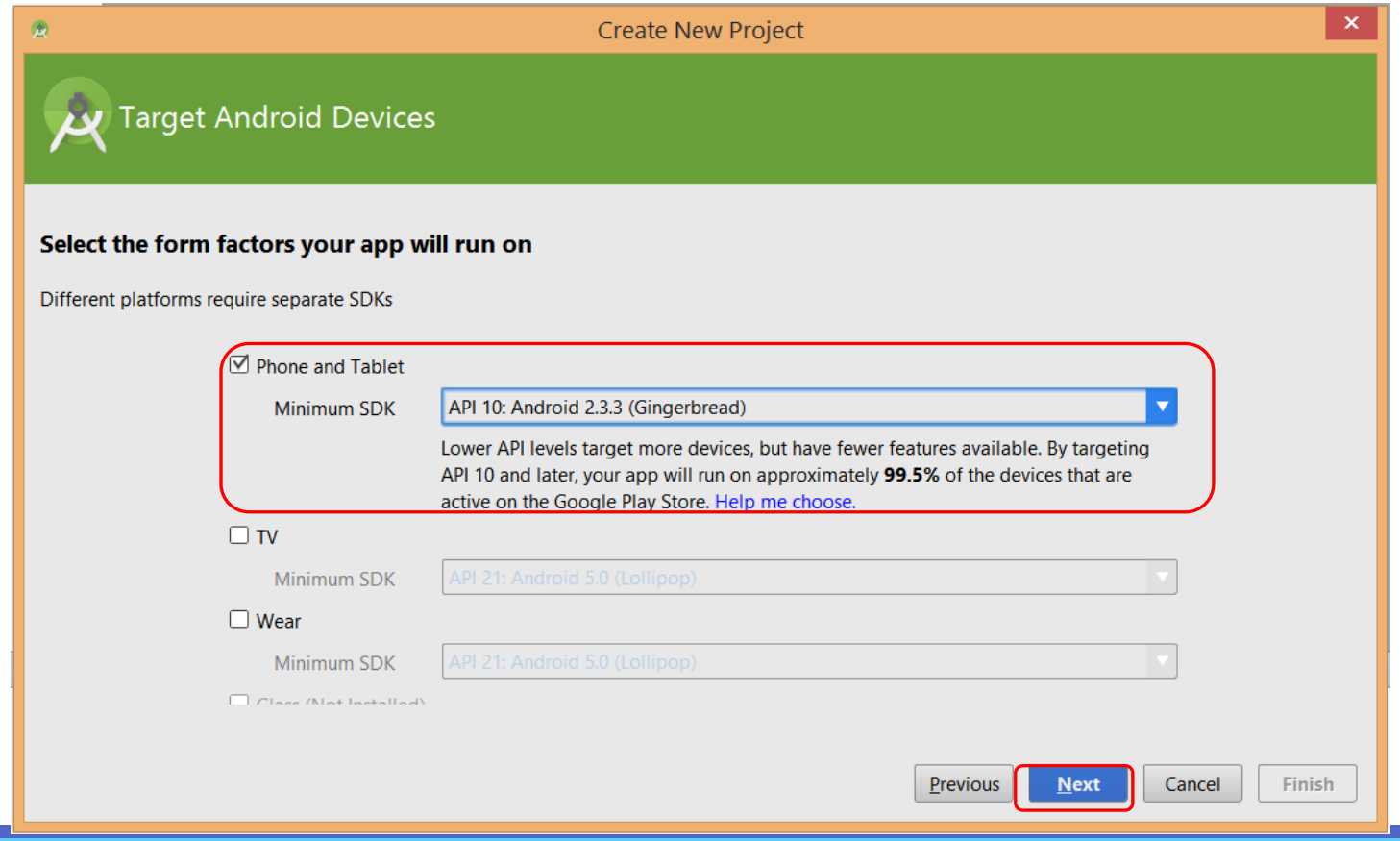
You have successfully created your AVD.

Click play, and you will be able to open up your virtual device.



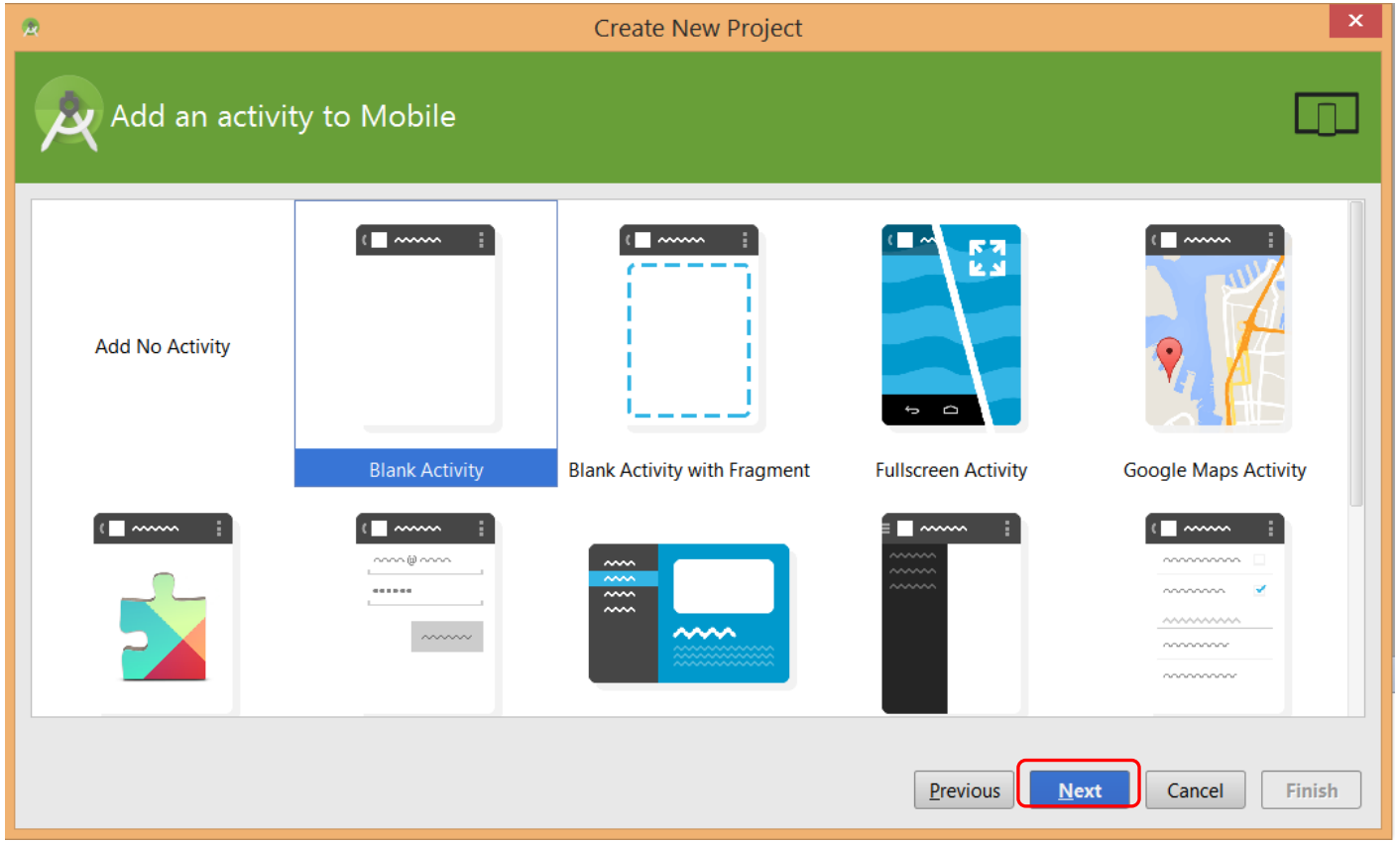
Creating an Android project

Click phone and Tablet, set API 10 as a minimum SDK version. Then, click next.



