+ Queens University Belfast

Android Studio

CSC3054 / CSC7054

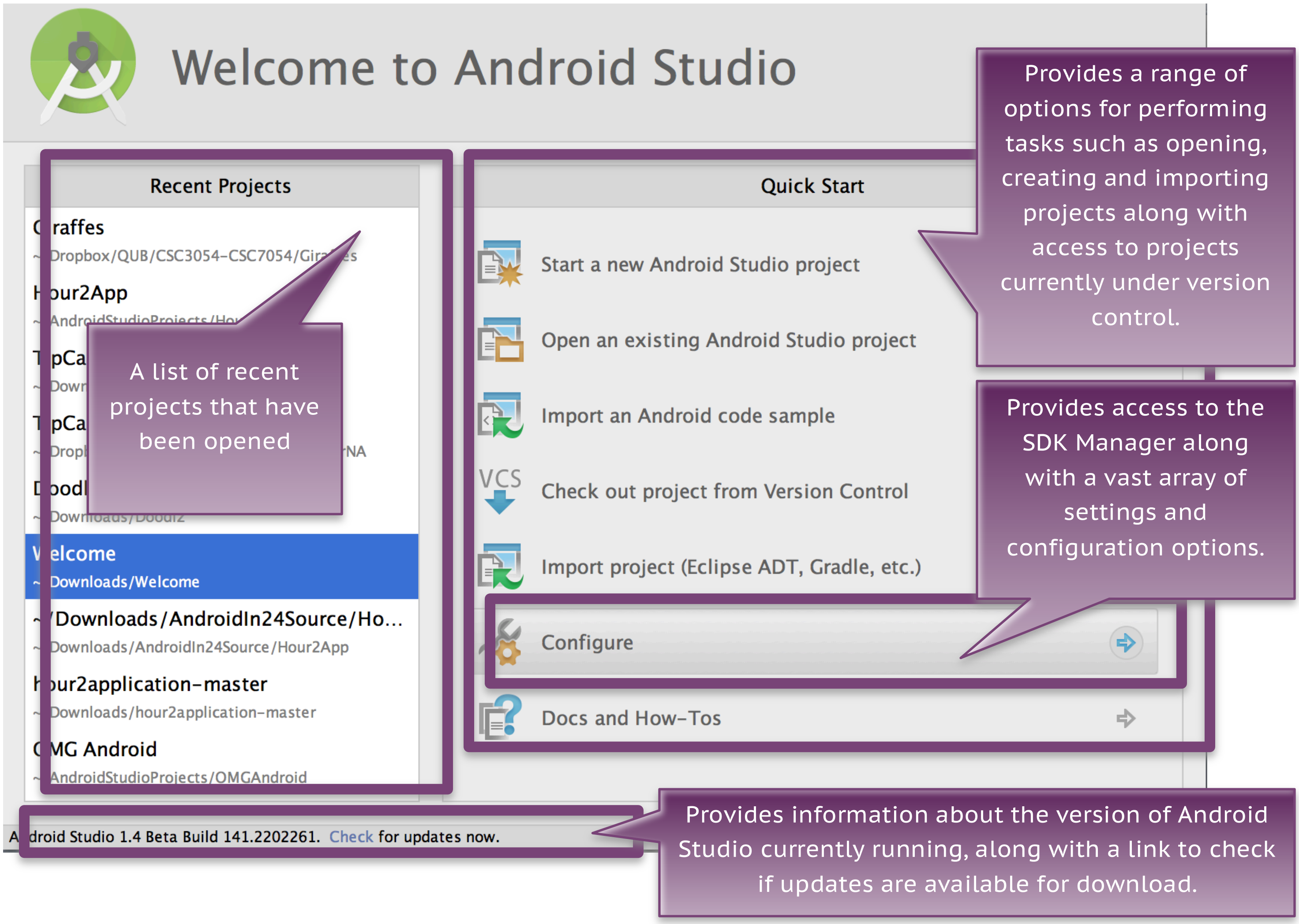
Create your first Android App

Creating your first Android App

By this stage you should have set up Android Studio on your computer. This tutorial will provide you with an initial overview of the various areas and components that make up the Android Studio environment. It will also show you how to create your first app allowing you to gain familiarity with the layout and organisation of the Android Studio user interface.

The Welcome Screen

This screen is displayed any time that **Android Studio** is running with no projects currently open (open projects can be closed at any time by selecting the **File** → **Close Project** menu option). If Android Studio was previously exited while a project was still open, next time it is launched, it will by-pass the welcome screen, automatically opening the previously active project.


