Android 4.2 の注意事項

この記事を書く上で、Android プログラムをチェックしている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)

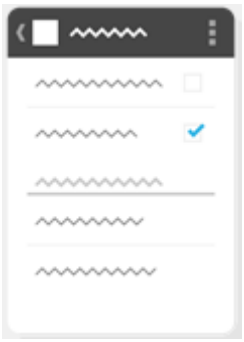
1. プロジェクト作成時にアクティビティの種類を選べるようになりました
Android 4.2 では以下の 5 種類のアクティビティを選択できるようになりました。
・BlankActivity

・FullscreenActivity

・LoginActivity

・MasterDetailFlow

・SettingsActivity



2.　Android Lint Checks

以前のバージョンよりコンパイルのチェックが厳しくなりました。このような厳しいチェックを「Android Lint Checks」と言います。

Abdroid プログラミング Bible で掲載しているプログラムにおいて「Android Lint Checks」による主な警告エラーは以下です。

①「Avoid object allocations during draw/layout operations (preallocate and reuse instead)」

これは Paint オブジェクトや Bitmap オブジェクトを onDraw メソッド内で生成している場合に発生します。たとえば以下のような場合です。

```
private class GView extends View {
    public GView(Context context) {
        super(context);
```

```
        }
        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。従って以下のように書き換えます。

```
    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);
        }
    }
```

同様に、

```
    private class GView extends View {
        public GView(Context context) {
            super(context);
        }
        protected void onDraw(Canvas canvas) {
            Bitmap img = BitmapFactory.decodeResource(getResources(),
R.drawable.ic_launcher);
            canvas.drawBitmap(img,0,0,null);
```

```
        }
    }
```
は以下のように書き換えます。
```
    private class GView extends View {
        private Bitmap img;
        public GView(Context context) {
            super(context);
            img=BitmapFactory.decodeResource(getResources(),
R.drawable.ic_launcher);
        }
        protected void onDraw(Canvas canvas) {
            canvas.drawBitmap(img,0,0,null);
        }
    }
```

②非推奨クラス、メソッド、定数に対し以前のバージョンよりより厳しく警告エラーを出します。
API8 以後は ACTION_POINTER_2_DOWN や ACTION_POINTER_2_UP などの定数は非推奨となっています。以前は以下のようなコードを記述していましたが、ACTION_POINTER_2_DOWN で警告エラーとなります。
```
    public boolean onTouchEvent(MotionEvent event) {
        int action=event.getAction();
        if (action==MotionEvent.ACTION_POINTER_2_DOWN){

        }
    }
```
警告を避ける最も簡単な方法は「action==MotionEvent.ACTION_POINTER_2_DOWN」の代わりに「action==(MotionEvent.ACTION_POINTER_DOWN|0x0100)」とします。しかしこれは ACTION_POINTER_2_DOWN を非推奨にした趣旨に反しますので、以下のように、イベント内容（DOWN,UP,MOVE）と何番の指かの情報を分離して判定します。マルチタッチした場合に何番目にタッチした指かの情報は event.getAction()の上位バイトに格納されています。event.getAction()と MotionEvent.ACTION_MASK（0x00ff）との&をとれば上位バイトがマスクされ、どの指のダウンでも MotionEvent.ACTION_POINTER_DOWN（0x0005）の値となります。
何番目の指かは event.getAction()の下位バイトをマスクした値を 8 ビット右シフトすると、最初の指が「0」、2 番目の指が「1」、3 番目の指が「2」という値になり、この値で判定で

きます。MotionEvent.ACTION_POINTER_INDEX_MASK は「0xff00」、MotionEvent.ACTION_POINTER_INDEX_SHIFT は「0x0008」という値に定義されています。

```
public boolean onTouchEvent(MotionEvent event) {
    int action=event.getAction() & MotionEvent.ACTION_MASK;
    if (action==MotionEvent.ACTION_POINTER_DOWN){
        int n=(event.getAction() & MotionEvent.ACTION_POINTER_INDEX_MASK)>>MotionEvent.ACTION_POINTER_INDEX_SHIFT;
        if (n==0){
            // 1 番目の指のダウン
        }
        else if (n==1){
            // 2 番目の指のダウン
        }
        else if (n==2){
            // 3 番目の指のダウン
        }
    }
```

3. 非推奨クラス、メソッド、定数

Android 4.2 では以下のクラスおよびメソッドが非推奨となっています。代替えがあるものは代替えしますが、ないものはそのまま使用しても正常に動作します。

・Gallery クラス
・Notification クラス
・getWidth()、getHeight()メソッド
・managedQuery メソッド
・Sensor.TYPE_ORIENTATION 定数
・SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS 定数

画面の幅と高さを取得する `getWidth()`と `getHeight()`は `getSize()`で代替えします。この場合「`@SuppressLint("NewApi")`」が必要です。

```
@SuppressLint("NewApi")
@Override
```

```
    public void onCreate(Bundle savedInstanceState) {

        WindowManager
wm=(WindowManager)getSystemService(WINDOW_SERVICE);
        Display disp=wm.getDefaultDisplay();
        Point p=new Point();
        disp.getSize(p);
        int w=p.x;
        int h=p.y;
```

4. デフォルトで生成されるファイル名

・レイアウトファイル名が main.xml から activity_main.xml に代わりました。

・クラス名が「プロジェクト名 Activity」から「MainActivity」に変わりました。これに伴い Java ソースファイル名も「プロジェクト名 Activity.java」から「MainActivity.java」に変わりました。

5. LinearLayout と RelatireLayout

新しいバージョン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 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、本書の例題は LinearLayout を使います。

6. 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。古いバージョンの activity_main.xml に生成されるレイアウトは fill_parent を生成していましたが、新しいバージョンでは match_parent を生成します。

7. フィンガープリント

Android 4.2 とは関係ありませんが、新しい環境でフィンガープリントを取得するには以下の注意点があります。

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
```