Android 実習マニュアル

第零版 revision03

大黒学

Android 実習マニュアル・第零版 revision03
著者——大黒学

2010 年　1 月 29 日（金）　第零版発行
2010 年　7 月 20 日（火）　第零版 revision03 発行

目次

第 1 章　Android の基礎

1.1　Android の概要

1.1.1　この節について

この「Android 実習マニュアル」は、Android の上で動作するアプリケーションを開発する方法について解説することを目的とする文章です。そこでまず、この節では、Android というものについて、その概要を説明したいと思います。

1.1.2　プラットフォーム

ハードウェアとアプリケーションの中間に位置していて、アプリケーションに対してさまざまな機能を提供する一群のソフトウェアは、「プラットフォーム」(platform) と呼ばれます。パソコンの場合は、Windows や MacOS や Linux などのオペレーティングシステムが、プラットフォームとしての役割を果たしています。

ハードウェアの構造というのは機種ごとにさまざまですので、基本的には、ある機種の上で動作するアプリケーションは、別の機種の上では動作しません。しかし、ハードウェアがどれほど異なっていたとしても、それらの上で共通のプラットフォームが動作しているならば、そのプラットフォームの上で動作するように作られたアプリケーションは、それらのハードウェアのうちのどれの上でも動作することになります。

1.1.3　Android とは何か

Android というのは、Google によって開発された、携帯電話などのモバイル機器の上で動作するプラットフォームです。

ちなみに、モバイル機器のプラットフォームとしては、Android のほかに、Symbian、Windows Mobile、BREW などがあります。

1.1.4　Android の開発言語

Android の上で動作するアプリケーションは、「Android アプリケーション」(Android application) と呼ばれます。

Android アプリケーションを開発するためには、何らかのプログラミング言語を使ってプログラムを書く必要があります。現在のところ、Android アプリケーションを開発するためのプログラミング言語としてサポートされているのは、Java のみです。

したがって、この「Android 実習マニュアル」という文章は、Java を使って Android アプリケーションを開発する方法について解説していくことになります。ただし、この文章の中に、Java 自体についての説明は含まれていません。ですから、Java についてまったく知らない人は、この文章を読む前に、あらかじめ Java について学習しておく必要があります。

1.1.5　Dalvik VM

Java のコンパイラ (javac) は、Java で書かれたプログラムを、「バイトコード」(bytecode) と呼ばれるプログラムに変換して、`.class` という拡張子を持つファイル（.class ファイル）にそれを出力します。そして、バイトコードは、「Java VM」(Java Virtual Machine) と呼ばれる仮想マシンの上で実行されます。

Android も、仮想マシンの上でバイトコードを実行するように作られています。ただし、Android が持っている仮想マシンは、Java VM ではなくて、「Dalvik VM」(Dalvik Virtual Machine)[1] と呼ばれるものです。

Dalvik VM の上で動作するバイトコードは、「Android バイトコード」(Android bytecode) と呼ばれます。Android バイトコードは、`.dex` という拡張子を持つファイルに保存されます。このファイルは、「Dalvik 実行可能形式ファイル」(Dalvik executable format file)、またはその拡張子から、「.dex ファイル」(.dex file) と呼ばれます。

[1] Dalvik VM という名前は、この仮想マシンを開発した Dan Bornstein さんという人の先祖が住んでいた、アイスランドにある Dalvík という漁村の名前にちなんだものです。

1.1.6　Android パッケージファイル

ひとつの Android アプリケーションは、Android バイトコードを含むいくつかのファイルから構成されます。Android アプリケーションは、それを構成するいくつかのファイルをひとつにまとめた形で配布されます。

Android アプリケーションを構成するファイルをひとつにまとめたものは、`.apk` という拡張子を持つファイルに保存されます。このファイルは、「Android パッケージファイル」(Android package file)、またはその拡張子から、「.apk ファイル」(.apk file) と呼ばれます。

1.2　開発環境

1.2.1　必要なソフトウェア

この節では、Android アプリケーションを開発するために必要となる環境について説明したいと思います。

Android アプリケーションの開発環境としては、通常、次のソフトウェアを使います。

- JDK(Java Development Kit)
- Android SDK
- Eclipse
- ADT(Android Development Tools)

これらのソフトウェアは、すべて、インターネットで配布されていて、無料で利用することができます。

1.2.2　JDK

JDK(Java Development Kit) は、Java のコンパイラやクラスライブラリなどから構成されているソフトウェアで、Sun Microsystems によって配布されています。

JDK のダウンロードやインストールの方法については、

```
http://java.sun.com/
```

というサイトを参照してください。

1.2.3　Android SDK

Android SDK は、Android エミュレーター（パソコンの上で Android アプリケーションを動作させるプログラム）や、.class ファイルを.dex ファイル（Dalvik 実行可能形式ファイル）に変換するプログラムや、.apk ファイル（Android パッケージファイル）を作るプログラムなどから構成されているソフトウェアで、Google によって配布されています。

