Android Programming

Androidプログラミング

[完全入門]

アイデアさえよければ稼げる
Androidアプリを作って学生企業家をめざせ！

河西朝雄 著

本書の各節は、基本的に「文法や概念の説明」、「例題」、「練習問題」という３つで構成し、初心者がステップを踏んで学習できるように配慮しました。

KASAI.SOFTWARELAB

定価 1,620 円（税込）

はじめに

Android は、スマートフォンやタブレット PC などの携帯情報端末を主なターゲットとしたプラットフォーム（OS）です。Android は 2007 年に Google を中心にした規格団体「Open Handset Alliance」から発表され、2008 年から Android 対応のスマートフォンが多数販売されるようになりました。また、アプリケーションマーケットである Google Play Store(旧名称は Android Market)が提供されていて、2013 年 7 月時点で有料、無料含め 100 万を超えるアプリケーションが提供されています。Google Play Store を通して企業だけでなく、一般ユーザーが自作のアプリケーションを販売することができる点もいままでにない利点です。つまり、ソフト会社の技術者以外にも、学生を中心に一般の人でも Android アプリで商売や起業ができるようになる可能性があり Android アプリ市場は今後急速に普及すると思います。

Android のアプリケーションを開発するためには Android SDK、Android ADT、Eclipse などのツールをダウンロードし、Eclipse に Android ADT をインストールすることで Eclipse 上で開発を行います。Android のアプリケーションを開発するための言語は Java と XML です。

本書は何らかの言語でプログラム経験はあるが、Java や Android アプリを初めて勉強する人を主な対象とします。Android アプリを作るためには Java と XML の知識が必要になります。Java や XML を本格的に学ぶにはそれぞれ入門書が必要になります。本書では Android アプリを作りながら Java も XML も手っ取り早く学べるように工夫してあります。

そこでまず、Android グラフィックスを利用して画面に文字、直線、イメージなどを描画するプログラムを例に Java の基本的な言語仕様について 2 章で学びます。次にボタンやテキストビューなどのウイジェットを XML で記述する方法を 3 章で学びます。ここまでが Java と XML の基本事項になります。4 章、5 章、7 章、8 章でウイジェットの配置方法作や使い方をさらに説明します。Android 的プログラミングの基礎としてタッチイベント、インテントとアクティビティ、Handler とサービスについて 6 章、9 章、10 章で説明します。各種処理としてサウンドとムービー、アニメーション、グラフィックス、ファイル処理について 11 章～14 章で説明します。Android 特有の機能として GoogleMap、センサーとカメラ、音声合成、音声認識について 15 章～18 章で説明します。最後にプログラミング力をアップするために 19 章でリバーシーゲームを段階を追って作成します。ということで本書は以下のような章構成となっています。

1 章　Java による Android アプリの作り方
2 章　Android グラフィックスによる Java 入門
3 章　ウイジェットと XML
4 章　レイアウトと Graphical Layout

5 章　main.xml を使わずにレイアウトする
6 章　タッチイベント
7 章　トースト、ダイアログ
8 章　各種ウィジェット
9 章　インテントとアクティビティ
10 章　Handler とサービス
11 章　サウンドとムービー
12 章　アニメーション
13 章　グラフィックス
14 章　ファイル処理
15 章　GoogleMap
16 章　センサーとカメラ（実機のみ）
17 章　音声合成
18 章　音声認識（実機のみ）
19 章　リバーシーゲーム

本書の各節は、基本的に「文法や概念の説明」、「例題」、「練習問題」という３つで構成し、初心者がステップを踏んで学習できるように配慮しました。本書一冊で Android の基礎すべてが完全理解できます。

これから Android アプリの開発を志す方々にとって、本書が少しでもお役に立てば幸いです。

2014 年 6 月　河西朝雄

○本書の開発環境は以下です。
・Android SDK
　android-sdk_r21.0.1-windows
・Eclipse
　Eclipse 4.2 Juno(英語版)
○エミュレータでの開発ターゲットは Android 4.2(API 17)ですが Android 2.2(API 8)でも動作確認しています。
○実機は「SAMSUNG GALAXY S」（Android 2.2(API 8)）と「SAMSUNG GALAXY S Ⅲ」（Android 4.1(API16)）で確認しました。
○Android、Android SDK、Eclipse の最新情報についてはカサイ．ソフトウェアラボの電子書籍サイト（http://kasailab.jp/）を参照して下さい。

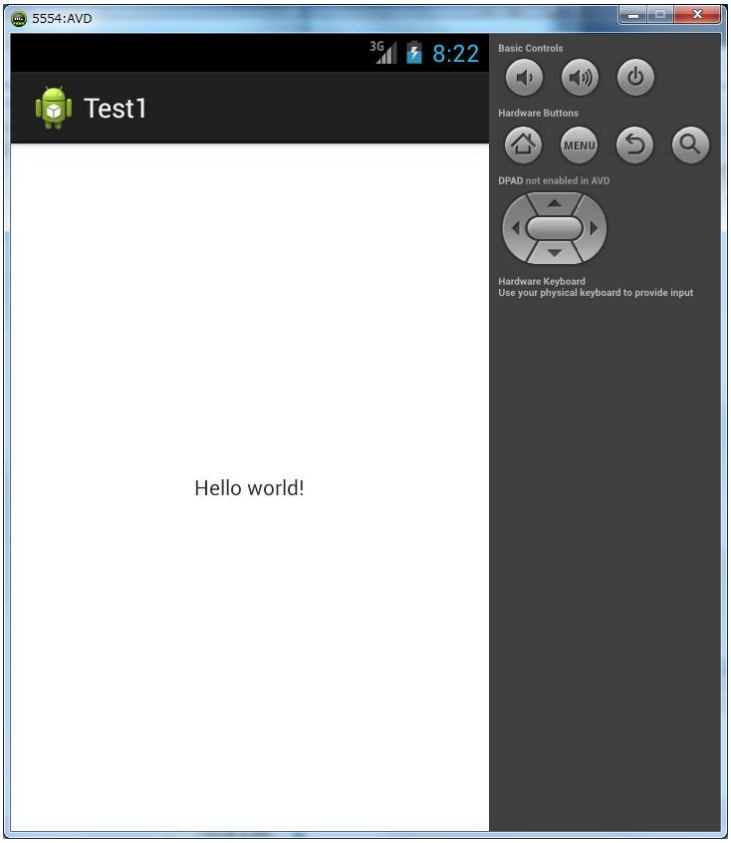
目次

1章　Javaによる Androidアプリの作り方

以下の作業が終わって Eclipse で Android アプリを開発する環境が整備されているものとして話を進めます。

・Android SDK のインストール
・Eclipse のインストール
・Eclipse への Android Plugin（ADT）のインストール
・Android アプリを実行するために必要な AVD（Android Virtual Device）の作成

この章ではデフォルトのスケルトン（システムが自動生成するプログラムの骨格）を使って「Hello world!」というメッセージを TextView（テキストビュー）に表示するプログラムの作り方の手順を説明します。また作成されたファイルの意味と役割を説明します。



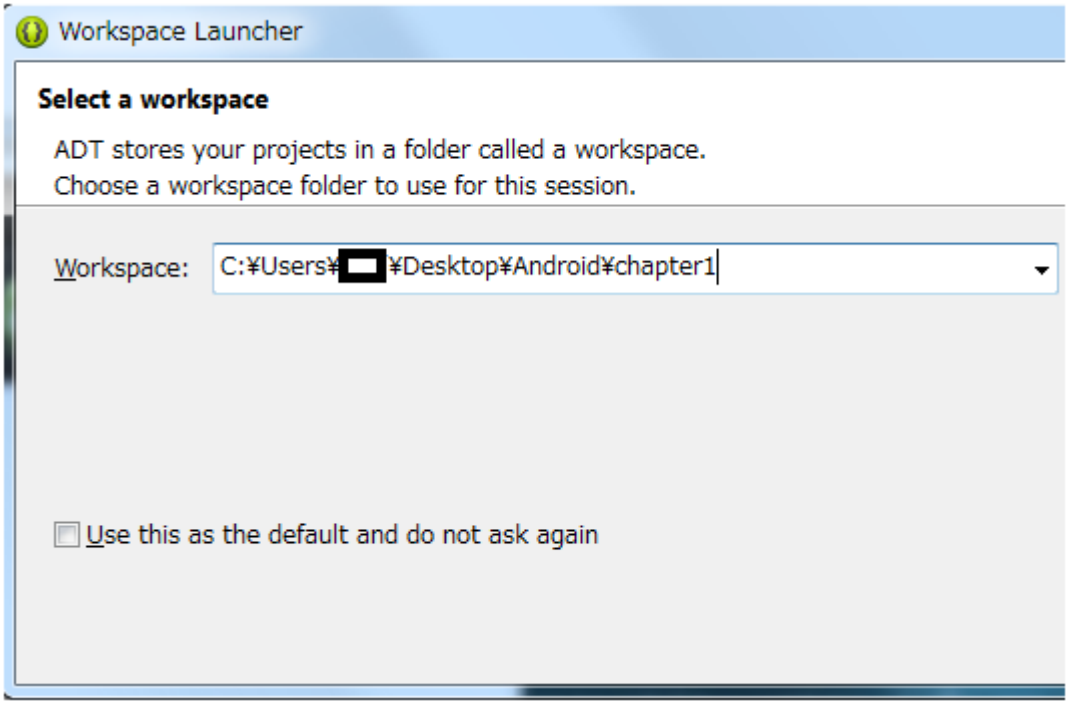
「注」Eclipse や Android SDK のバージョンにより作業手順が異なる場合があります。

1-1　プロジェクトの作り方

Android アプリはプロジェクトで管理します。プロジェクトを保管するためのフォルダをワークスペースと呼びます。

1.　ワークスペースの作成

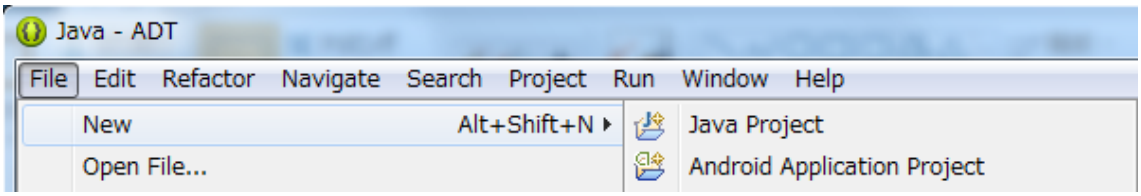
Eclipse を起動し「Workspace Launcher」画面でワークスペース名を入力します。ここでは、デスクトップの「Android」フォルダにワークスペースを「chapter1」として作成します。「Use this as the default and do not ask again」にチェックが入っているとワークスペースの設定画面は出ません。



2.　プロジェクトの作成

以下に示す①～⑤の手順でプロジェクトを作成します。

①「File」－「New」－「Android Application Project」を選択します。



②アプリケーション名等の入力

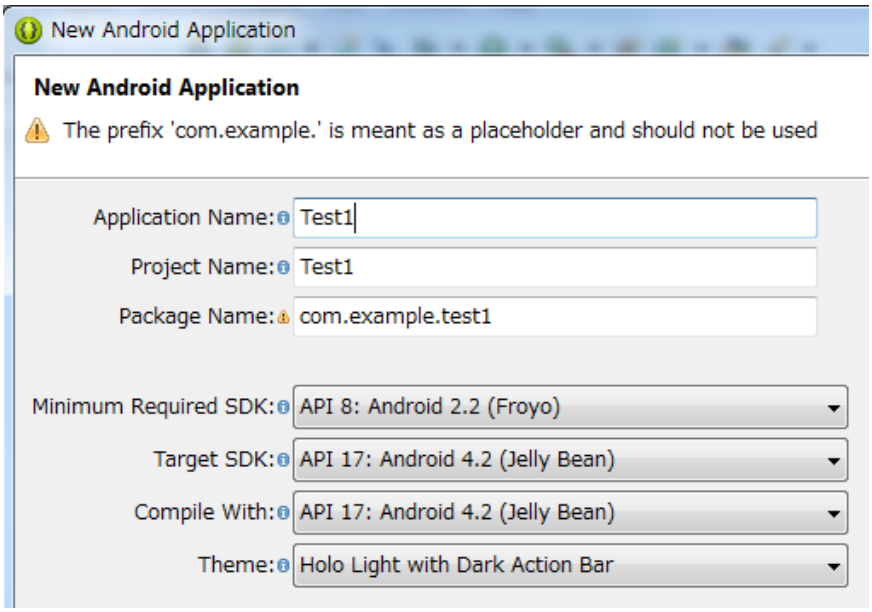
・Application Name、Project Name、Package Name

Application Name を「Test1」と入力します。自動で Project Name は「Test1」、Package Name は「com.example.test1」となりますので、そのままとします。 1 つの Java アプリを構成する各種ファイルを管理するための基本をプロジェクトと呼びます。プロジェクト名のフォルダ内に各種ファイルが格納されます。

・ SDK バージョン

AndroidSDK のバージョン（Target SDK と Compile With）を選択します。ここでは「API 17:Android 4.2」を選択しました。

Minimum Required SDK はアプリケーションが動作する最低の Android SDK バージョンです。ここでは「API 8:Android 2.2」を選択しました。



「注」SDK バージョンと API レベル

Android SDK バージョン	API レベル	コードネーム
4.3	18	Jelly Bean
4.2	17	Jelly Bean
4.1	16	Jelly Bean
4.0.3	15	Ice Cream Sandwich
4.0	14	Ice Cream Sandwich
3.2	13	Honeycomb
3.1	12	

3.0	11	
2.3.4(2.3.3)	10	Gingerbread
2.3	9	
2.2	8	Froyo
2.1	7	Eclair
2.0.1	6	
2.0	5	
1.6	4	Donut
1.5	3	Cupcake
1.1	2	非公開
1.0	1	非公開

③Configure Project

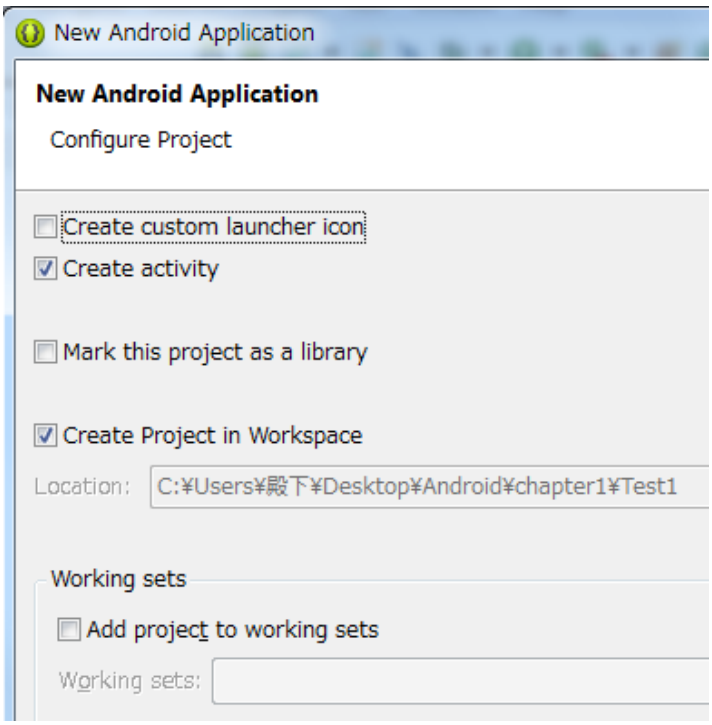
プロジェクトの構成を指定します。

・Create custom launcher icon

独自に launcher icon を作る場合はチェックを入れます（デフォルト）。ここではデフォルトのアイコンを使用するのでチェックを外します。

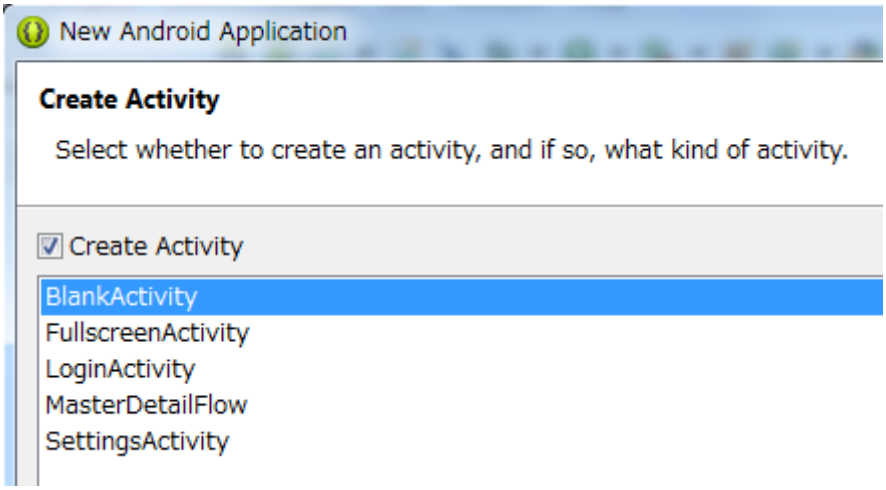
・Create activity

Android 画面の基本は Activity で、このクラスを自動的に作る場合に□にチェックを入れます（デフォルト）。



④アクティビティの作成

BlankActivity（通常のアクティビティ）を選択します。



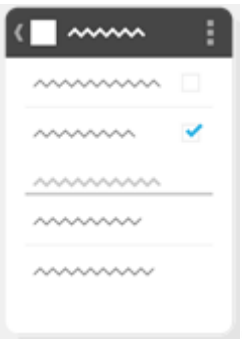
「注」Android 4.2 では以下の 5 種類のアクティビティを選択できるようになりました。


BlankActivity


FullscreenActivity


LoginActivity


MasterDetailFlow

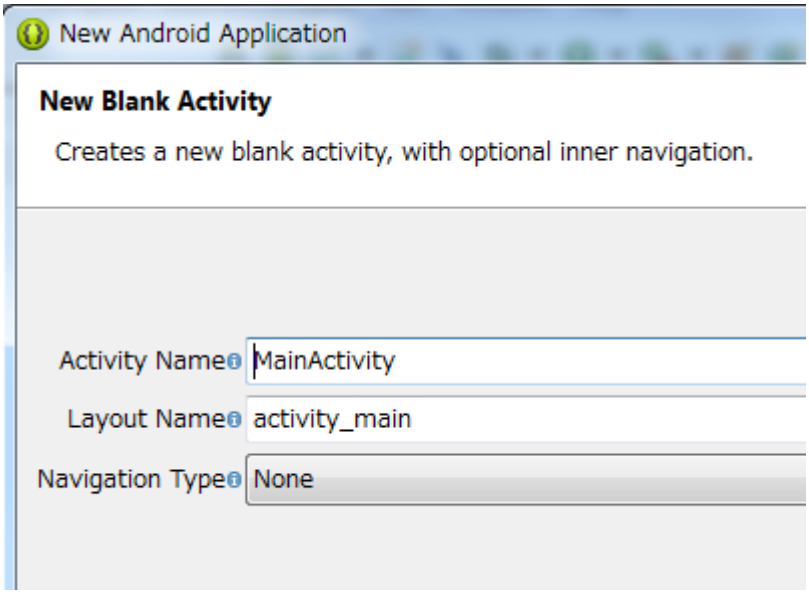

SettingsActivity

⑤Activity Name、Layout Name の入力

デフォルトで以下のような名前になっていますので、このままとします。古いバージョンでは Activity Name は「Test1Activity」、Layout Name は「main」となっていました。

