Lab 10
Hello Android

NetDB

CS, NTHU,
Fall, 2013

Outline

- Android Fundamental
- User Interface Overview
- Environment Set Up
- Your First App
- Today’s Mission
- Reference

Outline

- Android Fundamental
- User Interface Overview
- Environment Set Up
- Your First App
- Today’s Mission
- Reference

App Components

- Activities
  - An activity represents a single screen with a user interface
- Services
  - A service is a component that runs in the background to perform long-running operations or to perform work for remote processes
- Content Providers
  - A content provider manages a shared set of application data
- Broadcast Receivers
  - A broadcast receivers is a component that responds to system-wide broadcast announcements

Activities

- An app component that provides a screen with which users can iterate in order to do something
- An application usually consists of multiple activities that are loosely bound to each other
- Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack
- A different application can start any other application's activities (if it got the permission)

Creating an Activity

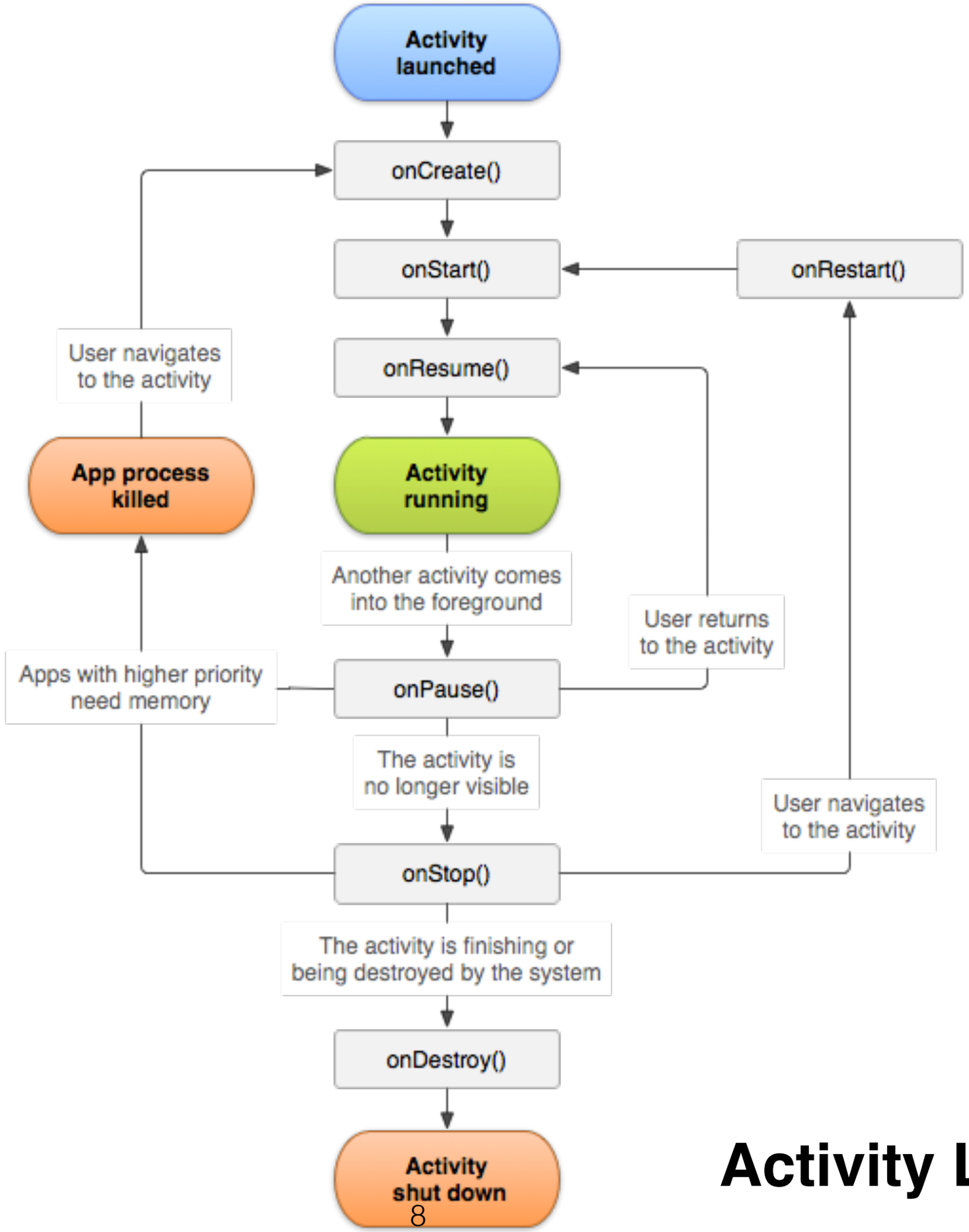
- To create an activity, you must create a subclass of Activity and implement callback methods
- **onCreate()**
  - The system calls this method when creating your activity
- **onPause()**
  - The system calls this method as the first indication that the user is leaving your activity (not always mean the activity is being destroyed)
- You can implement other callback methods that the system calls when the activity transitions between various state of its lifecycle

Starting an Activity

- You can start another activity by calling **startActivity()**, passing it an Intent that describes the activity you want to start

- We will discuss Intent further in next section

```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```



Activity Lifecycle

Three Essential States

- Resumed (running)
  - The activity is in the foreground of the screen and has user focus
- Paused
  - Another activity is in the foreground and has focus, but this one is still visible
- Stopped
  - The activity is completely obscured by another activity
  - A stopped activity is also still alive but it is no longer visible to the user and it can be killed by the system when memory is needed elsewhere.

Three Lifetimes

- Entire lifetime
  - Between the call to **onCreate()** and **onDestroy()**
  - Your activity should perform setup in **onCreate()** and release all remaining resource in **onDestroy()**
- Visible lifetime
  - Between the call to **onStart()** and **onStop()**
  - During this time, the user can see the activity on-screen and interact with it
- Foreground lifetime
  - Between the call to **onResume()** and **onPause()**
  - During this time, the activity is in front of all other activities on screen and has user input focus

For More Information

- For more information about Activities, please visit these pages:
  - Activities
  - Fragments
  - Managing the Activity Lifecycle

Intent & Intent Filter

- Activities, services, and broadcast receivers are activated through messages, called **intents**
- A passive data structure holding an abstract description of an operation to be performed

```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```

Inside an Intent Object

- An Intent object is a bundle of information that contains information of interest to the component that receives the intent
- Intent Object
  - Component Name
  - Action
  - Data
  - Category
  - Extras
  - Flags

Intent Filter

- Activities, services, and broadcast receivers can have one or more intent filters that inform the system which implicit intents they can handle
- Each filter describes a capability of the component, a set of intents that the component is willing to receive
- Intent filter are set up in the application's manifest file

Manifest

- Every application must have an **AndroidManifest.xml** file (with precisely that name) in its root directory
- The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code

Manifest

- The manifest file does the following:
  - Names the java package
  - Describes the components of the application
  - Declares the permissions
  - Declares the minimum level of the Android API
  - Lists the libraries that the application needs
  - ….

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myfirstapp"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-permission android:name="android.permission.RECEIVE_SMS" />

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.myfirstapp.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.example.myfirstapp.ResultActivity"
            android:label="@string/title_activity_result" >
        </activity>
    </application>

</manifest>
```

App Resource

- Applications usually have resources such as images and string
- You should always externalize resources from your application code, so that you can maintain them independently
  - It's good for maintenance
  - Provide alternative resources that support specific device configurations such as different languages or screen sizes

Providing Resources

- You should place each type of resource in a specific subdirectory of your project's res/ directory
- The resource directory names are important and are described in table 1

```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            icon.png
        layout/
            main.xml
            info.xml
        values/
            strings.xml
```

Providing Resources

- The resources that you save in the subdirectories defined in table 1 are your **default resources**. That is, these resources define the default design and content for your application

- However, different types of Android-powered devices might call for different types of resources

  - e.g. different layout for landscape orientation, different string file for different languages

Alternative Resources

- Almost every application should provide alternative resources to support specific device configurations
- Create a new directory in res/ named in the form <resources_name>-<config_qualifier>
  - <resources_name> is the directory name of the corresponding default resources
  - <qualifier> is a name that specifies an individual configuration for which these resources are to be used

Alternative Resources

- For example, if we want to set alternative resources for high density-screen, you could do this:

```
res/
    drawable/
        icon.png
        background.png
    drawable-hdpi/
        icon.png
        background.png
```

- More detailed qualifier names are described in <u>table 2</u>

Accessing Resources

- When your application is compiled, the **R class** is generated, which contains resource IDs for all the resources in your res/ directory
- A resource ID is always composed of the resource type and the resource name
  - In code, use <resource_type>.<resource_name>

```
imageView.setImageResource(R.drawable.myimage);
```

  - In XML, use <resource_type>/<resource_name>