Welcome to Android Studio
Recent Projects
Giraffes
~/Dropbox/QUB/CSC3054-CSC7054/Giraffes
Hour2App
~/AndroidStudioProjects/Ho
TipCa
~/Down
TipCa
~/Dropb
NA
Doodl
~/Downloads/Doodlz
Welcome
~/Downloads/Welcome
~/Downloads/AndroidIn24Source/Ho…
~/Downloads/AndroidIn24Source/Hour2App
hour2application-master
~/Downloads/hour2application-master
OMG Android
~/AndroidStudioProjects/OMGAndroid
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
VCS
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos

Android Studio 1.4 Beta Build 141.2202261. Check for updates now.

A list of recent projects that have been opened

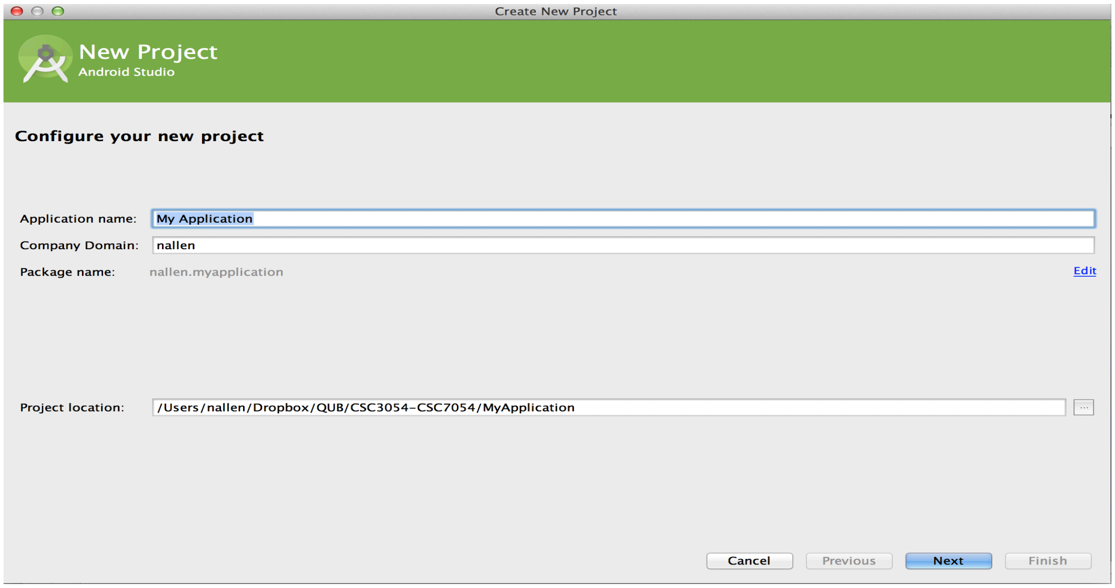
Provides a range of options for performing tasks such as opening, creating and importing projects along with access to projects currently under version control.

Provides access to the SDK Manager along with a vast array of settings and configuration options.

Provides information about the version of Android Studio currently running, along with a link to check if updates are available for download.

Step1 Create a new project

From the **Quick Start** menu select **Start a new Android Studio Project** to display the first screen of the **New Project** wizard. **Configure** the project as outlined in step 2. Press **next**.