Activity Name:MainActivity

Layout Name:activity_main



2章　Androidグラフィックスによる Java 入門

Androidアプリを作るためにはJavaとXMLの知識が必要になります。JavaやXMLを本格的に学ぶにはそれぞれ入門書が必要になります。本書ではAndroidアプリを作りながらJavaもXMLも手っ取り早く学べるように工夫してあります。そこでまず、Androidグラフィックスを利用して画面に文字､直線､イメージなどを描画するプログラムを例にJavaの基本的な言語仕様について学びます。Androidグラフィックスを例にしたのは、Androidグラフィックスではの知識は必要ないこと､視覚的にも興味のある結果が得られ学習のモチベーションが上がることです。

この章ではAndroidアプリを作る上で当面必要なJavaの基礎知識として以下の内容を説明します

・オブジェクト指向言語特有の概念
　クラス、インスタンス（オブジェクト）、コンストラクタ、メソッド
・C言語などの基本言語と共通の概念
　変数、演算子、for文やif文などの流れ制御文、配列
・Androidグラフィックスに特有な概念
　描画メソッド、ビットマップ

「注」この章のプログラムリストについて

最初の例題のみ全リストを掲載してありますが、その後の例題はGViewクラス内のみが変更されるので、GViewクラスだけをリストとして掲載してあります。最初の例題をGraph1としています。その後の例題はGraph2、Graph3・・などと別なプロジェクトを新たに作ってもよいですが､Graph1のonDraw内だけを変更してもよいです。ただしこの場合は前に作ったプログラムは無くなってしまいますので注意してください。

2-1　Android グラフィックスの基礎

最初の例は「Android」という文字をグラフィックスで表示します。Android グラフィックスは View クラスを元にして行います。描画処理は onDraw メソッドの中に記述します。描画に当たっては、まず Paint クラスのメソッドを使って描画色やテキストサイズを指定し、次に Canvas クラスのメソッドを使ってテキストの描画を行います。

1.　View クラス

グラフィックスを描画する画面は View クラスを継承して作ります。以下は GView という名前のユーザ定義クラスを作っています。コンストラクタの GView はスーパークラスのコンストラクタを呼び出す部分でこれが定型です。ユーザが行う描画処理は onDraw メソッド内に記述します。onDraw メソッドが呼び出されたときに Canvas クラスの引数 canvas に描画オブジェクトが渡されますので、この canvas に対し描画メソッドを使ってグラフィックス処理を行います。onDraw メソッドの仮引数を「Canvas canvas」としていますが、仮引数の名前はなんであってもよいので「Canvas c」などとしても構いません。

```
private class GView extends View { ←View クラスを継承したユーザ定義クラス Gview
    public GView(Context context) { ←コンストラクタ
        super(context);
    }
    protected void onDraw(Canvas canvas) { ←onDraw メソッド
        描画内容を記述                    ↑Canvas クラスの引数 canvas
    }
}
```

この GView クラスを実際に画面に設定するには、「setContentView(R.layout.main);」の代わりに「setContentView(new GView(this));」とします。前者はウイジェットを配置したレイアウトを表示し、後者はグラフィックス画面を表示します。

2.　Paint クラスとオブジェクト（インスタンス）

Paint クラスは描画色や文字サイズなどの描画情報を扱うクラスです。クラスからインスタンスを生成するには new 演算子を用いて、以下のように宣言します。コンストラクタはクラス名と同じ名前の特別なメソッドで初期化を行うものです。これで paint という名前の Paint クラスのオブジェクトが生成されます。以後 paint に対しメソッドを適用します。

```
↓クラス名          ↓コンストラクタ
Paint paint=new Paint();
      ↑オブジェクト（インスタンス）
```

「注」オブジェクトとインスタンス

クラスはオブジェクトを生成するための金型（テンプレート）のようなものと考えることができます。オブジェクト指向言語では、金型から実際に生成された実体をオブジェクトと言い、クラスからオブジェクトを作ることをインスタンス化（instantiation）と言います。C++ではクラスを実体化したものをオブジェクトと呼び、Java ではインスタンス化したオブジェクトをインスタンスと呼びます。この場合インスタンス＝オブジェクトと考えてよいです。本書では「インスタンス」という言葉が適切な場合以外は「インスタンス」と「オブジェクト」を使い分けずに「オブジェクト」と呼ぶことにします。

3. Paint クラスのメソッド

描画色を緑、フォントのサイズを 40 ピクセルに設定するには、paint オブジェクトに対し setColor メソッド、setTextSize メソッドを使って以下のようにします。「Color.BLUE」は青色を示す Color クラスの定数です。

```
paint.setColor(Color.BLUE); ←青色
paint.setTextSize(40); ←テキストのサイズを 40 ピクセル
```

4. Canvas クラスの描画メソッド

onDraw メソッドの引数 canvas に対し描画を行うには drawText メソッドを用います。指定する x,y 座標はテキストの左下隅の座標です。

```
canvas.drawText("Android", 10,50, paint);
                                  ↑描画位置の x,y 座標
```

Android

↑左下隅の座標を表示位置として指定

「注」メソッドとは

メソッドはある処理を行う機能単位で、ユーザは引数というデータをこのメソッドに渡して目的の処理を行います。

「注」メソッドの種類

onDraw と drawText はどちらもメソッドですが、呼び出し方（呼び出され方）が異なります。通常 onXXX メソッドはシステムから呼び出されるメソッドで、処理内容をユーザが定義します。たとえば、onDraw メソッドは画面に描画を行う必要が生じたときに呼び出されます。これに対し「オブジェクト.メソッド(引数,・・・)」の形式で呼び出すメソッドはユーザが直接メソッドを呼び出します。

「注」インデント

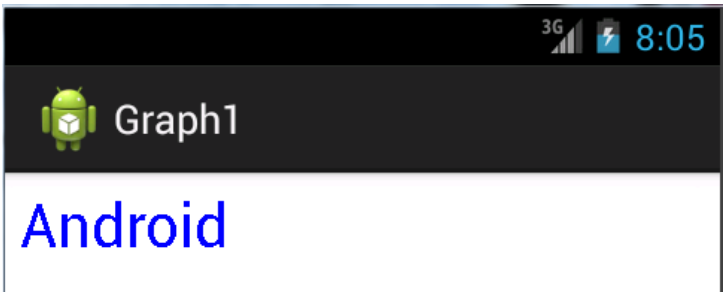
プログラムのブロック構造を分かりやすくするため内側のブロックになるほど書き出す位置を右にずらすことをインデント（字下げ）と言います。Eclipse では自動的に 4 文字幅のインデントをとります。

「例題 2-1」以上をまとめた全リストを示します。それぞれの位置関係を確認してください。

・MainActivity.java

```
package com.example.graph1;

import android.app.Activity;
import android.content.Context;
import android.graphics.*;
import android.os.Bundle;
import android.view.View;
import android.view.Menu;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new GView(this));
    }
    private class GView extends View {
        private Paint paint;
        public GView(Context context) {
            super(context);
            paint=new Paint();
```

```
        }
        protected void onDraw(Canvas canvas) {
            paint.setColor(Color.BLUE);
            paint.setTextSize(40);
            canvas.drawText("Android",10,50,paint);
        }
    }

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

}
```



「注」onCreateOptionsMenu メソッドは削除しても構いません。

「注」背景色

古い開発環境（2012 年 6 月頃までの Android SDK）ではデフォルトの背景色は黒背景でしたが、新しい開発環境では白背景です。デフォルトの背景色を変えるには onDraw メソッドの先頭で以下のように背景色を設定します。背景白にするには Color.WHITE とします。

```
Protected void onDraw(Cavas canvas){
    canvas.drawColor(Color.BLACK);
```

「注」Android Lint Checks

古い開発環境では以下のように onDraw メソッド内で Paint オブジェクトを取得しても警告エラーになりませんでした。

```
protected void onDraw(Canvas canvas) {
    Paint paint=new Paint();
    paint.setColor(Color.BLUE);
    paint.setTextSize(40);
    canvas.drawText("Android",10,50,paint);
}
```

新しい開発環境では「Paint paint=new Paint();」のところで「Avoid object allocations during draw/layout operations (preallocate and reuse instead)」という警告エラーがでます。これは onDraw が呼ばれるたびに Paint オブジェクトを生成するという無駄を止め、GView 内で一度生成したものを使いまわしなさいということです。このような厳しいチェックを「Android Lint Checks」と言います。

このため、本書では paint をメソッド外で宣言し、GView コンストラクタ内で Paint オブジェクトを生成し、onDraw メソッドで paint を使用するという多少煩雑な方法をとっています。

```
private class GView extends View {
    private Paint paint; ←paintの宣言
    public GView(Context context) {
        super(context);
        paint=new Paint(); ←Paintオブジェクトの生成
    }
    protected void onDraw(Canvas canvas) {
        paint.setColor(Color.BLUE); ←paintの使用
        paint.setTextSize(40);
        canvas.drawText("Android",10,50,paint);
    }
}
```

警告エラーを無視するなら上のように onDraw 内に全てまとめる方が簡単です。

3章　ウイジェットとXML

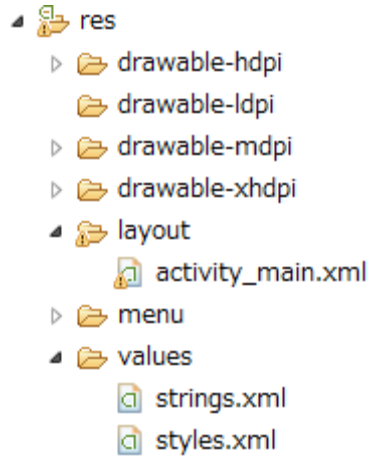
Androidではテキストビューやボタンなどのの GUI 部品をウイジェット（Widget）と呼んでいます。ウイジェットは main.xml という XML ファイル中で定義します。

XML(Extensible Markup Language : 拡張可能マークアップ言語)は文書やデータの意味や構造を記述するためのマークアップ言語の一つです。Android ではレイアウトファイル（main.xml）、文字列リソースファイル（string.xml）、マニフェストファイル（AndroidManifest.xml）などが XML で記述されています。XML では利用者が自由に要素や属性を定義でき、プログラムで XML 要素を操作できる（具体的には findViewById メソッドで main.xml に設定したウイジェットを取得するなど）というメリットがあります。

この章ではウイジェットを記述するための XML の一般的な書き方と、ボタン、テキストビュー、エディトテキスト、チェックボックス、ラジオボタン、スピナー、リストビュー、イメージビューなどの個々のウイジェットの具体的な使い方を説明します。ウイジェットをクリック（タッチ）したときのアクションを監視するものをリスナーと呼びます。このリスナーをウイジェットに組み込む方法を説明します。

3-1　Android の XML ファイル

Android で使用している XML はレイアウトファイル（activity_main.xml）や文字列リソースファイル（string.xml）に記述されています。これらのファイルは res/layout や res/values フォルダに格納されています。



・activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world" />

</RelativeLayout>
```

「注」古いバージョンではレイアウトファイルのデフォルト名は main.xml でした。

「注」LinearLayout と RelatireLayout

新しいバージョンの Android SDK ではデフォルトで生成される activity_main.xml の

レイアウトがRelatireLayoutになっています。古いバージョンではLinearLayoutでした。これは4章で説明するGraphical Layoutでウィジェットを配置する場合、Relative Layoutの方が都合よいからです。XMLコードで直接指定する場合はLinearLayoutの方が簡単なので、この章の例題はLinearLayoutを使います。

・string.xml

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

    <string name="app_name">Test1</string>
    <string name="hello_world">ようこそAndroid!</string>
    <string name="menu_settings">Settings</string>

</resources
```

1. XMLの書式

XMLでは、文書構造を構成する個々のパーツを「要素」（エレメント：Element）と呼びます。要素は要素を示すタグと内容で構成されます。タグは開始タグと終了タグがあり、その2つのタグの間に「内容」を記述します。この「内容」は定義する文字列であったり別の要素であったりします。開始タグの中にはより細かな指定を属性で行うことができます。属性にはそれぞれ値を指定します。これらの各項目は行を分けて書いても、1行にまとめて書いても同じです。

<開始タグ
　　属性=値
　　属性=値>
　　内容
</終了タグ>

たとえばstring.xmlには以下のような要素が記述されています。

↓開始タグ　　↓内容
<string name="app_name">Test1</string>
　　↑属性　　↑終了タグ

「注」本書では XML の各要素のインデントはネストが深くなるごとに 4 文字の空白を置くことにしています。

2. XML 宣言（<?xml ?>）

XML 宣言はそのファイルが xml で記述されていることを XML パーサに指示するためのもので「<?xml」で始まり「?>」で終わります。version 属性に XML バージョン、encoding 属性に文字エンコーディングを指定します。現在の XML バージョンは 1.0 です。Android の文字エンコーディングは utf-8 です。

・UTF-8

UTF-8（Unicode Transformation Format）は Unicode を 8 ビット単位の可変長コード（1～4 バイト）にエンコードする方式でインターネットで広く使われているエンコード方式です。ASCII コード対応文字は ASCII コードのまま 1 バイトですが、日本語は 3~4 バイトになります。

3. XML 名前空間（xmlns）

この XML ファイルで使用する要素や属性に関する名前空間（NameSpace）が定義されている場所を指定します。この指定はルート要素にだけ置きます。

```
xmlns:android="http://schemas.android.com/apk/res/android"
        ↑接頭辞          ↑名前空間 URI（定義場所）
```

このように指定することで、接頭辞の「android」を使って、各要素の属性を「android:id」や「android:layout_width」のように指定します。

4. レイアウト

ボタンやテキストビューなどのウイジェットを配置する方法をレイアウトと呼び、Android では LinearLayout、RelativeLayout、FrameLayout、TableLayout などがあります。古いバージョンのデフォルトのレイアウトは LinearLayout ですが、新しいバージョンのデフォルトのレイアウトは RelativeLayout です。XML で直接記述する場合は LinearLayout の方が簡単なので本書では基本的には LinearLayout を使用します。LinearLayout は<LinearLayout>で始まり</ LinearLayout>で終わりますす。この 2 つのタグの間にウイジェットを置きます。LinearLayout では宣言されたウイジェットの順序で単純（直線的）にウイジェットを並べます。レイアウトに指定する主な属性は以下です。

・android:orientation

ウイジェットを並べる方法で"vertical"(垂直)、"horizontal"(水平)を指定します。指定し

なければ"horizontal"とみなされます。

・android:layout_width、android:layuot_height

レイアウトの幅と高さを指定します。値には"fill_parent"（可能な限り拡大）または"wrap_content"（表示に必要なサイズ）または具体的な数値を指定します。

「注」fill_parent と match_parent

API 8 からは fill_parent の代わりに match_parent を使用することが推奨されていますが、古い環境のことも考えて本書では従来の「fill_parent」を使用しています。新しい環境に合わせたい場合を「match_parent」を使用してください。

5. ウイジェット

ウイジェットとは一般に、デスクトップの好きな位置に置いておくことのできる小さなアプリケーション（カレンダー、ノートパッド、地図、検索など）を指します。Apple や Yahoo!などが「ウィジェット」と呼び、Google や Microsoft は「ガジェット」と呼んでいます。widget、gadget とも元の英語の意味は「小さな道具」という意味です。

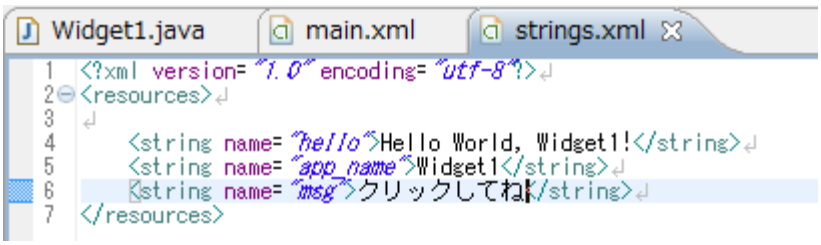
Android ではボタンやテキストビューのような予め定義された UI（UserInterface）部品を標準ウイジェット（単にウイジェット）と呼んでいます。

ボタンは XML では次のように定義します。

```
<Button
    android:id="@+id/button"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="クリックしてね"
/>
```

LinearLayout で android:orientation="vertical"の場合は縦に並べていくのでウイジェットの layout_width は"fill_parent"（横幅一杯に広げる）、layuot_height は"wrap_content"（実際に表示される内容の大きさ）とします。

「注」チェックの厳しい環境では文字列リソースを使わずに「android:text="クリックしてね"」のように直接文字列を指定すると警告エラーとなります。警告を回避するには string.xml に以下のようにリソースを定義し、「android:text="@string/msg"」とします。



本書では簡便のために、文字列リソースを使わずに直接文字列を指定していますので警告エラーは無視してください。

6. ウイジェットの ID（android:id）

ウイジェットの ID は「android:id」属性に指定します。Java コードから findViewById メソッドを使ってウイジェットを取得する際の ID を指定します。ID の指定方法には以下の 2 種類があります。

①android:id="@+id/xxx"

リソースファイル（main.xml など）で定義しているウィジェットのユーザが定めた ID を指定します。文字列の先頭にあるアットマーク(@)は、XML パーサに解析をさせ、id 以降の文字列を展開し、それを ID リソースとして識別させるということを指示しています。プラスマーク(+)は、生成される R.java ファイル内で、リソースのひとつとして追加される必要がある新しいリソースの名称であることを意味しています。

②android:id="@android:id/xxx"

android 名前空間でシステムが予め定めている ID を指定します。ListView などで使用することがあります。Android のリソース ID を参照する場合、プラスマークは不要ですが、「android:id="@android:id/empty"」のように android パッケージネームスペースを追加する必要があります。パッケージネームスペース付きの場合はローカルのリソースクラスからではなく、android.R リソースから ID を参照するようになります。

7. ウイジェットのテキスト（android:text）

「android:text」にはウイジェットに表示するテキストを指定します。文字列リソースを使う場合は「@string/リソース名」を指定します。

android:text="@string/hello"

直接文字列を指定する場合は""内にその文字列を指定します。

android:text="クリックしてね"

8. 空タグ

ボタンやテキストビューなどの多くのウイジェットは開始タグと終了タグの間に別のタグを持ちません。このようなタグを空タグと呼びます。正規な書き方なら、次のように開始タグと終了タグを書きます。

```
<Button
    属性=値>
</Button>
```

しかしこれは煩雑になるので、開始タグの最後を/>で終わることで、終了タグを省略する便法があります。

```
<Button
    属性=値/>
```

9. R.java

Java からリソースにアクセスするためのクラスが gen フォルダの R.java に自動生成されています。このファイルは自動生成されるもので、ユーザが変更を加えてはいけません。たとえば、以下のようなリソースを定義したとします。