```
android:text="@string/submit"
```

Outline

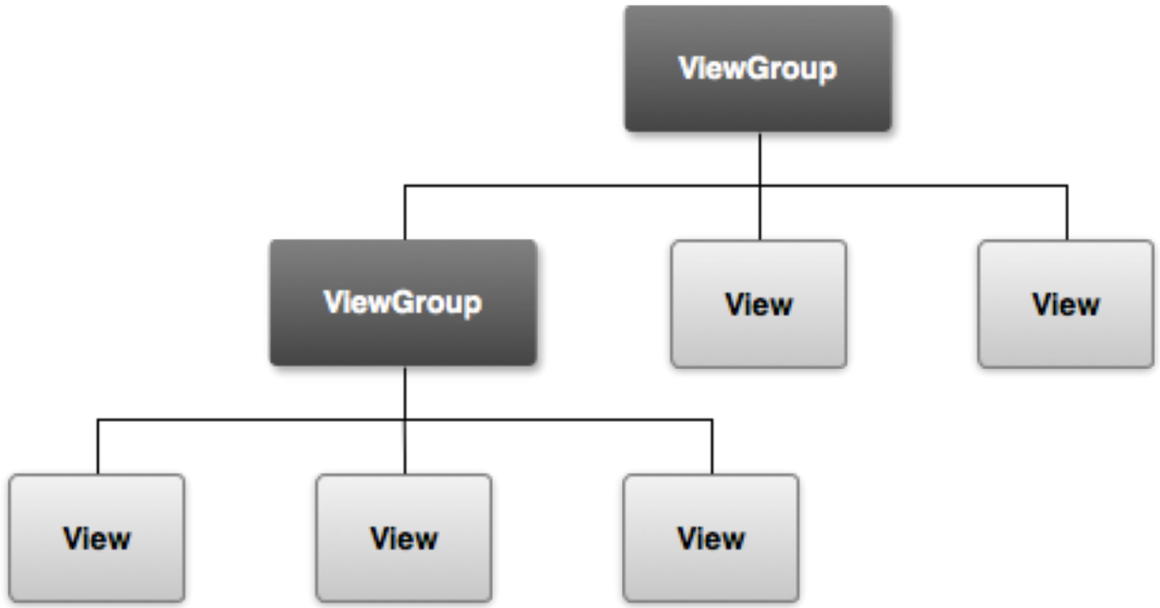
- Android Fundamental
- User Interface Overview
- Environment Set Up
- Your First App
- Today’s Mission
- Reference

User Interface

- All user interface elements in an Android app are built using View and ViewGroup objects
- View
  - An object that draws something on the screen that the user can interact with
- ViewGroup
  - An object that holds other View (and ViewGroup) objects in order to define the layout of the interface

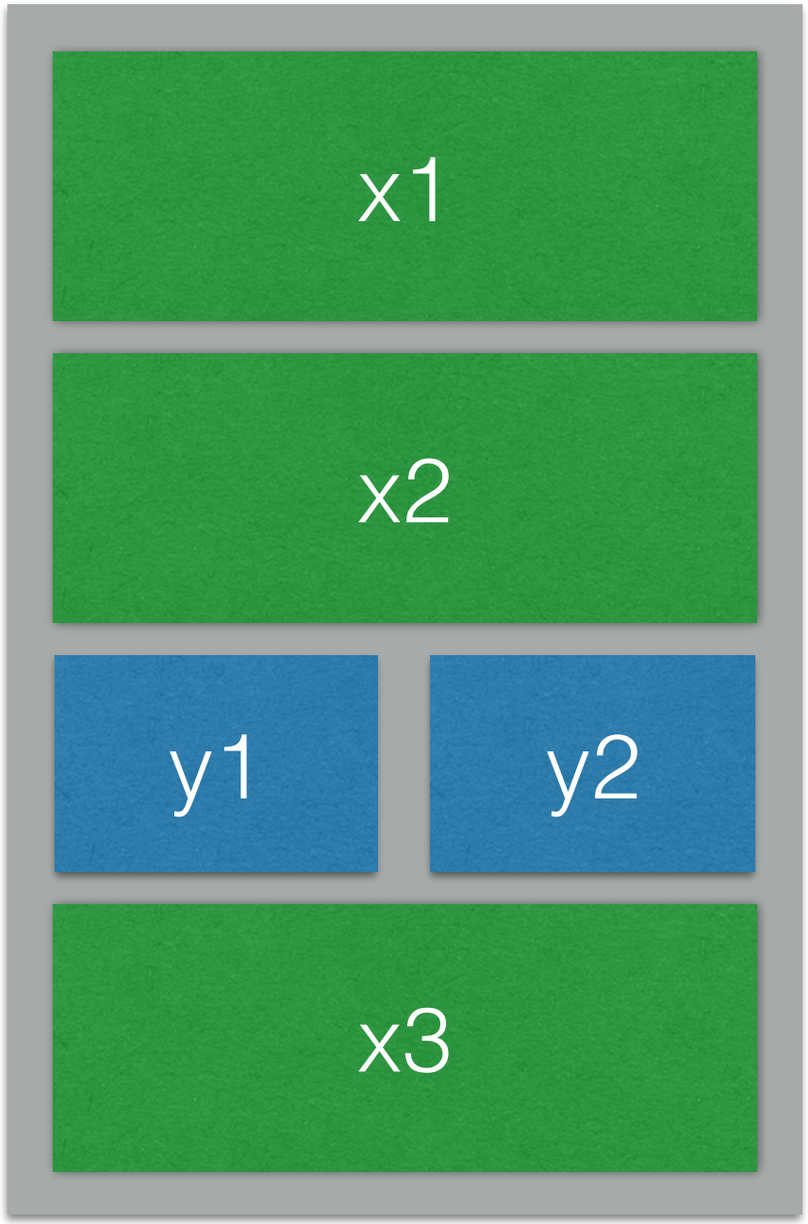
View Hierarchy

- The user interface for each component of your app is defined using a hierarchy of View and ViewGroup objects



View Hierarchy

- Linear Layout (Vertical)
  - x1
  - x2
  - Linear Layout (Horizontal)
    - y1
    - y2
  - x3



View Hierarchy

- To declare your layout, you can instantiate View objects in code and start building a tree
- The easiest and most effective way to define your layout is with an XML file

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I am a Button" />
</LinearLayout>
```

Layouts

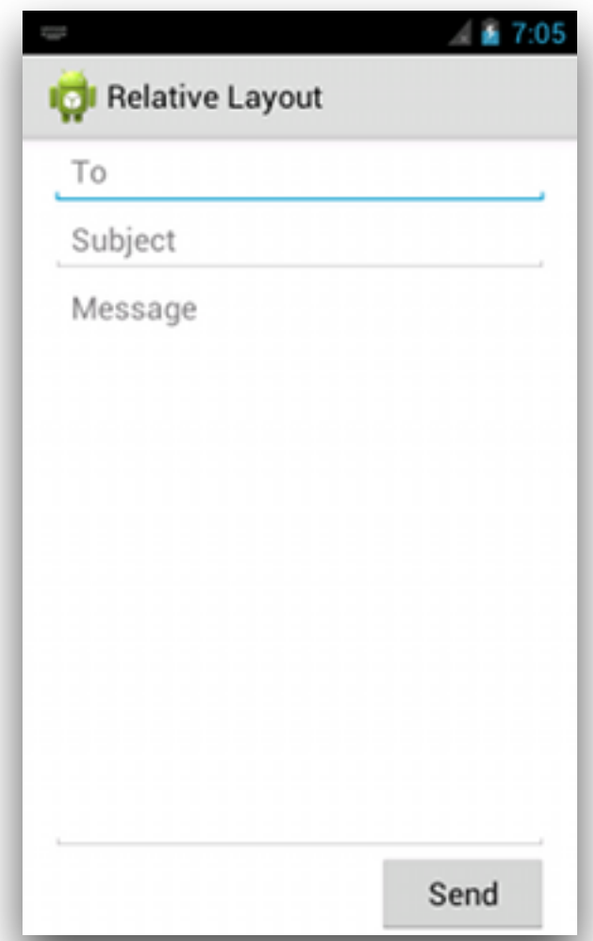
- A layout defines the visual structure for a user interface
  - Linear Layout
  - Relative Layout
  - List View
  - Grid View

Linear Layout

- LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally

- You can assign a weight to individual children state that how much space is should occupy on the screen

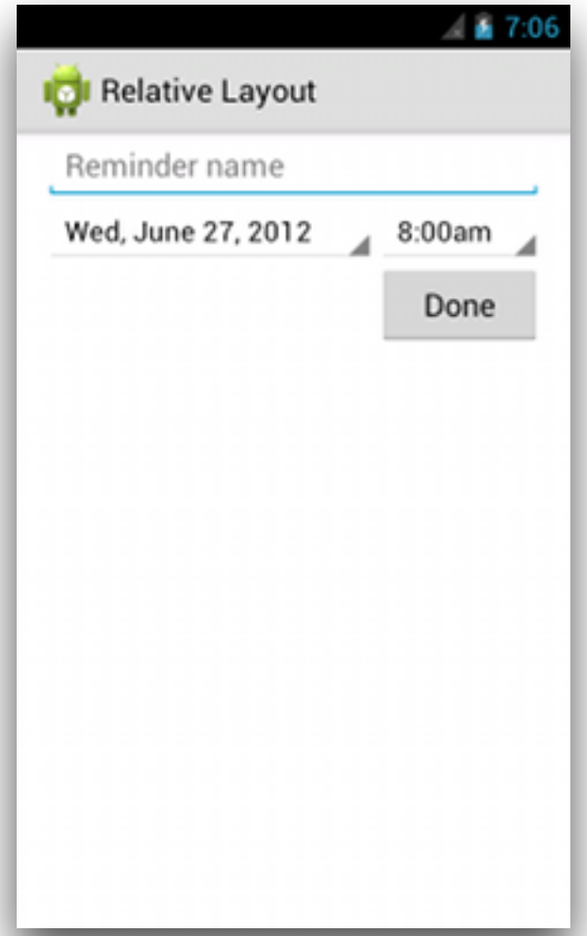
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



Relative Layout

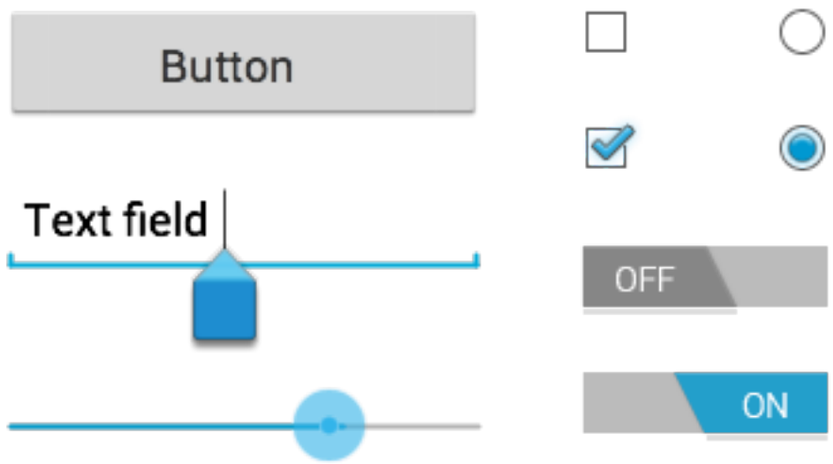
- RelativeLayout is a view group that displays child views in relative positions
- The position of each view can be specified as relative to sibling elements

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



Input Control

- Input controls are the interactive components in your app's user interface
- Android provides a wide variety of controls you can use in your UI
  - Button
  - Text fields
  - Seem bars
  - Checkboxes



Input Event

- Each input control supports a specific set of input events so you can handle events such as when the user enters text or touches a button
- For each event, you should handle it in a callback method in the Activity that host the layout

Input Event

- For example, to define the click event handler for a button, add the *android:onClick* attribute to the <Button> element in your XML layout

```
<?xml version="1.0" encoding="utf-8"?>
<Button xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

- Within the Activity that hosts this layout, the following method handles the click event