Android SDK のダウンロードやインストールの方法については、

```
http://developer.android.com/
```

というサイトを参照してください。

1.2.4　Eclipse

Eclipse は、「IDE」と呼ばれる種類のソフトウェアで、Eclipse Foundation によって配布されています。IDE というのは、統合開発環境 (integrated development environment)、つまり、コンパイラやテキストエディターやデバッガーなどを単一のユーザーインターフェースから操作することができるようにするソフトウェアのことです。

Eclipse のダウンロードやインストールの方法については、

```
http://www.eclipse.org/
```

というサイトを参照してください。

1.2.5　ADT

ADT(Android Development Tools) は、Android アプリケーションを開発するための機能を Eclipse に追加するプラグインで、Google によって配布されています。

ADT をインストールする方法については、Android SDK と同様、

```
http://developer.android.com/
```

を参照してください。

1.3 Eclipse の使い方

1.3.1 プロジェクトの作成

この節では、Android アプリケーションを開発する場合の Eclipse の使い方について説明したいと思います。

Eclipse を使ってアプリケーションを開発する場合、まず最初にしないといけないことは、「プロジェクト」(project) と呼ばれるものを作成することです。

プロジェクトというのは、アプリケーションを開発する過程で扱われるファイルとディレクトリ（フォルダ）から構成されるツリーのことです。

それでは、Android アプリケーションのプロジェクトを作成してみましょう。まず、Eclipse のメニューで、

[File]→[New]→[Android Project]

を選択してください。すると、New Android Project というダイアログが開きます。このダイアログは、次のような項目から構成されています。

Project name　プロジェクト名を入力する項目です。プロジェクト名はプロジェクト全体のディレクトリ名になりますので、ディレクトリ名として使うことのできる名前でないといけません。

Contents　まったく新しいプロジェクトを作るのか、それとも既存のソースからプロジェクトを作るのか、ということを選択する項目です。また、Use default location というチェックボックスのチェックをはずして、Location という項目にパス名を入力することによって、デフォルト以外の場所にプロジェクトのディレクトリを作ることもできます。

Build Target　作成するアプリケーションのターゲット、つまり、それを動作させる Android のバージョンを選択する項目です。

Properties　作成するアプリケーションの属性を入力する項目です。

Application name は、アプリケーション名を入力する項目です。アプリケーション名というのは、Android がアプリケーションの名前として画面に表示する文字列です。日本語を使うこともできます。

Package name は、パッケージ名を入力する項目です。パッケージ名というのは、アプリケーションを構成するクラスが所属する Java のパッケージの名前です。アプリケーションの作成者、またはその人が所属している組織が保有しているドメイン名を、逆の順序で並べ換えて、その最後にプロジェクト名を置きます。たとえば、プロジェクト名が `namako` で、保有しているドメイン名が `example.org` だとすると、

```
org.example.namako
```

というパッケージ名を使うことになります。

Create Activity は、アクティビティーを作るかどうかを選択するチェックボックスです。アクティビティーというのは、画面を生成するオブジェクトのことです。このチェックボックスにチェックが入っている場合、アクティビティーを生成するクラスを定義するソースコードが自動的に生成されます。その場合、チェックボックスの右側の項目には、そのクラスの名前を入力する必要があります。

Min SDK Version は、ターゲットに対応する、API Level と呼ばれる整数を入力する項目です。たとえば、ターゲットが Android 2.2 ならば、API Level は 8 になります。

それでは、このダイアログに次のように入力してください。

Project name	`sample`
Contents	そのまま
Build Target	Android x.x のうちのどれかひとつを選択
Application name	サンプル

Package name	`org.example.sample`
Create Activity	`SampleActivity`
Min SDK Version	ターゲットに対応する API Level

入力できましたら、Finish というボタンをクリックしてください。そうすると、Eclipse のワークスペースとして設定されているディレクトリの下に、`sample` という名前のディレクトリが作成されます。このディレクトリと、その下にあるディレクトリとファイルが、作成されたプロジェクトです。

Create Activity にチェックを入れて Android アプリケーションのプロジェクトを作成すると、自動的に、

`Hello World,` アクティビティーのクラス名

という文字列を画面の上に表示する Android アプリケーションが生成されます。ですから、プロジェクトを作成した直後の Android アプリケーションを、そのまま実行することも可能です。

1.3.2 アプリケーションの実行

それでは、プロジェクトを作成した直後の Android アプリケーションを、そのまま実行してみましょう。Eclipse のメニューで、

[Run]→[Run]

を選択してください。すると、Run As というダイアログが開きますので、その中にある Android Application をクリックして、それから OK をクリックしてください。そうすると、Android エミュレーターが起動して、アプリケーションを実行します。ただし、Android エミュレーターは、起動した直後は画面がロックされています。エミュレーターの画面をマウスで操作することによって画面のロックを解除すると、

```
Hello World, SampleActivity
```