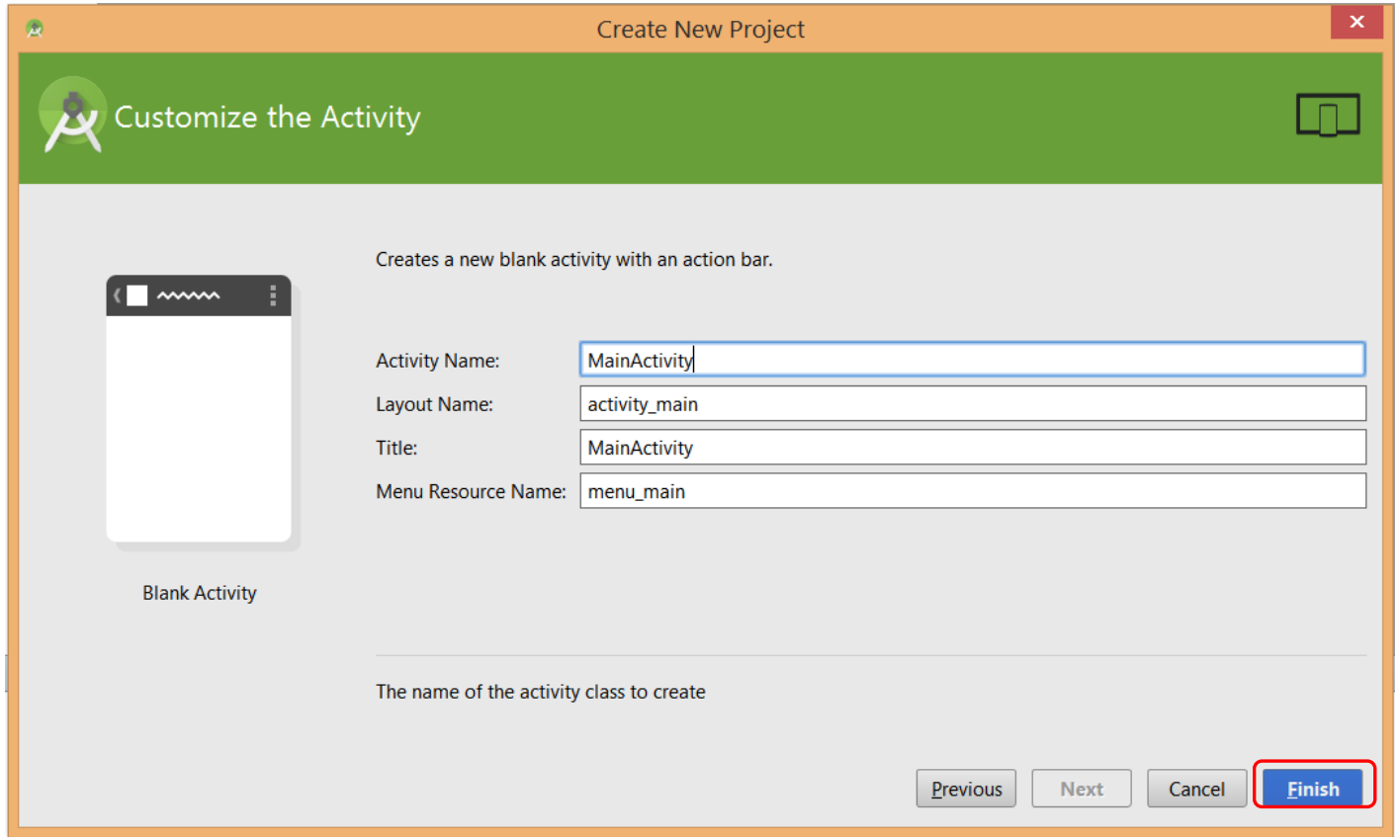
Creating an Android project

Select Blank Activity and click next.