Create New Project

New Project

Android Studio

Configure your new project

Application name: My Application

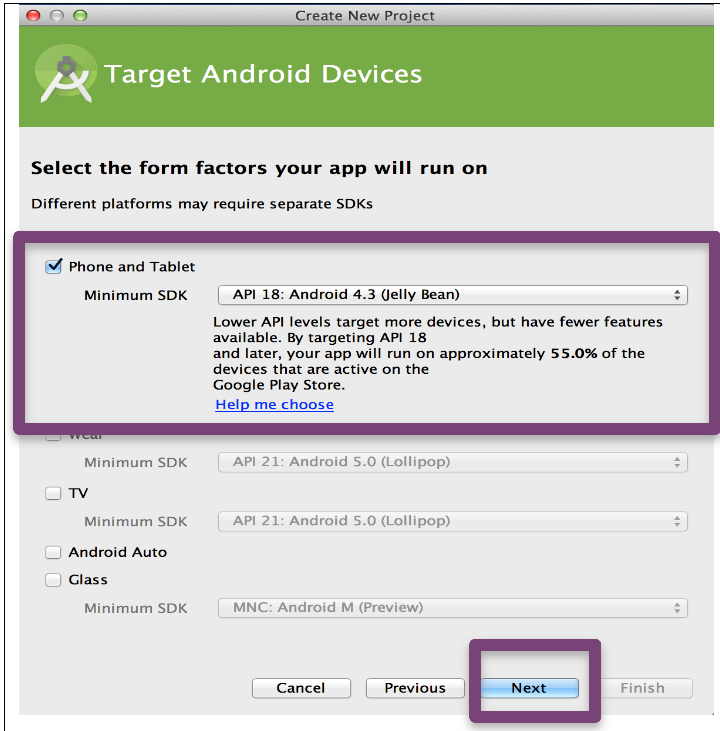
Company Domain: nallen

Package name: nallen.myapplication

Project location: /Users/nallen/Dropbox/QUB/CSC3054-CSC7054/MyApplication

Step 2	Configure your project	
Application Name	MyFirstApp	This is the name by which the application will be referenced and identified with Android Studio. It is also the name that will be used when the completed application goes on sale in Google Play store.
Company Domain	com.mycompany .MyFirstApp	The Package Name is used to uniquely identify the application within the Android application ecosystem. The IDE will use that to compute the package name for the application. It should be the reverse URL of your domain name followed by the name of the application. The IDE reverses the company domain and then tacks on the application name. This is all in lower case. The means that the package names created this way are less likely to conflict with the package names of other applications that might end up on the same device
Project Location	Choose your default home directory	This will be the default location of where you want the project to be stored and may be changed by clicking on the button to the right of the text field containing the current path setting

Step 3 Form Factors



Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☑ Phone and Tablet

Minimum SDK: API 18: Android 4.3 (Jelly Bean)

Lower API levels target more devices, but have fewer features available. By targeting API 18 and later, your app will run on approximately **55.0%** of the devices that are active on the Google Play Store.

Help me choose

□ Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

□ TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

□ Android Auto

□ Glass

Minimum SDK: MNC: Android M (Preview)

This screen asks you what kind of devices you want this application to run on.

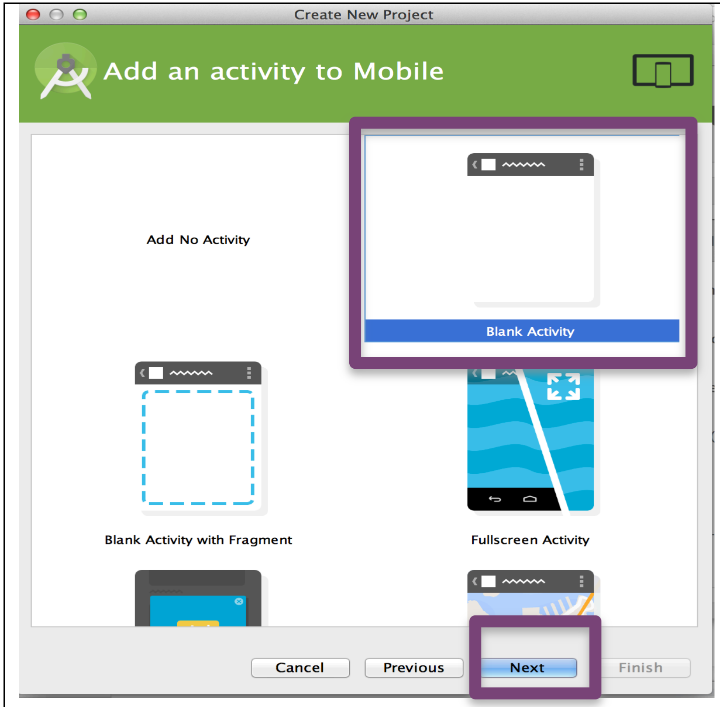
Make sure `Phone and Tablet` is ticked.

Choose an `API level e.g. 18`. This will correspond to Android 4.3 as the oldest platform this app will run on.

At this time approximately 55% of the devices on the Google Play store are running API, level 21 or newer. Therefore, in practice you might want to choose a lower API level.

Click on the `Next` button

Step 4 Add and Activity

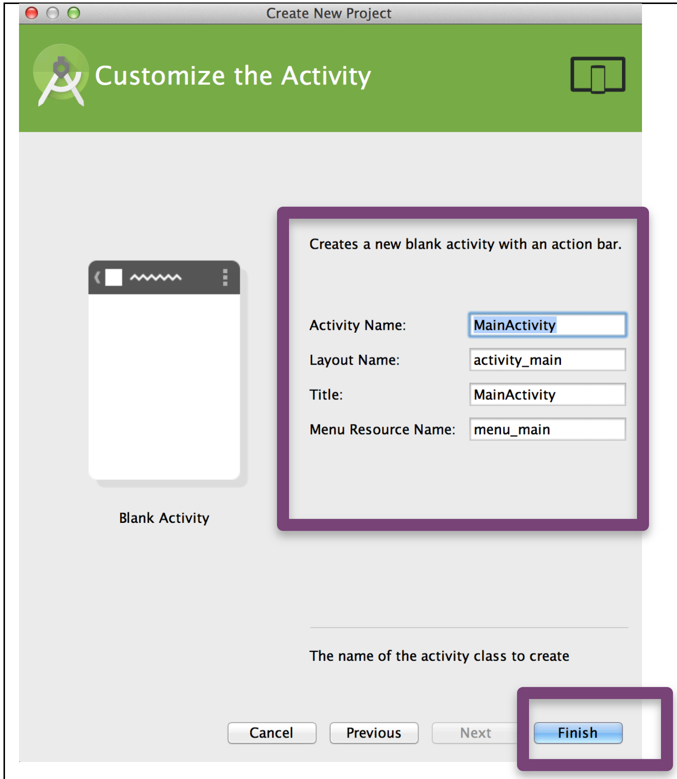


This dialog is asking what kind of UI template files you would like the IDE to pre-build for you.

