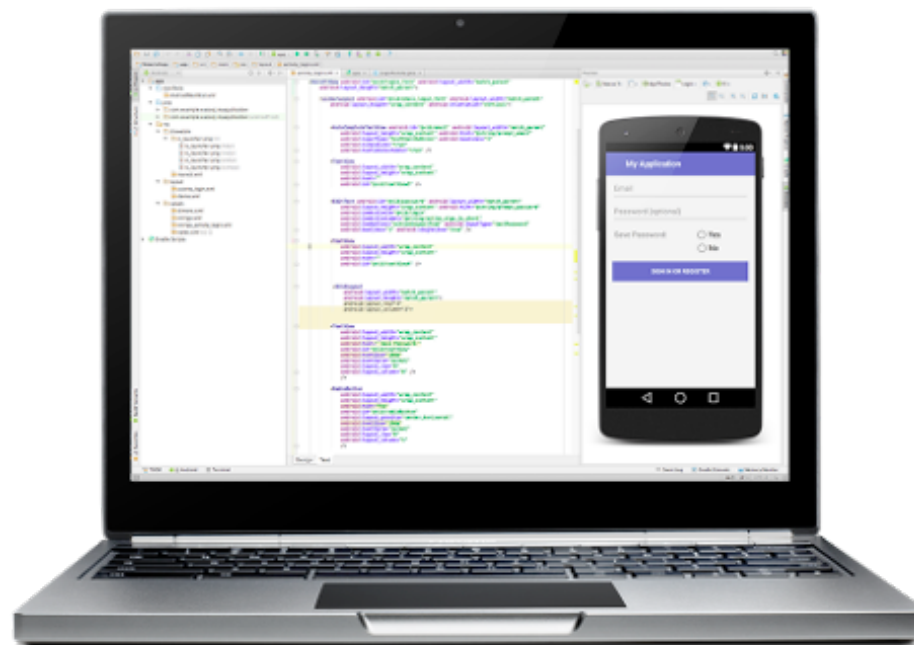


Android入門講座

はじめてのアプリ作成

2019年3月8日



アプリ作成に必要なファイルの準備

ダウンロードした **work.zip** を、任意のディレクトリ（たとえばデスクトップ）に展開します。
（**work** ディレクトリができあがります。）

work ディレクトリ内のファイル構成は以下のとおりです。

- ・ **lesson**
当講習会で開発するアプリのソースコードを格納しているフォルダ
- ・ **Android入門講座（はじめてのアプリ作成）.pdf**
講習会前半で利用する資料（当資料）
- ・ **Android入門講座（センサーを活用したアプリ作成）.pdf**
講習会後半で利用する資料

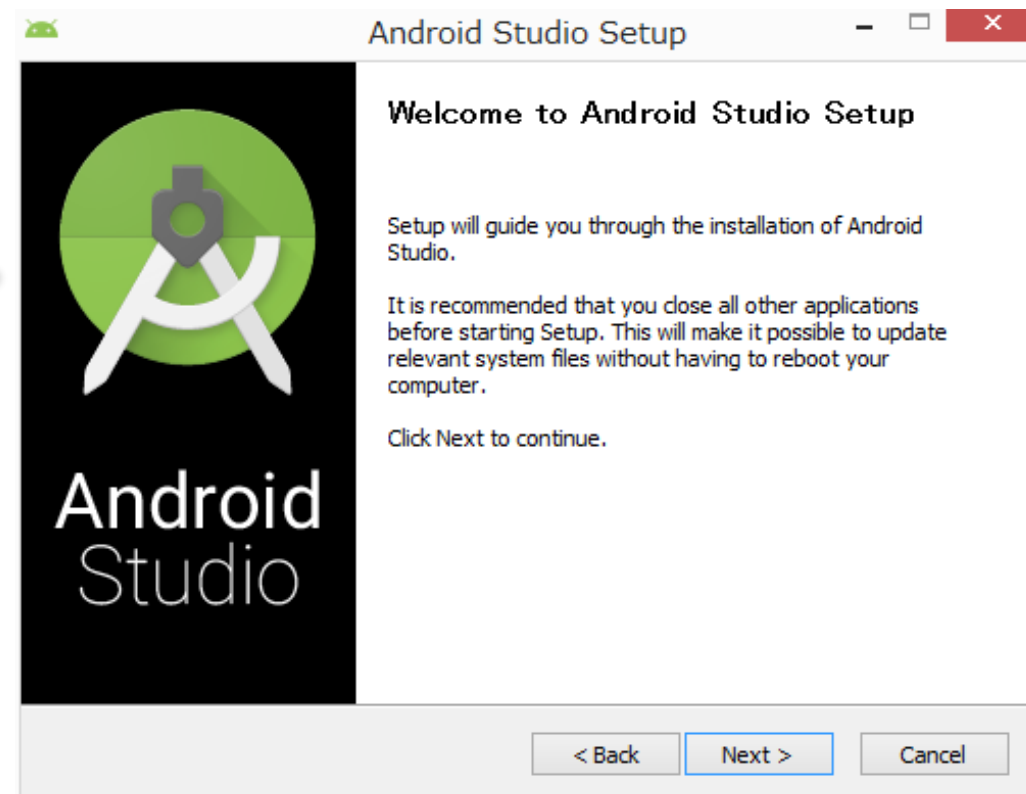
参考：Android Studio のインストール（Windows 64bit の場合。その1）

Android Studio 公式ページ（ <https://developer.android.com/studio/?hl=ja> ）

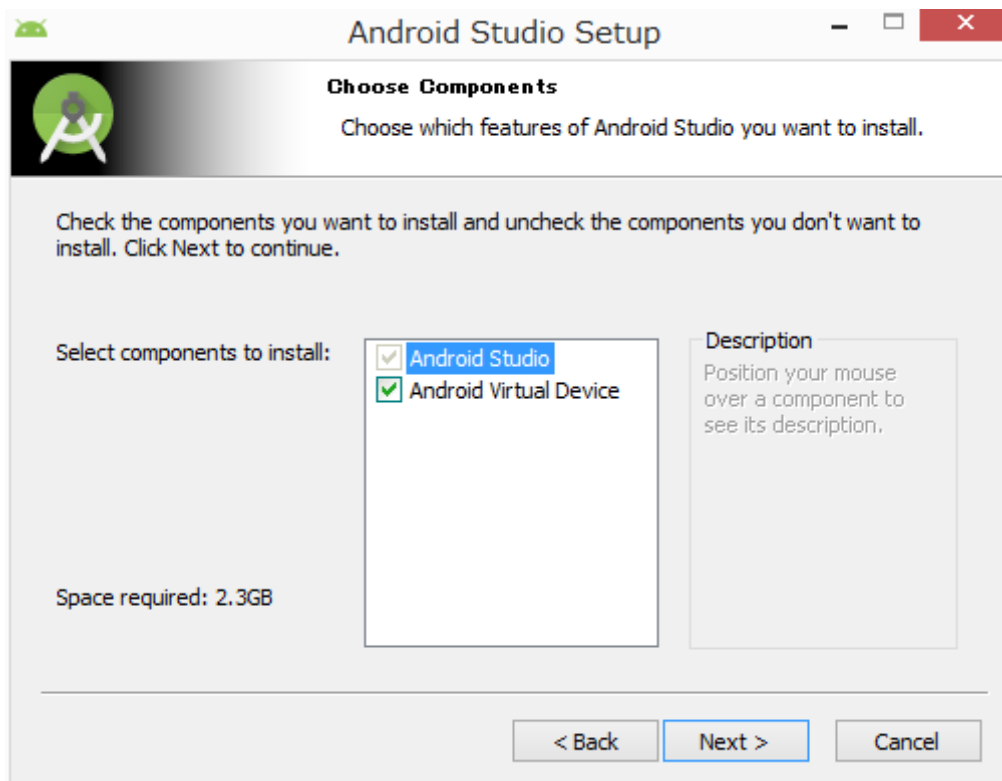
から取得したインストーラファイル（ 2019 年 2月時点では `android-studio-ide-181.5056338-windows.exe` が最新）
をダブルクリックします。

インストールには管理者権限が必要となります。

そのまま「Next」をクリックします。



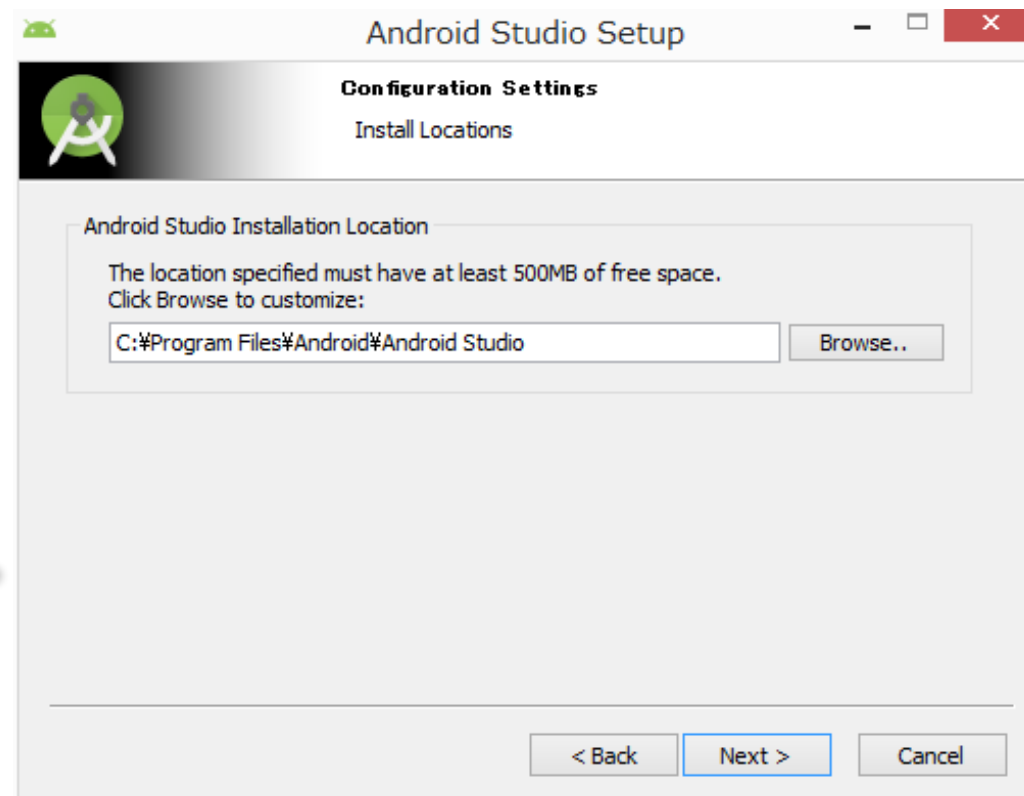
参考：Android Studio のインストール（Windows 64bit の場合。その2）