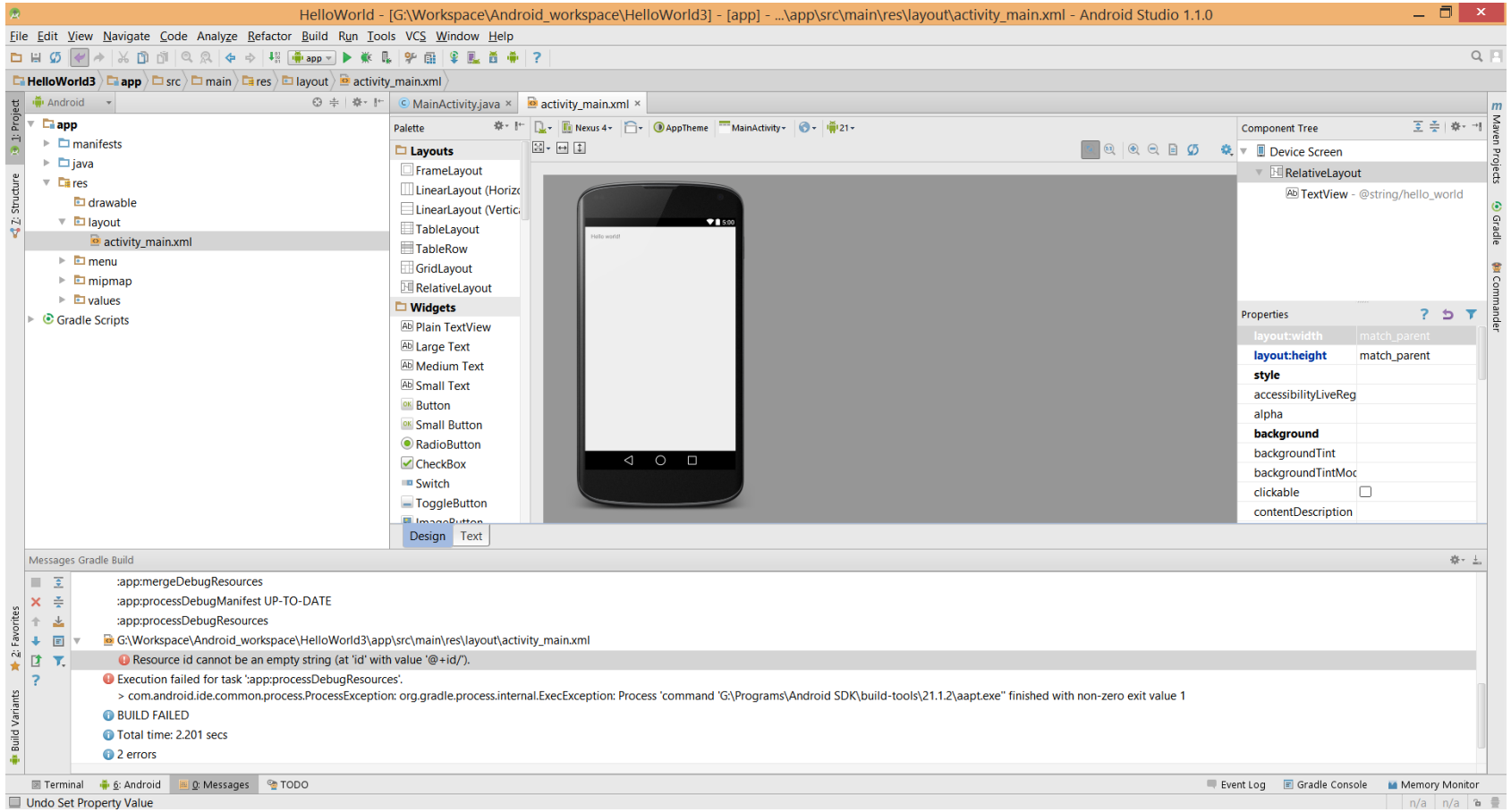


Creating an Android project

Click finish

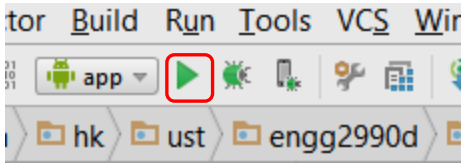


What you should see..



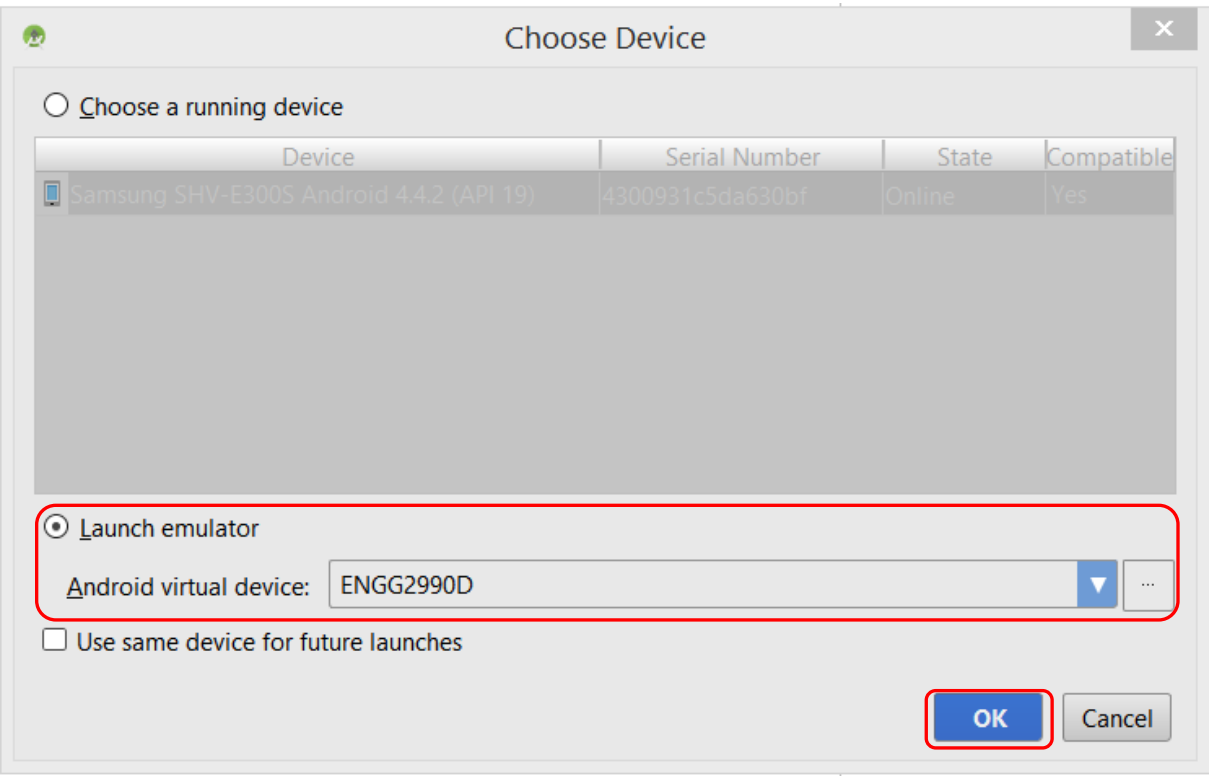
Run your application on AVD

Find the run icon on top, and click the button.



Run your application on AVD

Choose launch emulator, and select appropriate AVD. Click OK.



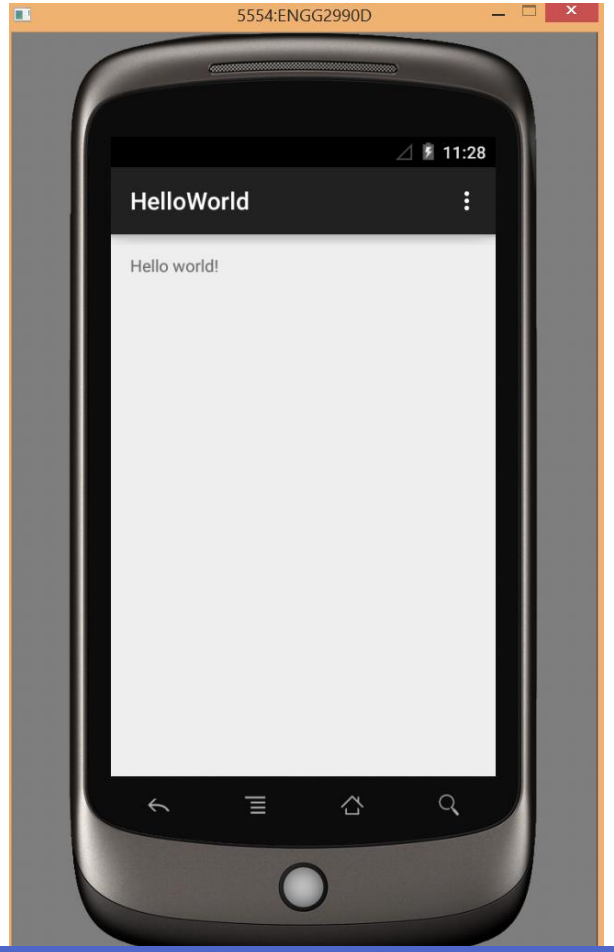
Run your application on AVD

Be patient, as it usually takes a long time.