At this stage, keep it simple, and choose a single `Blank Activity`.

Click on the `Next` button

Step 5 Customize the Activity



Creates a new blank activity with an action bar.

Activity Name: MainActivity

Layout Name: activity_main

Title: MainActivity

Menu Resource Name: menu_main

The name of the activity class to create

This dialogue asks you how you want to name some of the application's source code file.

Accept the `defaults`

Click `Finish`.

The IDE will generate the application's source code and layout files.

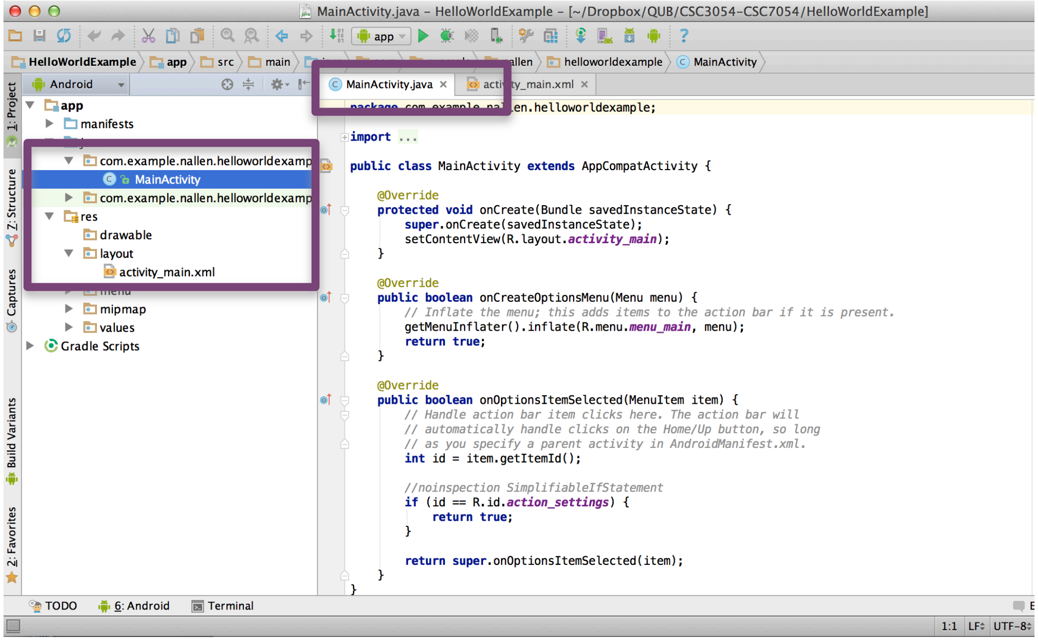
Once this is completed, the IDE will display a file called `activity_main.xml`. This may take some time.

This file will define the application's user interface layout.

Based on the contents of this file, the IDE will show whatever it can about how this application will look when it runs.

