Android 2.x / 4.x 対応

Android

プログラミングBible

上級 各種処理

河西 朝雄 著

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、Linux カーネル層、ライブラリ層、Android ランタイム層、アプリケーションフレームワーク層、アプリケーション層などで構成されます。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。

Android や iPone などのスマートフォンや iPad などのタブレット端末のユーザーインターフェースは指のタッチを基本とし、カメラやセンサを内蔵し、音声認識・音声合成などが簡単に利用できる画期的なコンピュータです。マウス、キーボード、ディスプレイが主なユーザーインターフェースとするパソコンとは大きく異なります。「コンピュータ＝パソコン」の時代から「コンピュータ＝スマートフォン、タブレット端末」の時代に急速にパラダイムシフトしようとしています。スマートフォンは子供から女性、シニアまでの広い層に渡って、今までのパソコンユーザとは比べ物にならない数のユーザが見込まれます。

スマートフォンを iPone VS Android という構図で見た場合どちらにもメリット、デメリットがあり、一概にどちらが良いとは言えません。アプリケーションの開発言語の違いで見ると iPone（OS 名は iOS）は Objective-C、Android は Java です。Objective-C はややマイナーな言語であるのに対し Java はネットワーク関連ではメジャーな言語であるということです。このため数多くいる Java 経験者には Android の方が移行しやすい環境であると思います。

本書は Android のアプリを開発することを目的にしていますので、話を Android に絞ります。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。

本シリーズは、Android アプリを開発するための基本的なテクニックをすべて網羅するように 34 の章（カテゴリ）に分類し、「初級　基礎編」、「中級　Android 的プログラミング法」、「上級　各種処理」の 3 分冊で構成することにし、本書はその中の「上級　各種処理」です。

34 の章というのはかなり多い章分けですが、細かく章分けをすることでカテゴリが分かり易く、各章のサイズは小さくなり初心者には、ひとつのまとまった単位がボリュームが

少ないので、取りかかり易くなります。また、章の順序ではなく、知りたい章を先に学習することもできます。

既存の書籍やネット上の情報は重要な内容とそうでない情報がまぜこぜになっていたり、このプログラムをどこに書けばいいのかが曖昧だったり、サンプルが長すぎたりなど、初心者には理解しにくい内容が多いです。本シリーズでは Android アプリを作る上で必要な技術的要素やテクニックを切り出し短いサンプルを付けて簡潔に提示します。

「中級　Android 的プログラミング法」は Android の特徴を生かしたプログラミング法を説明しました。「上級　各種処理」はグラフィックス、ファイル処理、Google のメール、マップサービス、センサー、カメラなどのハードウエア制御、音声認識・音声合成、ネットワーク通信などの個別の処理について説明します。

グラフィックスについては「初級　基礎編」で説明してありますが 23 章でさらに詳しく説明します。ダブルバッファリングという手法を用いて表と裏の画面を切り替えながら描画処理を行う SurfaceView やグラフィックス処理のためのアプリケーションプログラミングインタフェースの OpenGL などの高度なグラフィックス機能について 24 章、25 章で説明します。

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。ファイルに対する読み書きの方法、ファイル名リストの取得方法やフォルダの操作方法などについて 26 章でを説明します。Android では SQLite というデータベース管理システムを組み込んでいて、API を使用してアプリケーションから利用することができます。SQLite について 27 章で説明します。

Google 社によるフリーメールサービスの Gmail や地図サービスの GoogleMap などのサービスを Android で利用する方法について 28 章、29 章で説明します。

Android 端末には加速度センサ、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、近接センサなど各種センサーが搭載されています。30 章でこれらセンサーの使い方を説明します。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、撮影した写真に日付をプリントしたり、撮影画像の上にイメージや手書きの絵を書き入れるといったオーバーレイ機能についてなどを 31 章で説明します。

Android では音声認識や音声合成（テキストの読み上げ）を簡単に行うことができます。音声認識ライブラリをインテント経由で使用することで、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、音声合成について 32 章、33 章で説明します。

ネットワーク通信を行うための主な方法として HTTP とソケットがあります。HTTP は、Web のサーバと、クライアント（ブラウザ）の間で、ウェブページを送受信するためのプロトコルです。ソケットは自分のコンピュータと相手のコンピュータをソケットで接続し

双方向でデータの受発信を行います。34 章でこれらのネットワーク通信の方法を説明します。ということで本書は次のような章の構成となります。

23 章　グラフィックス
24 章　SurfaceView
25 章　OpenGL
26 章　ファイル処理
27 章　SQLite
28 章　Gmail（実機のみ）
29 章　GoogleMap
30 章　センサー（実機のみ）
31 章　カメラ（実機のみ）
32 章　音声認識（実機のみ）
33 章　音声合成
34 章　ネットワーク通信

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 5 月
河西朝雄

本書のプログラムは「Eclipse 3.6 Helios」と「Android 2.2(API 8)」で開発しエミュレータ AVD の画面サイズは WVGA（480×800）です。実機は「SAMSUNG GALAXY S」で確認しました。

本書のプログラムはエミュレータ AVD の画面サイズを HVGA（320×480）でも確認しました。また「Eclipse 3.7 Indigo」と「Android 4.0.3(API 15)」でも確認しました。これらの環境において差異が生じるものは、その差異について個々の例題に「注」として記述しました。Android、Android SDK、Eclipse の特徴と注意点に関して「付録　Android、Android SDK、Eclipse のバージョン」にまとめてあります。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。

Android プログラミング Bible シリーズの他の本

Android プログラミング Bible シリーズは「初級　基礎編」、「中級　Android 的プログラミング法」、「上級　各種処理」の 3 分冊構成です。本書は「上級　各種処理」です。他の本の内容は以下です。

☆初級　基礎編

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
5 章　main.xml を使わずにレイアウトする
6 章　メニュー
7 章　トースト、ダイアログ、ログ
8 章　タッチイベント
9 章　キーイベント、フォーカスイベント

☆中級　Android 的プログラミング法

10 章　インテントとアクティビティ
11 章　Thread、Handler、Message
12 章　サービス
13 章　ブロードキャストレシーバ
14 章　コンテンツプロバイダ
15 章　マニフェスト
16 章　基本ウイジェットを機能強化したウイジェット
17 章　小物ウイジェット
18 章　高度なビュー系ウイジェット
19 章　アプリケーション・ウイジェット
20 章　マルチメデイア
21 章　リソース
22 章　アニメーション

目次（上級　各種処理）

23 章　グラフィックス

「初級　基礎編」の 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。

「注」この章の例題を HVGA（320×480）で表示する場合は座標値やテキストサイズのピクセル値を「1/1.5」にしてください。たとえば例題 23-1 なら以下のように変更します。

「canvas.drawLine(0,15,180,15,paint);」
→「canvas.drawLine(0,10,120,10,paint);」

「float[] p={0,30,90,90,90,90,180,30,180,30,0,30};」
→「float[] p={0,20,60,60,60,60,120,20,120,20,0,20};」

23-1　各種描画メソッド

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。

(x1,y1)－(x2,y2)に直線を描くには drawLine メソッドを使って以下のようにします。

```
canvas.drawLine(x1,y1,x2,y2,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くには drawLines メソッドを使って以下のようにします。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。

```
        float[] p={x1,y1,x2,y2,x3,y3,x4,y4・・・};
        canvas.drawLines(p,paint);
```

「例題 23-1-1」直線を描きます。

・Graphic1.java

```
package jp.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 Graphic1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new GView(this));
    }
    private class GView extends View {
        public GView(Context context) {
            super(context);
        }
        protected void onDraw(Canvas canvas) {
            Paint paint=new Paint();
            paint.setColor(Color.GREEN);
            canvas.drawLine(0,15,180,15,paint);
            float[] p={0,30,90,90,90,90,180,30,180,30,0,30};
            canvas.drawLines(p,paint);
        }
    }
}
```

「注」この節の以後の「例」は OnDraw メソッド内のコードのみ記述してあります。

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。

(x1,y1)位置に点を描画するには drawPoint メソッドを使って以下のようにします。

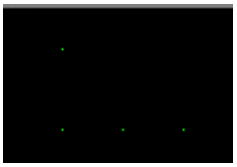
```
canvas.drawPoint(x1,y1,paint);
```

配列 p[ ]の 2 要素ごとに(x,y)とした点を描画くには drawPoints メソッドを使って以下のようにします。

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

「例」

```
Paint paint=new Paint();
paint.setColor(Color.GREEN);
canvas.drawPoint(30,20,paint);
float[] p={30,60,60,60,90,60};
canvas.drawPoints(p,paint);
```



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。

左上隅を(x1,y1)、右下隅を(x2,y2)とする矩形を描くには drawRect メソッドを使って以下のようにします。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。

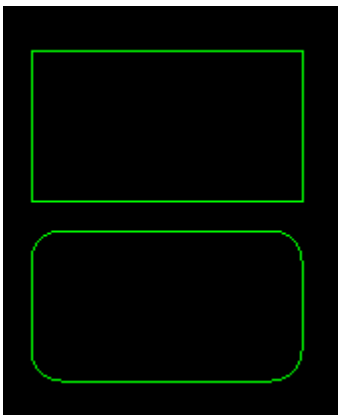
```
RectF r=new RectF(x1,y1,x2,y2);
canvas.drawRect(r, paint);
```

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 の丸みを付けるには drawRoundRect メソッドを使って以下のようにします。

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

「例」