・activity_main.xml に配置したウイジェットの TextView(id は text)と Button(id は button)

・drawable のイメージ ic_launcher.png と white.png

・string.xml で定義した文字列リソースの"hello_world"と"app_name"

この場合の R.java の定義内容は以下のようになります。リソースの種別ごとに、0x7f020000 番代は drawable リソース、0x7f050000 番代は id 属性で定義されたレイアウトやウイジェットの ID などと内部管理用の ID が割り当てられます。

・R.java

```
public final class R {
    public static final class array {
        public static final int items=0x7f050000;
    }
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000; ←イメージファイル
        public static final int white=0x7f020001;
    }
    public static final class id {
        public static final int button=0x7f080000; ←ウイジェットのID
        public static final int menu_settings=0x7f080002;
        public static final int text=0x7f080001;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000; ←レイアウトファイル
    }
    public static final class menu {
        public static final int activity_main=0x7f070000;
    }
    public static final class string {
        public static final int app_name=0x7f040000; ←文字列リソース
        public static final int hello_world=0x7f040001;
        public static final int menu_settings=0x7f040002;
    }
    public static final class style {
    // 省略
    }
}
```

R.java で定義される主なリソース ID クラスは以下があります。

・layout クラス

layout リソースの xml ファイル名。0x7f030000 からの通し番号が割り当てられます。

・id クラス

id 属性で定義されたレイアウトやウイジェットの ID。0x7f050000 からの通し番号が割り当てられます。

・drawable クラス

drawable リソースのイメージファイル名。0x7f020000 からの通し番号が割り当てられます。

・string クラス

string.xml に記述されている文字列リソース名。0x7f040000 からの通し番号が割り当てられます。

10. R.layout.main のロード

アプリケーションをコンパイルすると、各 XML レイアウトファイルは View リソース内にコンパイルされます。アプリケーションコードからレイアウトをロードする必要があり、そのコードは Activity.onCreate() のコールバックメソッドで実装することになります。setContentView()にレイアウトリソースの参照を「R.layout.レイアウトファイル名」の形で引数として渡してそのメソッドを呼び出します。例えば、XML レイアウトが activity_main.xml の場合、以下のようにしてアクティビティにロードします。

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

11. findViewById によるウィジェットの取得

main.xml で定義されているウイジェットを Java 側から操作するためには findViewById メソッドで、ウイジェットオブジェクトを取得します。main.xml でボタンの ID を「android:id="@+id/button"」と定義していた場合、R.java ではこの ID は Java 用に「R.id.button」と定義し直されていて、この ID を findViewById の引数に指定します。

```
Button bt=(Button)findViewById(R.id.button);
```

「注」キャスト

(Button)のように () 内にクラス名（Button や TextView など）や型（int や float など）を書いたものをキャストと呼びます。findViewById で取得するオブジェクトは Button であったり、TextView であったりするのでキャストによりそのオブジェクトのクラスまたは型を明示する必要があります。キャストがないとエラーとなります。

4 章　レイアウトと Graphical Layout

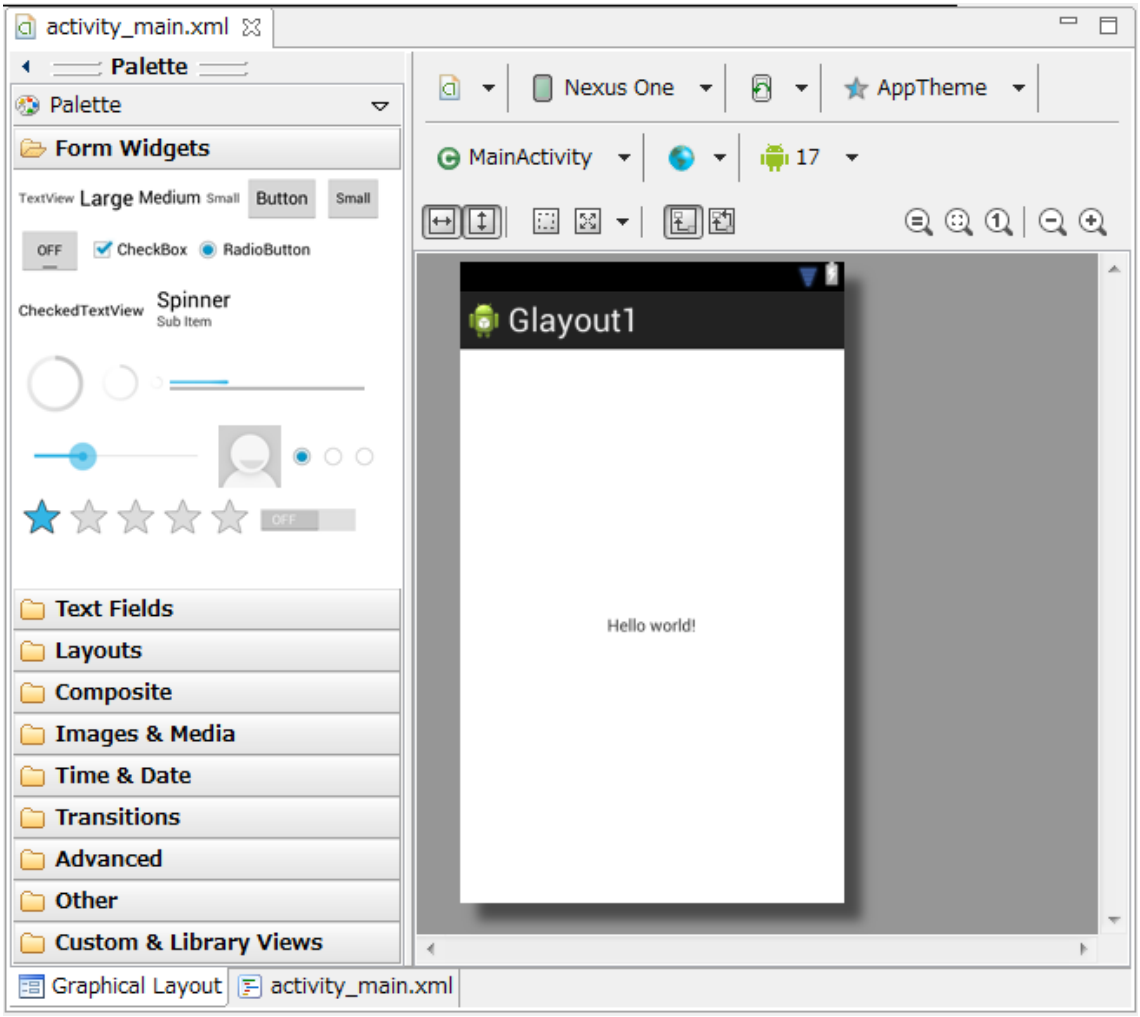
ウイジェットを配置するコンテナをレイアウトと呼びます。Android で指定できるレイアウトとしてリニアレイアウト、相対レイアウト、フレームレイアウト、テーブルレイアウトの 4 種類があります。

Android 2.x でのデフォルトのレイアウトは Linear Layout(Vertical)ですが、Android 4.x でのデフォルトのレイアウトは Relative Layout です。これは、Graphical Layout でウィジェットを配置する場合、Relative Layout の方が都合よいからです。

3 章では activity_main.xml に XML コードを直接記述してレイアウトやウィジェットを配置しました。この章では Graphical Layout を使って、パレットからマウス操作でレイアウトやウィジェットを配置する方法を説明します。

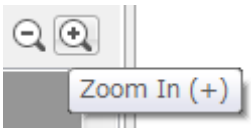
4-1　Graphical Layout の使い方

パッケージ・エクスプローラーから activity_main.xml を選択すると以下のような Graphical Layout 画面が表示されます。3 章では画面下部の「activity_main.xml」タブを選択し、XML コードを直接記述してレイアウトやウィジェットを配置しました。この章では Graphical Layout を使って、パレットからマウス操作でレイアウトやウィジェットを配置する方法を説明します。



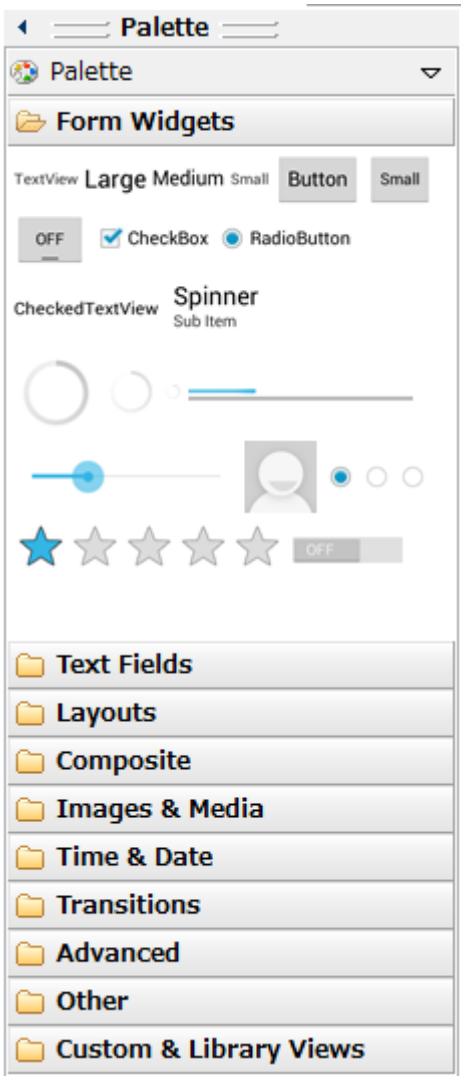
1.　レイアウト画面

「Graphical Layout」タブでレイアウト画面を表示します。デフォルトで表示されるレイアウト画面は小さいので右上にあるズームボタンでレイアウト画面を拡大します。



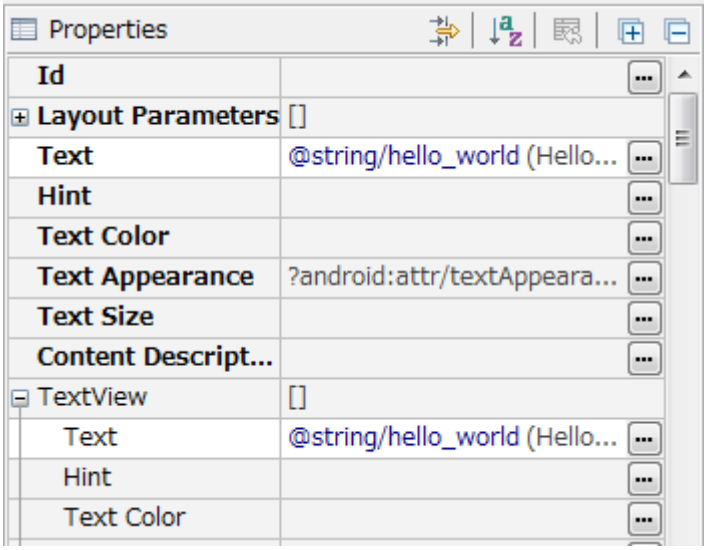
2.　パレット

　レイアウトやウィジェットの候補がパレットにカテゴリ別に登録されています。マウス操作で、パレットからレイアウト画面にレイアウトやウィジェットを配置します。レイアウト画面に配置したウィジェットは、ウィジェットを選択状態にし、「Delete」キーで削除できます。

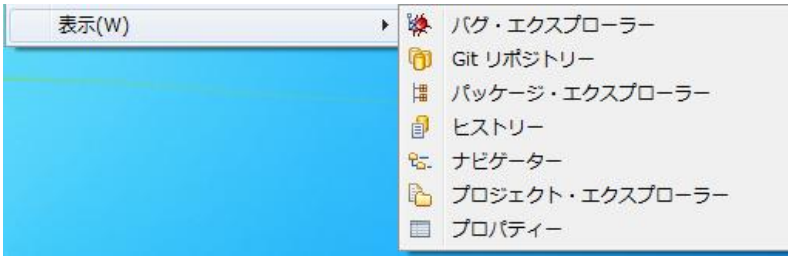


3.　プロパティー

レイアウトやウィジェットの属性（プロパティー）を設定します。



「注」プロパティー画面が表示されていない場合はウィジェットを右クリックし、「表示」ー「プロパティー」を選択します。画面下部に表示されますので、画面の右に移すにはプロパティーー画面をドラッグドロップして移動します。



5 章　activity_main.xml を使わずにレイアウトする

　レイアウトとそこに配置するウイジェットを記述する方法として主に以下のような 3 種類があります。すでに①の方法を説明してありますので、この章では②、③の方法を説明します。複数のウイジェットを配置する場合などは、activity_main.xml に記述するより、for 文などを用いて Java プログラムで記述した方が効率的な場合があります。

①activity_main.xml にレイアウトやウイジェットを記述する。

②activity_main.xml にレイアウトを記述せずに、Java プログラムコード中でレイアウトやウイジェットを作成する。

③activity_main.xml に記述したレイアウトに、Java プログラムコードでウイジェットやビューを追加する。

5-1　LinearLayout を Java コードで配置

リニアレイアウトを生成し、コンテントビューに設定するには次のようにします。

```
LinearLayout layout=new LinearLayout(this);
layout.setOrientation(LinearLayout.HORIZONTAL);
setContentView(layout);
```

ボタンを生成し、テキストを設定するには次のようにします。

```
Button bt1=new Button(this);
bt1.setText("1");
```

リニアレイアウト layout にボタン bt1 を配置するには次のようにします。

```
layout.addView(bt1,new LinearLayout.LayoutParams(WC,WC));
```

addView メソッドの第 2 引数には追加するウイジェットのサイズを指定します。サイズは LinearLayout.LayoutParams を使って指定します。wrap_content と fill_parent を示す定数を次のように定義しておくと便利です。

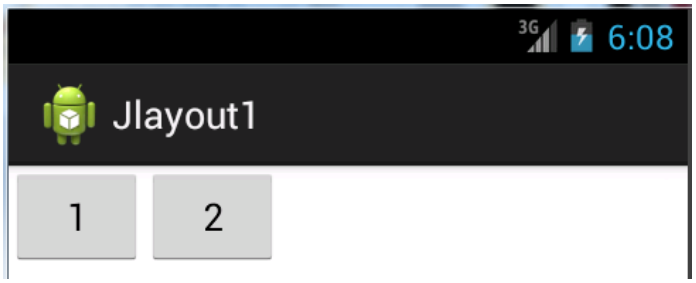
```
private final int WC=ViewGroup.LayoutParams.WRAP_CONTENT;
private final int FP=ViewGroup.LayoutParams.FILL_PARENT;
```

「例題 5-1」Button を 2 つリニアレイアウトに水平方向で配置します。

・MainActivity.java

```
package com.example.jlayout1;

import android.app.Activity;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.*;

public class MainActivity extends Activity {
    private final int WC=ViewGroup.LayoutParams.WRAP_CONTENT;
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout linearLayout=new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.HORIZONTAL);
        setContentView(linearLayout);
        Button bt1 = new Button(this);
        bt1.setText("1");
        linearLayout.addView(bt1,new LinearLayout.LayoutParams(WC,WC));
        Button bt2 = new Button(this);
        bt2.setText("2");
        linearLayout.addView(bt2,new LinearLayout.LayoutParams(WC,WC));
    }
}
```



「練習問題 5-1」Button にリスナーを付け、クリックしたボタンのテキストをタイトルバーに表示しなさい。

・MainActivity.java

```
package com.example.jlayout2;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.*;

public class MainActivity extends Activity implements OnClickListener{
```

```
    private final int WC=ViewGroup.LayoutParams.WRAP_CONTENT;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout linearLayout=new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.HORIZONTAL);
        setContentView(linearLayout);
        Button bt1 = new Button(this);
        bt1.setText("1");
        bt1.setOnClickListener(this);
        linearLayout.addView(bt1,new LinearLayout.LayoutParams(WC,WC));
        Button bt2 = new Button(this);
        bt2.setText("2");
        bt2.setOnClickListener(this);
        linearLayout.addView(bt2,new LinearLayout.LayoutParams(WC,WC));
    }
    public void onClick(View view) {
        Button bt=(Button)  ①  ;
        setTitle(   ②   );
    }
}
```

6章　タッチイベント

タッチスクリーン（ディスプレイ）は指で触れて操作します。今までボタンやリスト項目などを指でタッチする動作をクリックと呼び、OnClickListener で処理していました。タッチスクリーンに指で触れる動作を一般にタッチとかタップと呼び、主な操作方法の呼び方は以下です。

呼び方	動作
タップ	指で軽く叩く操作。マウスのクリックに相当。
ダブルタップ	2 回叩く操作。マウスのダブルクリックに相当。
ロングタッチ	一定時間（たとえば 1 秒以上）画面に触れてから離す操作。
ドラッグ（スライド）	指で押さえながら移動する操作。
スクロール	指で押さえながら上下左右に移動する操作。
フリング（フリック）	リストなどをスクロールする時に指で軽くはらう操作。
ピンチ	2 本指でのつまむ操作の総称。
ピンチアウト (ピンチオープン)	2 本指の間を広げて拡大する時の操作。
ピンチイン (ピンチクローズ)	2 本指の間を縮めて縮小する時の操作。

6-1　タッチアクションの種類

タッチイベントが発生すると onTouchEvent メソッドが呼び出されます。そのとき引数 event を使って event.getAction()で発生したイベントの種類（ACTION_DOWN、ACTION_UP、ACTION_MOVE）が取得できます。タッチ位置の座標は event.getX()と event.getY()で float 値で取得できます。

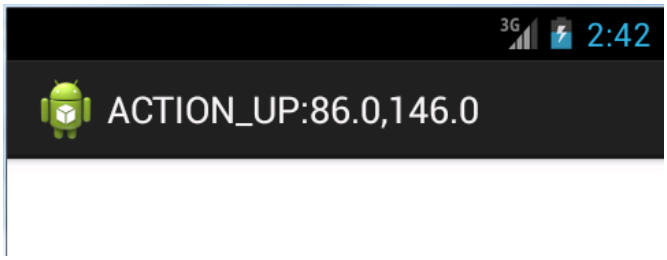
```
public boolean onTouchEvent(MotionEvent event) {
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN: break;
        case MotionEvent.ACTION_UP: break;
        case MotionEvent.ACTION_MOVE: break;
    }
    return super.onTouchEvent(event);
}
```

「例題 6-1」タッチイベントの種類とタッチ位置の x、y 座標値をタイトルバーに表示します。

・MainActivity.java

```
package com.example.touch1;

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

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        String action="";
        switch (event.getAction()) {
```

```
            case MotionEvent.ACTION_DOWN:
                action="ACTION_DOWN:";break;
            case MotionEvent.ACTION_UP:
                action="ACTION_UP:";break;
            case MotionEvent.ACTION_MOVE:
                action="ACTION_MOVE:";break;
        }
        setTitle(action+event.getX()+","+event.getY());
        return super.onTouchEvent(event);
    }
}
```



「練習問題 6-1」タッチダウンした位置からタッチムーブ位置までの距離をタイトルバーに表示しなさい。

・MainActivityTouch2.java

```
package com.example.touch2;

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

public class MainActivity extends Activity {
    float oldx,oldy;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