```
/** Called when the user touches the button */
public void sendMessage(View view) {
    // Do something in response to button click
}
```

For more information

- For more information, please check these pages on Google Android Development website:
  - User Interface Overview
  - Layouts
  - User Interface

Outline

- Android Fundamental
- User Interface Overview
- Environment Set Up
- Your First App
- Today’s Mission
- Reference

Go to this page and download the Android Windows bundle

Developers | Design | Develop | Distribute

Training | API Guides | Reference | Tools | Google Services | Samples

Developer Tools

- Download
  - Setting Up the ADT Bundle
  - Setting Up an Existing IDE
  - Android Studio
  - Exploring the SDK
  - Download the NDK
- Workflow
- Support Library
- Tools Help
- Revisions
- Samples
- ADK

Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in **ADT (Android Developer Tools)** to streamline your Android app development.

With a single download, the ADT Bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

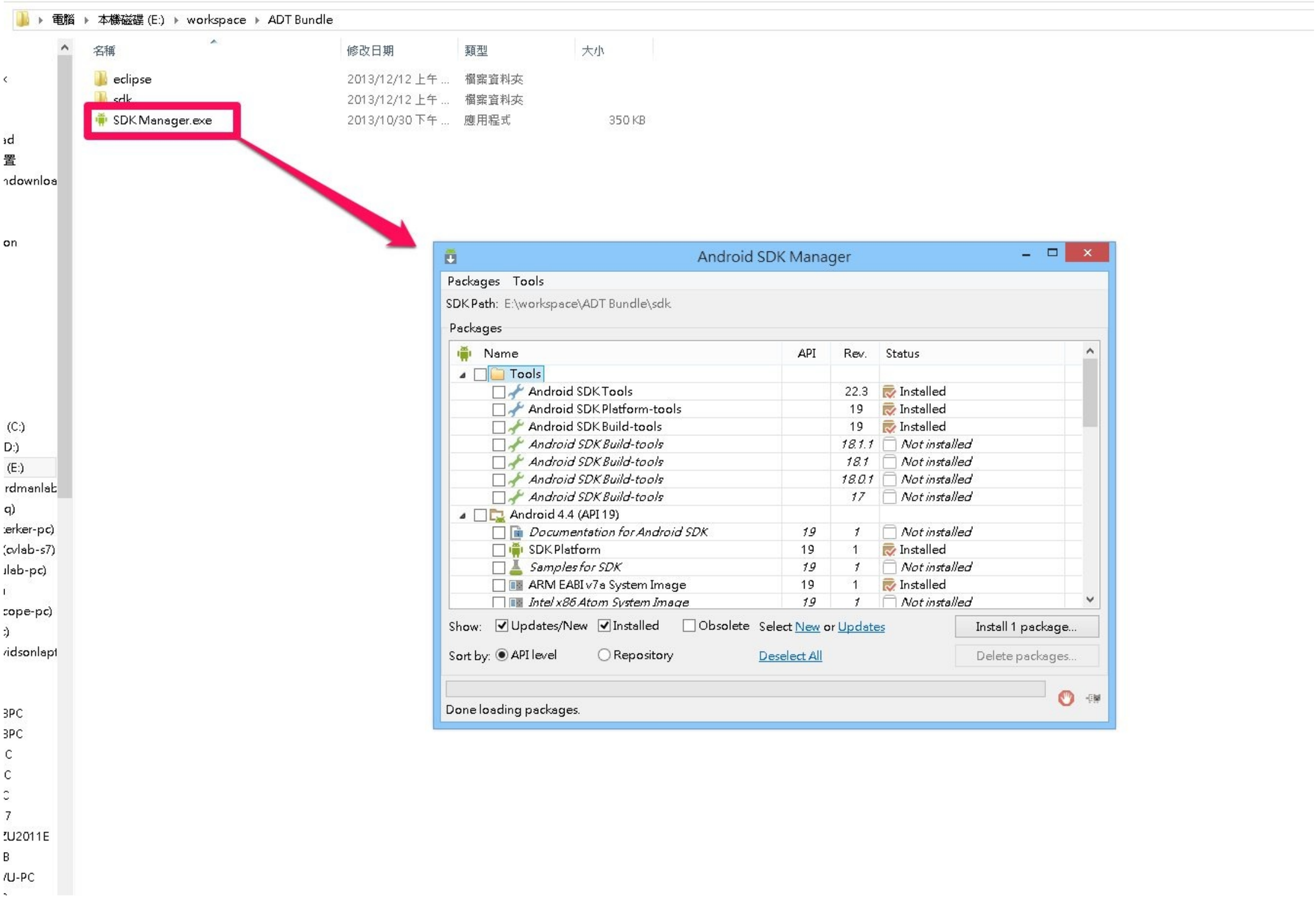
Download the SDK
ADT Bundle for Windows

Android Studio Early Access Preview

A new Android development environment called Android Studio, based on IntelliJ IDEA, is now available as an **early access preview**. For more information, see Getting Started with Android Studio.

If you prefer to use an existing version of Eclipse or another IDE, you can instead take a more customized approach to installing the Android SDK. See the following instructions:

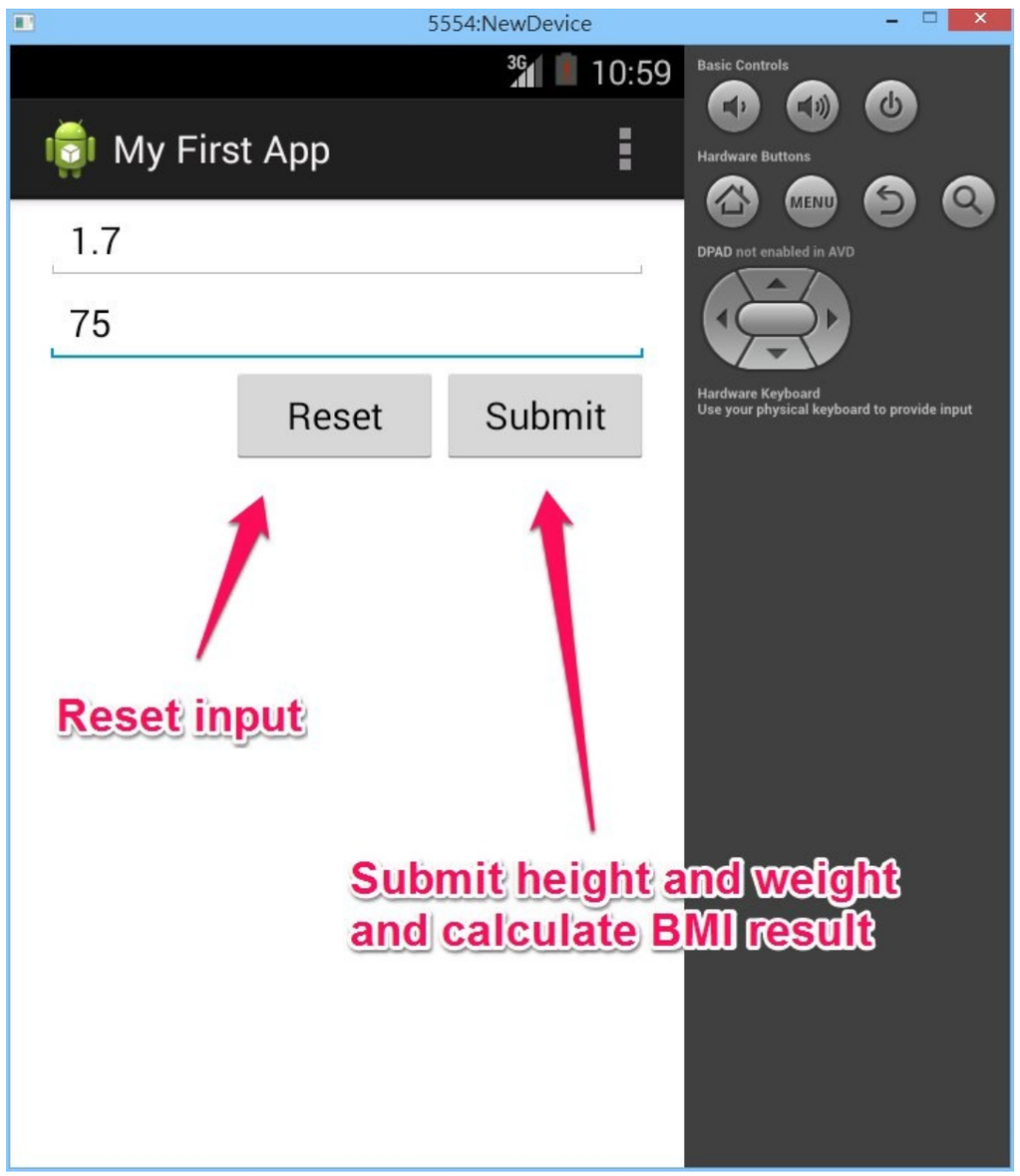
Download the latest SDK tools and platforms using the SDK Manager

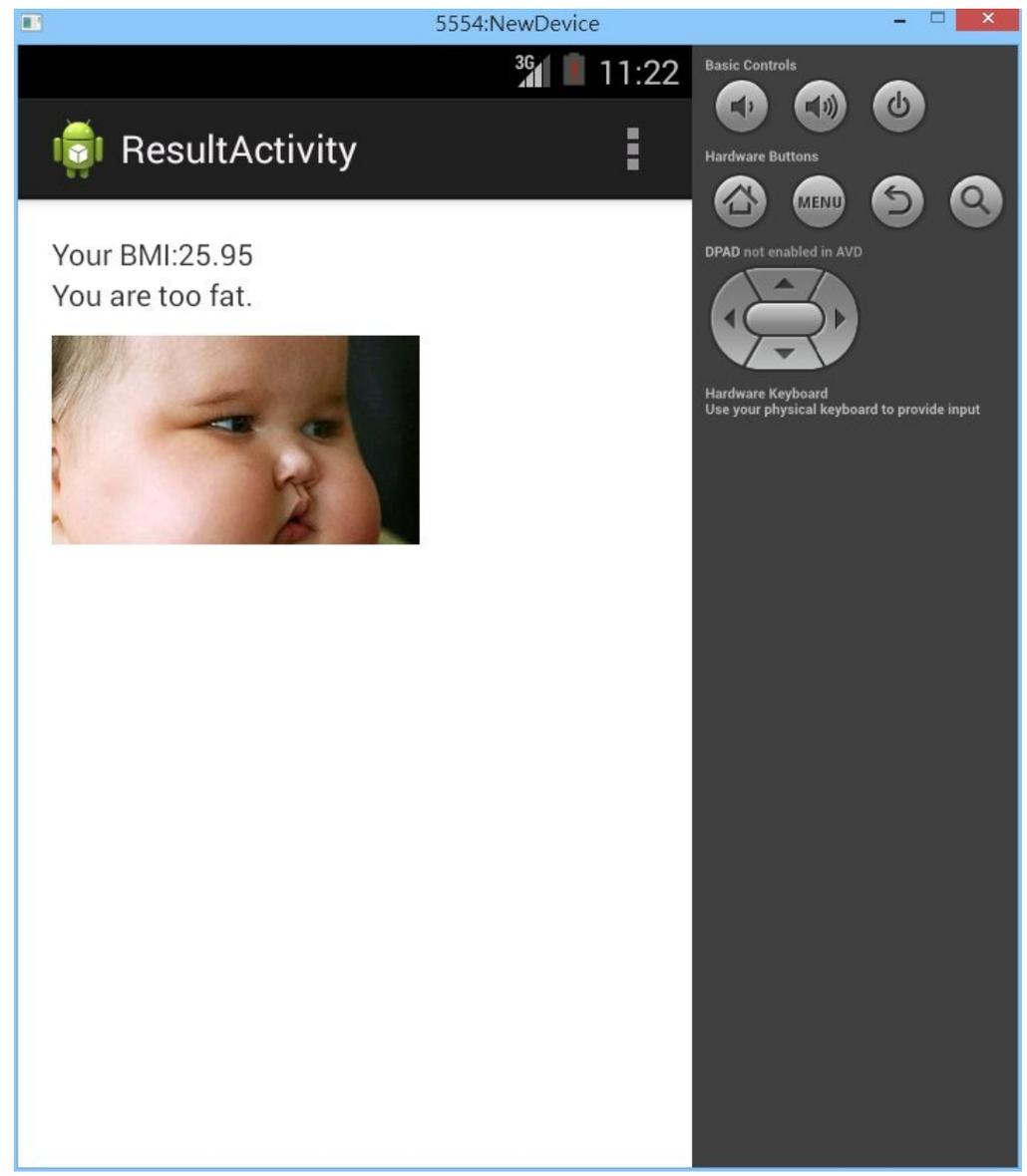


Outline

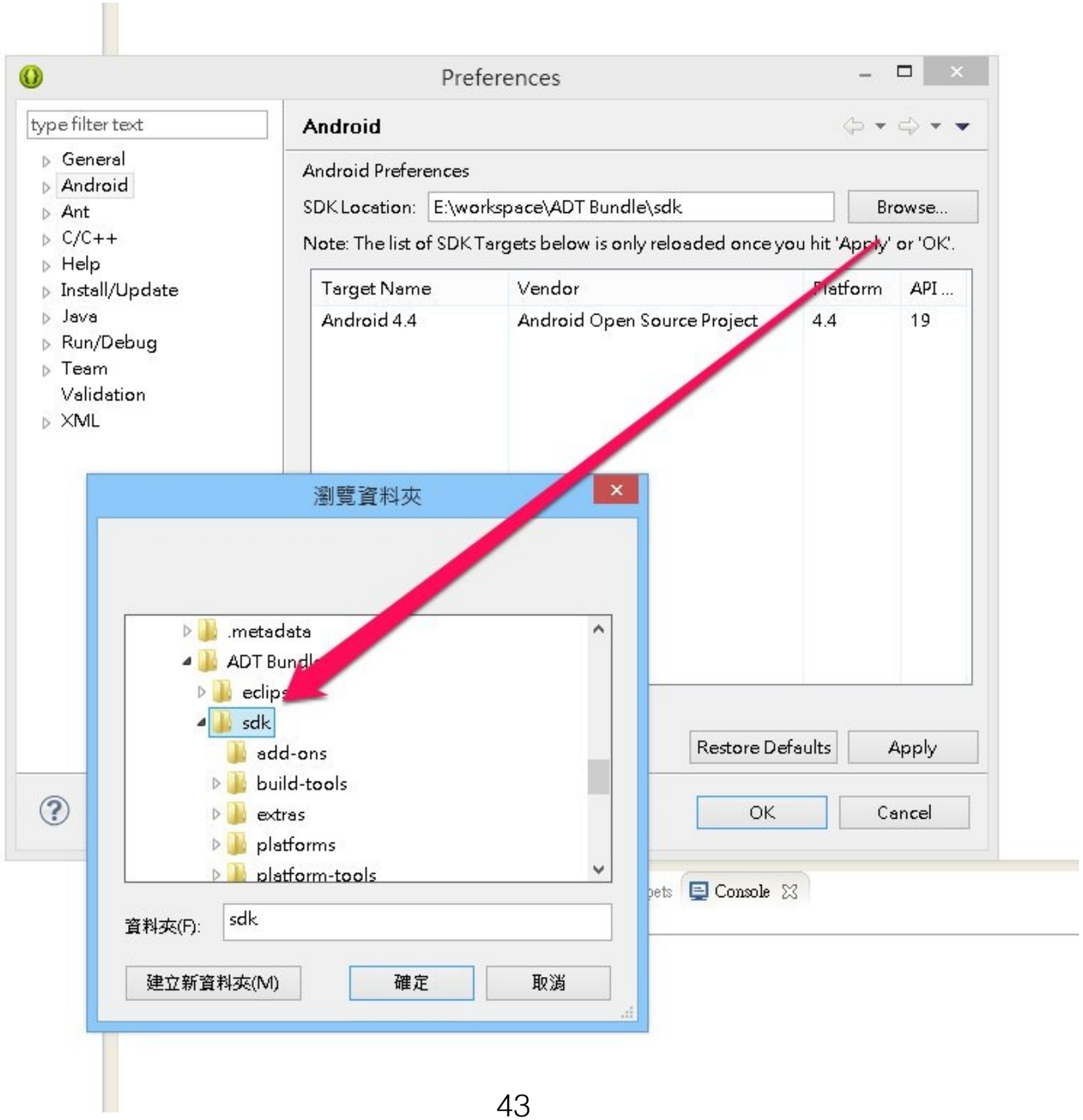
- Android Fundamental
- User Interface Overview
