コンピュータプログラミング
（Android アプリ編）

■もくじ

■第１章 Androidプログラミング編

さて、Java プログラミングの基礎が理解できたところで、今回からはスマートフォンに特化したアプリケーションソフトウェア（※以下アプリ）のプログラミングを始めましょう。

Lesson0　Androidを理解しよう

「Android」とは、Google 社によって作成された、携帯端末（スマートフォンやタブレット）のプラットフォーム（OS）です。オープンソースとして開発されており、利用の際に費用が発生しないため多くの携帯端末メーカーが採用しています。進化過程にある携帯端末の OS はバージョンの遷移が激しく、2009 年 10 月に 1.6 だったのが、2011 年 10 月には 4.0 と大きく進化を遂げています。2015 年 10 月には最新の Android バージョンは 6.0 となっています。

1．　なぜ Android なのか？

総務省情報通信政策研究所が発行している「平成 26 年 情報通信メディアの利用時間と情報行動に関する調査　平成 27 年 5 月」によると、日本のスマートフォン利用者は、平成 26 年には全年代において 62.3%の利用があり、かなりの利用者がスマートフォンを利用していることになる。皆さんの感覚からすると、「え？！そんなもん？もっといないの？」という感覚だろうが、20 代においては、94.4%の利用があり、そのような誤解を与えているのかもしれない。40 代まではフューチャーフォンよりスマートフォンの利用者数が多いが、50 代以上はまだまだフューチャーフォンの利用が多いのが全体の数字を少なく見せている要因です。

かなり浸透しているスマートフォン利用ですが、スマートフォンは小さなパソコンである為、そこには OS を搭載する必要があります。スマートフォンに搭載されている OS には、いくつかの種類があります。まず代表的なものが Apple 社の開発した iPhone に搭載されている iOS、さらに Android 端末に搭載されている Google 社の開発した AndroidOS、この他にも Microsoft 社が開発した Windows Phone などがありますが、Net Applications から 2015 年 12 月のモバイル OS シェアによると、iOS と Android で 92%のシェアを持っているのでこの 2 つに絞って考えて問題ありません。

Net Applications によると 2015 年の 12 月時点での Android のシェアは 57.29%、iOS は 35.43%となっています。こちらは世界シェアですが、市場調査会社 Kantar Worldpanel の、2015 年第 1 四半期におけるスマートフォンの市場シェアで日本のシェアを見ると、Android は 62.9%、iOS は 35.1%と余り大きな差はありません。他国の iOS のシェアが約 20%程度ということに比べると、日本の iOS の比率はかなり高くなっています。それでも、皆さんの周りを見たときにこれまたこの数字との違いを感じるかもしれません。この要因として、メーカー別で見ると Apple 社が 55.8%と 2 位の SONY の 12.7%から大きく差をつけています（BCN ランキング調べ）。Andoroid は多くのメーカーから採用されている為、これといった機種が存在しないのも事実です。そのことが、iPhone 利用者が圧倒的に多いという印象を与えているのではないでしょうか。

皆さんが興味を持ちそうな、iPhone のアプリ開発ですが、iPhone 用アプリの開発には様々な制約事項があり、Objective-C という言語も特殊な仕様になりますので、この演習では世界 No.1 シェアを誇る Android 用アプリを対象にしたプログラミングをマスターすることを目標としましょう。

2. Android の開発環境

Android は開発用アプリも無料で提供されており、作成したアプリを公開する Android Market(現在の Play ストア)もオープンしており、自由に配布したり販売したりすることが可能です。

Android プログラムは Java 言語を使用して作られていますので、Java 環境を構築する必要があります。Java でアプリケーションを開発するには JDK(Java Se Development Kit)をインストールする必要があります。JDK は Oracle 社から無料で入手できます。また、Android SDK (Android Software Development Kit)が必要なため、オープンソースの開発環境「Eclipse」というソフトウェアと Eclipse のプラグイン「Android Developer Tools」を用意する必要があります。

これらのプログラムのインストールと設定を行った上で、Android Virtual Device の作成が必要になります。このエミュレータがないと、作成したプログラムが、どのような動きをするのかという確認を実際の Android 搭載機(実機)で行わないといけなくなり、非常に手間がかります。

3. Android プログラミングで必要な Java の知識と用語

オブジェクト指向

「オブジェクト」とは英語で「もの」という意味です。「オブジェクト」は「インスタンス」と同義に扱われることがあります。インスタンスは実体を指します。オブジェクト指向とは、ソフトウェアを「オブジェクト」としてとらえオブジェクト同士のやり取りをプログラムで表現して組み立てる方法を指します。プログラムの内部構造や操作手順の詳細を知ることなく利用できるというメリットがあります。

パッケージ（package）

パッケージはたくさんある Java のクラスを分類するための仕組みです。パッケージ名はプロジェクト作成時に指定します。他のパッケージのクラスを呼び出すには import 文が必要です。

インポート文（import）

import することで他のパッケージのクラスを利用することができます。

クラス(class)

クラスとは、オブジェクト指向プログラミングの要となる要素で、Java を使って行う Android アプリケーション開発には欠かせない概念です。クラスとは、オブジェクトの情報を一般化した「定義」です。複数のクラスを作成することにより、機能（メソッド）ごとに処理を分けることができます。

記述方法は、

```
修飾子　class　クラス名{
}
```

クラスに使用できる修飾子は、

public	どのクラスからでも使用可能
abstract	インスタンスを生成できなくする
final	サブクラスを作成できなくする

となります。

スーパークラスとサブクラス

クラスは拡張することができます。既に存在するクラスを基にして、それに新しいメソッドやフィールドを追加したり上書きしたりして新しいクラスを宣言することができます。クラスを拡張してできた新しいクラスを「基のクラスのサブクラス」と呼びます。また逆に、基のクラスを「拡張してできたクラスのスーパークラス」と呼びます。

継承（extends）

サブクラスを利用するにはスーパークラスを指定する必要があります。スーパークラスに書かれた内容を継承（extends）することで、追加機能を加えたサブクラスを作成できます。

アクセス修飾子

アクセス修飾子とは、データを他のクラス、または他のパッケージから参照できないように指定するアクセス制限のことです。アクセス修飾子は、「クラス」「メソッド」「メンバ変数」に使用できますが、使用できる修飾子に違いがあります。

public	全てのクラスからアクセス可能
protected	同一クラスと継承したクラスだけアクセス可能
private	同一クラス内だけアクセス可能
なし（省略）	同一クラス内、同一パッケージからアクセス可能

これはクラスに利用できるアクセス修飾子になります。

@Override

スーパークラスにあるメソッドに対して有効なコマンドなので、スーパークラスにない名前を定義するとエラーとなります。また、自分で定義したサブクラスに対して定義してもエラーが出ます。

メソッド（機能）の作成

メソッドとは、クラス内部で定義できる機能のことです。

記述方法は、

修飾子　戻り値　メソッド名（引数）{

}

変数の型

変数は値を入れるための箱です。変数の型はその箱の形や大きさを表します。

Java で使用される基本データ型は次の通りです。

データ型	値
boolean	true or false
char	16 ビット Unicode 文字 ¥u0000～¥uFFFF
byte	8 ビット整数 -128～127
short	16 ビット整数 -32768～32767
int	32 ビット整数 -2147483648～2147483647
long	64 ビット整数 -9223372036854775808～9223372036854775807
float	32 ビット単精度浮動小数点数
double	64 ビット倍精度浮動小数点数

基本データ型には文字列を扱うデータ型がありません。文字列を利用する時は、String を使います。String はデータ型ではなく、標準クラスとして用意された String クラスとなります。

例外処理

コンパイルエラーは文章構造のエラーがあった場合に出ます。文章構造的に問題のないプログラムが実行されたとき、あるはずのないものを見に行ったりするなど実行時に予期せぬエラーが起こります。この実行時に発生するエラーに対処する動きを定めたものが、例外処理になります。

例外処理は例外処理が発生する可能性があるメソッドにおいて、そのメソッドを呼び出したメソッドに処理を任せるのか、呼び出されたメソッドに処理をさせるのかで表記が異なります。例外処理を呼び出したメソッドに反してしまう場合は throws を、呼び出されたメソッド内で例外処理をする場合は try – catch – finaly を指定します。

Lesson1　開発環境を理解しよう

さて、開発環境について。ここまで学習してきた Java には以下の環境がありました。

- JDK
- eclipse

JDK は、Java 言語でプログラミングを行う際に必要な最低限のソフトウェアのセットで無料配布されています。コンパイラやデバッカ、クラスライブラリ、Java プログラム実行環境が含まれます。

Eclipse は、先にも説明したように IBM によって開発された統合開発環境（IDE）の一つで、高機能でありながらオープンソースであるため無償で入手することが可能です。

Android 開発には以下の環境を追加する必要があります。

- Android SDK
- Android Developer Tools

そして、Android Virtual Device の作成が必要になります。これはエミュレータと言って、実機端末がなくてもスマートフォン上での動きを確認できるものです。

Android には様々なバージョンがありますが、最新バージョンを想定して作成した場合、古いバージョンでは動かない可能性があります。

上記の設定が終わったらいよいよ制作の手順に入ります。eclipse を起動してプロジェクトを用意し、実際に Android を動かしてみましょう。（何回か一緒にやってみますので、しっかりマスターしてください）

1．Android アプリケーション・プロジェクトを用意する。

eclipse を使って Android アプリケーションを制作するには、プロジェクトを用意する必要があります。プロジェクトを用意するとアプリを実行するのに必要なファイルを自動的に用意してくれます。作成したプロジェクトの中身は、次のようになっています。

先ず、Java のソースコードファイルが作成される場所となる「src」フォルダは、プロジェクト作成時にアクティビティー名をデフォルトで作成すれば「MainActivity.java」という Activity を用意してくれます。Activity は、複数のクラスを一つのパッケージにまとめる役割があります。プロジェクトを実行するとまず「MainActivity.java」が実行されます。

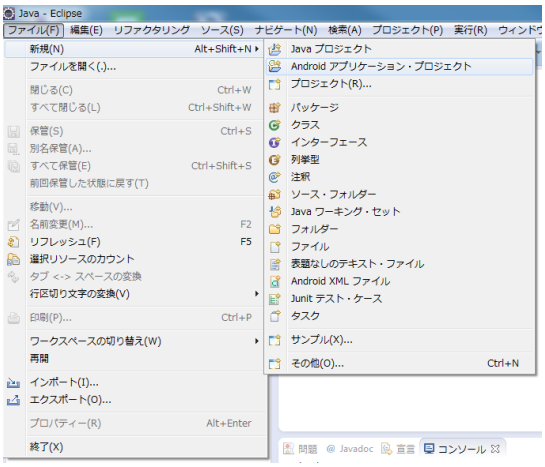
次に、リソースデータの保管場所となる「res」フォルダがあります。こちらには「drawable」「layout」「values」といったフォルダが作成され、画像ファイルやレイアウト用ファイル、コンテンツ用ファイルが保存されています。

更に、自動生成された Java のファイル（R.java 等）が作成される「gen」フォルダ、Android アプリケーション作成に必要なライブラリの設定となる「Google APIs」等、プロジェクト実行に必要なファイルを用意してくれます。

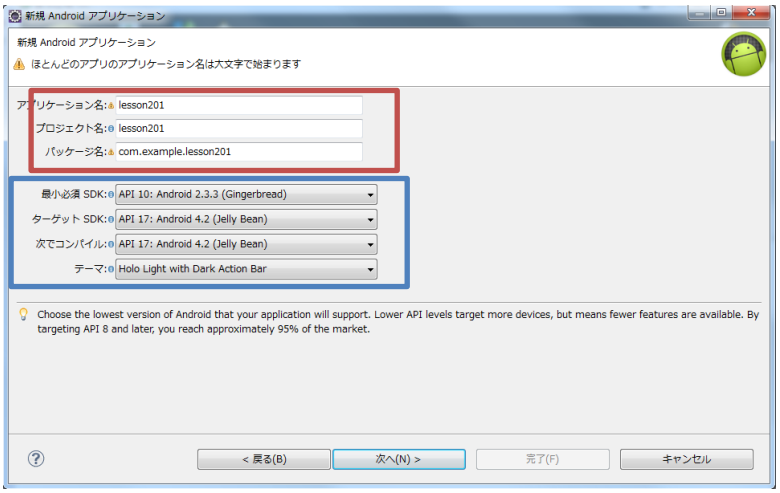
それでは、Ecripse を起動してプロジェクトを用意してみましょう。

① デスクトップ上にあるアイコン「Eclipse」をダブルクリック。

② 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



③ アプリケーション名を入力する

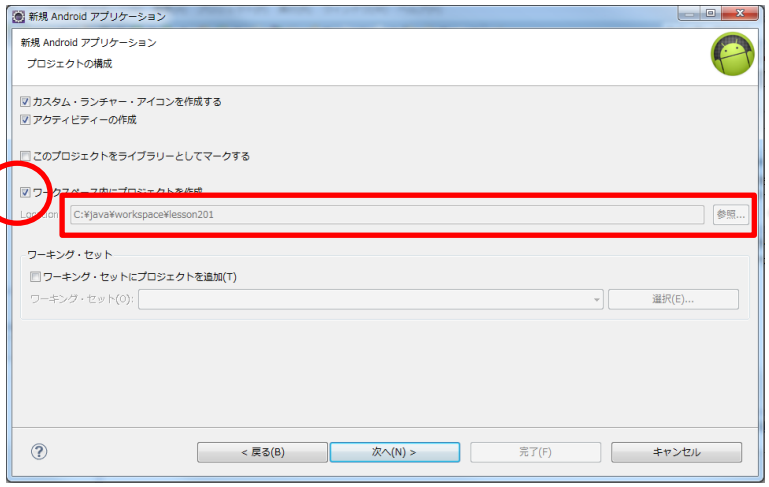


アプリケーション名を入力すると自動的にプロジェクト名とパッケージ名が入力されます。

※青線で囲まれた部分は、最少 SDK、ターゲット SDK・コンパイル SDK・テーマを指定するところですが、基本的には標準のままで問題ありません。動作を想定する Android のバージョンにあわせて変更すると使える機能を制限したり、増やしたりすることができます。

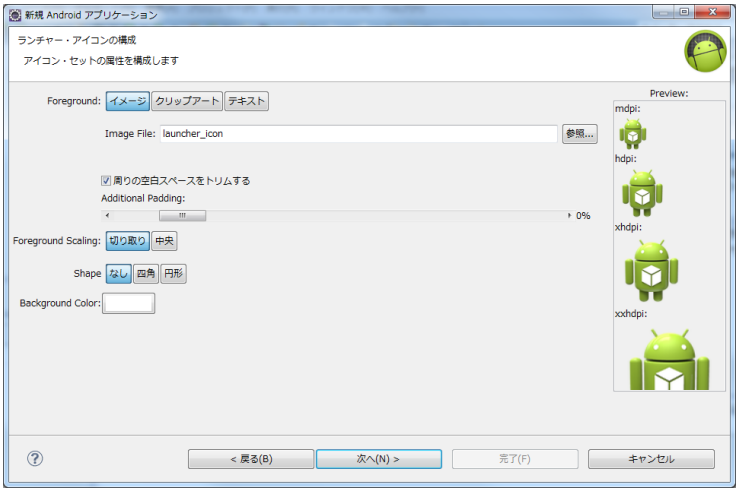
SDK 等を全て 19 にする。

④　プロジェクトの構成を確認し「次へ」をクリックする。この際ワークスペースを変更するのであれば、ワークスペース内にプロジェクトを作成のチェックをはずし、保存先を変更する。