Unlock your phone, then you will see your first app running.

If you reach up till this part,

you can get started with development phase!



Run application on your device

It depends on which phone you are trying to run an application with. You should stick to AVD for the time being, if you cannot work it out quite yet.

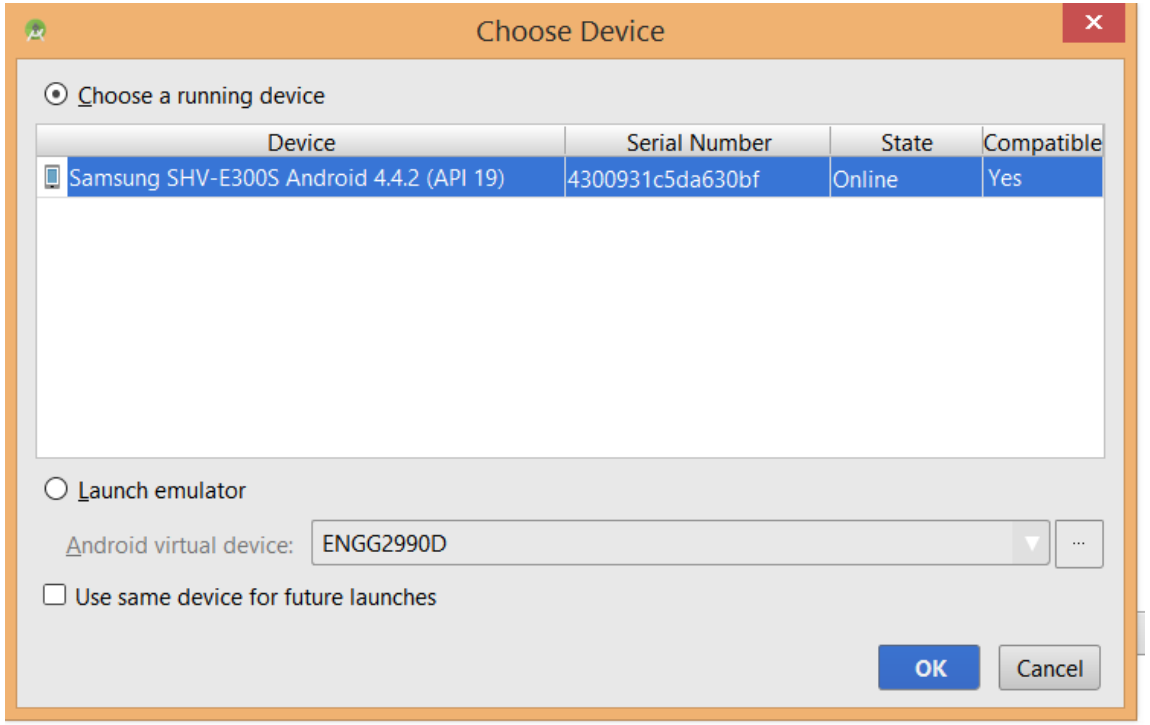
1. Make sure you enable USB debugging mode. *(Ex. If you are using Samsung Galaxy phone, go to settings. Navigate towards "about device" tab, and click Build number tab multiple times. Then, Developer option tab of which you can enable the debugging mode will show up.)*

2. Some phones might require phone specific USB driver.

3. Simply connect your device to the computer via USB cable.

Run application on your device

If you set up correctly you will be able to choose a running device. Click OK, and you will see your application running on your device!

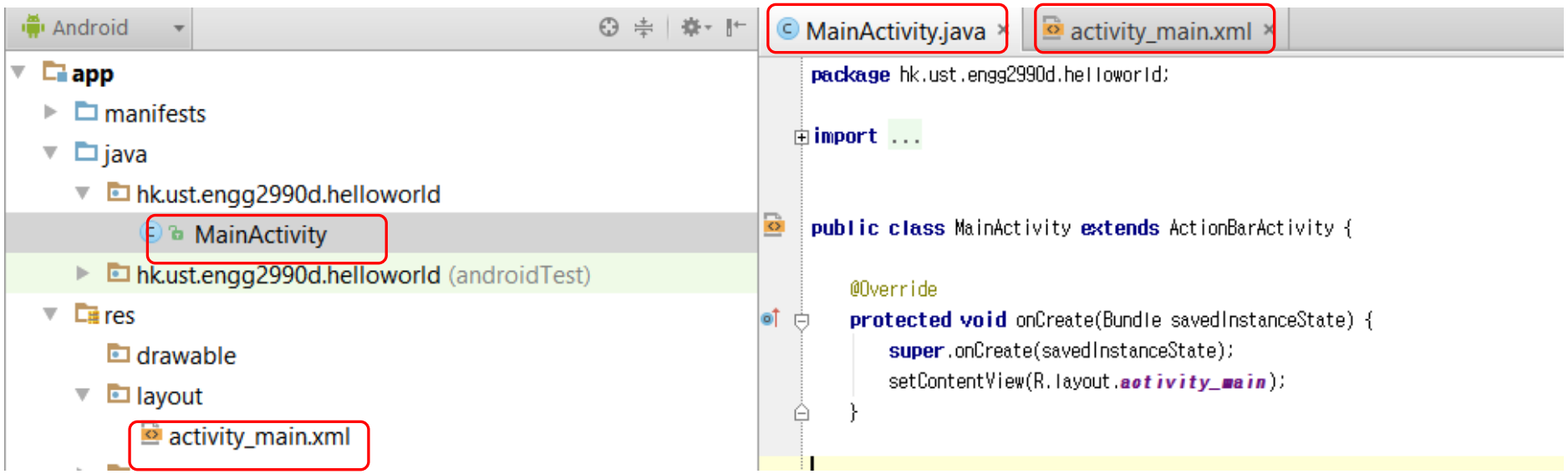


Switch between XML file and Java file

XML : *Markup Language* where you can design your **appearance** of the application.