- Environment Set Up
- Your First App
- Today's Mission
- Reference

Simple BMI

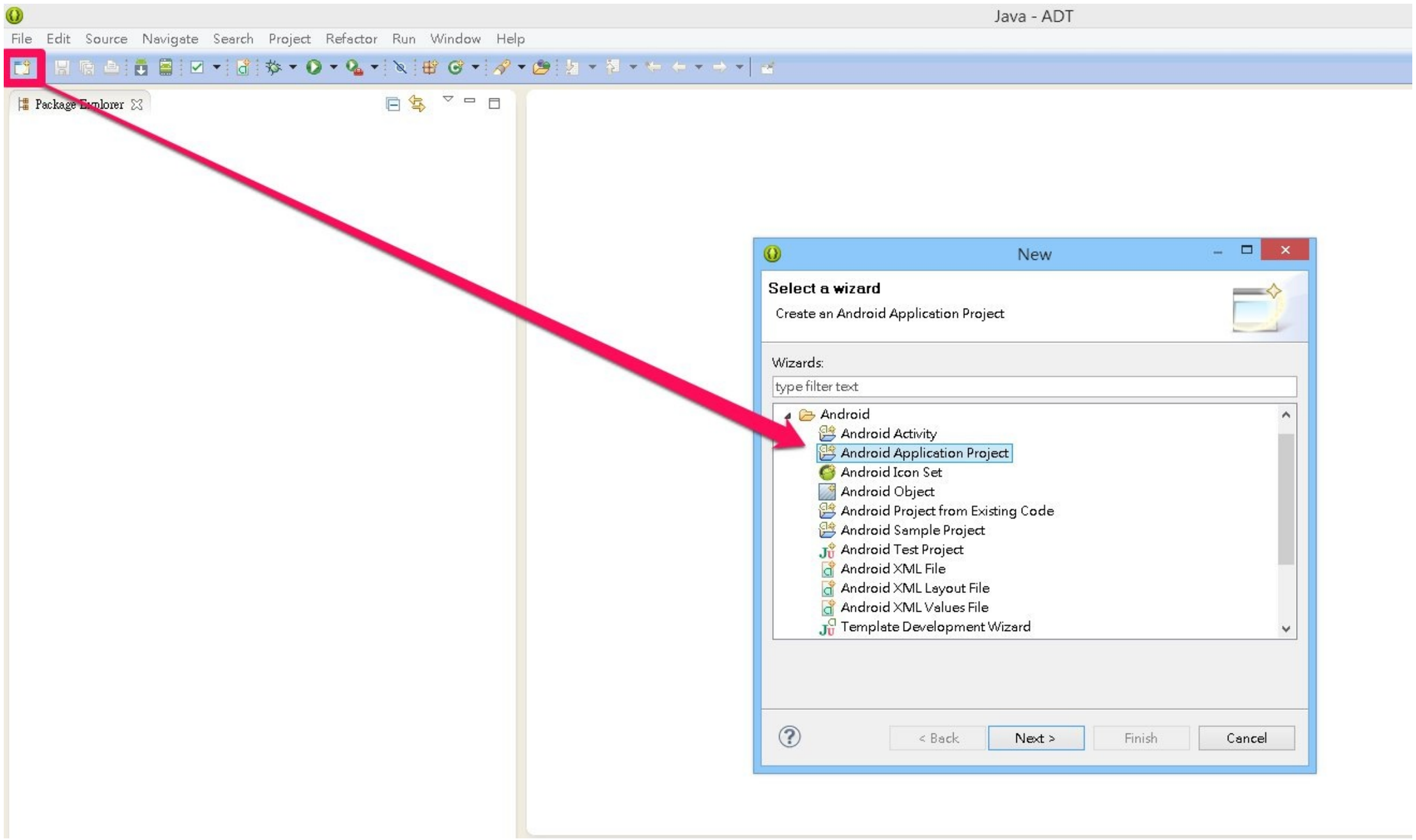

5554:NewDevice
10:59
My First App
1.7
75
Reset
Submit
Reset input
Submit height and weight
and calculate BMI result
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input


5554:NewDevice
11:22
ResultActivity
Your BMI:25.95
You are too fat.
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

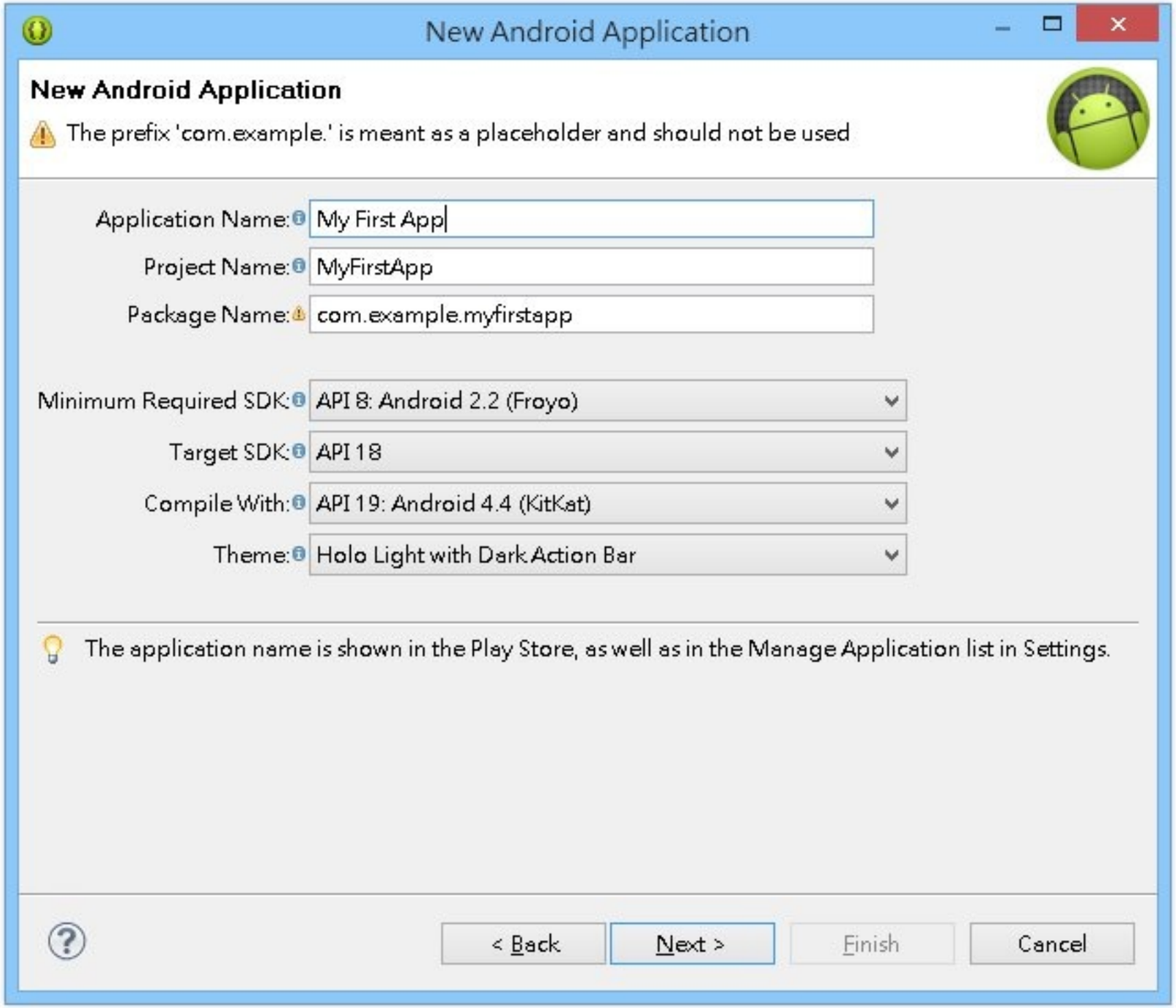
Reference the Android SDK



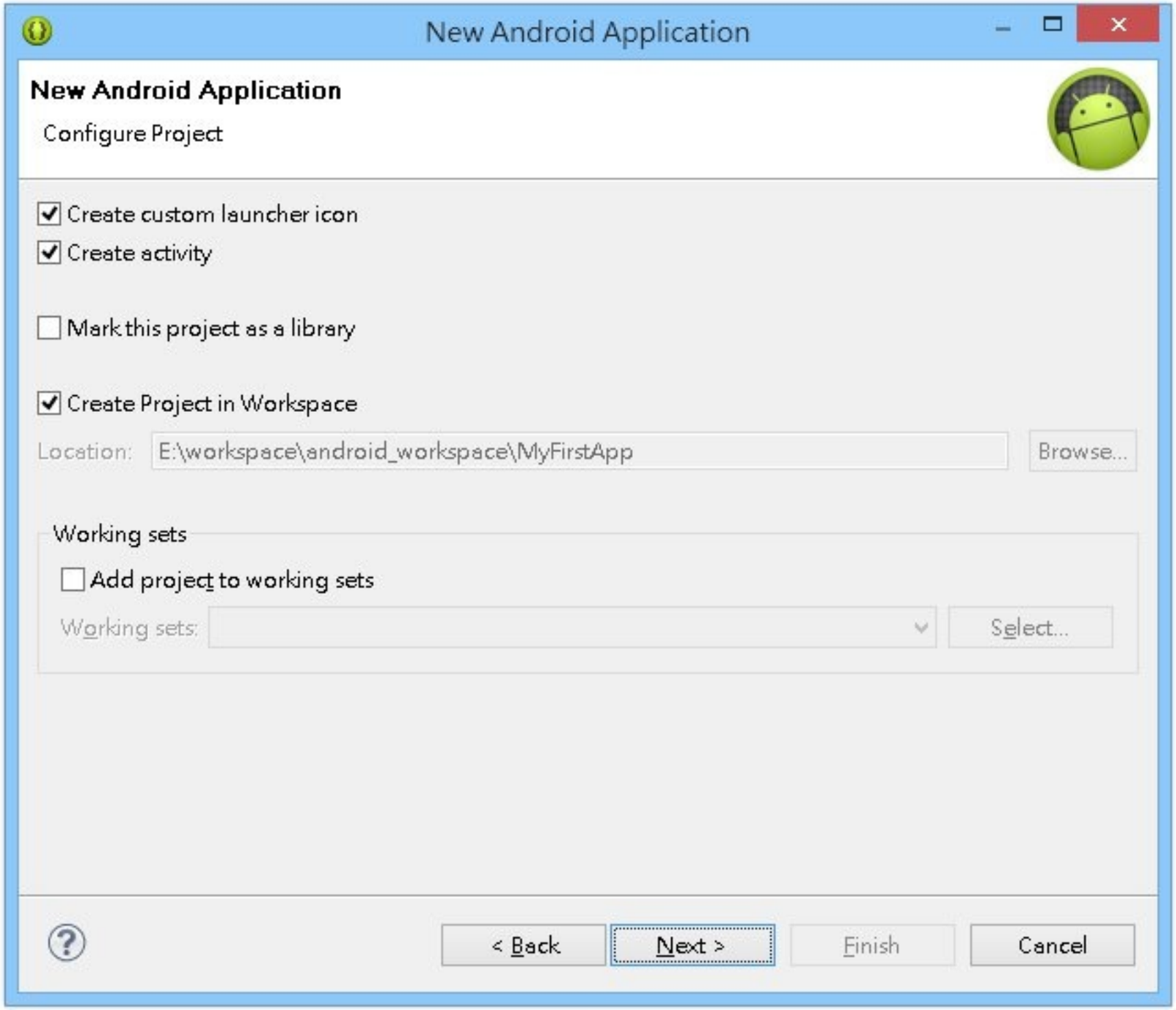
Create a new Project



Fill in the required information (Description of each field)



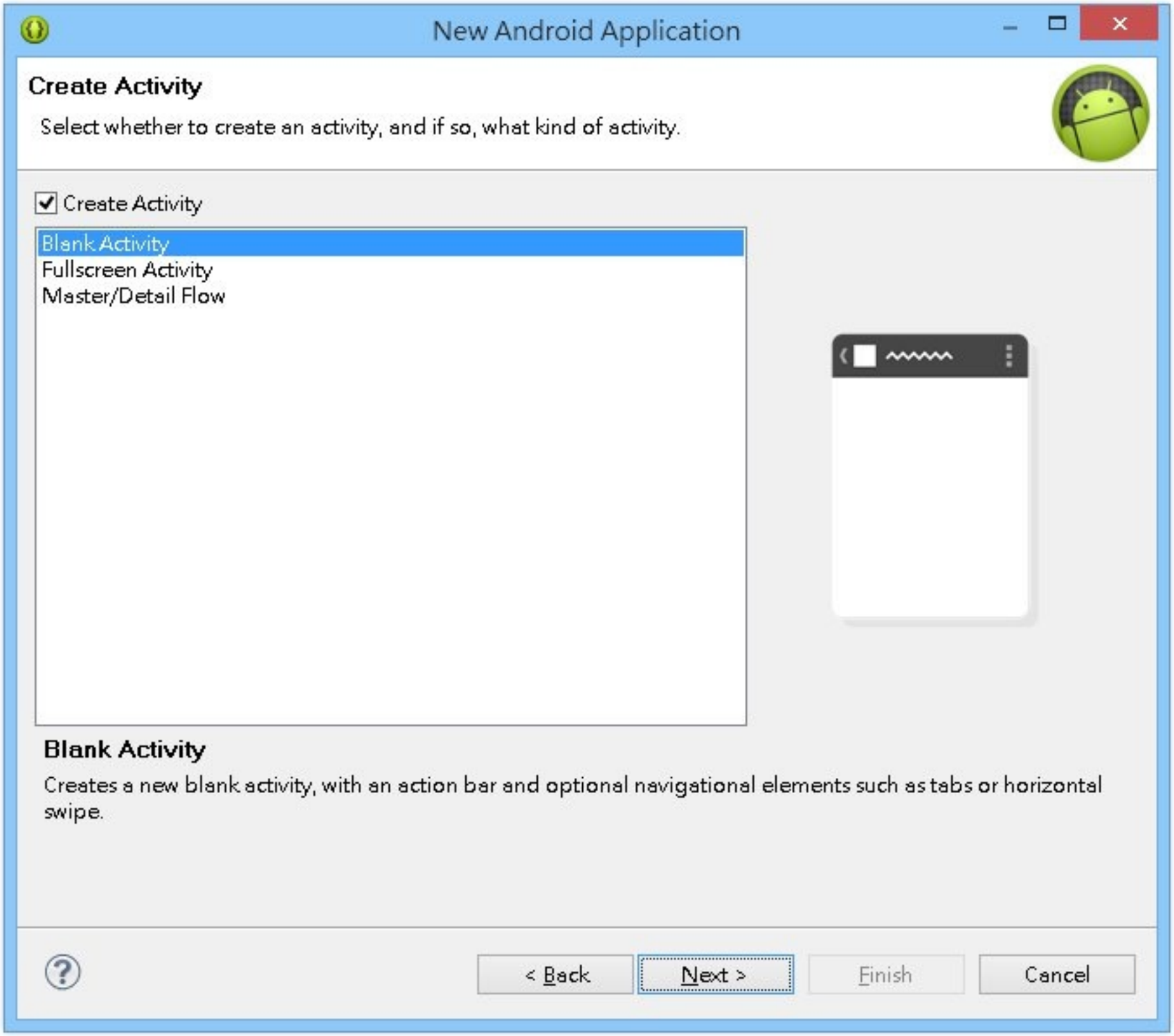
Leave the configuration as default and press Next



The next screen can help you create a launcher icon for your app. You can customize an icon in several ways and the tool generates an icon for all screen densities.