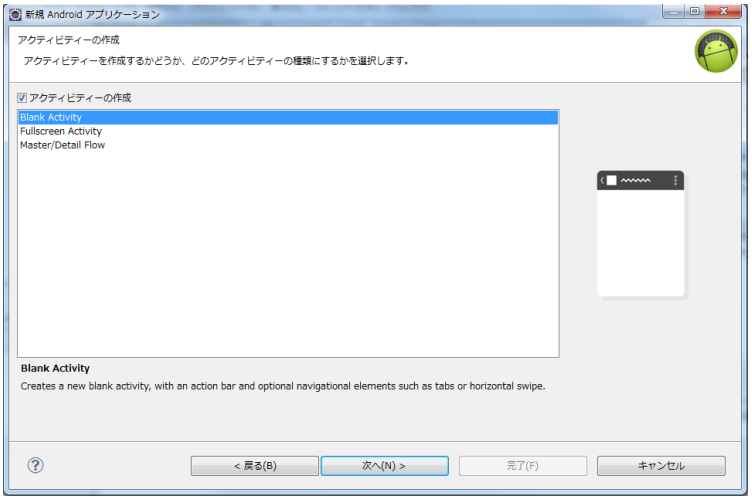


保存先はしっかり考えて保存しましょう

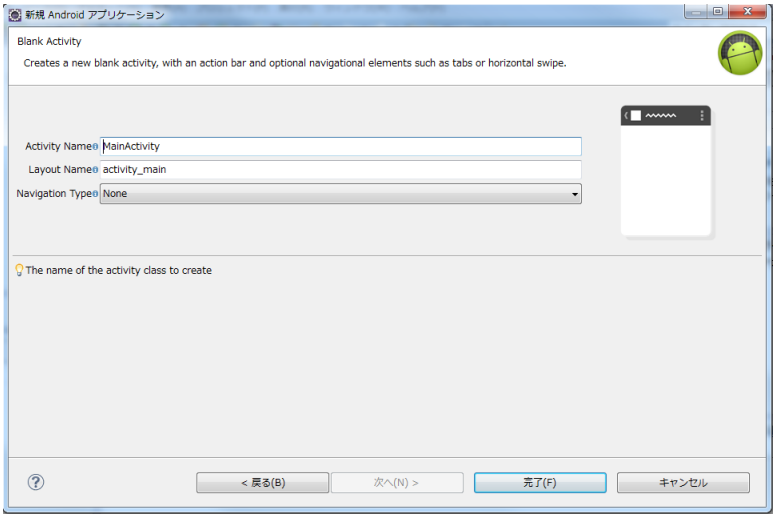
⑤　ランチャーアイコンの構成を確認し「次へ」をクリックする。



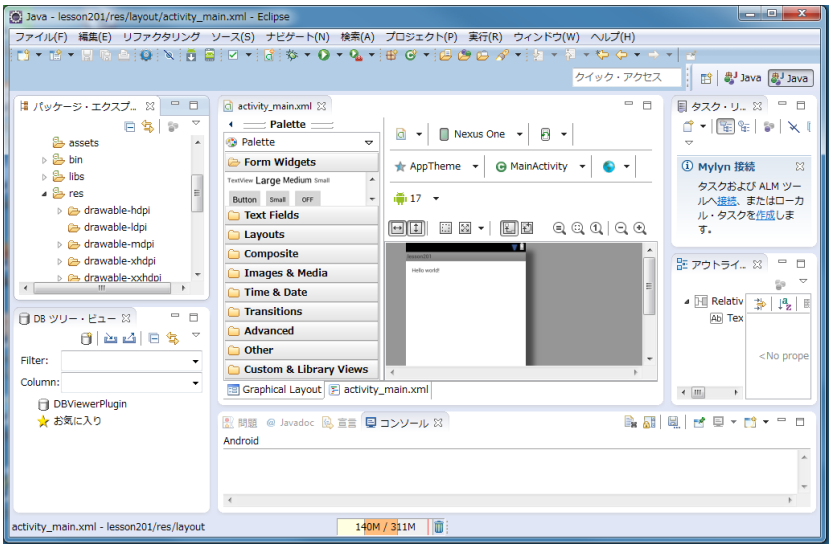
⑥　アクティビティーの作成を確認し「次へ」をクリックする。



⑦　アクティビティー名とレイアウト名・ナビゲーションタイプを確認し「完了」をクリックする。



⑧　プロジェクトが作成される

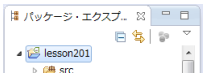


これでプロジェクトが用意でき、編集ができる状態になりました。

2.　Android を動かしてみよう

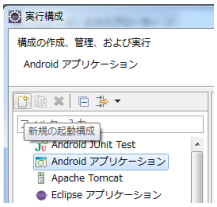
続いてエミュレータを使って実行してみましょう。

①　パッケージ・エクスプローラーよりプロジェクト「lesson201」が選択されていることを確認。

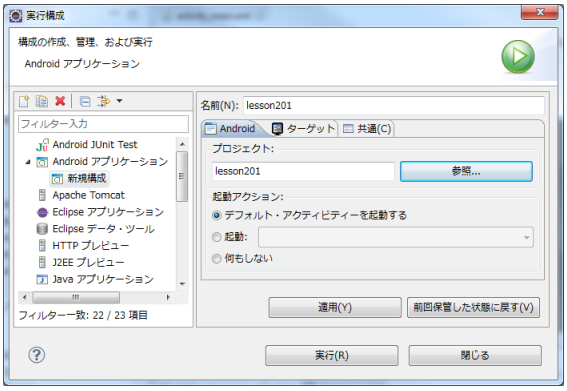


②　メニューより「実行」→「実行構成」

③ Android アプリケーション→新規の起動作成をクリックする。
もしくは Android アプリケーションをダブルクリックする。



④ 名前を決め（任意）、「参照」をクリックし、プロジェクトを選択する（必ずこれから実行するプロジェクトを選ぶ）



⑤ 「ターゲット」タブに切り替え→AVD4．4．2にチェックを入れる。

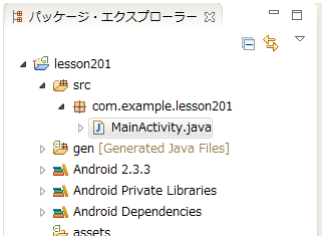
⑥ 実行ボタンをクリック。しばらく待つ。

⑦　実行結果が表示される

エミュレータ上での実行は以上でできます。2回目以降はヒストリーの実行から入力した構成名をクリックすれば実行されます。

3. プロジェクトの中身を確認してみよう

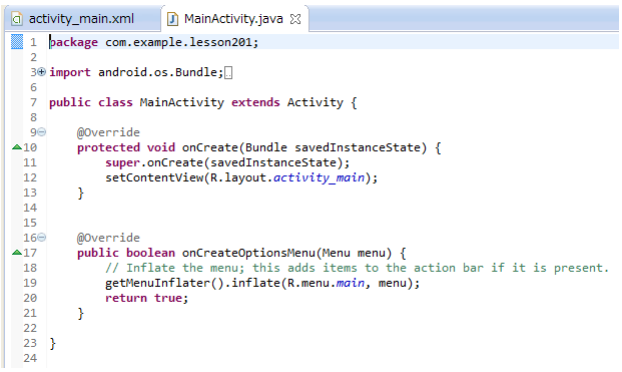
ソースは、MainActivity.java になります。ソースを表示する方法は、パッケージ・エクスプローラーに表示された今から編集したいプロジェクト名（今回は lesson201）の ▷ をクリックして、src の中の com.example.lesson201 の中にある MainActivity.java をダブルクリックします。



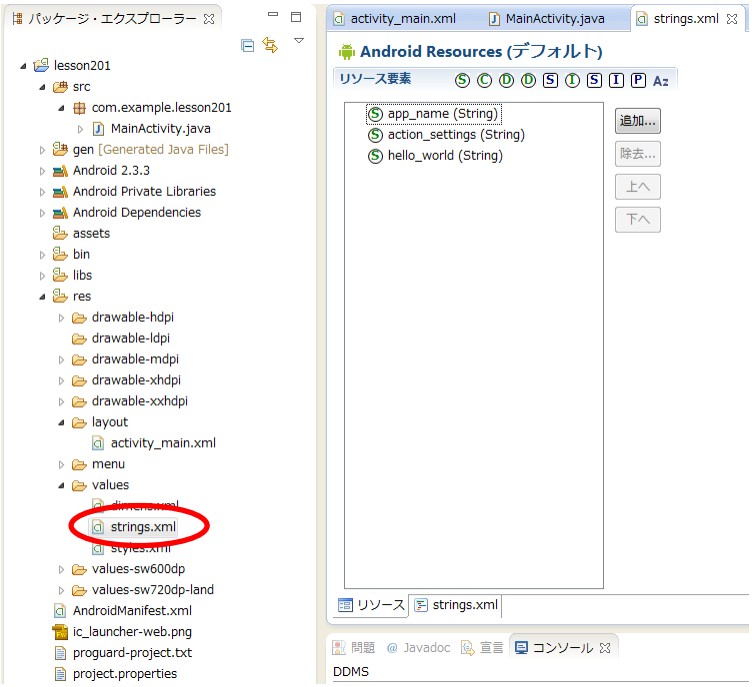
出てくるのは以下のようなソースです。

```
package com.example.lesson201;   //パッケージ名

import android.os.Bundle;      //インポート文
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends Activity {      //クラス

    @Override
    public void onCreate(Bundle savedInstanceState) {  //メソッド
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

実際はこんな感じです。



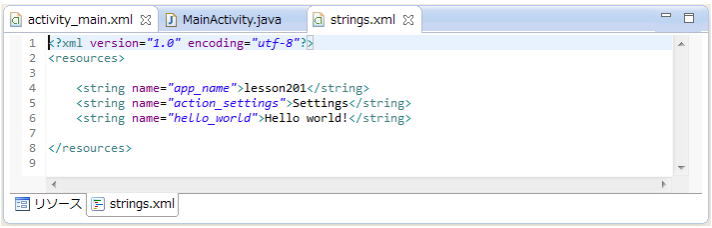
ただ、画面に対する表示はリソース用 XML ファイルで表示していますので、パッケージ・エクスプローラーの該当プロジェクトの res フォルダの中にある values フォルダの中に strings.xml ファイルがあります。res フォルダをダブルクリックして、values フォルダのダブルクリック、更に strings.xml タブをクリックすることでファイルを開けます。



string.xml タブをクリックすれば以下の様なソースが出てきます。

```
<resources>
    <string name="app_name">lesson201</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello World!</string>
</resources>
```

（実際の表示）



例えば「Hello World!」を「こんにちは」に変更するには以下のようにします。

●strings.xml を以下のように変更

<resources>

<string name="app_name">lesson201</string>

<string name="action_settings">Settings</string>

<string name="hello_world">こんにちは</string>

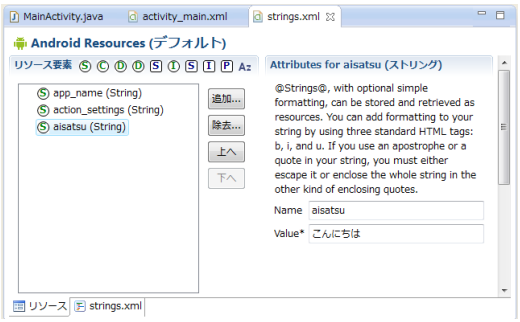
</resources>

アプリのタイトル

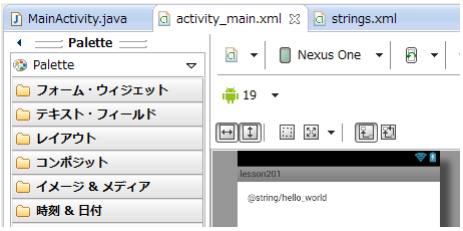
ここを変更する

上記のようにソースから変更できますが、リソース画面から変更することも可能です。

今度は、Name を変更してみましょう。



次に String name を変更したので、activity_main.xml にエラーが出ます。これを修正しますので、activity_main.xml を表示しましょう。

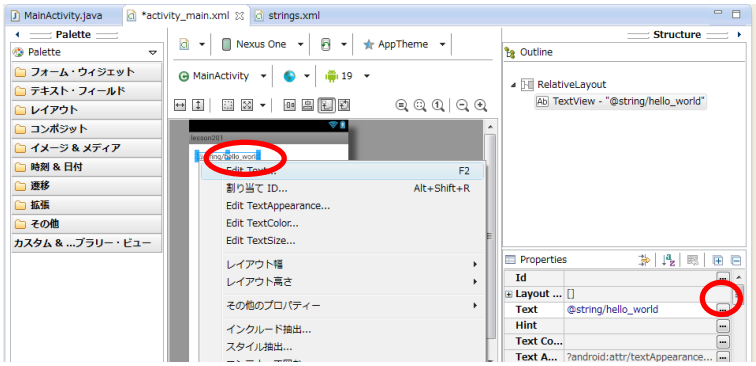


activity_main.xml を表示すると先程まで、「Hello World!」と表示していた部分が、「@string/hello_world」に変更されています。これは、ここに表示されるはずのテキストが、strings.xml に書かれたリソース要素 string の hello_word を表示するように設定されているに

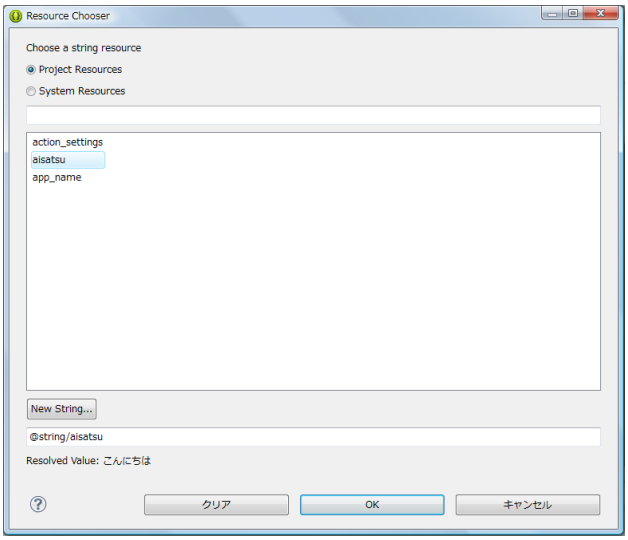
もかかわらず、strings.xml を変更したことによって消失したためです。なので、テキストエリアには再度何を表示するのか指定し直さなければいけません。ここでは、先程変更した aisatsu に変更する必要があります。

① 対象のテキストを選択、もしくは右クリックする。

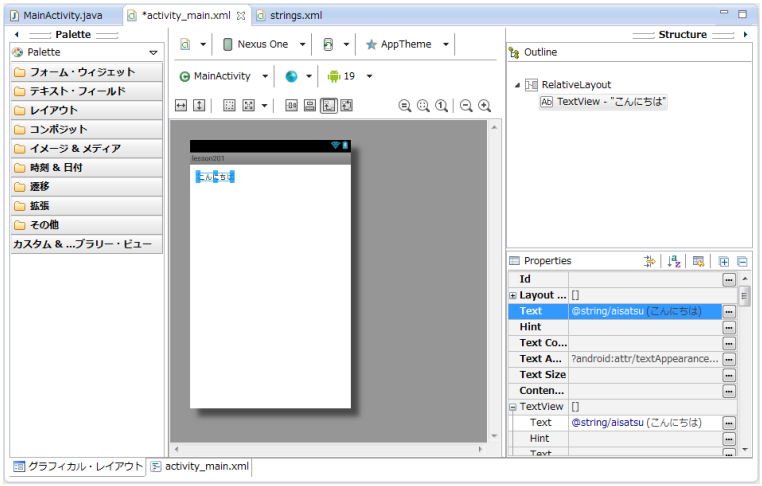
② Structure ビューの Text 項目にある  ボタンをクリックする、もしくは右クリックで表示したメニューから Edit Text…をクリックする。



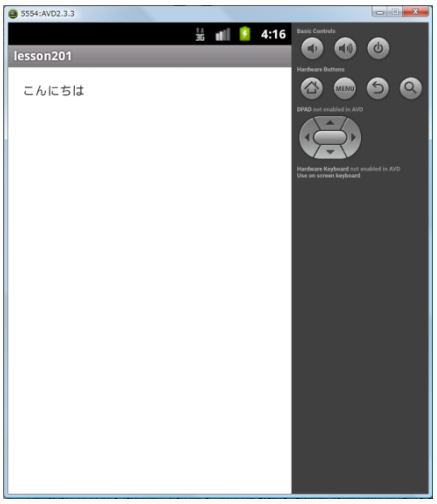
③ Resource Chooser が起動するので表示したいリソース名を選択する。



④　表示が変わったことを確認する。



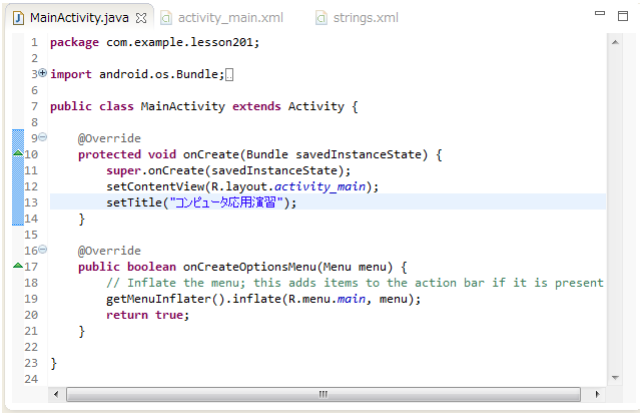
変更が完了したら実行してみましょう。



　表示の変更はソース（JAVA）ファイルを変更する必要はなく、XML ファイルを編集するだけです。それでは、実際にソースファイル（MainActivity.java）を変更してみましょう。

13 行目に「setTitle("コンピュータ応用演習");」を追加してみます。

（実際の表示）



（実行結果）

Activity のタイトル部分が「lesson201」から「コンピュータ応用演習」に変更されたのが確認できたでしょうか？ このように画面に対してタイトルをつけておくと今何をしている状態なのかを確認しやすくなりますので、有効活用しましょう。

アプリのタイトルは「string.xml」の「app_name」からも変更可能ですが、リソースを利用しなくても今回のようにプログラム上から変更することも可能です。Android アプリの制作には、リソースファイルに変数を記述し画面に表示する方法とプログラム上から変数を記述し画面に表示する方法の2通りあるので、この方法の違いをしっかり理解し、場合によって使い分けるようにしましょう。画面もまた同様に Activity を利用する方法と Java から直接生成する方法があります。自分にあった方法で作ると良いでしょう。

【練習問題】
自分の名前を表示してみましょう。なお string name は"my_name"にして下さい。

■strings.xml を以下のように変更