と画面に表示されるはずです。

1.3.3 パッケージエクスプローラー

Eclipse の画面の左端にある領域は、「パッケージエクスプローラー」(Package Explorer) と呼ばれます。この領域は、パッケージを構成しているさまざまな要素を、ディレクトリとファイルから構成されるツリーの形で表示します。

ソースコードを編集したいときは、パッケージエクスプローラーの中に表示されているディレクトリを開いて、編集したいソースコードのファイルをダブルクリックします。そうすると、そのソースコードが、Eclipse の画面の中央にあるエディターの領域に表示されます。

それでは、`sample` ディレクトリの下にある `Sample.java` というソースコードを編集してみましょう。ディレクトリの左側に表示されているプラスをクリックして、

```
sample/src/org.example.sample
```

を開いて、その下にある `SampleActivity.java` をダブルクリックしてください。そうすると、そのソースコードがエディターの領域に表示されます。

ところで、パッケージエクスプローラーには、パッケージ名として使われたドメイン名が、あたかもひとつのディレクトリの名前であるかのように表示されます。しかし、実際には、`src` の下にそのような名前を持つ単独のディレクトリが作られるわけではありません。パッケージ名として使われたドメイン名は、ドットで区切られたそれぞれの部分が、階層を構成するそれぞれのディレクトリの名前になるのです。たとえば、パッケージ名として、

```
org.example.sample
```

というドメイン名を使ったとすると、

```
org/example/sample
```

というディレクトリの階層が `src` の下に作られます。

1.3.4 テキストビュー

人間によって操作されるコンピュータのソフトは、人間から指示を受け取ったり、人間に情報を提供したりするための部分を持っています。そのような部分は、「ユーザーインターフェース」

(user interface) と呼ばれます。

Android では、ユーザーインターフェースを作るための基本的な部品のことを「ウィジェット」(widget) と呼びます。プロジェクトを作成したときに自動的に生成される Android アプリケーションは、「テキストビュー」(text view) と呼ばれるウィジェットを使うことによって文字列を画面に表示します。

ところで、プロジェクトを作成したときに自動的に生成される Android アプリケーションは、テキストビューを使って、

```
Hello World, アクティビティー名
```

という文字列を画面の上に表示するわけですが、このテキストビューについての記述は、Java のソースの中には書かれていません。実は、その記述は、「レイアウト XML」(layout XML) と呼ばれる XML 文書の中に書かれているのです。

レイアウト XML というのは、画面のレイアウトについて記述した XML 文書のことです。自動的に生成された Java のソースの中に書かれている、

```
setContentView(R.layout.main);
```

という文は、レイアウト XML の中に記述されたレイアウトを持つ画面を表示するという意味です。

画面を表示するためにはかならずレイアウト XML を書かないといけない、というわけではありません。レイアウト XML を使わないで、Java のソースだけを使って画面を表示することも可能です。たとえば、テキストビューを使って、

```
Hello World, SampleActivity
```

という文字列を画面の上に表示する Android アプリケーションは、レイアウト XML を使わなくても、Java のソースの中に、

```
TextView tv = new TextView(this);
tv.setText("Hello World, SampleActivity");
setContentView(tv);
```

と書くことによって作ることができます。

1.3.5 ソースの編集

それでは、ソースを編集してみましょう。`SampleActivity.java` を次のように書き換えてください。

プログラムの例 `SampleActivity.java`

```
package org.example.sample;

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

public class SampleActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("私は Android アプリケーションです。");
        setContentView(tv);
    }
}
```

ソースコードをこのように書き換えたのち、アプリケーションを実行してください。そうすると、Android エミュレーターの画面に、

```
私は Android アプリケーションです。
```

と表示されるはずです。

1.4 ログ

1.4.1 ログの基礎

Android は、自分の動作に関するメッセージを、「ログ」(log) と呼ばれるものに書き込みます。ログに書き込まれるメッセージは、Android アプリケーションが出力することも可能です。

ログにメッセージを書き込む記述をソースコードに挿入しておくと、それが実行された時点で、そのメッセージがログに書き込まれます。式の値をメッセージとしてログに書き込むことも可能です。ログは、アプリケーションをデバッグする上で、とても有用です。

1.4.2 ログにメッセージを出力するメソッド

デバッグのためにログにメッセージを書き込みたいときは、

```
android.util.Log
```

というクラスが持っている、

```
static int d(String tag, String msg)
```

という静的メソッドを使います。1 個目の引数は、メッセージを出力した主体を識別するために使われる、「タグ」(tag) と呼ばれる文字列で、2 個目の引数はメッセージです。

1.4.3 ログにメッセージを出力するアプリケーションの例

それでは、ログにメッセージを書き込むアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`log`
Application name	ログ
Package name	`org.example.log`
Create Activity	`LogActivity`

次に、`LogActivity.java` を次のように書き換えてください。

プログラムの例 `LogActivity.java`

```
package org.example.log;

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

public class LogActivity extends Activity {
    private static final String TAG = "LogActivity";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.d(TAG, "I was created.");
    }
}
```

これで完成です。このアプリケーションを実行すると、アクティビティーが生成された時点で、`LogActivity` というタグと `I was created.` というメッセージがログに書き込まれます。

1.4.4 LogCat

Eclipse の画面は、ログを閲覧することのできる、LogCat と呼ばれる領域を持っています。

それでは、Eclipse に LogCat を表示させてみましょう。まず、メニューで、

[Window]→[Show View]→[Other...]