Before you publish your app, you should be sure your icon meets the specifications defined in the Iconography design guide

Select an activity template from which to begin your app



Leave all the details for the activity in their default state and click Finish

You have created your first android application

Java - MyFirstApp/src/com/example/myfirstapp/MainActivity.java - ADT

File Edit Source Navigate Search Project Refactor Run Window Help

Package Explorer

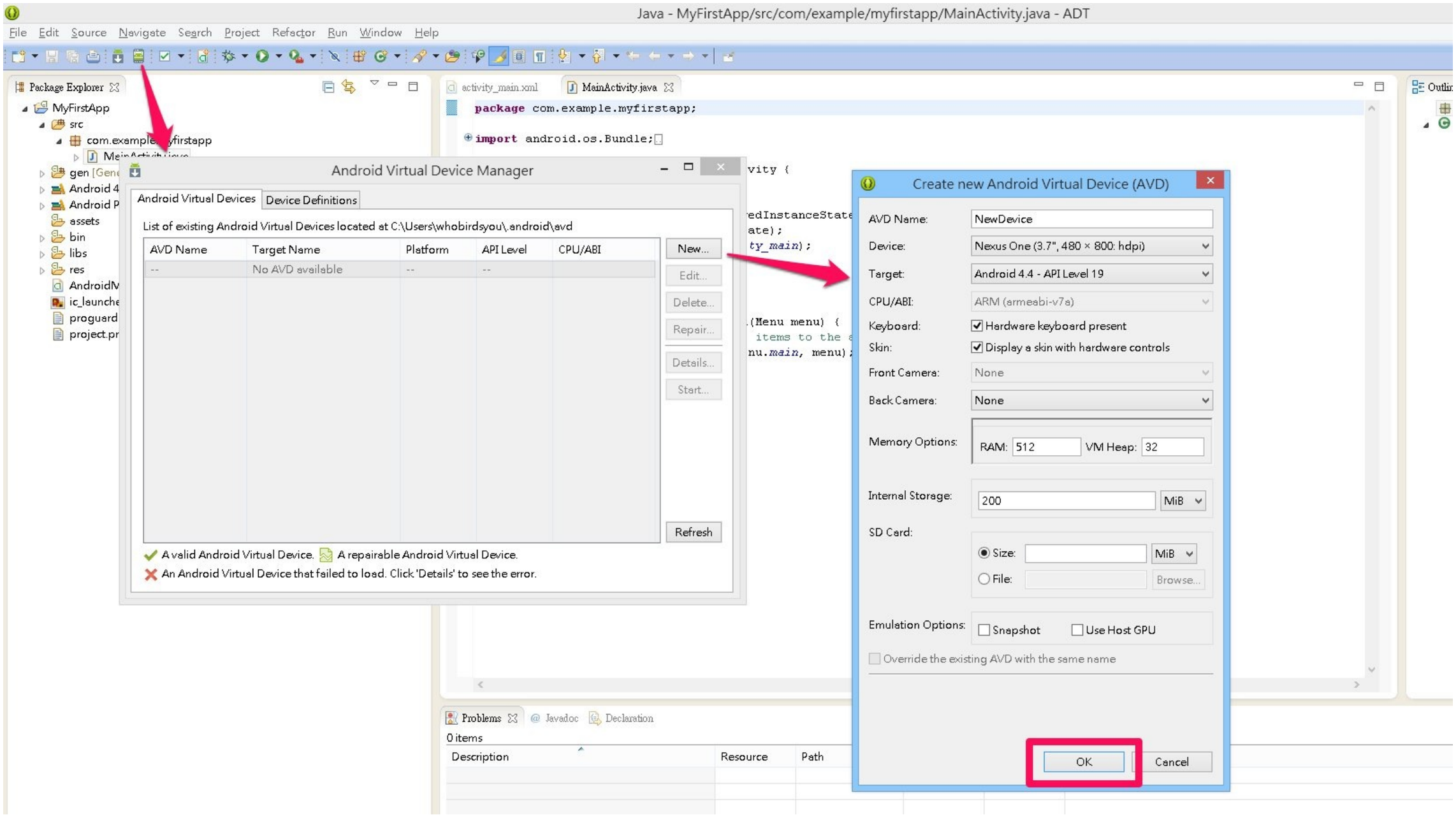
- MyFirstApp
  - src
    - com.example.myfirstapp
      - MainActivity.java
  - gen [Generated Java Files]
  - Android 4.4
  - Android Private Libraries
  - assets
  - bin
  - libs
  - res
  - AndroidManifest.xml
  - ic_launcher-web.png
  - proguard-project.txt
  - project.properties

activity_main.xml | MainActivity.java

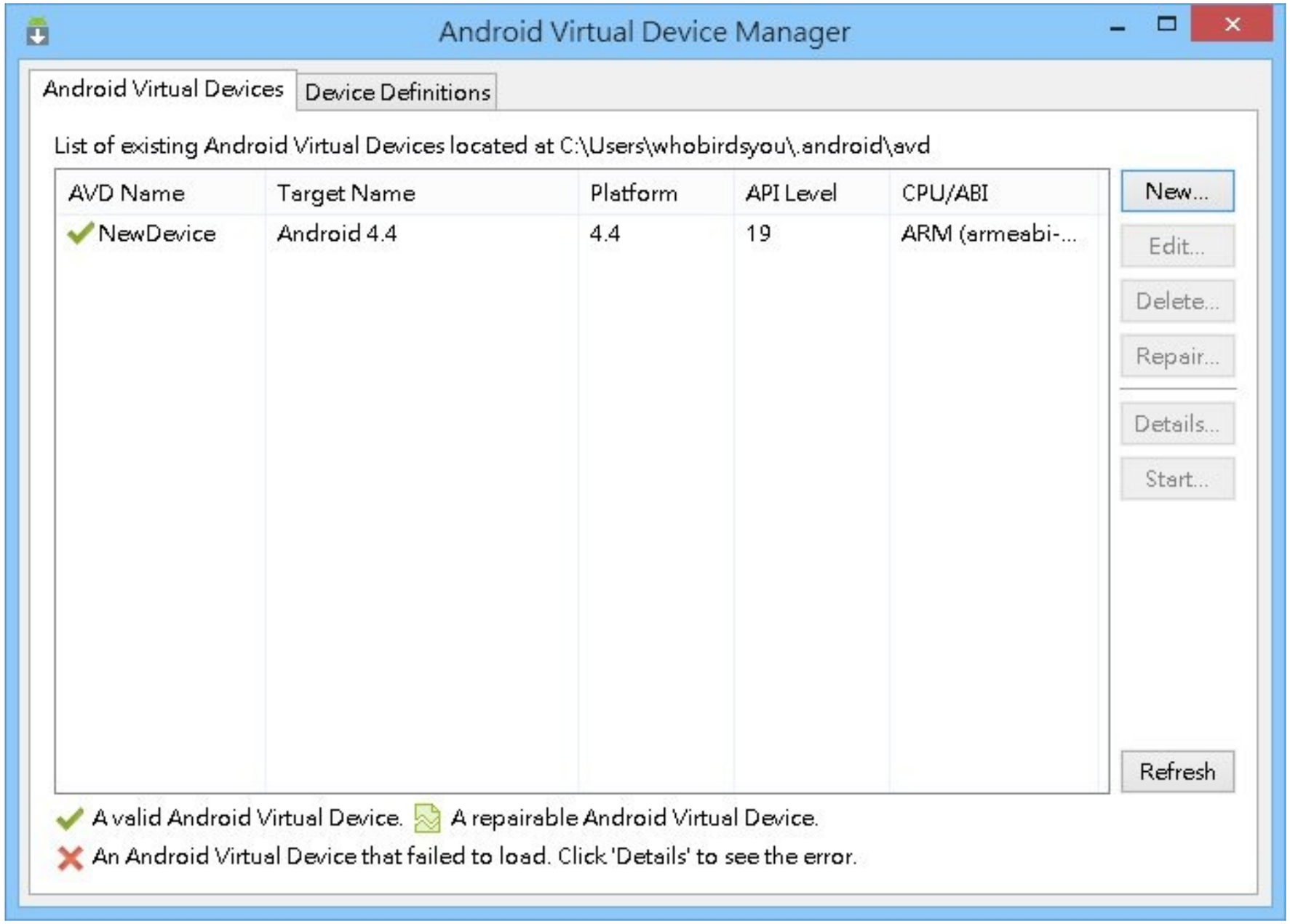
```java
package com.example.myfirstapp;