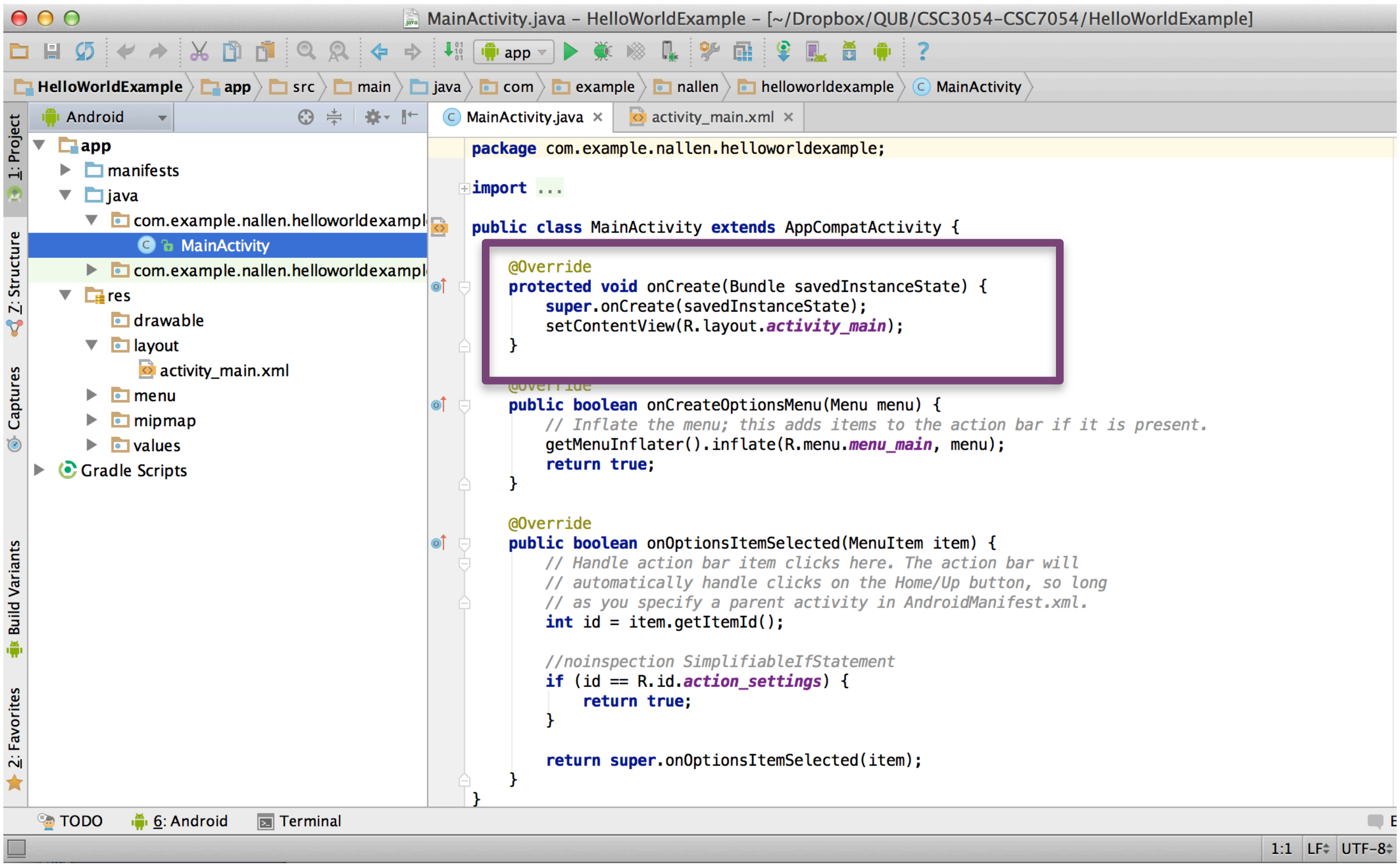
Step 6 - Look at the source code

This app will display a simple screen with the words `Hello world!` Open up the file `MainActivity.java` via the project window.



```java
package com.example.nallen.helloworldexample;

import ...

public class MainActivity extends AppCompatActivity {

    @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);
    }
}
```

This will opens up an editor window showing the contents of the file. When this application runs, the `onCreate()` method of this class will be called. This code will set up and display the application's user interface screen.

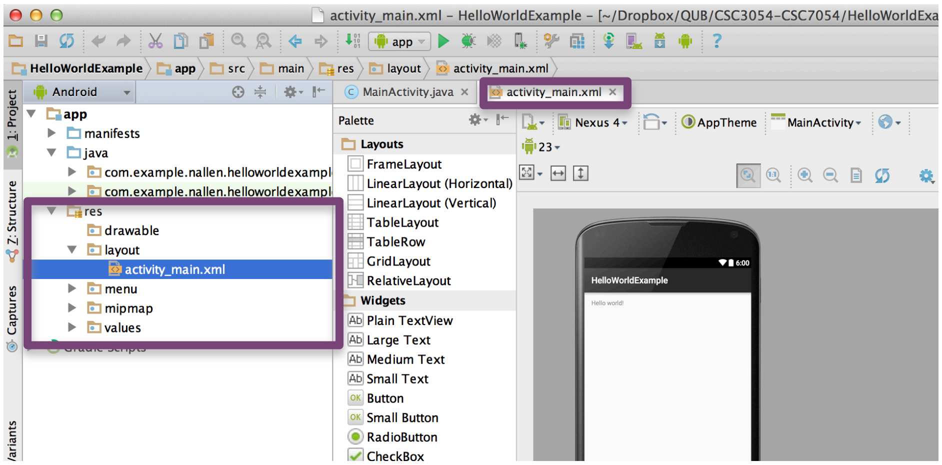


MainActivity.java – HelloWorldExample – [~/Dropbox/QUB/CSC3054-CSC7054/HelloWorldExample]
app
HelloWorldExample › app › src › main › java › com › example › nallen › helloworldexample › MainActivity
Android
MainActivity.java ×
activity_main.xml ×
app
manifests
java
com.example.nallen.helloworldexampl
MainActivity
com.example.nallen.helloworldexampl
res
drawable
layout
activity_main.xml
menu
mipmap
values
Gradle Scripts
1: Project
7: Structure
Captures
Build Variants
2: Favorites
TODO
6: Android
Terminal
1:1
LF÷
UTF-8÷

