Androidアプリ開発テキスト

目　次

１．Android 端末を接続する

（１）Android 端末と PC の USB 接続を解除する　→　ケーブルを繋いでいない状態にする。
（２）USB デバッグを ON にする。
設定　→　（その他）　→　端末情報　→　ビルド番号を７回タップする
→　すると「開発者向けオプション」がでてくる。
→　USB デバッグをタップし✔（チェック）を入れる。

（3）USB ドライバを設定する。

Android 端末と PC を USB で接続し、Eclipse を起動して下さい。
起動したらメニューから[ウィンドウ] → [パースペクティブを開く]を選択する（図１）。
そして一覧から「DDMS」を選択して、[OK]ボタンをクリックして下さい（図２）。
図のように「Devices」に接続した Android 端末が確認されていれば成功です（図３）。

図１

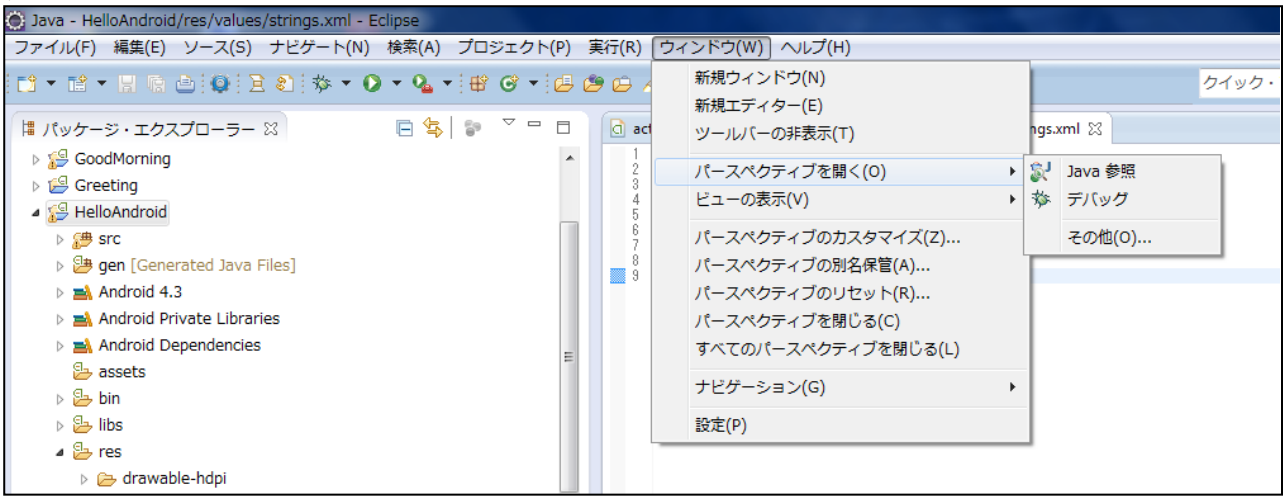


図２

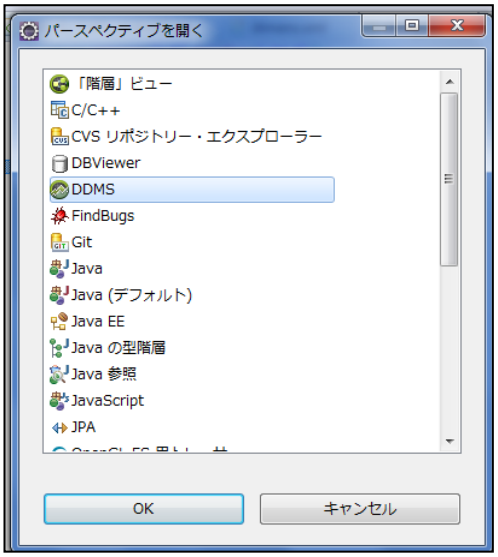
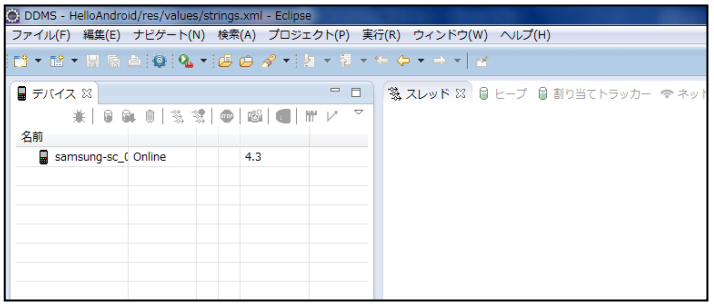


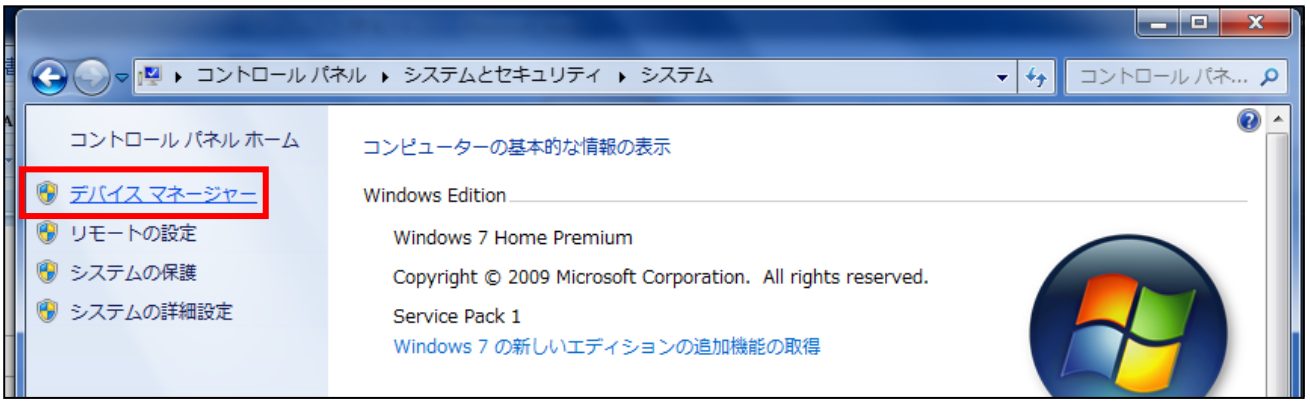
図３



２．ドライバーが入手できなかった場合の対処方法

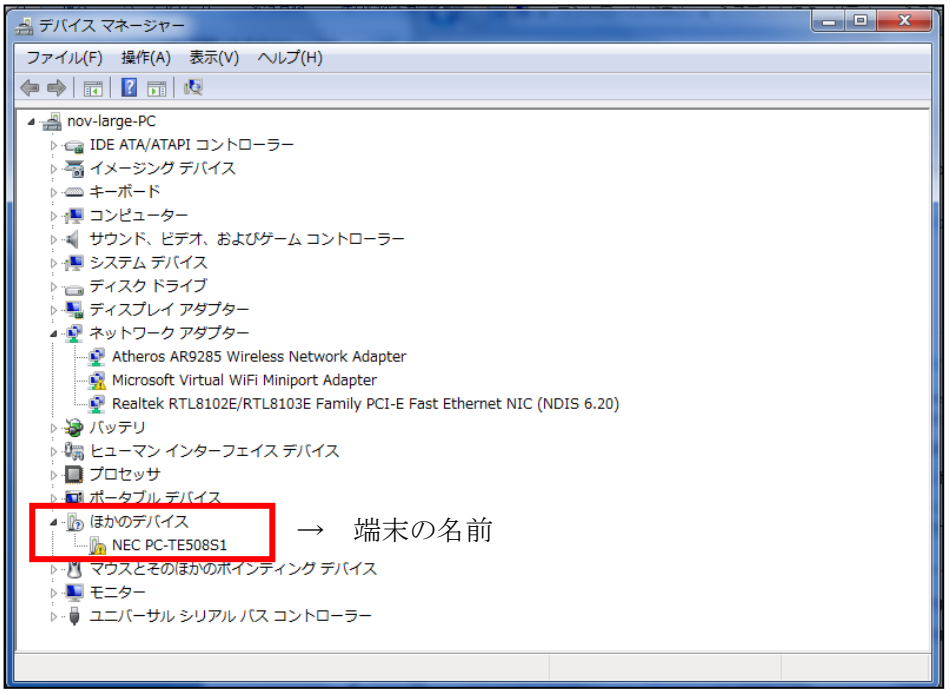
（１）コントロールパネル　→　デバイス・マネージャーを開きます（図１）。

図１



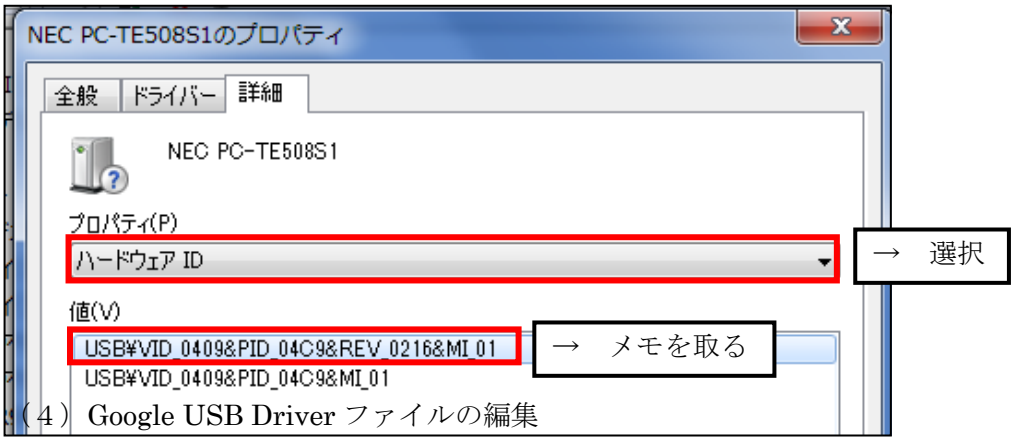
（２）「ほかのデバイス」をクリックします（図２）。

図２



（３）プロパティで「ハードウェア ID」をメモします（図３）。

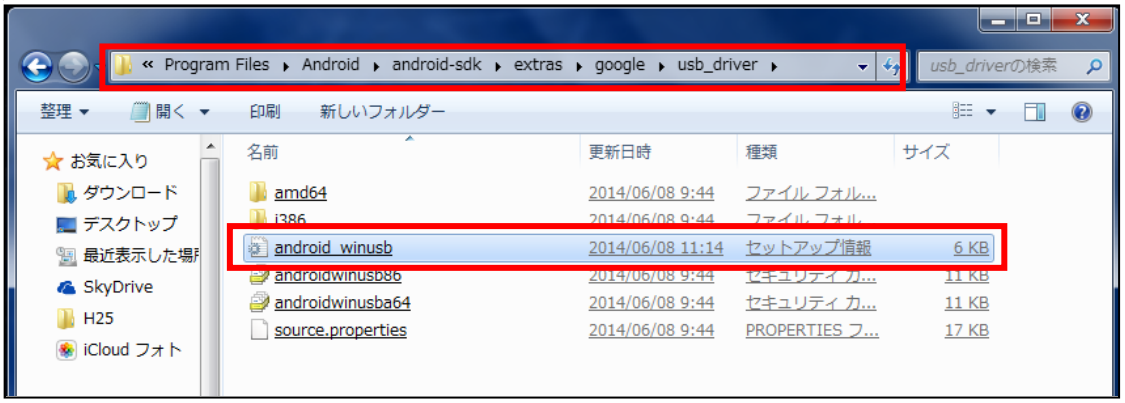
図３



（４）Google USB Driver ファイルの編集

▼編集先の例（図４）

図４



（５）利用している PC が 32bit の PC なら[Google.NTx86]のセクションを探して下さい、64bit の PC なら[Google.NTamd64]のセクションを探し、次の３行を追加して下さい（表１）。

表１

▼追加するコード

```
; (端末の名前)
%SingleAdbInterface% = USB_Install, USB¥VID_(      )&PID_(          )
%CompositeAdbInterface% = USB_Install, USB¥VID_0409&PID_04C9&MI_01
```

VID は「ハードウェア ID」を選択してメモした USB¥VID_の後ろの４ケタの部分、図３でいえば「0409」が VID になります。そして PID はおなじく PID_の後ろの４ケタの部分、図３でいえば「04C9」が PID になります。端末の名前は、利用している端末の名前を入れておくとよいでしょう。

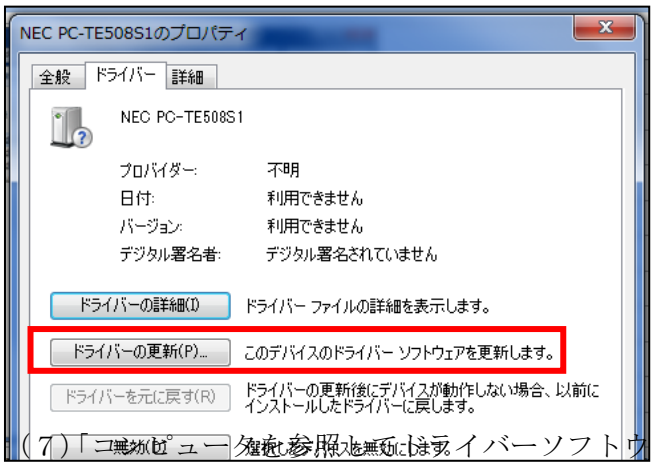
この端末の場合は

```
;NEC PC-TE508S1

%SingleAdbInterface% = USB_Install, USB¥VID_0409&PID_04C9

%CompositeAdbInterface% = USB_Install, USB¥VID_0409&PID_04C9&MI_01
```

※□の中の英数字は、デバイスによって異なります。

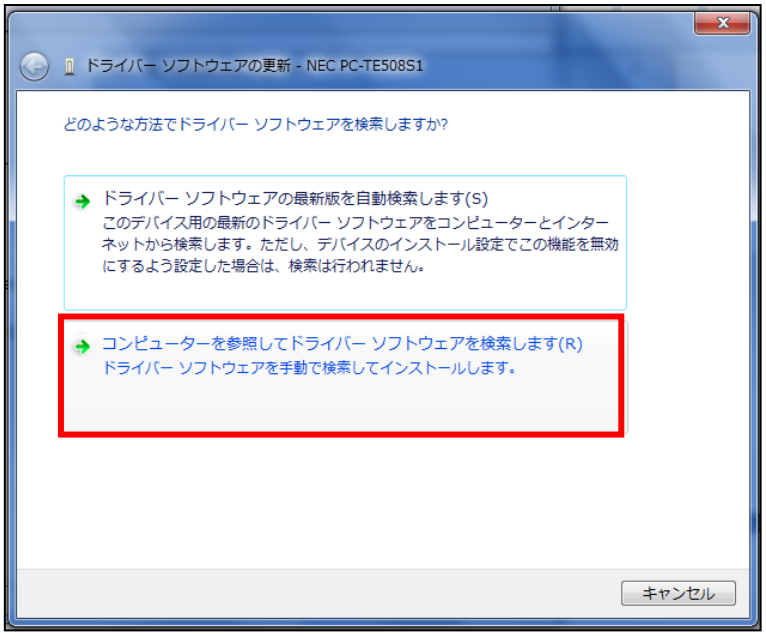
（６）「ドライバー」の「ドライバーの更新」をクリックします（図５）。

図５



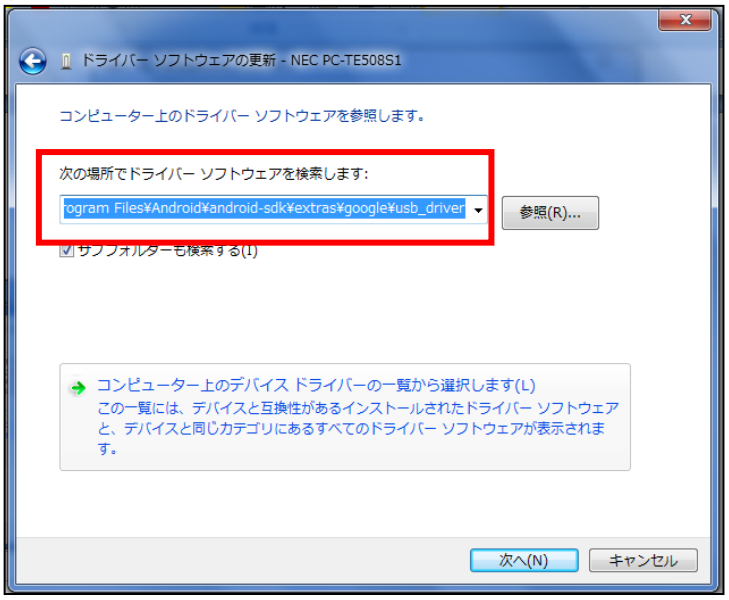
（７）「コンピュータを参照してドライバーソフトウェアを検索します（R）」をクリックします（図６）。

図６



（８）場所を指定して、次へをクリックします（図７）。

図７



（9）「このドライバーのソフトウェアをインストールします（I）」をクリックします（図８・９・10）。

図８

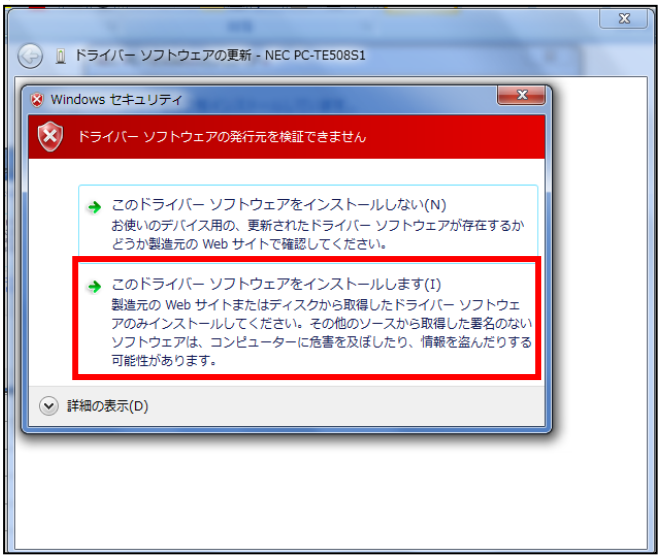
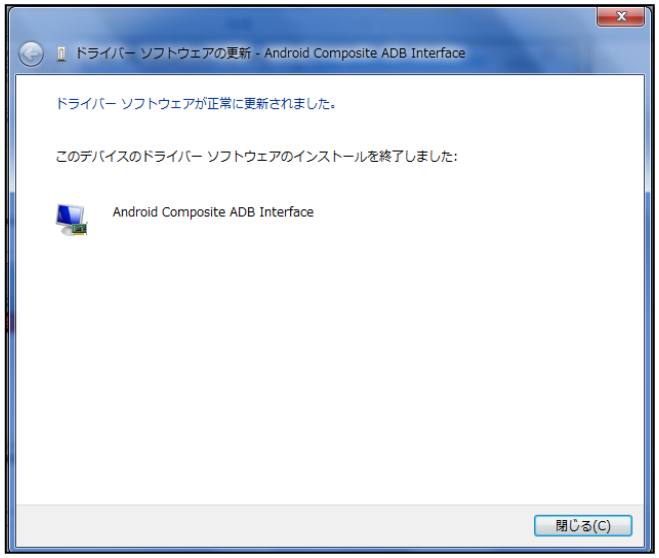
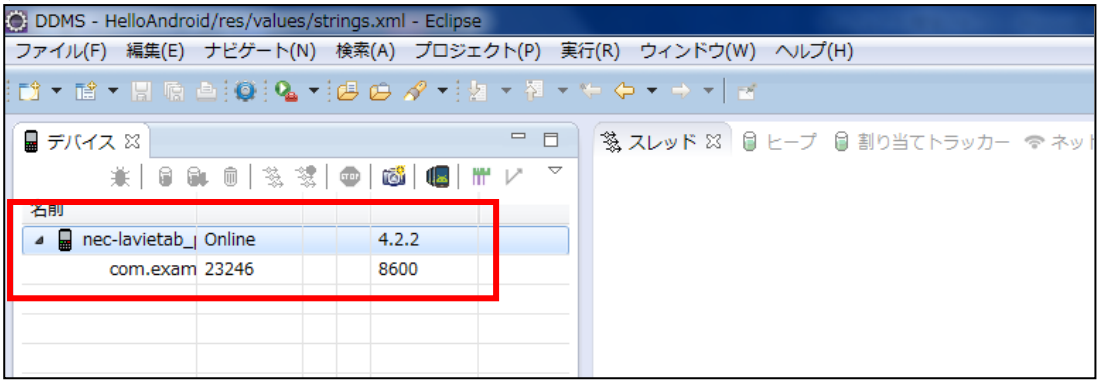