```
<resources>
    <string name="app_name">rensyu201</string>
    <string name="my_name">伊藤雅彦</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

（実行結果）

画面表示はこのように XML ファイルを変更するだけで良いのです。Java プログラムはそのまま使えるので、特に面倒なことはないと思います。

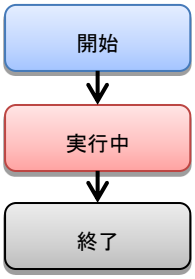
ワンポイント！
長い文章の時の改行は¥n つまり Java に準ずる

Lesson2　知っておきたい Android の基本用語

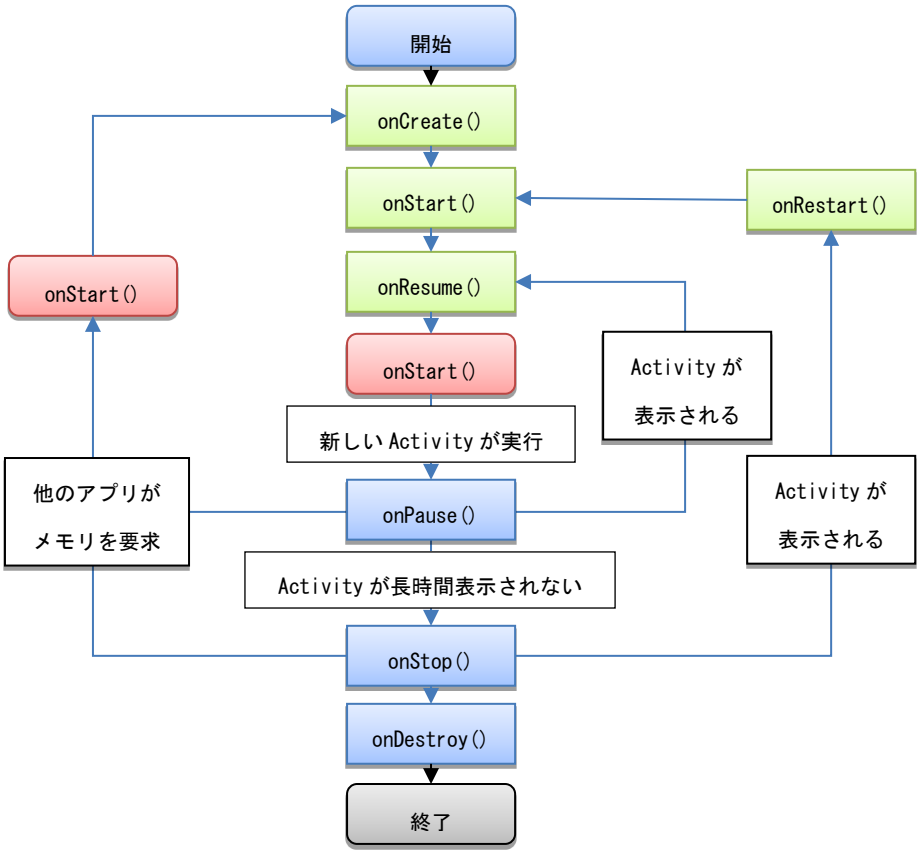
Activity

Activity とは、Android アプリの画面に相当します。まずはこの画面を用意することが、Android プログラミングの基本です。

アプリケーションには必ず「始まり」と「終わり」があります。この流れを「フローチャート（状態遷移）」といい、図にすることをフローチャート図といいます。**「開始」→「実行」→「終了」**をフローチャート図にすると以下のようになります。



もちろん Activity にも状態遷移が存在します。実行中の部分を詳細に表示した Activity のフローチャート図は以下のようになります。



先程のフローチャート図に比べるとかなり複雑に見えますが、基本は**「開始」→「実行」→「終**

了」です。実行する際の様々な流れがフローチャート図には示されています。この流れがわかるようになれば、Android プログラムの基本を理解していると言えます。現時点で皆さんが理解しておかなければいけない点は、実行されると「onCreate()」が呼び出され、そこから指示されたメソッドへと移行していくということです。

MainActivity.java

それでは、「MainActivity.java」の中身を確認してみましょう。

activity_main.xml | MainActivity.java | strings.xml

```
1  package com.example.lesson301;
2
3  import android.os.Bundle;
6
7  public class MainActivity extends Activity {
8
9      @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.activity_main);
13     }
14
15     @Override
16     public boolean onCreateOptionsMenu(Menu menu) {
17         // Inflate the menu; this adds items to the action bar if it is present.
18         getMenuInflater().inflate(R.menu.main, menu);
19         return true;
20     }
21
22 }
23
```

１行目は、このプログラムのパッケージ名を表示しています。今回の場合だとクラス「MainActivity」はパッケージ「com.example.lesson301」に属しているということになります。

３～５行目は、非表示になっていますが、import 文が書かれています。

７行目は、クラスの宣言をしています。宣言文を日本語に直すと「Activity というスーパークラスを基（extends）にクラス MainActivity という public 型（アクセス修飾子）を宣言している。」ということになります。

９行目の「@Override」は「スーパークラス（Activity）に用意されているメソッド(onCreate)をサブクラス(MainActivity)で書き換えて利用しますよ」という宣言です。

１０行目は、サブクラスを定義しています。これは必ず用意しなければいけないサブクラスです。「protected void onCreate(Bundle savedInstanceState)」は、「同一クラスと継承したクラスだけアクセス可能な戻り値を持たない onCreate（引数は Bundle 型の savedInstanceState（アプリの前回終了時の状態））の定義」という意味です。ここで定義されているクラス名「onCreate」は、JAVA の「main」に相当します。Activity が実行されることで最初に実行されるクラスであるということがわかります。

１１行目の「super.onCreate(savedInstanceState);」は、インスタンスの状態を保存すべきタイミングで呼び出されるという意味で、「決まり文句」と思っておいて問題ありません。同様に「onRestorInstanceState()」インスタンスの状態を復元するタイミングで呼び出されるというのもあります。

１２行目の「setContentView()」メソッドは Activity クラスに定義されているメソッドで、引数で

指定した画面レイアウトファイルと、メソッドが実行されたインスタンス（ここでは MainActivity オブジェクト）の紐づけを行っています。つまり、「MainActivity.java」が実行されたら「onCreate()」を読み込み、「activity_main.xml」を呼び出すということになります。

16行目の「onCreateOptionsMenu」はメニューボタンが押されたときに実行する内容となります。

後は、画面上に配置されたボタンが押された時など、イベント発生時に応じてサブクラスを追加して記述することでプログラムに様々な動きを持たせることができるようになります。

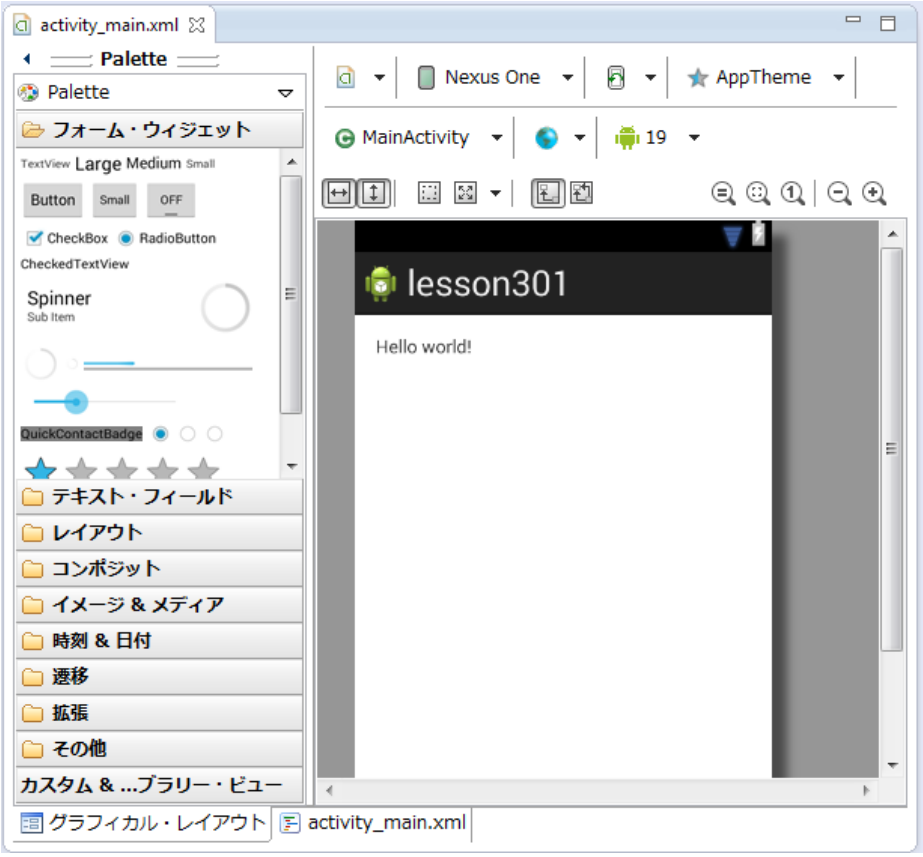
R.java

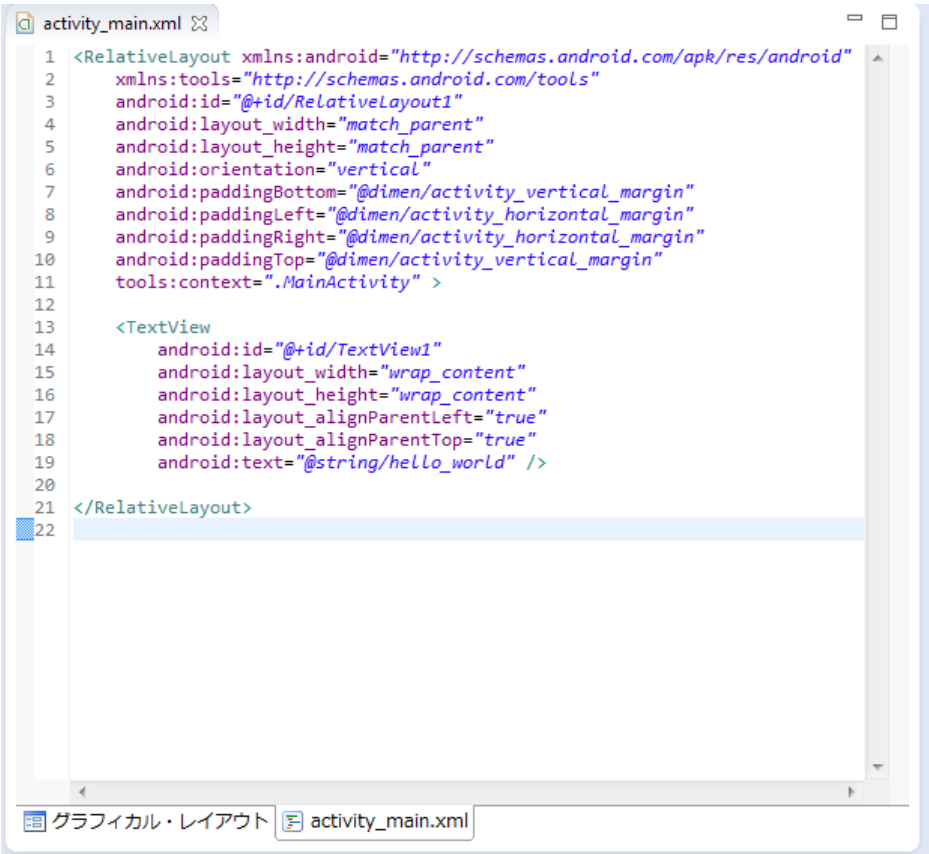
R,java は activity_main.xml や strings.xml に定義した要素を識別する ID を定義しています。そこから関連性を引出し、画面上に表示します。XML で定義された変数をプロジェクト内で関連を持たせてくれています。画像などを追加した場合もその画像がどこにあるのかといった情報を ID 化し管理しているクラスになります。Android では各リソースにリソース ID が割り当てられ、それを R クラスで管理しています。プログラムでリソースにアクセスするときは R クラスから取得することになります。

R.java はプロジェクト内のデータを更新すると自動的に記述されるため、自分で編集することはありません。また、そこに書かれた ID などを勝手に変更してしまうと正しく動作しなくなってしまうので、注意しましょう。

activity_main.xml

activity_main.xml はプログラムで使用するリソースデータで、レイアウト用 XML です。プログラムを実行したときに表示する画面のレイアウトデータとして用意しています。通常は「グラフィカル・レイアウト」ビューで表示しており、上部には画面のタイトル名が表示され、その下に様々な View（パーツ）を配置することが可能です。View の配置は「Palette」から画面上に表示させたい View をドラッグ操作で配置します。タブをソースビューに切り替えることで activity_main.xml を直接編集することも可能です。


activity_main.xml
Palette
Palette
フォーム・ウィジェット
TextView Large Medium Small
Button Small OFF
CheckBox RadioButton
CheckedTextView
Spinner
Sub Item
QuickContactBadge
テキスト・フィールド
レイアウト
コンポジット
イメージ & メディア
時刻 & 日付
遷移
拡張
その他
カスタム & ...ブラリー・ビュー
Nexus One
AppTheme
MainActivity
19
lesson301
Hello world!
グラフィカル・レイアウト
activity_main.xml


activity_main.xml
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2 xmlns:tools="http://schemas.android.com/tools"
3 android:id="@+id/RelativeLayout1"
4 android:layout_width="match_parent"
5 android:layout_height="match_parent"
6 android:orientation="vertical"
7 android:paddingBottom="@dimen/activity_vertical_margin"
8 android:paddingLeft="@dimen/activity_horizontal_margin"
9 android:paddingRight="@dimen/activity_horizontal_margin"
10 android:paddingTop="@dimen/activity_vertical_margin"
11 tools:context=".MainActivity" >
12
13 <TextView
14 android:id="@+id/TextView1"
15 android:layout_width="wrap_content"
16 android:layout_height="wrap_content"
17 android:layout_alignParentLeft="true"
18 android:layout_alignParentTop="true"
19 android:text="@string/hello_world" />
20
21 </RelativeLayout>
22
グラフィカル・レイアウト
activity_main.xml

ソースは XML というマークアップ言語を使い記述します。ソースビューを見ると 1 行目に「RelateveLayout」とあります。これは画面に配置されるパーツのレイアウト方法を相対的に指定しています。レイアウトパターンは大きく分けて 5 種類存在します。それぞれのレイアウトパターンを組み合わせて使用することも可能です。デフォルトでは RelateveLayout になっていますので、慣れるまでは LinearLayout に変更して利用するのがお勧めです。

RelativeLayout	複数の View を相対的に配置する
LinearLayout	View を縦または横方向に並べる
GridLayout	View を格子状に並べる
FreameLayout	View を左上を原点に重ねる
AbsoluteLayout	View を絶対座標で配置する

レイアウトパターンの変更方法は画面上を右クリックで「レイアウトの変更」を選択。Change Layout のダイアログボックスから目的のレイアウトパターンを選択します。

それぞれのレイアウトパターンには属性があります。

View は縦方向（layout_height）と横方向（layout_width）の属性を持ち、それぞれの値には、画面最大（match_parent）と必要最低限（wrap_content）があります。さらに、View には比率（layout_weight）があり、値には数字を入れます。ここで入力された数字の合計を画面全体で割り、入力された数字の割合で表示されます。