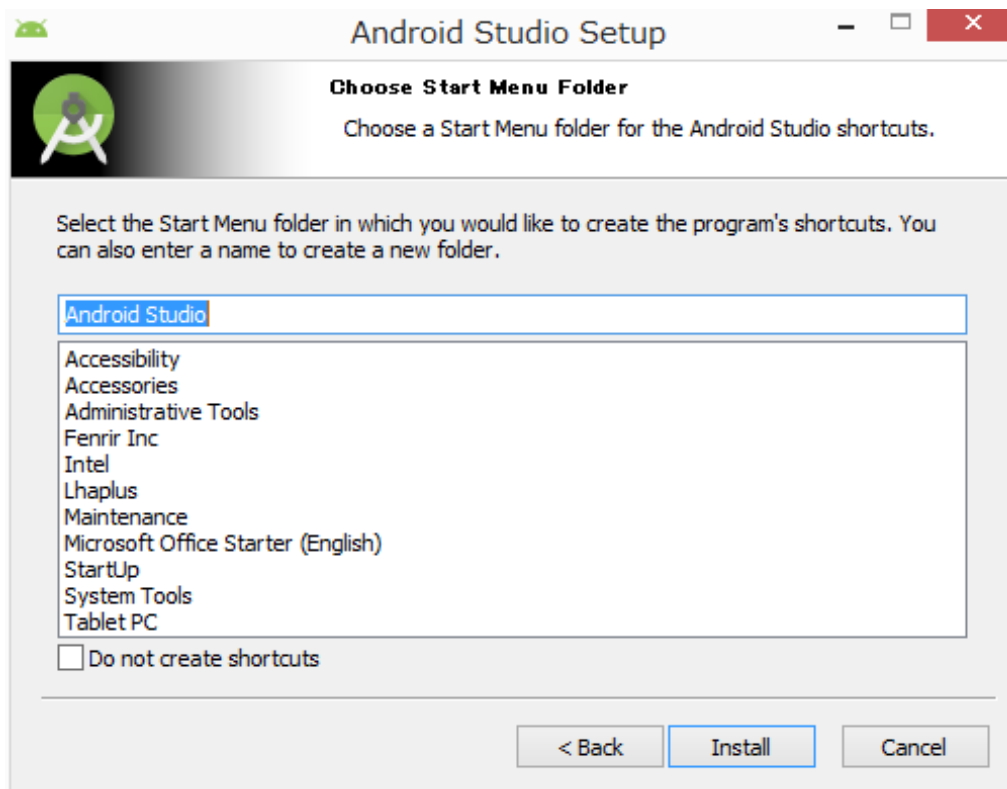
そのまま「Next」をクリックします。

「Android Virtual Device」（Android スマートフォンのエミュレータ）は、利用しない場合はチェックをはずしてもよいでしょう。

その後、「Next」をクリックします。

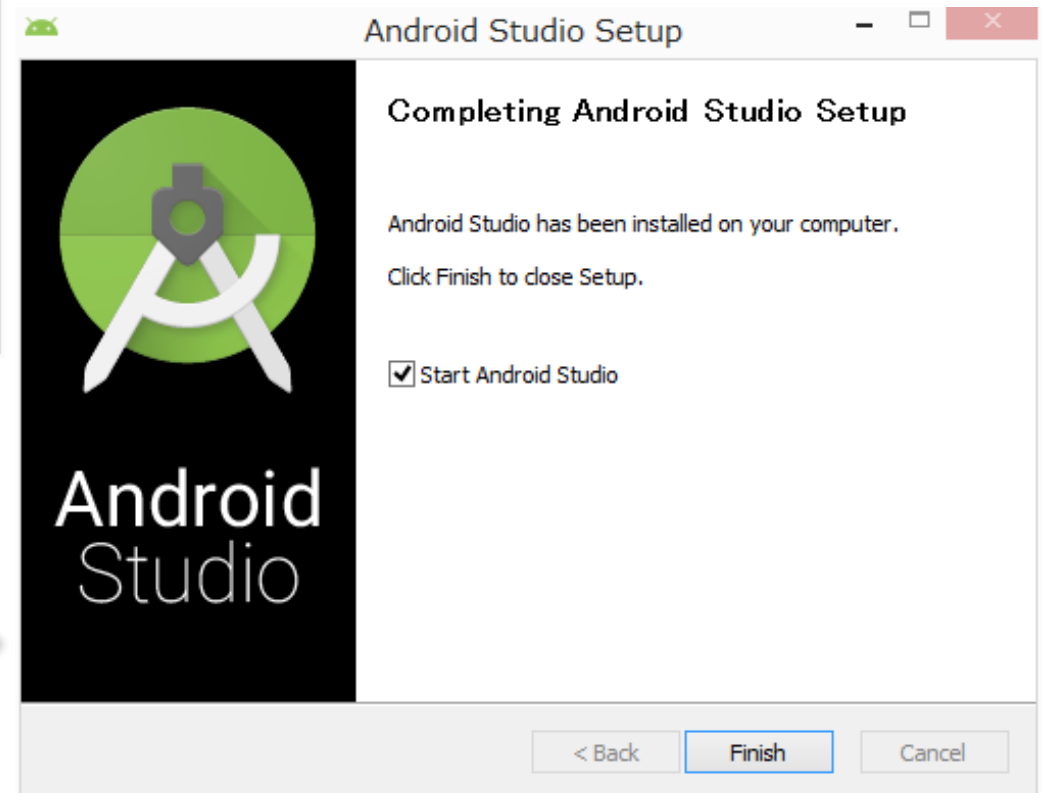


参考：Android Studio のインストール（Windows 64bit の場合。その3）



そのまま「Install」をクリックします。

しばらくしてインストールが完了したのち、
「Finish」をクリックして、Android Studio を起動します。

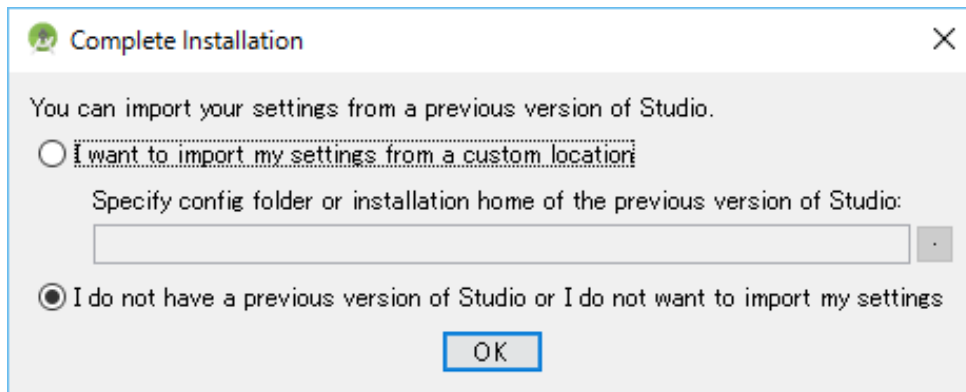


参考：Android Studio のインストール（Mac の場合。）

Android Studio 公式ページ（ <https://developer.android.com/studio/?hl=ja> ）
から取得したインストーラファイル（ 2019 年 2月時点では **android-studio-ide-181.5056338-mac.dmg** が最新）
をダブルクリックします。

インストールの手順は、Windows 64bit の場合と同様です。

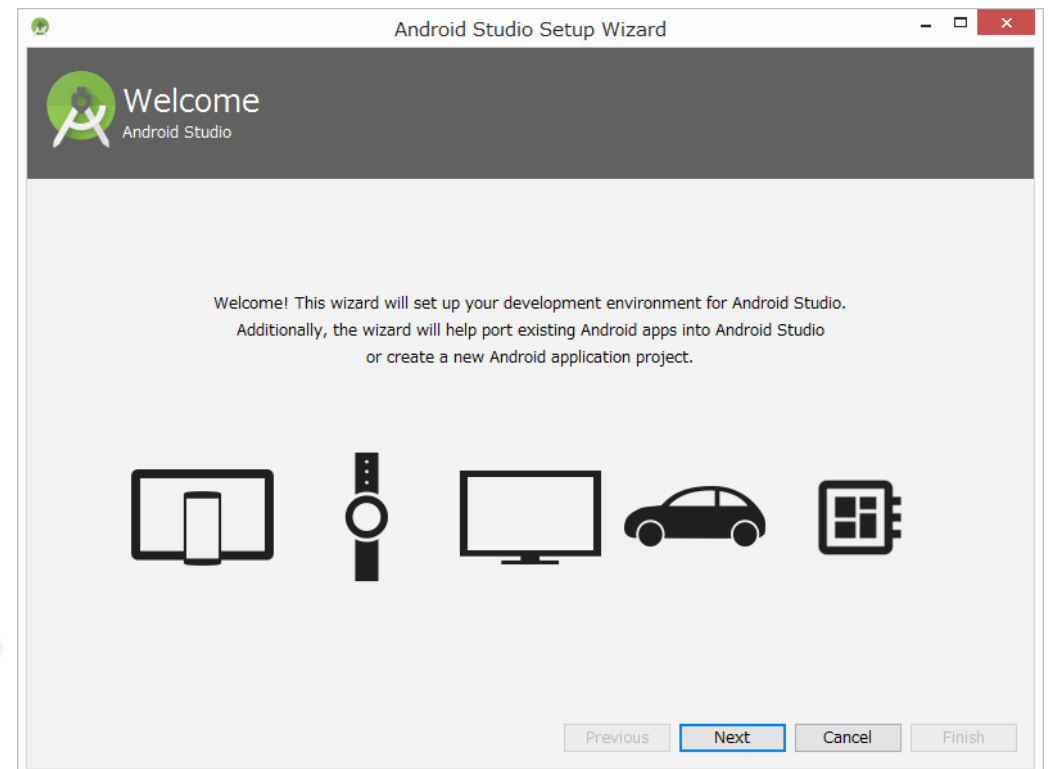
Android Studio の起動（その1）



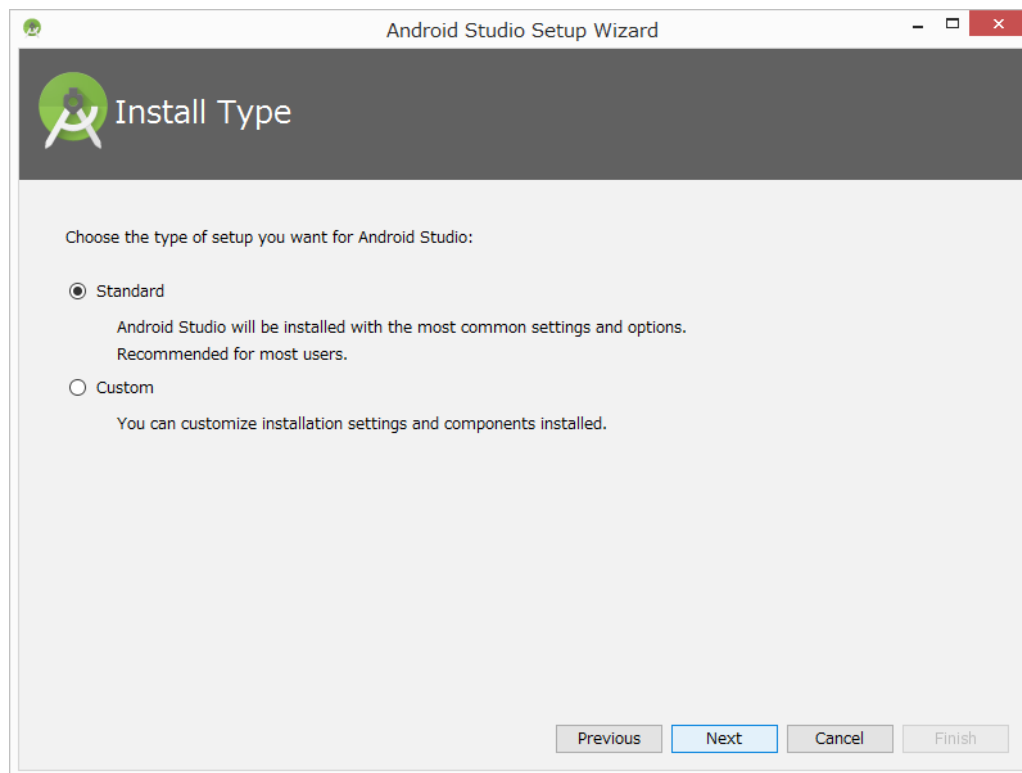
Android Studio の初回起動時に、このようなダイアログボックスが表示されましたら、今回は下側の

「I do not have a previous . . .」
をチェックして「OK」ボタンをクリックします。

そのまま「Next」をクリックします。

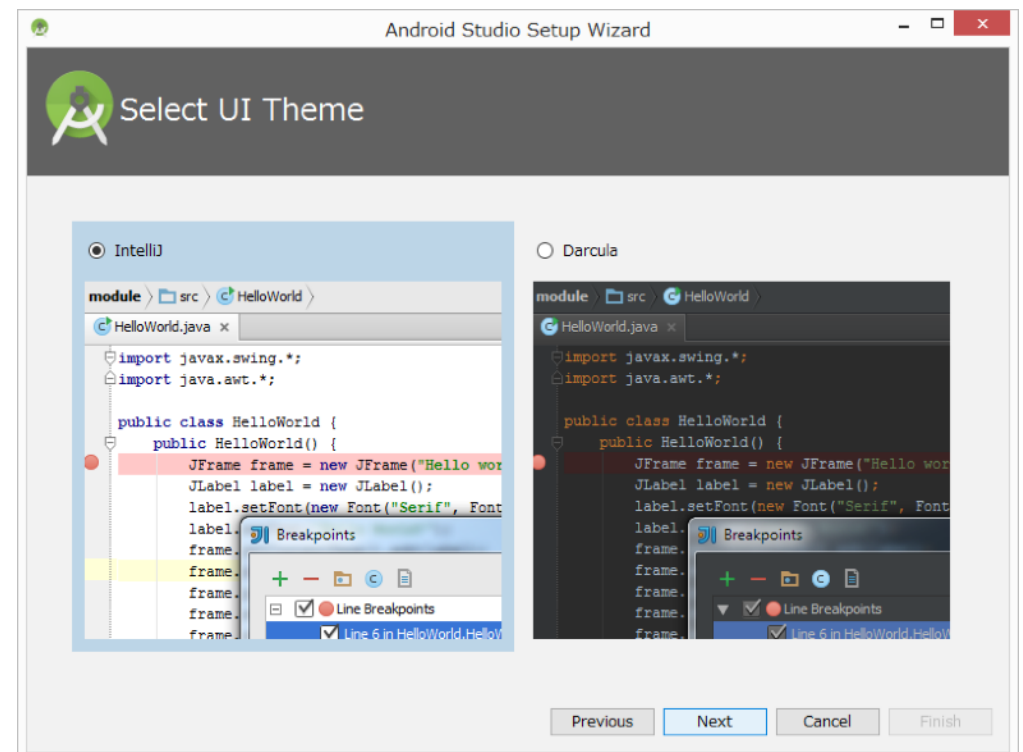


Android Studio の起動 (その2)