図９



（10）ドライバーが入手できたか確認する（図 10）。

図 10



３．Android エミュレータの基本操作

ここでは、本格的に Android の技術要素を学習する前に、最初のアプリケーションを作成した際にも起動させた Android エミュレータの基本的な操作について確認します。Eclipse と同様、いつも使用する開発ツールとなるので、使い方について習熟するよう努めましょう。

（１）Android エミュレータ

Android エミュレータは、Android の仮想端末であり、作成した Android アプリケーションをその上で仮想的に実行することができます。実際の携帯端末に標準で搭載されている電話やブラウザ、ＧＰＳ機能などもエミュレータ上で仮想的に実現されるため、それらの機能を利用したアプリケーションの動作検証も行えます。

（２）エミュレータの起動と終了

Android エミュレータの起動は、Eclipse で実行したいプロジェクトを選択し、「実行」→「Android アプリケーション」を選択します（図１・２）。

図１

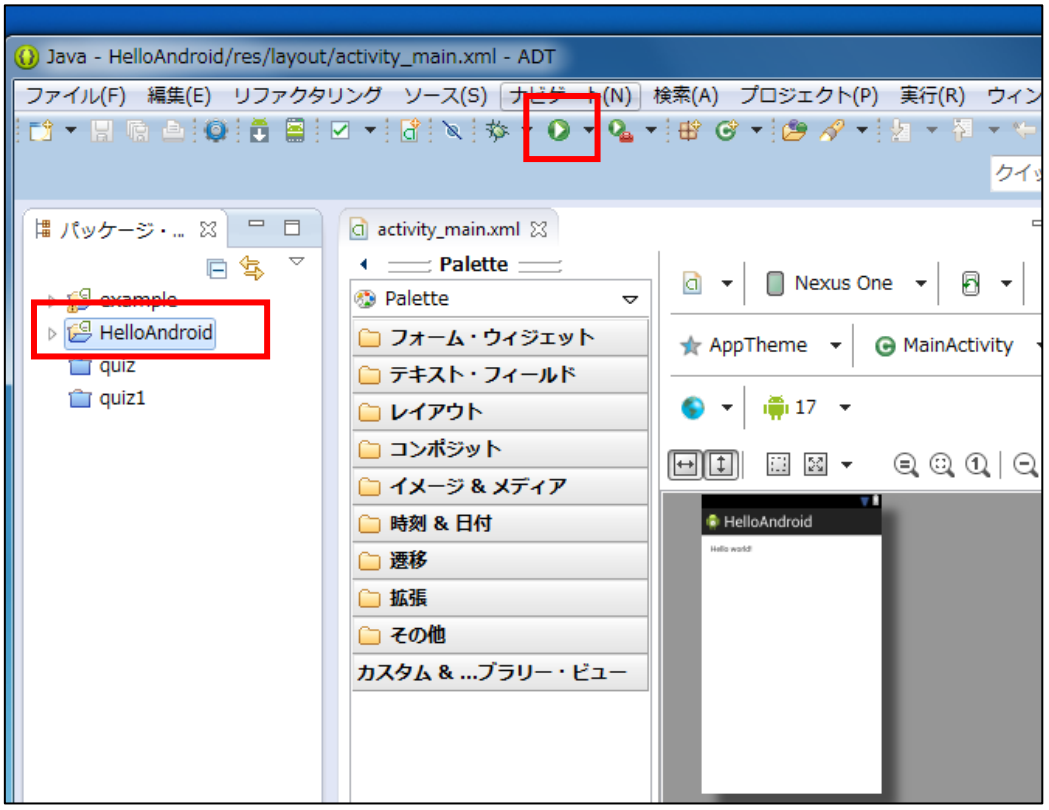
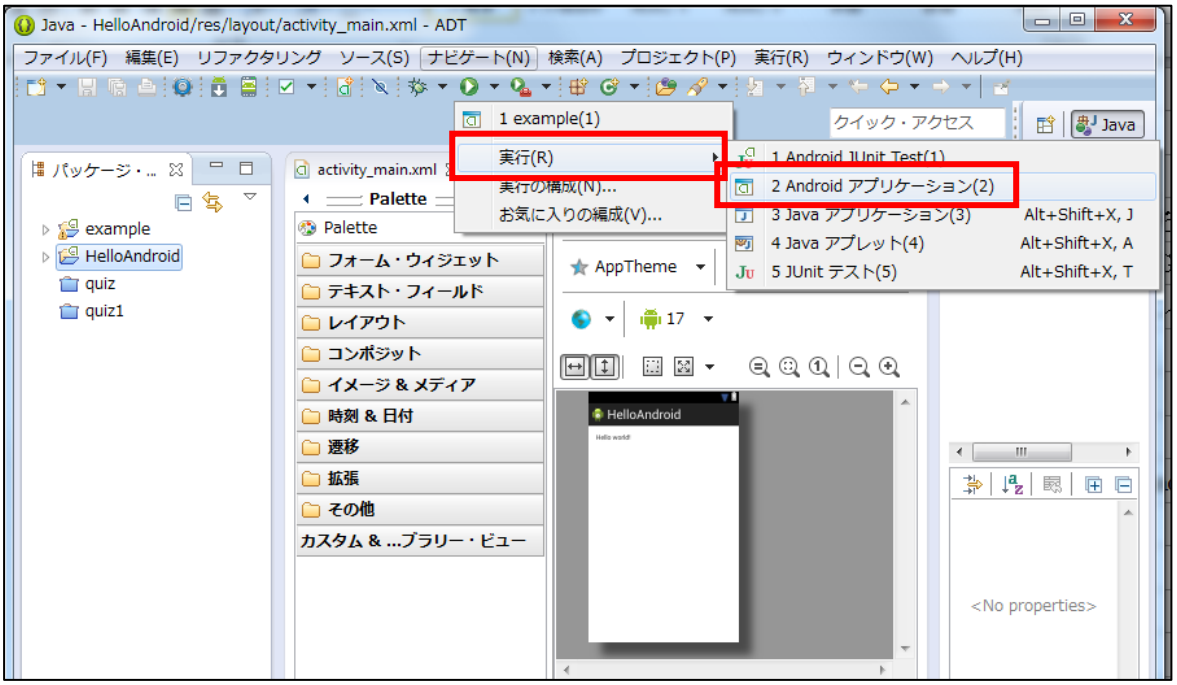


図２



（３）エミュレータの基本操作

次にエミュレータの基本操作について説明します。

エミュレータは、マウスクリックやキーボード入力によって、操作することが可能です。実際の携帯端末の画面をタッチする操作は、マウスを使ってクリックすることによって実現されます。ダブルタップやドラッグなどの操作もマウスを使います（図３・４・５・６）。

図３

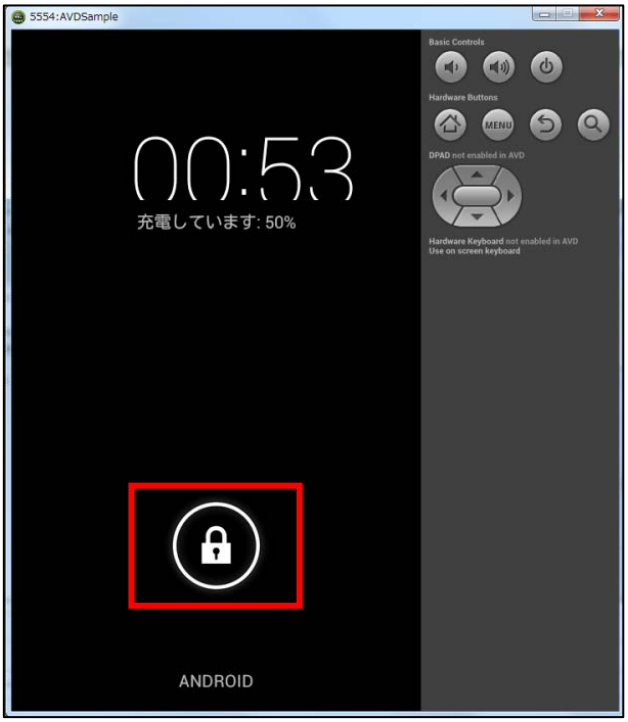


図４

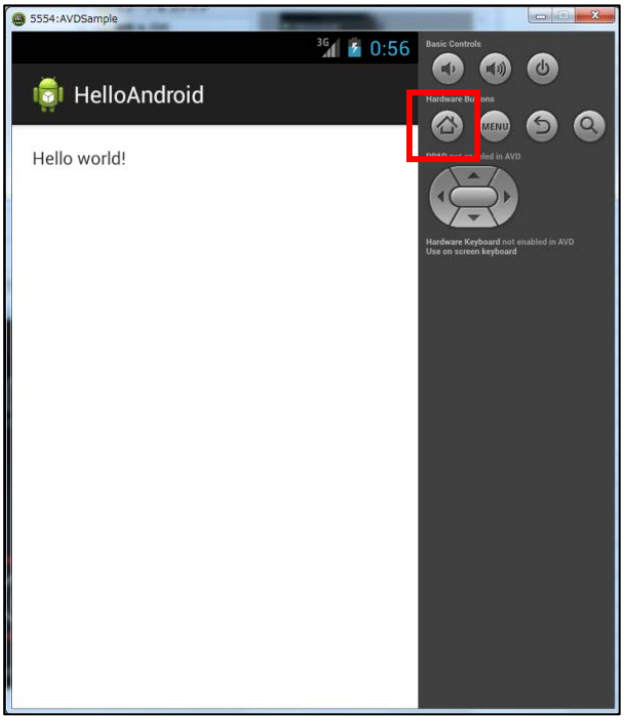


図５

図６

（４）アプリケーションのアンインストール

アプリケーション「HelloAndroid」をアンインストールすることにより、アンインストールの手順を確認しておきましょう。

「MENU」ボタンをクリックします（図７）。

表示されたメニューの右下の「アプリの管理」を選択してください（図８）。

図７

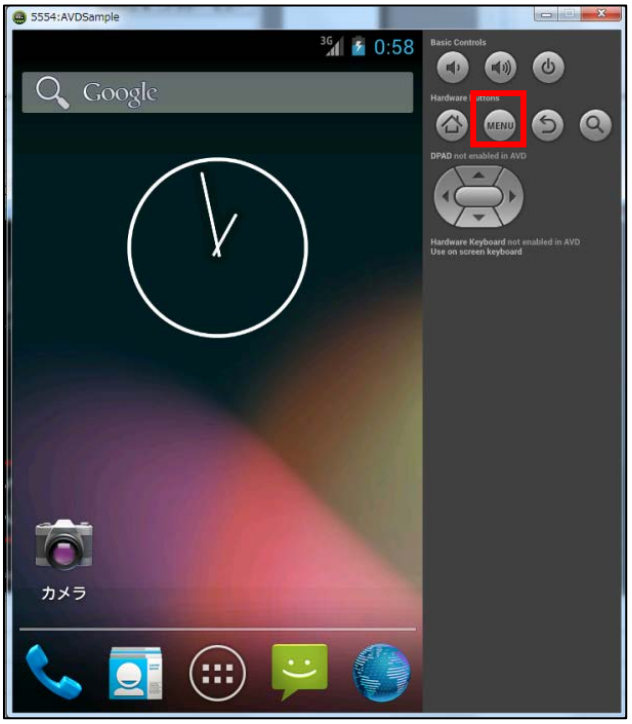


図８

（5）インストールされているアプリケーション一覧が表示されていますので、アンインストールしたいアプリケーションを選択してください（図９）。
アプリ情報画面で、「アンインストール」ボタンをクリックします（図 10）。