```
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        float distance,dx,dy;
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                oldx=event.getX();
                oldy=event.getY();
                break;
            case MotionEvent.ACTION_MOVE:
                dx=_____①_____
                dy=_____②_____
                distance=(float)Math.sqrt(dx*dx+dy*dy);
                setTitle("2点間の距離="+distance);
                break;
        }
        return super.onTouchEvent(event);
    }
}
```

7章　トースト、ダイアログ

カレントのActivityとは別画面にメッセージ表示する方法としてトーストとダイアログがあります。トーストは画面下部に表示され、ある時間が過ぎるとフェードアウトしていく別ウインドウです。Androidで使用できる標準ダイアログとしてAlertDialog、ProgressDialog、DatePickerDialog、TimePickerDialogがあります。ユーザが独自のカスタム・トーストやカスタム・ダイアログを作ることもできます。

7-1　トースト

トーストは画面下部に表示され、ある時間が過ぎるとフェードアウトしていく別ウインドウです。トーストは以下のようにして表示します。表示されている時間は「Toast.LENGTH_LONG」と「Toast.LENGTH_SHORT」が指定できます。

```
Toast.makeText(this,"メッセージ",Toast.LENGTH_LONG).show();
```

トーストは様々な局面で、ちょっとした情報を表示するのに使われます。標準のトーストは、画面の下部の中央に表示されます。この位置を以下のように変更することができます。

```
setGravity(Gravity 定数, x 位置, y 位置);
```

Gravity 定数には TOP、BOTTOM、CENTER、LEFT、RIGHT などを指定でき、これらを「|」演算子で組み合わせて指定することもできます。「Gravity.TOP | Gravity.LEFT」とすると画面左上隅になります。x,y 位置は Gravity 定数で指定した位置からの相対オフセット値になります。makeText や show メソッドを別々に書くと以下のようになります。

```
Toast toast=Toast.makeText(this,(++count)+"回目",Toast.LENGTH_LONG);
toast.setGravity(Gravity.TOP | Gravity.LEFT,0,0);
toast.show();
```

「例題 7-1」画面のタッチでトーストを表示します。その際、タッチした回数を変数 count にカウントして、その値をもとに「5 回目」のように表示します。

・MainActivity.java

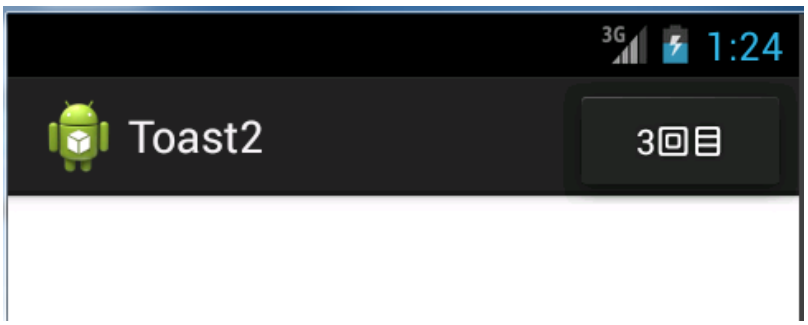
```
package com.example.toast1;

import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.widget.Toast;

public class MainActivity extends Activity {
    private int count=0;
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            Toast.makeText(this,(++count)+"回目",Toast.LENGTH_LONG).show();
        }
        return super.onTouchEvent(event);
    }
}
```

「練習問題 7-1」画面のタッチで画面の右上にトーストを表示しなさい。

・MainActivity.java

```
package com.example.toast2;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.MotionEvent;
import android.widget.Toast;

public class MainActivity extends Activity {
    private int count=0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            Toast toast=Toast.makeText(this,(++count)+"回目
",Toast.LENGTH_LONG);
            toast.setGravity(_____①_____);
            toast.show();
        }
        return super.onTouchEvent(event);
    }
}
```

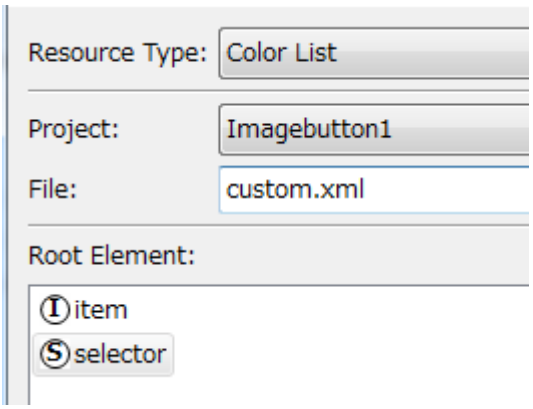


8章　各種ウィジェット

「3章　ウイジェットとXML」でButton、TextView、ImageView、EditText、CheckBox、RadioButton、Spinner、ListViewなどの基本ウィジェットについて説明しました。Androidにはこの他にも多くのウィジェットが用意されています。この章では比較的有用で使用頻度が高いImageButton、ToggleButton、CheckedTextView、AnalogClock、DigitalClock、ProgressBar、SeekBar、ScrollView、Gallery、GridViewについて説明します。

8-1　イメージボタン(ImageButton)

ImageButton はテキストの代わりにイメージを表示したボタンです。デフォルトではイメージボタンをクリックしてもイメージボタンの形状は変わりません。<ImageButton>の属性に「style="@style/btn"」を指定します。"@style/btn"の定義は res/values/style.xml ファイルで行い、さらに具体的なボタンを押したときのイメージ（bt1_pressed.png）と離したときのイメージ(bt1_normal.png)の指定は res/color/custom.xml で行います。color フォルダを作り、custom.xml の Resource Type は「Color List」、Root Element は「selector」とします。



ボタンを強制的に押されたイメージにするには以下のようにします。

```
ibt=(ImageButton)findViewById(R.id.ibutton);
ibt.setPressed(true);
```

これで custom.xml の「android:state_pressed="true"」が指定されている<item>に記述されているイメージがイメージボタンに表示されます。ImageButton のクリック処理の方法は Button と同じです。

「例題 8-1」「1」を示すイメージボタンの押/離イメージを表示します。

・activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >
```

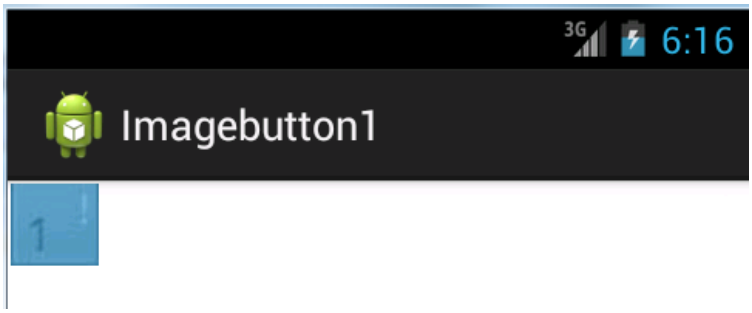
```
    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="2dp"
        style="@style/btn"
    />
</LinearLayout>
```

・res/color/custom.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:state_pressed="true"
        android:drawable="@drawable/bt1_pressed"
    />
    <item
        android:drawable="@drawable/bt1_normal"
    />
</selector>
```

・res/values/style.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="btn">
        <item name="android:background">@color/custom</item>
    </style>
</resources>
```



「練習問題 8-1」「1」と「2」を示すイメージボタンの押/離イメージを表示しなさい。

・activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >
    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="2dp"
        style="@style/btn1"
    />
    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="2dp"
        style="@style/btn2"
    />
</LinearLayout>
```

・res/values/style.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="btn1">
        <item name="android:background">@color/custom1</item>
    </style>
    <style name="btn2">
        <item name="android:background">　　①　　</item>
    </style>
</resources>
```

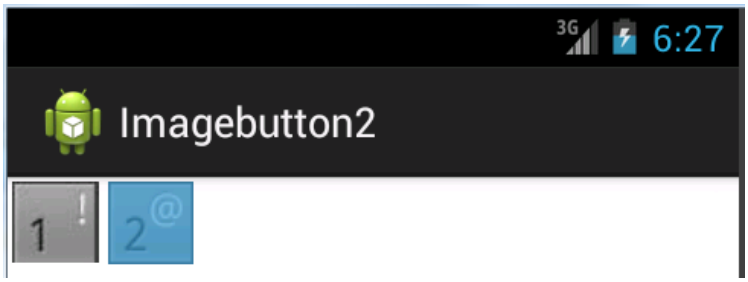
・res/color/custom1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
```

```
        android:state_pressed="true"
        android:drawable="@drawable/bt1_pressed"
    />
    <item
        android:drawable="@drawable/bt1_normal"
    />
</selector>
```

・res/color/custom2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:state_pressed="true"
        android:drawable=_____②_____
    />
    <item
        android:drawable="@drawable/bt2_normal"
    />
</selector>
```



9章　インテントとアクティビティ

Andoroid アプリケーションを構成するコンポーネントとして、アクティビティ、サービス、ブロードキャストレシーバーがあります。これら 3 種類のコンポーネントを起動するのにインテントを使います。この章ではインテントを使って別のアクティビティを起動する方法とアクティビティ間でデータ授受する方法を説明します。さらにインテントを使ってダイアラーやメーラーを起動する方法を説明します。またステータスバーへの通知（Notification）方法についても説明します。

9-1　インテントとは

Android では Intent クラスを使って別の Activity に状態を移すことができます。インテントには動作とデータを与えます。動作（Action）にユーザ定義クラスを指定する場合を「明示的インテント」と呼びます。Android が定める Action の種類を指定する場合を「暗黙的インテント」と呼び、以下のような Action があります。

Action	機能
ACTION_VIEW	別画面（別アクティビティ）を表示します。指定する URI が「http:」なら Web ページ、「geo:」なら GoogleMap などになります。最も一般的なアクションです。インテントを使って GoogleMap の表示する方法は 15 章の 15-2 で説明します。
ACTION_SEND	データを送ります。メールの送信などに使います。
ACTION_DIAL	ダイアラーを表示します。
ACTION_SET_WALLPAPER	壁紙の設定をします。
ACTION_PICK	データから項目を選択します。ギャラリーなどで使用します。

「英単」intent：意図，意向,目的、しっかりと向けられている

たとえば指定した URI の Web ページを表示するには次のようにします。

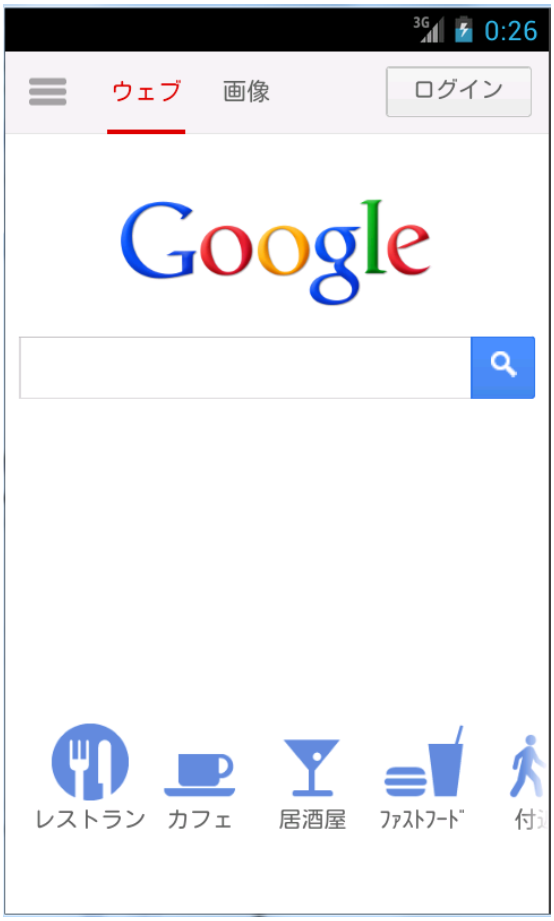
```
Uri uri = Uri.parse("http://www.google.co.jp/");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

インテントのアクションは「ACTION_VIEW」、データは「Uri.parse("http://www.google.co.jp/")」となります。このインテントを開始するには startActivity メソッドを使います。

「例題 9-1」画面のタッチで指定した URI の Web ページを表示します。

・MainActivity.java

```
package com.example.intent1;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.MotionEvent;

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            Uri uri = Uri.parse("http://www.google.co.jp/");
            Intent it = new Intent(Intent.ACTION_VIEW,uri);
            startActivity(it);
        }
        return super.onTouchEvent(event);
    }
}
```



「練習問題 9-1」ListView に登録してある Web ページを表示しなさい。

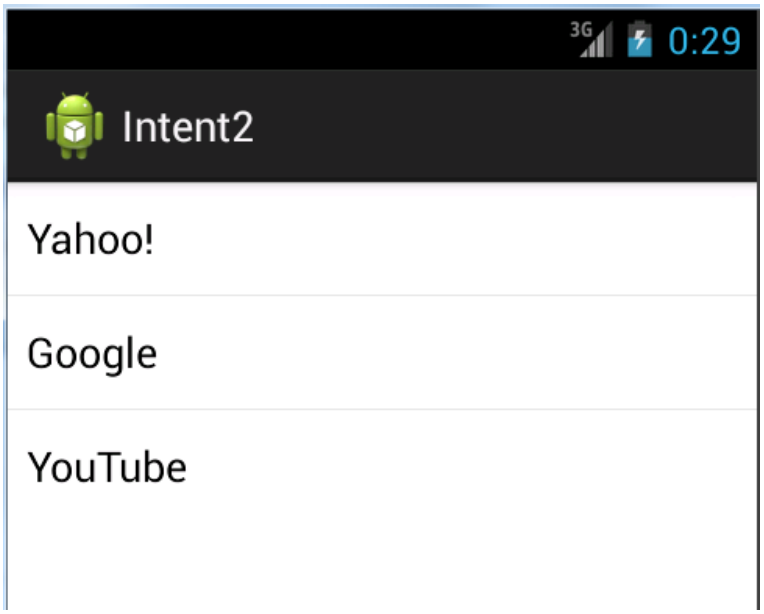
・activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <ListView
        android:id="@+id/listview"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

・MainActivity.java

```
package com.example.intent2;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
import android.widget.AdapterView.OnItemClickListener;

public class MainActivity extends Activity {
    private String name[]={"Yahoo!","Google","YouTube"};
    private String
uriStr[]={"http://www.yahoo.co.jp/","http://www.google.co.jp/","http://www.yo
utube.com/"};
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1);
        ListView lv=(ListView) findViewById(R.id.listview);
        for (int i=0;i<name.length;i++)
            adapter.add(__①__);
        lv.setAdapter(adapter);
        lv.setOnItemClickListener(new ItemClick());
    }
    class ItemClick implements OnItemClickListener {
        public void onItemClick(AdapterView<?> parent, View view,int position,
long id) {
            Uri uri = Uri.parse(____②____);
            Intent it = new Intent(Intent.ACTION_VIEW,uri);
            startActivity(it);
        }
    }
```

```
}
```


0:29
Intent2
Yahoo!
Google
YouTube

10 章　Handler とサービス

Java ではプログラムの一連の実行をスレッド（Thread）という小さい処理単位に分割し、必要に応じて実行することで複数のプログラムを同時に並列処理しているように見せることができます。いままでのプログラムでは、ボタンのクリックや画面のタッチなどのイベントの発生を受けて処理（イベントドリブン・プログラム）を行っていましたが、そのようなイベントと関係なく別の仕事を行いたい場合にスレッドを用います。

Android でもスレッドを使用することができます。ただし、Android の GUI はシングルスレッドにしか対応していないため、画面の描画処理を別スレッドで実行することを禁止しています。このような場合はハンドラ（Handler）を介することで裏ルートでのマルチスレッドを実現します。Handler を使って一定時間ごとの処理を行うことができます。

アクティビティは画面に表示されている表に現れる処理です。これに対してサービスやブロードキャストレシーバはアクティビティのバックグラウンドで動作する表に現れない処理です。サービスは長時間かかる処理をアクティビティと独立して行う仕組みです。また、サービスを呼び出したアクティビティが中断されてもサービス処理を継続することができます。

10-1　Thread と Handler

Java では一般にスレッドを用いた処理は Thread クラスと Runnable インターフェースを用いて行います。スレッドは start メソッドで開始します。スレッドが実行されたときに行う処理は run メソッド内に記述します。run メソッドは Runnable インターフェースのメソッドです。Thread を生成し、開始するには以下のようにします。start メソッドが実行されると、run メソッドが呼び出されます。

```
new Thread(new Runnable() {
    public void run() {
        //スレッド内で行う処理内容
    }
}).start();
```

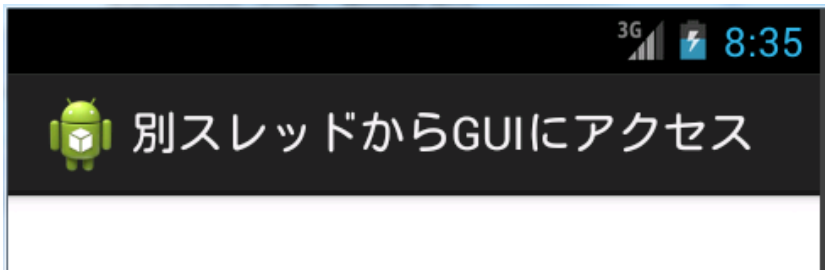
ところが Android の GUI はシングルスレッドにしか対応していないため、画面の描画処理を別スレッドで実行することを禁止しています。つまりアクティビティと異なるスレッドからアクティビティ画面を描画しようとすると実行時エラーとなります。これを回避するにはアクティビティ内でハンドラ hd を生成しておき、この hd に対し post メソッドを使って Runable オブジェクトを設定します。これによりこの Runable スレッドは、アクティビティと同じスレッドで実行されることになりエラーを回避できます。

```
private Handler hd;
hd=new Handler();

new Thread(new Runnable() {
    public void run() {
        hd.post(new Runnable() {
            public void run() {
                //アクティビティへの描画処理
            }
        });
    }
}).start();
```

「例題 10-1」画面のタッチで、別スレッドを開始し、その中でタイトルバーへの表示を行います。

・MainActivity.java

```
package com.example.handler1;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.view.MotionEvent;

public class MainActivity extends Activity {
    private Handler hd;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        hd=new Handler();
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            new Thread(new Runnable() {
                public void run() {
                    hd.post(new Runnable() {
                        public void run() {
                            setTitle("別スレッドからGUIにアクセス");
                        }
                    });
                }
            }).start();
        }
        return super.onTouchEvent(event);
    }
}
```



「練習問題 10-1」画面のタッチで、２つのスレッドを開始し、１つ目のスレッドで「A」の文字、2 つ目のスレッドで「B」の文字を TextView に表示しなさい。

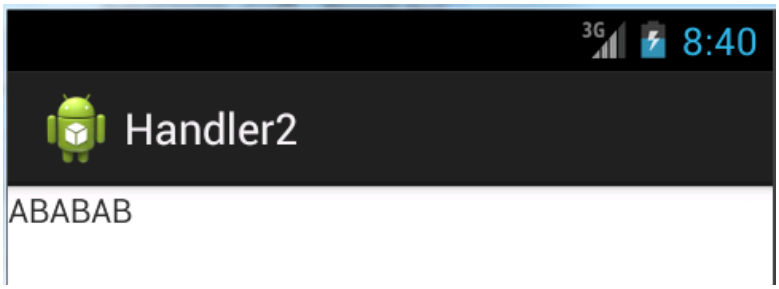
・activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/text1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

・MainActivity.java