を選択してください。そうすると、Show View というダイアログが開きますので、Android の下にある LogCat を選択して、OK ボタンをクリックしてください。

LogCat の中にある Filter というところに文字列を入力すると、その文字列を含んでいるメッセージだけが表示されます。たとえば、`was` という文字列を入力してみてください。そうすると、

I was created.

というメッセージだけが表示されるはずです。

1.5 Android アプリケーションのインストールと削除

1.5.1 この節について

この節では、Android エミュレーターに Android アプリケーションをインストールする方法と、Android エミュレーターから Android アプリケーションを削除する方法について説明したいと思います。

1.5.2 インストール

Eclipse を使って Android アプリケーションを開発した場合、その Android アプリケーションは、Android エミュレーターに自動的にインストールされますので、インストールの操作をする必要はありません。しかし、自分以外の誰かが開発した Android アプリケーションを Android エミュレーターで実行するためには、それをインストールするという操作をする必要があります。

Android エミュレーターに Android アプリケーションをインストールしたいときは、Android エミュレーターが動作している状態のときに、

```
adb install パス名
```

というコマンドをシェルに入力します。このコマンドの「パス名」というところには、インストールしたい Android アプリケーションの.apk ファイルのパス名を書きます。たとえば、インストールしたい Android アプリケーションの.apk ファイルの名前が `Sample.apk` だとするならば、そのファイルが格納されているディレクトリをカレントディレクトリにして、

```
adb install Sample.apk
```

というコマンドをシェルに入力すればいいわけです。

1.5.3 削除

Android エミュレーターから Android アプリケーションを削除したいときは、Android エミュレーターのシェルを使います。

Android エミュレーターのシェルは、Android エミュレーターが動作している状態のときに、

```
adb shell
```

というコマンドをシェルに入力することによって起動することができます。このシェルに対しては、`pwd`、`cd`、`ls`、`rm`、`exit` など、UNIX の基本的なコマンドを入力することができます。

Android アプリケーションは、その.apk ファイルを `rm` コマンドで削除することによって、Android エミュレーターから削除することができます。

.apk ファイルは、Android エミュレーターのファイルシステムの中にある、

```
/data/app
```

というディレクトリの下に格納されています。

1.6 コンポーネント

1.6.1 コンポーネントとは何か

Android アプリケーションは、「コンポーネント」(component) と呼ばれるオブジェクトから構築されます。ただし、Android で「コンポーネント」と呼ばれるものは、AWT や Swing で「コンポーネント」と呼ばれるものとはまったく別のものです。

コンポーネントには、次の 4 種類のものがあります。

- アクティビティー (activity)
- サービス (service)
- ブロードキャストレシーバー (broadcast receiver)

- コンテントプロバイダー (content provider)

1.6.2　アクティビティーとは何か

アクティビティーは、画面を生成する機能を持っているコンポーネントです。通常、ひとつのアクティビティーはひとつの画面を生成します。

アクティビティーは、`Activity` というクラスから生成されるオブジェクトです。

Android アプリケーションのプロジェクトを作成するとき、Create Activity というチェックボックスにチェックを入れて、そのチェックボックスの右側にクラス名を入力すると、その名前のクラスを定義するソースコードが自動的に生成されるわけですが、そのクラスは、`Activity` クラスのサブクラスです。

1.6.3　サービスとは何か

サービスは、時間のかかる処理をバックグラウンドで（画面の背後で）実行し続けたい、というときに使われるコンポーネントです。

アクティビティーには、それが生成した画面が表示されていないときに処理を中断させられてしまう可能性があります。ということは、もしも、音楽を再生するというような処理をアクティビティーに実行させていたとするならば、それとは別の画面が表示されているときに、その音楽が中断してしまう可能性がある、ということになります。ですから、そのような処理は、サービスを使って実行するほうが好ましいわけです。