図９

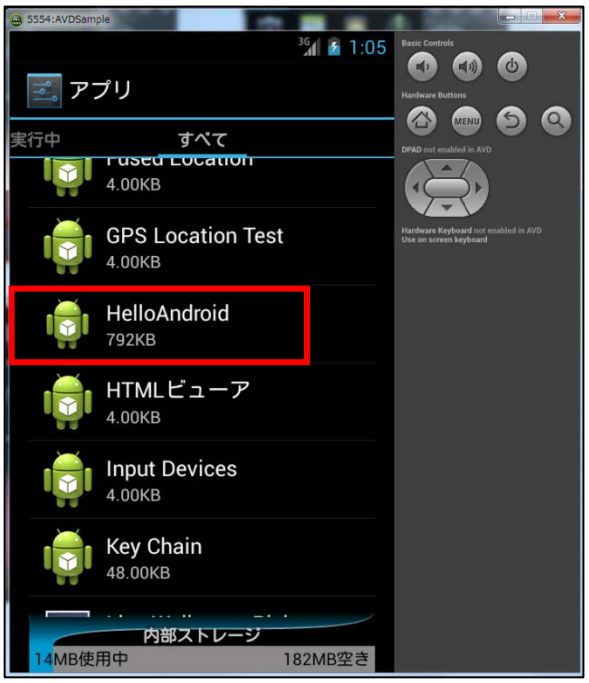


図 10

（6）アンインストールの確認画面が表示されますので、「OK」ボタンをクリックしてください。

図 11

図 12

（7）エミュレータを閉じる時は、右上の「閉じる」ボタンで終了します。

４．アプリケーションの開発手順

（１）プロジェクトの作成

Eclipse のメニューから「ファイル」→「新規」→「プロジェクト」→「Android アプリケーション・プロジェクト」を選択します（図１・２）。

図１

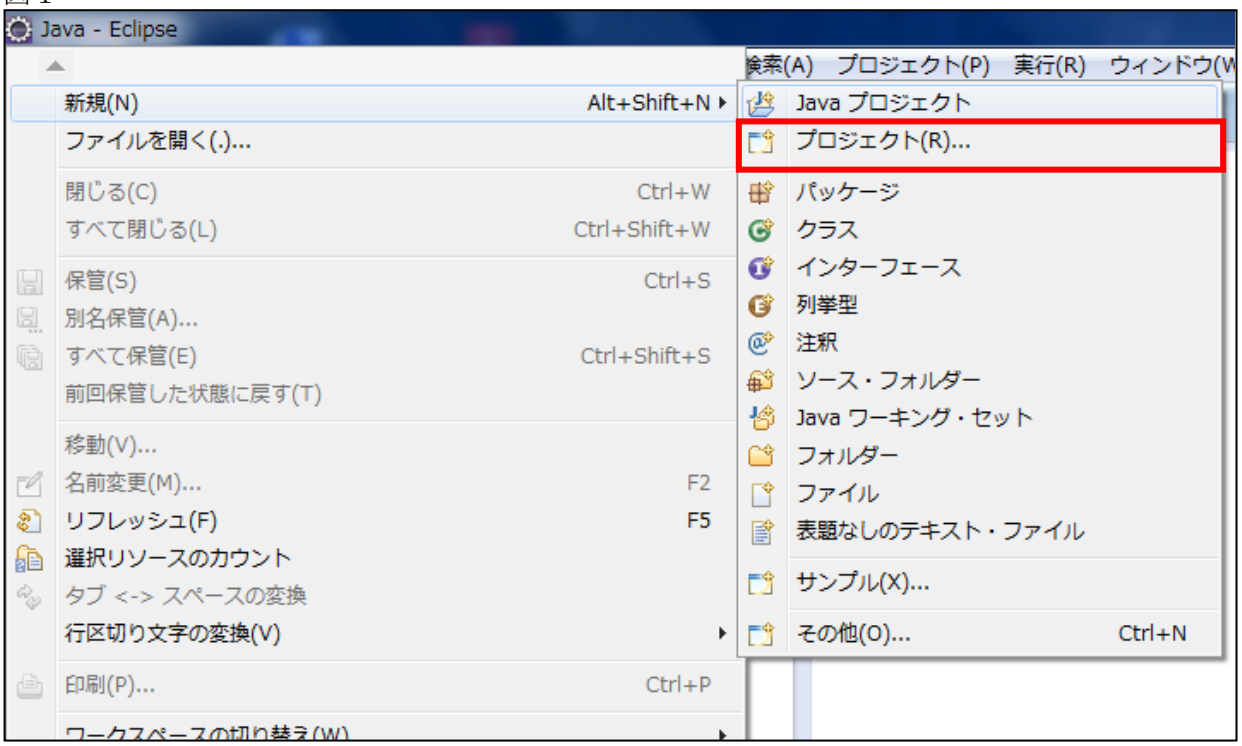
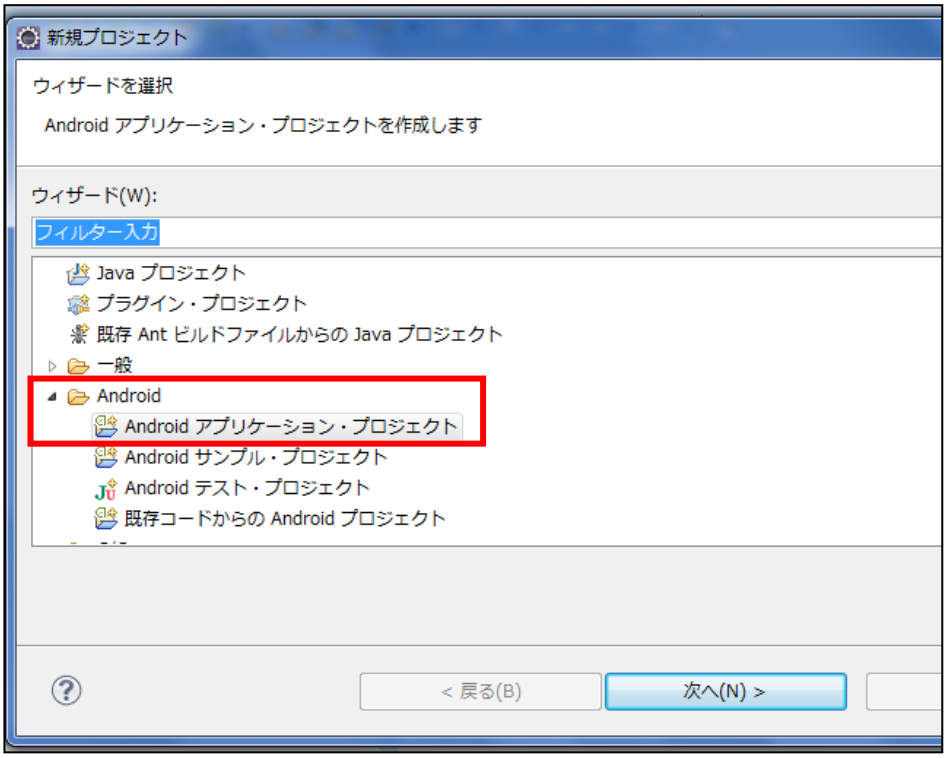
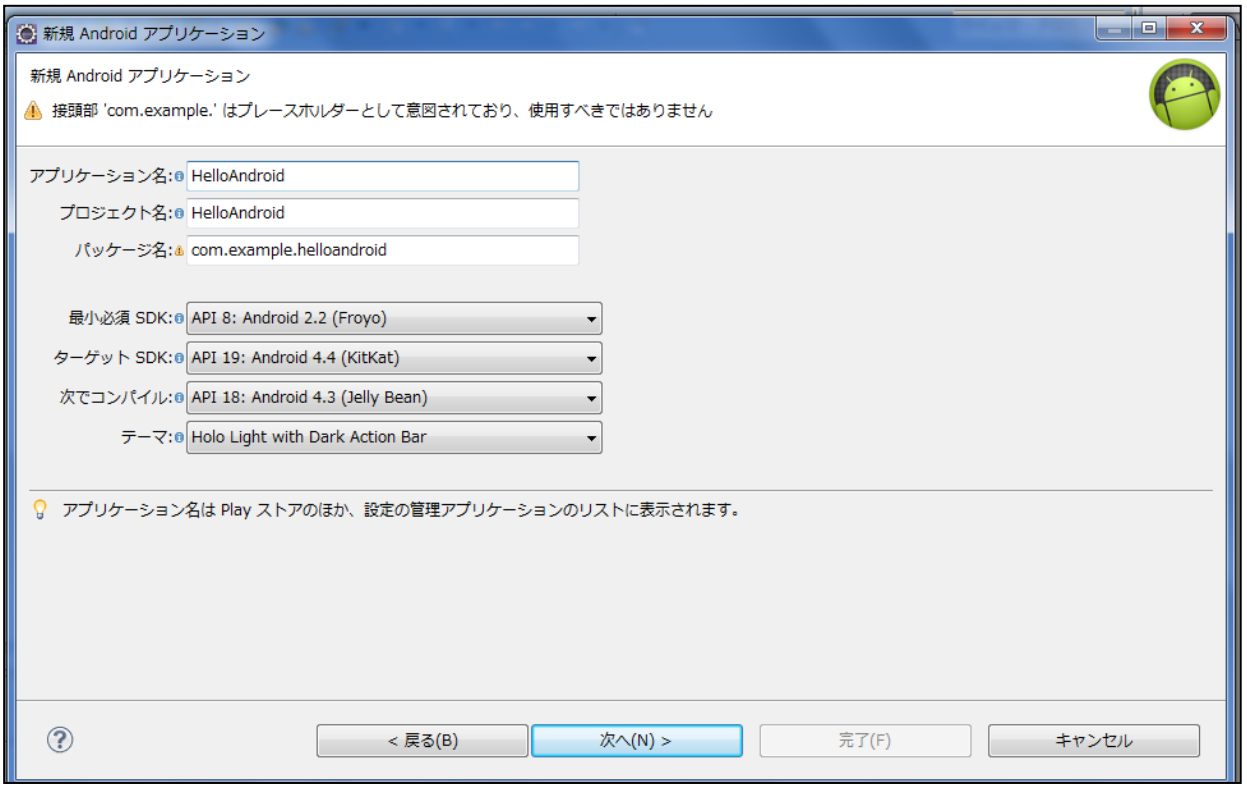


図２



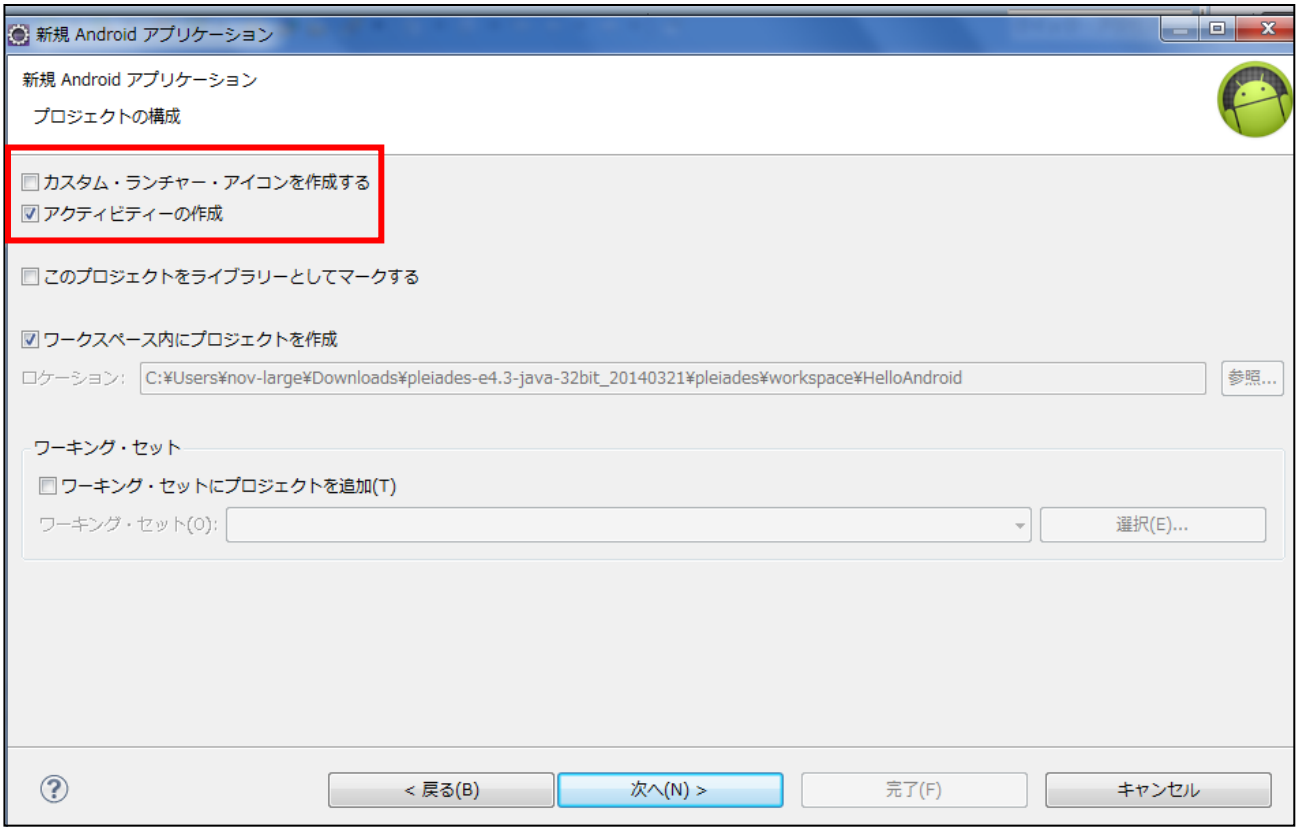
（2）新規の Android プロジェクトを作成する「新規 Android アプリケーション」ダイアログが表示されます。「アプリケーション名」と「プロジェクト名」に「HelloAndroid」、「パッケージ名」に「com.example.helloandoroid」と入力して「次へ」ボタンをクリックして下さい（図３）。

図３



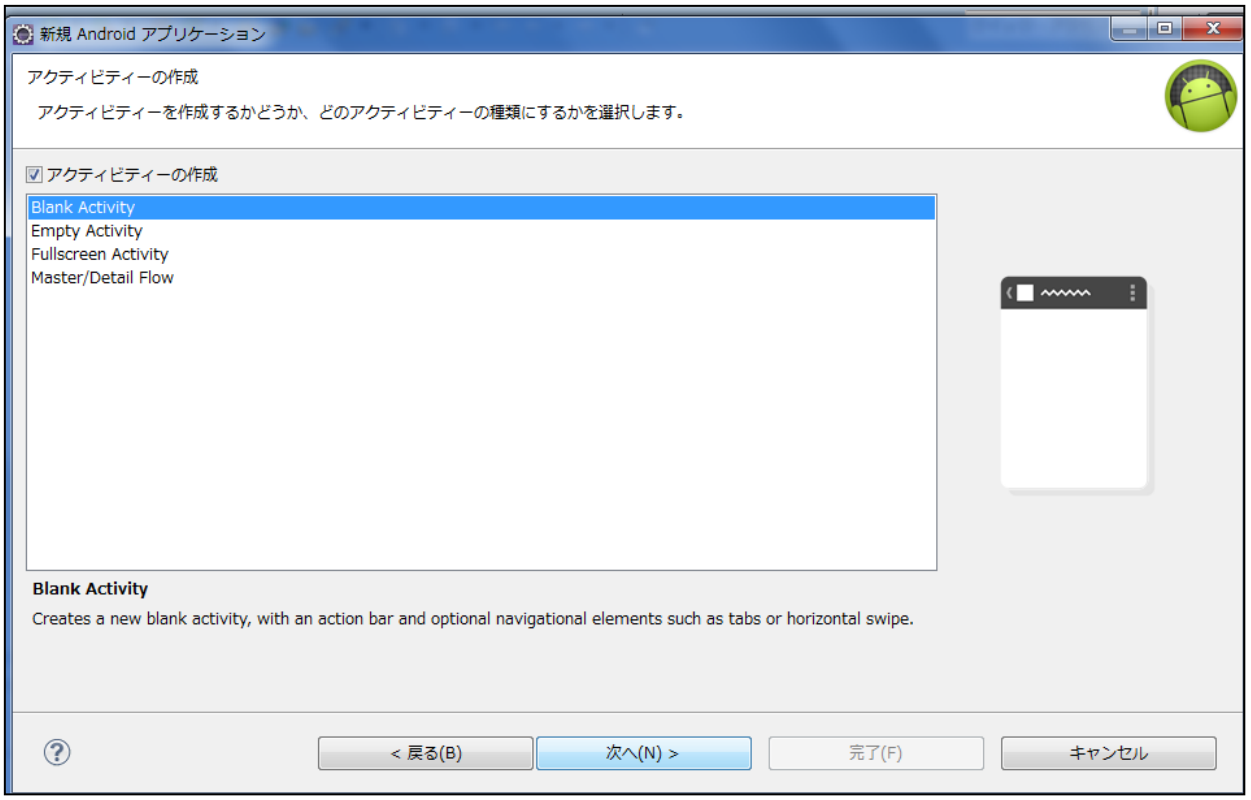
（3）「カスタム・ランチャー・アイコンを作成する」のチェックをはずして、「次へ」ボタンをクリックする（図４）。

図４



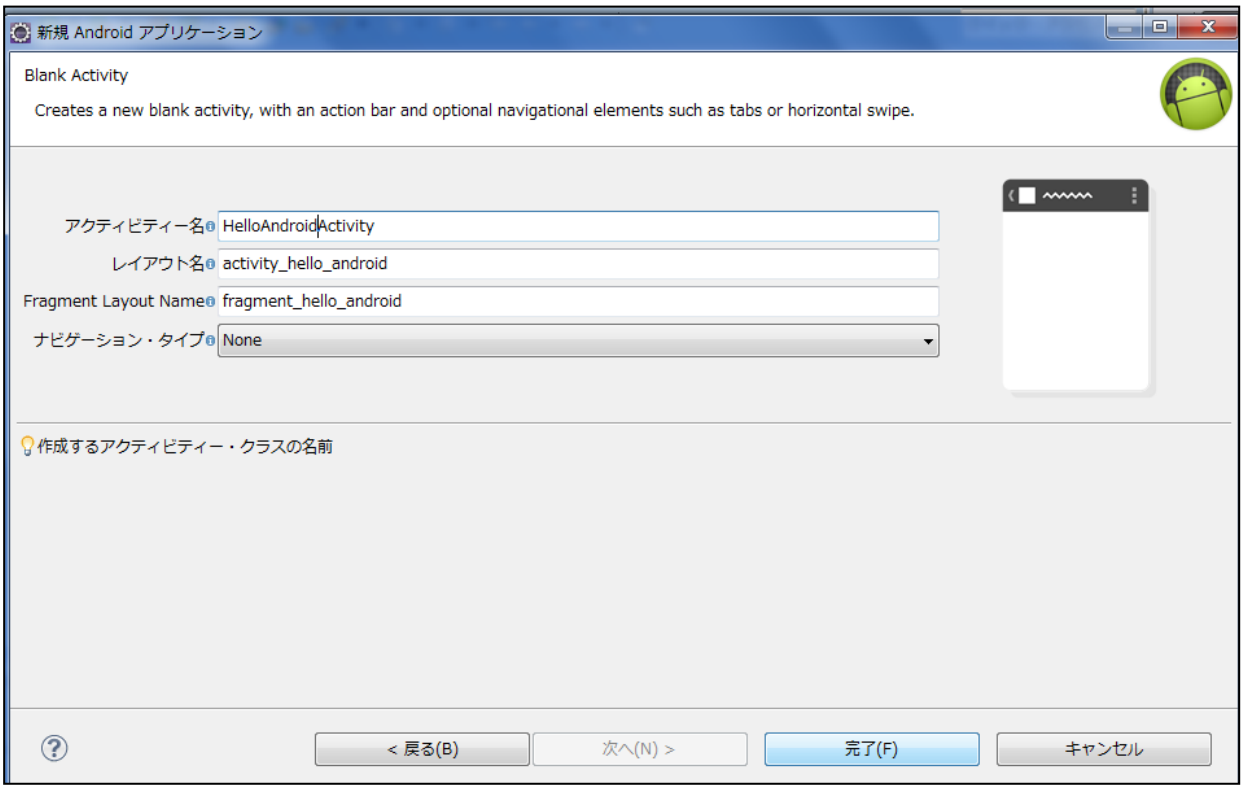
（４）次の画面で「アクティビティーの作成」にチェックをして、「Blank Activity」を選択し、「次へ」ボタンをクリックします（図５）。

図５



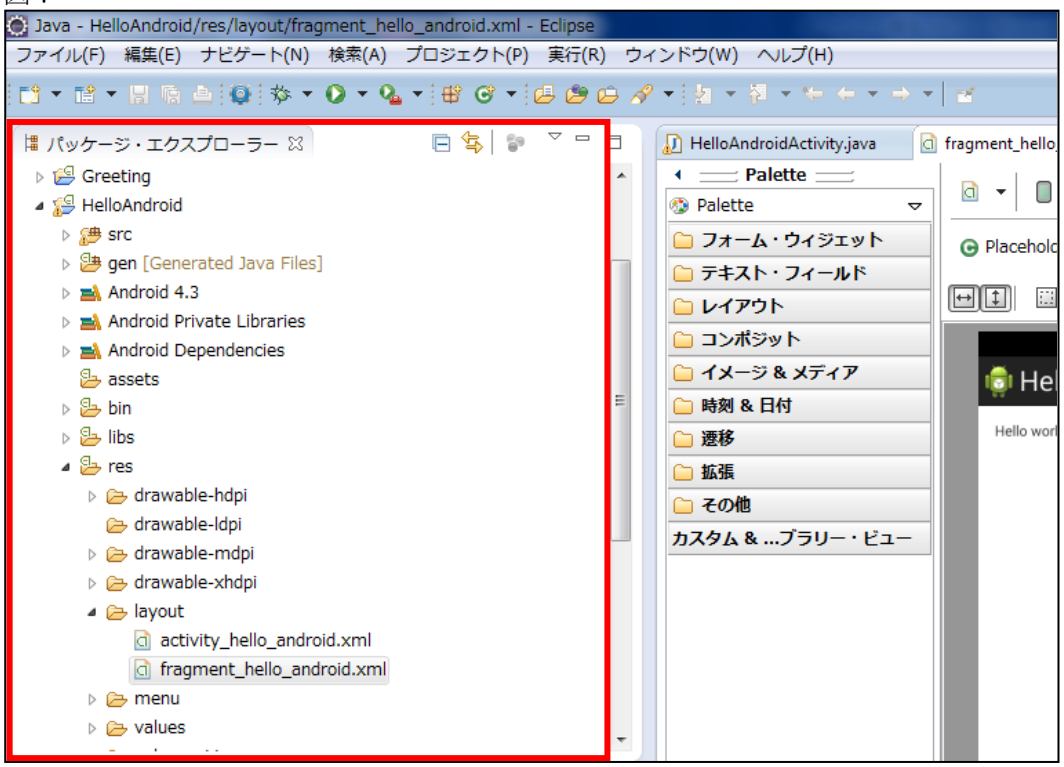
（５）次の画面で「アクティビティー名」に「HelloAndroidActivity」、「レイアウト名」に「Activity_hello_android」を入力し、「ナビゲーション・タイプ」に「None」を選択し、「完了」ボタンをクリックします（図６）。