```
package com.example.handler2;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.view.MotionEvent;
import android.widget.TextView;

public class MainActivity extends Activity {
    private Handler hd;
    TextView text1;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        hd=new Handler();
        text1=(TextView)findViewById(R.id.text1);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            new Thread(new Runnable() {
                public void run() {
                    hd.post(new Runnable() {
                        public void run() {
                            text1.setText(_____①_____);
                        }
                    });
                }
            }).start();
            new Thread(new Runnable() {
                public void run() {
                    hd.post(new Runnable() {
                        public void run() {
                            text1.setText(_____②_____);
                        }
                    });
                }
            }).start();
        }
        return super.onTouchEvent(event);
    }
}
```



11章　サウンドとムービー

サウンドとムービーを扱うウイジェットとしてMediaPlayerとVideoViewがあります。MediaPlayerはサウンドの演奏を行い、VideoViewは動画の再生を行います。これらは「android.widget」パッケージで定義されています。ウイジェットの分類ではありませんが、この他にマルチメデイアを扱うクラスとして「android.media」パッケージのSoundPool、ToneGenerator、MediaRecorder、「android.webkit」パッケージのWebViewがあります。この章ではこれらのクラスの使用方法を説明します。

11-1　サウンドの再生

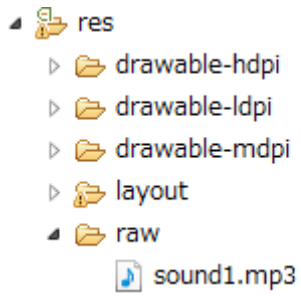
1.　音楽ファイル形式の種類

Android の仕様では MP3/AAC/3GP/MID/WMA/IMY/OTA/OGG などに対応することになっていますが、実機に搭載されたメディアプレーヤーにより再生できるフォーマットは異なります。代表的なフォーマットとして以下があります。

音楽ファイル形式	特徴
MP3 (MPEG Audio Layer-3)	映像データ圧縮方式の MPEG-1 で利用される音声圧縮方式。
WMA (Windows Media Audio)	Microsoft 社が開発した Windows 標準の音声圧縮方式。
3GP (Third Generation Partnership)	3GP プロジェクト(3GPP)が定めた標準規格のファイルフォーマット。
OGG	Xiph.org Foundation が開発した、ライセンスフリーな音声圧縮方式。

2.　MediaPlayer

サウンドファイル（~.mp3）をリソースフォルダの res/raw に配置しておきます。



サウンドの再生は MediaPlayer クラスを使用して次のように行います。create メソッドの第 1 引数には「this」または「getApplicationContext()」を指定します。seekTo(0)で先頭位置に設定します。

```
MediaPlayer mp;
mp=MediaPlayer.create(this,R.raw.sound1);
mp.seekTo(0);
mp.start();
```

再生を停止するには次のようにします。

```
mp.stop();
mp.release();
mp=null;
```

「例題 11-1」「演奏開始」ボタンで sound1.mp3 ファイルを演奏し、「演奏停止」ボタンで停止します。

・activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="演奏開始"
    />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="演奏停止"
    />
</LinearLayout>
```

・MainActivity.java

```
package com.example.sound1;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
```

```
import android.widget.*;

public class MainActivity extends Activity implements OnClickListener{
    private MediaPlayer mp=null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button bt1=(Button)findViewById(R.id.button1);
        bt1.setOnClickListener(this);
        Button bt2=(Button)findViewById(R.id.button2);
        bt2.setOnClickListener(this);
    }
    public void onClick(View view) {
        switch (view.getId()){
            case R.id.button1:mp=MediaPlayer.create(this,R.raw.sound1);
                              mp.seekTo(0);
                              mp.start();
                              break;
            case R.id.button2:if (mp!=null){
                                  mp.stop();
                                  mp.release();
                                  mp=null;
                              }
                              break;
        }
    }
}
```

「練習問題 11-1」リストビューに登録された音楽ファイルを選択して演奏します。
res/raw フォルダにサウンドファイルとして sound1.mp3～sound3.mp3 を配置しておきます。

・activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/listview"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

・MainActivity.java

```
package com.example.sound2;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
import android.widget.AdapterView.OnItemClickListener;
```

```
public class MainActivity extends Activity {
    private MediaPlayer mp=null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1);
        adapter.add("序曲");
        adapter.add("勇者の行進");
        adapter.add("明日への道");
        adapter.add("演奏停止");
        ListView lv=(ListView)findViewById(R.id.listview);
        lv.setAdapter(adapter);
        lv.setOnItemClickListener(new ItemClick());
    }
    class ItemClick implements OnItemClickListener {
        public void onItemClick(AdapterView<?> parent, View view,int position,
long id) {
            int mid[]={R.raw.sound1,R.raw.sound2,R.raw.sound3};
            if (mp!=null){
                mp.stop();
                mp.release();
                mp=null;
            }
            if (  ①  ){
                mp=MediaPlayer.create(getApplicationContext(),  ②  );
                mp.seekTo(0);
                mp.start();
            }
        }
    }
}
```

「注」MediaPlayer.*create*(getApplicationContext(),mid[position]);の第 1 引数に this を使う場合は ItemClick メソッドを別クラスで構成しているため「MainActivity.this」とします。

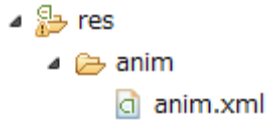

1:57
Sound2
序曲
勇者の行進
明日への道
演奏停止

12 章　アニメーション

アニメーションのタイプは大きく 2 つに分類できます。Tween アニメーション(Animation)と Frame アニメーション(AnimationDrawable)です。Tween アニメーションは 1 つのイメージを連続に変化させるタイプで、移動、フェード、回転、拡大・縮小、それらを組み合わせて変化させます。Frame アニメーションは順番にイメージを並べて表示してアニメーションにするタイプでパラパラ漫画のような表示を行います

12-1　Tween アニメーション

XML で定義されたフェード、伸縮、移動、回転などといった変化を実行するアニメーションです。Tween アニメーションを行うにはアニメーションの内容を res/anim フォルダに anim.xml として定義します。



アニメーション定義は<set>要素内に<alpha>、<scale>、<translate>、<rotate>などのアニメーション要素を記述します。<set>のアニメーション要素は複数指定できます。

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="4000"
    />
</set>
```

このアニメーション定義ファイルをイメージビューiv に設定するには以下のようにします。Tween アニメーションは「android.view.animation.Animation」クラスを使って制御します。

```
    Animation anim=AnimationUtils.loadAnimation(this, R.anim.anime);
    iv.startAnimation(anim);
```

1. <set>

アニメーション要素(<alpha>、<scale>、<translate>、<rotate>）や他の <set> 要素を保持するコンテナです。AnimationSet クラスを表します。

属性	意味
interpolator	Interpolator リソースへのリファレンス。
shareInterpolator	子要素に同じ Interpolator を共有させる（true)、させない（false)。

2. <alpha>

アニメーションのフェードイン、フェードアウトします。AlphaAnimation クラスを表します。以下の属性で不透明度を 0.0～1.0 の float 値で指定します。0.0 は透明、1.0 は不透明です。

属性	意味
fromAlpha	開始の不透明度。
toAlpha	終了の不透明度。
duration	アニメーション時間（ミリ秒)。

以下は透明度を 0.0～1.0 まで 4000 ミリ秒の間で変化させます。

```
<alpha
    android:fromAlpha="0.0"
    android:toAlpha="1.0"
    android:duration="4000"
/>
```

3. <scale>

アニメーションの拡大・縮小を行います。ScaleAnimation クラスを表します。以下の属性で拡大・縮小の倍率と中心位置を float 値で指定します。倍率の「1.0」は変化しない（等倍）を意味します。pivotX と pivotY が（0,0）の場合は左上隅を中心に拡大・縮小はすべて下と右に伸縮します。

属性	意味
fromXScale	開始の X 倍率。
toXScale	終了の X 倍率。

fromYScale	開始の Y 倍率。
toYScale	終了の Y 倍率。
pivotX	拡大・縮小の中心 X 座標。
pivotY	拡大・縮小の中心 Y 座標。
duration	アニメーション時間（ミリ秒)。

以下は縮尺を 1.0～0.0 まで中心を軸にして 4000 ミリ秒の間で縮小させます。

```
<scale
    android:fromXScale="1.0"
    android:toXScale="0.0"
    android:fromYScale="1.0"
    android:toYScale="0.0"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="4000"
/>
```

4. <translate>

アニメーションの X 方向、Y 方向の水平移動を行います。TranslateAnimation クラス を表します。以下の属性で移動量を float 値またはパーセンテージで指定します。通常の位置に対する相対ピクセル("5"など)、要素の幅に対する相対パーセンテージ("5%"など)、親の幅に対する相対パーセンテージ("5%p"など)のいずれか一方で表現します。

属性	意味
fromXDelta	開始の X オフセット値。
toXDelta	終了の X オフセット値。
fromYDelta	開始の Y オフセット値。
toYDelta	終了の Y オフセット値。
duration	アニメーション時間（ミリ秒)。

以下は対象となるイメージの左端から右端まで 4000 ミリ秒の間で平行移動します。

```
<translate
    android:fromXDelta="0%"
    android:toXDelta="100%"
    android:duration="4000"
/>
```

5. <rotate>

アニメーションの回転を行います。RotateAnimation クラス を表します。 以下の属性で回転角度と回転中心座標を float 値で指定します。pivotX、pivotY はパーセンテージ指定もできます。

属性	意味
fromDegrees	開始位置の角度（度数）。
toDegrees	終了位置の角度（度数）。
pivotX	回転の中心となる X 座標。
pivotY	回転の中心となる Y 座標。
duration	アニメーション時間（ミリ秒）。

以下はイメージの中央を原点として 0°～360°まで 4000 ミリ秒の間で回転します。

```
<rotate
    android:fromDegrees="0"
    android:toDegrees="360"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="4000"
/>
```

「例題 12-1」イメージを回転しながら縮小します。イメージは画面中央に配置します。

・res/anim/anim.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false"
    >
    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="4000"
    />
    <scale
```

```
        android:fromXScale="1.0"
        android:toXScale="0.0"
        android:fromYScale="1.0"
        android:toYScale="0.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="4000"
    />
</set>
```

・activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center"
    >
    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/sakura"
    />
</LinearLayout>
```

・Anim1.java

```
package com.example.anim1;

import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.animation.*;
import android.widget.*;

public class MainActivity extends Activity {
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            ImageView iv=(ImageView) findViewById(R.id.image);
            Animation anim=AnimationUtils.loadAnimation(this, R.anim.anim);
            iv.startAnimation(anim);
        }
        return super.onTouchEvent(event);
    }
}
```

「練習問題 12-1」イメージの幅の 2 倍の距離を 0°～360°回転しながら水平に移動しなさい。

・res/anim/anim.xml

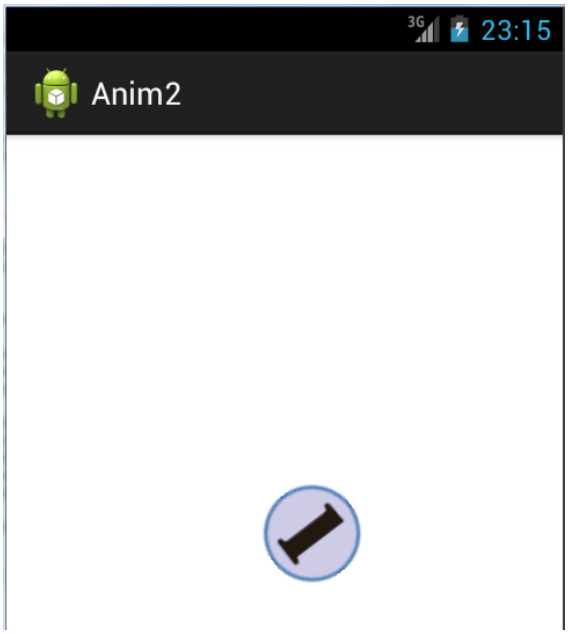
```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false"
    >
    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="4000"
    />
    <translate
        android:fromXDelta=  ①
        android:toXDelta=  ②
        android:duration="4000"
    />
</set>
```

・activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center"
    >
    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/num1"
    />
</LinearLayout>
```

・Anim2.java

例題 12-1 と同じ



13 章　グラフィックス

2 章でグラフィックスを使った例題を説明しましたが、ここでは、さらに詳しくグラフィックスに関して説明します。描画を行うには 4 つの要素を使います。描画を行う Canvas、描画データを保持する Bitmap、描画の色やスタイルを決める Paint、Rect や Path などの描画プリミティブです。図形をオブジェクトとして扱うための抽象クラスの Drawable を使用することもできます。

グラフィックスの主な操作は Canvas クラスの drawLine メソッドや drawText メソッドを使って、直線や円などの図形やテキストを描画することです。この操作を補助する機能として座標変換やクリップ領域の設定があります。

キャンバスに描画する代わりに Path に対して仮想的に描画を行い、その情報を保持しておくことができます。必要なときにその Path データを用いてキャンバスに描画することができます。

13-1　基本図形の描画

Canvas クラスの中で、直線、点、矩形、円、楕円、円弧などの基本図形を描く描画メソッドを説明します。

1.　座標を示すクラス

点を示すクラスとして Point(int x, int y)と PointF(float x, float y)があります。前者は int 型、後者は float 型です。座標(x,y)を示す点は以下のように生成します。

```
PointF p=new PointF(x,y);
```

矩形領域を示すクラスとして Rect(int left, int top, int right, int bottom)と RectF(float left, float top, float right, float bottom) があります。前者は int 型、後者は float 型です。(x1,y1)を左上隅座標、(x2,y2)を右下隅座標とする矩形領域は以下のように生成します。

```
RectF r=new RectF(x1,y1,x2,y2);
```

2.　色

色の設定は setColor メソッドで「paint.setColor(Color.WHITE);」のように行います。色は「Color.WHITE」のように色を示す定数を指定する他に rgb メソッドを使って Color.rgb(red,green,blue)のように指定することもできます。red,green,blue は赤、緑、青の成分を 0～255 の範囲で指定します。値が大きいほどその色の成分がでます。Color.argb(alpha,red,green,blue)は透過度 alpha を 0（完全な透明）～255（完全な不透明）で設定できます。

```
paint.setColor(Color.argb(128,0,0,255));
```

3.　直線の描画

直線の描画は drawLine メソッドまたは drawLines メソッドで行います。

```
canvas.drawLine(x1,y1,x2,y2,paint);
```
・・・(x1,y1)－(x2,y2)に直線を描きます。

```
float[] p={x1,y1,x2,y2,x3,y3,x4,y4・・・};
canvas.drawLines(p,paint);
```
・・・配列 p[ ]の 4 要素ごとに始点、終点のペアとみなして、それぞれの直線を描きま

す。つまり(x1,y1)－(x2,y2)の直線、(x3,y3)－(x4,y4)の直線・・・と描きます。連続した直線を描くには{x1,y1,x2,y2,x2,y2,x3,y3・・・}のように前の直線の終点を次の直線の始点とするようなデータにします。

4. 点の描画

点の描画は drawPoint メソッドまたは drawPoints メソッドで行います。

```
canvas.drawPoint(x1,y1,paint);
```

・・・(x1,y1)位置に点を描画します。

```
float[] p={x1,y1,x2,y2・・・};
canvas.drawPoints(p,paint);
```

・・・配列 p[ ]の 2 要素ごとに(x,y)とした点を描画します。

5. 矩形の描画

矩形の描画は drawRect メソッドまたは drawRoundRect メソッドで行います。矩形、楕円、円、円弧などを描画する際にデフォルトの描画スタイルは内部を塗りつぶすので、線だけで描画するには「paint.setStyle(Style.STROKE);」とします。

```
RectF r=new RectF(x1,y1,x2,y2);
canvas.drawRect(r, paint);
```

・・・左上隅を(x1,y1)、右下隅を(x2,y2)とする矩形を描きます。矩形、楕円、円弧を描くメソッドは矩形領域を示す RectF を引数にしますが、drawRect は drawRect(x1,y1,x2,y2, paint)のように直接各点を引数にすることもできます。

```
RectF r=new RectF(x1,y1,x2,y2);
canvas.drawRoundRect(r,rx,ry,paint);
```

・・・左上隅を(x1,y1)、右下隅を(x2,y2)とする矩形を描き四隅を x 方向の半径 rx,y 方向の半径 ry の丸みを付けます。

6. 円、楕円、円弧

円は drawCircle メソッド、楕円は drawOval メソッド、円弧は drawArc メソッドで行います。

```
canvas.drawCircle(x,y,r,paint);
```

・・・中心(x,y)、半径 r の円を描きます。

RectF r=new RectF(x1,y1,x2,y2);
canvas.drawOval(r,paint);
・・・(x1,y1)－(x2,y2)の矩形に内接する楕円を描きます。

RectF r=new RectF(x1,y1,x2,y2);
canvas.drawArc(r,a1,a2,true,paint);
・・・(x1,y1)－(x2,y2)の矩形に内接する楕円の a1°～(a1+a2)°までの扇形を描きます。true の代わりに false を指定すると円弧になります。角度は時計回りの方向を正とします。

7. キャンバス全体を塗る

キャンバス全体を指定色で塗るには drawColor メソッドを使います。キャンバスを白にするには「canvas.drawColor(Color.WHITE);」とします。画面クリア動作にも使用できます。

8. 画面の幅と高さ

画面の幅と高さは View クラスの getWidth メソッド、getHeight メソッドで取得できます。これらのメソッドは Gview コンストラクタ内では幅と高さが確定していないので値を取得できません。onDraw メソッド内で取得します。

int w=getWidth();
int h=getHeight();

Canvas クラスの getWidth メソッド、getHeight メソッドを使って次のように取得することもできます。ただしこの場合、幅は View クラスの getWidth メソッドと同じですが、高さはステータスバーとタイトルバーの高さも含んでいます。

int w=canvas.getWidth();
int h=canvas.getHeight();

画面サイズを別の方法で求めるには 11 章－「11-8　ブラウザ（WebView）」の「「注」Display クラスの `getWidth()`と `getHeight()`」を参照。

9. Paint クラス

基本図形を描く際の表示スタイル（外枠だけか、塗りつぶすか）は setStyle メソッド、点線のスタイルは setPathEffect メソッドで指定します。Paint クラスの主なメソッドとして以下があります。