(You should be familiar with XML if you have an experience with HTML)

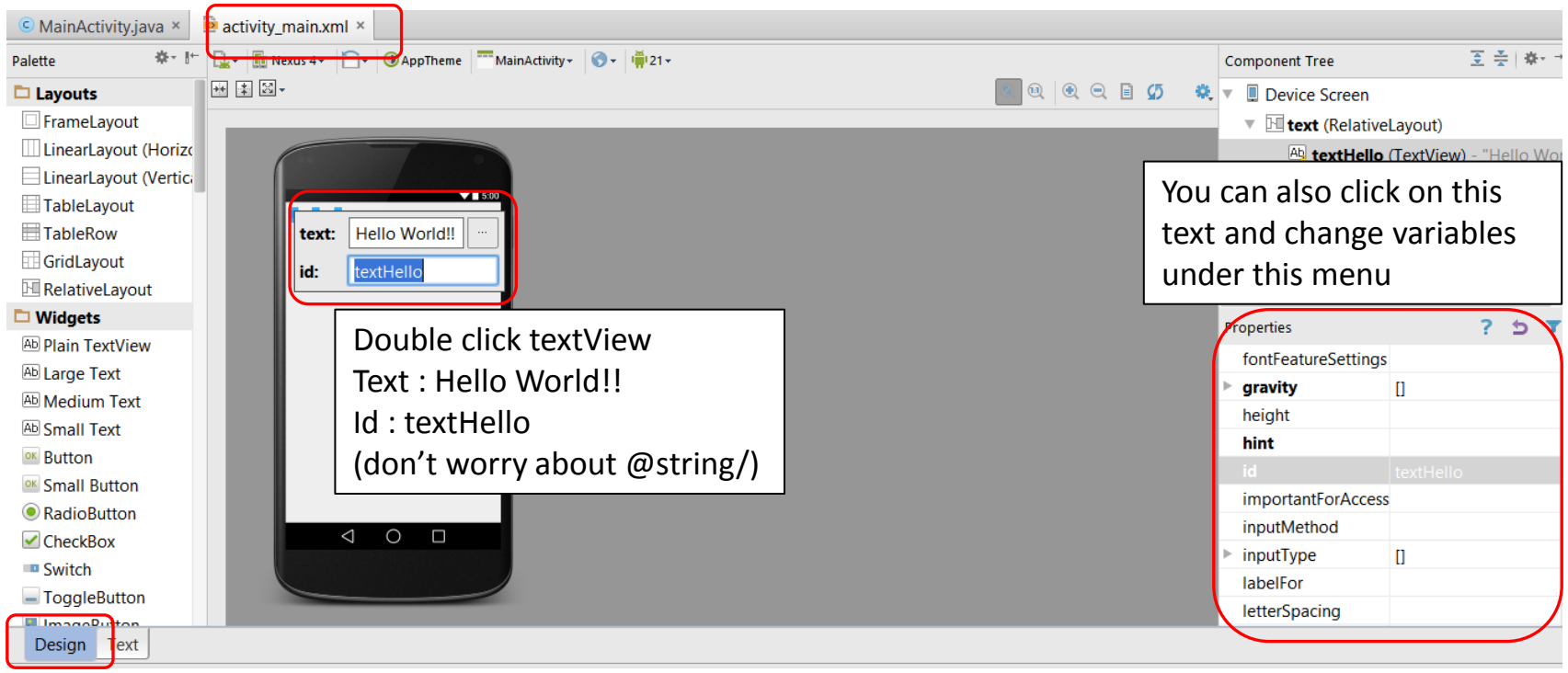
Java : *Programming language* where you are allowed to command your application to **act**.



Modify XML code – Assign ID

You must assign unique ID to every single objects on your screen for Java file to access it.

Double click the Hello World text on the screen, change text and id as shown below.



Modify XML code – Make it look better

Set layout:width => match_parent.

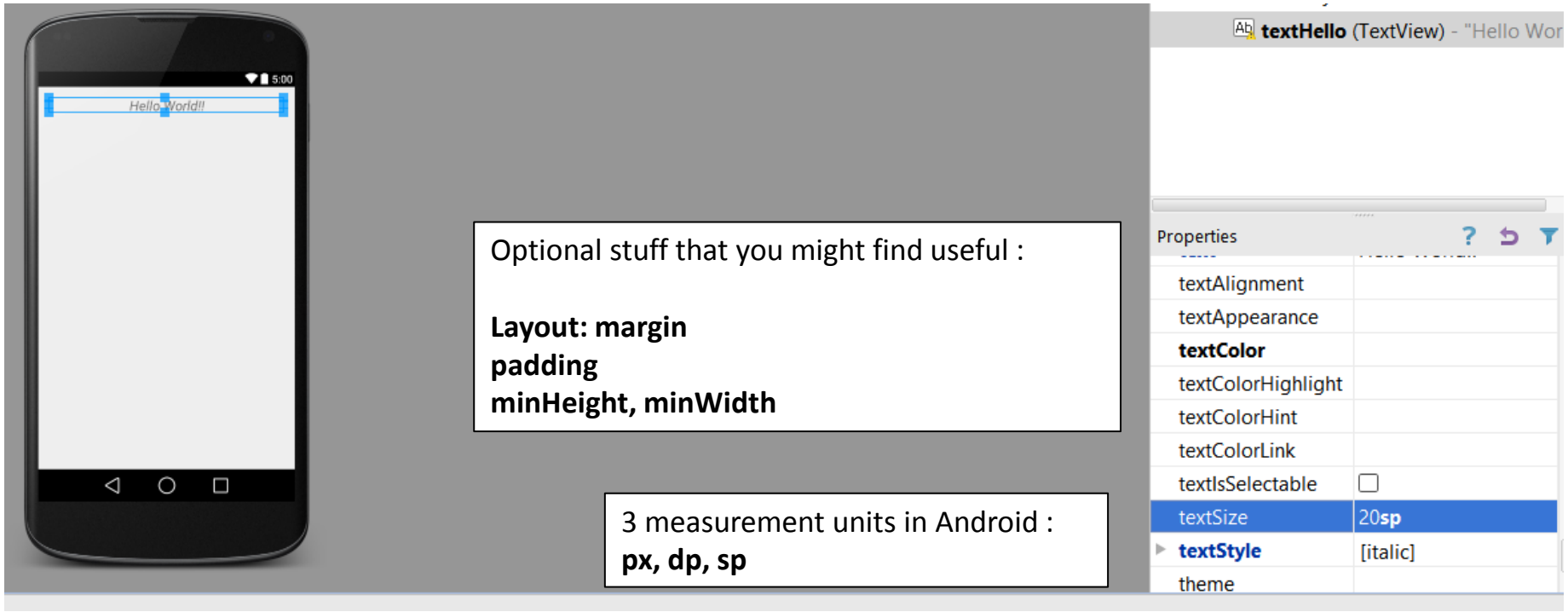
Modify XML code – Make it look better

Set gravity = “center_horizontal”

Modify XML code – Make it look better

Set textSize = “20sp”

There are much more rooms left to explore!! Try running the application again.



Optional stuff that you might find useful :

Layout: margin
padding
minHeight, minWidth

3 measurement units in Android :
px, dp, sp

Java file - the first intimidating look

MainActivity.java × activity_main.xml ×

```java
package hk.ust.engg2990d.helloworld;

import ...


public class MainActivity extends ActionBarActivity {

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


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

It is the time to have a look at Java file. This is what you would get as a default.

For those who barely have any programming experience, Do not let this put you off before you even try.

We will try to start explaining them in a vague way to let you understand the program structure!!

MainActivity.java × activity_main.xml ×

```java
package hk.ust.engg2990d.helloworld;

import ...

public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Pay attention to bolded parts only.

We are not going to care about any blocks covered in red!

- The java code is divided and covered by “blocks”.
- Each blocks are covered by curly brackets **{ }**.