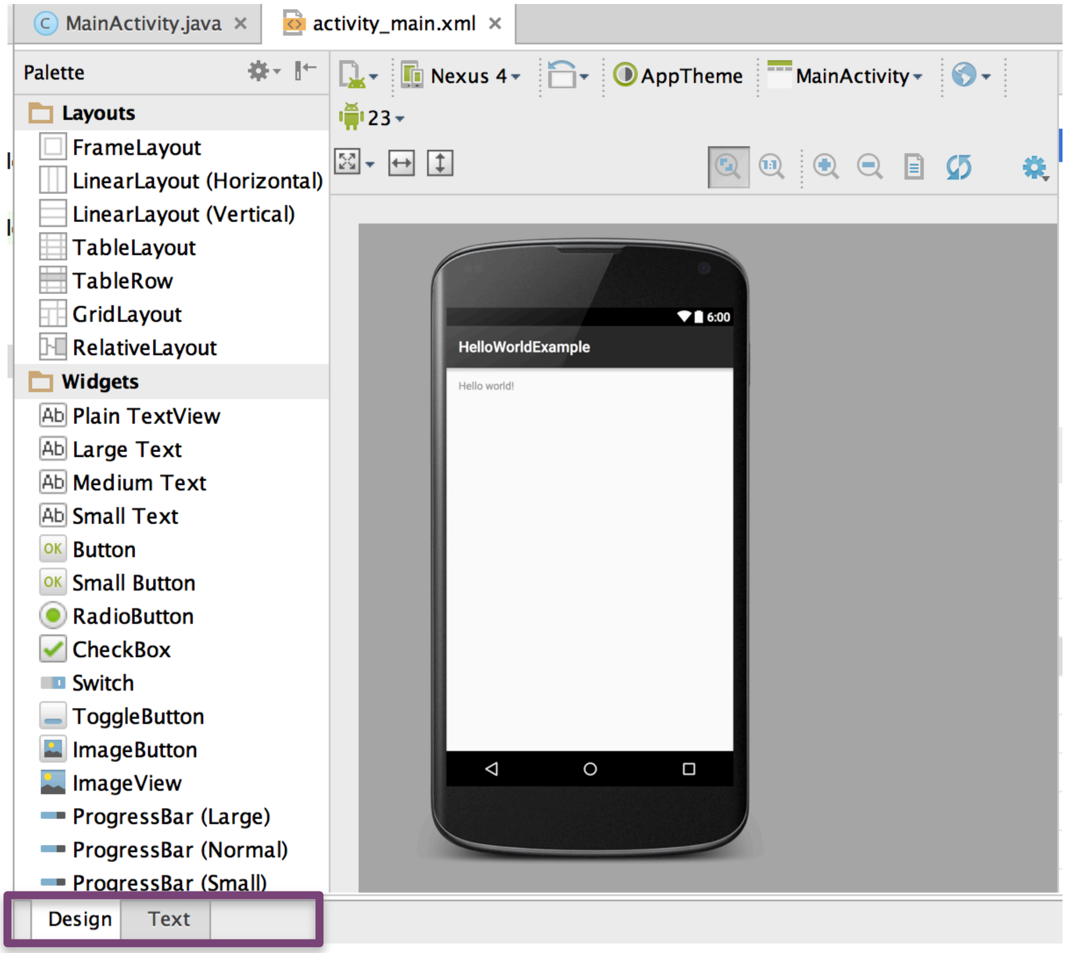
```java
package com.example.nallen.helloworldexample;

import ...

public class MainActivity extends AppCompatActivity {

    @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);
    }
}
```