```
Paint paint=new Paint();
paint.setColor(Color.GREEN);
paint.setStyle(Style.STROKE);
RectF r1=new RectF(15,30,150,105);
canvas.drawRect(r1, paint);
RectF r2=new RectF(15,120,150,195);
canvas.drawRoundRect(r2,15,15,paint);
```



「注」paint.setStyle(Style.STROKE);を使用する場合は「import android.graphics.Paint.Style;」が必要です。

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。

中心(x,y)、半径 r の円を描くには drawCircle メソッドを使って以下のようにします。

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

(x1,y1)－(x2,y2)の矩形に内接する楕円を描くには drawOval メソッドを使って以下のようにします。

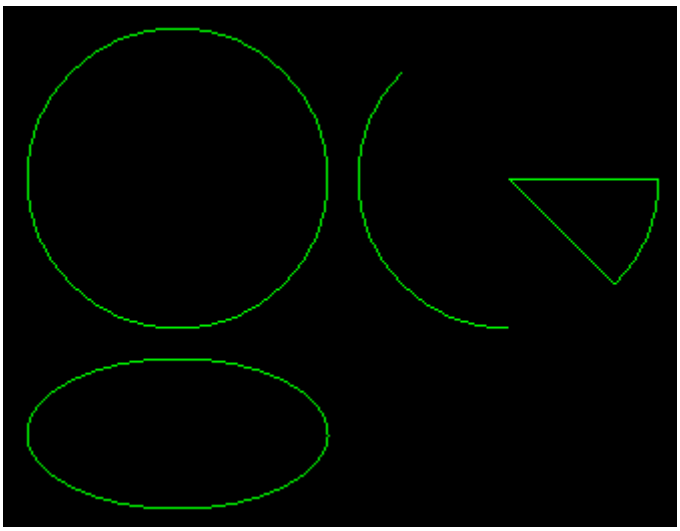
```
RectF r=new RectF(x1,y1,x2,y2);
canvas.drawOval(r,paint);
```

(x1,y1)－(x2,y2)の矩形に内接する楕円の a1°～(a1+a2)°までの扇形を描くには drawArc メソッドを使って以下のようにします。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。

```
RectF r=new RectF(x1,y1,x2,y2);
canvas.drawArc(r,a1,a2,true,paint);
```

「例」

```
Paint paint=new Paint();
paint.setColor(Color.GREEN);
paint.setStyle(Style.STROKE);
canvas.drawCircle(90,90,75,paint);

RectF r1=new RectF(15,180,165,255);
canvas.drawOval(r1,paint);

RectF r2=new RectF(180,15,330,165);
canvas.drawArc(r2,0,45,true,paint);
canvas.drawArc(r2,90,135,false,paint);
```

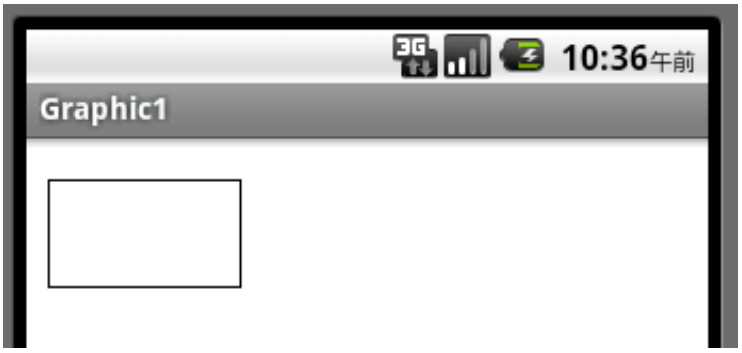


7.　キャンバス全体を塗る

　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。

「例」

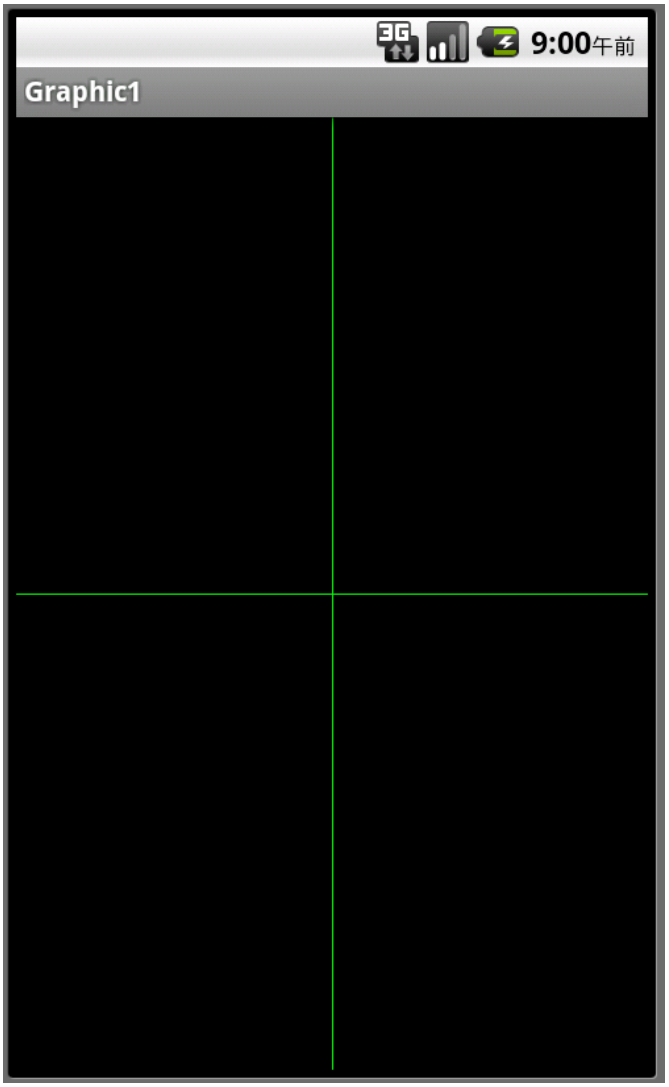
```
Paint paint=new Paint();
paint.setColor(Color.BLACK);
canvas.drawColor(Color.WHITE);
paint.setStyle(Style.STROKE);
RectF r=new RectF(15,30,150,105);
canvas.drawRect(r,paint);
```



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。

```
protected void onDraw(Canvas canvas) {
    Paint paint=new Paint();
    paint.setColor(Color.GREEN);
    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 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();
```

「注」以下のように WindowManager クラスを使って画面の幅と高さを取得することもできます。

```
int w=getWindowManager().getDefaultDisplay().getWidth();
int h=getWindowManager().getDefaultDisplay().getHeight();
```

「例題 23-1-2」花の各点のデータを直線でつなぎます。配列 p[ ]のデータは花の右半分のデータです。p[i]=80-p[i];とすることで左半分のデータを作ります。

・Graphic2.java

```
protected void onDraw(Canvas canvas) {
    Paint paint=new Paint();
    paint.setColor(Color.GREEN);
    float[] p={40,0,50,10,50,10,74,0,74,0,80,40,80,40,70,50,70,50,40,50,
            40,50,40,120,40,120,60,80,60,80,80,80,80,80,60,120,60,120,40,120};
    canvas.drawLines(p,paint); // 右半分
    for (int i=0;i<p.length;i+=2)
        p[i]=80-p[i];
    canvas.drawLines(p,paint); // 左半分
}
```

「参考」 下側に鏡像の花を描きます。

```
Paint paint=new Paint();
paint.setColor(Color.GREEN);
float[] p={40,0,50,10,50,10,74,0,74,0,80,40,80,40,70,50,70,50,40,50,
          40,50,40,120,40,120,60,80,60,80,80,80,80,80,60,120,60,120,40,120};
canvas.drawLines(p,paint);
for (int i=0;i<p.length;i+=2)
    p[i]=80-p[i];
canvas.drawLines(p,paint);
for (int i=1;i<p.length;i+=2)
    p[i]=240-p[i];
canvas.drawLines(p,paint);
for (int i=0;i<p.length;i+=2)
    p[i]=80-p[i];
canvas.drawLines(p,paint);
```

24 章　SurfaceView

SurfaceView は View のサブクラスです。SurfaceView はアプリケーションのスレッドと描画処理のスレッドが独立しているため、onDraw によらずにユーザが描画したいタイミングで描画処理を行うことができます。SurfaceView はダブルバッファリングという手法を用いて表と裏の画面を切り替えながら描画処理を行います。

「注」この章の例題を HVGA（320×480）で表示する場合は座標値やテキストサイズのピクセル値を「1/1.5」にしてください。たとえば例題 24-1 なら以下のように変更します。
「canvas.drawCircle(90,90,75,paint);」→「canvas.drawCircle(60,60,50,paint);」

24-1　SurfaceView の概要

SurfaceView に対し描画処理を行うための方法を説明します。

1.　SurfaceView と SurfaceHolder

サーフェスビューは SurfaceView クラスを継承して作ります。SurfaceView への描画には、SurfaceHolder というインターフェイスを利用し、描画処理は SurfaceHolder のコールバックとして実装します。「getHolder().addCallback(this);」でコールバックの登録をします。