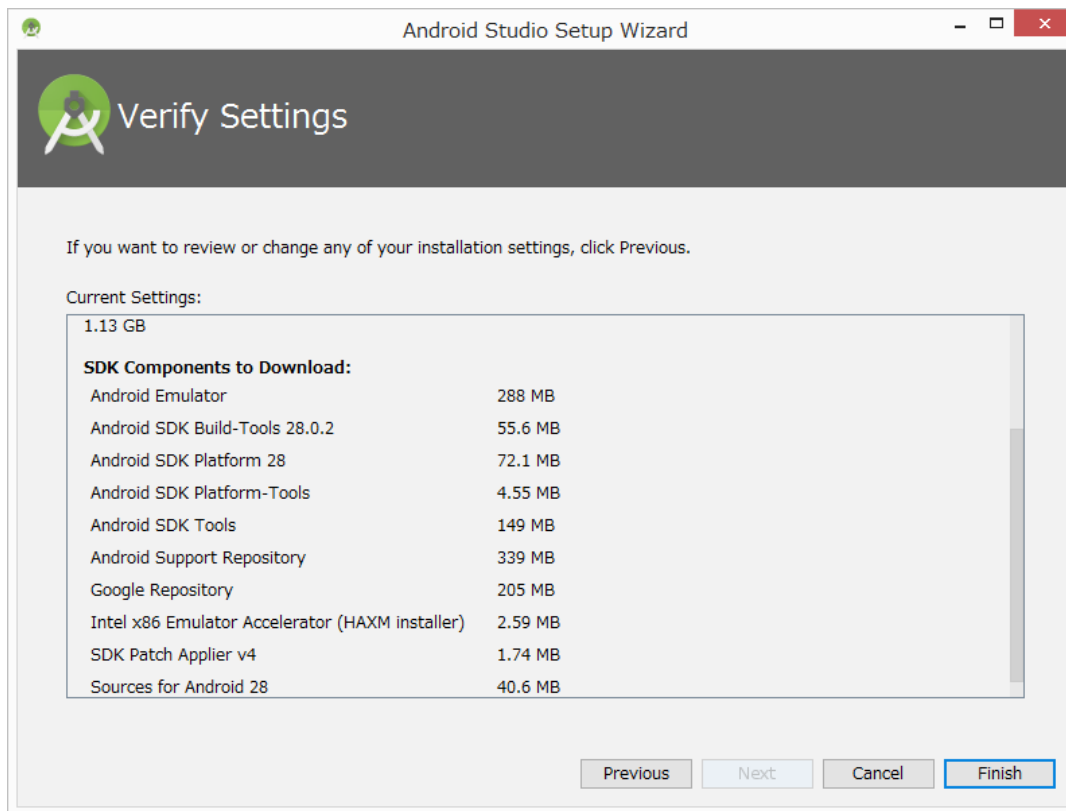


そのまま「Next」をクリックします。

UI テーマはお好みで選択して、
「Next」をクリックします。

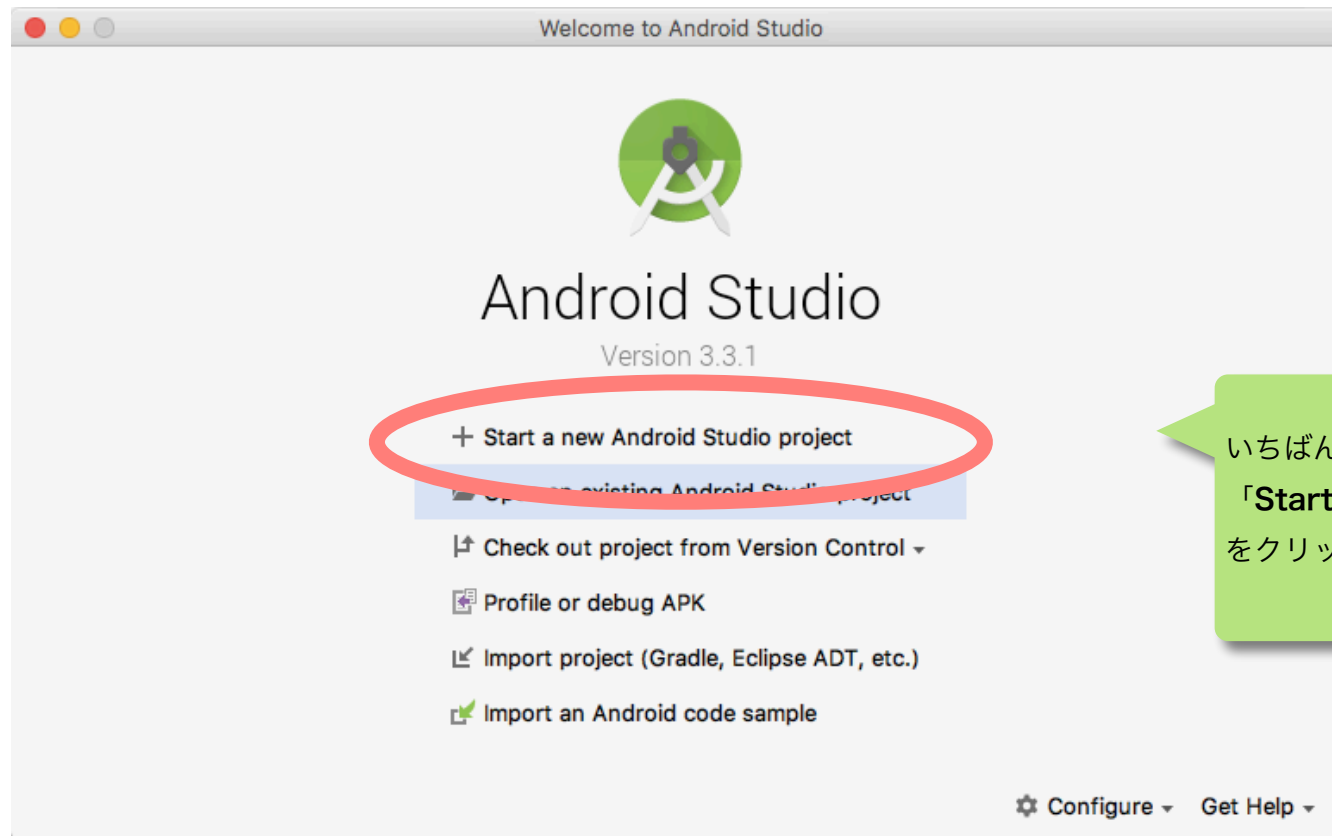


Android Studio の起動 (その3)



「Finish」をクリックして、初期設定処理を完了します。

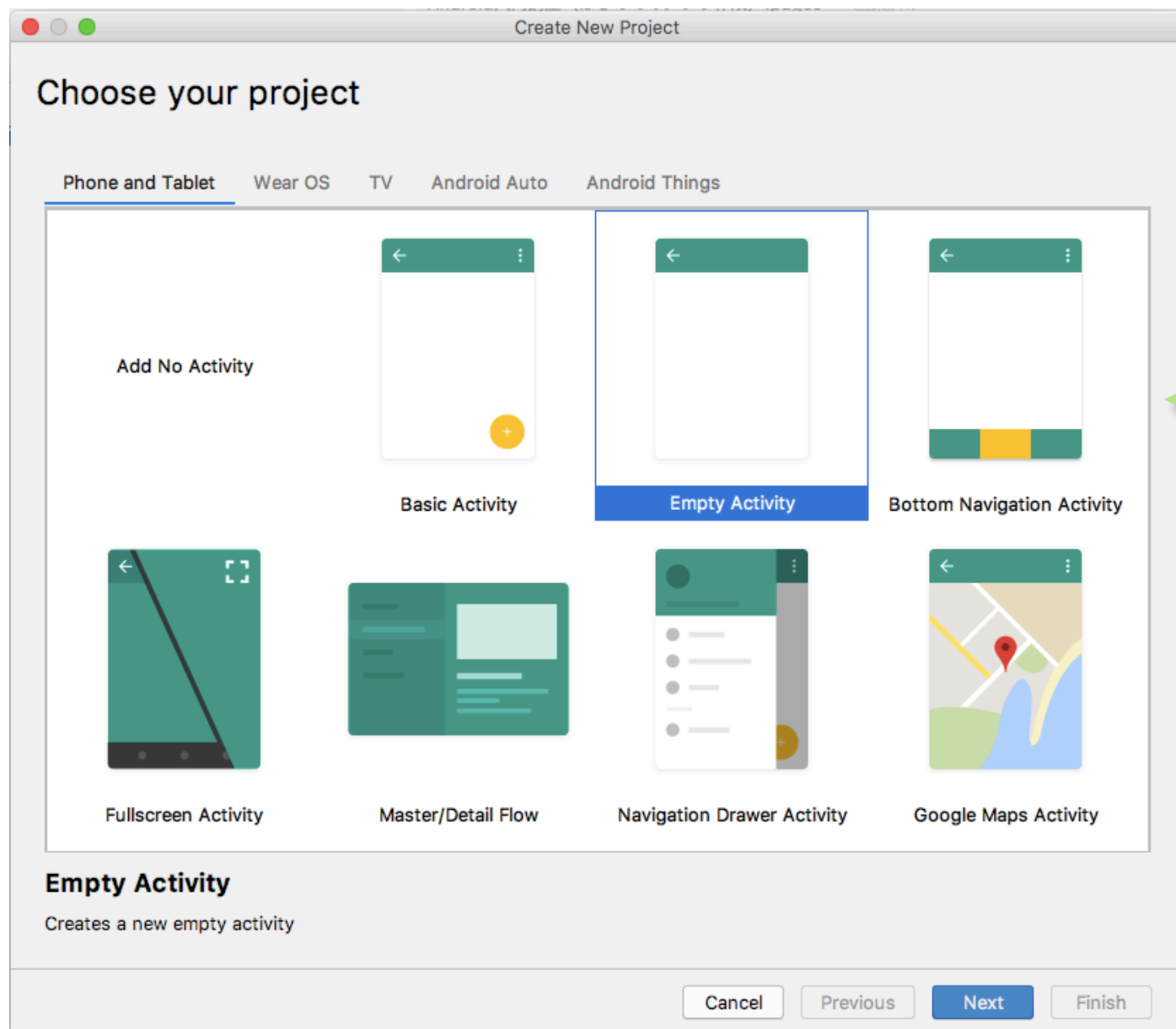
はじめてのアプリ作成（その1）



いちばん上の

「**Start a new Android Studio project**」
をクリックします。

はじめてのアプリ作成（その2）



「Empty Activity」を選択して「Next」をクリックします。

はじめてのアプリ作成（その3）

Create New Project

Configure your project

Name
My Application

Package name
jp.ac.yuge.myapplication

Save location
/Users/[user]/Desktop/work/lesson/MyApplication

Language
Java

Minimum API level
API 19: Android 4.4 (KitKat)

i Your app will run on approximately **95.3%** of devices.
[Help me choose](#)

☐ This project will support instant apps
☐ Use AndroidX artifacts

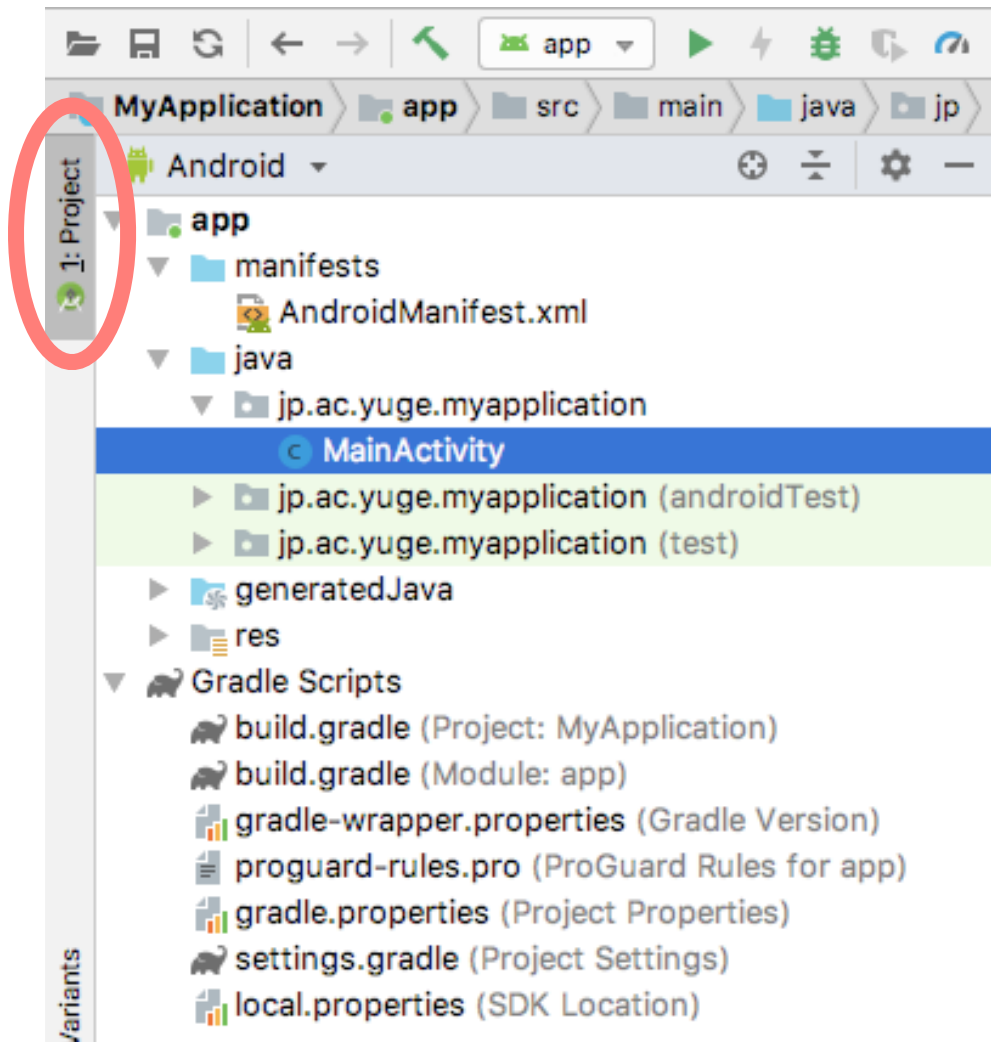
Empty Activity

Creates a new empty activity

Cancel Previous Next Finish

「Name」に「**My Application**」、
「Package name」に、たとえば
「**jp.ac.yuge.myapplication**」、
「Save location」に、たとえばデスクトップの
「**work/MyApplication**」フォルダを指定します。
その状態で「**Finish**」をクリックします。

はじめてのアプリ作成（その4）



ウィンドウ左側に、Android Studio が自動生成したファイルが表示されます。

（表示されていない場合は、ウィンドウ左端の「Project」タブをクリックします。）

はじめてのアプリ 実行（その1。スマートフォンの設定）



スマートフォンの「設定」アプリを起動して、「端末情報」をタップします。
表示された項目の中の「ビルド番号」を7回連続してタップして、デベロッパー（開発者）モードに設定します。