図６



（6）「完了」ボタンのクリックにより「パッケージエクスプローラー」に作成したプロジェクトが表示されます（図７）。

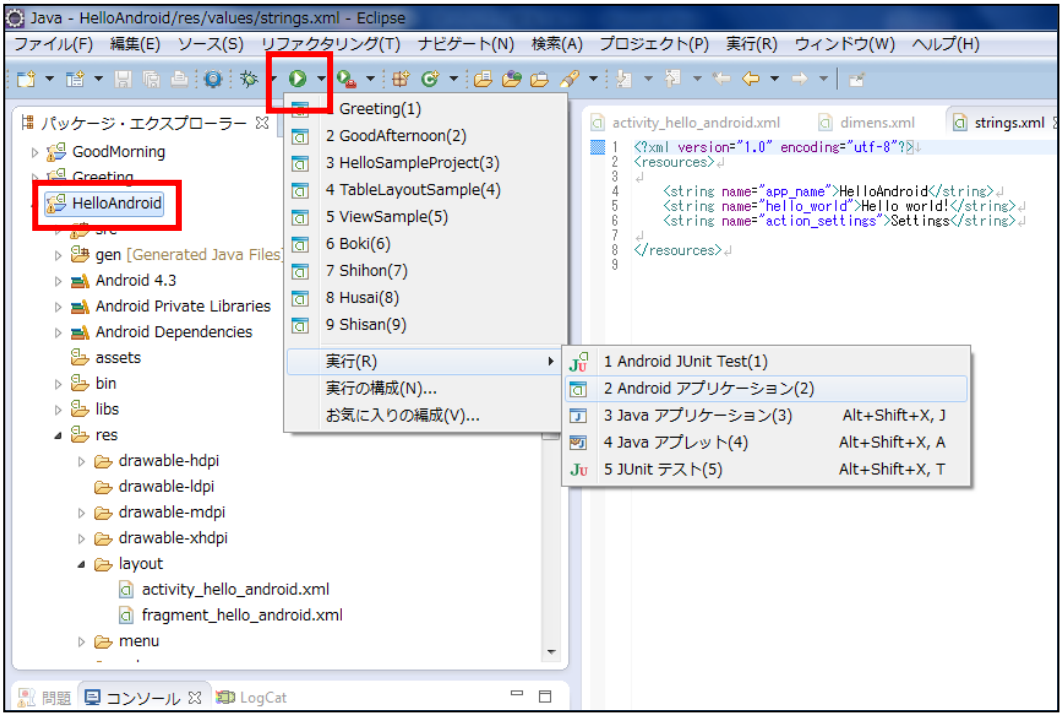
図７



（7）作成したアプリケーションを実行してみましょう。

パッケージエクスプローラーにある「HelloAndroid」プロジェクトを選択し、実行ボタンをクリックします（図８）。

図８



５．Androidプロジェクトの構成

（1）プロジェクトの構成

Androidプロジェクトの各ディレクトリやファイルなどの配置はルール化されており、その通りに配置しないとアプリケーションが適切に動作しませんので、配置ルールをよく理解しておく必要があります。

アプリケーションを開発するときに中心となるのが、「src」ディレクトリ、「res」ディレクトリ、「AndroidManifest.xml」ファイルです。

「src」ディレクトリが、Javaプログラムを中心としたソースファイルを格納するディレクトリです。このディレクトリの配下にパッケージを作成し、その中に各プログラムファイルを配置するようにしましょう。

「res」ディレクトリもよく使用するディレクトリです。Android プロジェクトでは、画像データ、レイアウトとビューの設定ファイル、各レイアウトとビュー、文字列定義ファイルなどをリソースと呼びます。これらのリソースは、「res」ディレクトリのサブディレクトリに種類ごとに格納します。

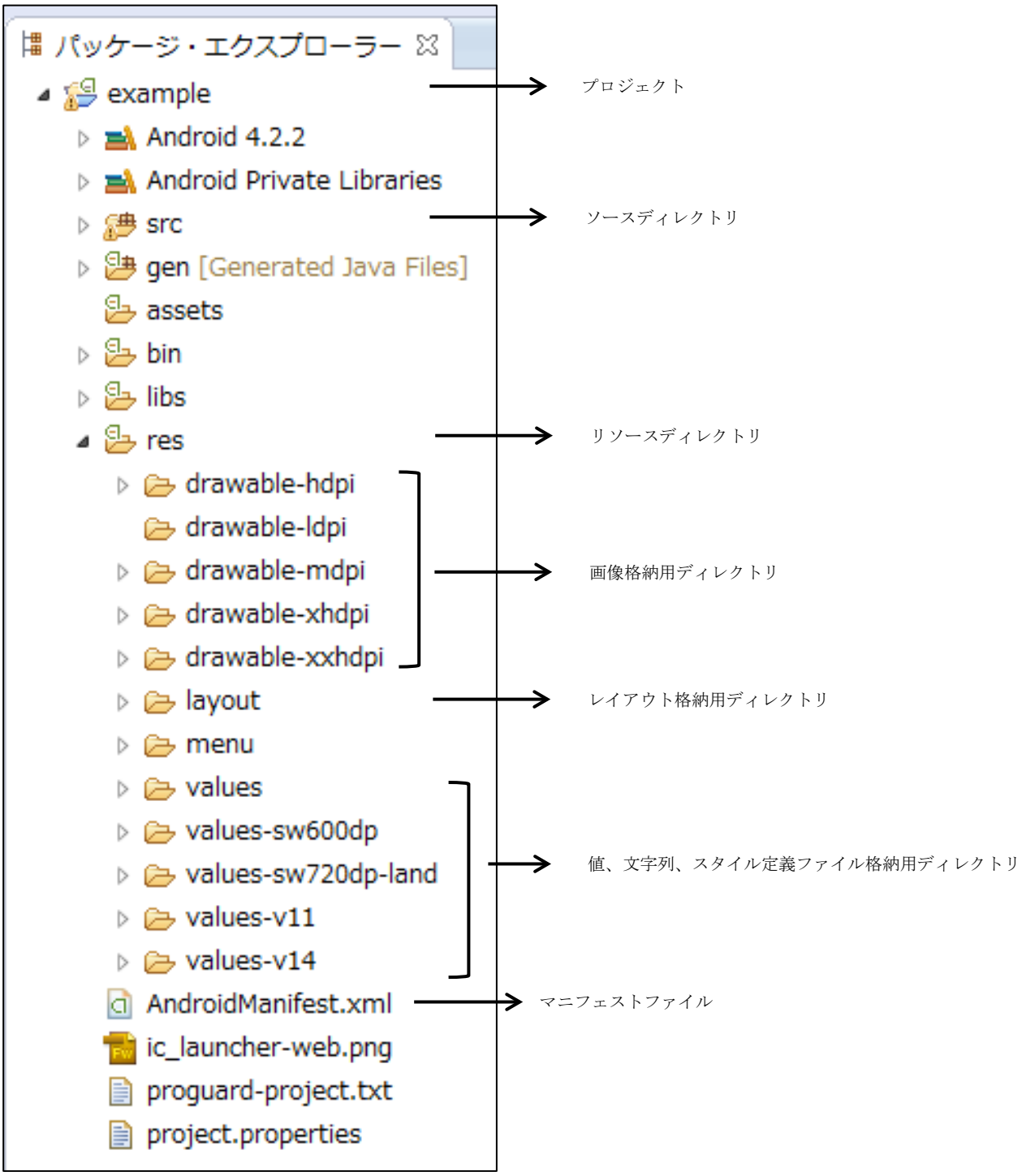
「AndroidManifest.xml」ファイルは、アプリケーションに関する必要な情報を設定します。アプリケーション名やアプリケーションを一意に識別するためのパッケージ名、各基本機能を実装したクラス名をこのマニフェストファイルに設定します。その他、外部アプリケーションからのアクセス権限の設定やアプリケーションを実行するために必要な最低限のＡＰＩのレベルなどを定義します。プロジェクトを作成した時に、基本的な設定は自動的に記述されていますので、クラスを作成してAndroidシステムに認識するように追記をしたり、アクセス権限の追記をしたりする時にこのファイルを編集することになります。

表１　プロジェクト配下の主要ディレクトリとファイル

ディレクトリ・ファイル名	説明
ｓｒｃ	作成するプログラムファイルを格納するソースディレクトリ。その配下にパッケージプログラムファイルを配置する。
ｒｅｓ	各リソースを格納するディレクトリ。リソースとは、画像データ、レイアウトとビューの設定ファイル、文字列定義ファイルなどを指す。リソースの種類によって、サブディレクトリごとに管理する。
ＡｎｄｒｏｉｄＭａｎｉｆｅｓｔ．ｘｍｌ	アプリケーション実行に必要な情報を記載したマニフェストファイル。

表２　サブディレクトリ名と格納するリソース

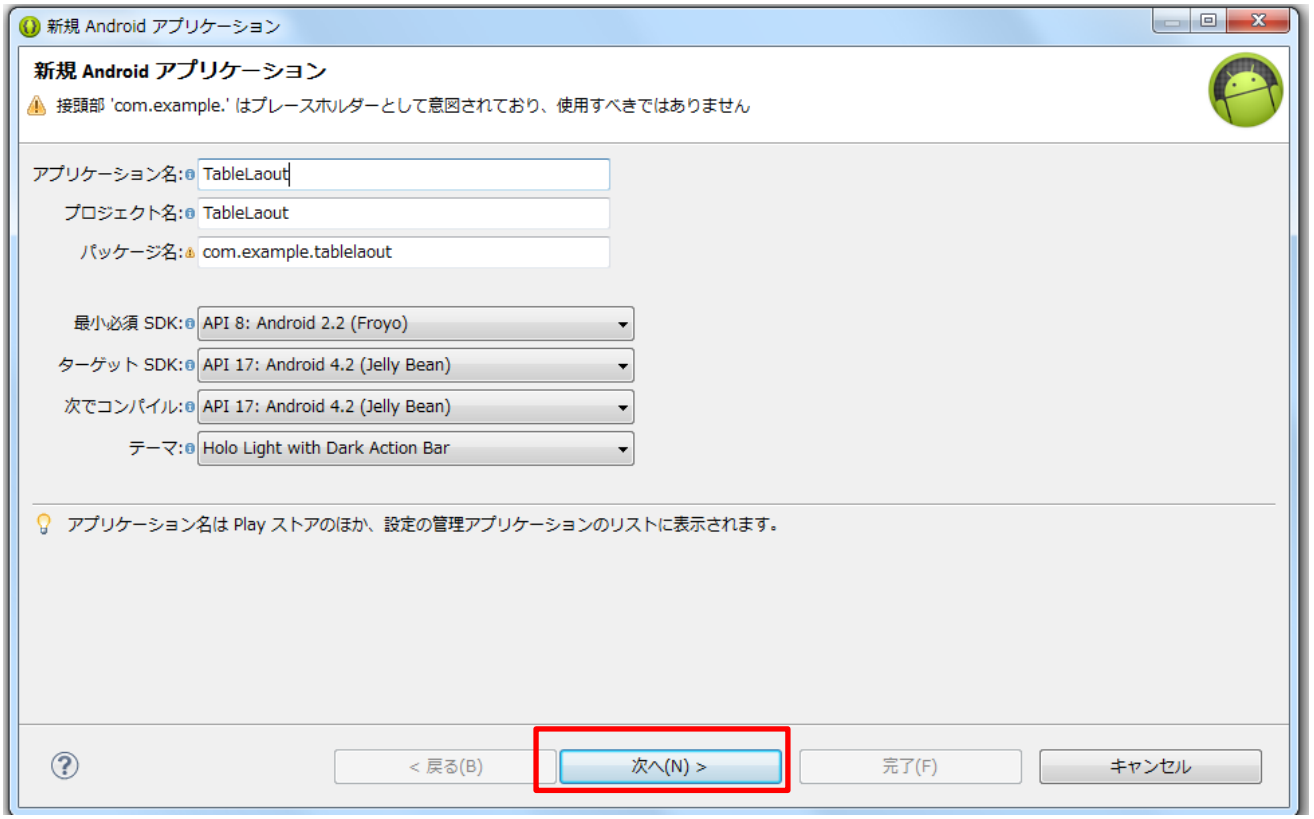
サブディレクトリ名	格納するリソースの種類
ａｎｉｍ	アニメーションを定義したＸＭＬファイル。
ｃｏｌｏｒ	カラーリストを定義したＶＭＬファイル。
ｄｒａｗａｂｌｅ	画像ファイルや形状などを定義したＸＭＬ。
ｌａｙｏｕｔ	レイアウトとビューを定義したＸＭＬファイル。
ｍａｎｕ	アプリケーションのメニューを定義したＸＭＬファイル。
ｖａｌｕｅｓ	文字列、文字列の配列を定義したＸＭＬファイルや画面に配置されるビューのスタイルを定義したＸＭＬファイルや値を定義したＸＭＬファイル。

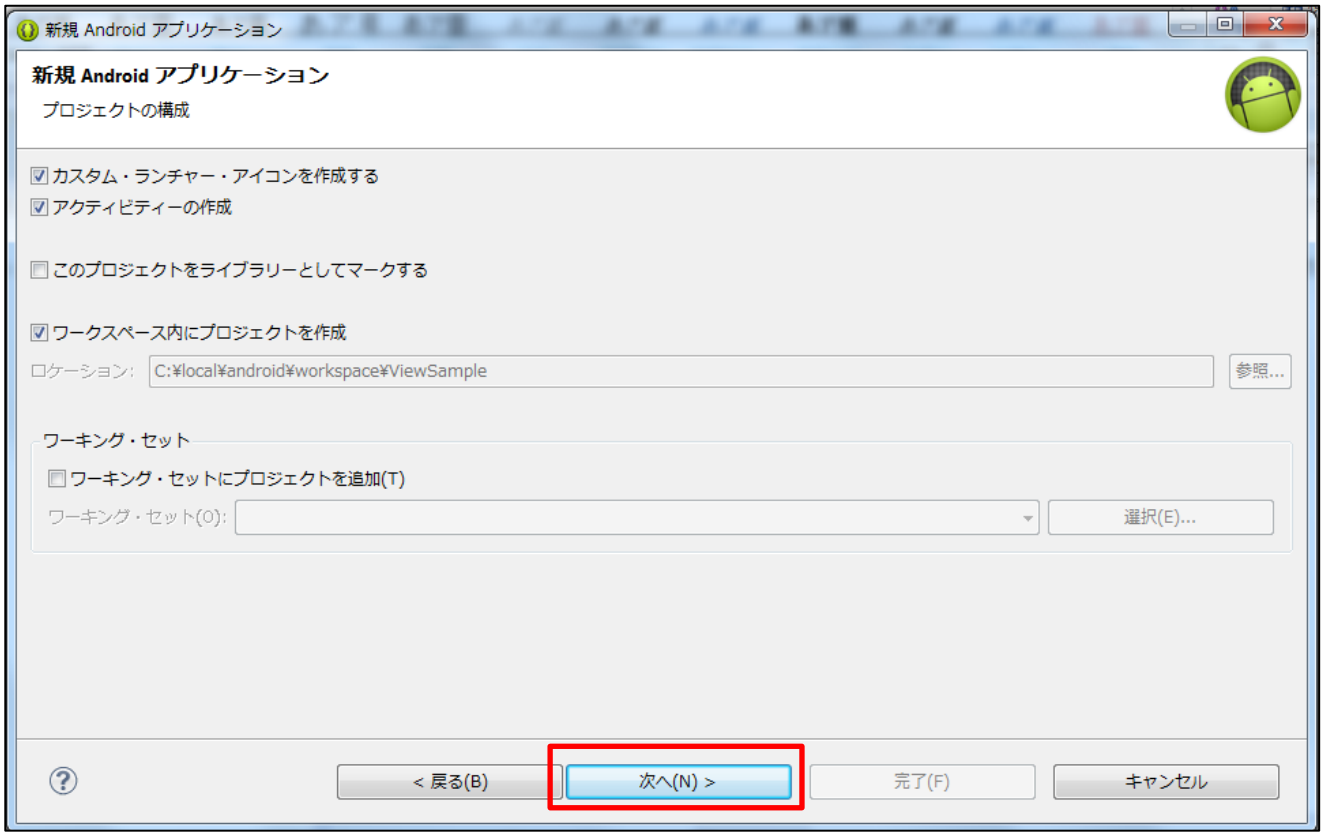

パッケージ・エクスプローラー
example
Android 4.2.2
Android Private Libraries
src
gen [Generated Java Files]
assets
bin
libs
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
drawable-xhdpi
drawable-xxhdpi
layout
menu
values
values-sw600dp
values-sw720dp-land
values-v11
values-v14
AndroidManifest.xml
ic_launcher-web.png
proguard-project.txt
project.properties
プロジェクト
ソースディレクトリ
リソースディレクトリ
画像格納用ディレクトリ
レイアウト格納用ディレクトリ
値、文字列、スタイル定義ファイル格納用ディレクトリ
マニフェストファイル