13行目を見てみると「TextView」が用意されており、縦も横も最小サイズ（TextView の中に入る文字の大きさに合わせる）となっており、表示させる文字は「android:text="@string/hello_world"」に書かれているコンテンツ用リソースデータ（strings.xml）に記述された「hello_world」を表示するという意味になります。

strings.xml

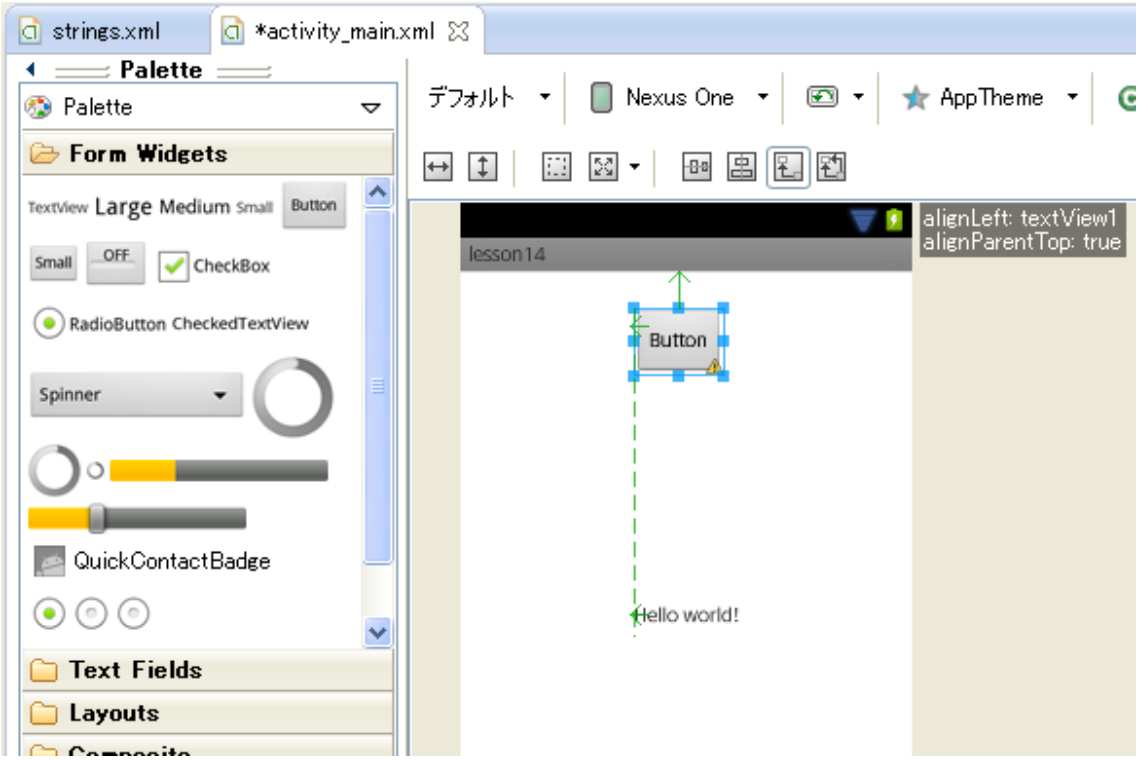
strings.xml は、プログラムで使用するリソースデータで、コンテンツ用 XML になります。リソースは、直接レイアウト用 XML（activity_main.xml 等）に記述することも可能ですが、レイアウト用 XML には変数を指定し、変数の中身を strings.xml のような値専用のデータに作成します。表示する文字データを別に用意することで、プログラムと切り離して利用することが可能になります。

Android アプリケーション・プロジェクトでは、複数のファイルと連携を取って作成していくことで、より複雑な動きやプログラムを分けて考えることができ、簡単に作成することが可能になります。

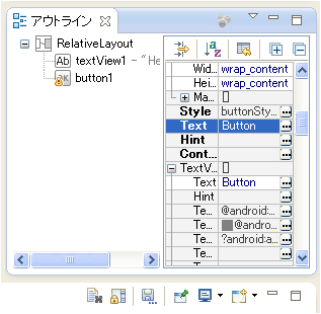
さあ、やっと入り口に差し掛かったところという具合です、色々なプログラムを組みながらマスターして行きましょう！

Lesson3　テキストビューとボタン、画像の配置

今回はテキストビューとボタン、画像の配置を学びましょう。
activity_main.xml を開き、グラフィカルレイアウトタブをクリック。テキストビューは、はじめから表示されている「Hello world!」のことです。ここで、パレットのフォームウィジットから、Button をドラッグ&ドロップしてみて下さい。ボタンが出来上がります。テキストビューを追加したい場合は、同様の操作を行えば追加できます。

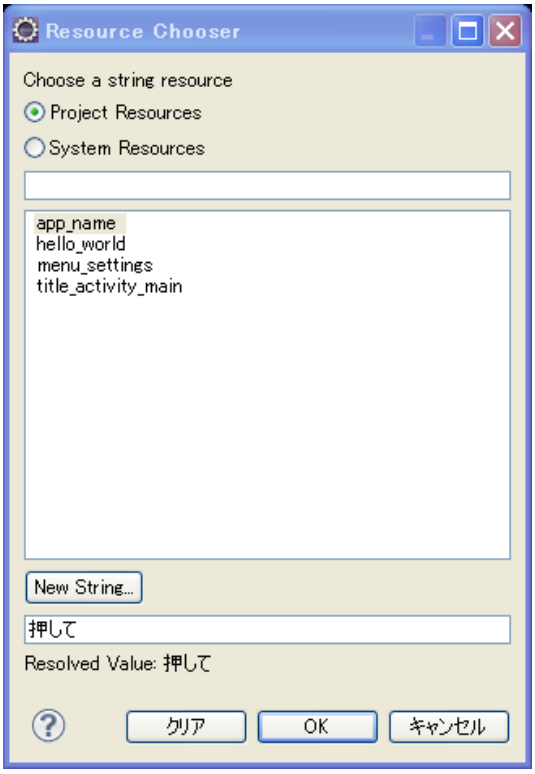


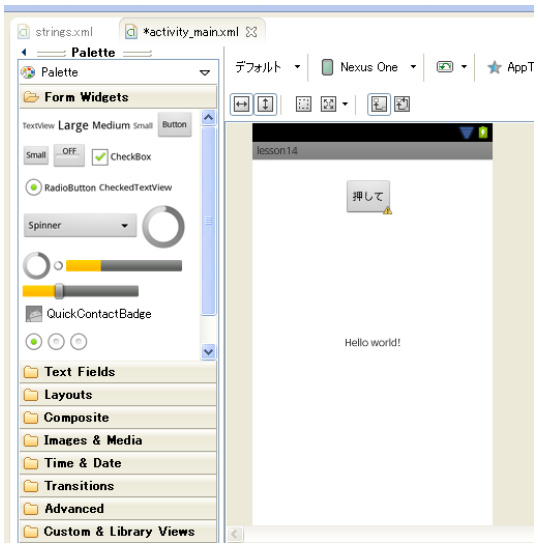
ボタンの初期表示は button ですから、これを変更しましょう。Text と書いている行の右のボタンをクリックしましょう。



もしくは挿入したボタンを直接右クリックし、出てきたメニューから「Edit Text…」を選択しても同じ画面が出ます。

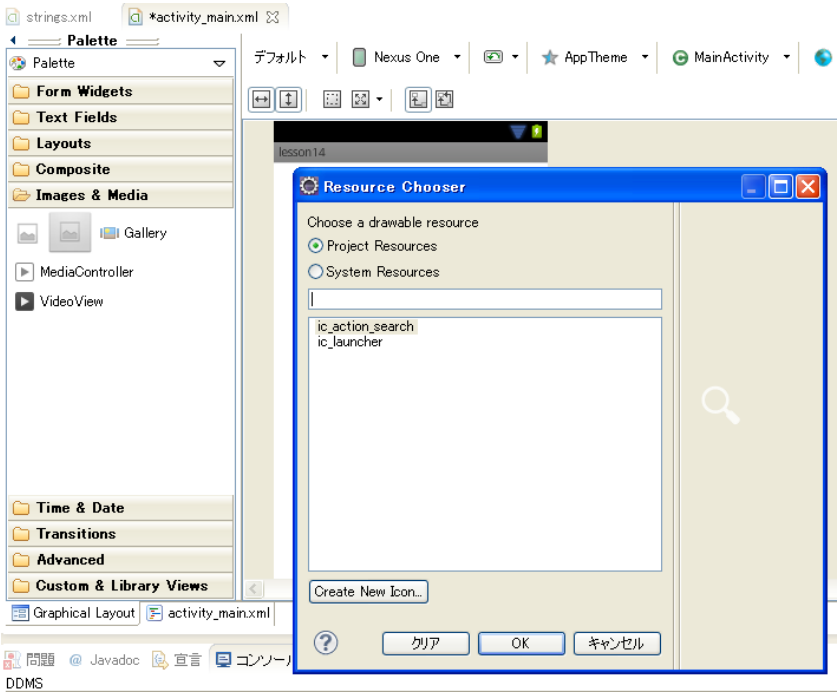
button のプロパティの中の Resolved Value:の文字を「押して」に変えて OK を押しましょう。



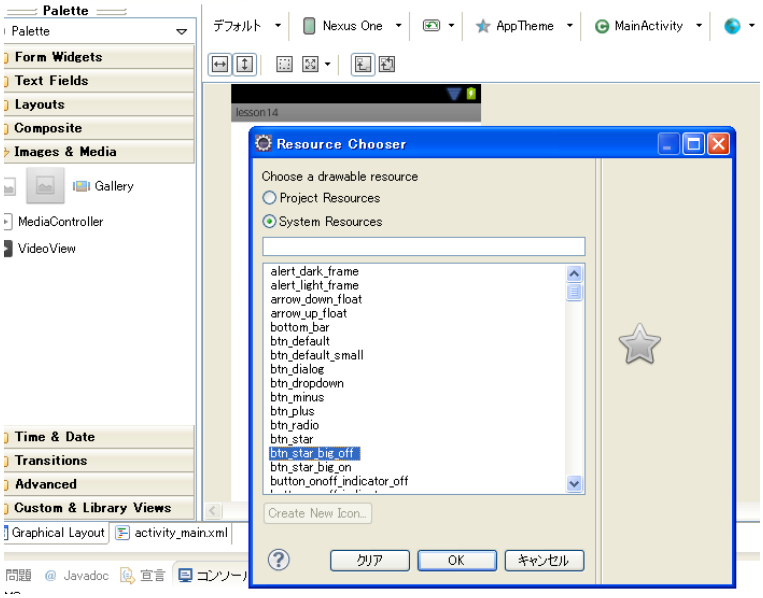


上記のようにボタンの表示が変わります。

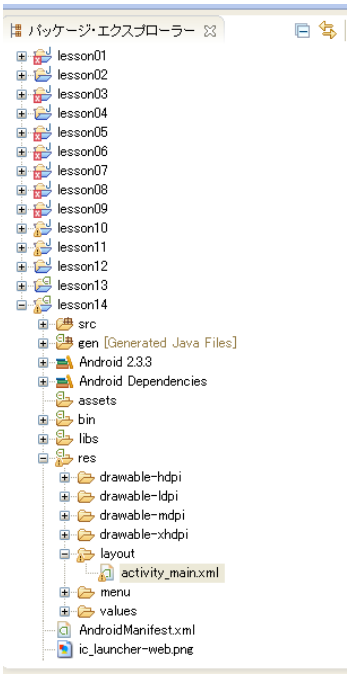
画像の場合には、パレットの Images & Media から左上にある ImageView をドラッグ&ドロップします。Resource Chooser ダイアログが表示されます。



画像は2種類あり、ひとつが Project Resources というプロジェクトごとに用意した独自の画像。もうひとつが System Resources というシステムに最初から同梱されているデフォルト画像です。まず、System Resources を使います。その中の好きな画像を選択して OK を押してください。

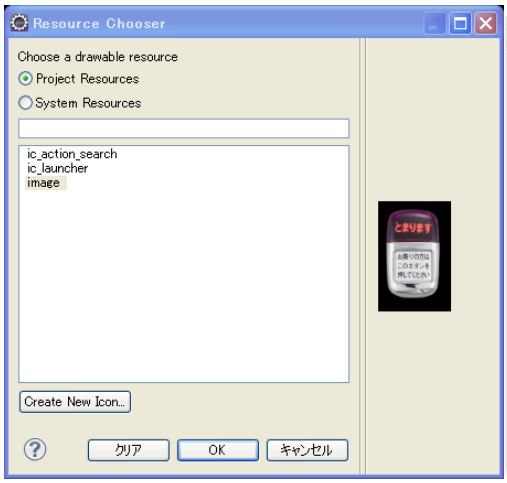


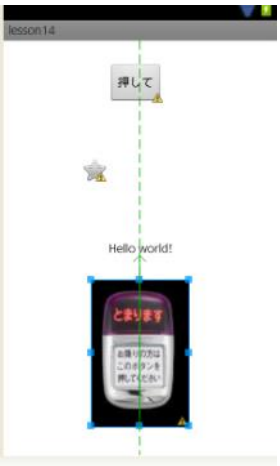
次に自分で用意した画像を表示してみましょう。まずパッケージ・エクスプローラーを見て下さい。res の中に drawable-xxx というフォルダがいくつかあります。この drawable-ldpi フォルダにドラッグ &ドロップで画像を入れます。**ファイル名に–や頭数字、大文字は×！**



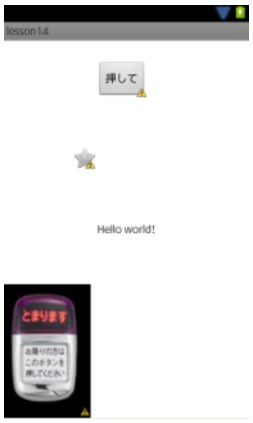
使用する画像ファイル名は必ず半角英数字で一文字目に数字は使えません。また、ロングファイルネームも避けるようにしてください。画像ファイルのサイズが大きすぎても表示ができない可能性があるので十分注意しましょう。

パレットのイメージ＆メディアから左上にある ImageView をドラック&ドロップします。Resource Chooser ダイアログが表示されます。Project Resources を選択して入れた画像の名前（拡張子は表示されません）を選択します。実行してみましょう。

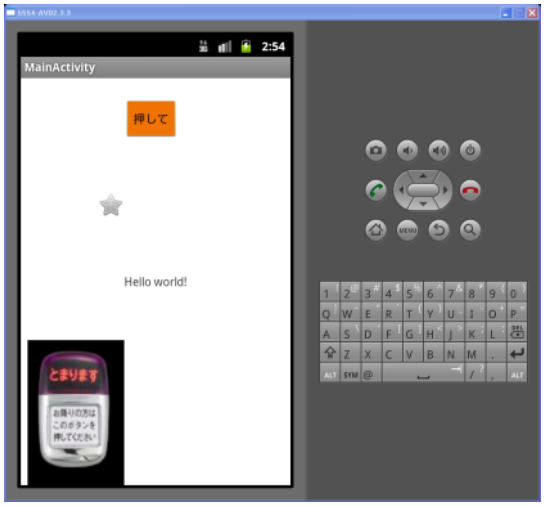




画像の位置はマウスで自由に変更することができます。



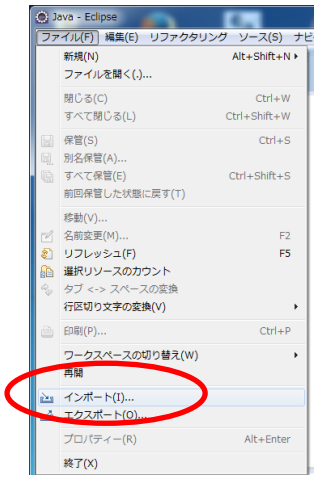
（実行結果）

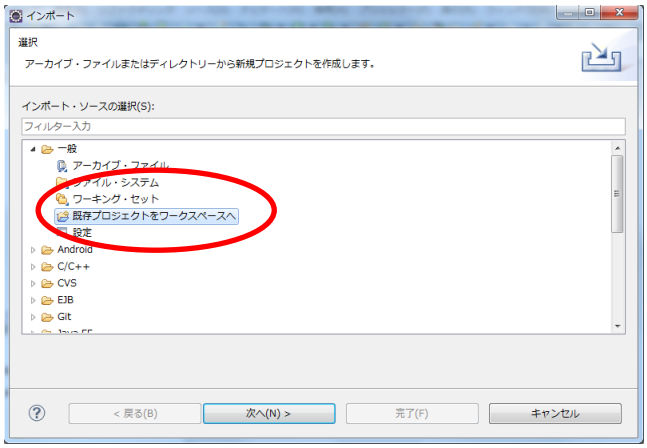


1.　プロジェクトのインポート

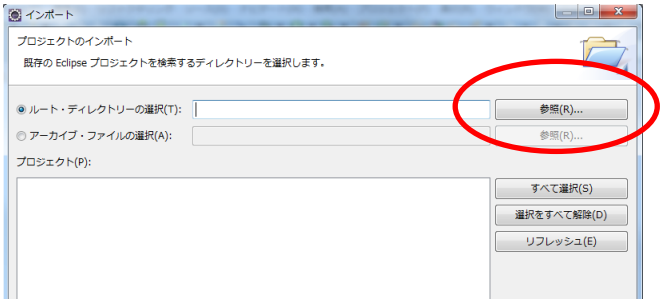
　演習ではコンピュータのシステム上、毎回プロジェクトを用意する必要がありますので、毎回作成するのですが、その際、前回のプロジェクトを継続して作成したいという場合もあります。前回の続きを作る場合、わざわざソースを打ち直す必要はありません。プロジェクトをインポートすればいいのです。

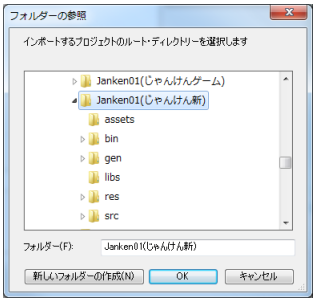
　「ファイル」より「インポート」をクリックし、「一般」から「既存プロジェクトをワークスペースへ」を選択し、「次へ」をクリックする。