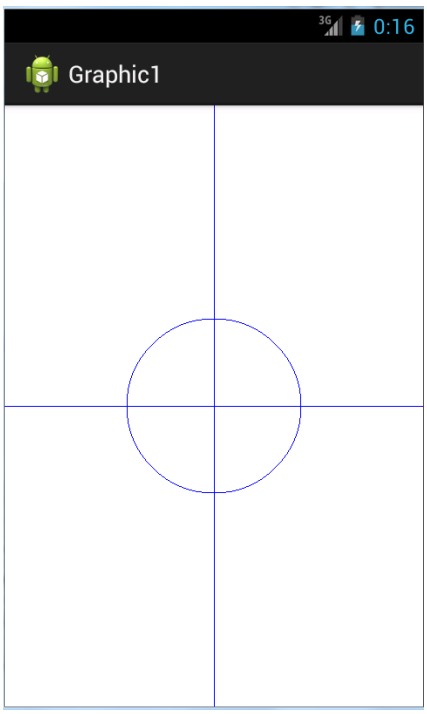
Paint クラスのメソッド	意味
setColor(int color)	描画色を Color 値で指定。
setARGB(int a, int r, int g, int b)	描画色を RGB とアルファ値で指定。
setAlpha(int alpha)	描画色にアルファ値のみを単独で指定。
setAntiAlias(boolean alias)	アンチエイリアスを設定するかどうかを指定。true を指定すると円などが滑らかに表示されます。
setStrokeWidth(float width)	線の太さを指定。
setPathEffect(PathEffect effect)	点線の形状を指定。
setTextSize(float textSize)	描画する文字の大きさを指定。
setStyle(Paint.Style style)	塗りつぶしか輪郭線の描画かを指定。Style として以下が指定できます。 FILL：指定領域内を塗りつぶす。デフォルト。 STROKE：指定領域の外枠だけ描画。 FILL_AND_STROKE：FILL と STROKE 動作。
reset()	デフォルトの設定に戻します。

「例題 13-1」画面中央に十字線と円を描きます。

・MainActivity.java

```
package com.example.graphic1;

import android.app.Activity;
import android.os.Bundle;
import android.graphics.*;
import android.graphics.Paint.*;
import android.view.View;
import android.content.Context;

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(new GView(this));
    }
    private class GView extends View {
        private Paint paint;
        public GView(Context context) {
            super(context);
            paint=new Paint();
        }
        protected void onDraw(Canvas canvas) {
            paint.setColor(Color.BLUE);
            paint.setStyle(Style.STROKE);
            int w=getWidth();
            int h=getHeight();
            canvas.drawLine(0,h/2,w,h/2,paint);
            canvas.drawLine(w/2,0,w/2,h,paint);
            canvas.drawCircle(w/2,h/2,100,paint);
        }
    }
}
```



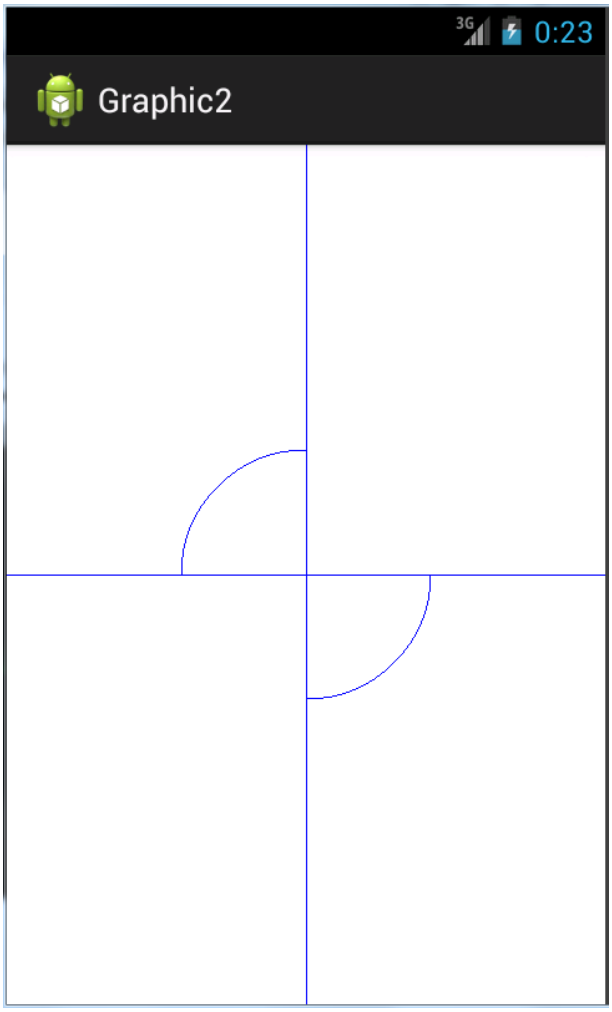
「注」この章の以後の「例題、練習問題」は GView クラス内のコードのみ記述してあります。

「練習問題 13-1」第 2 象限と第 4 象限に円弧を描きなさい。

・MainActivity.java

```
private class GView extends View {
    private Paint paint;
    public GView(Context context) {
        super(context);
        paint=new Paint();
    }
    protected void onDraw(Canvas canvas) {
        paint.setColor(Color.BLUE);
        paint.setStyle(Style.STROKE);
        int w=getWidth();
        int h=getHeight();
        canvas.drawLine(0,h/2,w,h/2,paint);
        canvas.drawLine(w/2,0,w/2,h,paint);
        RectF rect=new RectF(w/2-100,h/2-100,w/2+100,h/2+100);
        canvas.drawArc(rect,  ①  ,false,paint);
        canvas.drawArc(rect,  ②  ,false,paint);
    }
}
```

「注」`RectF rect=new RectF(w/2-100,h/2-100,w/2+100,h/2+100);`は警告エラーとなります。「練習問題 6-3」参照。


0:23
Graphic2

14 章　ファイル処理

Android はファイルを作成することができる権限を持つ特別なフォルダに対しファイルの読み書きが行えます。このフォルダにユーザは直接ファイルを置くことはできませんが、プロジェクトの assets フォルダにファイルを配置しておき、そのファイルを読みだすことができます。また、SD カードへのファイルの読み書きを行うこともできます。ファイルへの読み書きにはバイナリー・ストリームとテキスト・ストリームという手法があります。ファイルに対する読み書きの方法や、ファイル名リストの取得方法やフォルダの操作方法を説明します。

14-1　バイナリー・ストリーム

バイナリー・ストリームは FileInputStream/FileOutputStream クラスの read/write メソッドを使ってバイト単位の読み書きを行います。

ファイルへの書き込みは FileOutputStream クラスを使用し openFileOutput メソッドでファイルを開きます。ファイルへの書き込みは write メソッドで行います。引数は文字列を getBytes でバイト配列に変換したものを指定します。close メソッドでファイルを閉じて完了です。

```
FileOutputStream out=openFileOutput("test.dat",MODE_PRIVATE);
String msg="書き出すテキスト";
out.write(msg.getBytes());
out.close();
```

openFileOutput で指定できるファイルモードには以下が指定できます。

ファイルモード	機能
MODE_APPEND	既にファイルがあった場合、追加で開く。
MODE_PRIVATE	他のアプリからアクセスできない private file として生成。
MODE_WORLD_READABLE	他のアプリへ読み込み権限を与える。
MODE_WORLD_WRITEABLE	他のアプリへ書き込み権限を与える。

ファイルからの読み込みは FileInputStream クラスを使用し openFileInput メソッドでファイルを開きます。ファイルからの読み込みは read メソッドで行います。引数はバイト配列を指定します。close メソッドでファイルを閉じて完了です。

```
FileInputStream in=openFileInput("test.dat");
byte[] dat = new byte[in.available()];
in.read(dat);
in.close();
```

「例題 14-1」「書き込み」ボタンで EditText に入力したテキストを test.dat ファイルに書き込み、「読み出し」ボタンで test.dat ファイルからデータを読み出し TextView に表示します。

・activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <EditText
        android:id="@+id/editText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
    <Button
        android:id="@+id/write"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="書き込み"
    />
    <Button
        android:id="@+id/read"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="読み出し"
    />
    <TextView
        android:id="@+id/textView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

・MainActivity.java

```
package com.example.file1;

import java.io.*;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button bt1=(Button)findViewById(R.id.read);
        bt1.setOnClickListener(new Read());
        Button bt2=(Button)findViewById(R.id.write);
        bt2.setOnClickListener(new Write());
    }
    class Read implements OnClickListener {
         public void onClick(View v) {
            try {
                FileInputStream in=openFileInput("test.dat");
                byte[] dat=new byte[in.available()];
                in.read(dat);
                TextView textView=(TextView)findViewById(R.id.textView);
                textView.setText(new String(dat));
                in.close();
            } catch (IOException e) { }
        }
    }
    class Write implements OnClickListener {
        public void onClick(View v) {
            try {
                FileOutputStream out=openFileOutput("test.dat",MODE_PRIVATE);
```

```
                EditText editText=(EditText)findViewById(R.id.editText);
                String msg=editText.getText().toString();
                out.write(msg.getBytes());
                out.close();
            } catch (IOException e) { }
        }
    }
}
```

「練習問題 14-1」例題 14-1 と同じことを追加書き込みモードで行いなさい。

・activity_main.xml
例題 14-1 と同じ

・MainActivity.java

```
package com.example.file2;

import java.io.*;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button bt1=(Button)findViewById(R.id.read);
        bt1.setOnClickListener(new Read());
        Button bt2=(Button)findViewById(R.id.write);
        bt2.setOnClickListener(new Write());
    }
    class Read implements OnClickListener {
         public void onClick(View v) {
            try {
                FileInputStream in=openFileInput("test.dat");
                byte[] dat=new byte[in.available()];
                in.read(dat);
                TextView textView=(TextView)findViewById(R.id.textView);
                textView.setText(new String(dat));
                in.close();
            } catch (IOException e) { }
        }
```

```
    }
    class Write implements OnClickListener {
        public void onClick(View v) {
            try {
                FileOutputStream out=openFileOutput("test.dat",  ①  );
                EditText editText=(EditText)findViewById(R.id.editText);
                String msg=editText.getText().toString();
                out.write(msg.getBytes());
                out.close();
            } catch (IOException e) { }
        }
    }
}
```

15 章　GoogleMap

GoogleMap を Android で使用するには「Android Maps API Key」を取得する必要があります。またプロジェクトの作成に当たっては通常のプロジェクトとは以下の点が異なります。

・Google APIs ライブラリのインストール（一度設定すればよい）
・map 用 AVD の生成（一度設定すればよい、実機では必要ない）
・ビルド・ターゲットに「Google APIs」を指定（その都度指定）

GoogleMap を利用してできることは以下のようなものです。この章ではこれらの使用方法を説明します。

・地図（通常の地図、衛星写真、交通情報）の表示
・マップコントローラによる位置やズームの制御
・ロケーション API による現在位置の取得
・オーバーレイ機能（本書では説明しません）

15-1　Android アプリから GoogleMap を使うのに必要なもの

1.　Android Maps API Key の取得

　①～②の手順で行ってください。場合によっては Google のアカウント（gmail のアカウントと同じ)が必要になる場合もあります。

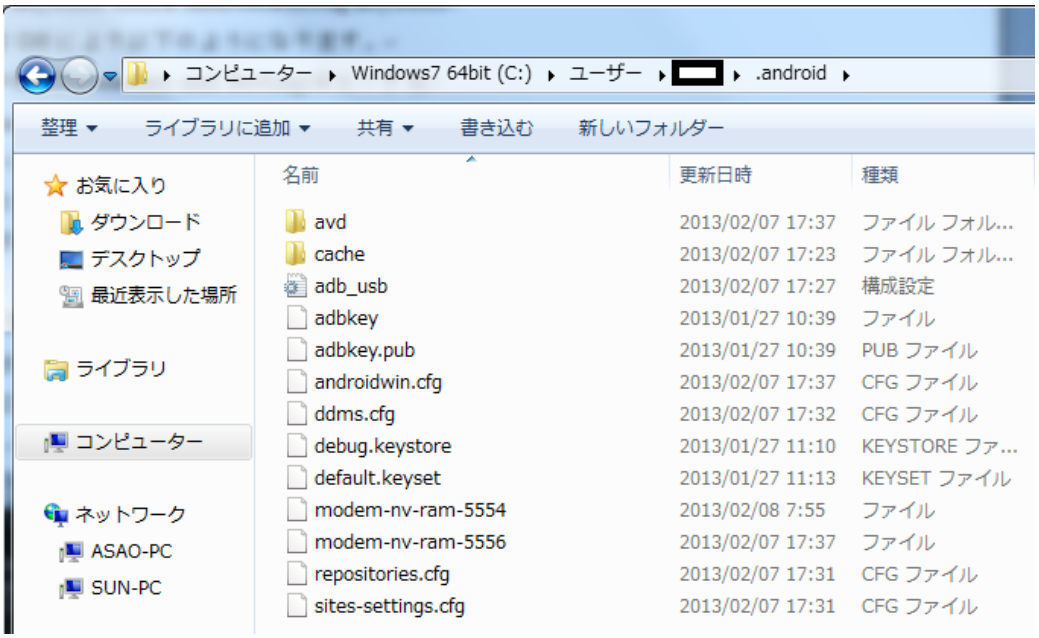
①証明書のフィンガープリント(MD5)の取得

　JDK をインストールしたフォルダの bin フォルダにある keytool で、証明書のフィンガープリント(MD5)を表示します。

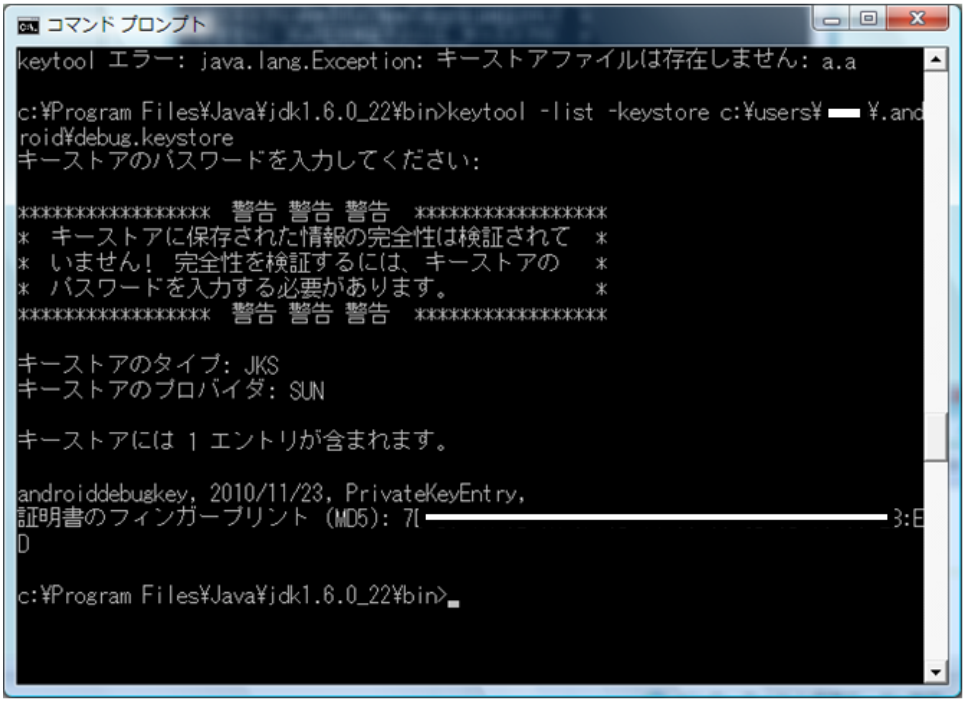
>keytool -list -keystore XXX¥.android¥debug.keystore

「注」JDK はたとえば以下のようなフォルダにインストールされていますので、カレントディレクトリを bin フォルダに移してから keytool コマンドを実行します。
C:¥Program Files¥Java¥jdk1.6.0_07¥bin

「注」XXX¥.android¥debug.keystore は KEYSTORE ファイルへのパスです。
XXX は OS により以下のようになります。
WindowsXP⇒C:¥Documents and Settings¥ユーザ名
WindowsVista/7⇒ C:¥Users¥ユーザ名
Mac/Linux⇒~/.android/debug.keystore

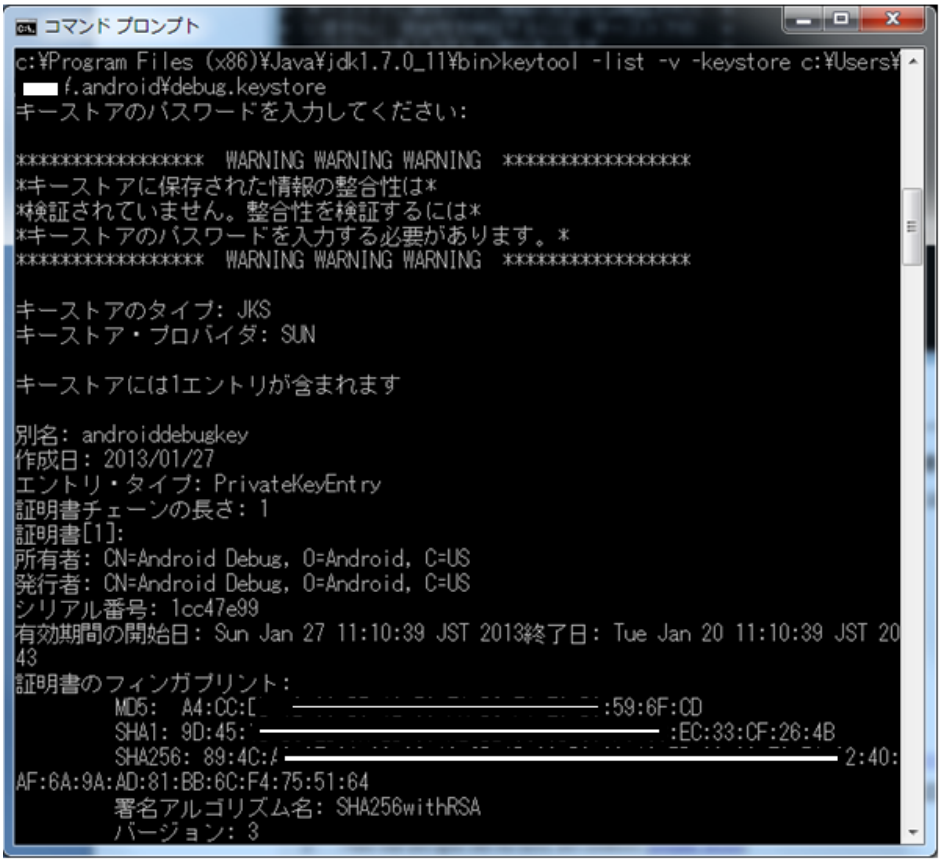


パスワードを聞かれますので、そのままリターンキーを押すと「証明書のフィンガープリント(MD5)」が表示されます。この値（たとえば「XX:XX:XX:XX:80:B5:12:59:FA:BC:F1:E8:D9:59:XX:XX」）を Maps API Key の取得の際に使います。



「注」従来の署名アルゴリズムは MD5 でしたが、よりセキュリティレベルの高い SHA1 に移行しつつあります。JDK1.7 の keytool コマンドで取得できるデフォルトの証明書のフィンガープリントは SHA1 です。MD5 の証明書のフィンガープリントを取得するには「-v」オプションを指定します。