「設定」アプリのトップ画面に戻って、「開発者向けオプション」が表示されていれば、デベロッパーモードの設定完了です。



参考：HUAWEI 製スマートフォンの設定

HUAWEI nova lite2 など、HUAWEI 製のスマートフォンをアプリ検証用には、スマートフォン側で特別な操作が必要になることがあるようです。

具体的には、

- 1) 電話アプリで *****#2846579#***** をダイヤルして、隠しメニューを表示。
- 2) 表示された隠しメニューから、アプリ検証用に必要な設定を行う。

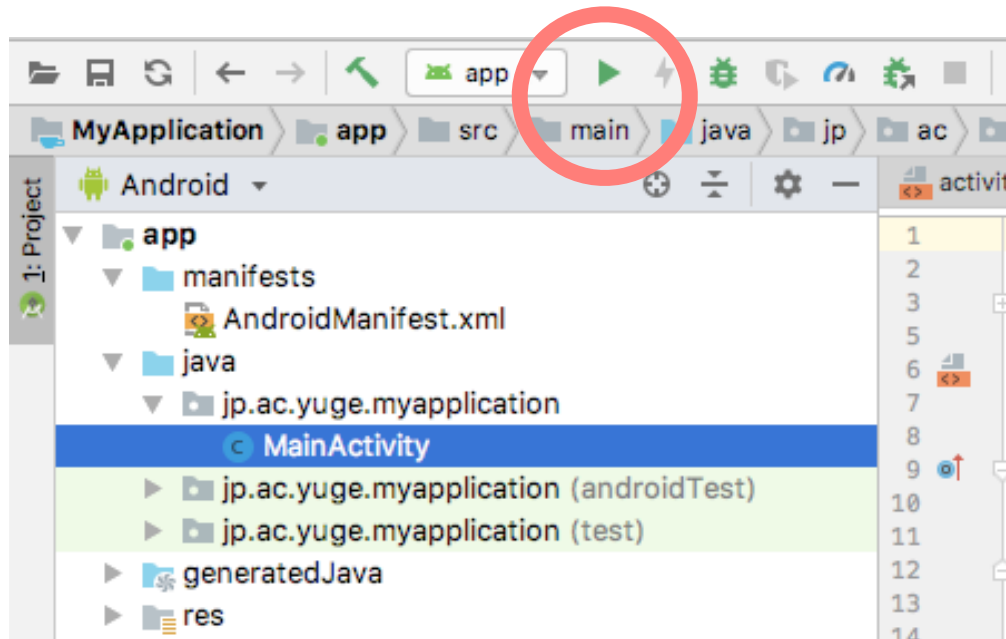
という手順になるようです。

詳細については、以下ページをご参照ください。

HUAWEI nova lite2 でUSBデバッグを有効にする方法

<http://takeshich.hatenablog.com/entry/2018/07/21/162444>

はじめてのアプリ 実行（その2。アプリケーションの実行）

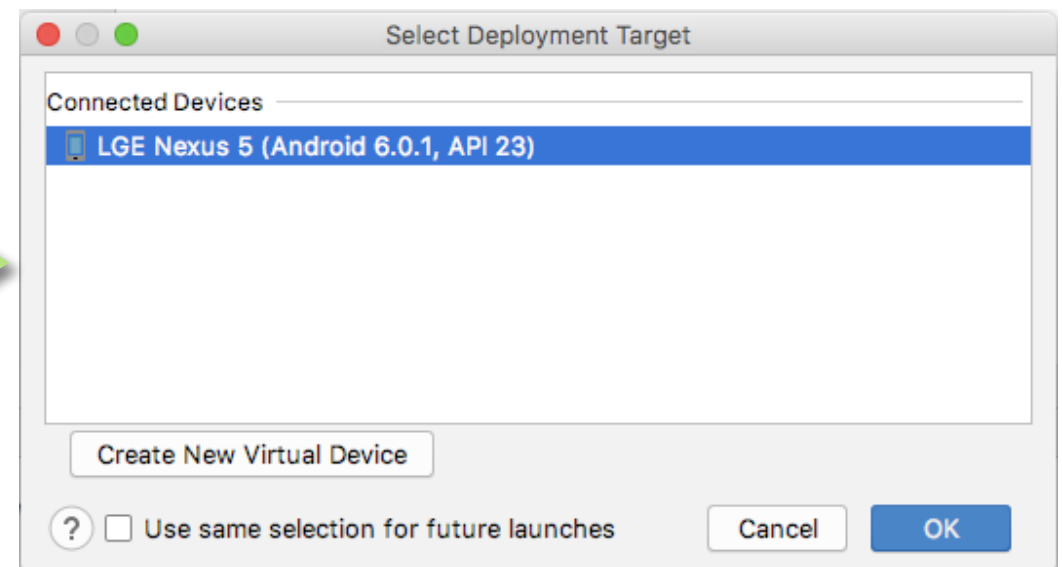


スマートフォンを PC に USB ケーブルで接続します。
その状態で「Run」ボタン（緑色の三角形のボタン）をクリックします。

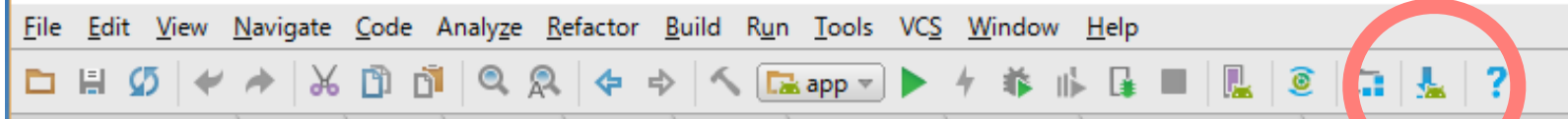
スマートフォンが正しく接続されていると、このように、
ダイアログボックスにそのスマートフォンのモデル名が表示
されます。

（スマートフォン側で「このコンピュータを信頼しますか」
のようなダイアログボックスが表示された場合は「信頼」
を選択します。）

「OK」ボタンをクリックして、アプリを実行します。

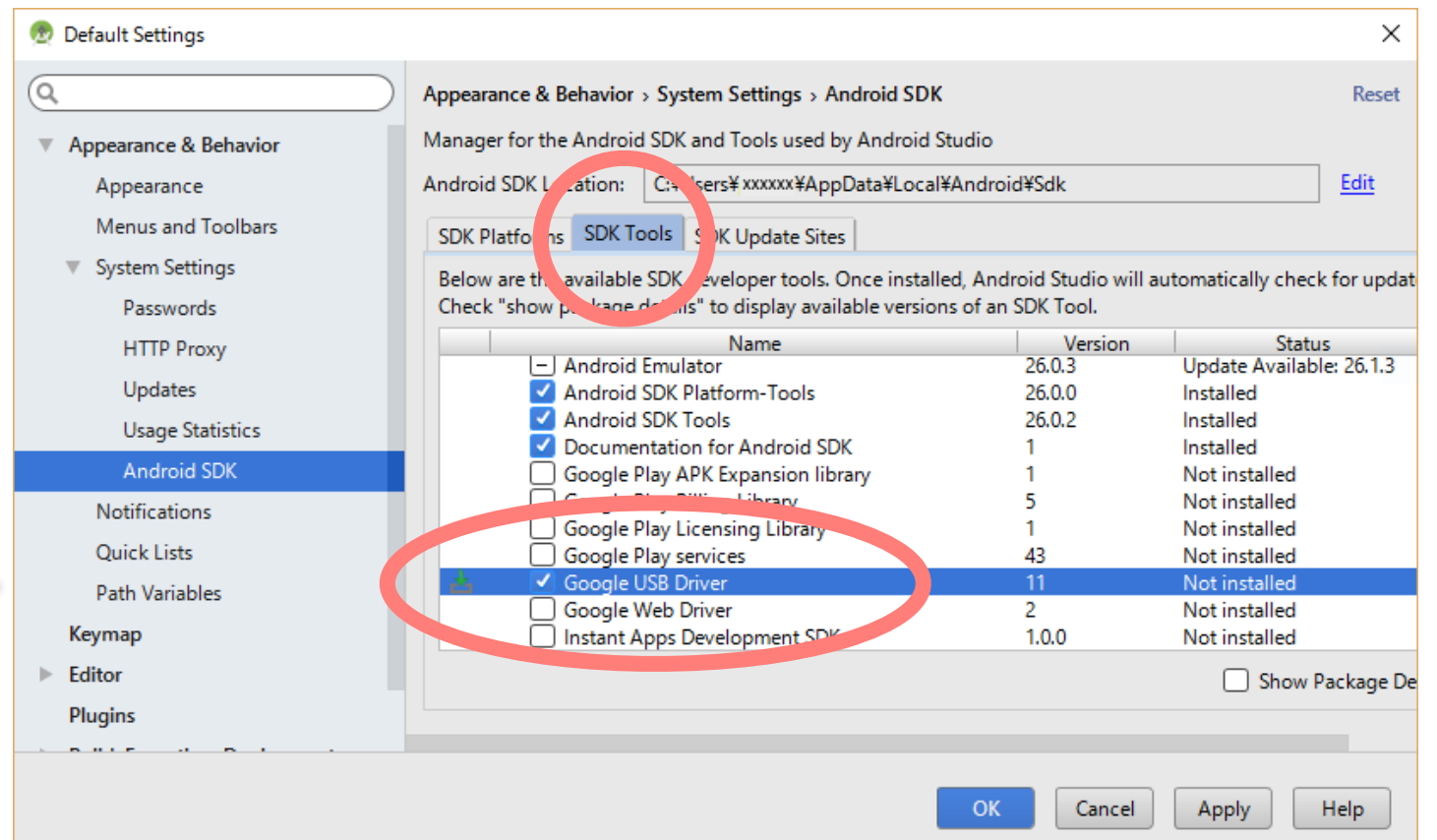


参考：はじめてのアプリ 実行（Google USB Driverインストール（Windowsのみ））

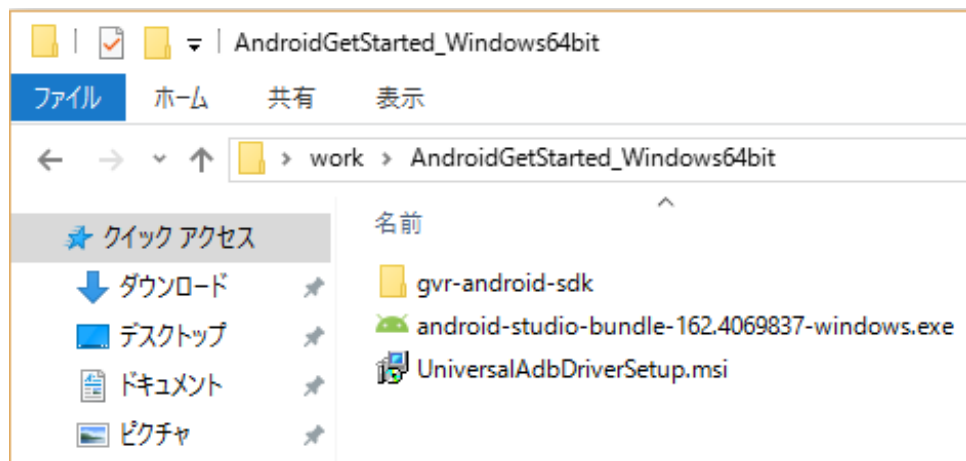


Windows 環境でスマートフォンが認識できない場合、
まずは「Google USB Driver」のインストールを試してみます。
メニューバーから「SDK Manager」のボタンをクリックします。

「SDK Tools」タブを選択して、
その下側に表示される一覧から
「Google USB Driver」
にチェックをつけて、
「OK」ボタンをクリックします。



参考：はじめてのアプリ 実行（Universal ADB Driverインストール（Windowsのみ））



Windows 環境で、「Google USB Driver」をインストールしても、まだスマートフォンが認識されない場合、

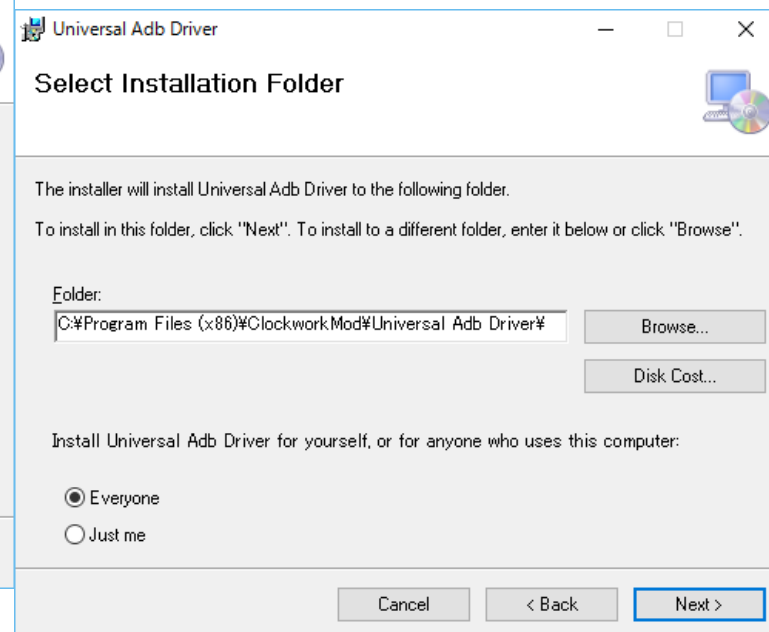
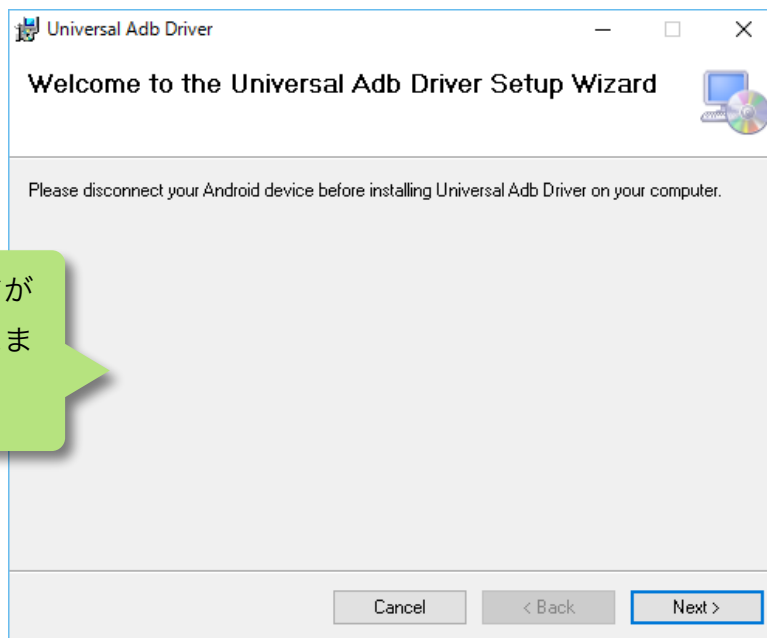
「Universal ADB Driver」のインストールを試してみます。