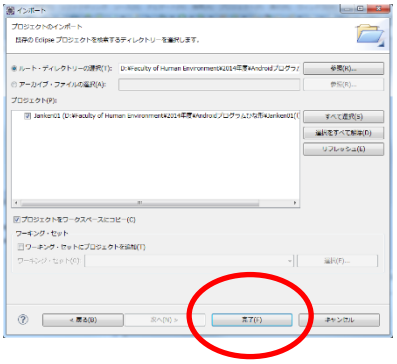


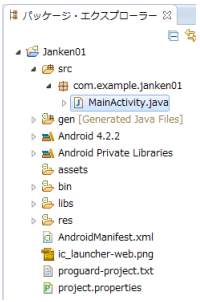
インポートしたいプロジェクトを選択するため「参照」をクリックし、インポートしたいプロジェクトが入ったフォルダを選択する。





「完了」をクリックするとパッケージ・エクスプローラーにプロジェクトが読み込まれたのがわかる。





2. レイアウト

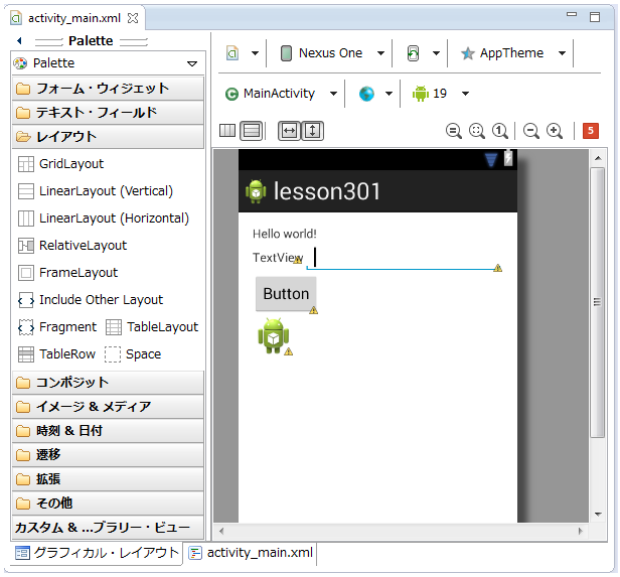
スマートフォンはレイアウトが大事です。見た目と感覚で操作ができるスマートフォンは、画面の作りでプログラムの善し悪しが変わってきます。もちろん基本となるソース部分が一番大事ですが、画面をないがしろにしてはいけません。レイアウトについてしっかり理解をしておきましょう。

activity_main.xml

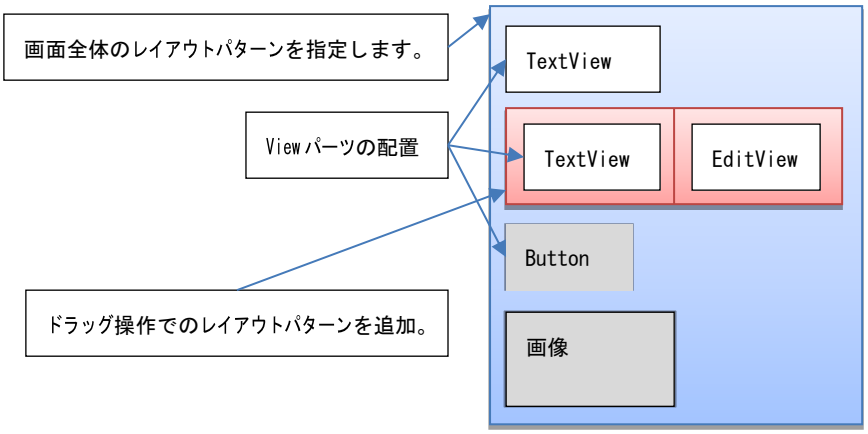
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/TextView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="TextView" />

        <EditText
            android:id="@+id/editText1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10" >

            <requestFocus />
        </EditText>

    </LinearLayout>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />

</LinearLayout>
```

グラフィカル・レイアウト　activity_main.xml

（上図をグラフィカル・レイアウトで見ると）



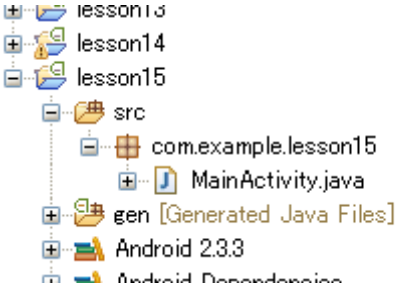
レイアウトパターンを「LinearLayout（vertical）」に変更し、TextView の下に新たに「LinearLayout（horizon）」を追加し、その中にさらに TextView と EditText を配置し、その下に Button を配置し、更に画像を配置した例です。マークアップ言語は始めと終わりがあります。全体のレイアウトが1～53行目に設定されており、その中に13～17行目に TextView の配置した後、19～39行目に「LinearLayout（horizon）」のレイアウト方法で TextView と EditView を配置しています。その後に Button と ImageView を配置しているという流れになります。



それぞれの TextView と Button に表示する値は、android:text で指定された値になります。ImageView は、android:src で指定された値になります。表示する画像は「res」フォルダの「drawable」に用意されている必要があります。

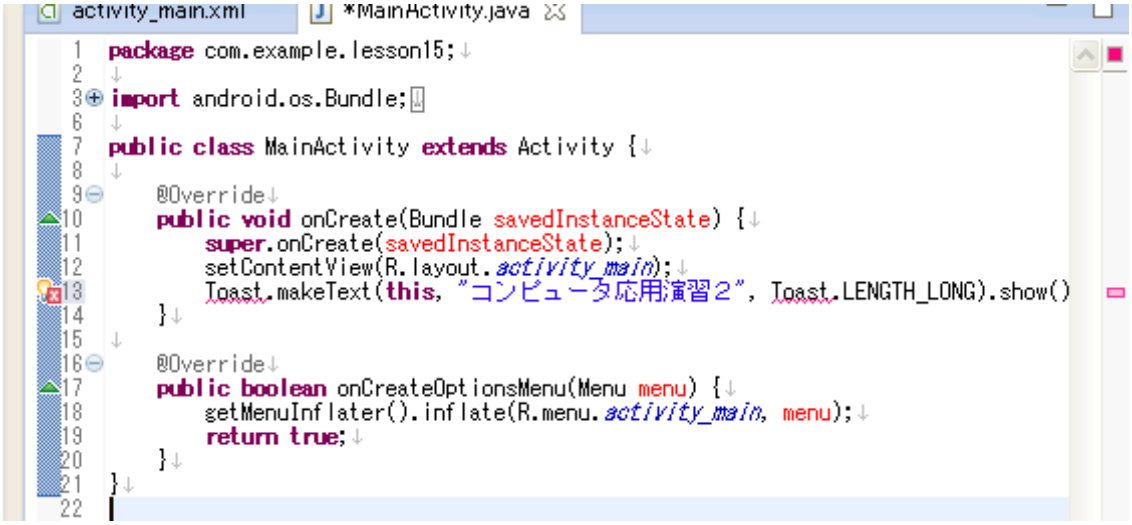
Lesson4　メッセージダイアログの表示

メッセージダイアログを表示しましょう。使うのは Toast クラスです。
Toast を使ってログを表示するには
Toast.makeText(this, "任意のメッセージ", Toast.LENGTH_LONG).show();
と記述します。"LENGTH_LONG" は表示する時間を表しています(LONG は 4 秒)。
短い時間表示したい時は "LENGTH_SHORT" とします(SHORT は 2 秒)。
これを MainActivity.java を W クリックして onCreate() に追加すると。



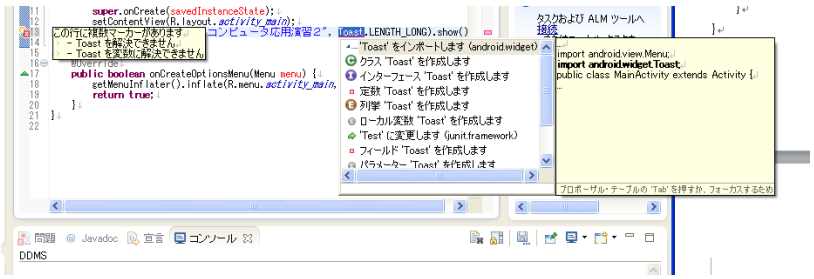

```
package com.example.lesson203;

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

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toast.makeText(this, "コンピュータ応用演習2", Toast.LENGTH_LONG).show();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

となります、

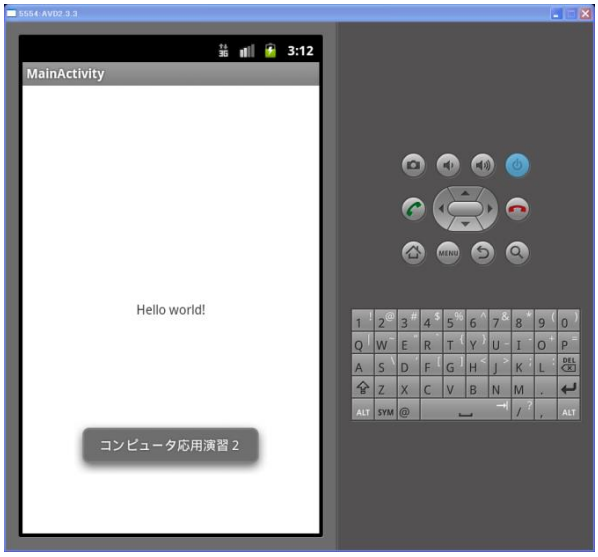


13 行目の前に X 印が出ています。これはエラーです、この X 印をクリックすると下の様なメッセージが出ます。ここで、「Toast をインポートします」をダブルクリックすれば、問題を自動的に解決してくれます。

ワンポイント！
import 文は以下の操作で自動的に補完されます。
Ctrl+Shift を押しつつ O(オー)のキーを押す（覚えておこう！）



これを実行すれば以下の様になります。ダイアログは一瞬で消えます。



また、このメッセージダイアログの表示位置を変更することも可能です。

記述は少々複雑になりますので、十分に注意して入力しましょう。

まずは前述で書いた、

Toast.makeText(this, "コンピュータ応用演習２", Toast.LENGTH_LONG).show();

の部分を

Toast toast = Toast.makeText(**getApplicationContext()**, "コンピュータ応用演習２", Toast.LENGTH_LONG);

に書き換えます。さらに下記の 2 行を追加します。

toast.setGravity(**Gravity.AXIS_CLIP, 0, 0**);

toast.show();

これで完了です。

表示位置の指定は、setGravity で書かれた太字部分を変更することでできます。第一引数で表示位置の指定、第二引数で X 方向の位置、第三引数で Y 方向の位置を指定します。

第一引数には下記の値が使用できます。

値名	意味
AXIS_CLIP	中央表示
BOTTOM	下部表示
CENTER	中央表示
FILL	フル
FILL_HORIZONTAL	上下中央左右フル
FILL_VERTICAL	上下フル左右中央
HORIZONTAL_GRAVITY_MASK	
LEFT	左端に表示
RIGHT	右端に表示

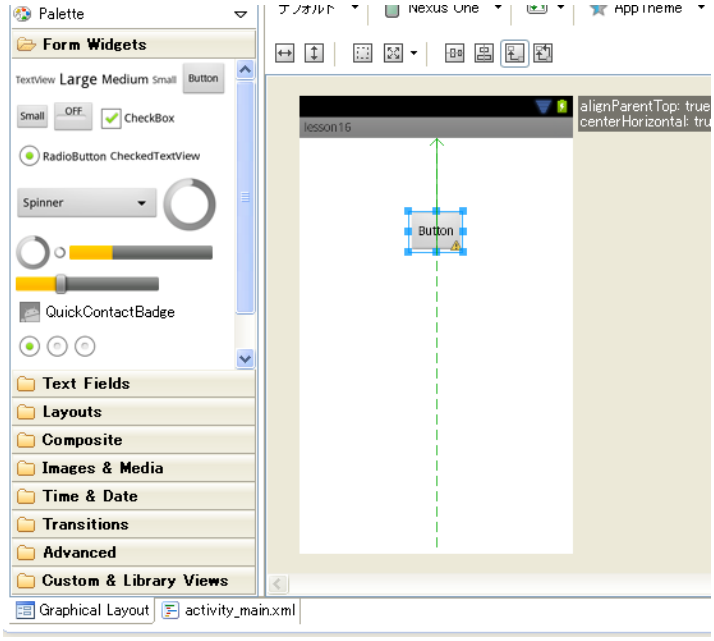
TOP	上部に表示
VERTICAL_GRAVITY_MASK	

ここに示した値は一例です。この他にもいろいろあるので、目的に合った値を入力して表示を変更してみましょう。

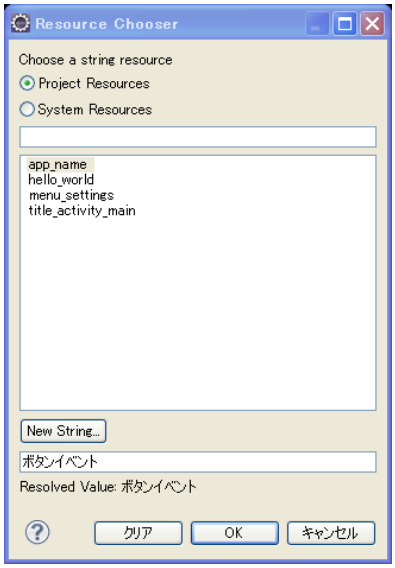
Lesson5　ボタンのイベントの取得方法

今回はボタンのイベントの取得方法について。まず新規で lesson204 というパッケージを作成しましょう。

次に Graphical Layout のタブをクリック(最初から出てる場合が多い)する。画面には Hello World と文字が表示されているのでそれをクリックして DELETE キーを押し削除しましょう。Form Widgets から Button を選び Hello World のあったあたりに置いてください。

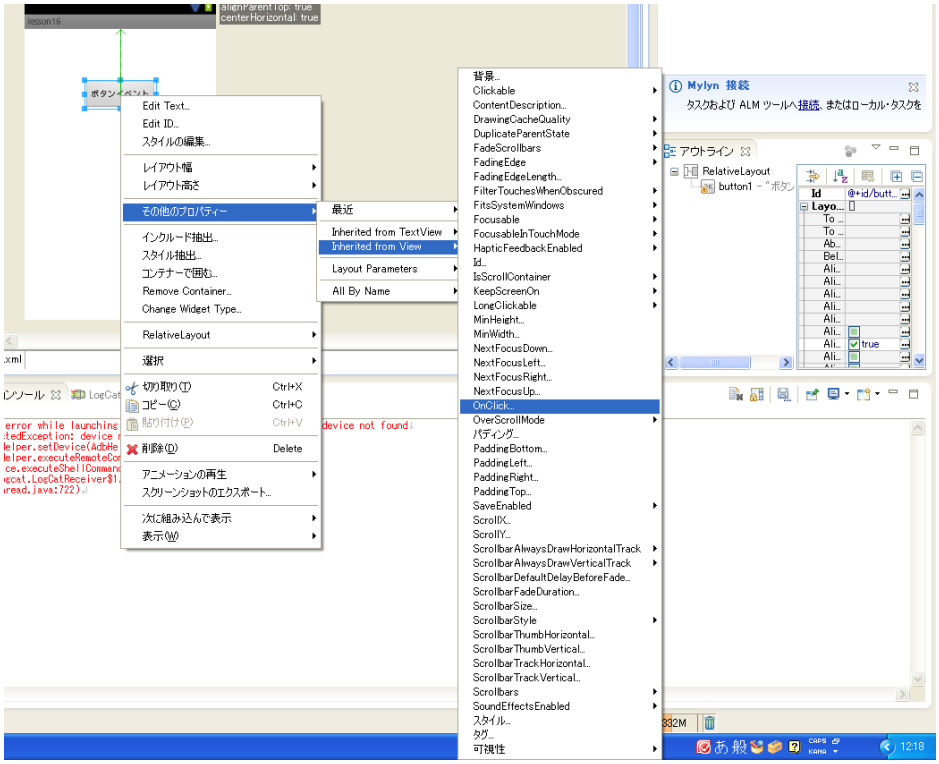


そのボタンを右クリックし、Edit Text を選択しましょう。Resolved Value: Button と書いてある上の窓にボタンイベントと入れましょう。ボタンの文字が変わりました。

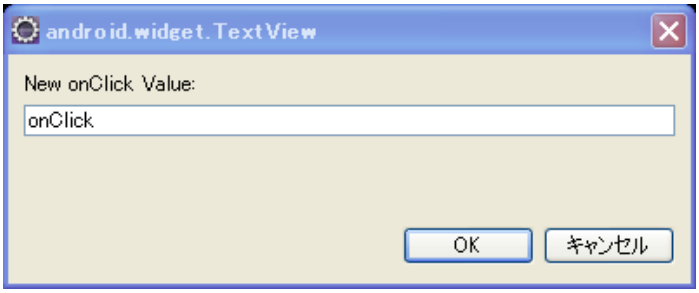


自作する時は好きな文字を入れていいのです。そのボタンを再度右クリックしてその他のプロパテ

ィから Inherited from View の OnClick を選択してください。



ダイアログボックスが出てくるので、そこに onClick と入れましょう。そして OK をクリック。



次に MainActivity.java（パッケージ・エクスプローラーの src の中の com.example.xxxx の中にあります）をダブルクリックしましょう。それが Java のメインプログラムソースです。これを以下の様に書き換えてください。

```
package com.example.lesson204;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.Toast;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setTitle("ボタンイベントの取得");
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
    public void onClick(View view) {
        Toast.makeText(this, "ボタンを押しましたね。", Toast.LENGTH_SHORT).show();
    }
}
```