Usually, each blocks are always given a name.

Public class **MainActivity** extends ActionBarActivity **{**
(…program commands goes here)
}

Protected void **onCreate(**Bundle savedInstanceState**){**
(…program commands goes here)
}

Try to see that onCreate() is included in MainActivity block.

None of them are really true, but it should give you a general idea about program structure.

Java file – basic program structure

We are bringing our scope down to just two blocks {} : MainActivity and onCreate()

```
public class MainActivity extends ActionBarActivity
{

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}
}
```

- onCreate() function is called right after application is launched, and executes all commands inside its block.

- setContentView(…); command links all contents in our XML file to actual android application.

- This is how we get to see “hello world” text after launching android application.

onCreate() function is placed inside the MainActivity block, covered by curly brackets {}.

Two program commands are placed inside of the onCreate() function. Note that all commands ends with semi-colon **;**

Let’s modify Java file – write commands

Our aim today is to write some commands that allow us to dynamically change properties of “Hello World” text in XML file. Now we can go ahead and add some program commands ourselves.

For now, we only know how onCreate() works. We are going to write commands under onCreate() function. then these commands will be executed right after application is launched.

Before we do that though, one setup is required before we can access the text in XML file.

Let’s modify Java file – create a variable

Recall that we assigned ID for Hello World text by modifying XML file.

Android let us to create a variable that refers to all information about any object in XML file, using Java.

It can be done by adding two lines of code inside the MainActivity class:

Variable type : TextView

Variable name: TextHello

```
TextView textHello;        // declare a variable named textHello, with variable type of textView
textHello = (TextView) findViewById(R.id.textHello);
```

findViewById(…) function performs appropriate operation and store its values into textHello variable.

Let’s modify Java file – create a variable

```
package hk.ust.engg2990d.helloworld;

import ...

public class MainActivity extends ActionBarActivity
{

    TextView textHello;
    textHello = (TextView) findViewById(R.id.textHello);

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}
}
```

This is how the code should now look like.

Write the code inside the MainActivity block, but outside of any other blocks!

you can actually set variable name to anything as long as the name is unique. Still, to prevent confusion, set variable name similar to the ID of object in XML.

Rectifying errors

```
TextView textHello;
Cannot resolve symbol 'TextView'
```

We can solve the problem by importing appropriate class.

```
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends ActionBarActivity
{
    TextView textHello;
    Import Class
    Create Class 'TextView'
    Create Enum 'TextView'
    Create Inner Class 'TextView'
    Create Interface 'TextView'
    Make 'private'
    Make 'protected'
    Make 'public'
```

Place the cursor on TextView, highlighted in red.

Press Alt + Enter, and then click the first one that appears.

Then The program will automatically fix error for you.

It does not work for Every single type of errors!

Rectifying errors

```
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity
{

    TextView textHello;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textHello = (TextView) findViewById(R.id.textHello);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}
}
```

You will see that Android Studio have imported appropriate class for you.

You can also type the import statement by yourself to fix the error.

Write a function for a reference variable

We have successfully created a reference variable called “textHello”.

Now it is time to write a function for it, a function that changes the property of textHello.

We can use functions that are provided by TextView class.

Add two lines of code inside of onCreate()

```
textHello.setText(“GoodBye World!!!”);
textHello.setSize (40);
```

Set the size	of textHello	With the input parameter integer value of 40
Set the text	of textHello	With the input parameter String value of “GoodBye World!!!”

Our code so far…

```
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity
{

    TextView textHello;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textHello = (TextView) findViewById(R.id.textHello);

        textHello.setText("GoodBye World!!!");
        textHello.setTextSize(40);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}
}
```

1 After application is launched, 2 onCreate() is called.

Then,

3 Resources and contents in XML file is linked to the application

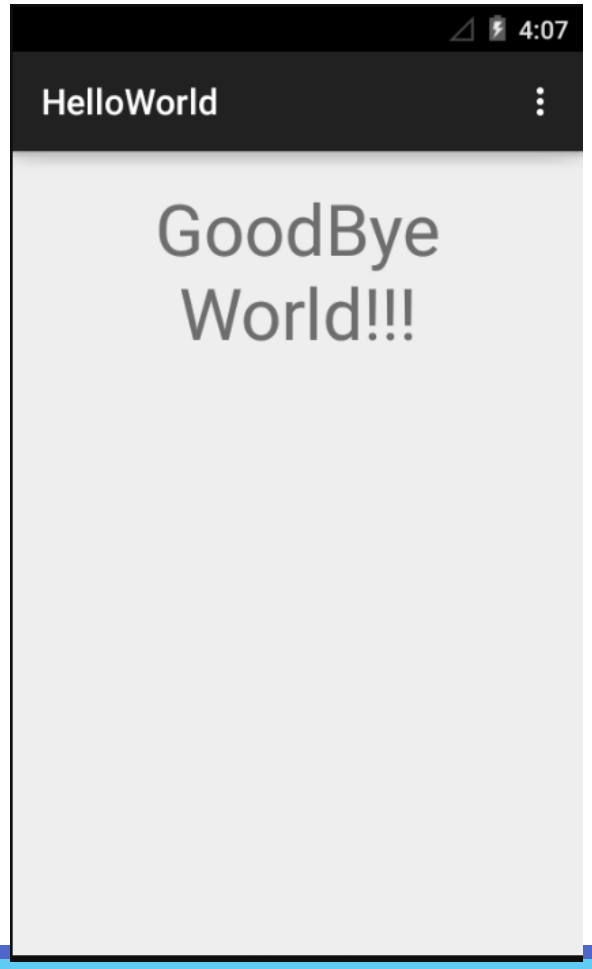
4 textHello variable is assigned with all the properties of *textHello* in XML

5 textHello changes its text to “GoodBye World!!!”

6 textHello changes its text size to integer value 40.

Keep track of brackets and location of each code!

Test the application



We need interactive components

We can make our TextView variable do things : change some of its own properties.