「AndroidGetStarted_Windows64bit」フォルダの中にある、

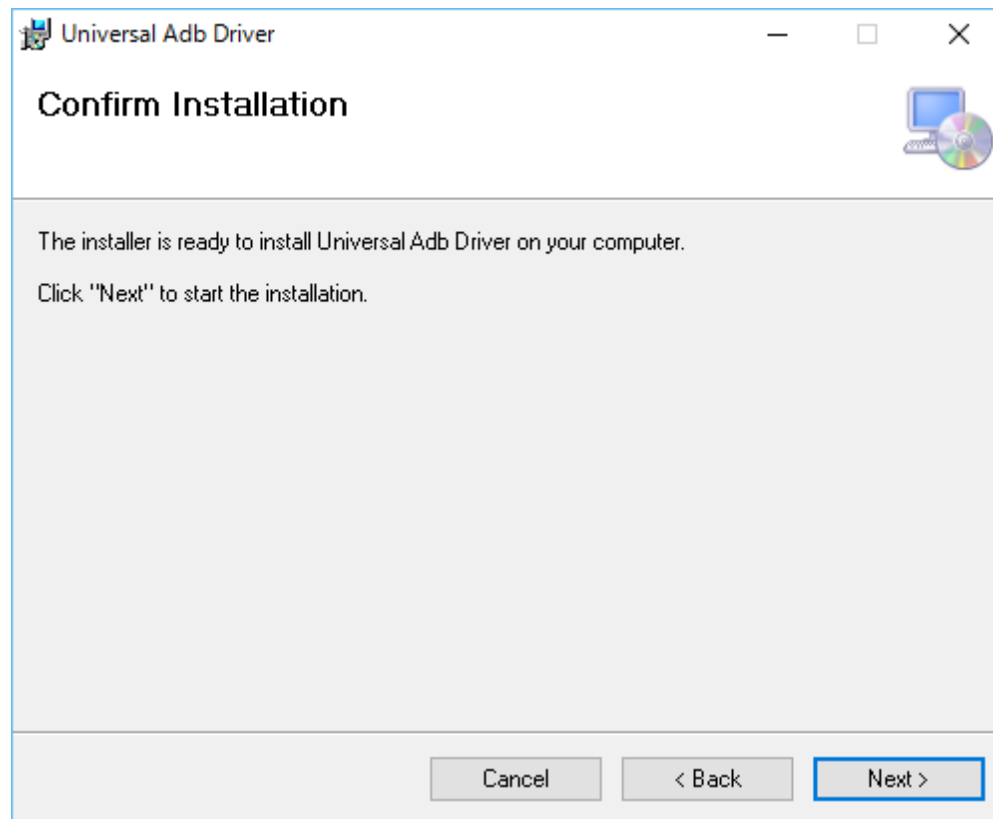
「UniversalAdbDriverSetup.msi」

をダブルクリックして実行します。（管理者権限での操作が必要です。）

このようなインストールウィザードが表示されますので、いずれもそのまま「Next」をクリックします。

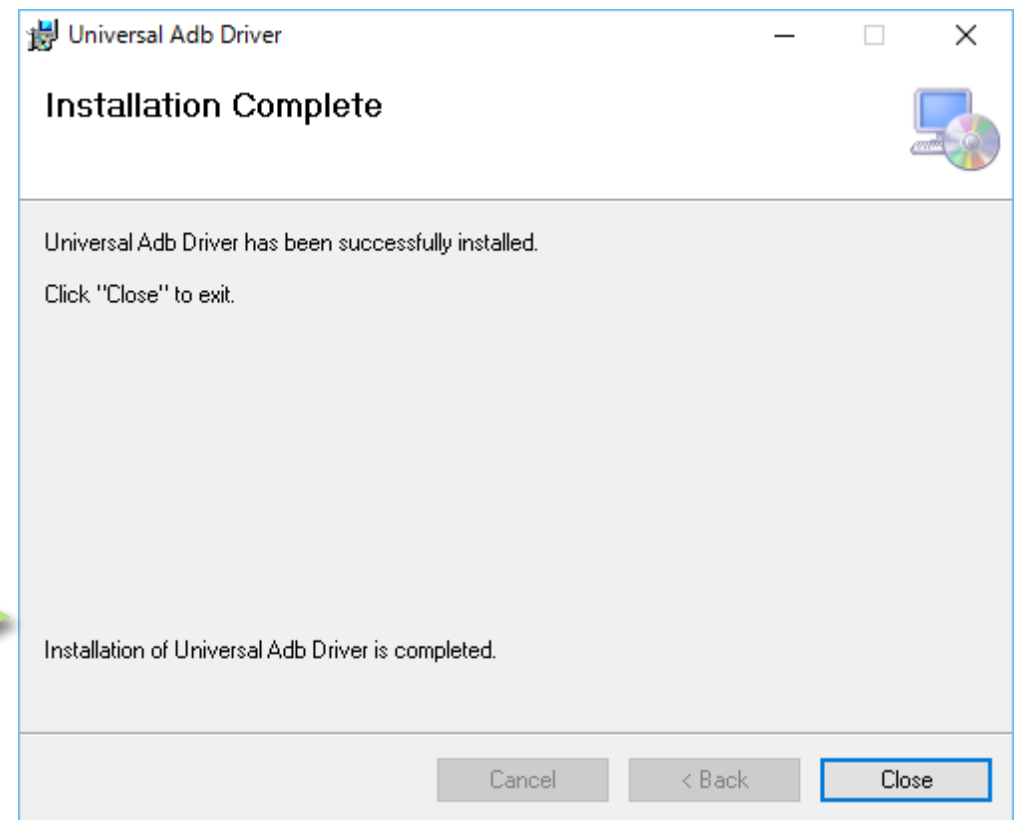


参考：はじめてのアプリ 実行（Universal Adb Driverインストール（Windowsのみ））

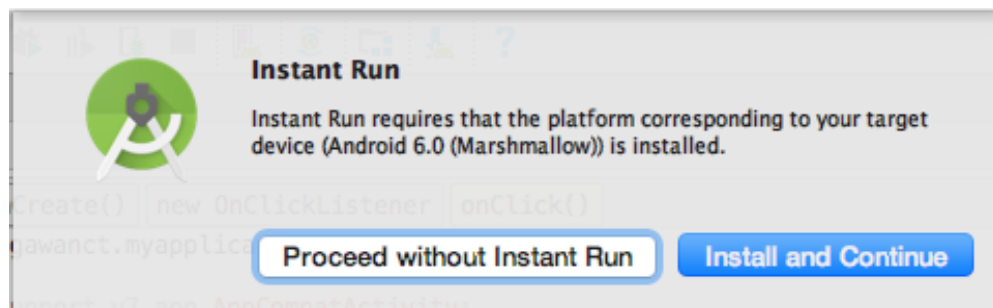


そのまま「Next」をクリックします。

インストール完了後、「Close」をクリックします。

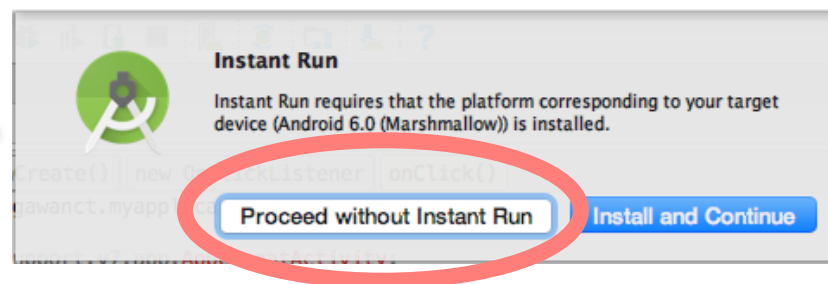


はじめてのアプリ 実行 (その3。Instant Run の設定)

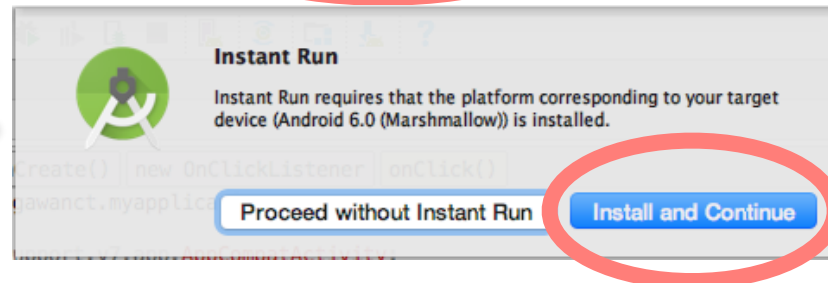


アプリ実行時にこのようなメッセージが表示される場合は、以下いずれかの対応を行います。

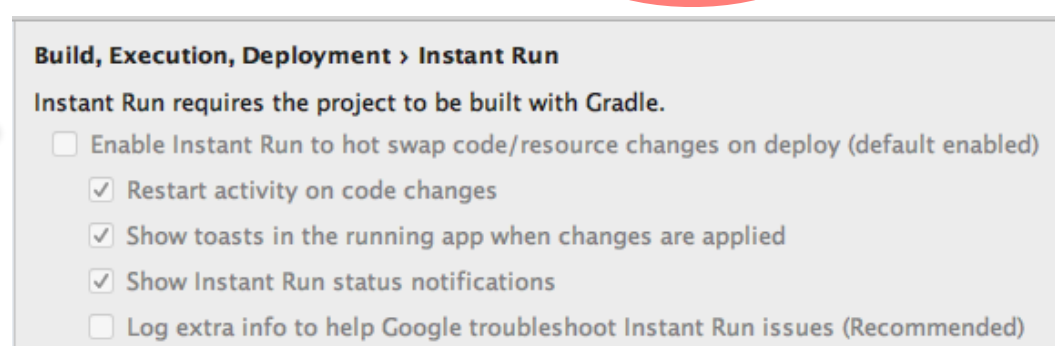
(A) 「**Proceed without Instant Run**」をクリックして、処理を続行する。(毎回、同手順を実施する必要があります。)



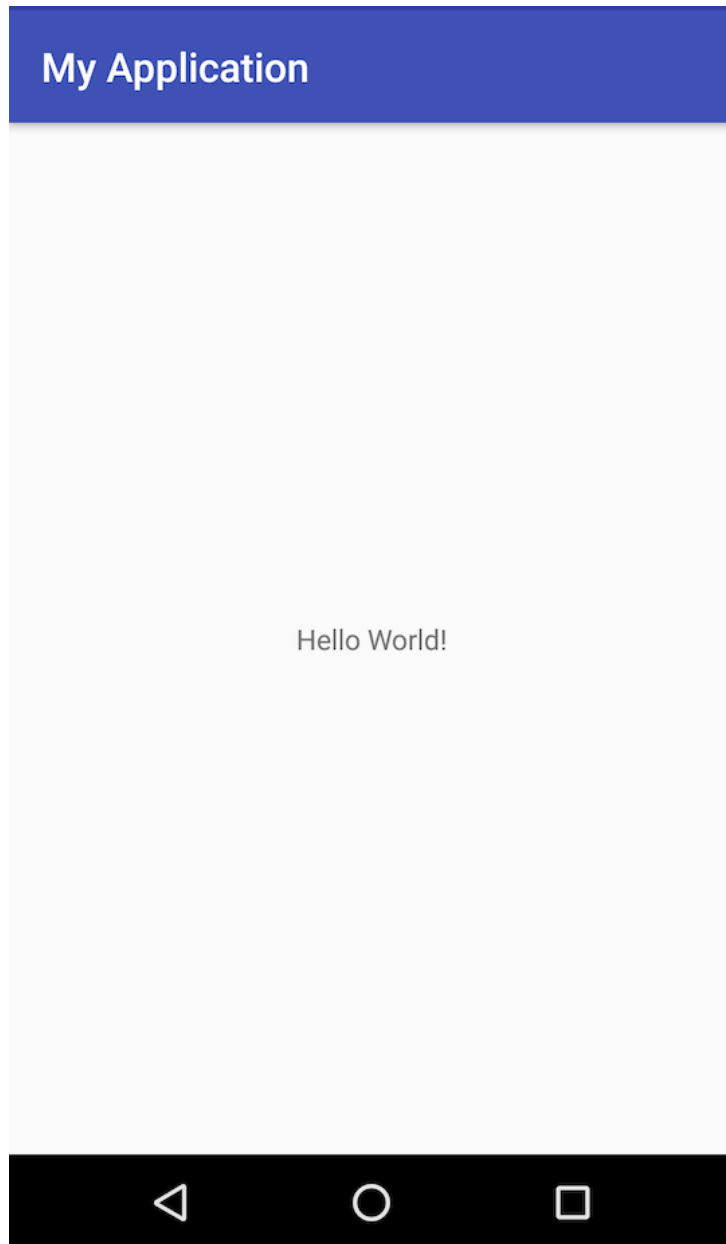
(B) 「**Install and Continue**」をクリックして、必要なプラットフォームSDKをインストールする。