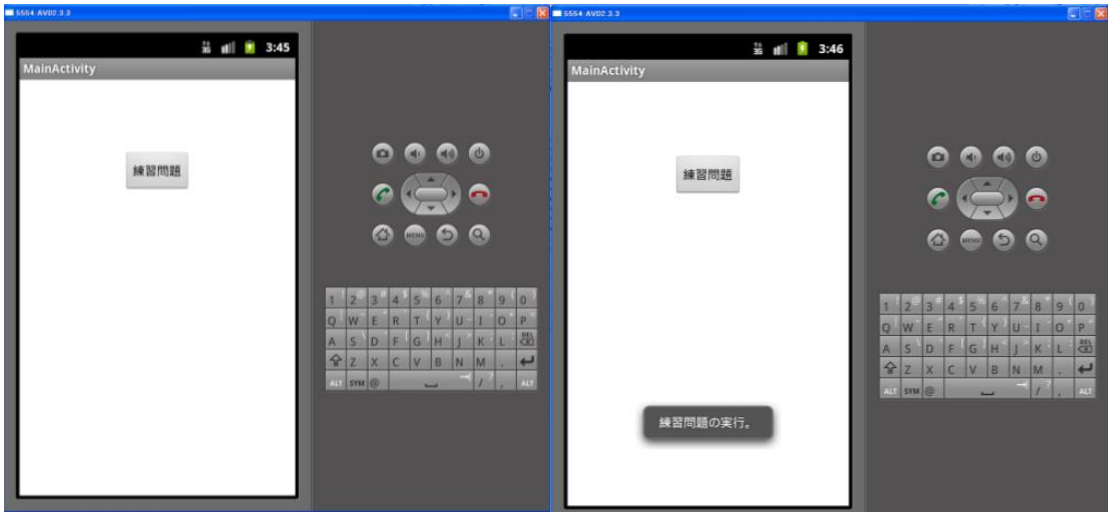
＊最後に　ctrl　＋　Shift　＋　o　（オー）を押して import を補足します。

では実行してみましょう。

（実行手順のおさらい）メニューの実行から実行構成、Android アプリケーションを選択し、プロジェクトは click,ターゲットタブをクリックしたら AVD 名は AVD2.3.3 を選択して実行をクリック。MENU ボタンをクリックしてロックを解除。ボタンイベントボタンをクリックしてみましょう。

ボタンをクリックした時に画面下にメッセージが出ましたか？

練習としてボタンの表面の文字とメッセージを変更して実行してみましょう。



とりあえずとしてはこれで完成です。ですがこのままではボタンを押すたびに同じ動きしかしませんので、少し面白くありません。前期の内容を踏まえて、ボタンを複数回押したときに表示する内容を変更してみましょう。ヒントは条件分岐です。

その他、ボタンが複数ある場合にそれぞれが違う動きをするといったようなことも考えて見ましょう。こちらのヒントは「onClick」イベント。

Lesson6　文字列を表示する

プログラムを実行したときに画面上に文字を表示するには様々な方法があります。これまで使ってきた方法では Activity(activity_main.xml)にテキスト情報を表示させるには、Activity に TextView を配置し、リソース(string.xml)で定義した ID を指定する方法と、TextView に直接文字列を指定する方法をやりました。

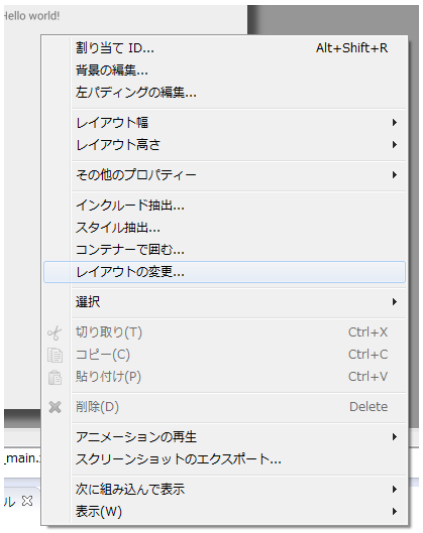
今回は、リソースを使わずにプログラムで表示する値を指定する方法を 2 種類やります。

Android プログラミングを行う上で、プログラムと画面、リソースの関係を理解しておくことは非常に大事です。

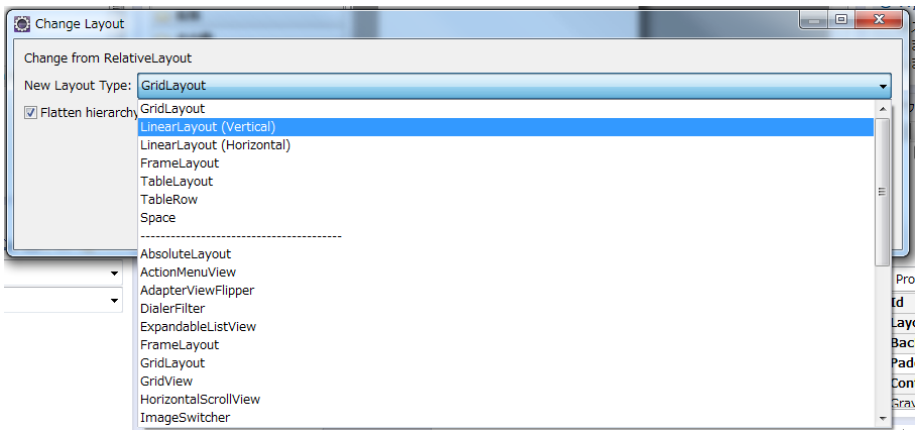
1.　TextView を XML による指定と JAVA による表示

まずは、プログラム(MainActivity.java)上から Activity に用意された既存の TextView へ文字を表示する方法を学びましょう。

はじめにレイアウトの変更を行います。この作業は特に必要というわけではありませんが、レイアウトを学ぶ上で、わかりやすい表示に切り替える為の作業です。レイアウトの種類については「Lesson15 知っておきたい Android の基本用語」に書かれた「activity_main.xml」を参照しましょう。Activity は標準で RelativeLayout になっていますが、これを LinearLayout に変更します。変更方法は下記のように行います。

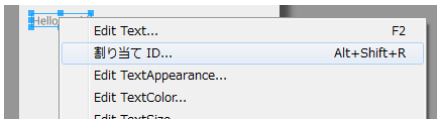


グラフィカル・レイアウトに切り替えて、画面上を右クリックする。出てきたメニューから「レイアウトの変更」を選択する。

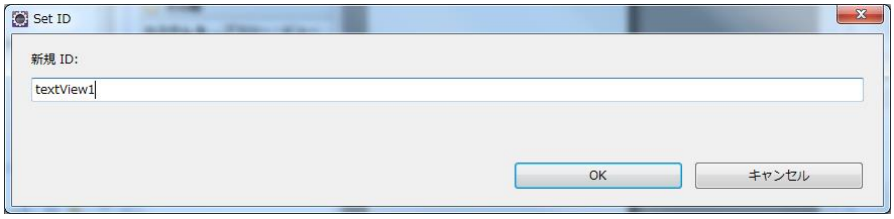


「New Layout Type:」から「LinearLayout(Vertical)」を選択し、「OK」をクリックする。

それでは、本題です。TextView を定義（もしくは変更）しましょう。



もともと配置されていた TextView（Hello world の文字が表示しています）を右クリックし、出てきたメニューから「割り当て ID」を選択する。
※TextView が無い場合は、自分で配置する。



「textView1」を入力し「OK」をクリックする。※「textView1」は変数名にあたるので半角英数であればなんでもよい。大事なのは、この際、「Set ID」に入力した値はしかり覚えておくこと。

activity_main.xml のソースを確認し、下記のソースと同じか確認しておく。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
```

```
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</LinearLayout>
```

ここで必ず上記のファイルを保管しておきましょう。
次に Java のソースを以下のように作成します。

```
package com.example.Lesson206;

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

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //テキストビューの設定
        final TextView kotoba = (TextView)findViewById(R.id.textView1);
        kotoba.setText("文字が変わりましたか？");
    }
    @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;
    }
}
```

変数名

Set ID で入力した値

どうですか？ うまくいきましたか？ テキストの表示には様々な方法があります。自分にあった表示方法を見つけてしっかりと使い分けるようにしましょう。

※次の操作の為、このプロジェクトは残して、新しいプロジェクトを用意しましょう。

2. TextView と LayoutView を Java で生成

先程は、XML で定義した TextView を利用して文字を表示させましたが、次は、XML を使用せずに JAVA プログラムだけで、Activity（レイアウトビュー）と TextView を生成し、文字を表示させる方法です。

先程と違い、「activity_main.xml」の中は変更しません。変更するのは「MainActivitiy.java」のみになります。

MainActivity.java の内容を以下の様にします。

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //テキストビューを生成
        TextView text = new TextView(this);
        //テキストビューに文字を配置
        text.setText("プログラムの中で¥n テキストデータを¥n 発生させています。");
        //レイアウトビューの生成
        LinearLayout layout = new LinearLayout(this);
        Layout.setOrientation(LinearLayout.VERTIUAL);
        //レイアウトビューにテキストビューを追加
        layout.addView(text);
        //レイアウトビューを表示
        setContentView(layout);
    }
}
```

プログラムだけで見ると非常にシンプルですね。コメントに書かれた手順をしっかりと理解しておきましょう。

このように XML を利用することなく JAVA だけでプログラムを行っていくことは可能ですが、プログラム自体は非常に複雑になっていく可能性があります。プログラムでは、直接数字を入力す

るのではなく、変数を使うことが一般的です。同様に XML のようなリソースもうまく使うことで、より複雑なプログラムも簡単に作成できるようになります。

いずれにせよ、時と場合によって切り替えて作る必要がありますので、どちらの作り方も覚えておく必要はあります。

3. TextView を XML から変更する

さて、ここまで行ったのは画面上にテキストを表示するという非常に単純なものです。見た目もシンプルですが、このままではあまり面白くはないでしょう。前期のコンソール画面とは違い折角エミュレータを使用して表示させているのですから色をつけて見ましょう。

テキストに色を指定したり、TextView に背景色を指定したりすることができます。また、文字の大きさも変更することが可能です。

文字の色を変更したい場合は、

android:textColor=”#ff0000”

背景色を変更したい場合は、

android:background=”#ffffff”

と入力すれば変更できます。色の指定には16進数で RGB の指定をすれば可能です。

文字の大きさを変更したい場合は、

android:textSize=”20dp”

と入力すれば変更できます。単位は、dp,sp 等があります。または、

android:textAppearance="?android:attr/textAppearanceSmall"

という指定方法で、Small,Midium,Large が選択できます。

複数の TextView を用意し、それぞれに背景色と文字色をつけて表示してみましょう。
こちらの変更は XML が必要な為、先に作ったプロジェクトを利用しましょう。

4. TextView を JAVA から変更する

XML を利用したときは XML に定義された TextView に指定を追加することで、文字の背景色と文字色を変更することができました。もちろんこれを JAVA から指定することも可能です。

背景色の変更には、

```
text.setBackgroundColor(0xffffff00);
```

文字の色の変更には

```
text.setTextColor(0xff00ff00);
```

を使用します。

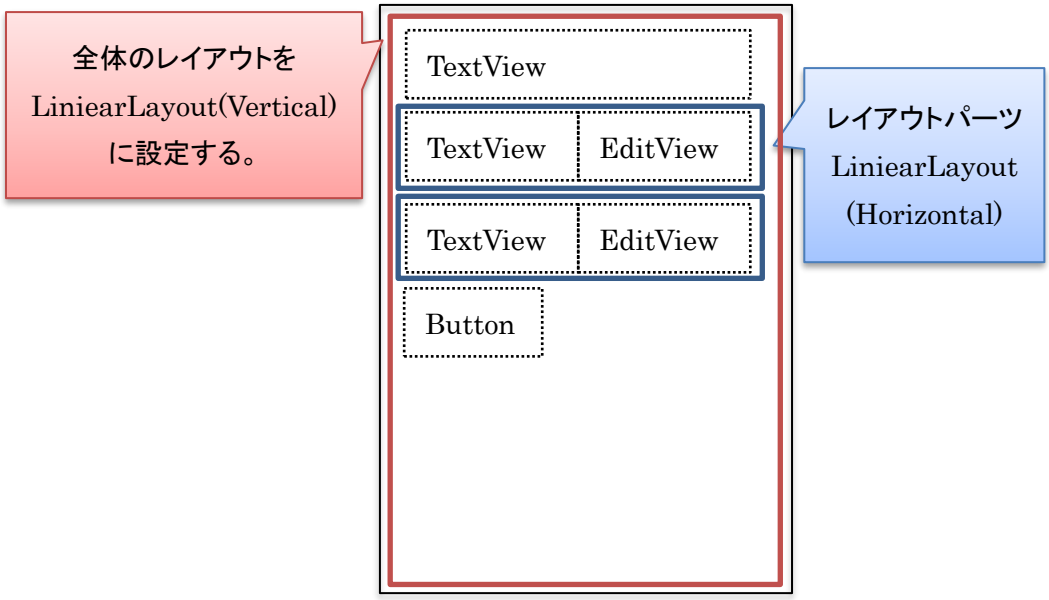
色の指定は XML の時と同様に 16 進数の RGB 形式で指定します。

こちらの変更には先程との違いを確認する為にも後で作ったプロジェクトを利用しましょう。

Lesson7　入力した文字列を連結して表示する

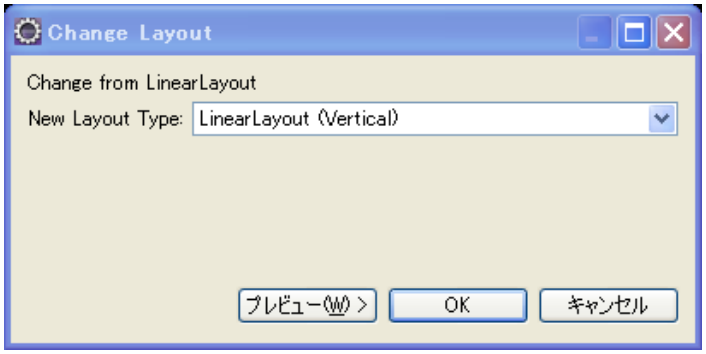
まず、新規で lesson207 というパッケージを作成しましょう。今回作成するプログラムは、文字列の連結だけで無く、レイアウトの項目も含まれています。これまでもレイアウトの問題は出てきていましたが、ここでしっかり確認しておきましょう。

今回作成するレイアウトは、以下の図のようになります。これを前提に、各パーツをはめ込み、XML を完成させましょう。

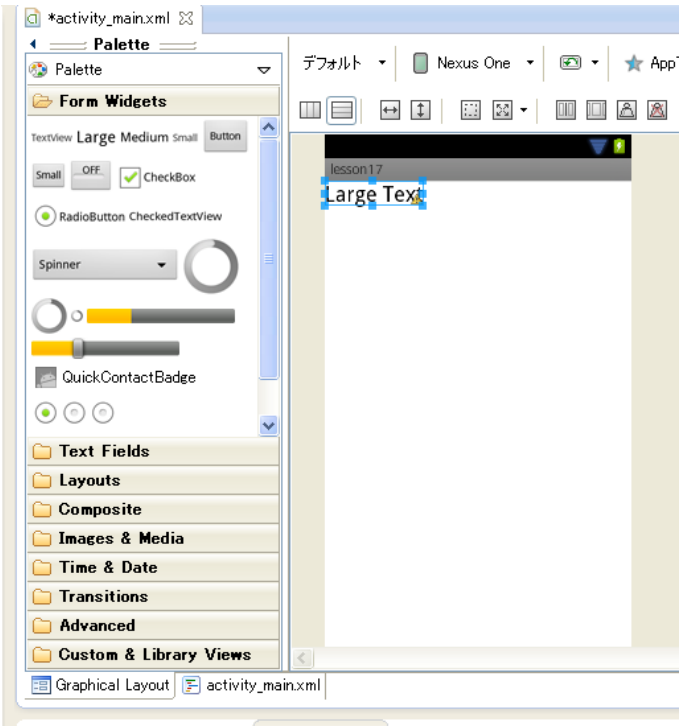


このイメージを基に作成していきましょう。activity_main.xml を開き、Graphical Layout のタブをクリック（最初から出てる場合が多い）する。画面には Hello World と文字が表示されているのでそれをクリックして DELETE キーを押し削除しましょう。