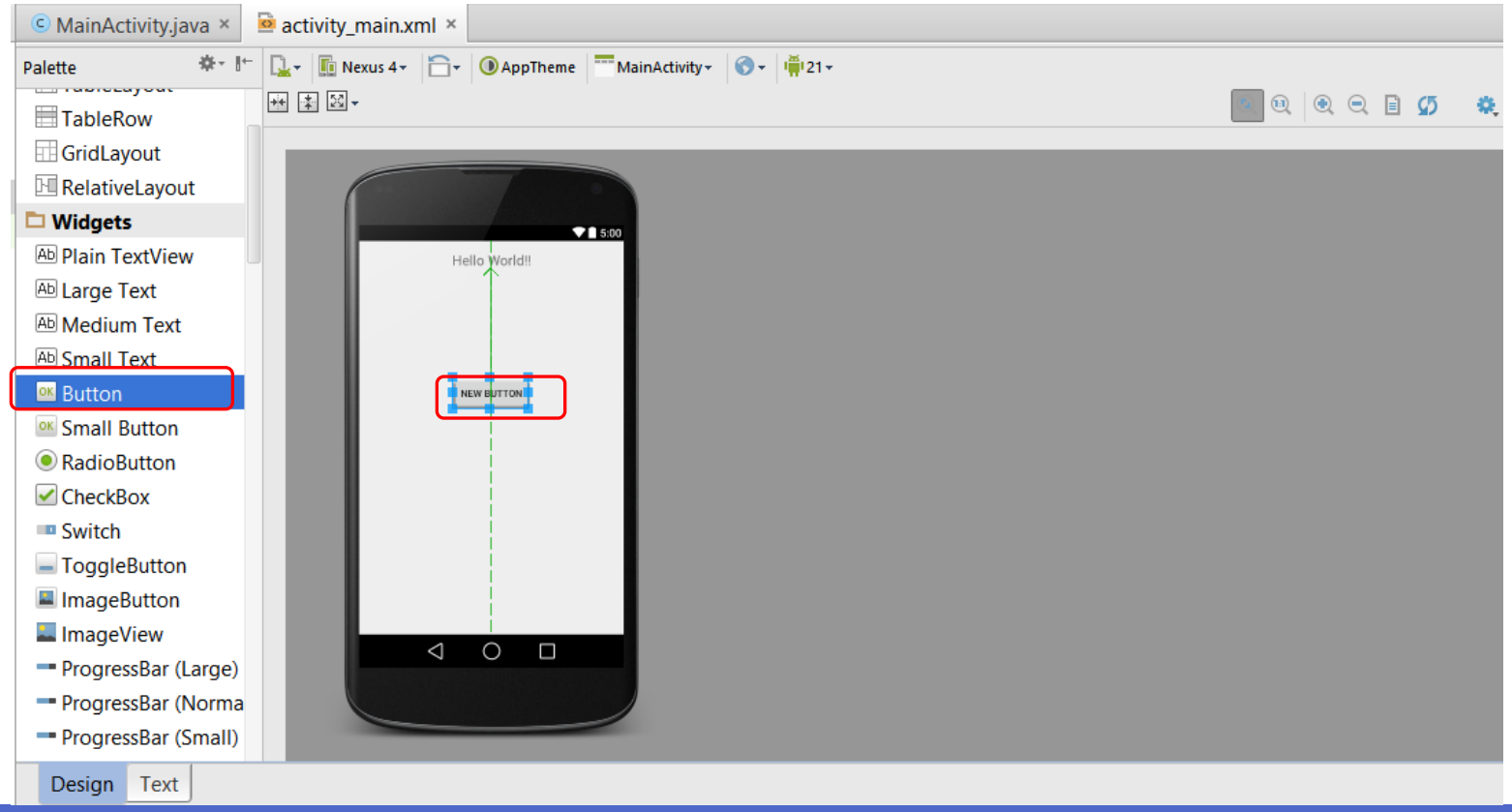
Although intriguing, this is not good enough. We want the application to interact with user. Our final aim is to build an airship controller application which can activate multiple motors by clicking buttons and sliders, via Bluetooth communication.

For today's lab, we are going to make a button that allow us to change a text on screen.

Let's go back to XML file and create a button.

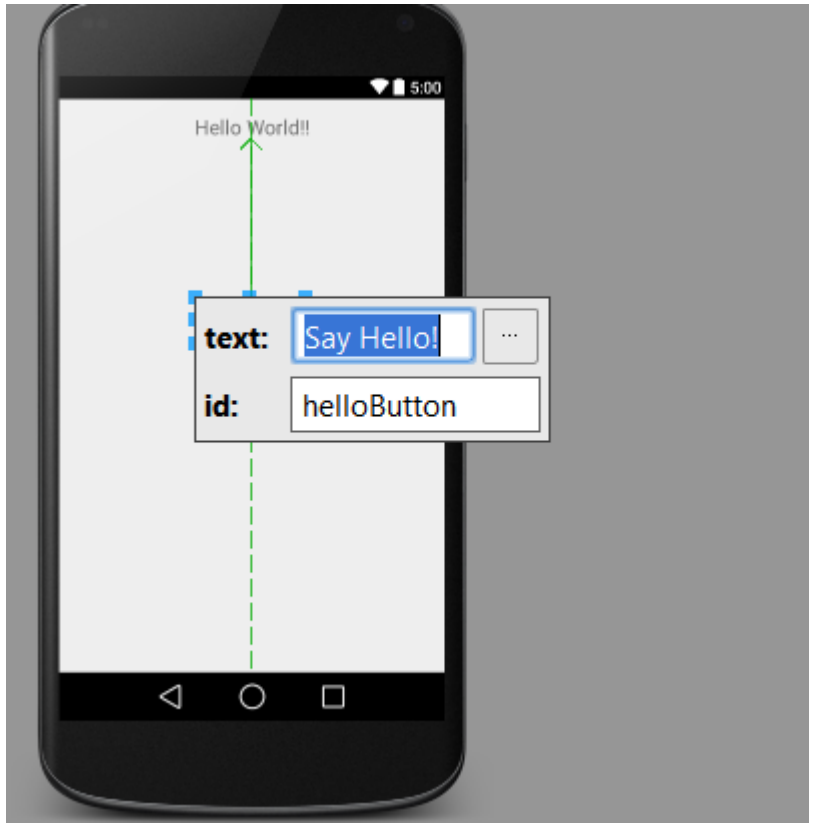
Create a button in XML file

Under the Widgets tab, drag one button to the screen.



Necessary procedure, set ID

Set ID = helloButton



Go back to Java file

```java
public class MainActivity extends ActionBarActivity
{

    TextView textHello;
    Button helloButton;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textHello = (TextView) findViewById(R.id.textHello);
        helloButton = (Button) findViewById(R.id.helloButton);


    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}
}
```

We are doing the same thing basically.

Create a reference variable.

After adding the code, put your cursor on Button, press Alt+Enter to import appropriate class

Feel free to erase setText(); and setTextSize();

OnClickListener()

We have a reference variable to a button, but we do not know how to handle a button click event.

We are going to define OnClickListener() function to do that.

Just like onCreate() function which is called whenever the application is launched,

OnClickListener() function will be called by Android whenever any button is clicked.

Add the code into Java file

```java
public class MainActivity extends ActionBarActivity implements OnClickListener
{

    TextView textHello;
    Button helloButton;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textHello = (TextView) findViewById(R.id.textHello);
        helloButton = (Button) findViewById(R.id.helloButton);


    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}
}
```

Highlighted in red. You know what to do.

Press Alt + Enter to rectify error.

More error to fix

```java
package hk.ust.engg2990d.helloworld;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;


public class MainActivity extends ActionBarActivity implements View.OnClickListener
{

    TextView textHello;
    Button helloButton;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {...}

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}

    @Override
    public void onClick(View v) {

    }
}
```

Opps, we still have errors. Alt+Enter again.

If things do not work out, you can type the code by yourself.