(C) Android Studio の設定画面から「**Build, Execution, Deployment**」->「**Instant Run**」を選択して、Instant Run を無効にする。

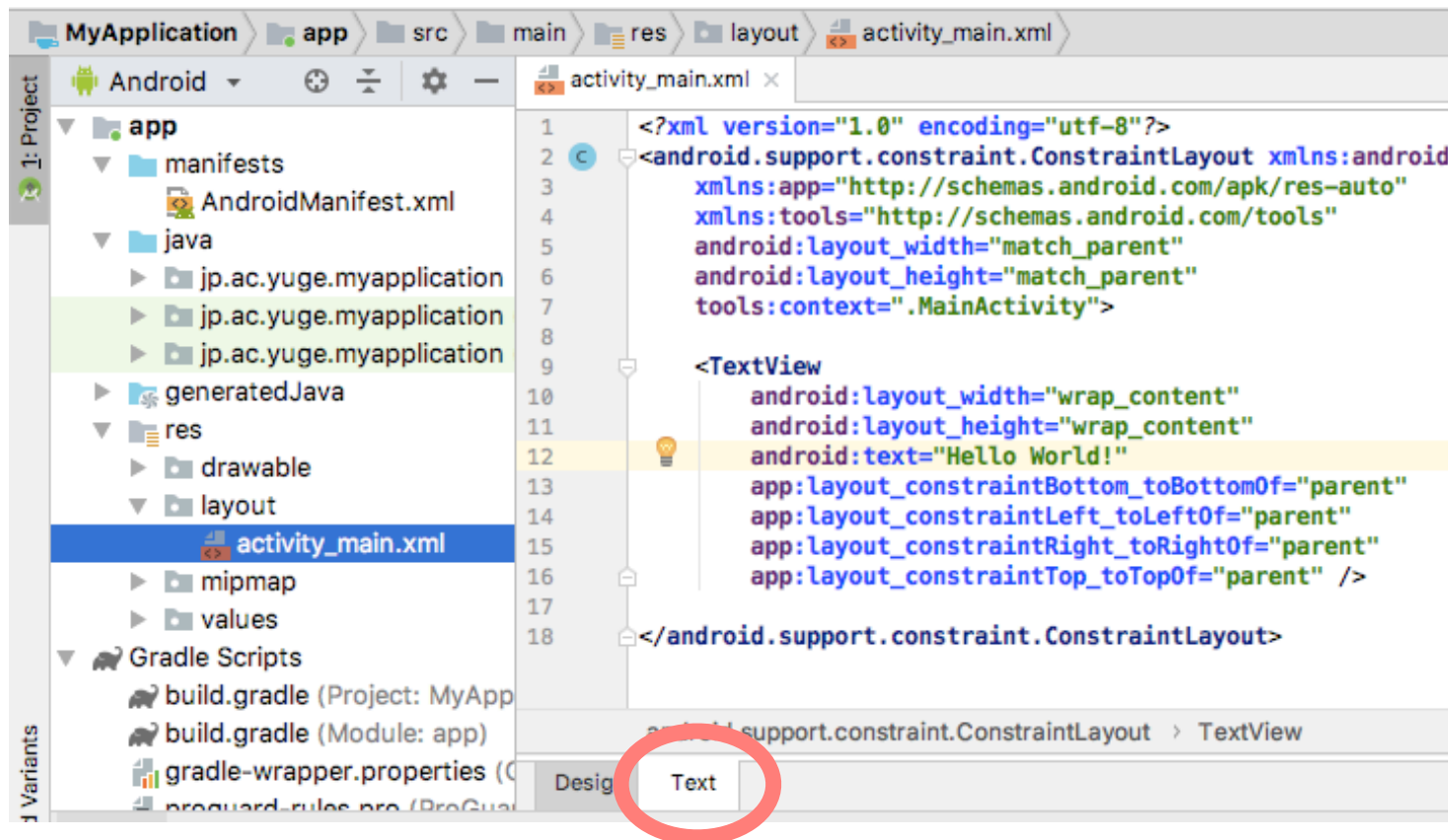


はじめてのアプリ 実行（その4）



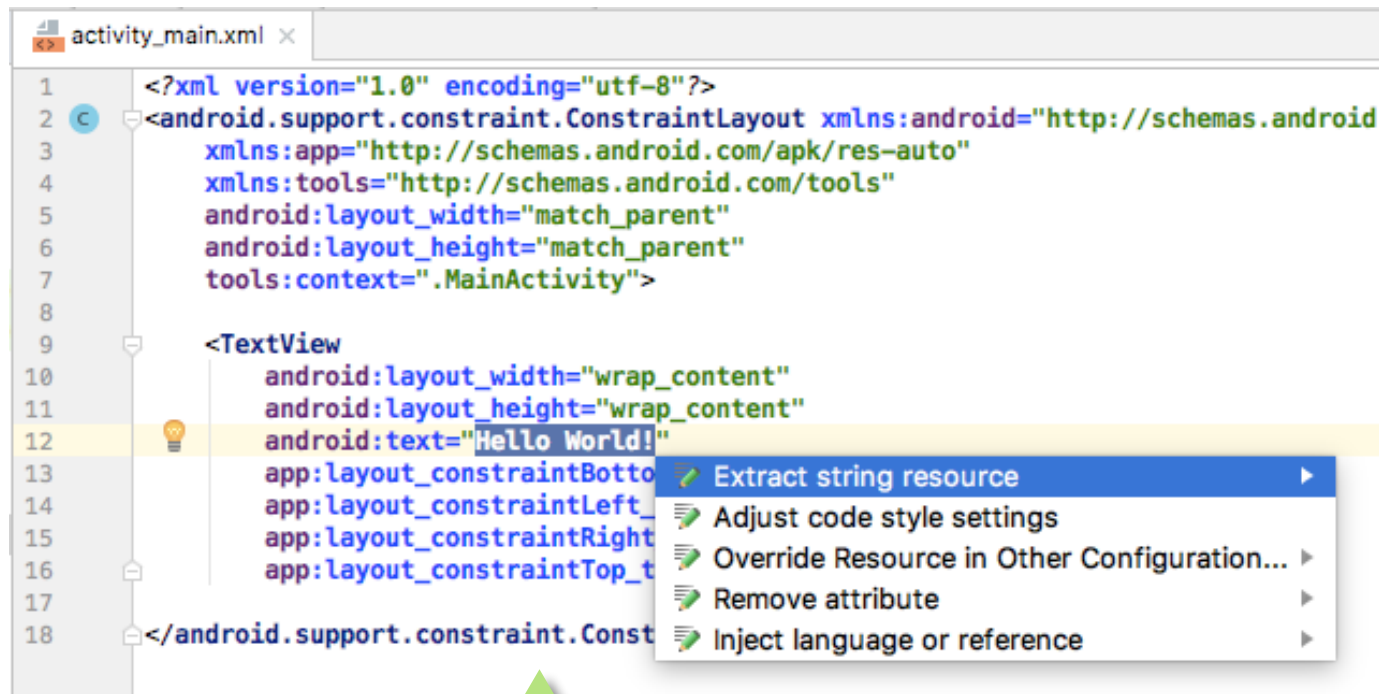
正しくアプリが実行できた場合、このように、
「My Application」というタイトルと、
「Hello World!」というメッセージのある画面が表示されます。

画面表示メッセージの変更（その1）



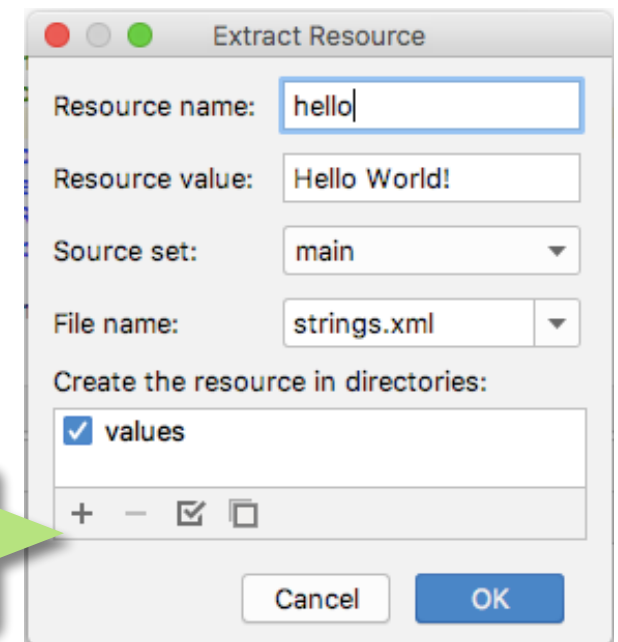
「Hello World!」というメッセージが定義されているのは、
「app」→「res」→「layout」フォルダの、「**activity_main.xml**」ファイルです。
Android Studio ツール左側の「Project」ビューから、この「**activity_main.xml**」をダブルクリックすると、右側にその内容が表示されます。
このとき、下側のタブは、最初は「Design」が選択されていますので、これを「**Text**」に切り替えます。

画面表示メッセージの変更（その2）



「**android:text=**」のあとの、ダブルクォーテーション（**”**）で囲われた部分に記入されている「**Hello World!**」の部分にカーソルを合わせて、
「**option + enter**」キー（Windows の場合は「**Alt + Enter**」）を押します。
ポップアップ表示されたダイアログボックスの中から「**Extract string resource**」を選択します。

このような画面が表示されますので、いちばん上の「**Resource name**」に「**hello**」と入力して「**OK**」をクリックします。

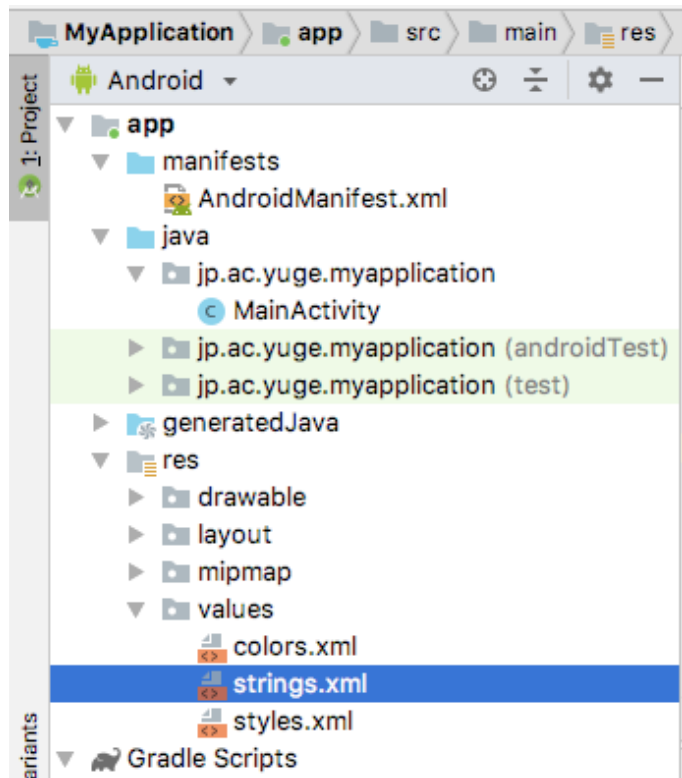


画面表示メッセージの変更（その3）

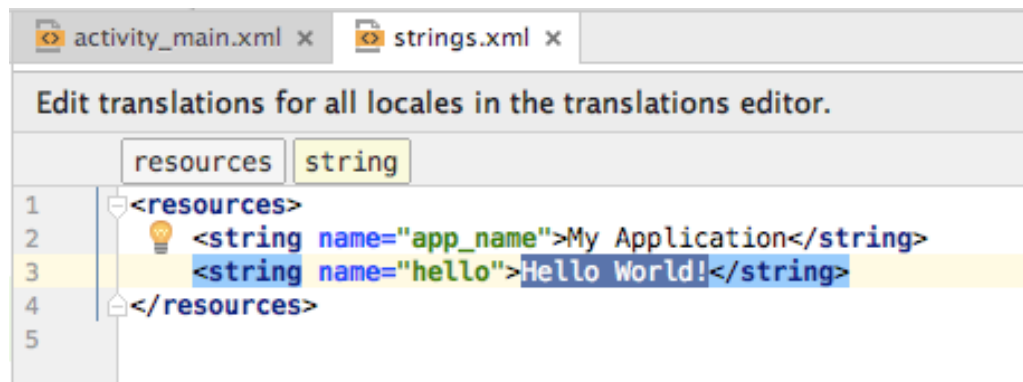
```
activity_main.xml ×
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="@string/hello"
13         app:layout_constraintBottom_toBottomOf="parent"
14         app:layout_constraintLeft_toLeftOf="parent"
15         app:layout_constraintRight_toRightOf="parent"
16         app:layout_constraintTop_toTopOf="parent" />
17
18  </android.support.constraint.ConstraintLayout>
```