次に**画面上で右クリックし、ショートカットメニューからレイアウトの変更**を選択します。LiniearLayout(Vertical)を選択します。

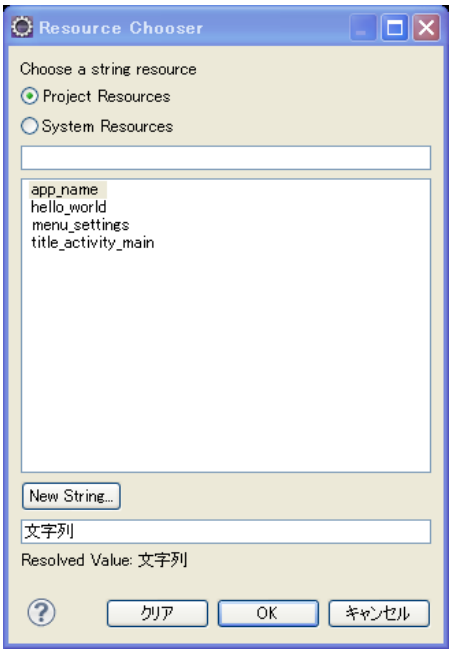


次に、Form Widgets から Large を選んでドラッグ &ドロップする。

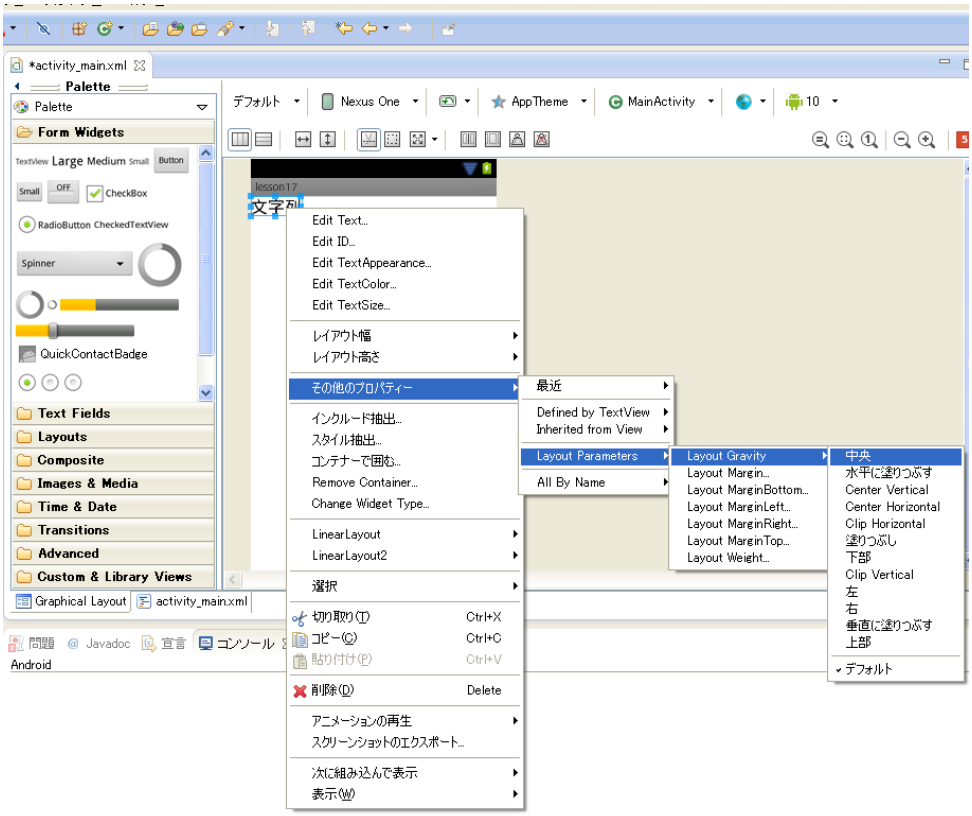


Large Text の上を右クリックし、Edit Text を選択する。

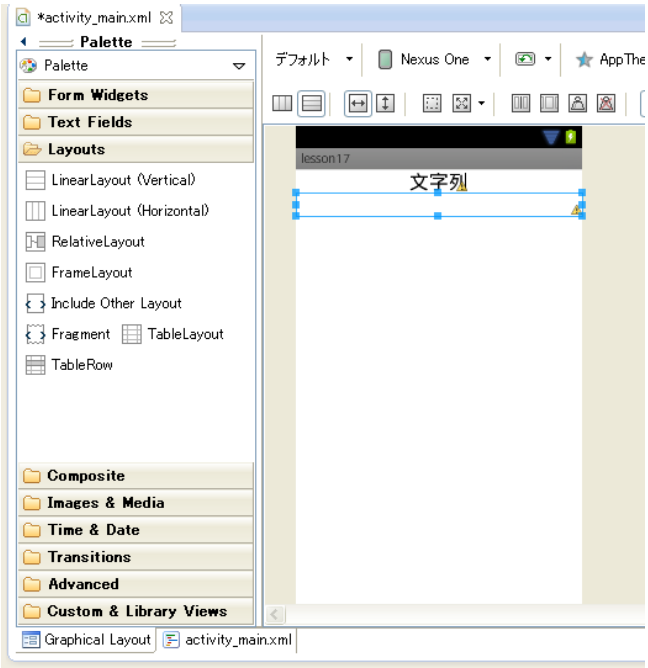
Resolved Value の上の箱の中に「文字列」と入力。OK。



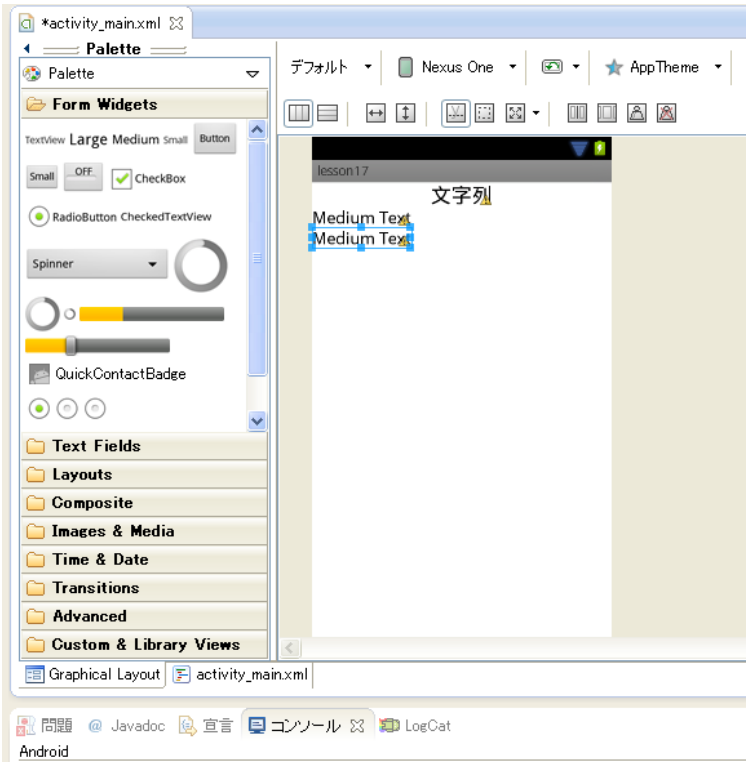
次に最初の文字列の上で右クリックし、その他のプロパティ→Layout Parameters→Layout Gravity→中央として文字列を画面中央にする。



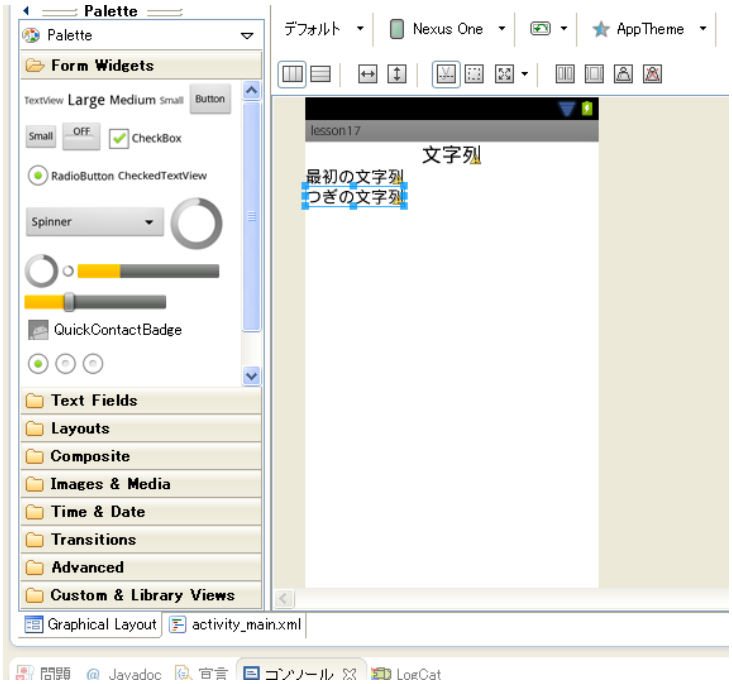
次に Layouts の LinearLayout(Horizontal)を最初の文字列の下に2回ドラッグ＆ドロップ



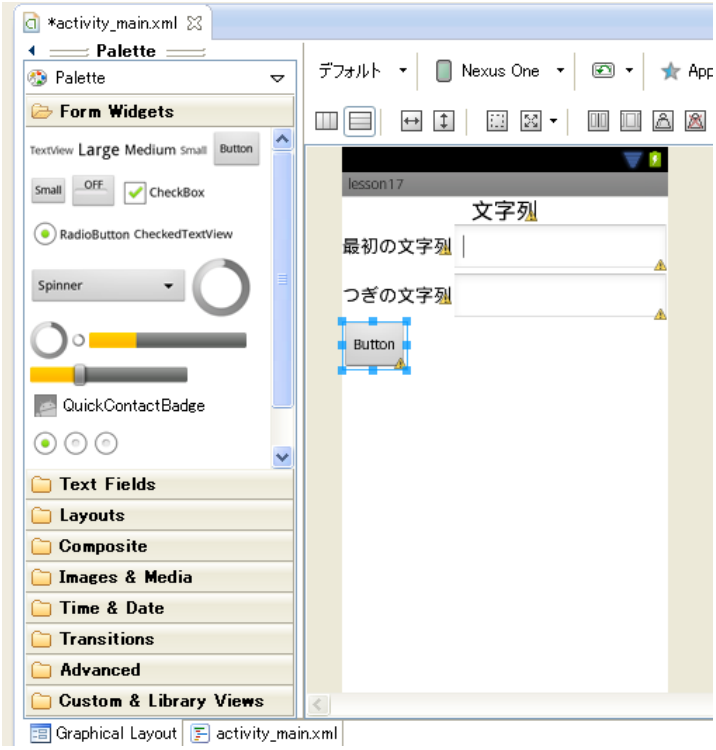
Form Widgets の Medium を入れた LinearLayout(Horizontal)の上に 2 回ドラッグ＆ドロップ。



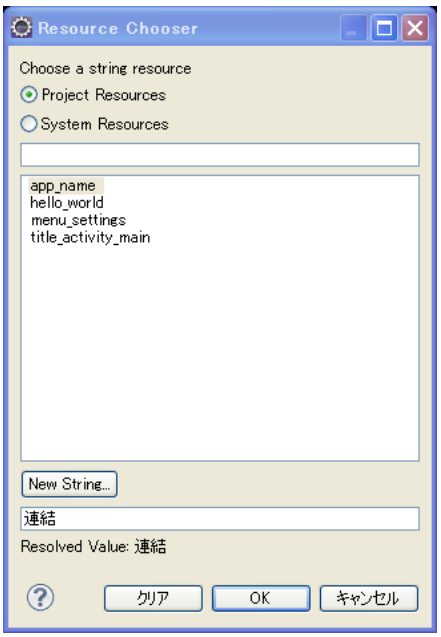
MediumText を最初の文字列、つぎの文字列にそれぞれ変更する。



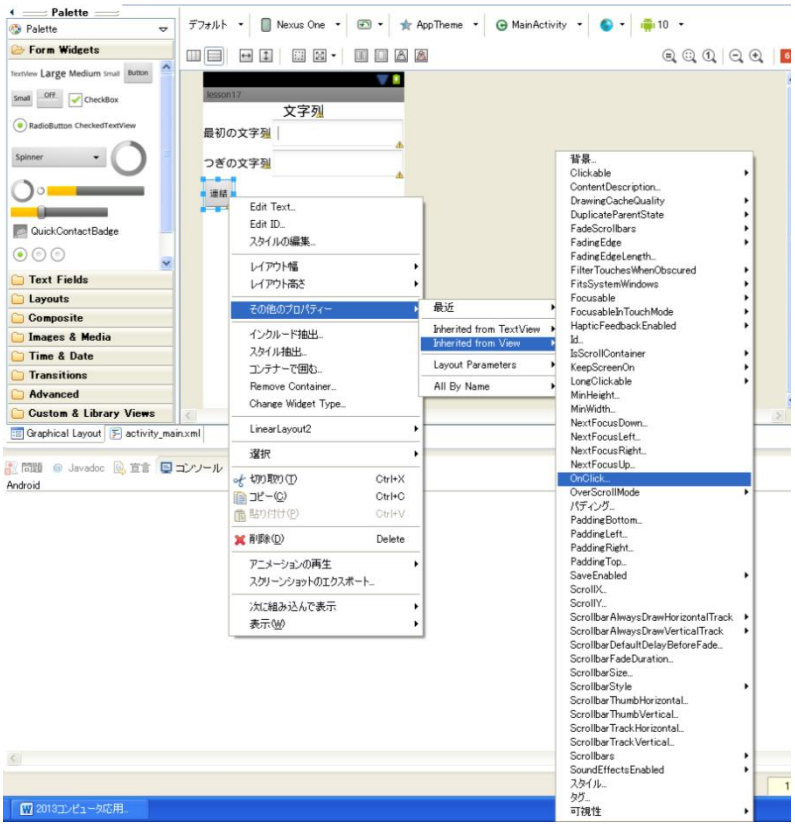
次に、Text Field にある FirstnameLastname を最初の文字列、つぎの文字列の横にドラッグ＆ドロップ、そして Form Widgets の Button をその下にドラッグ＆ドロップ。



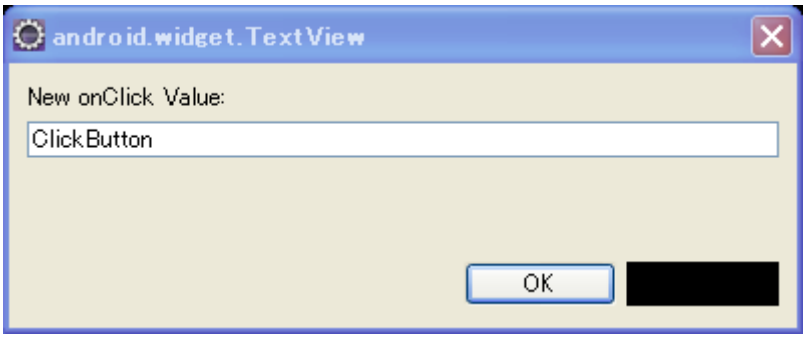
Button の上を右クリックして、「連結」に変更する。



ボタンの上を右クリック→その他のプロパティ→Inherited Form View→On Click



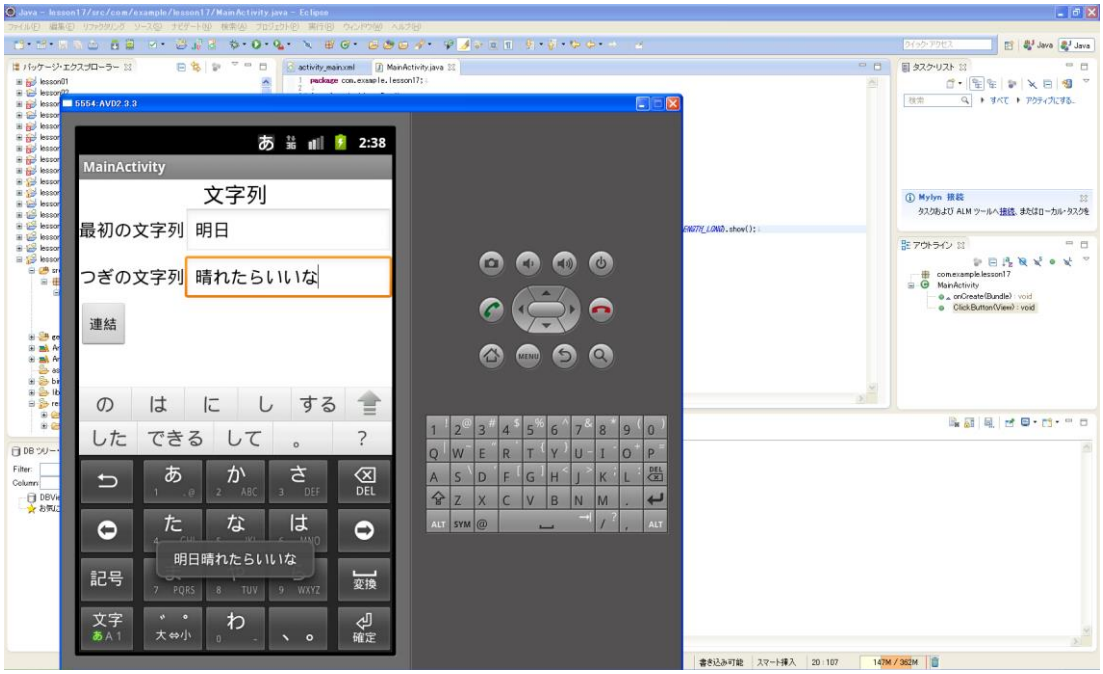
ClickButton という名前を付ける、そして上書き保存。



次に、MainActivity.java を開き以下のソースを打ち込みましょう。

```
package com.example.lesson207;

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

public class MainActivity extends Activity {

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

    public void ClickButton(View view){
        EditText edit01 = (EditText)findViewById(R.id.editText1);
        EditText edit02 = (EditText)findViewById(R.id.editText2);
        Toast.makeText(this,edit01.getText().toString()  +  edit02.getText().toString()  ,
Toast.LENGTH_LONG).show();
    }
}
```