```
class SView extends SurfaceView implements SurfaceHolder.Callback {
    public SView(Context context) {
        super(context);
        getHolder().addCallback(this); //  コールバックの登録
    }
```

SurfaceHolder.Callback インターフェースを実装することにより以下のメソッドを実装する必要があります。

SurfaceHolder.Callback の実装メソッド	機能
surfaceChanged	表示が変更された時の処理。
surfaceCreated	生成時の処理。
surfaceDestroyed	破棄時の処理。

2.　SurfaceView への描画

SurfaceView への描画は SurfaceHolder を介して行います。SurfaceHolder は画面を保持するための抽象インターフェースです。surfaceCreated メソッドの引数にこの SurfaceHolde オブジェクトの holder が渡されますので、

```
Canvas canvas=holder.lockCanvas();
```

でキャンバスを取得します。グラフィックス描画はこの canvas オブジェクトに対し行います。描画処理メソッドは 23 章で示した View に対するメソッドと全く同じです。この時点では描画結果は表示されません。描画処理終了後に「holder.unlockCanvasAndPost(canvas);」で取得していた holder を解除することで、実際の画面に描画が行われます。

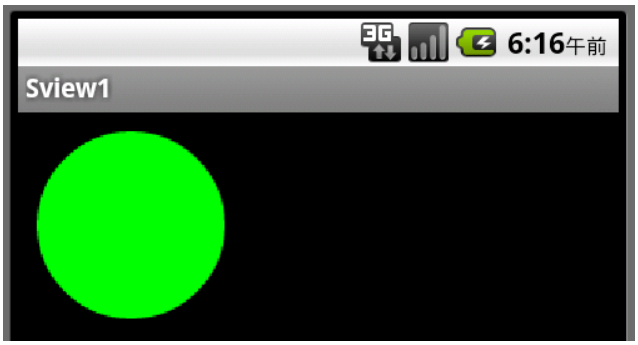
```
    public void surfaceCreated(SurfaceHolder holder) {
        Canvas canvas=holder.lockCanvas();
        // canvas への描画処理
        holder.unlockCanvasAndPost(canvas);
    }
```

「例題 24-1」SurfaceView に円を描きます。

・Sview1.java

```
package jp.sview1;

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

public class Sview1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new SView(this));
    }
    class SView extends SurfaceView implements SurfaceHolder.Callback {
        public SView(Context context) {
            super(context);
            getHolder().addCallback(this); //  コールバックの登録
        }
        public void surfaceChanged(SurfaceHolder holder,int format,int width, int
height) {
            // 表示が変更された時の処理
        }
        public void surfaceCreated(SurfaceHolder holder) {
            // 生成時の処理
            Canvas canvas=holder.lockCanvas();
            Paint paint=new Paint();
```

```
                paint.setColor(Color.GREEN);
                paint.setAntiAlias(true);
                canvas.drawCircle(90,90,75,paint);
                holder.unlockCanvasAndPost(canvas);
            }
            public void surfaceDestroyed(SurfaceHolder holder) {
                // 破棄時の処理
            }
        }
}
```



25 章　OpenGL

OpenGL は Silicon Graphics 社が中心となって開発し、現在は Khronos グループが策定している、グラフィックス処理のためのアプリケーションプログラミングインタフェース（API）です。OpenGL は 2 次元・3 次元コンピュータグラフィックス両方が扱えます。携帯端末向けオペレーティングシステムでは OpenGL のサブセットである OpenGL ES（OpenGL for Embedded Systems）が採用されています。

「注」 HVGA（320×480）ではビューポートの値を 1/1.5 に縮小します。つまり「gl.glViewport(0,0,450,450);」→「gl.glViewport(0,0,300,300);」に変更します。個々の座標データの値は変更しなくてよいです。

25-1　Renderer インターフェースのメソッド

Android から OpenGL を利用する場合、View として GLSurfaceView クラスの派生クラスを用います。GLSurfaceView は OpenGL のレンダリングが可能なビューを提供します。

GLSurfaceView.Renderer インターフェースを実装することで以下の 3 つのメソッドを実装します。描画処理は主に onDrawFrame に記述することになります。

Renderer インターフェースのメソッド	機能
onSurfaceCreated	初期化時に呼ばれます。
onSurfaceChanged	主に landscape と portrait の切り替えのときに呼ばれます。
onDrawFrame	描画時に呼ばれます。

1.　初期設定

OpenGL の品質を設定するには glHint メソッドを使います。target に対象、mode に品質を指定します。

glHint(int target, int mode)

target 定数	意味
GL_FOG_HINT	ファグの計算精度。
GL_LINE_SMOOTH_HINT	線のサンプリング精度。
GL_PERSPECTIVE_CORRECTION_HINT	カラーとテクスチャ座標の補間精度。
GL_POINT_SMOOTH_HINT	点のサンプリング精度。
GL_POLYGON_SMOOTH_HINT	ポリゴンのサンプリング精度。

mode 定数	意味
GL_FASTEST	効率的。
GL_NICEST	高品質。
GL_DONT_CARE	品質保証なし。

gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST); は「カラーとテクスチャ座標の補間精度」を「効率的」に設定しています。

2. 色の指定

背景色の設定は glClearColor メソッド、描画色は glColor4f メソッドで行います。それぞれ、red,green,blue,alpha の値を 0.0f~1.0f の範囲で指定します。

```
glClearColor(float red, float green, float blue, float alpha)
glColor4f(float red, float green, float blue, float alpha);
```

3. 画面クリア

各種バッファをクリアし、画面を glClearColor メソッドで設定した背景色で塗りつぶすには glClear メソッドを使います。mask にはクリアするバッファを指定します。mask の値は「|」で複数指定できます。

```
glClear(int mask);
```

Mask の値	意味
GL_COLOR_BUFFER_BIT	カラーバッファ。
GL_DEPTH_BUFFER_BIT	デプスバッファ。
GL_STENCIL_BUFFER_BIT	ステンシルバッファ。

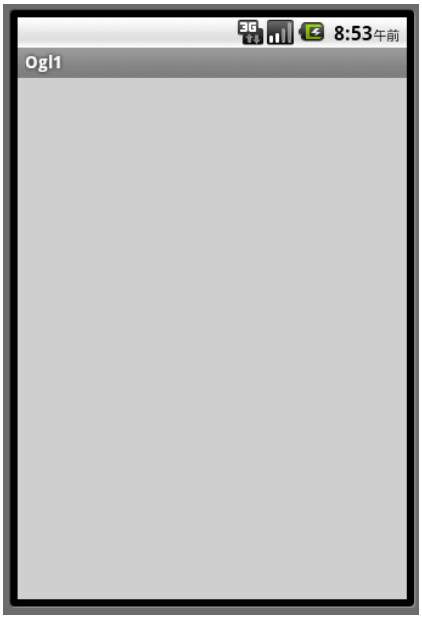
gl.glClear(GL10.GL_COLOR_BUFFER_BIT|GL10.GL_DEPTH_BUFFER_BIT); は「カラーバッファ」と「デプスバッファ」をクリアします。

「例題 25-1」背景色を灰色に設定します。

・Ogl1.java

```
package jp.Ogl1;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.opengl.GLSurfaceView;
import android.app.Activity;
import android.os.Bundle;

public class Ogl1 extends Activity {
    private GLSurfaceView gls;
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        gls=new GLSurfaceView(this);
        gls.setRenderer(new GLView());
        setContentView(gls);
    }
}
class GLView implements GLSurfaceView.Renderer {
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        //カラーとテクスチャ座標の補間精度
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
GL10.GL_FASTEST);
        //画面クリア色の指定
        gl.glClearColor(0.8f,0.8f,0.8f,1.0f);
    }
    public void onDrawFrame(GL10 gl){
        //画面クリアとバッファーのクリア
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT |
GL10.GL_DEPTH_BUFFER_BIT);
    }
    public void onSurfaceChanged(GL10 gl, int width, int height) {
    }
}
```



26 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 フォルダまたは raw 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。データを、キー名と値の組み合わせでファイルに保存しておき、キー名から値を取得するプリファレンス・ファイルについて説明します。独自のコンテンツプロバイダを作り、このコンテンツプロバイダ経由でファイルにアクセスする方法を説明します。画面をキャプチャーしたイメージをファイルとして保存する方法を説明します。

26-1　ファイルへの読み書き

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。どちらも FileInputStream クラスと FileOutputStream クラスのメソッドを使ってファイルオープンします。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。テキスト・ストリームは BufferedReader と PrintWriter クラスの println/readLine メソッドを使って行単位の読み書きを行います。

なお、ファイル処理は例外を伴うので必ず try catch 文で囲みます。

1.　バイト単位の読み書き（FileInputStream/FileOutputStream）

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();
```

「参考」以下はファイル a.dat の内容をファイル b.dat にコピーするものです。

```
FileInputStream in=openFileInput("a.dat");
OutputStream out=openFileOutput("b.dat",MODE_PRIVATE);
int d;
while ((d=in.read())!=-1){
    out.write(d);
}
in.close();
out.close();
```

2. 行単位の読み書き(BufferedReader/PrintWriter)

バイナリ・ストリームは速くて効率的ですが、行単位のテキストを扱うには不向きです。そこでファイルを行単位で管理するテキスト・ストリームを使います。テキスト・ストリームでのファイルオープンの方法はバイナリー・ストリームと同じです。

テキスト・ファイルに 1 行のテキストを書くには PrintWriter クラスの println メソッドを使います。

```
OutputStream out=openFileOutput("test.dat",MODE_PRIVATE);
PrintWriter pw=new PrintWriter(new OutputStreamWriter(out));
pw.println("Candy,21");
pw.println("Lisa,19");
pw.close();
out.close();
```

