3　画面にウィジェットを表示する

Android のアプリ開発では通常、XML ファイルと Java ファイルの両方を作ります。画面内の GUI 部品などの配置（レイアウト）は XML ファイルを記述し、これらの部品を用いるロジックを Java ソースで書きます。この単元では、この方法で説明していきます。

なお、ゲームアプリではレイアウトもロジックも Java ソースで書く場合がありますが、この方法は、Android 基礎で説明します。

それでは、画面にウィジェットを表示してみます。ウィジェット（widget）とは、ボタンやテキスト入力フィールドのような画面を構成する部品群のことです。

3.1　文字列定数を追加する。

画面に表示する文字列の定義をリソースとしてソースコードの外に定義、管理することが一般的です。Android では、文字列を res/values/strings.xml の中に定義します。

ファイル名：　res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>  //①
    <string name="app_name">DICalculator</string>  //②
    <string name="label_discomfortIndex">不快指数を計算します</string>
    <string name="label_temperature">気温(℃)</string>
    <string name="label_humidity">湿度(%)</string>
    <string name="button_calculate">計算</string>
</resources>
```

①<resources>

strings.xml は、文字列定数を定義するリソースファイルです。リソースファイルは、resource 要素と、子要素からなる XML で表現します。

②<string name="app_name">DICalculator</string>

アプリケーション名です。文字列定数は string 要素として定義します。string 要素の name 属性に指定された文字列(app_name)をキーとして、string 要素の内容(DICalculator)を他のリソースファイルやプログラムから参照することができます。

文字列の定義

※temperature: témp(ə)rətʃʊə　ﾃﾝﾊﾟﾁｬ

※humidity: hju:mídəṭi　ﾋｭﾐﾃﾞｨﾃｨ

3.2　ビュー(Viewクラス)

画面に配置できるオブジェクトを**ビュー（View クラス）**といいます。

例えば、ボタン（Button)、テキスト(TextView)、画像(ImageView)、ビューグループ(ViewGroup)などがあります。

ビューグループのうち、張り付けるビューの間隔や、左寄せや右寄せといったように配置を指定できるものを**レイアウト**といいます。

レイアウトには、順番にビューを並べるだけの LinearLayout や各部品の相対位置を指定できる RelativeLayout などがあります。※Linear…直線

これらのビューは XML ファイルで記述します。

3.3 XML

XML(eXtensible Markup Language)はマークアップ言語のひとつです。これは、データを作成する人が、タグ(tag 〈〉で囲まれた部分)と呼ばれる情報を自由にデータの中に埋め込むことができます。タグには開始タグと終了タグがあり、終了タグにはタグの先頭に「/」を付けます。

XML ファイルにすることで、デザインを行う人間とコードを作成する人間とで開発を分担することができます。また、一度作成したデザインやコードを再利用しやすくするという利点もあります。

XML のタグは

〈要素　属性＝“値”＞内容＜／要素＞

と記述します。

例　〈LinearLayout　android:属性１ ＝ 値 /〉

3.4 ウィジェット(widget)を定義する

ウィジェットの定義をリソースとしてソースコードの外に定義､管理することが一般的です。Android では、res/layout/ の中に定義します。　widget:部品

ウィジェットの定義

なお、縦向きのレイアウトは、layout/main.xml ファイル、横向きのレイアウトは、layout-land/main.xml に書きます。

ファイル名：res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout   xmlns:android="http://schemas.android.com/apk/res/android"
//①
    android:layout_width="fill_parent"  //②
    android:layout_height="fill_parent"
    android:orientation="vertical" >  //③
    <TextView   //④
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/label_discomfortIndex"  //⑤
        android:textSize="20sp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/label_temperature"
        android:textSize="20sp" />
```

```
    <EditText
        android:id="@+id/text_temperature"  //⑥
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#F0F0F0"
        android:inputType="number"  //⑦
        android:maxLength="2" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/label_humidity"
        android:textSize="20sp" />
    <EditText
        android:id="@+id/text_humidity"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#F0F0F0"
        android:inputType="number"
        android:maxLength="3"
        android:textColor="#000000" />
    <Button
        android:id="@+id/button_calculate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_calculate"
        android:textSize="20sp" />
</LinearLayout>
```