（実行結果）



今回の文字の連結は、最終的にトーストで表示する方法ですが、これまで習った方法を利用すると、XML に更に TextView を追加し、そこに結果を表示させるという方法もありますし、プログラムからレイアウトビューを生成し、そこに結合した文字を表示するという方法もあります。

また、2 つの EditText に入力された文字列の比較を行うことで、パスワード確認などを行うことも可能です。

今回は、文字の連結でしたが、この技術と前期に習った方法を利用すれば、数字を扱うこともできます。たとえば、

```
String text1 = edit01.getText().toString();
int a = Integer.parseInt(text1);
String text2 = edit02.getText().toString();
int b = Integer.parseInt(text2);
EditText edit03 = (EditText)findViewById(R.id.editText3);
edit03.setText(Integer.toString(a+b));
```

というように EditText でとったデータを int 型に変更し、新たに設けた EditText に出力する。または、TextView に出力するという形をとることもできます。

Lesson8　画像を表示する（JAVA による ImageView の指定）

単純に画像を表示するには、Activity に Palette の Images&Media から ImageView をドラッグし、表示させたい画像を選択することで可能です。TextView 同様に ImageView もプログラムから指定して表示することができます。今回は、XML に用意した ImageView にプログラムから画像を指定する方法をやってみましょう。

まずは準備として画像をひとつ用意してください。その画像ファイルのファイル名を「photo」（拡張子はそれぞれ異なるので変更は禁止）として、横 240px 程度に編集して（画面サイズを超えない為に少し小さめにします）drawable フォルダにコピーします。※使用しないファイルがあった場合は必ず削除しておくこと！

画像の表示には ImageView を用います。まずは XML（activity_main.xml）の定義。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/msg1_label"
        />

    <ImageView
        android:id="@+id/imgview1_id"
        android:layout_width="280dip"
        android:layout_height="wrap_content"
        />

</LinearLayout>
```

画像を表示する枠の大きさを指定します。横は指定サイズで固定し、縦は画像の縦横費に合わせるように Wrap_content を指定します。

次に MainActivity.java の記述を以下の様にします。

```
package com.example.gazou;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.widget.ImageView;

public class MainActivity extends Activity {

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

        //イメージビューの取得
        ImageView imgView1 = (ImageView)findViewById(R.id.imgview1_id);
        //イメージビューに表示する画像を指定
        imgView1.setImageResource(R.drawable.photo);
    }

    @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;
    }

}
```

ImageView につけた ID を入力します

drawable に入れた画像のファイル名を入力します

これで画面上に画像が表示できるようになります。これができるようになると、プログラムの結果によって表示させる画像を簡単に変更することができるようになります。例えば、数字をランダムで発生させ、その結果によって表示する画像を変えるといった内容です。これは、次の Lesson でやってみましょう。

Lesson9　ボタンで画像がランダムに表示されるアプリ

それでは、画像を 5 つ用意してください、それを横 240px 程度に編集して保存、drawable フォルダに入れましょう。フォルダはどの分でもいいです。ファイル名は今回の場合一番目が photo1（拡張子はファイルによって変わるので強制的に変えてはダメ！）以降 photo2, photo3, photo4, photo5 まで作りましょう。

activity_main.xml は以下の様にします。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/msg1_label"
        />
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="280dip"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="@string/botan" />
</LinearLayout>
```

MainActivity.java の記述は以下のようにします。

```
package com.example.gazou;
import java.util.Random;
import android.app.Activity;
```

```
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.ImageView;
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;
    }
    public void onClick(View view){
        Random rnd = new Random();
        int ransu = rnd.nextInt(5);
        ImageView imgView1 = (ImageView)findViewById(R.id.imageView1);
        switch(ransu){
        case 0:imgView1.setImageResource(R.drawable.photo1);
                break;
        case 1:imgView1.setImageResource(R.drawable.photo2);
                break;
        case 2:imgView1.setImageResource(R.drawable.photo3);
                break;
        case 3:imgView1.setImageResource(R.drawable.photo4);
                break;
        case 4:imgView1.setImageResource(R.drawable.photo5);
                break;
        }
    }
```

これでボタンを押すごとに画像が入れ替わるようになります。今回はランダムで表示していますが、順番に表示していくパターンも作って見ましょう。

Lesson10 画面遷移

大きなプログラムを作ろうとすると、一つの Activity では表現が複雑になり、イベント毎にテキストビューやイメージビューを作成し直すなど、プログラミングが困難に感じることがあるかもしれません。Activity は一つだけでなく複数用意することができます。クラスやメソッド、イベント等に合わせて Activity を切り替えることで動きを持たせることができます。

実はこの Activity の表示は既にこれまでのプログラムに登場しています。

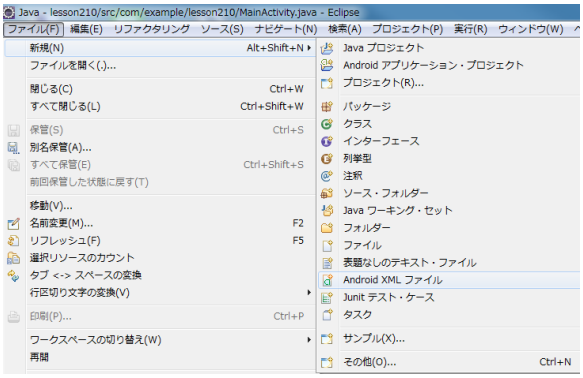
setContentView(R.layout.activity_main);

のことです。

これは、プロジェクトを作成した時に初めに用意されているレイアウト用 XML です。このレイアウト用 XML を追加することで、次の Activity を用意することができます。

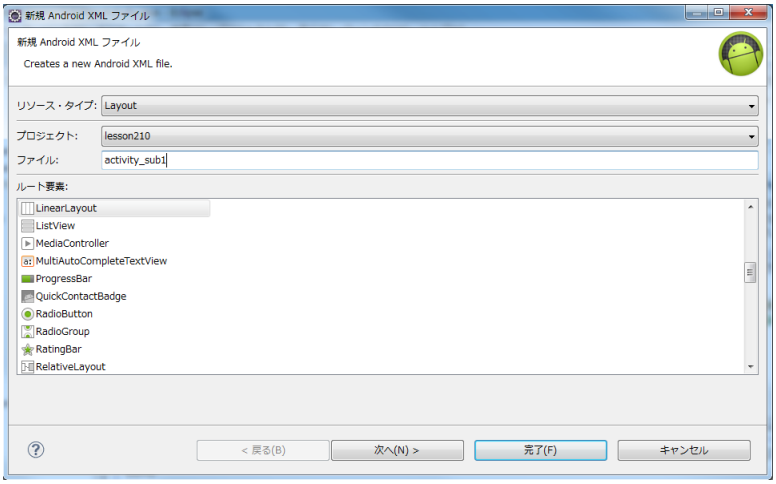
それでは、レイアウト用 XML を追加してみましょう。

① ファイル→新規→Android XML ファイルをクリック

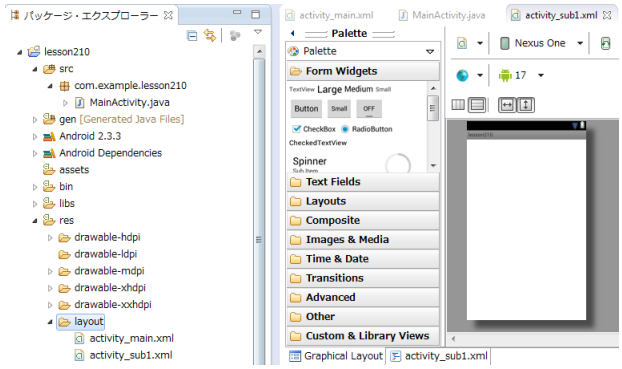


② その他→Android-XML レイアウトファイル

③ ファイル名を指定し「完了」をクリック



④　パッケージ・エクスプローラーから lesson210 の res の layout に XML ファイルが追加されたことを確認する。

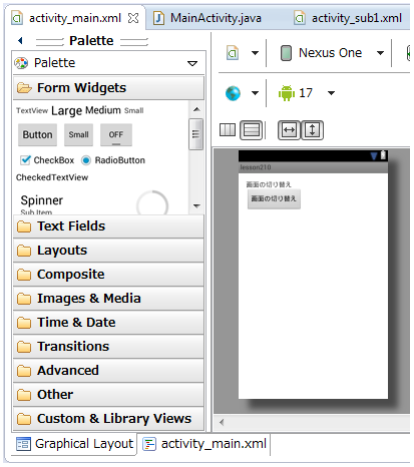


これで Activity が2つ用意できたことになります。後は、ソースでどの Activity を使うのかを指定するだけです。

それでは実際に、ボタンを押すことで「activity_main.xml」から「activity_sub1.xml」へ切り替わるプログラムを作成してみましょう。

まずは、下記のように Activity にボタンを追加し、イベントを設定します。

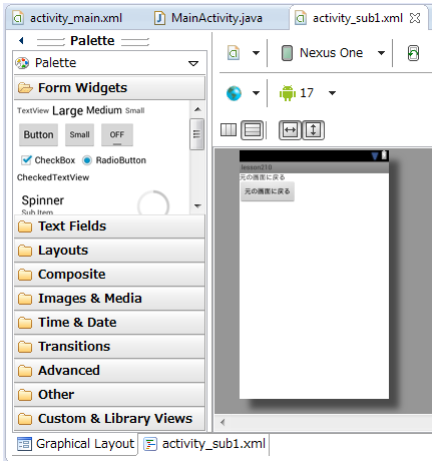
（activity_main.xml のグラフィカル・レイアウト）



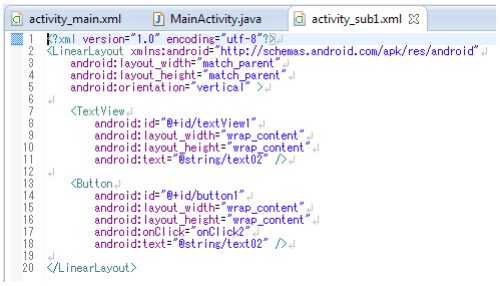
（activity_main.xml のソース）

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/text01" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClick1"
        android:text="@string/text01" />

</LinearLayout>
```

（activity_sub1.xml のグラフィカル・レイアウト）



（activity_sub1.xml のソース）



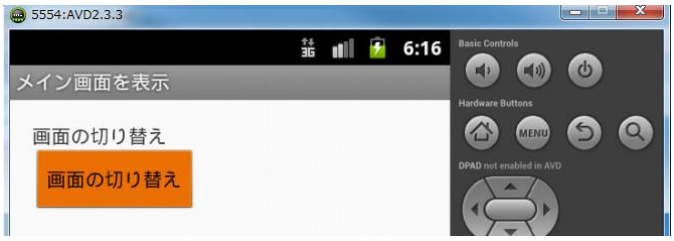
次に MainActivity.java にソースを入力します。

```
package com.example.lesson210;

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

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setTitle("メイン画面を表示");
    }
```

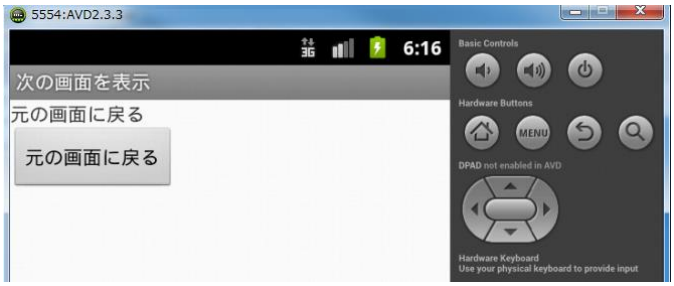
```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    public void onClick1(View view){
        setContentView(R.layout.activity_sub1);
        setTitle("次の画面を表示");
    }

    public void onClick2(View view){
        setContentView(R.layout.activity_main);
        setTitle("メイン画面を表示");
    }

}
```

入力が終わったら実行してみましょう。

（実行結果）



（実行結果）



XML で固定の画像を表示しておくとわかりやすいかもしれません。試してみましょう。

Lesson11 じゃんけんゲーム

ここからは、本格的にゲームアプリの制作を行っていきましょう。もちろんそれなりに完成度の高いものは時間も手間もかかりますので、比較的簡単なものから作って行きたいと思います。まずは、じゃんけんゲームを作りましょう。

XML ファイルは以下の通りです。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="184dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="51dp"
        android:layout_marginTop="30dp"
        android:text="じゃんけんしましょ。"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

    <Button
        android:id="@+id/button_g"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:onClick="onClick"
        android:text="グー" />

    <Button
        android:id="@+id/button_c"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:onClick="onClick"
        android:text="チョキ" />

    <Button
        android:id="@+id/button_p"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:onClick="onClick"
        android:text="パー" />

    </LinearLayout>

    <Button
        android:id="@+id/button1_99"
        android:layout_width="91dp"
        android:layout_height="wrap_content"
        android:onClick="onClick1"
        android:text="する。" />

</LinearLayout>
```

次に Java の記述は以下の通りです。

```
package com.example.janken01;
import java.util.Random;
```

```
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.TextView;

public class MainActivity extends Activity {
    private Random rnd = new Random();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        setTitle("じゃんけんゲーム");
    }
    @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;
    }
    public void onClick1(View view){
        final TextView gu_tyoki_pa = (TextView)findViewById(R.id.textView1);
        final Button bt1 = (Button)findViewById(R.id.button_g);
        final Button bt2 = (Button)findViewById(R.id.button_c);
        final Button bt3 = (Button)findViewById(R.id.button_p);
        gu_tyoki_pa.setText("じゃんけんホイ！");
        bt1.setOnClickListener(new OnClickListener() {
         public void onClick(View view) {
            gu_tyoki_pa.setText("私は、グー");
            int n = rnd.nextInt(3);
            switch(n) {
            case 0:
                gu_tyoki_pa.setText("PC はグー、あいこです");
```

```
            break;
        case 1:
            gu_tyoki_pa.setText("PC はチョキ、勝った！");
            break;
        case 2:
            gu_tyoki_pa.setText("PC はパー、負けた");
            break;
        }
    }
});
bt2.setOnClickListener(new OnClickListener() {
    public void onClick(View view) {
        gu_tyoki_pa.setText("私は、チョキ");
        int n = rnd.nextInt(3);
        switch(n) {
        case 0:
            gu_tyoki_pa.setText("PC はグー、負けた");
            break;
        case 1:
            gu_tyoki_pa.setText("PC はチョキあいこ");
            break;
        case 2:
            gu_tyoki_pa.setText("PC はパー、勝った！");
            break;
        }
    }
});
bt3.setOnClickListener(new OnClickListener() {
  public void onClick(View view) {
    gu_tyoki_pa.setText("私は、パー");
    int n = rnd.nextInt(3);
    switch(n) {
    case 0:
        gu_tyoki_pa.setText("PC はグー、勝った！");
        break;
    case 1:
```

```
                    gu_tyoki_pa.setText("PC はチョキ、負けた！");
                    break;
                case 2:
                    gu_tyoki_pa.setText("PC はパー、あいこ");
                    break;
                }
            }
            });
    }
}
```

正しく動作しましたか？ これまでと違いプログラムもかなり長くなってきましたので、打ち間違いなどがないように十分注意してください。この程度のプログラムを作る場合であってもこれだけの命令文が必要になります。これまでとは違いゲームなど大規模なものを作成するときには必ず設計図が必要になってきます。どこでどういった動きをさせるのか、その名称などはどうするのか、といったような内容を明確にしておかなければ、製作中に混乱してしまい、うまくいかなくなることが多々あります。しっかりと整理をしてから制作するようにしましょう。

氏　　名:

2019 年 1 月 21 日　初版発行

著者:伊藤雅彦、松本清一
発行:大阪アクティブラーニングスクール