これでファイル test.dat には「Candy,21¥nLisa,19¥n」が書きこまれます。「¥n」は改行コードです。

テキスト・ファイルから 1 行を読むには BufferedReader クラスの readLine メソッドを使います。ファイルの終わりで readLine は null を返します。readLine で読み込んだデータには「¥n」は含まれませんので、改行を行いたい場合はユーザが"¥n"を加えます。

```
FileInputStream in=openFileInput("test.dat");
BufferedReader br=new BufferedReader(new InputStreamReader(in));
```

```
String txt="",s="";
while ((s=br.readLine())!=null){
    txt+=s+"¥n";
}
br.close();
in.close();
```

「補足」FileReader は InputStreamReader のサブクラスで、InputStreamReader より簡単に扱えますが、文字コードには常に省略時文字コードが用いられます。

```
FileInputStream in=openFileInput("test.dat");
BufferedReader br=new BufferedReader(new InputStreamReader(in));
```

と

```
FileReader in=new FileReader("test.dat");
BufferedReader br=new BufferedReader(in);
```

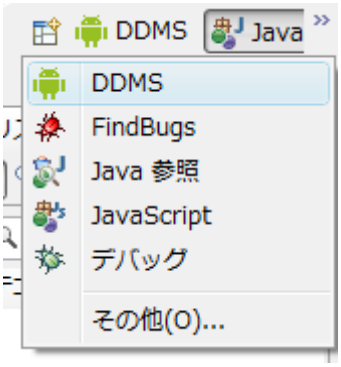
はほぼ等価ですが、ファイル権限の扱いが異なります。上のプログラムを FileReader で行うと「/test.dat(No such file or directory」というエラーになります。逆に SD カードに保存したテキストファイルを FileInputStream で読むと強制終了してしまいますので、この場合は FileReader を使います。FileReader の例は「例題 26-6-2」を参照してください。

3. ファイルの保存場所

openFileOutput や openFileInput メソッドは Activity の Context クラスのメソッドで、Android であらかじめ用意された保存可能な場所にファイルを置いてくれます。具体的には openFileOutput メソッドでファイルを作成すると、「/data/data/パッケージ名/files/ファイル名」に保存されます。

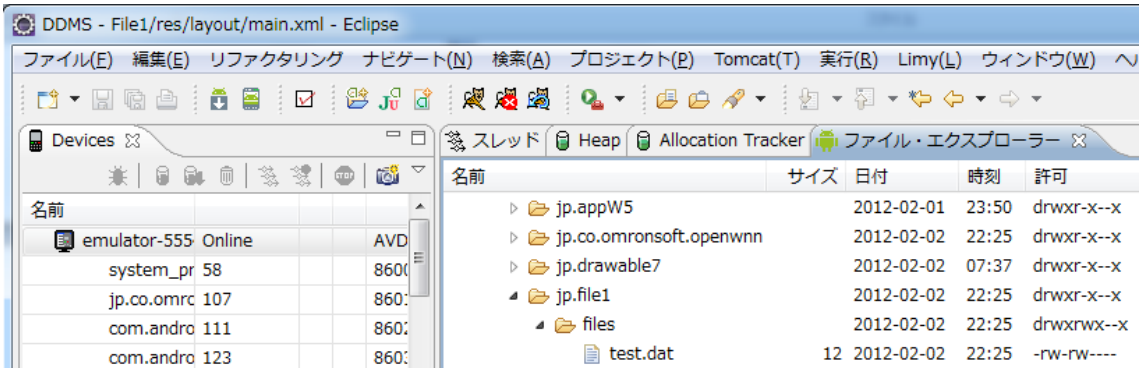
普通のプロジェクトエクスプローラやパッケージエクスプローラではこのフォルダの内容を確認することができません。しかし DDMS を使って確認することができます。

①DDMS を開く（エミュレータが実行中であること）。

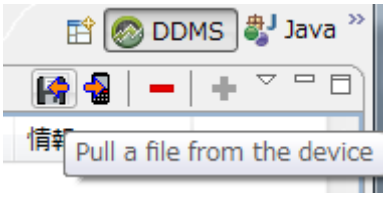


②「Devices」の「emulator-XXX」を選択。

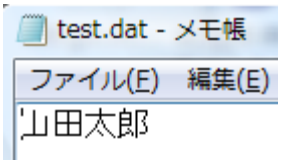
③「ファイル・エクスプローラー」タブを選択し「data」－「data」－「jp.file1」－「files」フォルダを開く。



④ファイルの内容を確認するには「ファイル・エクスプローラ」タブの右にある  「Pull a file from device」を選択し、ファイルをローカルディスクの適当な場所（たとえばデスクトップなど）に移す。



⑤メモ帳などで内容を見る



「例題 26-1」「書き込み」ボタンのクリックで EditText に入力されているテキストをファイル名が「test.dat」のファイルに書き込みます。「読み込み」ボタンのクリックでファイルの内容を読み出し EditText に表示します。

・main.xml

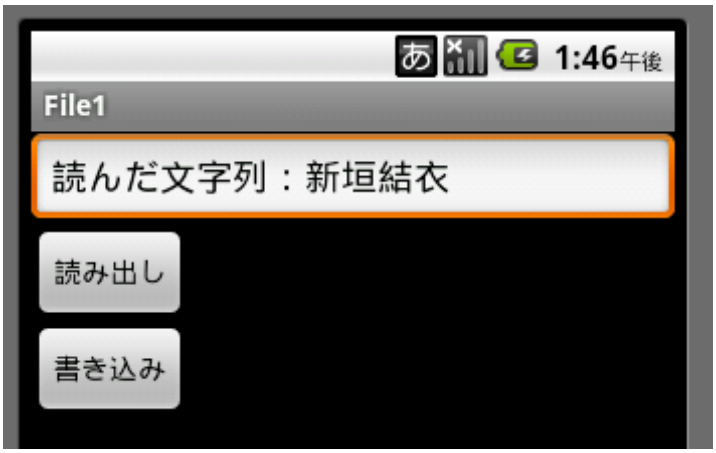
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <EditText
        android:id="@+id/text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
    <Button
        android:id="@+id/read"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="読み出し"
    />
    <Button
        android:id="@+id/write"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="書き込み"
    />
</LinearLayout>
```

・File1.java

```
package jp.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 File1 extends Activity {
    private EditText text;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        text=(EditText)findViewById(R.id.text);
        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);
                text.setText("読んだ文字列："+ 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);
                    String msg=text.getText().toString();
                    out.write(msg.getBytes());
                    out.close();
                } catch (IOException e) {}
            }
        }
}
```



27 章　SQLite

SQLite はデータベース管理システム（DBMS）のひとつですが、クライアント・サーバー型の本格的な DBMS とは異なり、DBMS サーバーの概念が存在せずデータベースの内容はすべてローカルファイルに保存される方式の簡易 DBMS です。Android では標準で SQLite を組み込んでいて、Android が用意している API を使用して、アプリケーションから利用することができます。Android の SQLite の場合、データベースはデータベースを作成したアプリケーション専用です。

27-1　データベースの作成

データーベース akb.db を作成しデータを表示する例を通して SQLite の処理方法を説明します。

1.　SQL とは

SQL(Structured Query Language)はデータベースの操作を行うための言語の一つです。データベースに対する処理要求(問い合わせ)を文字列として表したものをクエリー（Query）と呼びます。クエリーとしてレコードの抽出を行う SELECT 文、テーブルにレコードを挿入する INSERT 文、レコードを削除する DELETE 文などがあります。

2.　テーブルの構成

ここで作成するデータベースのテーブルはテーブル名を"akb_table"とし、自動インクリメントされる"id"、キーを示す"topic"、内容を示す"memo"の 3 項目からなるものとします。

Android の SQLite で使用できるデータ型は以下の 5 種類です。"id"は integer 型、"topic"と"memo"は text 型とします。

integer・・・符号付整数
real・・・浮動小数点
text・・・テキスト
blob・・・バイナリデータ
null・・・NULL

3.　DatabaseHelper クラスの作成

SQLiteOpenHelper クラスを継承した次のような DatabaseHelper クラスを作成します。DatabaseHelper コンストラクタでデータベース名とデータベースのバージョンを設定します。onCreate メソッドが呼ばれるとテーブルがなければテーブルを作成します。

onUpgrade メソッドが呼ばれるとテーブルがすでに存在すれば削除し、新規に作成します。execSQL メソッドでテーブルの生成と削除を行っています。

```
public class DatabaseHelper extends SQLiteOpenHelper {
    public DatabaseHelper(Context context) {
        super(context,"データベース名" , null,データベースのバージョン);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
```

```
            db.execSQL(
                "create table if not exists テーブル名 (" +
                " id integer primary key autoincrement," +
                " 項目 型,"+
                " 項目 型)");
        }
        @Override
        public void onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion) {
            db.execSQL("drop table if exists テーブル名");
            onCreate(db);
        }
    }
```

4. データベースの作成

DatabaseHelper クラスの helper オブジェクトを生成し、helper の getWritableDatabase メソッドを使って書き込み用データベースオブジェクト db を作成します。

```
helper=new DatabaseHelper(this);
db=helper.getWritableDatabase();
//データベースへの処理
db.close();
```

データベースへのデータの追加は insert メソッドを使います。追加するデータは ContentValues 型のオブジェクト cv に put メソッドを使って設定します。id 項目は自動インクリメントを指定しているので、データとして put しません。