>keytool -list -v -keystore XXX¥.android¥debug.keystore

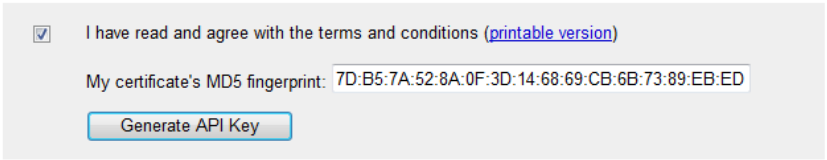


②Maps API Key の取得

Google の

https://developers.google.com/maps/documentation/android/v1/maps-api-signup?hl=ja

で示す Web ページを開きます。チェックとフィンガープリント(MD5)を入力します。



「注」Maps API Key の取得の Web ページの URL は変更されることがあります。以前の URL は「http://code.google.com/intl/ja/android/maps-api-signup.html」でした。

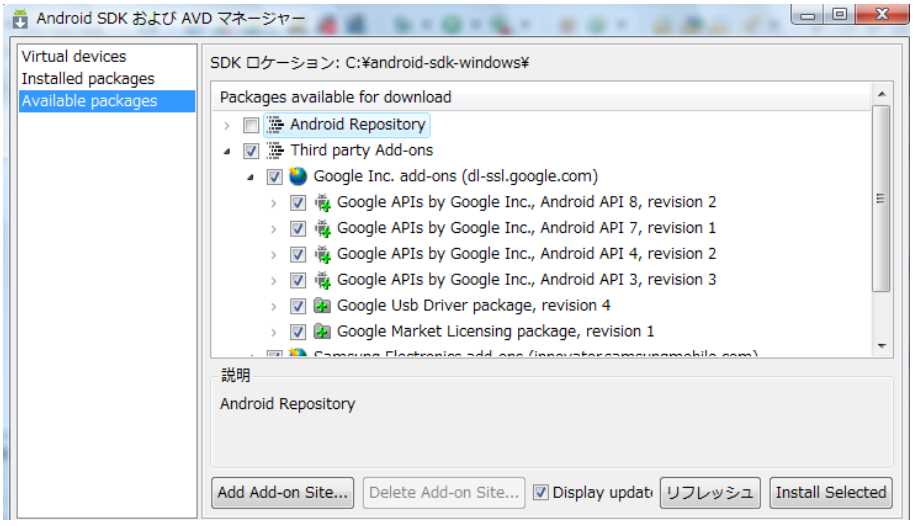
以下の MapView を示す XML テキストが得られますので、これを activity_main.xml に指定します。

```
<com.google.android.maps.MapView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:apiKey="0O4FkQeWz-4UvMj7ueE9iFUQGMepbSY6sC_QydA"
/>
```

「注」Android Maps API Key を開発環境で使用する場合、有効期限があります。有効期限が過ぎた後で、プロジェクトを修正してコンパイルし直した場合 Map は表示されません。修正しないものは再コンパイルしても表示されます。有効期限が過ぎた場合は①、②の処理をやり直して新しい Android Maps API Key を取得してください。

2. Google APIs ライブラリのインストール

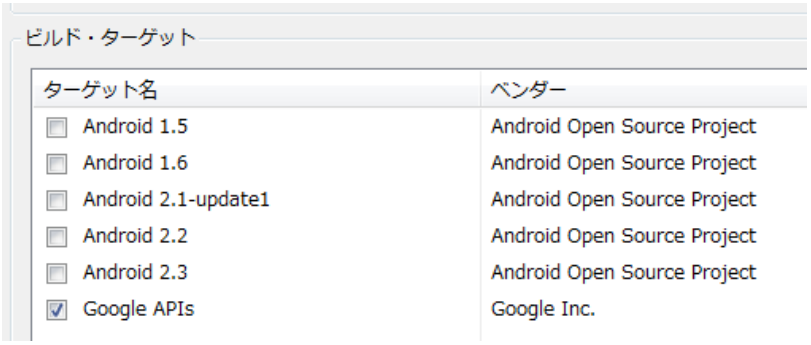
Eclipse の「ウインドウ」－「Android　SDK および AVD マネージャー」を選択します。「Available packages」を選択し「Third party Add-ons」をチェックし、「Google APIs」関連をチェックしインストールします。



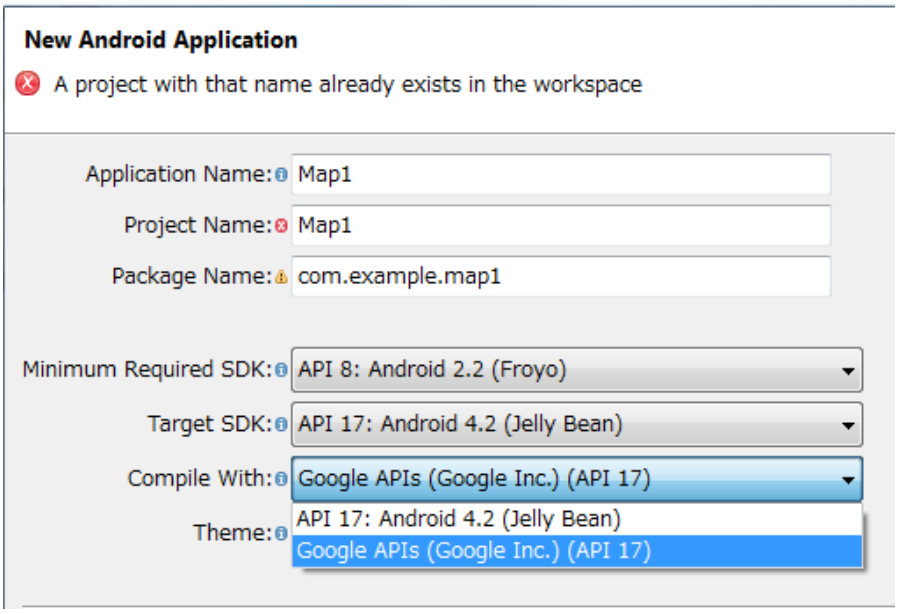
3.　Map プロジェクトの作成

Map アプリは「MapActivity」クラスを継承します。Map の表示は「MapView」に対して行います。以下の①~⑥の手順でプロジェクトを作成します。

①ビルド・ターゲットに「Google APIs」を選択

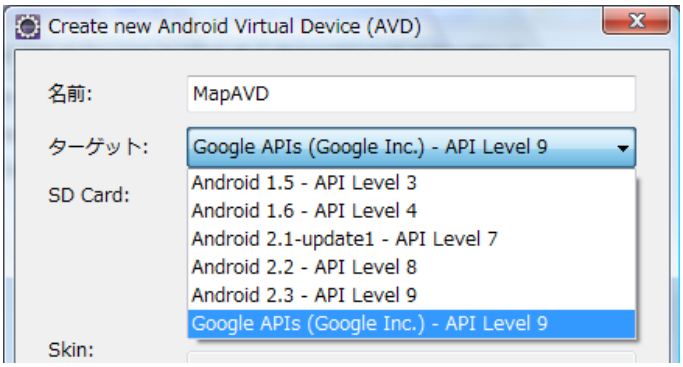


または

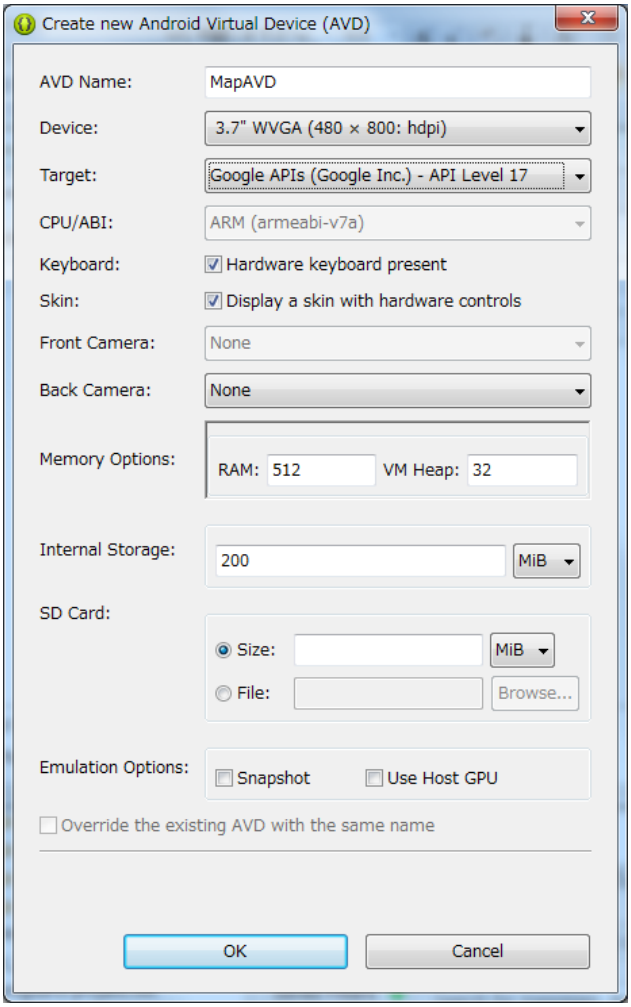


②map 用 AVD の生成（一度だけ）

「ウインドウ」－「Android　SDK および AVD マネージャー」を選択



または



③マニフェストにライブラリとパーミションを設定
　下線部を追加します。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        ・
        ・
    <uses-permission android:name="android.permission.INTERNET"/>
    <application android:icon="@drawable/ic_launcher"
android:label="@string/app_name">
          ・
          ・
        </activity>
        <uses-library android:name="com.google.android.maps"/>
    </application>
</manifest>
```

④activity_main.xml の記述
　マップを表示するウイジェットに<com.google.android.maps.MapView>を指定します。apiKey に先に取得してある Android Maps API Key を指定します。

・activity_main.xml

```
<?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"
    >
    <com.google.android.maps.MapView
        android:id="@+id/mapview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:enabled="true"
        android:clickable="true"
        android:apiKey="0O4FkQeWz-4W1Zt7V14-vmp3wnDGr8WW9nya5vg "
    />
```

</LinearLayout>

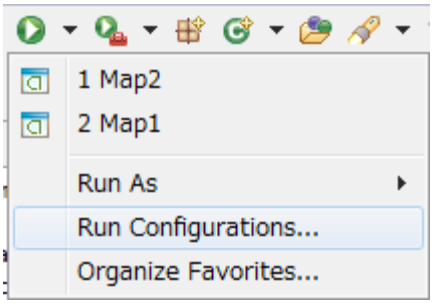
⑤Java ソースコードの記述

地図を表示するだけなら、MapView を置いてある、activity_main.xml を「setContentView(R.layout.activity_main);」で表示するだけでよいです。

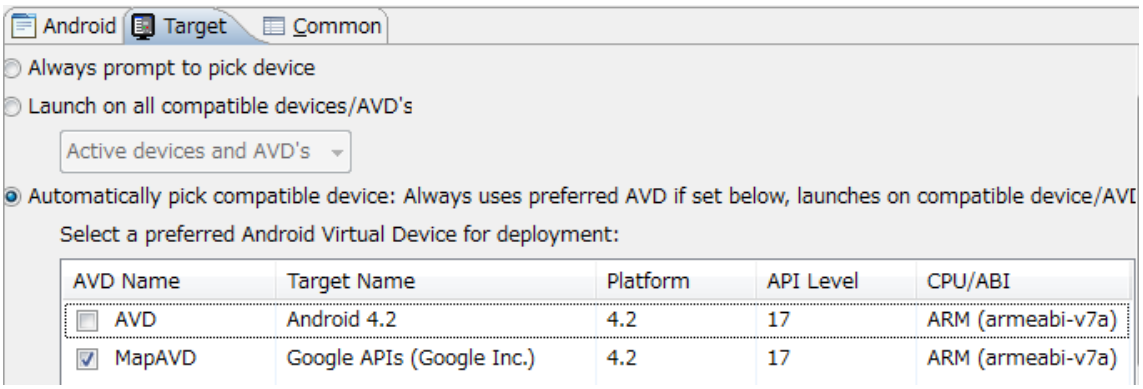
MapActivity クラスでは isRouteDisplayed メソッドを実装しなければいけません。ルート情報を表示する場合は true を、そうでない場合は false を返します。ただし、Android Maps には走行方向が分かる機能が用意されているわけではないので、こうしたルート情報は自前で実装しなければなりません。従って通常は false を返すだけの処理となります。

⑥実行

通常の実行で MapAVD が起動しない場合は「Run Configurations(実行の構成)」を選択します。



Target で通常の「AVD」かマップ用の「MapAVD」かを選択します。



「例題 15-1」単に地図表示だけをします。地図の位置を指定していないので、デフォルト位置が採用されます。エミュレータではアメリカ、実機では日本です。

・マニフェスト（AndroidManifest.xml）

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
・
・
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
・
・
        <activity
・
・
       </activity>
        <uses-library android:name="com.google.android.maps"/>
    </application>
</manifest>
```

・activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <com.google.android.maps.MapView
        android:id="@+id/mapview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:enabled="true"
        android:clickable="true"
        android:apiKey="0O4FkQeWz-4UvMj7ueE9iFUQGMepbSY6sC_QydA"
    />
</LinearLayout>
```

・Map1.java

```
package com.example.map1;

import com.google.android.maps.MapActivity;
import android.os.Bundle;

public class MainActivity extends MapActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    protected boolean isRouteDisplayed() {
        return false;
    }
}
```

・エミュレータ

・実機

16 章　センサーとカメラ（実機のみ）

Android 端末には各種センサーが搭載されています。加速度センサ、磁界（磁気）センサ、方位センサ、ジャイロセンサ、輝度（照度）センサ、圧力センサ、温度センサ、近接センサなどです。どのセンサーが使えるかは実機ごとに異なります。

センサーを使用できるようにするには、センサーマネージャーを使って、使用したいセンサーを取得します。さらに取得したセンサーにリスナーを付けセンサーの変化で処理を行います。この章では方位センサーと加速センサーの使用方法を説明します。

Android のカメラ機能は Camera クラスのオブジェクトを使用してプレビュー、オートフォーカス、撮影といった動作を行います。カメラのプレビュー画面は SurfaceView に表示します。プレビュー、オートフォーカス、撮影という動作は Camera クラスのメソッドで行うことができます。撮影した画像は内部メモリに保存されていますので、実際のファイルとして保存するのはユーザが別途コードを書かなければなりません。このようなカメラの撮り方や画像の保存方法、フォーカスの設定方法などの基本処理と撮影した写真に日付をプリントする方法を説明します。

16-1　方位センサー

方位センサー（Sensor.TYPE_ORIENTATION）は端末が置かれている向きを、方位（東西南北）、ピッチ（表裏）、ロール（縦横）の 3 方向で調べます。

1.　getSensorList と getDefaultSensor

センサーを使用できるようにするには SensorManager を使ってセンサーマネージャー sm を取得します。

```
SensorManager sm=(SensorManager)getSystemService(SENSOR_SERVICE);
```

センサーマネージャーsm からセンサーオブジェクトを取得するには getSensorList メソッドを使う方法と getDefaultSensor メソッドを使う方法があります。getSensorList メソッドは要求されたタイプのセンサーを全て Sensor リストに取得します。たとえば、方位センサーが複数の種類組み込まれている場合などです。この中から希望するものを選んで使用することができます。たいがい「sensors.get(0);」で取得する第 1 番目のセンサを使用します。これに対し getDefaultSensor メソッドは第 1 番目（デフォルト）のセンサーを単純に取得します。従って

```
List<Sensor> sensors =sm.getSensorList(Sensor.TYPE_ORIENTATION);
if (sensors.size()>0) {
    Sensor sensor=sensors.get(0);
```

と

```
Sensor sensor=sm.getDefaultSensor(Sensor.TYPE_ORIENTATION);
```

は同じです。

2.　センサーリスナーの設定

取得したセンサーマネージャーsm に反応するリスナーを registerListener メソッドで設定します。

```
sm.registerListener(this,sensor,SensorManager.SENSOR_DELAY_NORMAL);
```

registerListener メソッドの第 3 引数はセンサーを呼び出す感度（次に呼び出すまでの遅延時間）で以下の 4 種類が指定できます。()内の遅延時間は目安です。

センサー感度（遅延時間）	意味
SENSOR_DELAY_FASTEST	高速（遅延時間 0 ミリ秒）。
SENSOR_DELAY_GAME	ゲーム用（遅延時間 20 ミリ秒）。
SENSOR_DELAY_UI	ユーザインターフェース用（遅延時間 60 ミリ秒）。
SENSOR_DELAY_NORMAL	ノーマル（遅延時間 200 ミリ秒）。

SensorEventListener リスナーをインプリメントした場合に実装しなければならないメソッドは以下です。

SensorEventListener のメソッド	意味
onSensorChanged	センサーの値が変わった時に呼び出されます。
onAccuracyChanged	センサーの精度が変わった時に呼び出されます。

使用できるようになったセンサーの状態が変化すると onSensorChanged メソッドが呼び出されます。

```
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType()==Sensor.TYPE_ORIENTATION) {

    }
}
```

引数の event.value[0]、event.value[1]、event.value[2]でセンサーの各種値を取得できます。たとえば方位センサの場合の値の意味は以下のようになります。

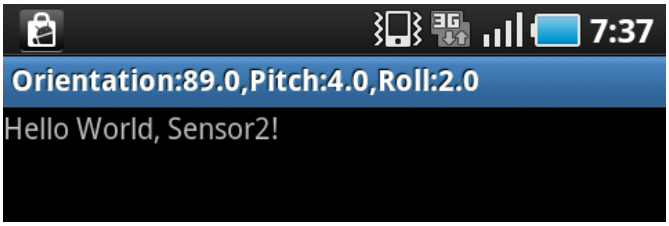
値	意味
values[0]	方位（z 軸回りの回転角度）で 0~359。 0：北、90：東、180：南、270：西
values[1]	ピッチ（x 軸回りの回転角度）で-180~+180。 上向き水平：0、縦長正立：-90、裏向き：-180(+180)、縦長逆立：90
values[2]	ロール（y 軸回りの回転角度）で-90~+90。 水平：0、右肩上：90、左肩上：-90

「例題 16-1」方位センサの方位、ピッチ、ロールの各値をタイトルバーに表示します。

・MainActivity.java

```
package com.example.sensor1;

import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;

public class MainActivity extends Activity implements SensorEventListener {
    private SensorManager sm;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sm=(SensorManager)getSystemService(SENSOR_SERVICE);
        Sensor sensor=sm.getDefaultSensor(Sensor.TYPE_ORIENTATION);
        sm.registerListener(this,sensor,SensorManager.SENSOR_DELAY_NORMAL);
    }
    public void onSensorChanged(SensorEvent event) {
        String result="";
        if (event.sensor.getType()==Sensor.TYPE_ORIENTATION) {
            result="Orientation:"+event.values[0];
            result+= ",Pitch:"+event.values[1];
            result+= ",Roll:"+event.values[2];
        }
        setTitle(result);
    }
    public void onAccuracyChanged(Sensor arg0, int arg1) {