サービスは、`Service` という抽象クラスのメソッドを実装したサブクラスから生成されるオブジェクトです。

1.6.4　ブロードキャストレシーバーとは何か

ブロードキャストレシーバーは、何らかの告知が発せられたときに、それを受信して何らかの処理をしたい、というときに使われるコンポーネントです。

ブロードキャストレシーバーは、`BroadcastReceiver` という抽象クラスのメソッドを実装したサブクラスから生成されるクラスのオブジェクトです。

1.6.5　コンテントプロバイダーとは何か

コンテントプロバイダーは、複数のアプリケーションでデータを共有したい、というときに使われるコンポーネントです。

コンテントプロバイダーは、`ContentProvider` という抽象クラスのメソッドを実装したサブクラスから生成されるオブジェクトです。

1.6.6　Android マニフェスト

Eclipse を使って Android アプリケーションのプロジェクトを作成すると、プロジェクトのフォルダのすぐ下に、`AndroidManifest.xml` という名前のファイルが自動的に生成されます。これは、すべての Android アプリケーションが持っていないといけないファイルです。

`AndroidManifest.xml` の中に格納されているのは、「Android マニフェスト」(Android manifest) と呼ばれる XML 文書です。Android アプリケーションを作るためには、Android マニフェストの中に、その Android アプリケーションを構成しているコンポーネントについての記述を書く必要があります。

Android アプリケーションのプロジェクトを作成するとき、Create Activity にチェックを入れると、自動的に生成される Android マニフェストの中には、1 個のアクティビティーについての記述が書き込まれます。

第 2 章　リソース

2.1　リソースの基礎

2.1.1　リソースとは何か

Android アプリケーションは、Android バイトコードだけではなくて、さまざまな種類のデータから構成されています。

Android アプリケーションを構成しているデータのうちで、Android バイトコード以外のものは、「リソース」(resource) と呼ばれます。
Android は、画面のレイアウト、文字列、色、大きさ、画像など、さまざまな種類のデータをリソースとして扱うことができます。

2.1.2 リソースファイル

リソースが定義されているファイルは、「リソースファイル」(resource file) と呼ばれます。リソースファイルの形式は、レイアウト、文字列、色、大きさなどの場合は XML 文書、画像の場合は PNG や JPEG などです。
プロジェクトの中にリソースファイルを置く場所は、リソースの種類ごとに、次のように決められています。

`res/layout` レイアウト。
`res/values` 文字列、色、大きさなど。
`res/drawable` 画像。

2.1.3 リソースインデックス

Eclipse で Android アプリケーションのプロジェクトを作成すると、Java のソースが置かれる場所に、`R.java` という名前のファイルが自動的に生成されます。このファイルに格納されているのは、「リソースインデックス」(resource index) と呼ばれる Java のソースです。
リソースインデックスは、文字列や色や大きさなどのリソースを Java のソースの中から参照することができるように、それらのリソースの名前を定数として定義しています。リソースを作成すると、その時点で自動的に、リソースの名前を定数として定義する記述がリソースインデックスに追加されます。ですから、`R.java` というのは、その内容をエディターで編集する必要性がまったくないファイルです。

2.2 レイアウト

2.2.1 レイアウト XML

第 1.3 節で説明したように、画面のレイアウトは、Java のソースの中に記述することもできますが、Java のソースから独立したリソースにすることも可能です。
リソースとしてのレイアウトは、XML によって記述されます。レイアウトを記述した XML 文書は、「レイアウト XML」(layout XML) と呼ばれます。
Eclipse で Android アプリケーションのプロジェクトを作成すると、

```
res/layout/main.xml
```

というレイアウト XML が自動的に生成されます。

2.2.2 ウィジェットの作り方

第 1.3 節で説明したように、Android では、ユーザーインターフェースを作るための基本的な部品のことを「ウィジェット」(widget) と呼びます。
Android では、テキストビュー、ボタン、エディットテキスト、チェックボックス、ラジオボタン、……というような、さまざまなウィジェットを使うことができます。
ウィジェットを作る方法は、二つあります。
ひとつは、Java のソースの中に、ウィジェットのクラスのインスタンスを `new` で生成する記述を書くという方法です。たとえば、

```
TextView tv = new TextView(this);
```

と書くことによって、ひとつのテキストビューを作ることができます。
ウィジェットを作るもうひとつの方法は、レイアウト XML の中に、ウィジェットを作るための要素を書くというものです。たとえば、`TextView` という要素型の要素を書くことによって、テキストビューを作ることができます。

2.2.3 ウィジェットの大きさを設定する属性

ウィジェットを作る要素を書く場合には、その大きさを決めるための次の二つの属性に対して、何らかの値を設定する必要があります。

`android:layout_width`	横の長さ。
`android:layout_height`	縦の長さ。

これらの属性に対しては、具体的な長さを設定することもできますが、次のような単語を設定することもできます。

`fill_parent`	親の長さに合わせる。
`wrap_content`	内容の長さに合わせる。

2.2.4 文字列を設定する属性

テキストビュー、ボタン、チェックボックス、ラジオボタンなどのウィジェットは、その上に 1 個の文字列を表示することができます。それらのウィジェットの上に表示される文字列は、

```
android:text
```

という属性によって決定されます。この属性に対しては、表示したい文字列そのものを設定することもできますし、リソースとして定義されている文字列を参照する記述を設定することもできます（リソースを参照する記述の書き方については、第 2.3 節で説明します）。