```
ContentValues cv=new ContentValues();
cv.put("topic",topic のデータ);
cv.put("memo",memo のデータ);
db.insert("akb_table",null,cv);
```

5. Cursor

Cursor はデータベースのテーブルの各行を取得するためのオブジェクトです。Cursor オブジェクトを取得するには、query メソッドまたは rawQuery メソッドの第 1 引数にテーブル名、第 2 引数に行の構成要素を示す文字列配列を指定します。

```
Cursor c=db.query("akb_table",new String[]
{"id","topic","memo"},null,null,null,null,null);
```

カーソルを操作する事で、1 行ずつデータを取得します。moveToFirst メソッドで最初の行に移動して、moveToNext メソッドで次の行に移動します。moveToFirst メソッドや moveToNext メソッドなどの行移動メソッドは、移動する行が無い場合は false を返します。

6　データの読み出し

データの読み出しをするには、db に「helper.getReadableDatabase()」を取得し、読み出すデータの型に応じて getInt または getString メソッドを使います。これらの引数は取得する項目の番号（0 スタート）です。

```
db=helper.getReadableDatabase();
Cursor c = db.query("akb_table", new String[] {"id","topic","memo"},null, null, null,
null, null);
while (c.moveToNext()){
    c.getInt(0)で id 項目の取得
    c.getString(1)で topic 項目の取得
    c.getString(2)で memo 項目の取得
}
c.close();
db.close();
```

7.　データベースの保存場所

データベースの作成場所をストレージにするか、メモリにするかを選択できます。ストレージに作成するかメモリに作成するかは、DatabaseHelper クラスのコンストラクタの第 2 引数で指定します。ここが null ならメモリ上に保存されます。

```
public DatabaseHelper(Context context) {
    super(context, null, null, 1);
}
```

ファイル名を指定した場合は「/data/data/<パッケージ名>/database/<ファイル名>」に、データベースファイルが作成されます。指定するのはファイル名だけです。

```
public DatabaseHelper(Context context) {
    super(context,"akb.db",null,1);
}
```

8. データーベースの削除

アプリケーションをアンインストールするとファイルとして作成したデータベースも自動的に削除されます。

「例題 27-1」AKB に関するデータベースを作成し、ボタンのクリックでデータを読み出します。実行するたびに、同じレコードが追加されないように「if (!c.moveToFirst())」で判定しています。データがない新規の状態で「c.moveToFirst()」は false を返します。すでにデータがある状態で追加しようとする場合「c.moveToFirst()」はテーブルの先頭ではなくデータの追加先頭位置への移動になります。

・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"
    >
    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="データベースの表示"
    />
    <TextView
        android:id="@+id/text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

・DatabaseHelper.java

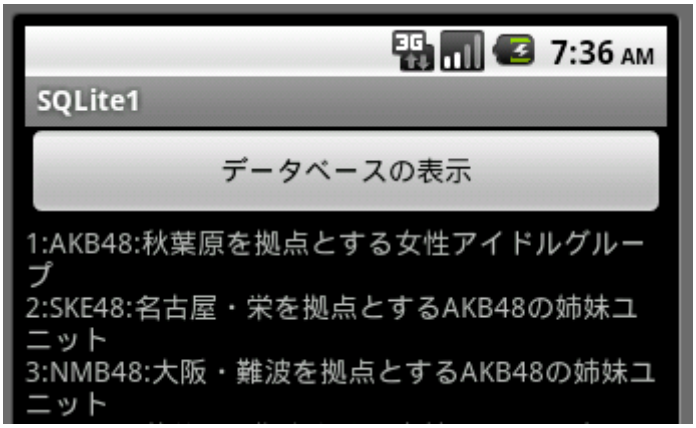
```
package jp.sqlite1;

import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.content.Context;

public class DatabaseHelper extends SQLiteOpenHelper {
	public DatabaseHelper(Context context) {
		super(context,"akb.db", null,1);
	}
	@Override
	public void onCreate(SQLiteDatabase db) {
		db.execSQL(
			"create table if not exists akb_table (" +
			" id integer primary key autoincrement," +
			" topic text,"+
			" memo text )");
	}
	@Override
	public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion) {
		db.execSQL("drop table if exists akb_table");
		onCreate(db);
	}
}
```

・SQLite.java

```
package jp.sqlite1;

import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
```

```
import android.view.View.OnClickListener;
import android.widget.*;

public class SQLite1 extends Activity {
	private SQLiteDatabase db;
	private DatabaseHelper helper;
	private TextView tv;
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		tv=(TextView)findViewById(R.id.text);
		Button bt1=(Button)findViewById(R.id.button1);
		bt1.setOnClickListener(new Disp());
		String key[]={"AKB48","SKE48","NMB48"};
		String value[]={
			"秋葉原を拠点とする女性アイドルグループ",
			"名古屋・栄を拠点とする AKB48 の姉妹ユニット",
			"大阪・難波を拠点とする AKB48 の姉妹ユニット"};
		helper=new DatabaseHelper(this);
		db=helper.getWritableDatabase();
		Cursor c=db.query("akb_table",new String[]
{"id","topic","memo"},null,null,null,null,null);
		if (!c.moveToFirst()){
			for (int i=0;i<key.length;i++){
				ContentValues cv=new ContentValues();
				cv.put("topic",key[i]);
				cv.put("memo",value[i]);
				db.insert("akb_table",null,cv);
			}
		}
		db.close();
	}
	class Disp implements OnClickListener {
		public void onClick(View v) {
			db=helper.getReadableDatabase();
```

```
                Cursor c=db.query("akb_table",new String[]
{"id","topic","memo"},null,null,null,null,null);
                String s="";
                while (c.moveToNext()){
                    s+=c.getInt(0)+":"+c.getString(1)+":"+c.getString(2)+"¥n";
                }
                tv.setText(s);
                c.close();
                db.close();
            }
        }
}
```



「補足」すでにテーブルがある場合は全データを削除して新たにデータを追加するには以下のようにします。ただし id の値は 1 にクリアされずに増加していきます。

```
    if (c.moveToFirst())
          db.delete("akb_table", null, null);
    for (int i=0;i<key.length;i++){
        ContentValues cv=new ContentValues();
        cv.put("topic",key[i]);
        cv.put("memo",value[i]);
        db.insert("akb_table",null,cv);
    }
```

28 章　Gmail(実機のみ)

Android では次の 3 つのメールが使用できます。

(1)Gmail
(2)キャリアメール(SMS/MMS)
(3)E-mail(PC メール)

キャリアメールは docomo の場合は「sp モードメール」、SoftBank の場合は「S!メール」、au の場合は「C メール」となります。Gmail は、Google 社によるフリーメールサービスで、Android では各機種で共通です。

インテントを使ったメール送信の方法は「中級　Android 的プログラミング法」の「10 章　インテントとアクティビティ」で説明しました。この章では Gmail を受信した時の各種処理方法を説明します。

28-1　Gmail の受信状況

Gmail の受信状況を調べるには、query メソッドの第 1 引数のコンテンツプロバイダ URI に「"content://gmail-ls/conversations/XXXXXXX@gmail.com"」を指定し、第 3 引数にメールボックスの種別を指定します。コンテンツプロバイダについては「中級　Android 的プログラミング法」の「14 章　コンテンツプロバイダ」を参照してください。

```
Cursor c=getContentResolver().query(
    Uri.parse("content://gmail-ls/conversations/XXXXXXX@gmail.com"),
    null,
    "label:^i",
    null,
    null);
```

メールボックスの種別として以下があります。

メールボックスの種別	意味
"^f"	SENT
"^i"	INBOX
"^r	DRAFT
"^u"	UNREAD
"^k"	TRASH
"^s"	SPAM
"^t"	STARRED
"^b"	CHAT（BUZZ）
"^vm"	VOICEMAIL
"^g"	IGNORED
"^all"	ALL
"^^vmi"	VOICEMAIL_INBOX
"^^cached"	CACHED
"^^out"	OUTBOX

Gmail の受信ボックスのカーソルデータ c からデータ（たとえば件名）を取得するには以下のようにします。

```
String subject=c.getString(c.getColumnIndex("subject"));
```

getColumnIndex の引数に指定できる主なキーとして以下があります。この他に labelIds、numMessages、maxMessageId、hasAttachments、hasMessagesWithErrors、forceAllUnreadl などがあります。このカーソルでは受信メッセージの本文は取得できません。「28-2　受信メッセージの本文の取得」を参照してください。

キー	意味
_id	メッセージ ID。
subject	件名。
snippet	本文の要約。
date	受信日時。
fromAddress	送信者アドレス。
personalLevel	アドレス帳のグループ。

「例題 28-1」画面のタッチで Gmail 受信メッセージの一覧（件名、受信日時、要約）をトーストで表示します。

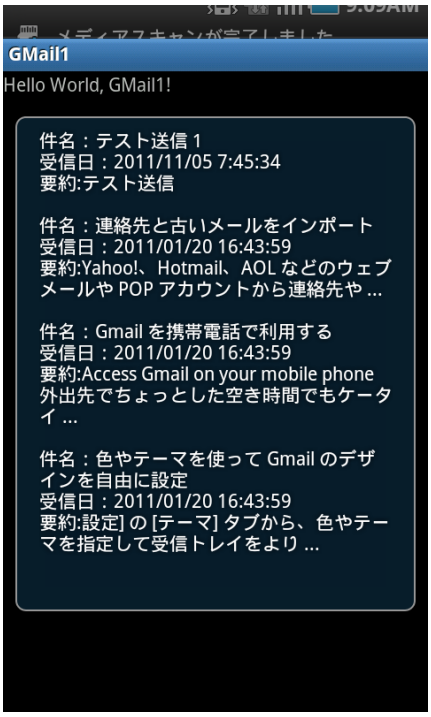
・マニフェスト（AndroidManifest.xml）