Pay attention to highlighted parts.

Review on the program structure

```java
package hk.ust.engg2990d.helloworld;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity implements View.OnClickListener
{

    TextView textHello;
    Button helloButton;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textHello = (TextView) findViewById(R.id.textHello);
        helloButton = (Button) findViewById(R.id.helloButton);

    }

    @Override
    public void onClick(View v) {

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}

}
```

We have four blocks, all included in one MainActivity block.

Out of four blocks in MainActivity, we are only paying attention to onCreate() and onClick()

setOnClickListener()

```
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    textHello = (TextView) findViewById(R.id.textHello);
    helloButton = (Button) findViewById(R.id.helloButton);
    helloButton.setOnClickListener(this);

}
```

Right after assigning values to helloButton variable,

Set onClickListener to helloButton.

By adding this code, helloButton will start responding to button click action.

Write code for Onclick(View v)

We are going to make the application such that pressing button will change the text.

```
@Override
public void onClick(View v) {
    textHello.setText("GoodBye World!!!");
}
```

Add the code textHello.setText(“GoodBye World!!!”); under OnClick() function.

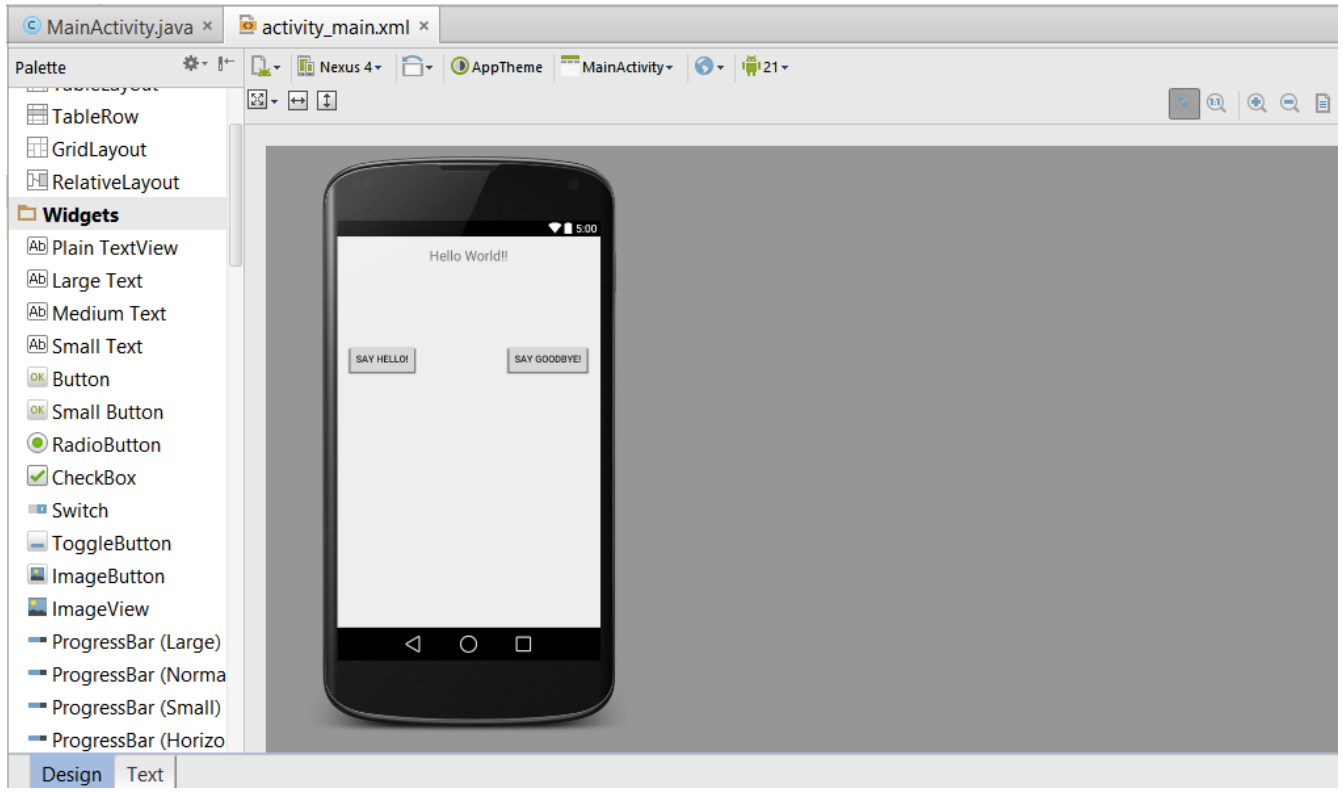
Any button pressing action will call onClick() function.

Then, program codes inside onClick() block will be executed.

Run the application and see what happens.

Add one more button

Now we are going to add one more button that makes application to say hello again.



Create reference for second button

Create reference variable, assign values to it, and set onClickListener. Then, run the application.

```
TextView textHello;
Button helloButton;
Button goodByeButton;

@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    textHello = (TextView) findViewById(R.id.textHello);
    helloButton = (Button) findViewById(R.id.helloButton);
    helloButton.setOnClickListener(this);
    goodByeButton = (Button) findViewById(R.id.goodByeButton);
    goodByeButton.setOnClickListener(this);

}

@Override
public void onClick(View v) {
    textHello.setText("GoodBye World!!!");
}
```

Decision making (if statements)

You would have noticed that Hello World text responds in a same way no matter which button you click.

To distinguish these buttons, we can make use of

- Unique ID assigned to each buttons, and
- **if statements**

```
if (true)
{
    // program enters the block
}
if (false)
{
    // program skips the block
}
```

Idea is simple. Program enters or skips the entire block depending on whether the statement inside the bracket is true of not.

Decision making (if statements)

```
@Override
public void onClick(View v) {

    if (v.getId() == R.id.helloButton)
    {
        textHello.setText("Hello World!!!");
    }

    if (v.getId() == R.id.goodByeButton)
    {
        textHello.setText("GoodBye World!!!");
    }

}
```

Modify onClick() function to use if statements as follows.

Decision making (if statements)

Variable type = View

Variable name = v

```
@Override
public void onClick(View v) {

    if (v.getId() == R.id.helloButton)
    {
        textHello.setText("Hello World!!!");
    }

    if (v.getId() == R.id.goodByeButton)
    {
        textHello.setText("GoodBye World!!!");
    }

}
```

Logical operation

When onClick() is called, input parameter is also delivered.

This variable, v is a reference variable of type View that contains information about the button that is clicked.

If helloButton is clicked, the program will only enter the first block, and ignore second one.

If goodByeButton is clicked, the program will only enter the second block.

Run the application again!

Challenge!!

Using the same method you used to create the “Say Hello!” and “Say Goodbye!” buttons, create three buttons which are labeled “Say Good Morning!”, “Say Good Afternoon!” and “Say Good Night!”. These buttons should serve the same function as the “Say Hello!” and “Say Goodbye!” buttons, where they change the text to an appropriate greeting.