    }
}
```

「注」Android 4.2 では `Sensor.TYPE_ORIENTATION` は非推奨です。



　上の結果は机の上に表向き水平に東の方向に向けて実機を置いた状態です。
タイトルバーに表示しきれない場合は Orientation→O のように文字数を減らしてください。

17 章　音声合成

Android の音声合成（テキストの読み上げ）は TextToSpeech engine（TTS エンジン）を用いて行います。主な手順は以下です。読み上げる音程や読み上げ速度を変更することもできます。

・TTS エンジン・リソースの生成
・言語の設定
・読み上げをする文章を指定して speak メソッドを呼び出す
・TTS エンジン・リソースの解放

読み上げることができる言語は英語、フランス語、ドイツ語、イタリア語、スペイン語など（機種依存します）で、日本語は機種によりサポートされていません。

音声合成はエミュレータ上でも確認できます。

17-1　英語テキストを発音する

TextToSpeech クラスの TTS エンジン・リソース ts を生成し、OnInitListener をインプリメントし、onInit でテキスト読み上げの準備を行い、setSpeechRate メソッドで読み上げます。

1.　テキスト読み上げの準備

以下のようにして TTS エンジン・リソース ts を生成します。

```
TextToSpeech ts=new TextToSpeech(this,this);
```

OnInitListener をインプリメントし、onInit でテキスト読み上げの準備を行います。status に TextToSpeech.SUCCESS が返されれば成功です。成功したら setLanguage で読み上げを行う言語を指定します。言語には Locale.ENGLISH や Locale.FRENCH を指定できます。Locale.JAPANESE は機種によりサポートされていません。。

```
public void onInit(int status) {
    if (status==TextToSpeech.SUCCESS){
        Locale locale=Locale.ENGLISH;
        if (ts.isLanguageAvailable(locale)>=TextToSpeech.LANG_AVAILABLE)
            ts.setLanguage(locale);
        else
            //この言語は未サポートです
    }
    else
        //音声合成できません
    }
```

2.　テキスト読み上げ

たとえば「Good morning」を読み上げるには speak メソッドを使い以下のようにします。

```
if (ts.isSpeaking()) ts.stop();
ts.setSpeechRate(1.0f);
ts.speak("Good morning",TextToSpeech.QUEUE_FLUSH,null);
```

キューモードは以下の通りです。読上げ速度は setSpeechRate メソッドで「0.1～2.0」の値を指定します。

TextToSpeech のキューモード	機能
QUEUE_ADD	再生キューへエントリを追加。
QUEUE_FLUSH	再生待ちのエントリをドロップしてエントリを実行。

TextToSpeech クラスの主なメソッドとして以下があります。

・boolean isSpeaking()
TTS エンジンが話中なら true を返します。

・int playEarcon(String earcon, int queueMode, HashMap<String, String> params)
earcon で示す通知音を出します。queueMode は QUEUE_ADD または QUEUE_FLUSH を指定します。

・int playSilence(long durationInMs, int queueMode, HashMap<String, String> params)
durationInMs で示すミリ秒の無音（サイレント）を出します。queueMode は QUEUE_ADD または QUEUE_FLUSH を指定します。

・int setLanguage(Locale loc)
loc で示すロケールの言語に設定します。

・int setPitch(float pitch)
声の高さを指定します。pitch には 0.1～2.0 の範囲を指定します。1.0 がノーマルのデフォルト。

・int setSpeechRate(float speechRate)
読上げ速度を指定します。speechRate には 0.1～2.0 の範囲を指定します。1.0 がノーマルのデフォルト。

・void shutdown()
TTS エンジン・リソースを解放します。

・int speak(String text, int queueMode, HashMap<String, String> params)
text で示すテキストを読み上げます。queueMode は QUEUE_ADD または QUEUE_FLUSH を指定します。

・int stop()
読み上げを停止します。

3. TTS エンジン・リソースの解放

使用している TTS エンジン・リソースの ts を Activity の終了で解放します。

```
protected void onDestroy(){
    super.onDestroy();
    if (ts!=null) ts.shutdown();
}
```

TTS エンジンは非同期で動作しているので、Activity がバックグラウンドに回っても読み上げは続きます。バックグラウンドに回ったら TTS エンジン・リソースを解放し読み上げを終了するには onDestroy メソッドでなく onPause メソッドで解放処理を行います。

```
protected void onPause () {
    super.onPause ();
    if (ts!=null) ts.shutdown();
}
```

「例題 17-1」「Good morning」を英語で読み上げます。

・MainActivity.java

```
package com.example.speech1;

import java.util.Locale;
import android.app.Activity;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
import android.view.MotionEvent;
import android.widget.*;

public class MainActivity extends Activity implements OnInitListener{
    private TextToSpeech ts;
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ts=new TextToSpeech(this,this);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            String txt="Good morning";
            if (ts.isSpeaking()) ts.stop();
            ts.setSpeechRate(1.0f);
            ts.speak(txt,TextToSpeech.QUEUE_FLUSH,null);
        }
        return super.onTouchEvent(event);
    }
    public void onInit(int status) {
        if (status==TextToSpeech.SUCCESS){
            Locale locale=Locale.ENGLISH;
            if (ts.isLanguageAvailable(locale)>=TextToSpeech.LANG_AVAILABLE)
                ts.setLanguage(locale);
            else
                Toast.makeText(this,"この言語は未サポートです
",Toast.LENGTH_LONG).show();
        }
        else
            Toast.makeText(this,"音声合成できません",Toast.LENGTH_LONG).show();
    }
    protected void onDestroy(){
        super.onDestroy();
        if (ts!=null) ts.shutdown();
    }
}
```

18 章　音声認識（実機のみ）

Android で音声認識を使用するには、android.speech パッケージの RecognizerIntent クラスを使用します。RecognizerIntent は音声認識ライブラリをインテント経由で使用するためのクラスです。このインテントを実行すると音声認識プロンプトが表示されますので、マイクに向かって話しかけると音声認識が実行されます。音声認識を行った後、そのデータを使って行える処理は以下の 2 つです。

・認識された音声を文字列として取得することができます。
（ACTION_RECOGNIZE_SPEECH）
・認識された音声を使用してウェブ検索した結果が、画面表示されます。
（ACTION_WEB_SEARCH）

音声認識の言語は日本の機種ではデフォルトで日本語ですが、英語やフランス語での入力もできます。

Android の音声認識機能は、端末とサーバとで処理を分担する分散型音声認識(DSR:Distributed Speech Recognition)と呼ばれる方式です。このため，音声認識機能を使用する際にはサーバに接続するために 3G または WiFi が有効である必要があります。マニフェストに記述する必要はありません。

18-1　音声入力した言葉をトーストで表示

1.　音声認識インテント

音声認識インテントの種類として以下の 2 つがあります。音声認識をし、認識された音声を文字列として取得するには次のようなインテントを生成します。

```
Intent it=new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
```

RecognizerIntent の種類	意味
ACTION_RECOGNIZE_SPEECH	認識された音声を文字列として取得することができます。
ACTION_WEB_SEARCH	認識された音声を使用してウェブ検索した結果が、画面表示されます。

このインテントに対し putExtra で付加情報を設定します。設定できる付加情報として以下があります。

putExtra で設定できる付加情報	意味
EXTRA_LANGUAGE_MODEL	音声モデル。必須の項目で以下の 2 つの定数が指定できます。 ・LANGUAGE_MODEL_FREE_FORM 自由形式音声認識に基づく言語モデル ・LANGUAGE_MODEL_WEB_SEARCH Web サーチ用語に基づく言語モデル
EXTRA_LANGUAGE	認識する言語を文字列で指定します。日本の機種のデフォルトは日本語。 以下のような文字列を指定。 Locale.JAPANESE.toString() Locale.ENGLISH.toString() Locale.FRENCH.toString() など
EXTRA_PROMPT	音声認識プロンプトに表示するユーザメッセージを文字列で指定します。
EXTRA_MAX_RESULTS	結果の最大数を整数値で指定します。指定しなければ 1 つ。Android 4.2 以後は指定しなければ複数。

音声モデルに LANGUAGE_MODEL_FREE_FORM を指定し、ユーザメッセージを指定してインテントを発行するには以下のようにします。ICode はインテントの ID でインテントの結果を処理する onActivityResult メソッドで照合用に使います。

```
it.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
t.putExtra(RecognizerIntent.EXTRA_PROMPT,"何かお話してね！");
it.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS,1);
startActivityForResult(it,ICode);
```

「注」Android 4.2 以後は 1 つだけの結果を得たい場合は結果の最大数を明示的に「1」に設定しなければなりません。

```
it.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS,1);
```

2. 結果の取得

音声認識インテントで表示される音声入力プロンプトに対し音声入力を行い、一定の無音状態が続くと onActivityResult メソッドが呼ばれ、入力結果が引数の data に返されます。「data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);」で入力音声が日本語テキストの配列リストとして取得できます。

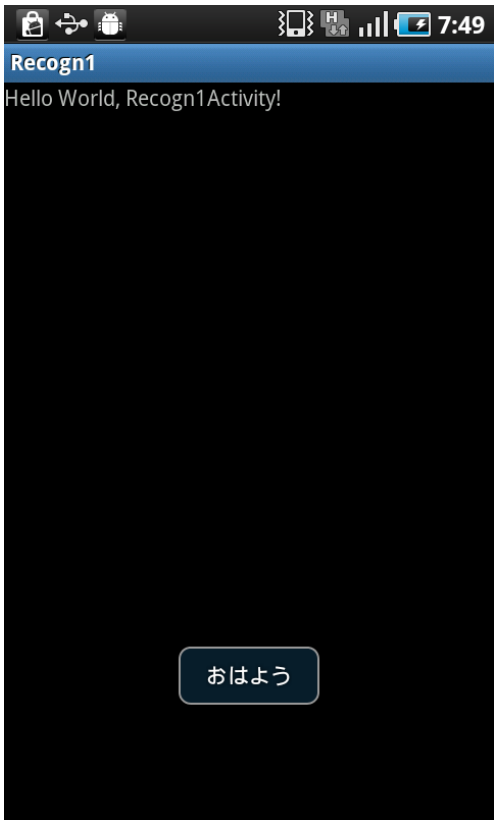
```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode==ICode && resultCode==RESULT_OK) {
        String msg="";
        ArrayList<String> results =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
        for (int i=0;i<results.size();i++) {
            msg+=results.get(i);
        }
        Toast.makeText(this,msg,Toast.LENGTH_LONG).show();
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

「例題 18-1」タッチで音声認識インテントを呼び出し、入力した音声を文書にしてトーストで表示します。

・MainActivity.java

```
package com.example.recogn1;

import java.util.ArrayList;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.view.MotionEvent;
import android.widget.Toast;

public class MainActivity extends Activity {
    private int ICode=0; // インテントID
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            try {
                Intent it=new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

it.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
                it.putExtra(RecognizerIntent.EXTRA_PROMPT,"しゃべったことをToastで表示するよ！");
                it.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS,1);
                startActivityForResult(it,ICode);
            } catch (ActivityNotFoundException e) { }
        }
        return super.onTouchEvent(event);
```

```
    }
    protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
        if (requestCode==ICode && resultCode==RESULT_OK) {
            String msg="";
            ArrayList<String>
results=data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
            for (int i=0;i<results.size();i++) {
                msg+=results.get(i);
            }
            Toast.makeText(this,msg,Toast.LENGTH_LONG).show();
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```



19 章　リバーシーゲーム

リバーシーゲームを以下のような段階を追って作ります。

1. 盤面を作る
2. 黒石を置く
3. 盤面の情報を配列に置く
4. 黒番白番で交互に置く
5. 石を置ける位置かどうかチェック
6. 自動的に反転する
7. コンピュータが手を打つ
8. コンピュータに戦略を持たせる
9. 完成版

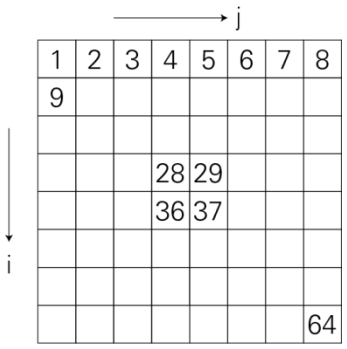
Reversi1.java を元に例題が進むごとに前の例題に対し、追加または変更された箇所を赤色で区別します。

19-1　盤面を作る

8×8 のマスに 3 種類のイメージを配置します。配置する ImageView には左上隅の 1~右下隅隅の 64 の通し番号（ID）を割り当てます。

green.jpg	緑（石を置いていない状態）のイメージファイル
white.jpg	白を置いた状態のイメージファイル
black.jpg	黒を置いた状態のイメージファイル

図 19-1 盤面



LinearLayout の child に横 8 個の ImageView を配置し、main に縦方向に 8 段配置します。ImageView の ID は「(i-1)*8+j」という式を使います。i と j が 1~8 まで変化する間に 1~64 の番号が作られます。ID の 28 と 37 は白、29 と 36 は黒、その他は緑とします。イメージリソースは WVGA 画面では 60×60 ピクセルを drawable-hdpi、HVGA 画面では 40×40 ピクセルを drawable-mdpi に置きます。画面密度が異なっても対応できるようにイメージのサイズは「WH=green.getWidth();」で取得し、取得した「WH」の値で ImageView のサイズを設定します。

```
for (int i=1;i<=8;i++){
    child = new LinearLayout(this);
    child.setOrientation(LinearLayout.HORIZONTAL);
    for (int j=1;j<=8;j++){
        ImageView iv=new ImageView(this);
        ・
        ・
        iv.setId((i-1)*8+j);
        child.addView(iv,new LinearLayout.LayoutParams(WH,WH));
```

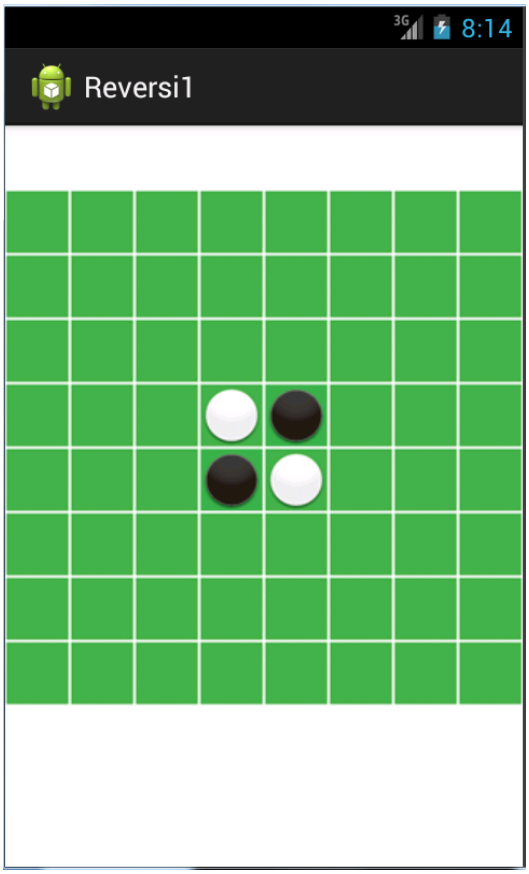
```
        }
        main.addView(child,new LinearLayout.LayoutParams(WC,WC));
    }
```

「例題 19-1」

・MainActivity.java

```
package com.example.reversi1;

import android.app.Activity;
import android.content.res.Resources;
import android.graphics.*;
import android.os.Bundle;
import android.view.*;
import android.widget.*;

public class MainActivity extends Activity {
    private int WC=ViewGroup.LayoutParams.WRAP_CONTENT;
    private int WH;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Resources r=getResources();
        Bitmap green=BitmapFactory.decodeResource(r,R.drawable.green);
        Bitmap white=BitmapFactory.decodeResource(r,R.drawable.white);
        Bitmap black=BitmapFactory.decodeResource(r,R.drawable.black);
        WH=green.getWidth();
        LinearLayout main=new LinearLayout(this);
        main.setOrientation(LinearLayout.VERTICAL);
        main.setPadding(0,WH,0,0);
        setContentView(main);
        LinearLayout child;
        for (int i=1;i<=8;i++){
            child = new LinearLayout(this);
            child.setOrientation(LinearLayout.HORIZONTAL);
            for (int j=1;j<=8;j++){
                ImageView iv=new ImageView(this);
```

```
                if (i==4 && j==4 || i==5 && j==5){
                    iv.setImageBitmap(white);
                }
                else if (i==4 && j==5 || i==5 && j==4){
                    iv.setImageBitmap(black);
                }
                else {
                    iv.setImageBitmap(green);
                }
                iv.setId((i-1)*8+j);
                child.addView(iv,new LinearLayout.LayoutParams(WH,WH));
            }
            main.addView(child,new LinearLayout.LayoutParams(WC,WC));
        }
    }
}
```



著者略歴

河西 朝雄（かさいあさお）
山梨大学工学部電子工学科卒（1974 年）。長野県岡谷工業高等学校情報技術科教諭、長野県松本工業高等学校電子工業科教諭を経て、現在は「カサイ．ソフトウエアラボ」代表。

「主な著書」

「入門ソフトウエアシリーズC言語」、「同シリーズJava言語」、「同シリーズC++」、「入門新世代言語シリーズVisualBasic4.0」、「同シリーズDelphi2.0」、「やさしいホームページの作り方シリーズHTML」、「同シリーズJavaScript」、「同シリーズHTML機能引きテクニック編」、「同シリーズホームページのすべてが分かる事典」、「同シリーズiモード対応HTMLとCGI」、「同シリーズiモード対応Javaで作るiアプリ」、「同シリーズVRML2.0」、「チュートリアル式言語入門VisualBasic.NET」、「はじめてのVisualC#.NET」、「C言語用語辞典」ほか（以上ナツメ社）

「構造化 BASIC」、「Microsoft Language シリーズ Microsoft VISUAL C++初級プログラミング入門上、下」、「同シリーズ VisualBasic 初級プログラミング入門上、下」、「C 言語によるはじめてのアルゴリズム入門」、「Java によるはじめてのアルゴリズム入門」、「VisualBasic によるはじめてのアルゴリズム入門」、「VisualBasic6.0 入門編、中級テクニック編、上級編」、「Internet Language 改訂新版シリーズ ホームページの制作」、「同シリーズ JavaScript 入門」、「同シリーズ Java 入門」、「New Language シリーズ標準VisualC++プログラミングブック」、「同シリーズ標準 Java プログラミングブック」、「VB.NET 基礎学習 Bible」、「原理がわかるプログラムの法則」、「プログラムの最初の壁」、「河西メソッド : C 言語プログラム学習の方程式」、「基礎から学べる VisualBasic2005 標準コースウエア」、「基礎から学べる JavaScript 標準コースウエア」、「基礎から学べる C 言語標準コースウエア」、「基礎から学べる PHP 標準コースウエア」、「なぞりがき C 言語学習ドリル」、「C 言語　標準ライブラリ関数ポケットリファレンス[ANSI C,ISO C99 対応]」、「 C 言語　標準文法ポケットリファレンス[ANSI C,ISOC99 対応]」ほか（以上技術評論社）

Android プログラミング完全入門

2014 年 6 月 1 日　初版　第 1 刷発行
著者＝河西 朝雄
発行者＝河西 朝雄
発行所＝カサイ．ソフトウエアラボ
長野県茅野市ちの 813　TEL.0266-72-4778

デザイン＝河西 朝樹

定価＝1,620 円（税込）