2.2.5 フォントを設定する属性

ウィジェットの上に文字列を表示するためのフォントは、

```
android:typeface
```

という属性によって決定されます。この属性に対しては、Android に最初から組み込まれている、次のようなフォントの名前を設定することができます。

`sans`	ひげ飾り (serif) のないフォント。
`serif`	ひげ飾りのあるフォント。
`monospace`	文字の横幅が一定のフォント。

2.2.6 三つのテキストビューを表示するアプリケーションの例

それでは、レイアウト XML を修正することによって、三つのテキストビューを表示するアプリケーションを作ってみましょう。

まず、次のようなプロジェクトを作成してください。

Project name	`layout`
Application name	レイアウト
Package name	`org.example.layout`
Create Activity	`LayoutActivity`

プロジェクトが作成できたら、次に、

```
res/layout/main.xml
```

をダブルクリックしてください。そうすると、そのレイアウト XML がエディターの領域に表示されます。レイアウト XML は、レイアウト形式とソース形式という二種類の形式で表示することができます。エディターの領域の下にあるタブをクリックすることによって、表示の形式を切り替えることができます。Layout のタブがレイアウト形式で、main.xml のタブがソース形式です。

次に、レイアウト XML を次のように修正してください。

レイアウト XML の例 `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:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:typeface="sans"
    android:text="I am a text view. (sans)"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:typeface="serif"
    android:text="I am a text view. (serif)"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:typeface="monospace"
    android:text="I am a text view. (monospace)"
    />
</LinearLayout>
```

これで完成です。実行してみてください。三つのテキストビューが画面の上に表示されて、それぞれのテキストビューの上に、異なるフォントを使って文字列が表示されるはずです。

2.3 文字列

2.3.1 文字列のリソースファイル

Android アプリケーションが扱う文字列は、Java のソースの中に書くこともできますし、レイアウト XML の中に書くこともできるわけですが、文字列のリソースファイルの中に書くということも可能です。

Eclipse で Android アプリケーションのプロジェクトを作成すると、

```
res/values/strings.xml
```

というファイルが自動的に生成されます。このファイルは、文字列を定義しているリソースファイルです。初期状態では、このリソースファイルの中で、テキストビューによって表示される文字列と、アプリケーションの名前が定義されています。

文字列や色や大きさのリソースファイルは、

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

というような、resources という要素型の要素をルート要素とする XML 文書です。

ひとつの文字列は、ひとつの string 要素によって定義されます。定義したい文字列を要素の内容として書いて、その文字列に与えたい名前を name 属性の値として書きます。たとえば、

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="message">私は文字列です。</string>
</resources>
```

というリソースファイルを書くことによって、「私は文字列です。」という文字列を、message という名前で定義することができます。

2.3.2 リソースを参照する記述

レイアウト XML からリソースを参照するためには、そのリソースを参照するための記述を書く必要があります。たとえば、TextView 要素の android:text 属性に対して、文字列のリソースを参照する記述を設定することによって、テキストビューの上に、その文字列を表示することができます。

リソースを参照する記述というのは、

@ リソースのタイプ / リソース名

という構文を持つ文字列のことです。「リソースのタイプ」というところには、参照したいリソースのタイプを示す名前を書きます。そして、「リソース名」のところには、リソースの名前（リ

ソースを定義したときに name 属性に設定した名前）を書きます。
　文字列、色、大きさというリソースのタイプは、それぞれ、次の名前によって示されます。

`string`　文字列。
`color`　色。
`dimen`　大きさ。

　たとえば、`hitode` という名前でリソースとして定義されている文字列は、

```
@string/hitode
```

という記述を書くことによって参照することができます。

2.3.3　二つの文字列を表示するアプリケーションの例

　それでは、文字列のリソースファイルに対して文字列の定義を追加することによって、二つの文字列を表示するアプリケーションを作ってみましょう。
　まず、次のようなプロジェクトを作成してください。

Project name	`string`
Application name	文字列
Package name	`org.example.string`
Create Activity	`StringActivity`

　プロジェクトが作成できたら、次に、

```
string/res/values/strings.xml
```

をダブルクリックしてください。そうすると、文字列のリソースファイルがエディターの領域に表示されます。文字列を定義しているリソースファイルは、リソース形式とソース形式という二種類の形式で表示することができます。エディターの領域の下にあるタブをクリックすることによって、表示の形式を切り替えることができます。Resources のタブがリソース形式で、strings.xml のタブがソース形式です。
　それでは、新しい文字列の定義をリソースファイルに追加しましょう。次のように、`strings.xml` の中に `string` 要素を書き加えてください。

文字列を定義する XML 文書の例　`strings.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, StringActivity</string>
    <string name="app_name">文字列</string>
    <string name="message">私は文字列です。</string>
</resources>
```

　それでは次に、レイアウト XML に対して、次のようにテキストビューの記述を追加してください。

レイアウト XML の例　`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:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/message"
    />
</LinearLayout>
```