```
<uses-permission android:name="com.google.android.gm.permission.READ_GMAIL"/>
```

・GMail1.java

```
package jp.gmail1;

import java.util.Date;
import android.app.Activity;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.*;
import android.widget.Toast;

public class GMail1 extends Activity {
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            try {
                Cursor
c=getContentResolver().query(Uri.parse("content://gmail-ls/conversations/akasai123@g
mail.com"),null,"label:^i",null,null);
                startManagingCursor(c);
                String msg="";
                while (c.moveToNext()){
                    String subject=c.getString(c.getColumnIndex("subject"));
                    String date=c.getString(c.getColumnIndex("date"));
                    Date d=new Date(Long.valueOf(date));
                    String snippet=c.getString(c.getColumnIndex("snippet"));
                    msg+="件名："+subject+"¥n 受信日："+d.toLocaleString()+"¥n 要
約:"+snippet+"¥n¥n";
                }
                Toast.makeText(this,msg,Toast.LENGTH_LONG).show();
            }
            catch (Exception e){}
        }
        return super.onTouchEvent(event);
    }
}
```



「補足」Date オブジェクト d を文字列に変換する方法として以下の 3 種類があります。
d.toString()は「2011-07-20」
d.toLocaleString()は「2011/07/20 14:45:37」
d.toGMTString()は「20 Jul 2011 05-45-37 GMT」。これは日本時間から「-9」した標準時間。

29 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

29-1　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 コマンドで、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

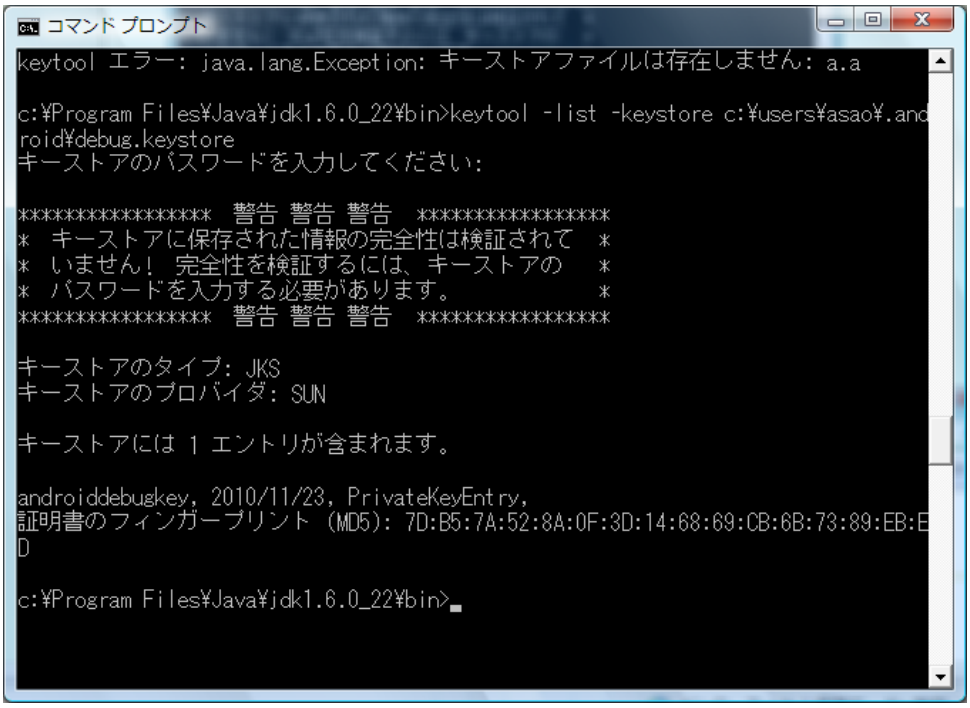
「注」XXX は OS により以下のようになります。

WindowsXP⇒C:¥Documents and Settings¥ユーザ名
WindowsVista/7⇒ C:¥Users¥ユーザ名
Mac/Linux⇒~/.android/debug.keystore

「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。
C:¥Program Files¥Java¥jdk1.6.0_07¥bin

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「7D:B5:7A:52:8A:0F:3D:14:68:69:CB:6B:73:89:EB:ED」）を Maps API Key の取得の際に使います。

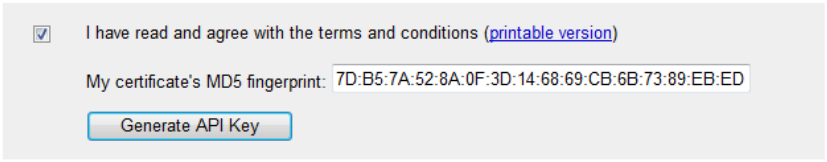


②Maps API Key の取得

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。

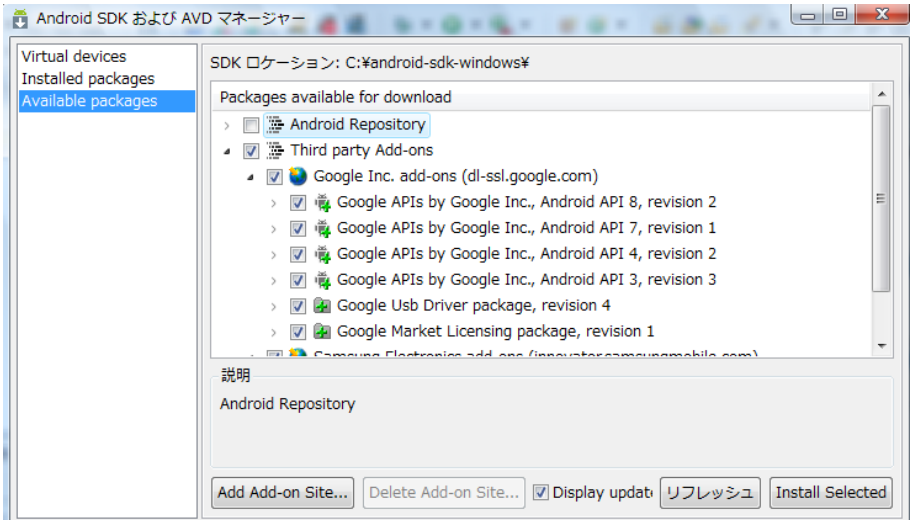
以下の MapView を示す XML テキストが得られますので、これを main.xml に指定します。

```
<com.google.android.maps.MapView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:apiKey="0O4FkQeWz-4XtxKCnntbnqHMMVUwaT-GQjEsK4g"
/>
```

「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

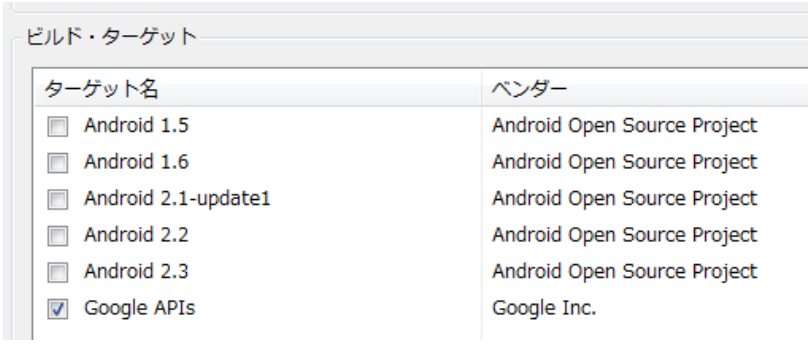
Eclipse の「ウインドウ」－「Android　SDK および AVD マネージャー」を選択します。「Available packages」を選択し「Third party Add-ons」をチェックし、「Google APIs」関連をチェックしインストールします。



3.　Map プロジェクトの作成

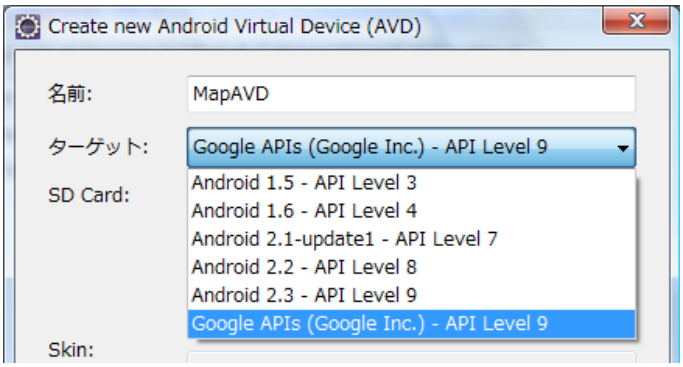
Map アプリは「MapActivity」クラスを継承します。Map の表示は「MapView」に対して行います。以下の①～⑤の手順でプロジェクトを作成します。

①ビルド・ターゲットに「Google APIs」を選択



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。



③マニフェストにライブラリとパーミションを設定

マニフェストに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>
```