①<LinearLayout

Activity が持つ UI は、Java プログラムとしても記述できますが，基本は XML でレイアウトを記述します。これは.NET Framework や Web アプリケーションなどと思想が似ています。Linearlayout は、縦・横どちらか一直線に View を配置するレイアウトです。

代表的な View グループ (Layout)

1	absolute layout	絶対座標指定の配置が可能
2	relative layout	相対座標指定の配置が可能
3	Linear（リニア） layout	縦／横指定の配置が可能
4	Table layout	表形式の配置が可能

②android:layout_width="fill_parent"

「wrap_content」、「fill_parent」、「match_parent」は三つとも幅の長さを指定しています。

「wrap_content」の場合は、幅を自動調整するという意味です。「wrap_content」に設定しておけば Android 側が適切な幅に設定してくれます。(原則として最小サイズ)
「fill_parent」と「match_parent」は設定された View の大きさではなく、その View を囲んでいる親 View の大きさを元に最大まで引き伸ばす設定です。

仮に「android:layout_width="fill_parent"」と設定したなら、横幅は画面いっぱいに

表示するということになります。

③android:orientation="vertical" >

vertical により、ウィジェットが縦に並んで配置されます。

android:orientation はタグに加えることのできる属性です。「vertical」と「horizontal」の 2 種類の指定値があります。「vertical」は縦、「horizontal」は横に並べます。

④<TextView

画面に文字列を表示する機能を持った TextView 属性を使っています。

それ以外にいろいろな GUI 部品があります。

・ラベル（TextView）

・テキストボックス(EditText)

・補完機能付きテキストボックス（AutoCompleteTextView）

・画像（ImageView）

・ボタン（Button）

・チェックボックス（CheckBox）

・ラジオボタン（RadioButton）

・プログレスバー（ProgressBar）

・ウェブビュー（WebView）

・ビデオビュー(VideoView)

⑤android:text="@string/label_discomfortIndex"

TextView 属性の「android:text」に指定した文字列が画面に表示されます。

「@string/label_discomfortIndex」は「strings.xml」で書かれた<string name="label_discomfortIndex">を意味し、そこで指定された「不快指数を計算します」を画面に表示します。

⑥android:id="@+id/text_temperature"

ここで設定した ID を、プログラムから「R.ID.名前」と指定することで利用することができます。

ビューに「@+id/名前」という属性を付けると、次のように R クラス内の id クラスに自動的にその名前の変数が組み込まれ、そのビューのリソース ID がセットされます。

```
public static final int text_temperature=0x7f070001;
```

⑦android:inputType="number"

入力する内容に応じたソフトウェアキーボードを表示します。

inputType の入力制限には以下のものが使用できます。制限は「|」で区切って複数指定できます。

種類	説明
none	入力不可。
text	普通のテキスト。
textCapCharacters	すべて大文字で入力する場合。
textCapWords	単語の先頭を大文字で入力する場合。

textCapSentences	文章の先頭を大文字で入力する場合。
textAutoCorrect	文字の入力を自動で修正する場合。
textAutoComplete	文字の補完入力する場合。
textMultiLine	文字を複数行入力する場合。
textImeMultiLine	通常の文字入力時は複数入力を許可せず、IME によって複数行入力を設定する場合。
textUri	URL を入力する場合。
textEmailAddress	メールアドレスを入力する場合。
textEmailSubject	メールの件名を入力する場合。
textShortMessage	ショートメッセージを入力する場合。
textLongMessage	ロングメッセージを入力する場合。
textPersonName	人名を入力する場合。
textPostalAddress	住所を入力する場合。
textPassword	パスワードを入力する場合。
textVisiblePassword	パスワードの文字を見せて入力する場合。
textWebEditText	HTML を入力する場合。
textFilter	他のデータでフィルタされた文字を入力。
textPhonetic	発音記号を入力する場合。
number	数値入力する場合。
numberSigned	符号付きの数値を入力する場合。
numberDecimal	小数入力する場合。
phone	電話番号を入力する場合。
datetime	日付時刻を入力する場合。
date	日付を入力する場合。
time	時刻を入力する場合。