これで完成です。実行してみてください。二つのテキストビューが画面の上に表示されて、それぞれのテキストビューによって、「Hello World, StringActivity」という文字列と、「私は文字列です。」という文字列が表示されるはずです。

2.3.4 Java のソースからのリソースの参照

リソースファイルの中で定義されているリソースは、Java のソースから参照することも可能です。

Java のソースからリソースを参照したいときは、

R. リソースのタイプ . リソース名

という形の式を書きます。たとえば、message という名前でリソースとして定義されている文字列を参照する式は、

```
R.string.message
```

と書きます。

リソースを参照する式を評価すると、その値として、「リソース ID」(resource ID) と呼ばれる、リソースを識別する整数が得られます。Android のオブジェクトが持っているメソッドの多くは、リソース ID を引数として受け取るように作られています。たとえば、テキストビューが持っている、

```
void setText(int resid)
```

というメソッドは、引数としてリソース ID を受け取って、それによって識別される文字列をテキストビューに設定します。

ちなみに、リソースを参照する式を評価することによってリソース ID を求めることができるのは、リソースの名前が、リソース ID を値とする定数として定義されているからなのですが、その定義が書かれているのが、第 2.1 節で紹介したリソースインデックス、つまり、R.java という名前のファイルに格納されている Java のソースなのです。

2.3.5 Java のソースからリソースを参照するアプリケーションの例

それでは、文字列のリソースファイルの中で定義されている文字列を Java のソースから参照するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	refer
Application name	文字列の参照
Package name	org.example.refer
Create Activity	ReferActivity

次に、strings.xml の中に、string 要素を次のように書き加えてください。

文字列を定義する XML 文書の例 strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, ReferActivity</string>
    <string name="app_name">文字列の参照</string>
    <string name="message">私はリソースです。</string>
</resources>
```

最後に、ReferActivity.java を次のように書き換えてください。

プログラムの例 ReferActivity.java

```
package org.example.refer;

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

public class ReferActivity extends Activity {
```

```
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText(R.string.message);
        setContentView(tv);
    }
}
```

これで完成です。実行してみてください。リソースファイルの中で定義した、「私はリソースです。」という文字列が、テキストビューの上に表示されるはずです。

2.4 色

2.4.1 色のリソースファイル

色も、文字列と同じように、resources という要素型の要素をルート要素とする XML 文書を書くことによって、リソースとして定義することができます。

ひとつの色は、ひとつの color 要素によって定義されます。定義したい色の記述を要素の内容として書いて、その色に与えたい名前を name 属性の値として書きます。

色は、16 進数の左側にシャープ (#) を書いたもので記述します。形式は、次の 4 種類のいずれかです。

#RGB
#ARGB
#RRGGBB
#AARRGGBB

A はアルファ値、R は赤、G は緑、B は青を示す 16 進数の 1 桁を意味しています。アルファ値については、第 5.1 節で説明したいと思います。

たとえば、

```
<color name="hitode">#f00</color>
```

という要素を書くことによって、赤色を、hitode という名前で定義することができます。

2.4.2 テキストビューの色を設定する属性

テキストビューは、TextView 要素が持っている、次の二つの属性に設定された色を使って画面に表示されます。

android:textColor	文字の色。
android:background	背景の色。

これらの属性には、#fff というような色の記述を設定することもできますし、リソースとして定義された色を参照する記述を設定することもできます。色を参照する記述は、

@color/ リソース名

と書きます。たとえば、hitode という名前の色を参照する記述は、

```
@color/hitode
```

と書きます。

2.4.3 リソースとして定義した色を扱うアプリケーションの例

それでは、リソースとして色を定義して、その色を使ってテキストビューを表示するアプリケーションを作ってみましょう。

まず、次のようなプロジェクトを作成してください。

Project name	color
Application name	色

Package name　　org.example.color

Create Activity　　ColorActivity

色を定義するリソースファイルは、自動的には生成されませんので、手動で作成する必要があります。作成する場所は、

res/values

の下で、ファイル名は任意です（ただし、XML 文書ですので、拡張子は .xml にします）。

それでは、色を定義するリソースファイルを作りましょう。パッケージエクスプローラの中に表示されている、

res/values

というフォルダを右クリックして、

[New]→[File]

を選択すると、New File というダイアログが開きますので、colors.xml というファイル名を入力して、Finish をクリックしてください。そうすると、ファイルが新しく作られて、それを編集するエディターの画面が開きますので、その中に次のように書いてください。

色を定義する XML 文書の例　colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="tcolor">#00f</color>
    <color name="bcolor">#fff</color>
</resources>
```