④main.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。

・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-4XtxKCnntbnqHMMVUwaT-GQjEsK4g"
    />
</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、main.xml を「setContentView(R.layout.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。

・Map1.java

```
package jp.map1;

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

public class Map1 extends MapActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    protected boolean isRouteDisplayed() {
        return false;
```

```
    }
}
```

「注」このプログラムでは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。

・エミュレータ

・実機

30 章　センサー（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。この章ではセンサーの各種使用方法を説明します。

30-1　センサーの種類

Android がサポートするセンサーとして以下があります。使用できるセンサーは実機ごとに何をサポートするか異なります。

センサーを示す定数	値	センサーの種類
TYPE_ACCELEROMETER	1	加速度センサ。
TYPE_MAGNETIC_FIELD	2	磁界（磁気）センサ。
TYPE_ORIENTATION	3	方位センサ。
TYPE_GYROSCOPE	4	ジャイロセンサ。
TYPE_LIGHT	5	輝度（照度）センサ。
TYPE_PRESSURE	6	圧力センサ。
TYPE_TEMPERATURE	7	温度センサ。
TYPE_PROXIMITY	8	近接センサ。

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。使用したいセンサーを getSensorList メソッドを使ってセンサーマネージャーから取得します。ここでは複数のセンサーを登録し、使用可能なセンサーの種類を調べるため、各センサーオブジェクトを sensors リストに追加していきます。

```
SensorManager sm=(SensorManager)getSystemService(SENSOR_SERVICE);
ArrayList<List<Sensor>> sensors=new ArrayList<List<Sensor>>();
sensors.add(sm.getSensorList(Sensor.TYPE_ACCELEROMETER));
```

同様に 8 種類のセンサーを sensors に追加したら、以下でセンサーの名前とタイプ（上の表の定数の値）を取得します。

```
for(List<Sensor> sensor：sensors){
    if (sensor.size()>0){
        sensor.get(0).getName() // センサーの名前
        sensor.get(0).getType() // センサーのタイプ
    }
}
```

「例題 30-1」使用できるセンサーを調べます。センサーの名前とタイプを TextView に表示します。

・main.xml

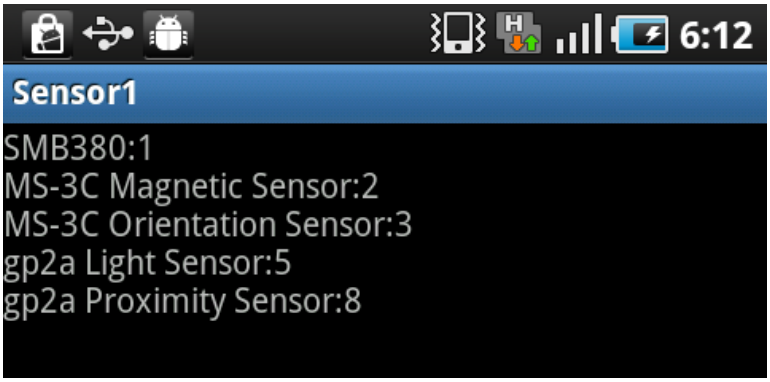
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/text"
        android:layout_width="fill_parent"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

・Sensor1.java

```
package jp.sensor1;

import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.hardware.*;
import android.os.Bundle;
import android.widget.TextView;

public class Sensor1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv=(TextView)findViewById(R.id.text);
        SensorManager sm=(SensorManager)getSystemService(SENSOR_SERVICE);
        ArrayList<List<Sensor>> sensors=new ArrayList<List<Sensor>>();
        sensors.add(sm.getSensorList(Sensor.TYPE_ACCELEROMETER));
```

```
            sensors.add(sm.getSensorList(Sensor.TYPE_MAGNETIC_FIELD));
            sensors.add(sm.getSensorList(Sensor.TYPE_ORIENTATION));
            sensors.add(sm.getSensorList(Sensor.TYPE_GYROSCOPE));
            sensors.add(sm.getSensorList(Sensor.TYPE_LIGHT));
            sensors.add(sm.getSensorList(Sensor.TYPE_PRESSURE));
            sensors.add(sm.getSensorList(Sensor.TYPE_TEMPERATURE));
            sensors.add(sm.getSensorList(Sensor.TYPE_PROXIMITY));
            String txt="";
            for(List<Sensor> sensor：sensors){
                if (sensor.size()>0)
                    txt+=sensor.get(0).getName()+":"+sensor.get(0).getType()+"¥n";
            }
            tv.setText(txt);
        }
}
```

「実行結果」GALAXY では以下のようになりました。加速度センサ、磁界（磁気）センサ、方位センサ、輝度（照度）センサ、近接センサをサポートし、ジャイロセンサ、圧力センサ、温度センサをサポートしないことがわかります。



31 章　カメラ（実機のみ）

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をまず説明します。

撮影した写真に日付をプリントしたり、撮影画像の上にイメージや手書きの絵を書き入れるといったオーバーレイ機能についても説明します。

31-1　カメラの映像をプレビュー表示し、シャッターを切る

カメラの映像をプレビュー表示し、シャッターを切るまでを説明します。写真はまだ保存できません。

1.　カメラ用のビュークラス

カメラのプレビュー画面は SurfaceView に表示するため次のようなカメラ用のビュークラスを定義します。

```
class CameraView extends SurfaceView implements
SurfaceHolder.Callback,Camera.PictureCallback {

}
```

サーフェスビューを管理するサーフェスフォルダを取得し、タイプにプッシュバッファ（SURFACE_TYPE_PUSH_BUFFERS）を設定します。

```
holder=getHolder();
holder.addCallback(this);
holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
```

2.　カメラのオープン

カメラをオープンし、プレビュー画面に上で取得したサーフェスフォルダを設定します。

```
camera=Camera.open();
camera.setPreviewDisplay(holder);
```

3.　プレビューとシャッター

プレビューの開始は以下で行います。

```
camera.startPreview();
```

シャッターを切るには次のように行います。シャッターを切ると自動的にシャッター音が出ます。

```
camera.takePicture(null,null,this);
```

takePicture メソッドの実行で onPictureTaken メソッドが呼び出されます。

```
public void onPictureTaken(byte[] data, Camera camera) {
    // シャッターを切ったときの処理
}
```

「注」Android 4.0 ではシャッター音はデフォルトではなりません。

4. カメラリソースの解放

カメラは他のアプリでも使用する共通のリソースです。ユーザが作成したアプリでカメラリソースに何らかの障害を与えると他のアプリで使えなくなる危険があります。アプリを終了する際にはきちんとカメラリソースを解放しておかなければなりません。

```
camera.setPreviewCallback(null);
camera.stopPreview();
camera.release();
camera=null;
```

5. マニフェスト（AndroidManifest.xml）

カメラを使用するには以下のパーミションをマニフェストに追加します。

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

また、カメラは横向きで使用するのが普通なのでマニフェストなら「android:screenOrientation="landscape"」を指定し、Java コードなら「setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);」を指定します。

「例題 31-1」カメラの映像をプレビュー表示し、画面のタッチでシャッターを切ります。

・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.CAMERA" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
    <application android:icon="@drawable/ic_launcher"
android:label="@string/app_name">
        ・
        ・
    </application>
</manifest>
```

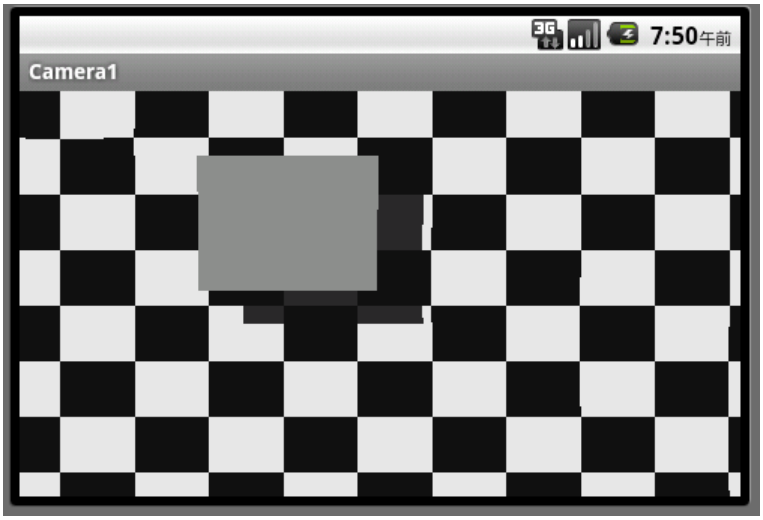
・Camera1.java

```
package jp.camera1;

import android.app.Activity;
import android.content.Context;
import android.content.pm.ActivityInfo;
import android.hardware.Camera;
import android.os.Bundle;
import android.view.*;

public class Camera1 extends Activity {
    private SurfaceHolder holder;
    private Camera camera;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRequestedOrientation(
ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE); //  横向き
        setContentView(new CameraView(this));
    }
    class CameraView extends SurfaceView implements
SurfaceHolder.Callback,Camera.PictureCallback {
        public CameraView(Context context) {
            super(context);
            holder=getHolder();
```