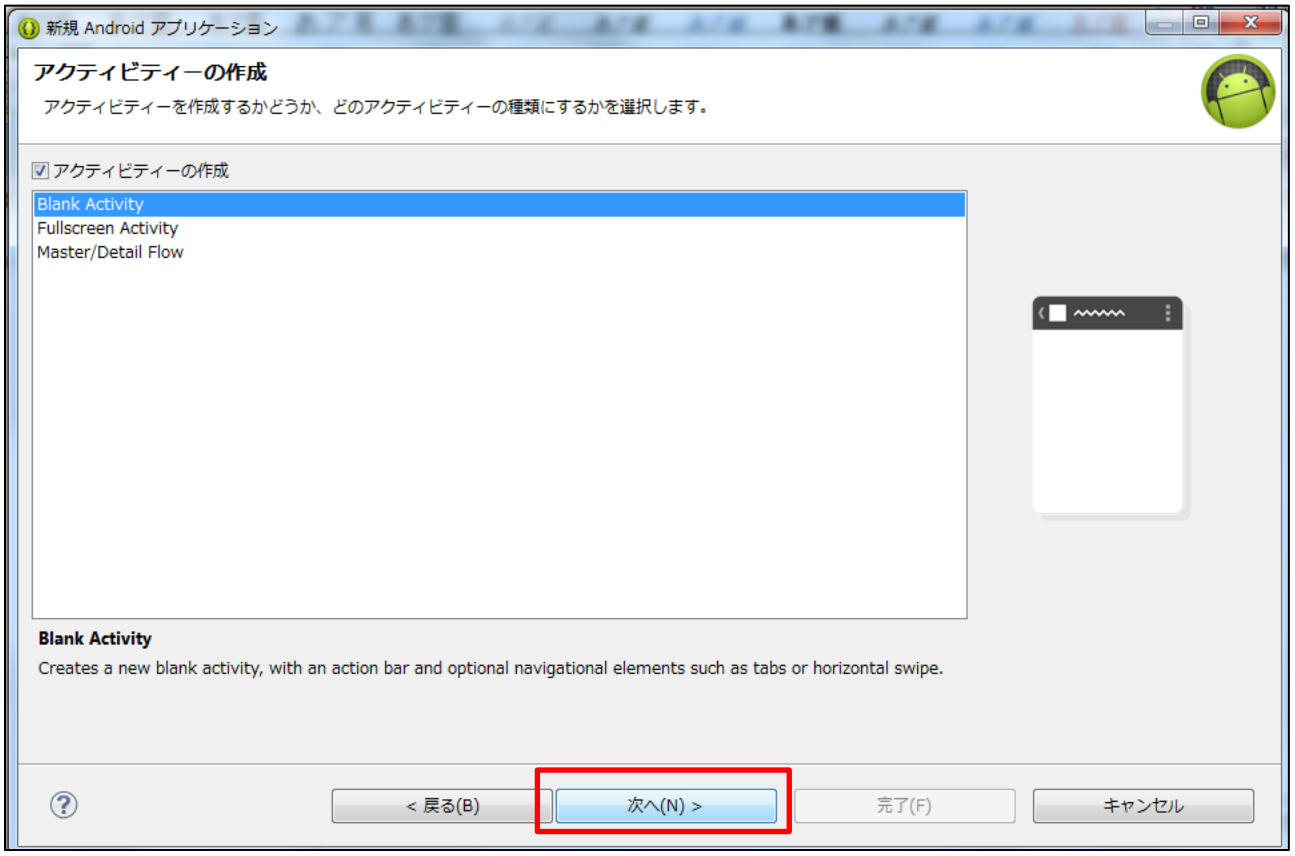
６．トレーニングアプリ開発①

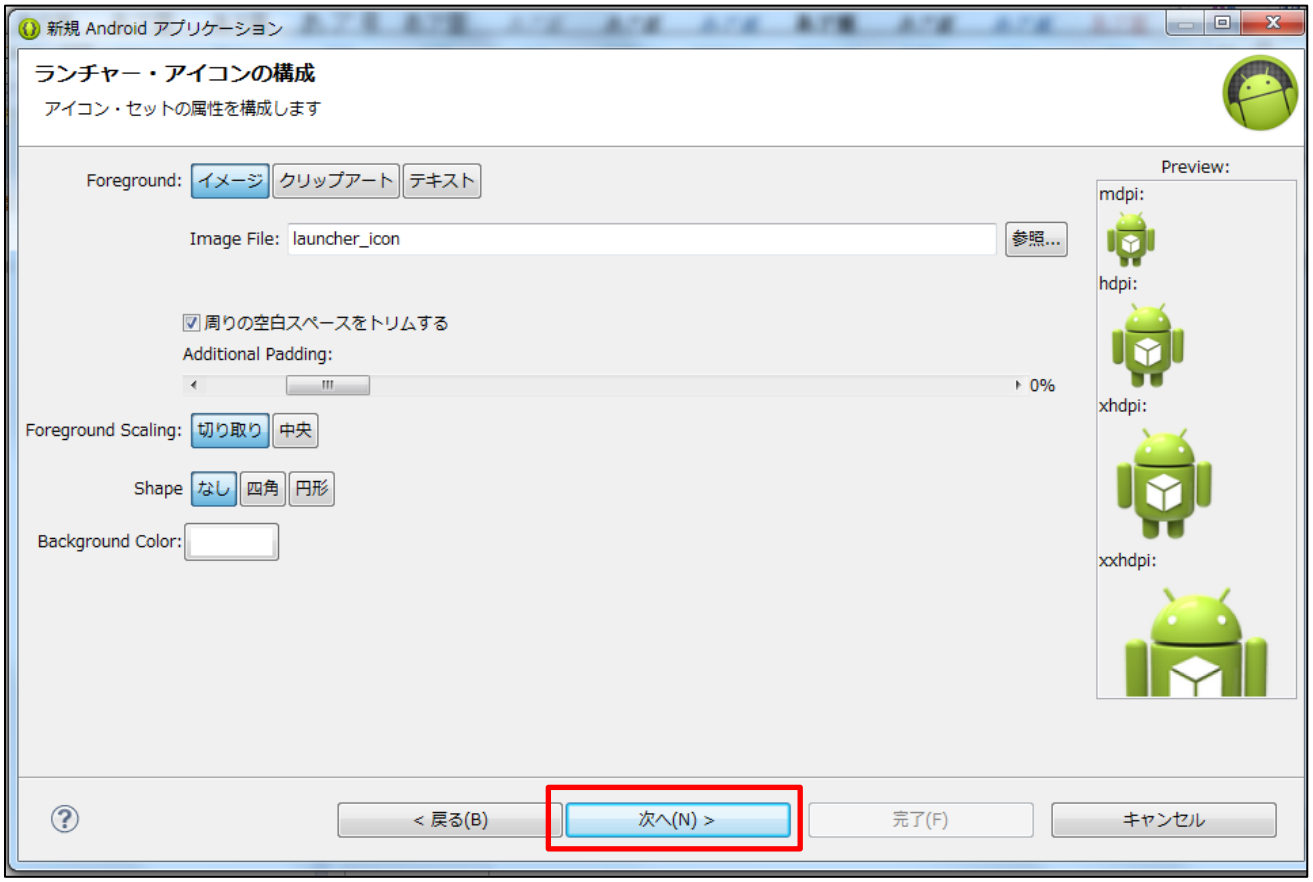
ここでは、チェックボックスとテキストボックスを表形式で配置するサンプルアプリケーションを作成します。

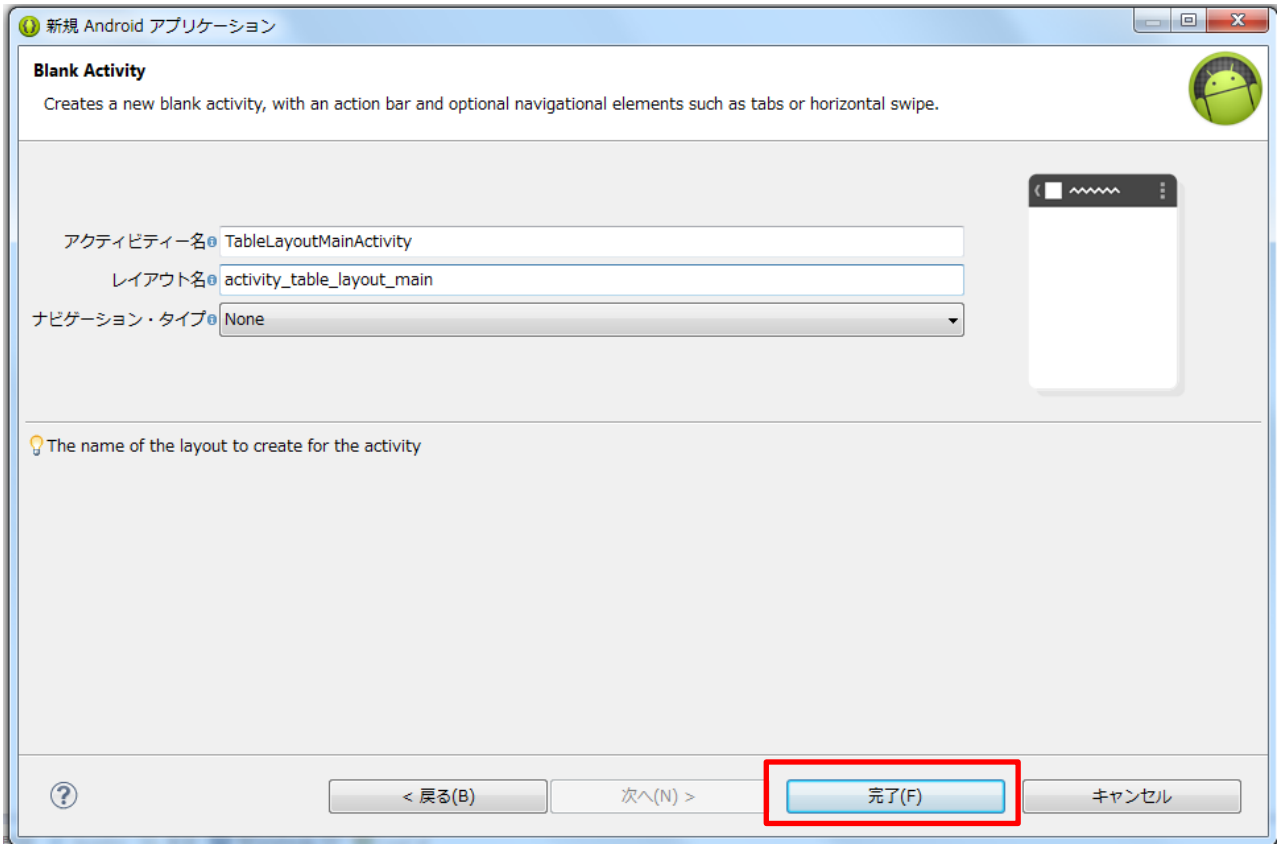
最初に「TableLayout」プロジェクトを次の設定で作成してください。






新規 Android アプリケーション
アクティビティーの作成
アクティビティーを作成するかどうか、どのアクティビティーの種類にするかを選択します。
アクティビティーの作成
Blank Activity
Fullscreen Activity
Master/Detail Flow
Blank Activity
Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.
< 戻る(B)
次へ(N) >
完了(F)
キャンセル

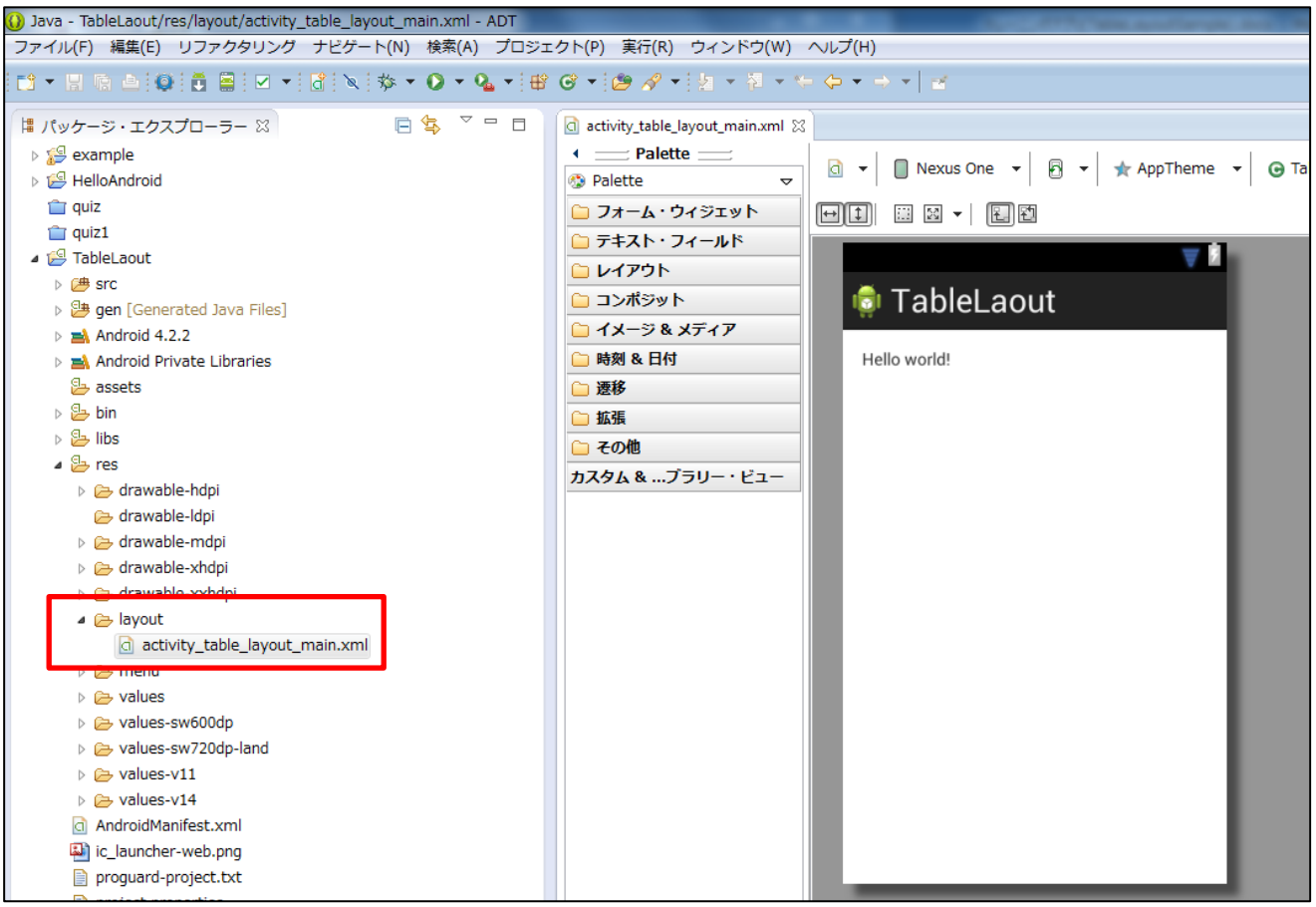

新規 Android アプリケーション
ランチャー・アイコンの構成
アイコン・セットの属性を構成します
Foreground:
イメージ
クリップアート
テキスト
Image File:
launcher_icon
参照...
周りの空白スペースをトリムする
Additional Padding:
0%
Foreground Scaling:
切り取り
中央
Shape
なし
四角
円形
Background Color:
Preview:
mdpi:
hdpi:
xhdpi:
xxhdpi:
< 戻る(B)
次へ(N) >
完了(F)
キャンセル


新規 Android アプリケーション
Blank Activity
Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.
アクティビティー名
TableLayoutMainActivity
レイアウト名
activity_table_layout_main
ナビゲーション・タイプ
None
The name of the layout to create for the activity
< 戻る(B)
次へ(N) >
完了(F)
キャンセル

XML ファイルに各ビューの定義を記述します。

（1）「Layout」→「activity_table_layout_main.xml」ファイルを選択する（図 1）。

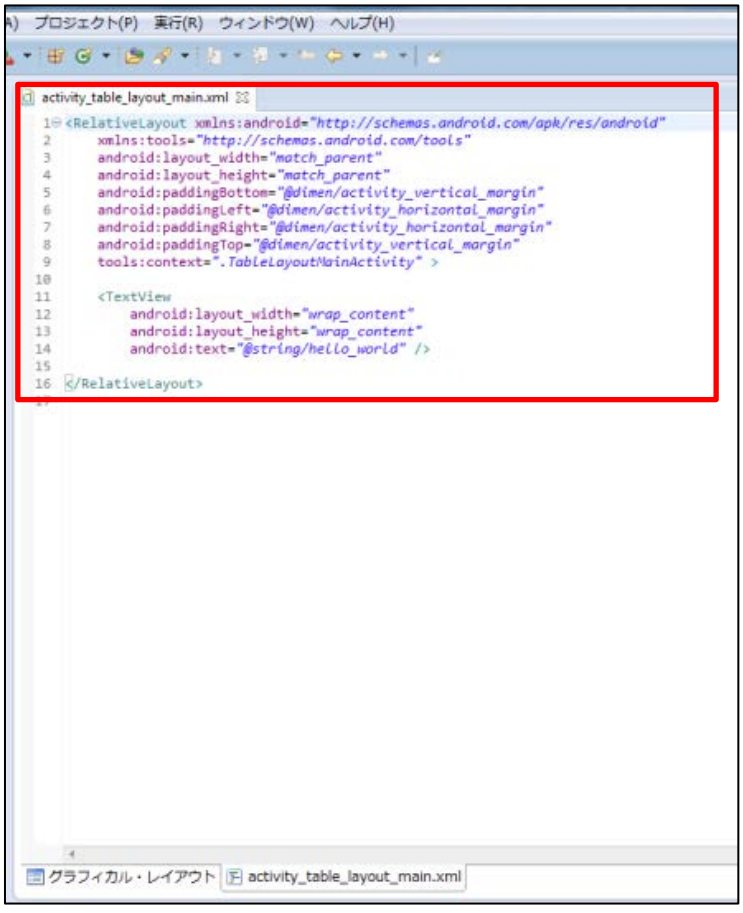
図 1



（２）デフォルト画面では「グラフィカルレイアウト」になっているので、「activity_table_layout_main.xml」を選択し（図２）、入力画面に変える（図３）。

図２

図３



（3）ソースコードを入力する。

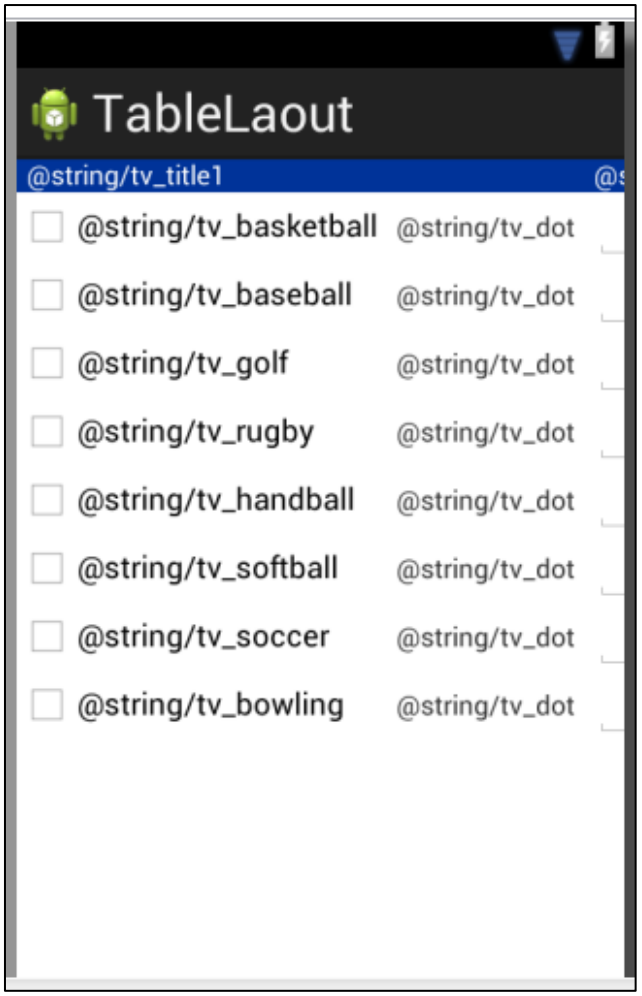
activity_table_layout_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1" >

    <TableRow android:background="#003399">
        <TextView
            android:layout_marginLeft="5dp"
            android:text="@string/tv_title1"
            android:textColor="#FFFFFF"
            android:layout_span="2" />
        <TextView
            android:layout_marginRight="5dp"
            android:text="@string/tv_title2"
            android:textColor="#FFFFFF" />
    </TableRow>
    <TableRow>
        <CheckBox android:text="@string/tv_basketball" />
        <TextView
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:text="@string/tv_dot" />
        <EditText android:inputType="number" />

    </TableRow>
    <TableRow>
        <CheckBox android:text="@string/tv_baseball" />
        <TextView
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:text="@string/tv_dot" />
        <EditText android:inputType="number" />
    </TableRow>
    <TableRow>
        <CheckBox android:text="@string/tv_golf" />
        <TextView
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:text="@string/tv_dot" />
        <EditText android:inputType="number" />
    </TableRow>
    <TableRow>
        <CheckBox android:text="@string/tv_rugby" />
        <TextView
```