import android.os.Bundle;

public class MainActivity extends Activity {

    @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.main, menu);
        return true;
    }

}
```

Android Virtual Device

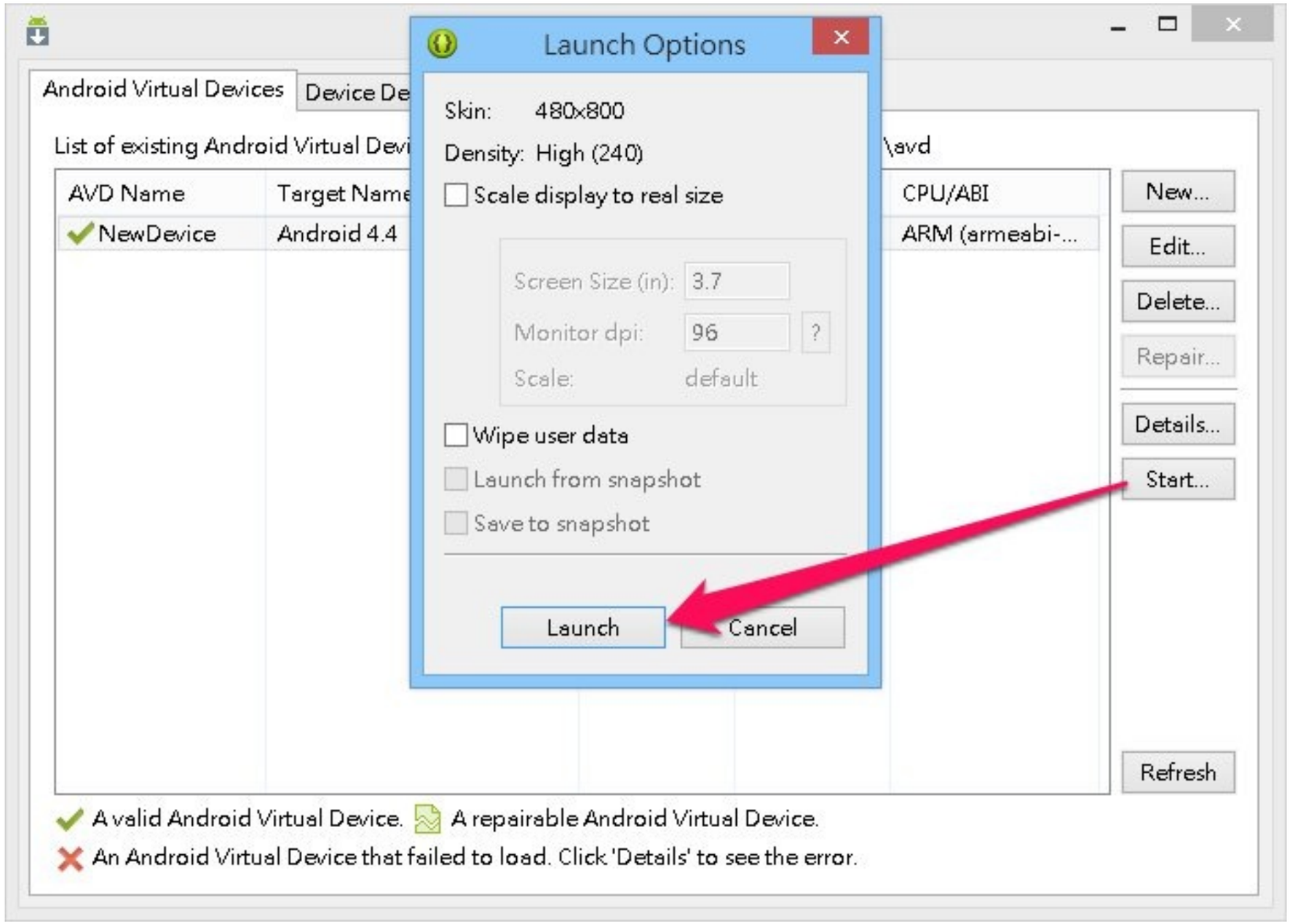
Create a new Android Virtual Device



After pressing the OK button, your new virtual device will be shown in the Device Manager



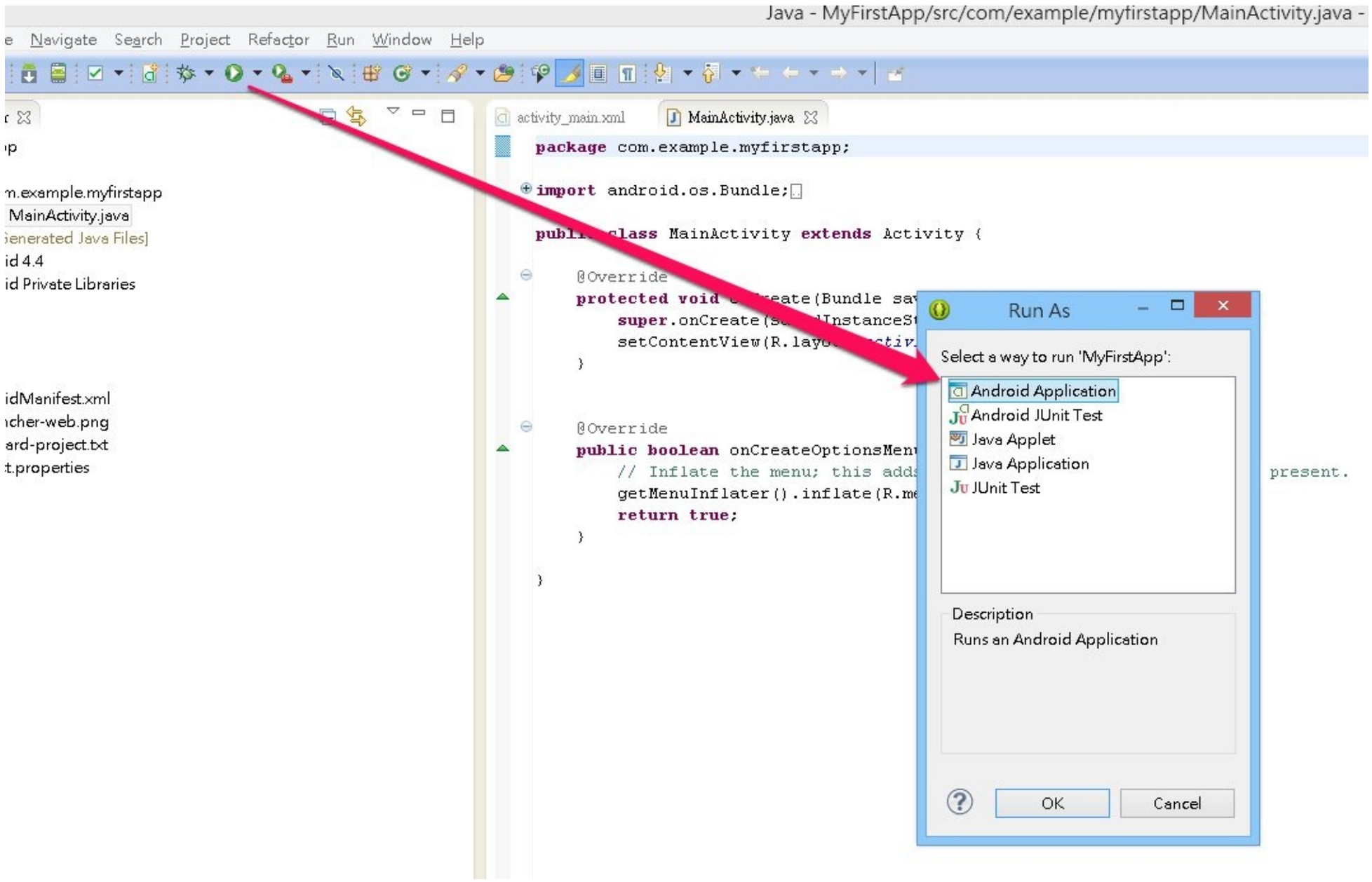
Launch the device



The virtual device will be launched and being initializing


5554:NewDevice
2:33
Google
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

Running an Android program



You can see the installation message shown in the console

Problems | @ Javadoc | Declaration | Console | LogCat

Android

```
[2013-12-15 15:33:45 - MyFirstApp] adb is running normally.
[2013-12-15 15:33:45 - MyFirstApp] Performing com.example.myfirstapp.MainActivity activity launch
[2013-12-15 15:33:45 - MyFirstApp] Automatic Target Mode: using existing emulator 'emulator-5554' running compatible AVD 'NewDevice'
[2013-12-15 15:33:45 - MyFirstApp] Uploading MyFirstApp.apk onto device 'emulator-5554'
[2013-12-15 15:33:48 - MyFirstApp] Installing MyFirstApp.apk...
[2013-12-15 15:33:53 - MyFirstApp] Success!
[2013-12-15 15:33:53 - MyFirstApp] Starting activity com.example.myfirstapp.MainActivity on device emulator-5554
[2013-12-15 15:33:55 - MyFirstApp] ActivityManager: Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.myfirstapp/.MainA
```

Start Building Our App

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/height_input"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:hint="@string/user_height" />
    <EditText
        android:id="@+id/weight_input"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:hint="@string/user_weight" />
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="right"
        android:orientation="horizontal" >

        <Button
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:onClick="resetInput"
            android:text="@string/btn_reset" />

        <Button
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:onClick="submitInput"
            android:text="@string/btn_submit" />
    </LinearLayout>
</LinearLayout>
```

In your res/layout/activity_main.xml

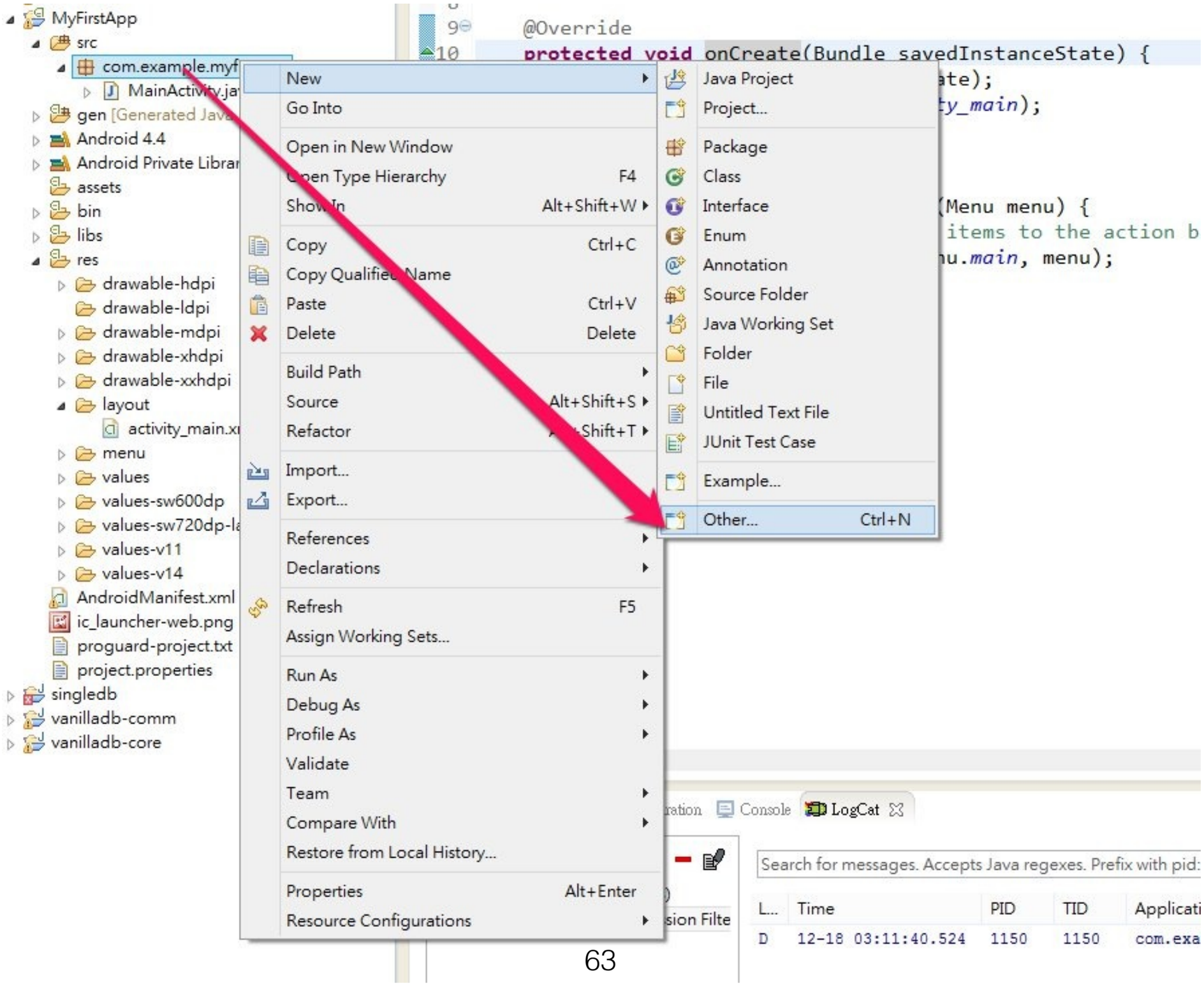
Add the String constants to the res/values/strings.xml