3.5　アクティビティを作成する

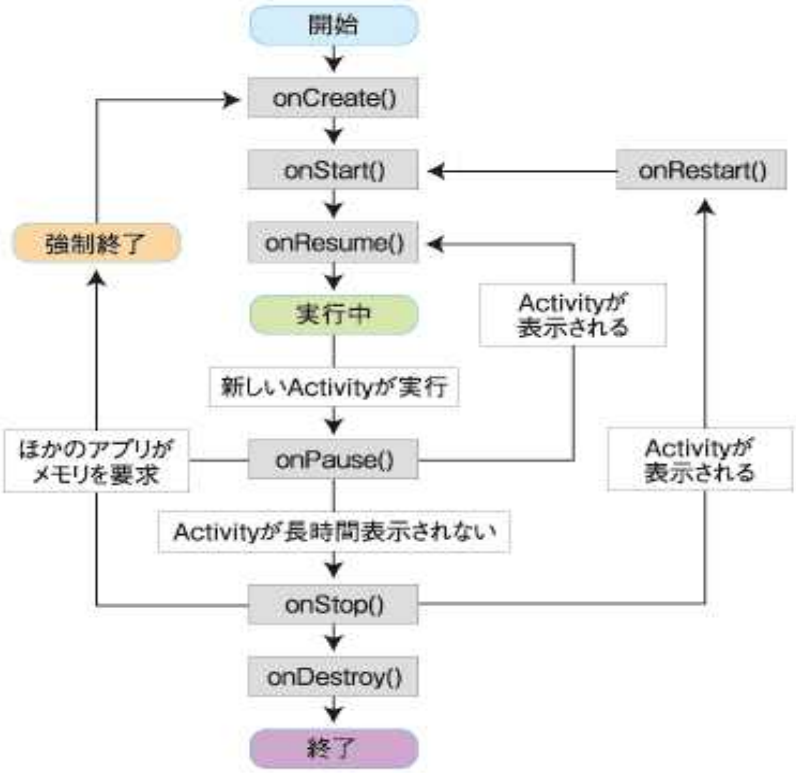
3.5.1　Android アプリの構成要素（フレームワーク）

構成要素	英語表記	内容
アクティビティ	Activity	ユーザーとアプリケーション間のやりとりを仲介するオブジェクト。1 つの画面に対して 1 つのアクティビティが対応付けられる。 （ユーザーインタフェース、イベント処理など）
ブロードキャストレシーバー	Intent	他のアプリや Android OS からのインテント（メッセージ）を受信し、各種処理を行うオブジェクト。プログラム間連携の仕組み。
サービス	Service	画面表示とは独立してバックグランドで処理を実行し続けることが可能なオブジェクト
コンテンツプロバイダ	Content Provider	データベースのデータを他のアプリに提供するオブジェクト

3.5.2　Activity のライフサイクル

通常は「1 アクティビティ＝1 画面」で作成し、レイアウトやユーザーのイベント、アクティビティの状況に応じた処理を記述します。

アクティビティは開始から終了まで、次の図のようなライフサイクルで動作します。



Android は、マルチタスクで複数の処理を同時に実行できますが，画面（フォアグラウンド）に表示されるアプリケーションは、常に 1 つと定められています。トランプのカードが重なっている様子を想像すると理解しやすいでしょう。カードの 1 枚ずつがアクティビティで，呼び出されたアクティビティが一番上（フォアグラウンド）に表示されます。

携帯端末上で動作するアプリケーションは，さまざまな要因で割り込みが発生し，その都度中断されます。例えば，電話が着信すれば着信画面に切り替わり，ユーザーが一定時間操作しなければ端末が「スリープ・モード」になります。

割り込みの要因が終了した場合、例えば電話が終わったり、ユーザーが操作してスリー

プ・モードを解除したりした際には、元のアクティビティを中断前の状態から再開する必要があります。ゲームのアプリケーションの場合、電話の着信のタイミングで一時停止しておかなければ、電話が終わった時にゲームオーバーの画面が表示されているという悲劇は避けられません。