「Hello World!」と表記されていた部分には、代わりに「@string/hello」と表示されます。

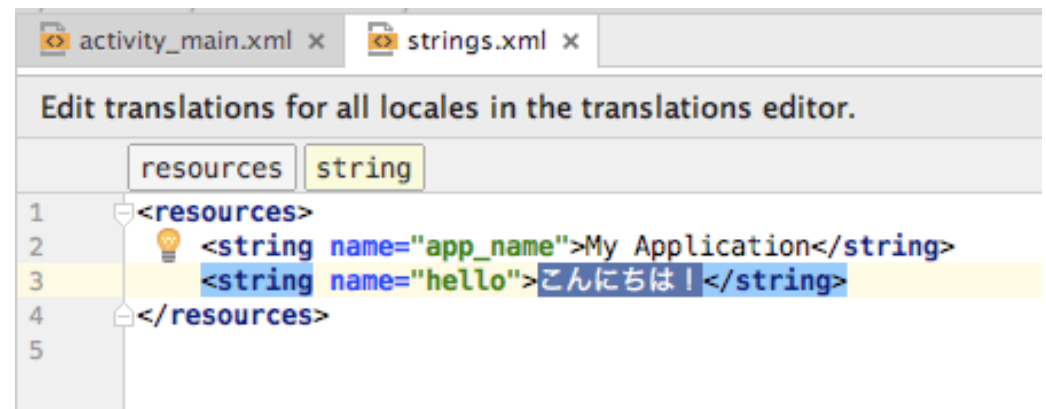
画面表示メッセージの変更（その4）



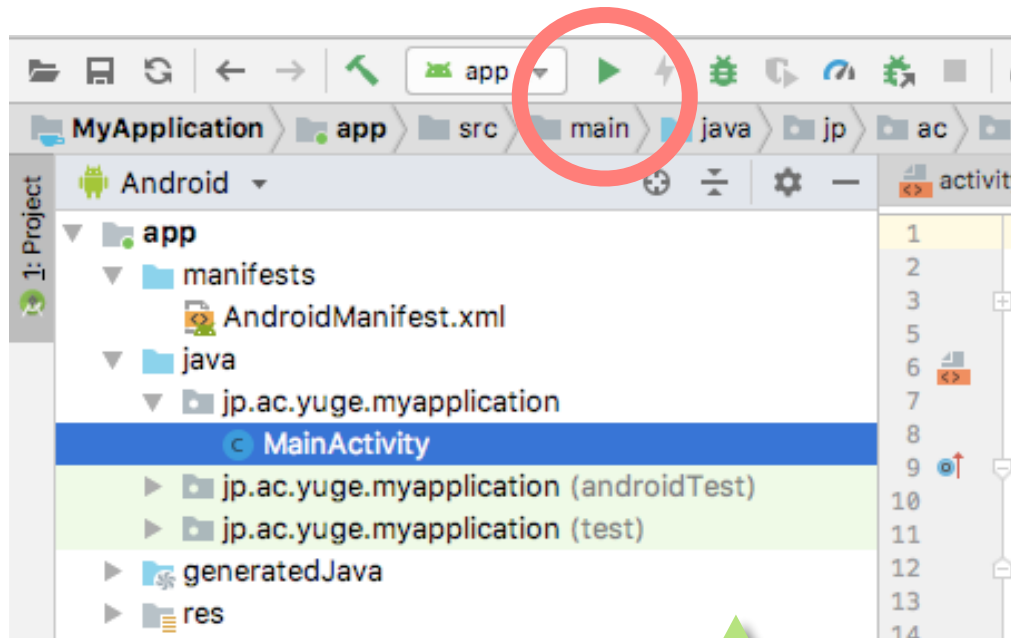
「Hello World!」というメッセージは、
「res」 → 「values」 フォルダの「**strings.xml**」のほうで、
「**<string name="hello">**」タグの要素として登録されています。



この「**strings.xml**」の中で登録されている
「Hello World!」という部分を、今回は
「**こんにちは!**」というメッセージに変更してみます。



画面表示メッセージの変更（その4）



再度、「Run」ボタン（緑色の三角形のボタン）をクリックして、アプリを実行します。

スマートフォン側で、アプリが再起動し、設定したメッセージが画面上に表示されます。



ボタンの追加（その1）

先ほどと同じ「**activity_main.xml**」ファイルに、
このように入力します。
(赤色が、新しく入力する部分です。)

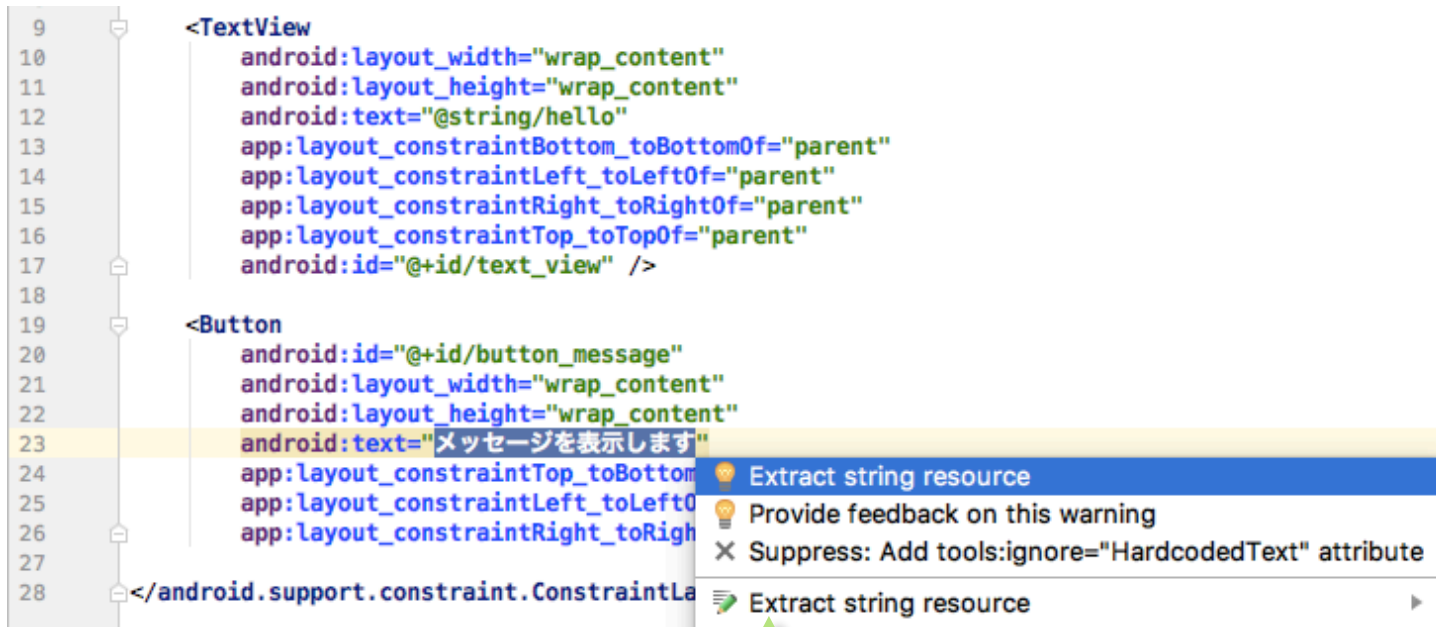
```
activity_main.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout xmlns:android="http://
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="@string/hello"
13         app:layout_constraintBottom_toBottomOf="parent"
14         app:layout_constraintLeft_toLeftOf="parent"
15         app:layout_constraintRight_toRightOf="parent"
16         app:layout_constraintTop_toTopOf="parent"
17         android:id="@+id/text_view" />
18
19      <Button
20         android:id="@+id/button_message"
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content"
23         android:text="メッセージを表示します"
24         app:layout_constraintTop_toBottomOf="@+id/text_view"
25         app:layout_constraintLeft_toLeftOf="parent"
26         app:layout_constraintRight_toRightOf="parent" />
27
28  </android.support.constraint.ConstraintLayout>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:id="@+id/text_view" />
```

```
<Button
    android:id="@+id/button_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="メッセージを表示します"
    app:layout_constraintTop_toBottomOf="@+id/text_view"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent" />
```

```
</android.support.constraint.ConstraintLayout>
```

ボタンの追加（その2）



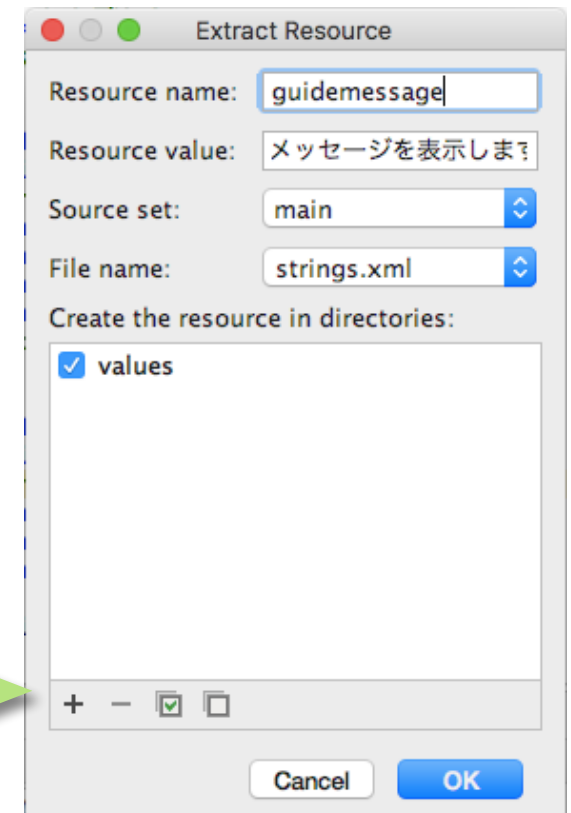
先ほどと同様に、「メッセージを表示します」の部分にカーソルを合わせて、

「**option + enter**」キー（Windows の場合は「**Alt + Enter**」）を押します。

ポップアップ表示されたダイアログボックスの中から

「**Extract string resource**」を選択します。

このような画面が表示されますので、いちばん上の「**Resource name**」に「**guidemessage**」と入力して「**OK**」をクリックします。

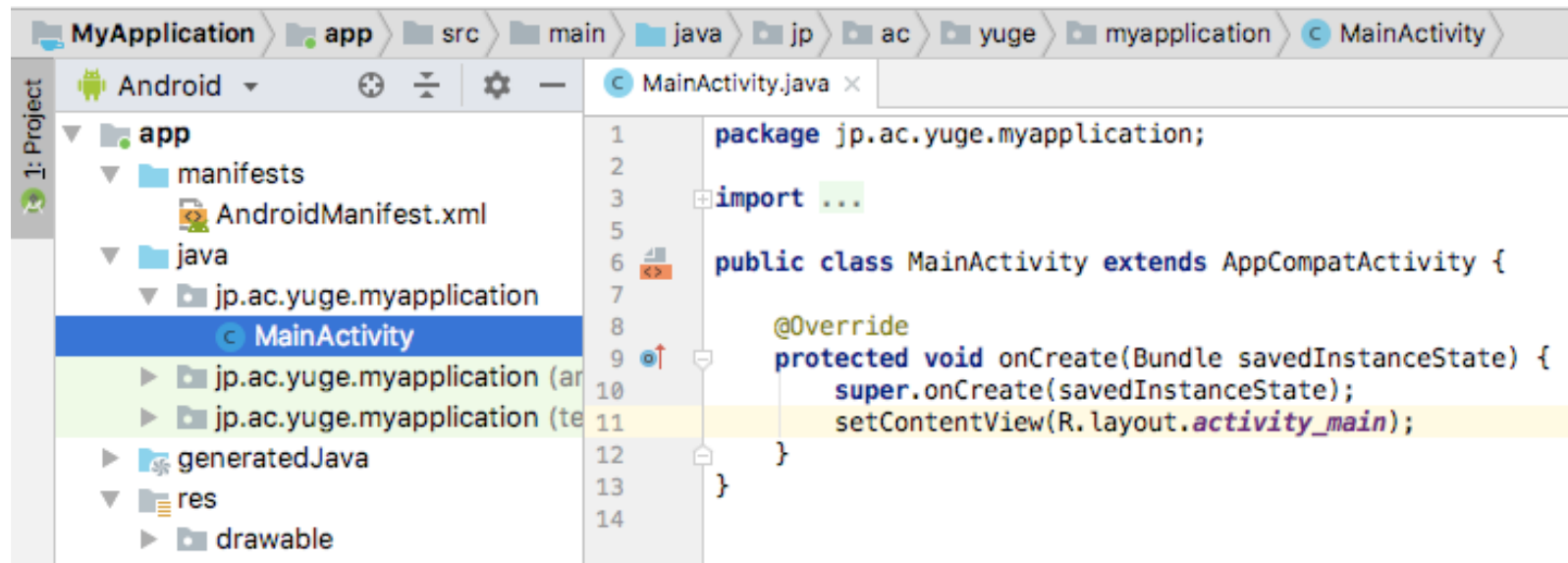


ボタンの追加（その3）



アプリを再実行すると、
メッセージの下に、ボタンが表示されます。

ボタンタップ時にメッセージを表示する（その1）



Android Studio ツール左側の「Project」欄から、
「app」 → 「java」 → 「～.myapplication」 → 「**MainActivity**」を選択してダブルクリックします。
右側に、この「**MainActivity**クラス（**MainActivity.java**）」の内容が表示されます。