Step 7 – Look at the XML Code

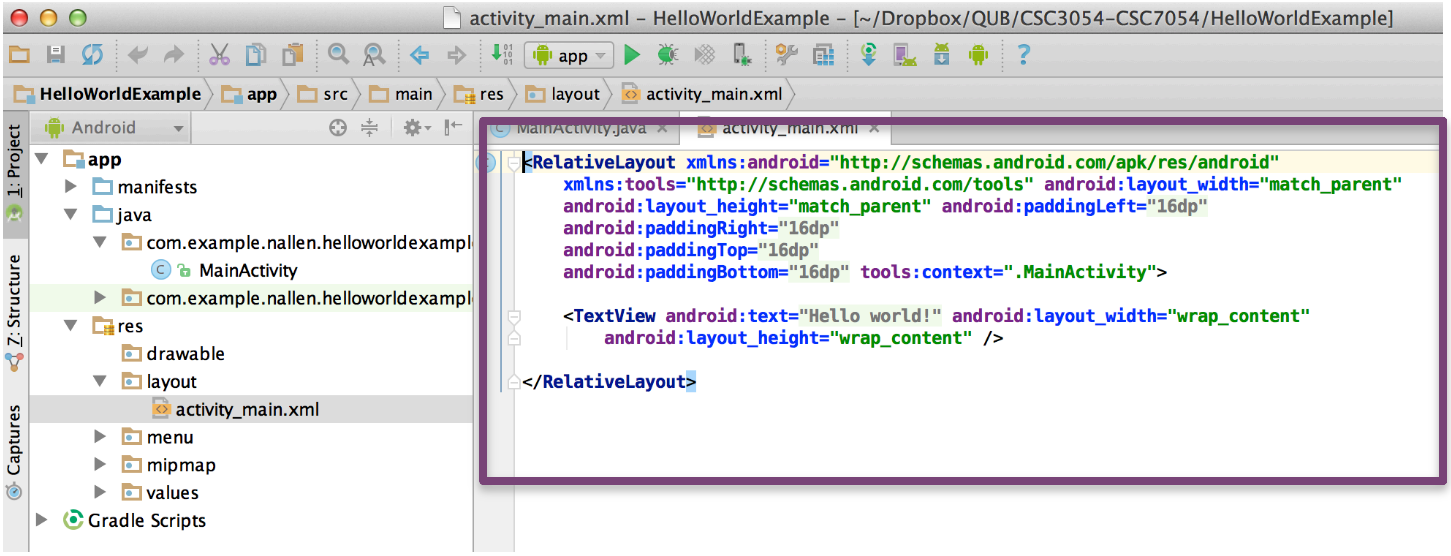
This app will display a simple screen with the words `Hello world!` Open up the file `activity_main.xml` (In the Project Window go to `res/layout.activity_main.xml`).



Switch from the `Design` view to `Text` view to see the XML code that sets up the layout for the app.