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

    <string name="app_name">My First App</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="title_activity_result">ResultActivity</string>

    <string name="user_height">Your height</string>
    <string name="user_weight">Your weight</string>
    <string name="btn_reset">Reset</string>
    <string name="btn_submit">Submit</string>


</resources>
```

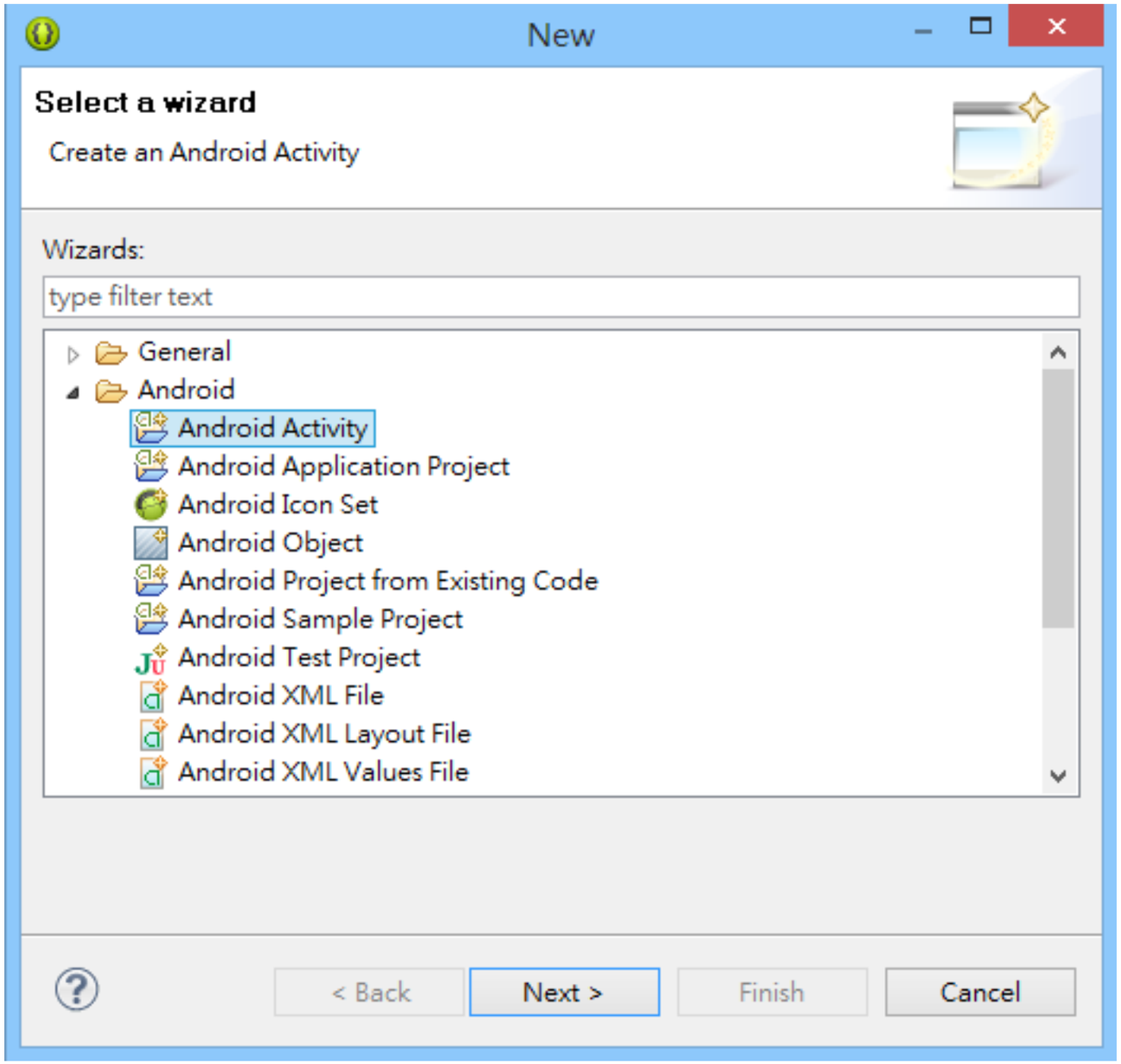
In the MainActivity.java

```
public class MainActivity extends Activity {
    private EditText heightInput;
    private EditText weightInput;

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

        heightInput = (EditText)findViewById(R.id.height_input);
        weightInput = (EditText)findViewById(R.id.weight_input);
    }
```

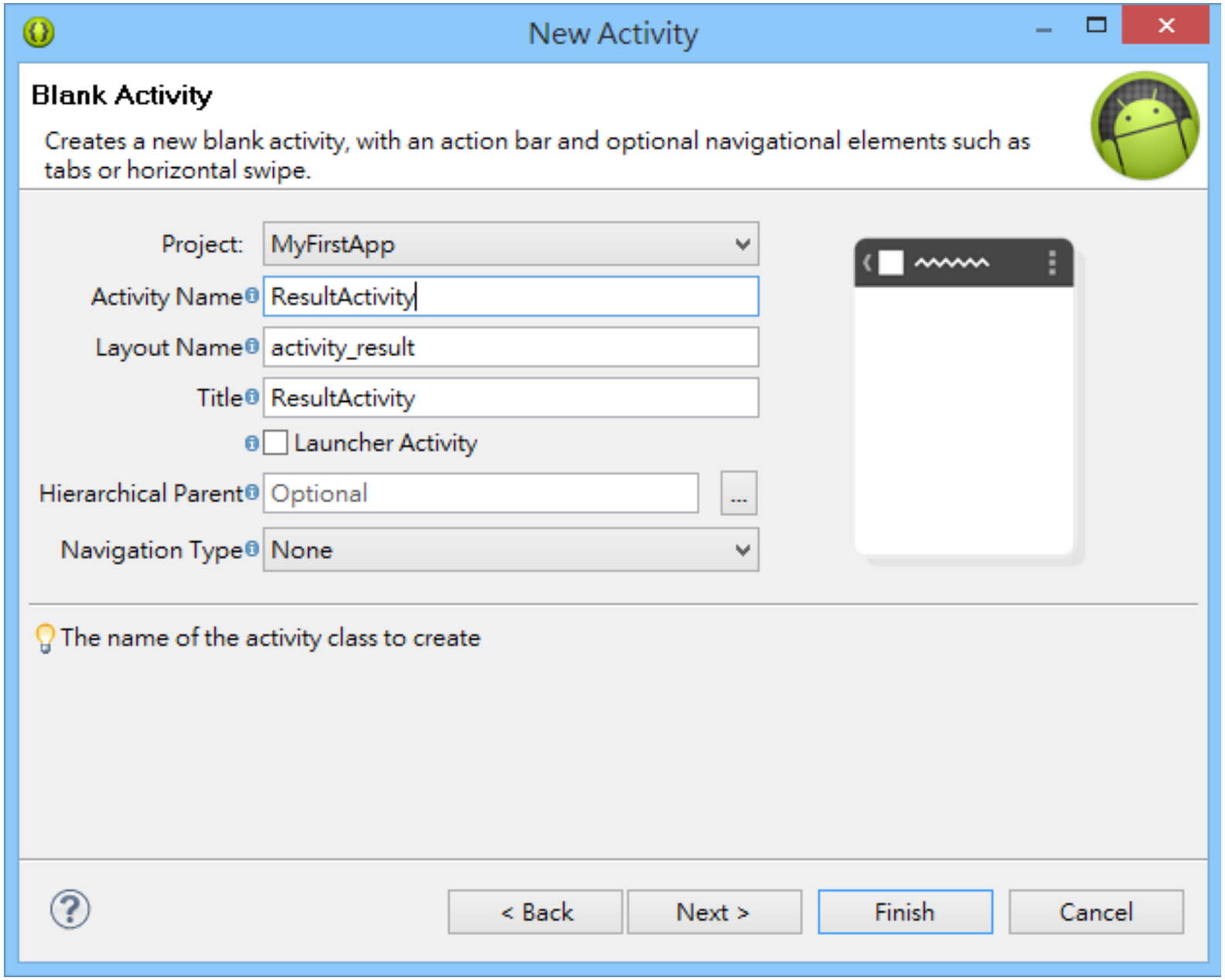
Create a new activity



Select “Android Activity”



Fill in the information of new activity



Create an intent, put some extra information (the height and weight) and start a new activity

```
public void submitInput(View view){
    Intent intent = new Intent(this, ResultActivity.class);
    intent.putExtra("height", heightInput.getText().toString());
    intent.putExtra("weight", weightInput.getText().toString());
    startActivity(intent);
}
```

Outline

- Android Fundamental
- User Interface Overview
- Environment Set Up
- Your First App
- Today’s Mission
- Reference

Today’s Mission

- Complete the Simple BMI app
  - Press “Submit” and jump to the Result Activity
    - Showing the BMI
    - Showing the “You are fat/healthy/thin” comment
    - Showing the corresponding image for fat, healthy, thin
  - Press “Reset” and clean the text field for height and weight

Hints

- Use getIntent() method of Activity to get the Intent
- Use getStringExtra(String s) method of Intent to get the data
- Use TextView to show the result
  - setText(String s) method for setting the text
- Use ImageView to show the image
  - Put the images into the res/drawable-xhdpi folder
  - setImageResource(R.drawable.XXX) can set the image resource to a ImageView

Reference

- Google Developer Guide