ボタンタップ時にメッセージを表示する（その2）

MainActivityクラスの「**onCreate**」メソッド内に、このようにコードを追記します。
（赤色が、新しく入力する部分です。）

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button buttonMessage = (Button)findViewById(R.id.button_message);  
    }  
}
```

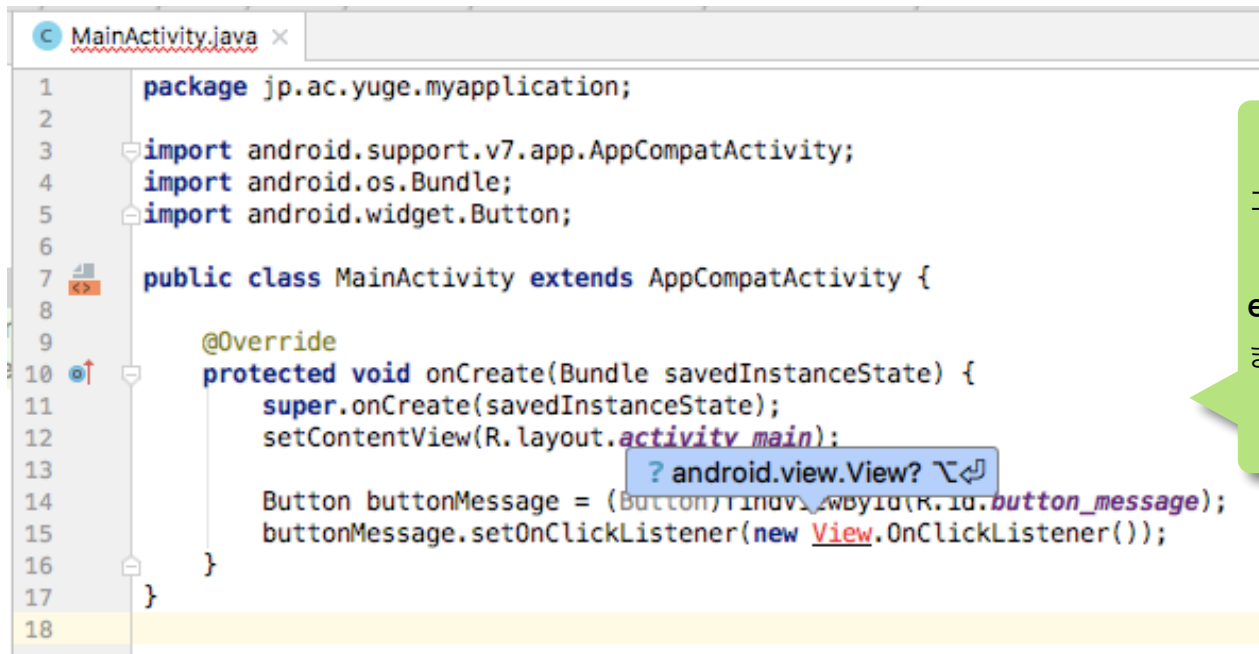


コード追記後、「**Button**」部分にカーソルを合わせて、「**option + enter**」キー
（Windowsの場合は「**Alt + Enter**」）を押します。
こうすることで「**Button**」クラスのimport文が自動的に追加されます。

ボタンタップ時にメッセージを表示する（その3）

先ほど入力したコードのすぐ下に、こちらのコードを追記します。
(赤色が、新しく入力する部分です。)

```
...  
    Button buttonMessage = (Button)findViewById(R.id.button_message);  
    buttonMessage.setOnClickListener(new View.OnClickListener());  
}  
}
```

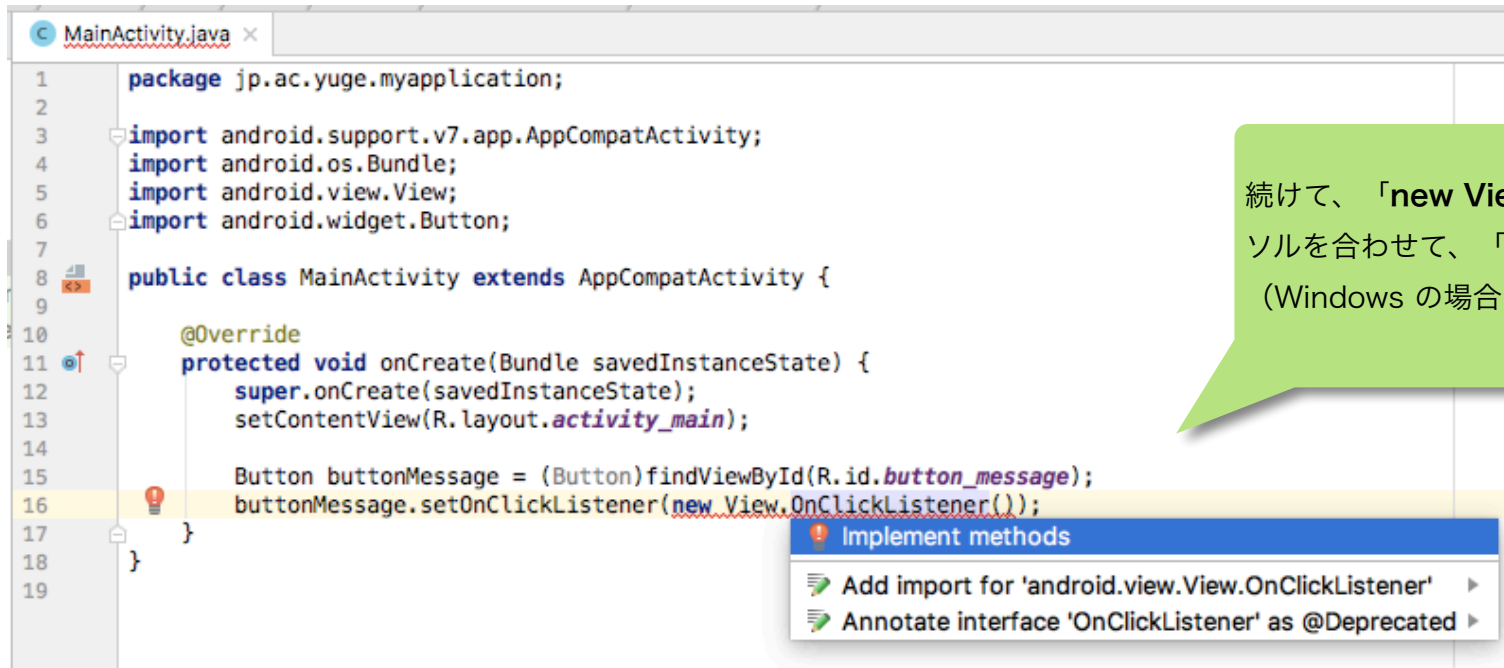


```
MainActivity.java x  
1 package jp.ac.yuge.myapplication;  
2  
3 import android.support.v7.app.AppCompatActivity;  
4 import android.os.Bundle;  
5 import android.widget.Button;  
6  
7 public class MainActivity extends AppCompatActivity {  
8  
9     @Override  
10    protected void onCreate(Bundle savedInstanceState) {  
11        super.onCreate(savedInstanceState);  
12        setContentView(R.layout.activity_main);  
13  
14        Button buttonMessage = (Button)findViewById(R.id.button_message);  
15        buttonMessage.setOnClickListener(new View.OnClickListener());  
16    }  
17 }  
18
```

A tooltip is visible over the `new View.OnClickListener()` line, displaying `? android.view.View? ↵`.

コード追記後、同様に、
「**View**」部分にカーソルを合わせて「**option + enter**」（Windowsの場合は「**Alt + Enter**」）を推します（Viewクラスのimport文が自動挿入されます）。

ボタンタップ時にメッセージを表示する（その4）



The screenshot shows the MainActivity.java file in an IDE. The code is as follows:

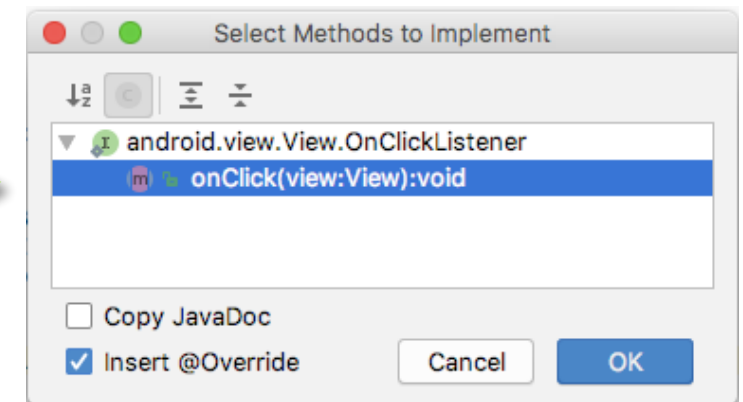
```
1 package jp.ac.yuge.myapplication;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.widget.Button;
7
8 public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14
15         Button buttonMessage = (Button)findViewById(R.id.button_message);
16         buttonMessage.setOnClickListener(new View.OnClickListener());
17     }
18 }
19
```

An IDE error message is shown below line 16, indicating that the `onClick` method is not implemented. The message box contains the following options:

- Implement methods
- Add import for 'android.view.View.OnClickListener'
- Annotate interface 'OnClickListener' as @Deprecated

続けて、「**new View.OnClickListener()**」部分にカーソルを合わせて、「**option + enter**」キー（Windows の場合は「**Alt + Enter**」）を押します。

ポップアップ表示されたダイアログボックスの中から「**Implement methods**」を選択して、「**onClick**」メソッドを追加します。



ボタンタップ時にメッセージを表示する（その5）

前の手順で自動的に追加された「onClick」メソッド内に、こちらのコードを追記します。
（赤色が、新しく入力する部分です。）

```
...  
buttonMessage.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        Toast.makeText(MainActivity.this, "ボタンがタップされました", Toast.LENGTH_SHORT).show();  
    }  
});  
...
```



```
1 package jp.ac.yuge.myapplication;  
2  
3 import android.support.v7.app.AppCompatActivity;  
4 import android.os.Bundle;  
5 import android.view.View;  
6 import android.widget.Button;  
7  
8 public class MainActivity extends AppCompatActivity {  
9  
10     @Override  
11     protected void onCreate(Bundle savedInstanceState) {  
12         super.onCreate(savedInstanceState);  
13         setContentView(R.layout.activity_main);  
14  
15         Button buttonMessage = (Button)findViewById(R.id.button_message);  
16         buttonMessage.setOnClickListener(new View.OnClickListener() {  
17             @Override  
18             public void onClick(View view) {  
19                 Toast.makeText(MainActivity.this, "ボタンがタップされました", Toast.LENGTH_SHORT).show();  
20             }  
21         });  
22     }  
23 }  
24
```

? android.widget.Toast? ↵

コード追加後、「Toast」部分にカーソルを合わせて、「option + enter」（Windowsの場合は「Alt + Enter」）して、Toastクラスのimport文を自動挿入します。

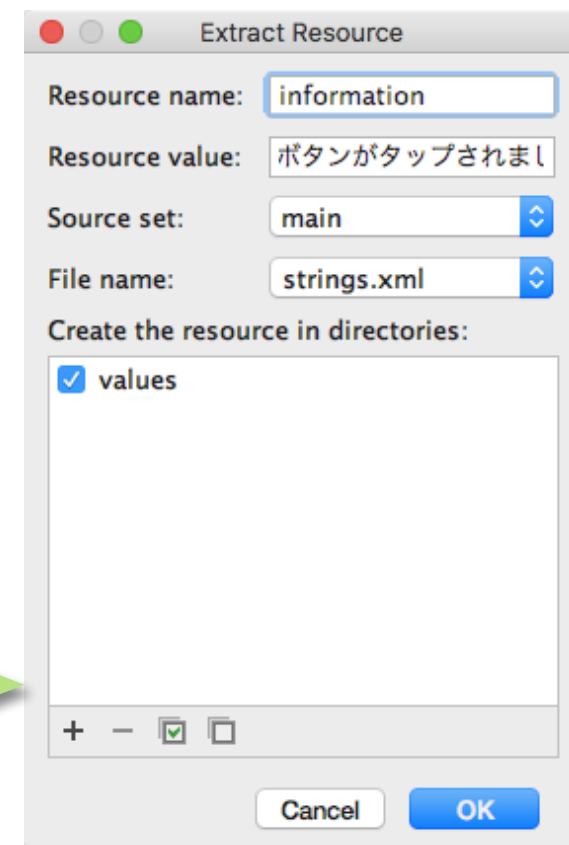
ボタンタップ時にメッセージを表示する（その6）

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button buttonMessage = (Button)findViewById(R.id.button_message);  
        buttonMessage.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                Toast.makeText(MainActivity.this, "ボタンがタップされました", Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

- ✓ Extract string resource ▶
- Convert to Basic Latin ▶
- Insert App Indexing API Code ▶
- Inject language or reference ▶

「ボタンがタップされました」の部分にカーソルを合わせて、
「Alt + Enter」キー（Macの場合は「option + enter」）
を押します。
ポップアップ表示されたダイアログボックスの中から
「Extract string resource」を選択します。

このような画面が表示されますので、いちばん上の「Resource name」
に「information」と入力して「OK」をクリックします。



ボタンタップ時にメッセージを表示する（その7）

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button buttonMessage = (Button)findViewById(R.id.button_message);  
        buttonMessage.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                Toast.makeText(MainActivity.this, R.string.information, Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

「ボタンがタップされました」と入力していた部分には、
「R.string.information」と表示されます。

ボタンタップ時にメッセージを表示する（その8）



Android Studio の「Run」ボタンを押して、アプリを再実行します。
アプリの画面内の「メッセージを表示します」ボタンをタップすると、画面下部にメッセージが表示されます。