```
            holder.addCallback(this);
            holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        }
        public void surfaceChanged(SurfaceHolder holder,int format,int width, int
height) {
            camera.startPreview();
        }
        public void surfaceCreated(SurfaceHolder holder) {
            try {
                camera=Camera.open();
                camera.setPreviewDisplay(holder);
            }
            catch (Exception e){}
        }
        public void surfaceDestroyed(SurfaceHolder holder) {
            camera.setPreviewCallback(null);
            camera.stopPreview();
            camera.release();
            camera=null;
        }
        public void onPictureTaken(byte[] data, Camera camera) {
            setTitle("撮影");
            camera.startPreview();
        }
        public boolean onTouchEvent(MotionEvent event) {
            if (event.getAction()==MotionEvent.ACTION_DOWN){
                camera.takePicture(null,null,this);
            }
            return super.onTouchEvent(event);
        }
    }
}
```

・実機

・エミュレータ



32 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。

32-1　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です。 以下のような文字列を指定します。 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。

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！");
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。

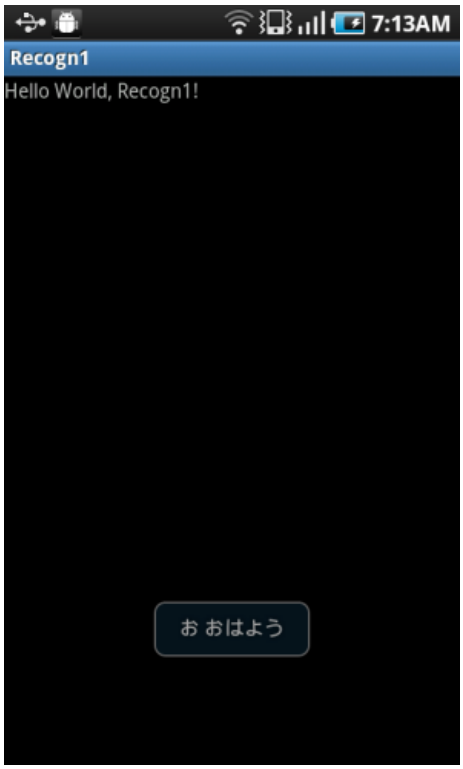
```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode==ICode && resultCode==RESULT_OK) {
        String msg="";
        ArrayList<String> results =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
        for (int i=0;i<results.size();i++)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

「例題 32-1」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。

・Recogn1.java

```
package jp.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 Recogn1 extends Activity {
    private int ICode=0; // インテント ID
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.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！");
                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);
        }
}
```



33 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はサポートされていません。そこで簡易的に日本語で読み上げを行う方法を説明します。

音声合成はエミュレータ上でも確認できます。

33-1　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。機種によっては Locale.JAPANESE は未サポートです。

```
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 メソッドを使います。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。

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

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（サイレンス）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 33-1」「Good morning」を英語で読み上げます。

・Speech1.java

```
package jp.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 Speech1 extends Activity implements OnInitListener{
    private TextToSpeech ts;
    @Override
```

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

34 章　ネットワーク通信

ネットワーク通信を行うための主な方法として HTTP とソケットがあります。HTTP（HyperText Transfer Protocol）は、Web のサーバと、クライアント（ブラウザ）の間で、ウェブページを送受信するためのプロトコルです。ソケットは自分のコンピュータと相手のコンピュータをソケットで接続し双方向でデータの受発信を行います。これらの機能は Android 特有のものではなく Java の「java.net」パッケージで提供される一般機能です。

Android 端末のネットワーク接続が現在 3G なのか、Wi-Fi なのかを調べたり、接続状態が ON なのか OFF なのかを調べたりするクラスとして ConnecttivityManager があります。この章ではこれらの機能について説明します。

「注」HTTP と Socket は Android4.0 では動作しません。原因は不明です。

34-1　ConnecttivityManager

ConnecttivityManager はネットワーク接続の状態を監視するためのクラスです。Android 端末のネットワーク接続が現在 3G（電話回線の 3rd Generation）なのか、Wi-Fi（インターネットの無線回線）なのかを調べたり、接続状態が ON なのか OFF なのかを調べたりすることができます。

1.　ネットワーク情報の取得

ConnecttivityManager オブジェクトの cm は以下のように生成し、この cm から getAllNetworkInfo メソッドを使って NetworkInfo クラスのネットワーク情報を info に取得します。

```
ConnectivityManager
cm=(ConnectivityManager)getSystemService(CONNECTIVITY_SERVICE);
NetworkInfo[] info=cm.getAllNetworkInfo();
```

NetworkInfo クラスの以下のメソッドを使って各種情報を取得できます。

NetworkInfo クラスのメソッド	機能
getType	ネットワークのタイプを定数（TYPE_MOBIL、TYPE_WIFI）で取得します。
getTypeName	ネットワークのタイプの名前を文字列（mobile、WIFI）で取得します。
getSubtypeName	ネットワークのサブタイプの名前を文字列（HSDPA など）で取得します。
isAvailable	ネットワーク接続が可能かどうかを true/false で取得します。
isConnected	ネットワーク接続されているかどうかを true/false で取得します。

「例題 34-1-1」Android 端末のネットワーク情報を TextView に表示します。

・マニフェスト（AndroidManifest.xml）

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

・main.xml

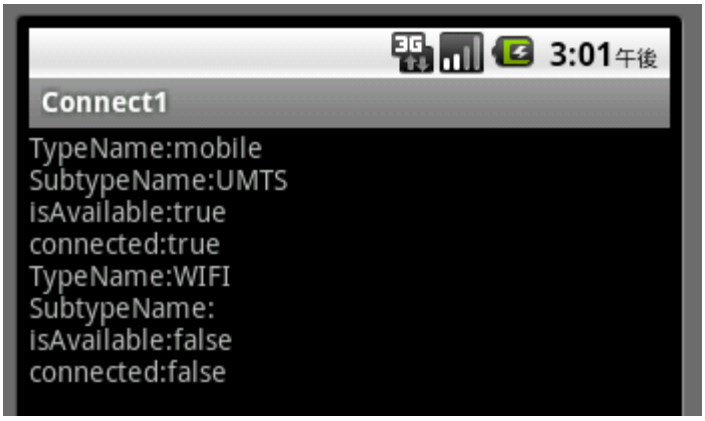
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
</LinearLayout>
```

・Connect1.java

```
package jp.connect1;

import android.app.Activity;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.widget.TextView;

public class Connect1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ConnectivityManager
cm=(ConnectivityManager)getSystemService(CONNECTIVITY_SERVICE);
        NetworkInfo[] info=cm.getAllNetworkInfo();
        String msg="";
        for(NetworkInfo i: info) {
            if (i.getType()==ConnectivityManager.TYPE_WIFI    | |
                i.getType()==ConnectivityManager.TYPE_MOBILE) {
                msg+="TypeName:"+i.getTypeName()+"¥n";
                msg+="SubtypeName:"+i.getSubtypeName()+"¥n";
                msg+="isAvailable:"+i.isAvailable()+"¥n";
```

```
                msg+="connected:"+i.isConnected()+"¥n";
            }
        }
        TextView tv=(TextView)findViewById(R.id.text);
        tv.setText(msg);
    }
}
```



2. ブロードキャストインテント

ネットワーク状態が変化した時にブロードキャストインテント（CONNECTIVITY_ACTION）が発行されますので、インテントフィルタに「android.net.conn.CONNECTIVITY_CHANGE」を指定し、BroadcastReceiver クラスの onReceive メソッドで処理を行うことができます。受信したインテントからネットワーク情報を取得するには getParcelableExtra メソッドを使います。

```
    public void onReceive(Context context, Intent intent) {
        String action=intent.getAction();
        if (action.equals(ConnectivityManager.CONNECTIVITY_ACTION)) {
            NetworkInfo
info=(NetworkInfo)intent.getParcelableExtra(ConnectivityManager.EXTRA_NETWOR
K_INFO);
            // 処理
        }
    }
```

「例題 34-1-2」ネットワーク状態の変化をブロードキャストレシーバ ConnectReceiver で受信し、情報を Toast で表示します。

・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.ACCESS_NETWORK_STATE" />
    <application android:icon="@drawable/ic_launcher"
android:label="@string/app_name">
          ・
          ・
        <receiver android:name=".ConnectReceiver">
            <intent-filter>
                <action
android:name="android.net.conn.CONNECTIVITY_CHANGE" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

・Connect2.java

```
package jp.connect2;

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

public class Connect2 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

・ConnectReceiver.java

```
package jp.connect2;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.widget.Toast;

public class ConnectReceiver extends BroadcastReceiver{
	@Override
	public void onReceive(Context context, Intent intent) {
		String action=intent.getAction();
		if (action.equals(ConnectivityManager.CONNECTIVITY_ACTION)) {
			NetworkInfo
info=(NetworkInfo)intent.getParcelableExtra(ConnectivityManager.EXTRA_NETWOR
K_INFO);
			if (info != null) {
				String type=info.getTypeName();
				boolean isConnected=info.isConnected();
				Toast.makeText(context,"NetworkType:"+type+",
isConnected:"+isConnected,Toast.LENGTH_LONG).show();
			}
		}
	}
}
```


5:24午後
ワイヤレスとネットワークの設定
機内モード
Wi-Fi
Wi-FiをONにする
Wi-Fi設定
ワイヤレスアクセスポイントの設定
Bluetooth
BluetoothをONにする
Bluetooth設定
端末の名前、接続や検出設定
NetworkType:mobile, isConnected:false
VPNの設定と管理
モバイルネットワーク

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 プログラミング入門

2014 年 5 月 26 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）