```
                android:layout_marginLeft="10dp"
                android:layout_marginRight="10dp"
                android:text="@string/tv_dot" />
            <EditText android:inputType="number" />
        </TableRow>
        <TableRow>
            <CheckBox android:text="@string/tv_handball" />
            <TextView
                android:layout_marginLeft="10dp"
                android:layout_marginRight="10dp"
                android:text="@string/tv_dot" />
            <EditText
                android:inputType="number" />
        </TableRow>
        <TableRow>
            <CheckBox android:text="@string/tv_softball" />
            <TextView
                android:layout_marginLeft="10dp"
                android:layout_marginRight="10dp"
                android:text="@string/tv_dot" />
            <EditText android:inputType="number" />
        </TableRow>
        <TableRow>
            <CheckBox android:text="@string/tv_soccer" />
            <TextView
                android:layout_marginLeft="10dp"
                android:layout_marginRight="10dp"
                android:text="@string/tv_dot" />
            <EditText android:inputType="number" />

        </TableRow>
        <TableRow>
            <CheckBox android:text="@string/tv_bowling" />
            <TextView
                android:layout_marginLeft="10dp"
                android:layout_marginRight="10dp"
                android:text="@string/tv_dot" />
            <EditText android:inputType="number" />
        </TableRow>

</TableLayout>
```

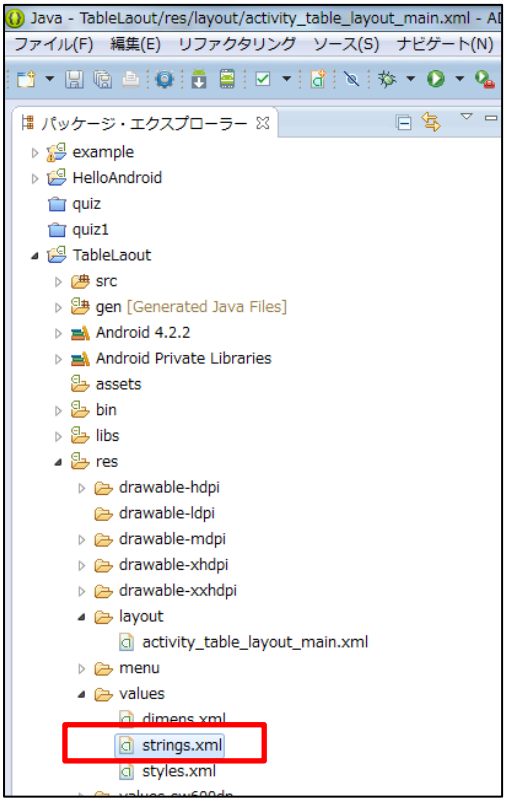
（4）「グラフィカルレイアウト」をクリックし、画面を確認する（図４）。

図４



各ビューに表示する文字列を文字列設定ファイル strings.xml で設定する。

（1）「values」→「strings.xml」ファイルを選択し、ソースコードを入力する。
図 5



strings.xml

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

    <string name="app_name">TableLayoutSample</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>

    <!-- 各 View で使用する文字列の定義 -->
    <string name="tv_title1">やりたいスポーツ</string>
    <string name="tv_title2">順位</string>
    <string name="tv_basketball">バスケットボール</string>
    <string name="tv_baseball">野球</string>
    <string name="tv_golf">ゴルフ</string>
    <string name="tv_rugby">ラグビー</string>
    <string name="tv_handball">ハンドボール</string>
    <string name="tv_softball">ソフトボール</string>
    <string name="tv_soccer">サッカー</string>
    <string name="tv_bowling">ボウリング</string>
    <string name="tv_dot">・・・・・・・・</string>
</resources>
```

（2）「グラフィカルレイアウト」をクリックし、画面を確認する（図６）。

図６

（3）画面を制御するアクティビティクラスは、自動生成されたソースのまま変更はありません。

TableLayoutSampleActivity.java

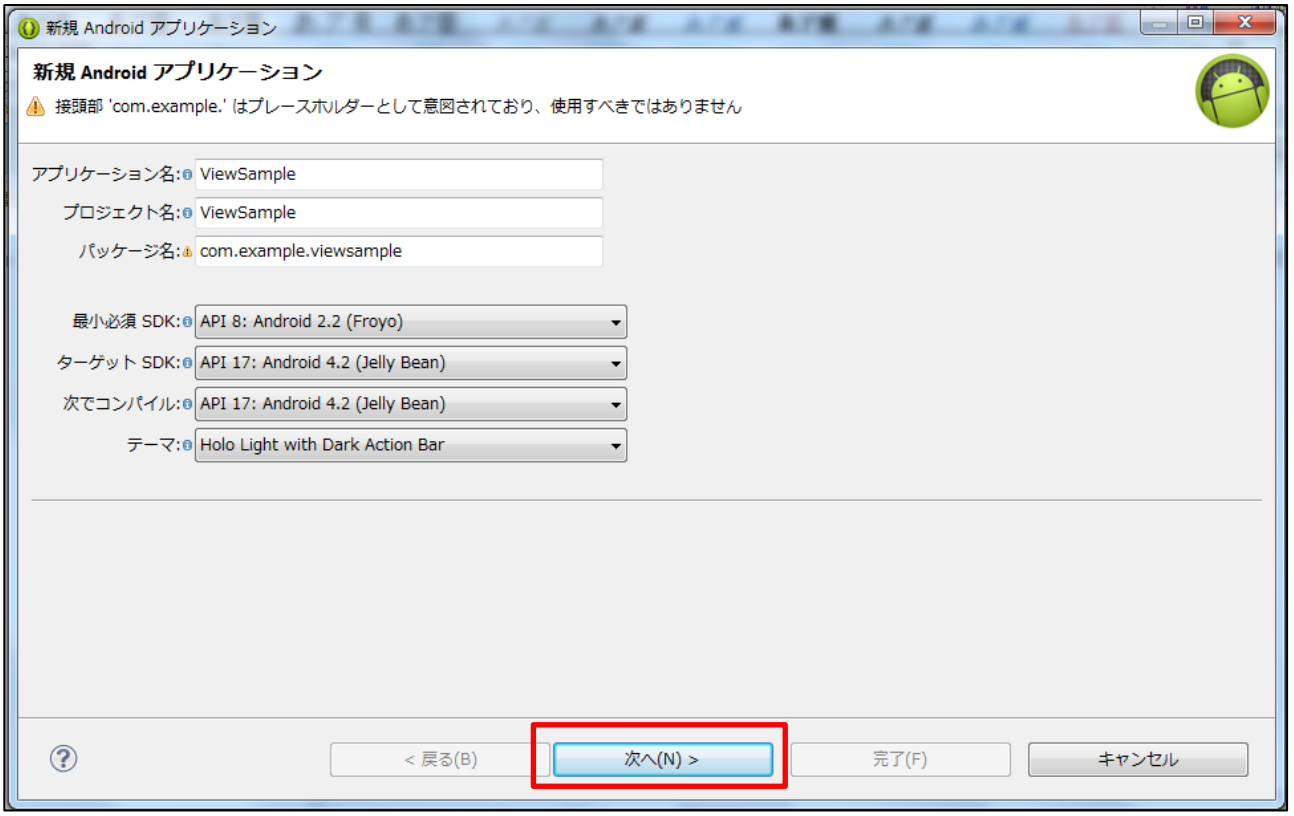
```
package example.android.tablelayoutsample;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class TableLayoutSampleActivity extends Activity {

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

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

}
```

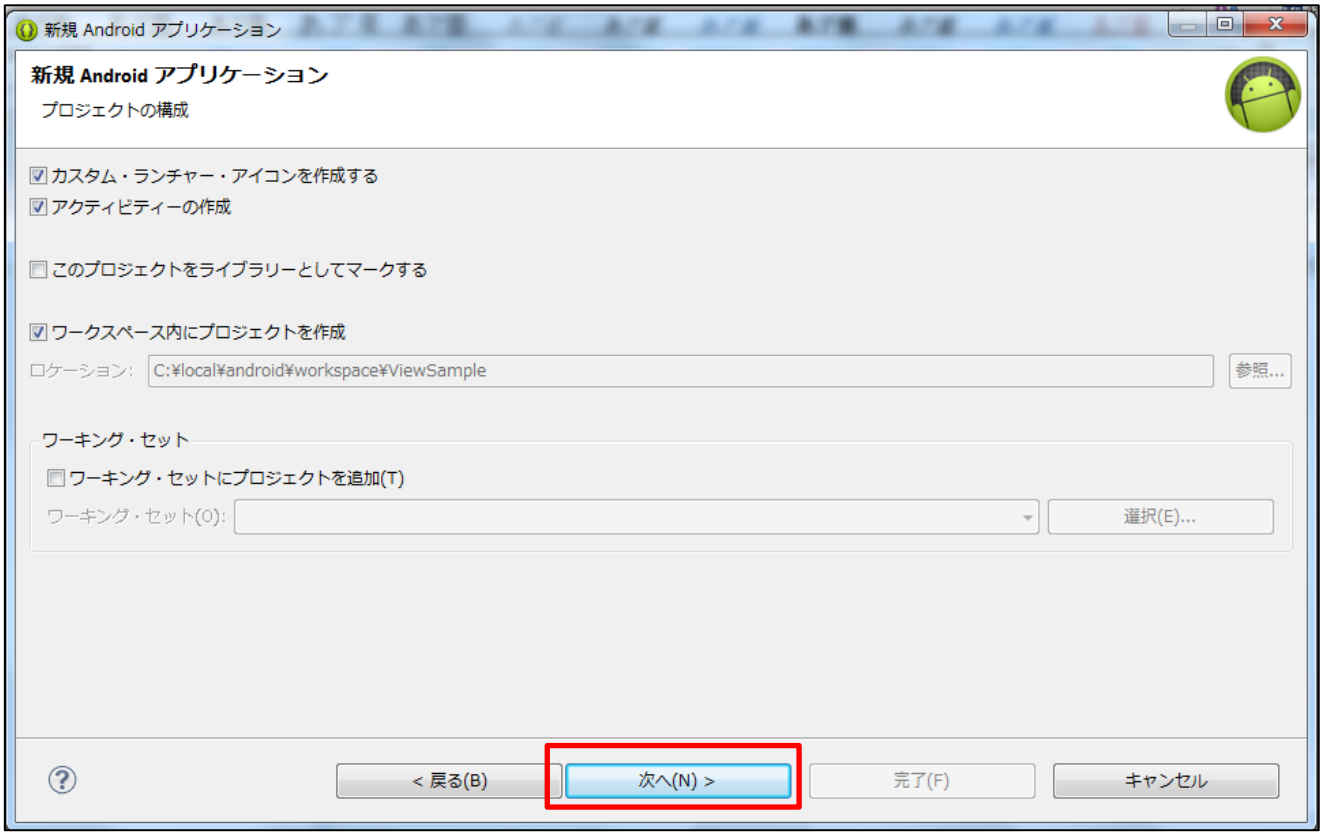
7．トレーニングアプリ開発②

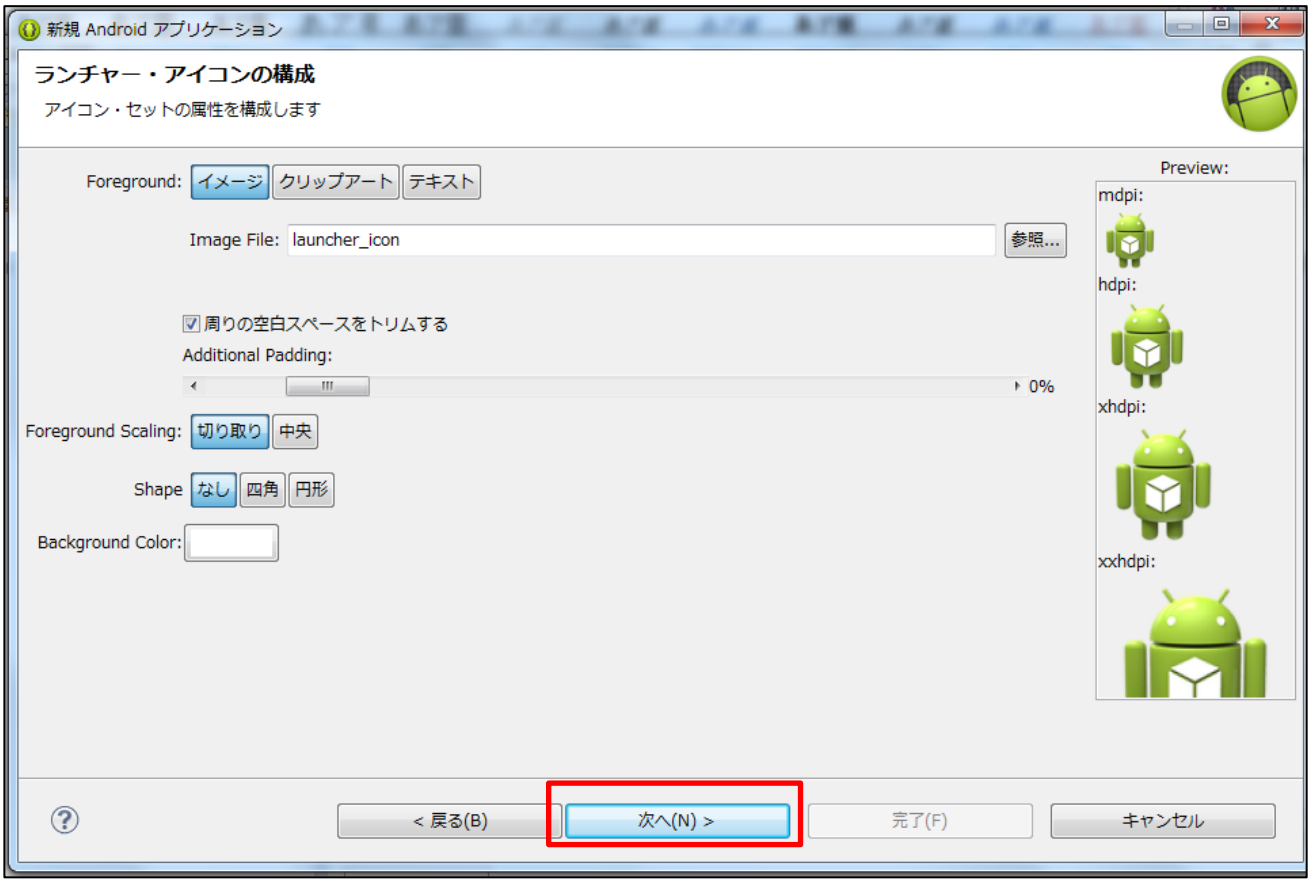
サンプルアプリケーションの作成

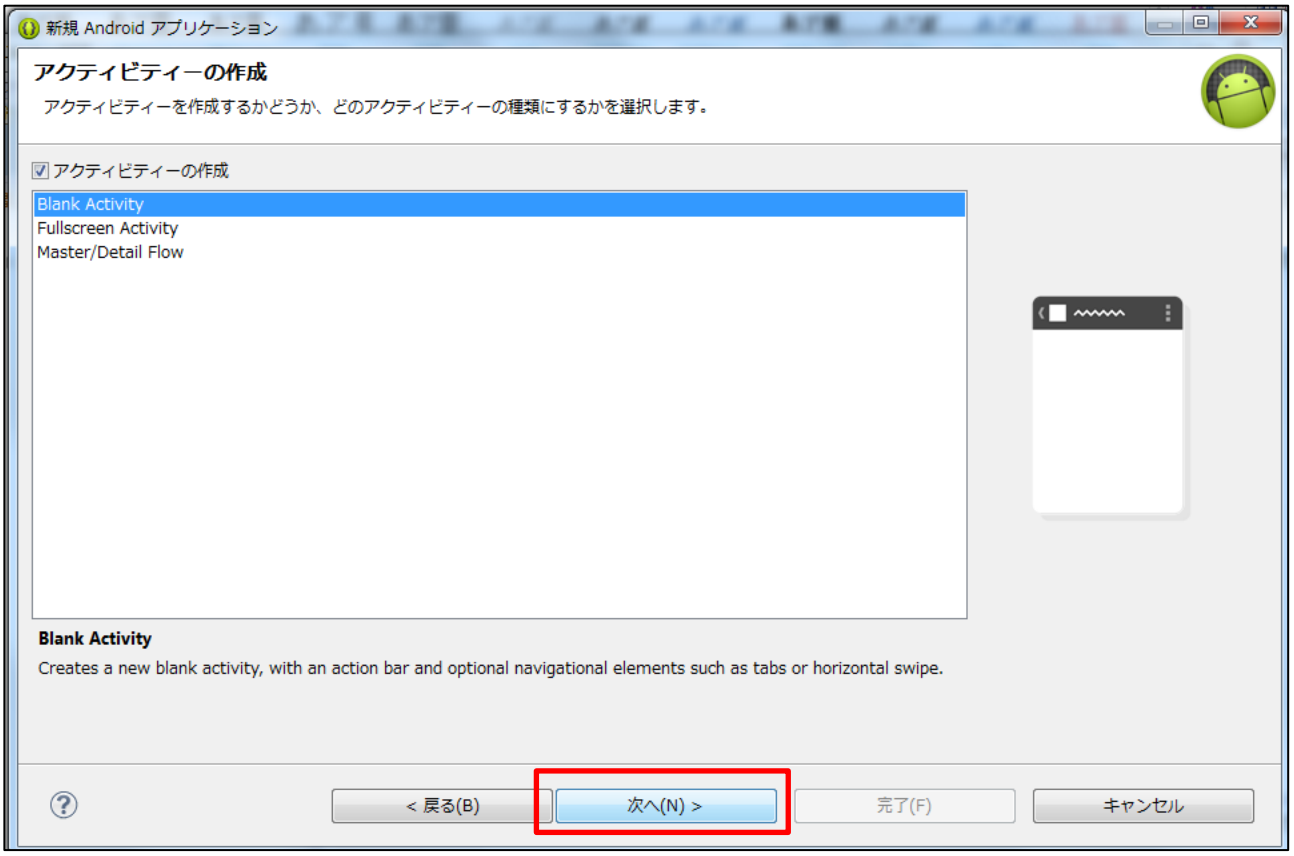
　代表的なビューを使用したサンプルアプリケーションを作成しましょう。その完成イメージを示します。