Activity Manager は、カーネル上の各種ドライバ、例えば、電源管理ドライバや電話機能管理ドライバの状態変化に応じて、現在起動中のアプリケーション（正確にはアクティビティ）にライフサイクルの変化を通知します。

これにより、表示中のアクティビティは、フレームワーク層より下（カーネルやドライバ、ハードウエア）を意識せずに、割り込みによる中断や再開、終了など、さまざまなタイミングで必要となる前処理を実行できます。

3.5.3 画面構成

3.5.4 DICalculatorActivity を新規作成する

ファイル名： src/jp.edu.mie/DICalculatorActivity.java （**新規作成**）

```
public class DICalculatorActivity extends Activity  //①
{
    @Override //②
    protected void onCreate(Bundle savedInstanceState)  //③
    {
        super.onCreate(savedInstanceState);  //④
        setContentView(R.layout.main);  //⑤
    }
}
```

※DICalculatorActivity.java を実行する前に次の作業を行ってください。

①余分な命令が作成されているので削除してください。

②res/menu/ DICalculatorActivity.xml を削除してください。

このソースを実行すると res/layout/main.xml に定義されているレイアウトを画面に表示するだけで、ボタンを押しても何にも反応しません。

①public class DICalculatorActivity extends Activity

DICalculatorActivity は、Activity クラスか、Activity クラスのサブクラスを継承して作成します。

②@Override

アノテーションと呼ばれる Java の機能で、注釈のようなものです。onCreate メソッドが親クラスである Activity の同名メソッドをオーバーライドするものだということを示しています。

③protected void onCreate(Bundle savedInstanceState)

onCreate メソッドは、アクティビティのライフサイクルに関係するイベントハンド

ラです。アクティビティが生成されるときに必ず、onCreate メソッドが呼ばれます。

Bundle は、キーとバリュー形式でデータを保存できるクラスです。Activity は、一時停止したり停止したりするときに、画面の状態を Bundle クラスのオブジェクトに保存しておきます。破棄されると値は消えてしまします。

④super.onCreate(savedInstanceState);

onCreate メソッドの中では、必ずスーパークラスの同名のメソッドを呼ばなければなりません。呼ぶことを怠ると実行時に例外となります。

端末が回転され、アクティビティが再起動されたときに、保存した Bundle データを基に以前の状態を復元する必要があります。しかし復元処理は親が行ってくれているということになります。

⑤setContentView(R.layout.main);

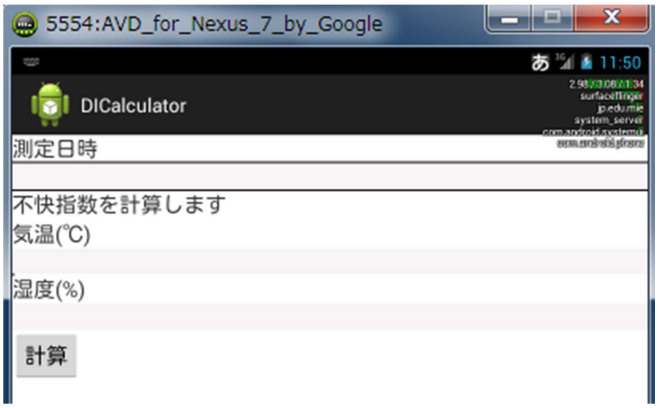
Activity クラスの setContentView メソッドは、アクティビティにテキストやボタンといった部品を配置するメソッドです。つまり、ユーザーインターフェイス(UI)を設定します。ここでは、res/layout/main.xml に定義されているレイアウトを適用しています。

R.layout.main とは、R クラスの layout というインナークラスの main という変数を指しています。R とは Resource の頭文字をとったものです。Android アプリは、画像ファイルやレイアウト XML ファイルなどのリソースにリソース ID という一意の整数値を割り当てています。プログラムでは、その整数値を指定することでリソースを扱うことができます。

演習

【演習１】この時点で実行してみる。

【演習２】気温と湿度の測定日時を入力する項目を、「不快指数を計算します」の表示の上に追加してください。なお、名称は任意で付けてください。



【演習３】res/layout/sample.xml を新規作成して、ユーティリティにより画面を作成してください。