The XML code should be displayed as shown below:



```
<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:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">

    <TextView android:text="Hello world!" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

Step 8 – Run your app

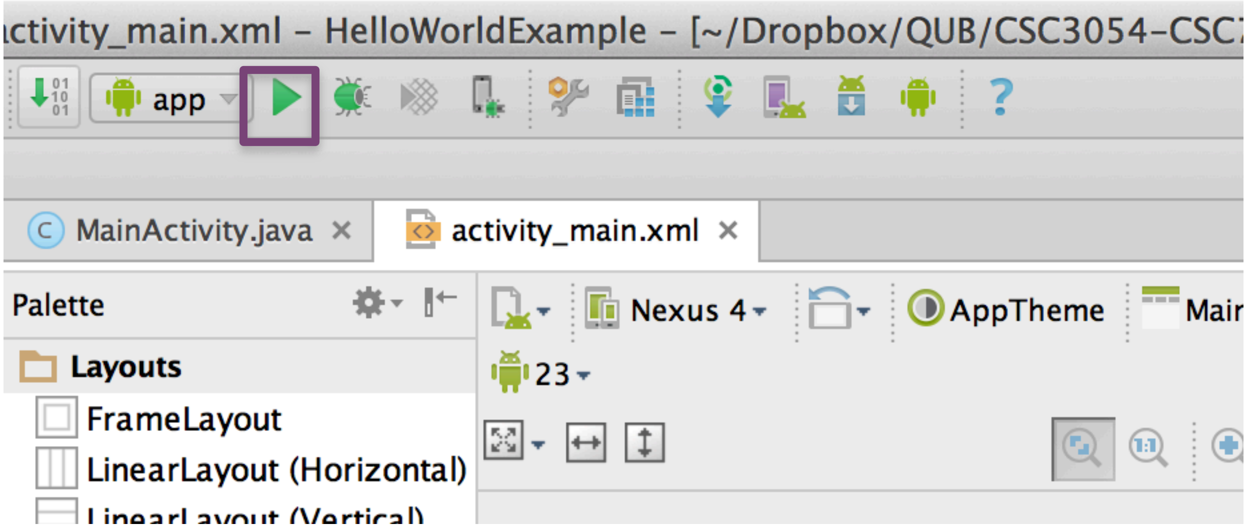
The app can be tested either using the emulator (setup is described in the next tutorial) or via an actual Android Phone.

To run the app using the emulator

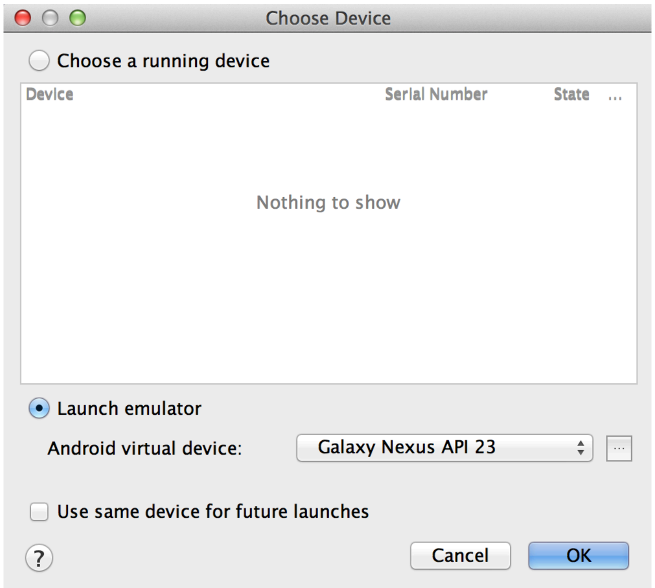
- Please refer to the tutorial "Working with the emulator"

To run the app on an Android Device

- Connect your phone to the computer via a USB cable
- Click on, `Run` in the application bar. This will build your application.



- A dialog box will pop up, asking where to run this application. When it appear press cancel.


Choose Device

○ Choose a running device

Device	Serial Number	State	...

Nothing to show

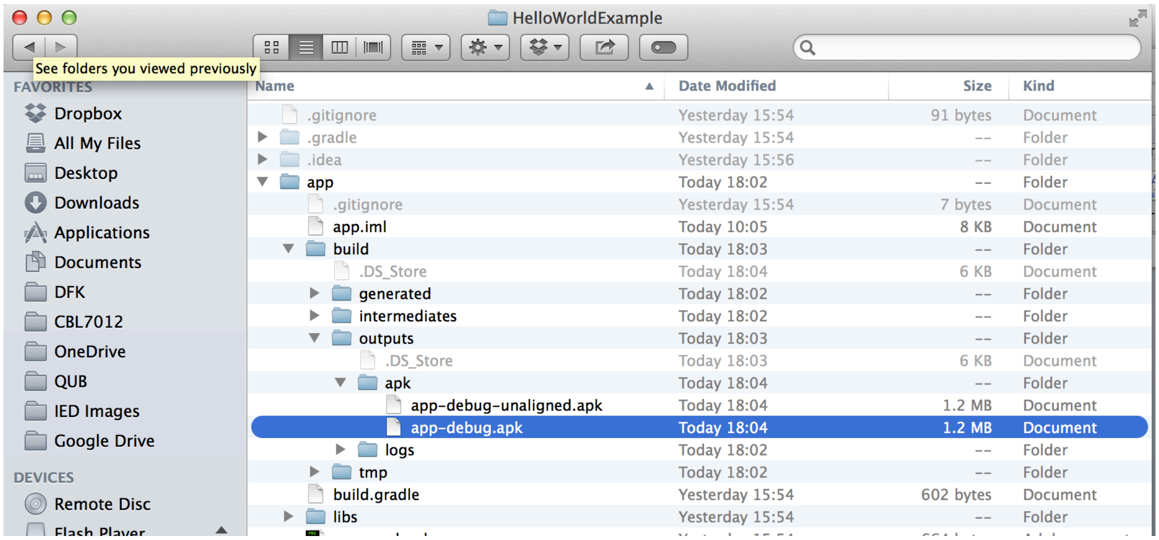
● Launch emulator

Android virtual device: Galaxy Nexus API 23

□ Use same device for future launches

- Go to Windows Explorer or Finder to your Android Studio Project

- Navigate to the `app/build/outputs/apk/apk-debug.apk` file



Name	Date Modified	Size	Kind
.gitignore	Yesterday 15:54	91 bytes	Document
.gradle	Yesterday 15:54	--	Folder
.idea	Yesterday 15:56	--	Folder
app	Today 18:02	--	Folder
.gitignore	Yesterday 15:54	7 bytes	Document
app.iml	Today 10:05	8 KB	Document
build	Today 18:03	--	Folder
.DS_Store	Today 18:04	6 KB	Document
generated	Today 18:02	--	Folder
intermediates	Today 18:02	--	Folder
outputs	Today 18:03	--	Folder
.DS_Store	Today 18:03	6 KB	Document
apk	Today 18:04	--	Folder
app-debug-unaligned.apk	Today 18:04	1.2 MB	Document
app-debug.apk	Today 18:04	1.2 MB	Document
logs	Today 18:02	--	Folder
tmp	Today 18:02	--	Folder
build.gradle	Yesterday 15:54	602 bytes	Document
libs	Yesterday 15:54	--	Folder

- Copy this file onto your `Android Device`

- On your `Android Device`, go to `My Files` and install the app onto the `Android Device`

- `Run` the app