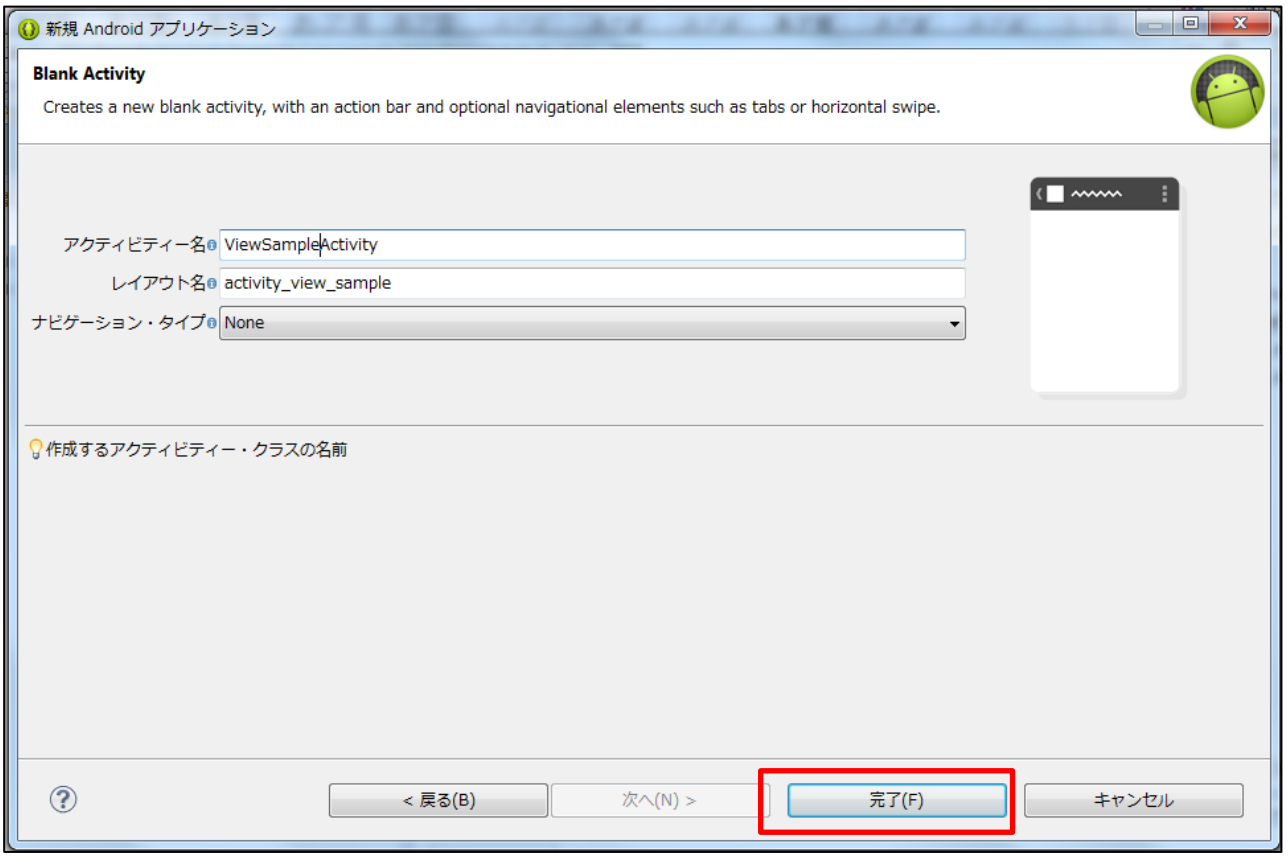
「ViewSample」プロジェクトを作成します。






新規 Android アプリケーション
ランチャー・アイコンの構成
アイコン・セットの属性を構成します
Foreground:
イメージ
クリップアート
テキスト
Image File:
launcher_icon
参照...
周りの空白スペースをトリムする
Additional Padding:
0%
Foreground Scaling:
切り取り
中央
Shape
なし
四角
円形
Background Color:
Preview:
mdpi:
hdpi:
xhdpi:
xxhdpi:
< 戻る(B)
次へ(N) >
完了(F)
キャンセル


新規 Android アプリケーション
アクティビティーの作成
アクティビティーを作成するかどうか、どのアクティビティーの種類にするかを選択します。
アクティビティーの作成
Blank Activity
Fullscreen Activity
Master/Detail Flow
Blank Activity
Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.
< 戻る(B)
次へ(N) >
完了(F)
キャンセル


新規 Android アプリケーション
Blank Activity
Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.
アクティビティー名
ViewSampleActivity
レイアウト名
activity_view_sample
ナビゲーション・タイプ
None
作成するアクティビティー・クラスの名前
< 戻る(B)
次へ(N) >
完了(F)
キャンセル

（１）「res」フォルダの「Layout」フォルダのレイアウトとビューの設定ファイル「Activity_view_Sample_xml」に各ビューの定義を記述します。

Activity_view_sample.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- 画面全体のスクロールバーの定義 -->
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <!-- レイアウトの定義 -->
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical" >

            <!-- TextView(テキスト)の定義 -->
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="5dp"
                android:layout_marginBottom="25dp"
                android:text="@string/tv_viewsample" />

            <!-- EditText(テキストボックス)の定義 -->
            <EditText
                android:id="@+id/editText1"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="5dp"
                android:layout_marginBottom="25dp"
                android:inputType="text" />

                <!-- CheckBox(チェックボックス)の定義 -->
            <CheckBox
                android:id="@+id/checkBox1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:paddingTop="10dp"
                android:paddingBottom="10dp"
                android:text="@string/cb_sample1" />

            <CheckBox
                android:id="@+id/checkBox2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/cb_sample2" />
```

```
        <!-- RadioButton(ラジオボタン)の定義 -->
        <RadioGroup
            android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:paddingTop="10dp"
            android:paddingBottom="10dp"
                android:orientation="vertical" >
            <RadioButton
                    android:id="@+id/radioButton1"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/rb_sample1" />

                <RadioButton
                    android:id="@+id/radioButton2"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/rb_sample2" />
        </RadioGroup>

            <!-- Spinner(選択ボックス)の定義 -->
        <Spinner
            android:id="@+id/spinner1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingTop="25dp"
            android:paddingBottom="5dp"
            android:entries="@array/lv_sample1" />

        <!-- ListView(リスト一覧)の定義 -->
        <ListView
            android:id="@+id/listView1"
            android:layout_width="match_parent"
            android:layout_height="250dp"
            android:paddingTop="25dp"
            android:paddingBottom="5dp"
            android:entries="@array/lv_sample1" />

        <!-- Button(ボタン)の定義 -->
        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/bt_sample1" />

    </LinearLayout>

</ScrollView>
```

（2）各ビューに表示する文字列は、「res」フォルダの「values」フォルダの中にある「strings.xml」で設定します。この文字列設定ファイルは、文字列情報を一元管理するために使用する設定ファイルです。この XML ファイルを開いて、次の通りに追記しましょう。

strings.xml

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

    <string name="app_name">ViewSample</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>

            <!-- 各 View で使用する文字列の定義 -->
    <string name="tv_viewsample">View のサンプルです。</string>
    <string name="cb_sample1">チェックボックス 1</string>
    <string name="cb_sample2">チェックボックス 2</string>
    <string name="rb_sample1">ラジオボタン 1</string>
    <string name="rb_sample2">ラジオボタン 2</string>
    <string name="bt_sample1">保存</string>

    <string-array name="lv_sample1">
        <item>あいうえお</item>
        <item>かきくけこ</item>
        <item>さしすせそ</item>
    </string-array>

</resources>
```

（3）画面を制御するアクティビティクラスは、自動生成されたソースのまま変更はありません。
「src」→「example.android.viewsample」の「ViewSampleActivity.java」がアクティビティクラスになります。

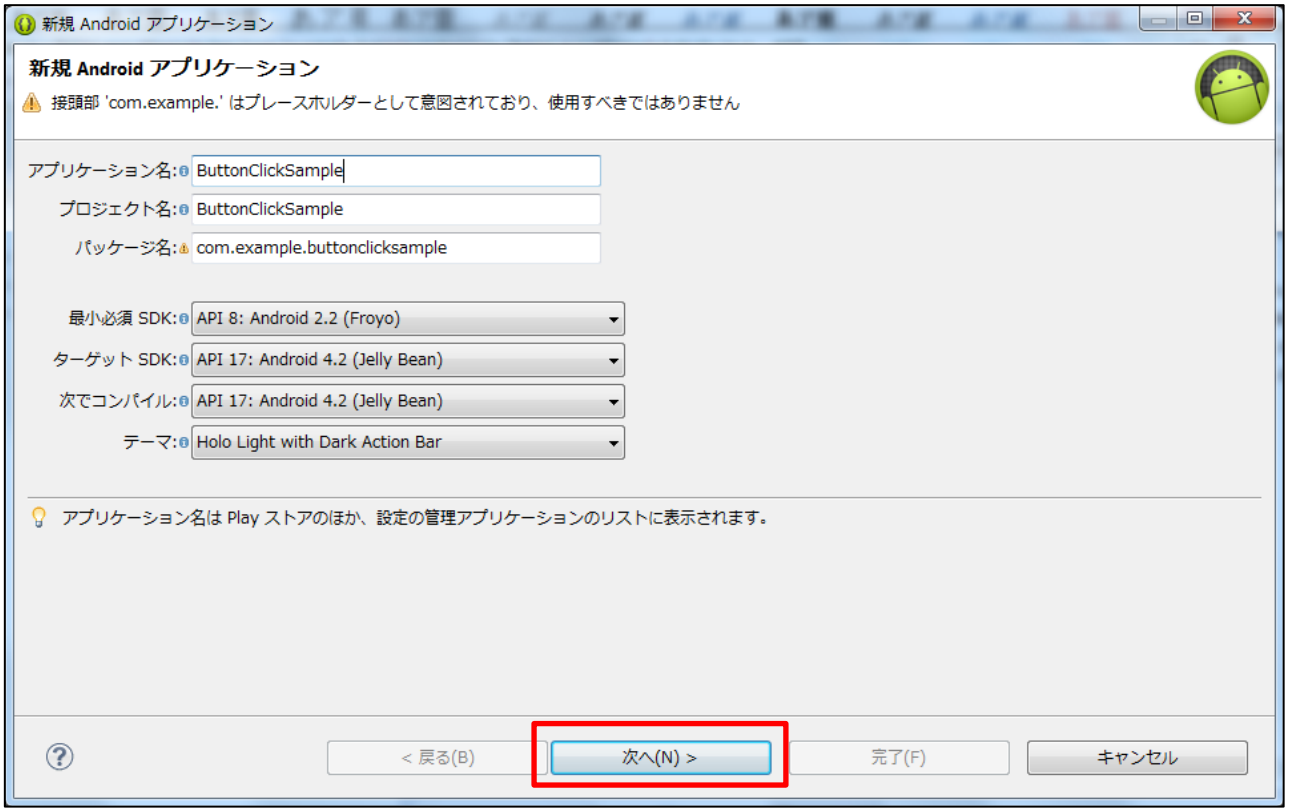
ViewSampleActivity.java

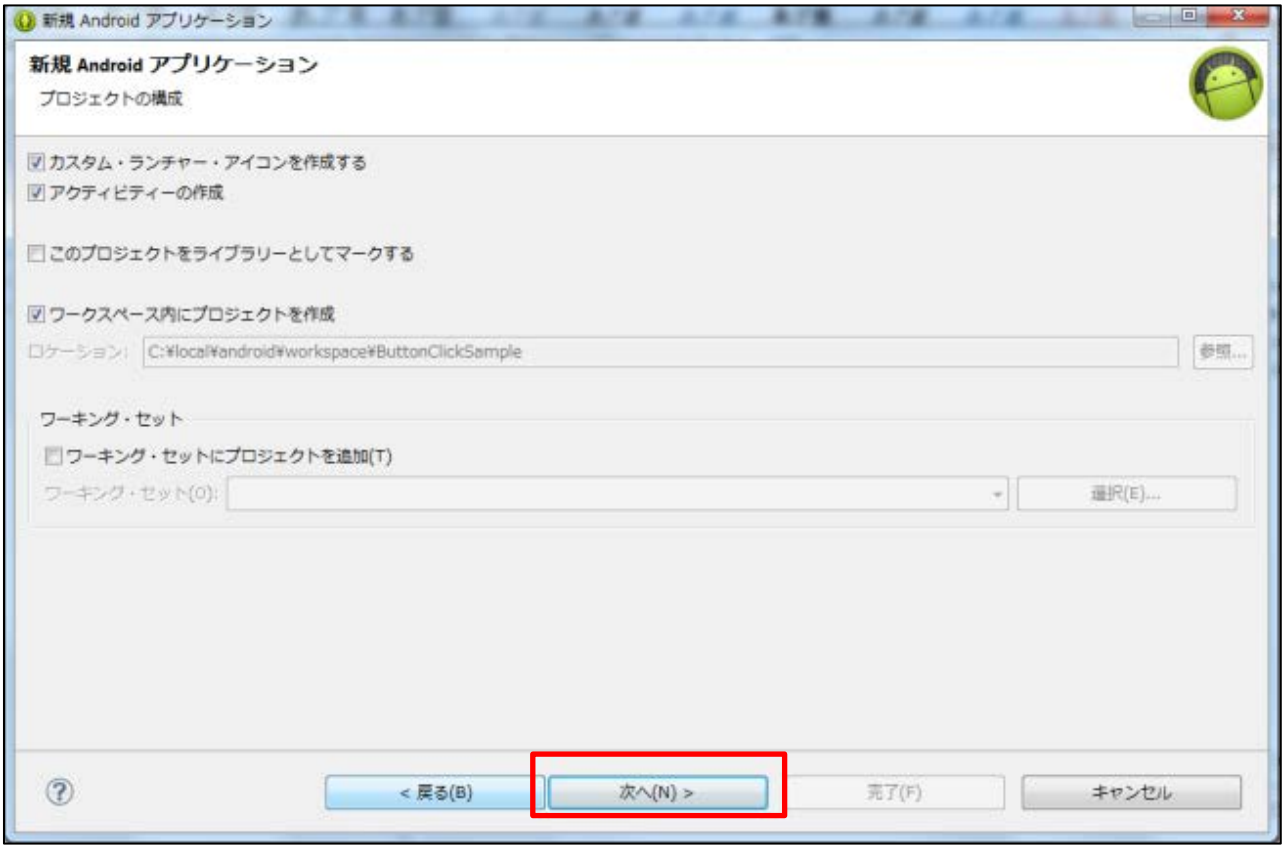
```
package example.android.viewsample;

import android.os.Bundle;
import android.app.Activity;

import android.view.Menu;

public class ViewSampleActivity extends Activity {

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

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

}
```

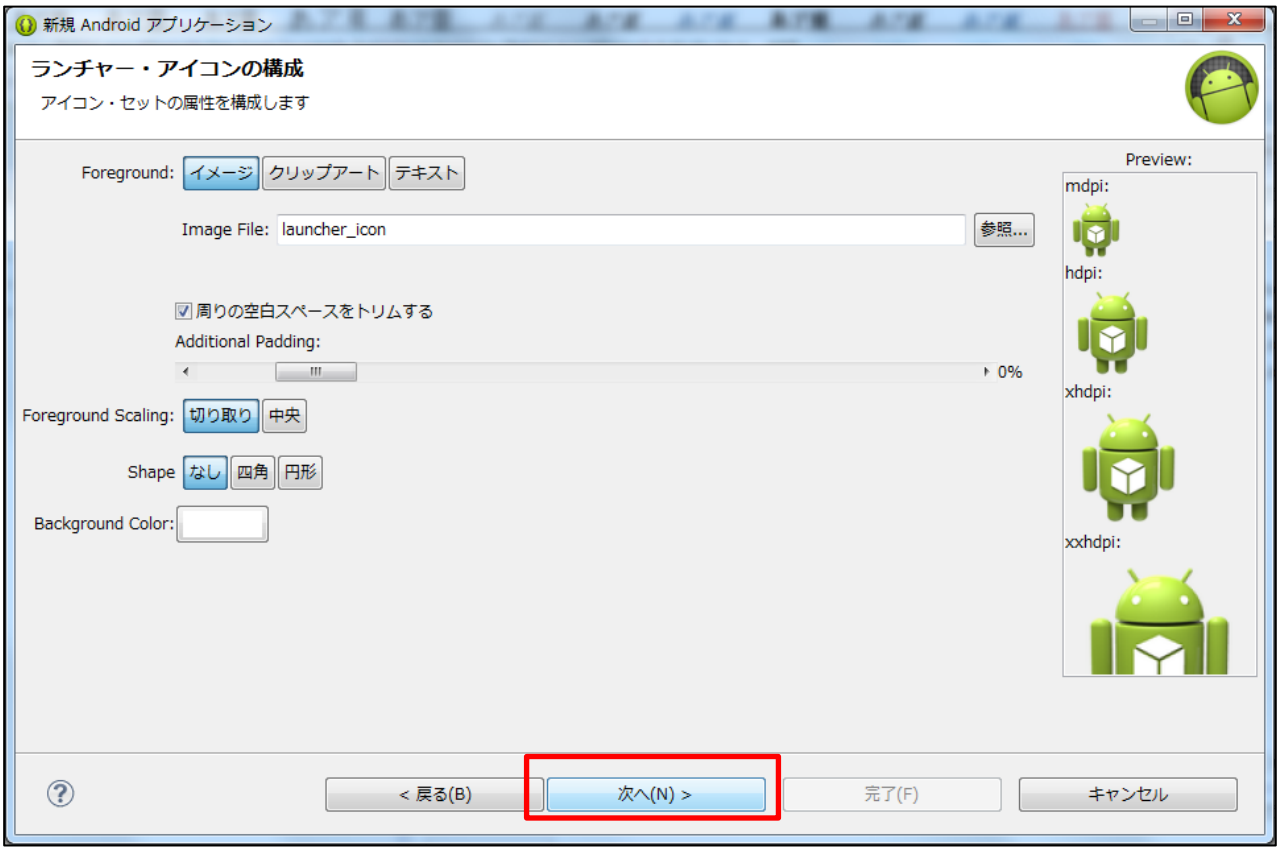
８．トレーニングアプリ開発③

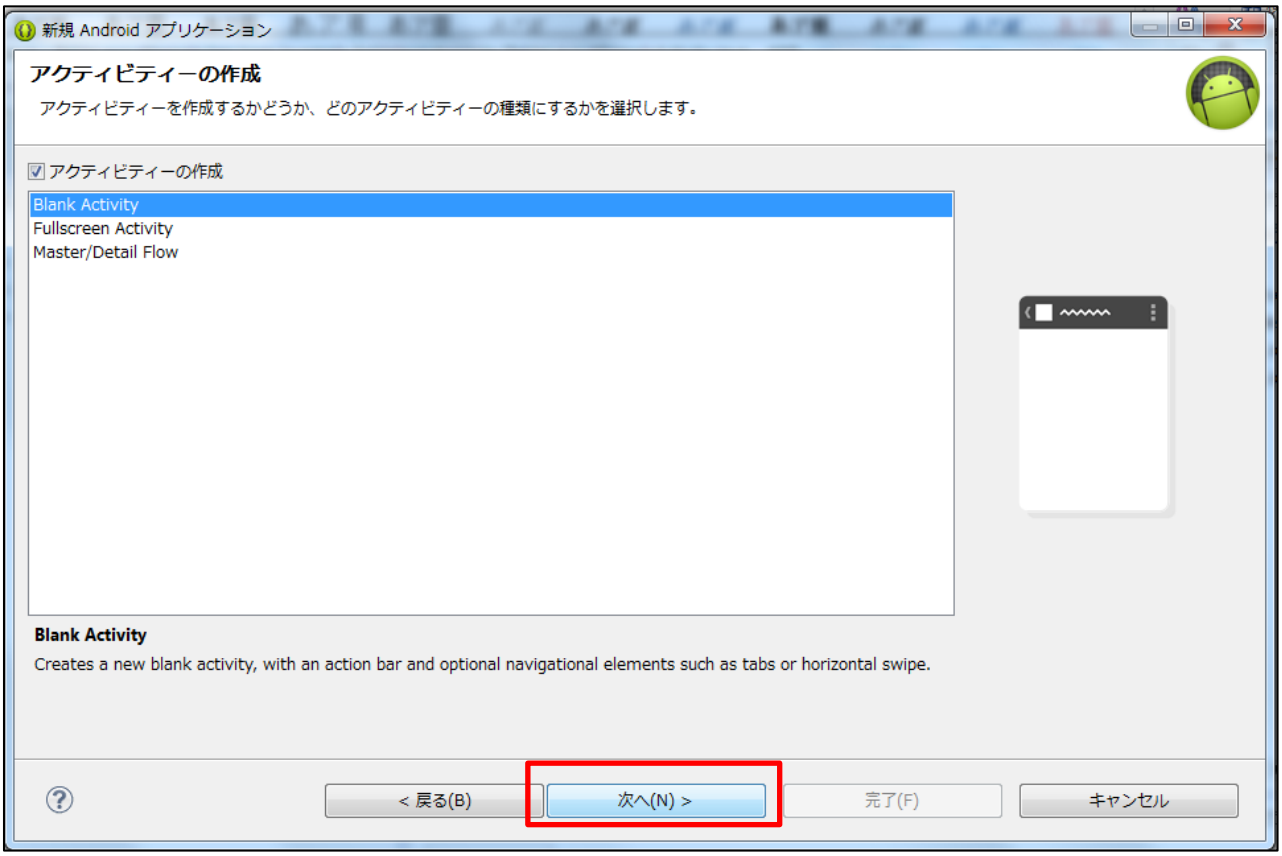
ここで開発アプリケーションは、画面に表示されたテキストボックスに文字を入力し、ボタンをクリックすると、入力された文字が画面の下にポップアップ表示されるというものです。

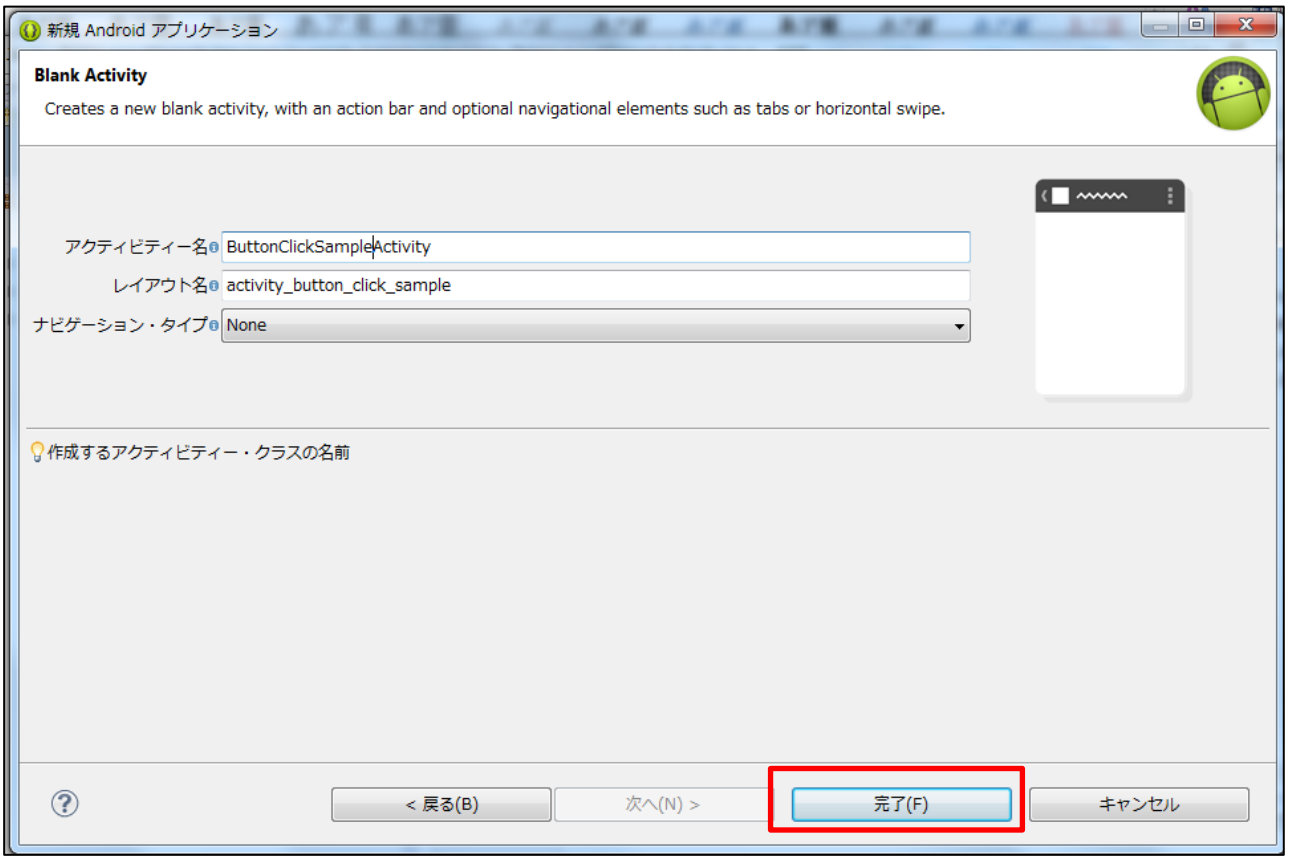
「ButtonClickSample」プロジェクトを作成します。






新規 Android アプリケーション
ランチャー・アイコンの構成
アイコン・セットの属性を構成します
Foreground:
イメージ
クリップアート
テキスト
Image File:
launcher_icon
参照...
周りの空白スペースをトリムする
Additional Padding:
0%
Foreground Scaling:
切り取り
中央
Shape
なし
四角
円形
Background Color:
Preview:
mdpi:
hdpi:
xhdpi:
xxhdpi:
< 戻る(B)
次へ(N) >
完了(F)
キャンセル


新規 Android アプリケーション
アクティビティーの作成
アクティビティーを作成するかどうか、どのアクティビティーの種類にするかを選択します。
アクティビティーの作成
Blank Activity
Fullscreen Activity
Master/Detail Flow
Blank Activity
Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.
< 戻る(B)
次へ(N) >
完了(F)
キャンセル


新規 Android アプリケーション
Blank Activity
Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.
アクティビティー名
ButtonClickSampleActivity
レイアウト名
activity_button_click_sample
ナビゲーション・タイプ
None
作成するアクティビティー・クラスの名前
< 戻る(B)
次へ(N) >
完了(F)
キャンセル

（1）レイアウトとビューの設定ファイル「activity_botton_click_sample.xml」に各ビューの定義を記述します。

Activity_button_click_sample.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/tv_name"/>

        <EditText
            android:id="@+id/et_name"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="text" />

        <Button
            android:id="@+id/bt_click"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/bt_click"/>

</LinearLayout>
```

（２）「strings.xml」には、画面に表示する文字列を定義するため、次の通り追記をします。

Strings.xml

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

    <string name="app_name">ButtonClickSample</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>

    <string name="tv_name">お名前を入力してください。</string>
    <string name="bt_click">ボタンクリック！</string>

  </resources>
```

（3）アクティビティクラスには、次の通りに記述します。

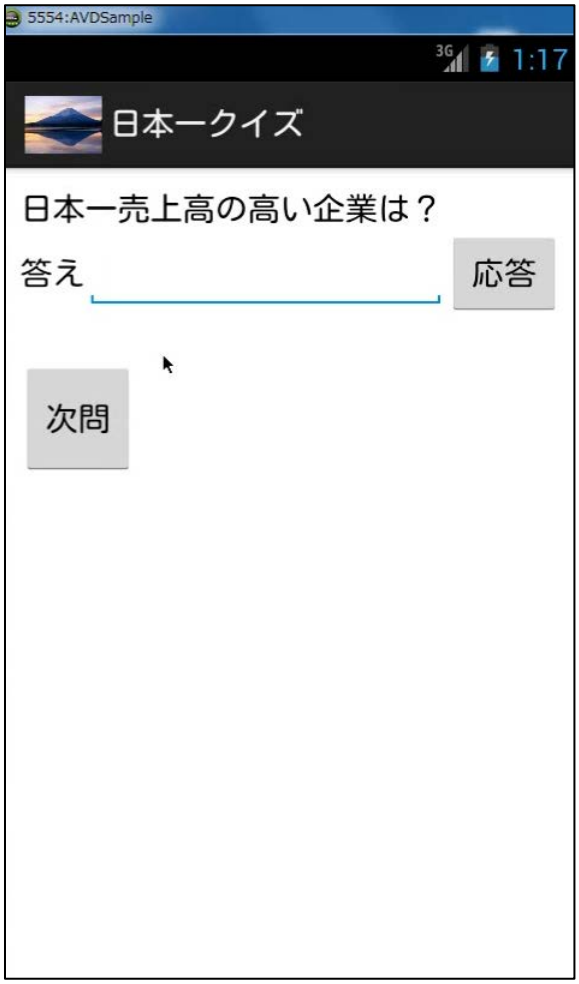
ButtonClickSampleActivity.java

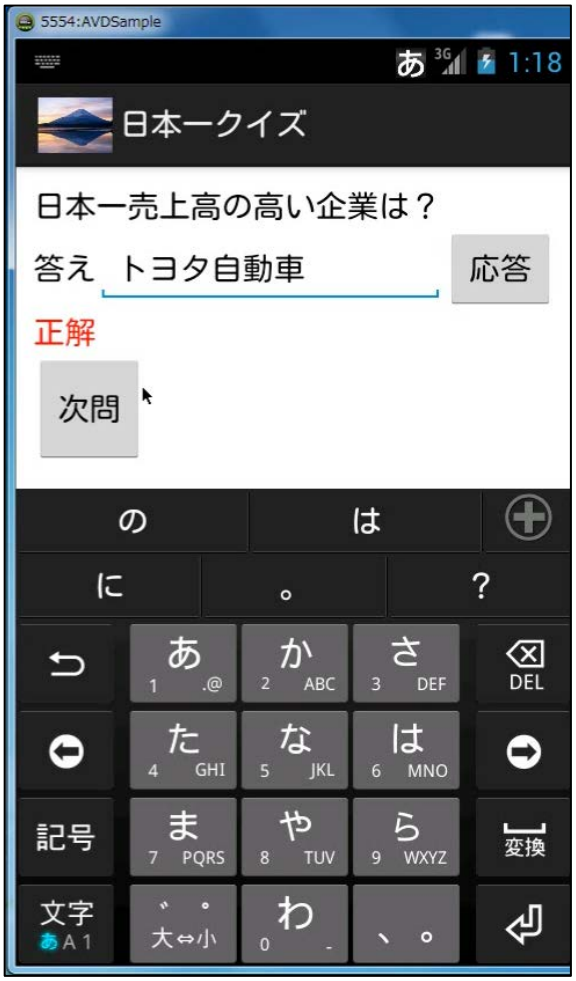
```
package example.android.buttonclicksample;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class ButtonClickSampleActivity extends Activity {

	// onCreate メソッド(画面初期表示イベントハンドラ)
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// スーパークラスの onCreate メソッド呼び出し
		super.onCreate(savedInstanceState);
		// レイアウト設定ファイルの指定
		setContentView(R.layout.activity_button_click_sample);

		// ボタンオブジェクト取得
		Button button = (Button)findViewById(R.id.bt_click);
		// ボタンオブジェクトにクリックリスナー設定
		button.setOnClickListener(new ButtonClickListener());
	}

	// クリックリスナー定義
	class ButtonClickListener implements OnClickListener {
		// onClick メソッド(ボタンクリック時イベントハンドラ)
		public void onClick(View v) {
			// テキストボックスオブジェクト取得
			EditText input = (EditText)findViewById(R.id.et_name);
			// 入力情報をトースト機能で画面表示
			Toast.makeText(ButtonClickSampleActivity.this,
					input.getText(),
				Toast.LENGTH_SHORT).show();
		}
	}

	// onCreateOptionsMenu メソッド(オプションメニュー生成)
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
	// オプションメニューを設定
		getMenuInflater().inflate(R.menu.button_click_sample, menu);
```

```
                return true;
        }

}
```

９．トレーニングアプリ開発④

下のようなクイズアプリを作成してください。
（６）のソースコードは穴埋め問題になっています。答えを考えて入力し、アプリを完成させてください。





（1）main.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:background="@color/baseColor"
    android:orientation="vertical"
    android:padding="10dip" >

<TextView
    android:id="@+id/question"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/black"
    android:textSize="18dip"/>
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="@color/black"
    android:textSize="18dip"
    android:text="答え"/>
<EditText
    android:id="@+id/answer"
    android:layout_width="200dip"
    android:layout_height="wrap_content"/>
<Button
    android:id="@+id/reply"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="onTouch"
    android:text="応答"/>
    </LinearLayout>
<TextView
    android:id="@+id/judge"
    android:layout_width="wrap_content"



    android:layout_height="wrap_content"
    android:textColor="@color/red"
    android:textSize="18dip" />

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="78dp"
```

```
        android:gravity="right|bottom"
        android:orientation="horizontal" >

</LinearLayout>

<Button
        android:id="@+id/ans"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onTouch"
        android:text="答え" />

<Button
        android:id="@+id/next"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onTouch"
        android:text="次問" />


</LinearLayout>
```

（２）strings.xml

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

    <string name="app_name">日本一クイズ</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>

</resources>
```

（３）colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="baseColor">#FFFFFF</color>
        <color name="black">#000000</color>
        <color name="red">#FF0000</color>
</resources>
```

（４）WatchwordCont.java

```
package jp.gihyo.mitoma.chap01_02;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;

public class WatchwordCont extends Activity {
        private WatchwordView view;
        private Watchword model;

    @Override
    public void onCreate(Bundle bundle){
            super.onCreate(bundle);
            this.view=new WatchwordView(this);
            this.model=new Watchword();
            //最初の質問の表示
            this.model.exeQuestion();
            this.view.exeDisplay(model);
    }
    public void onTouch(View v){
            int id=v.getId();
            if(id==R.id.reply){
//応答ボタンタッチのときの処理
                    String ans=this.view.getAnswer();
                    this.model.exeJudge(ans);
                    this.view.exeDisplay(model);
            }
            if(id == R.id.ans){
                    this.model.exeans();
                    this.view.exeDisplay(model);
                    this.view.setAnswer("");
            }
            if(id==R.id.next){
                    //次問ボタンタッチのときの処理
                    this.model.exeQuestion();
                    this.view.exeDisplay(model);



this.view.setAnswer("");
            }
    }
}
```

（５）WatchwordView.java

```
package jp.gihyo.mitoma.chap01_02;

import android.app.Activity;
import android.widget.EditText;
import android.widget.TextView;



public class WatchwordView {
        private Activity act;
        private TextView question;
        private EditText answer;
        private TextView judge;

        public WatchwordView(Object obj){
                //Activity オブジェクト変数の確保
                this.act=(Activity)obj;
                this.act.setContentView(R.layout.main);
                //UI エレメントのオブジェクト変数のセット
                this.question=(TextView)this.act.findViewById(R.id.question);
                this.answer=(EditText)this.act.findViewById(R.id.answer);
                this.judge=(TextView)this.act.findViewById(R.id.judge);
        }
public String getAnswer(){return this.answer.getText().toString();}
public void setAnswer(String text){this.answer.setText(text);}

public void exeDisplay(Watchword model){
        this.question.setText(model.getQuestion());
        this.judge.setText(model.getJudge());
}
}
```

（6）Watchword.java

```
1  package jp.gihyo.mitoma.chap01_02;
2  //乱数を作成する。
3  import java.util.　　①　　;
4
5  public class Watchword {
6                                          //質問集を配列で持つ。
7          Private final static String[]qArray={　　②　　};
8                                          //質問に対する正解を配列で持つ。
9          Private final static String[]aArray={　　③　　};
10         private int qNo;
11         private String 　④　;
12         private String 　⑤　;
13         private String 　⑥　;
14         private Random random;
15         //コンストラクタ
16         public Watchword(){
17                 this.qNo=-1;
18                 this.question="";
19                 this.answer="";
20                 this.judge="";
21                 this.random=new Random();
22         }
23         //質問作成
24         public void exe　　⑦　　(){        //質問文を呼び出す
25                 this.qNo=this.random.nextInt(　⑧　);//乱数を発生させる数
26                 this.question=Watchword.qArray[this.qNo];
27                 this.answer=Watchword.aArray[this.qNo];
28                 this.judge="";
29         }
30         //クイズの照合
31         public void exeJudge(String 　⑨　){  //答えを照合する
32                 　⑩　(ans.equals(this.answer))this.judge="　　⑪　";  //条件式
33                 　⑫　this.judge="　　⑬　";
34         }
35         //ゲッター
36         public String getQuestion(){return this.question;}
37         public String getJudge(){return this.judge;}
38 }
```

（７）WatchWord.java の解答

```
1  package jp.gihyo.mitoma.chap01_02;
2  import java.util.Random;
3
4  public class Watchword {
5          private final static String[]qArray={"日本一高い山は？"
6                                                ,"日本一長い川は？"
7                                                ,"日本一大きな湖は？"
8                                                ,"日本一広い平野は？"
9                                                ,"日本一高い建造物は？"
10                                               , "日本一売上の高い企業は？"
11                                               };
12         private final static String[]aArray={"富士山"
13                                               ,"信濃川"
14                                               ,"琵琶湖"
15                                               ,"関東平野"
16                                               ,"東京スカイツリー"
17                                               ,"トヨタ自動車"};
18         private int qNo;
19         private String question;
20         private String answer;
21         private String judge;
22         private Random random;
23         //コンストラクタ
24         public Watchword(){
25                 this.qNo=-1;
26                 this.question="";
27                 this.answer="";
28                 this.judge="";
29                 this.random=new Random();
30         }
31         //質問作成
32         public void exeQuestion(){
33                 this.qNo=this.random.nextInt(6);
34                 this.question=Watchword.qArray[this.qNo];
35                 this.answer=Watchword.aArray[this.qNo];
36                 this.judge="";
37         }
38         //クイズの照合
39         public void exeJudge(String ans){
40                 if(ans.equals(this.answer))this.judge="正解";
                   else this.judge="不正解";
           }
           //ゲッター
           public String getQuestion(){return this.question;}
           public String getJudge(){return this.judge;}
   }
```