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例　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:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/tcolor"
    android:background="@color/bcolor"
    android:text="私はテキストビューです。"
    />
</LinearLayout>
```

これで完成です。実行してみてください。リソースとして定義した色で、テキストビューが表示されるはずです。

2.5 大きさ

2.5.1 大きさのリソースファイル

大きさも、文字列や色と同じように、resources という要素型の要素をルート要素とする XML 文書を書くことによって、リソースとして定義することができます。

ひとつの大きさは、ひとつの dimen 要素によって定義されます。定義したい大きさの記述を要素の内容として書いて、その大きさに与えたい名前を name 属性の値として書きます。

大きさは、10 進数の右側に単位を書いたもので記述します。単位としては、次の 6 種類のうちのいずれかを使うことができます。

mm　ミリメートル (millimeter)。

in　インチ (inch)。1 インチは、約 25.4 ミリメートル。

pt　ポイント (point)。1 ポイントは、72 分の 1 インチ、約 0.35 ミリメートル。

px　ピクセル (pixel)。画素（画面を構成している正方形の単位）の一辺の長さ。画面の解像度 (resolution) によって変化します。

dp　密度非依存ピクセル (density-independent pixel)。解像度が 160dpi であるような画面を構成している画素の一辺の長さ。この単位を使って大きさを指定すれば、画面の解像度にかかわらず常に一定の大きさになります。

sp　倍率非依存ピクセル (scale-independent pixel)。密度非依存ピクセルと同様に、画面の解像度の影響を受けません。さらに、この単位は、ユーザーが設定している文字の大きさに応じて拡大されます。つまり、この単位を使って文字の大きさを指定すれば、ユーザーにとって望ましい大きさの文字が表示されるということです。

たとえば、

```
<dimen name="umiushi">16sp</dimen>
```

という要素を書くことによって、16sp という大きさを、umiushi という名前で定義することができます。

2.5.2　文字の大きさを設定する属性

テキストビューの上に表示される文字の大きさは、TextView 要素が持っている、

```
android:textSize
```

という属性に設定された大きさによって決定されます。

この属性には、直接、20sp というような大きさの記述を設定することもできますし、リソースとして定義された大きさを参照する記述を設定することもできます。大きさを参照する記述は、

```
@dimen/ リソース名
```

と書きます。たとえば、umiushi という名前の大きさを参照する記述は、

```
@dimen/umiushi
```

と書きます。

2.5.3　リソースとして定義した大きさを扱うアプリケーションの例

それでは、リソースとして大きさを定義して、その大きさの文字を使ってテキストビューを表示するアプリケーションを作ってみましょう。

まず、次のようなプロジェクトを作成してください。

Project name	dimen
Application name	大きさ
Package name	org.example.dimen
Create Activity	DimenActivity

大きさを定義するリソースファイルは、自動的には生成されませんので、手動で作成する必要があります。作成する場所は、

```
res/values
```

の下で、ファイル名は任意です（ただし、XML 文書ですので、拡張子は .xml にします）。

それでは、大きさを定義するリソースファイルを作りましょう。

```
res/values
```

というフォルダの下に、dimens.xml というファイルを作って、その中に次のように書いてください。

大きさを定義する XML 文書の例　dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="tsize">64sp</dimen>
</resources>
```

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例　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:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="@dimen/tsize"
    android:text="私はテキストビューです。"
    />
</LinearLayout>
```

これで完成です。実行してみてください。リソースとして定義した大きさの文字を使ってテキストビューが表示されるはずです。

第 3 章　ユーザーインターフェース

3.1　ユーザーインターフェースの基礎

3.1.1　この章について

第 1.3 節で説明したように、人間によって操作されるコンピュータのソフトは、人間から指示を受け取ったり、人間に情報を提供したりするための部分を持っています。そのような部分は、「ユーザーインターフェース」(user interface) と呼ばれます。

この章では、Android アプリケーションのユーザーインターフェースを構築する方法について説明していきたいと思います。

3.1.2　ウィジェットの名前

レイアウト XML の中に要素を書くことによってウィジェットを作った場合、Java のソースの中でそのウィジェットを取り扱うためには、そのウィジェットを取得するという処理をソースの中に記述する必要があります。そして、Java のソースの中で、レイアウト XML の中の要素によって作られたウィジェットを取得するためには、そのウィジェットに対して名前を与えておく必要があります。

ウィジェットに対して名前を与えたいときは、レイアウト XML の中でウィジェットを記述するときに、

```
android:id
```

という属性に対して、

```
@+id/ 名前
```

という形式の文字列を設定します。そうすると、「名前」のところに書かれた文字列がウィジェットに対して名前として与えられます。たとえば、

```
<TextView android:id="@+id/namako"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="私は namako です。"
    />
```

という要素を書くことによって、namako という名前を持つテキストビューを作ることができます。

3.1.3　ウィジェットの ID

Java のソースの中で、レイアウト XML の中の要素によって作られたウィジェットを取得するためには、そのウィジェットを識別する整数が必要です。ウィジェットを識別する整数は、ウィジェットの「ID」と呼ばれます。

Java のソースコードの中で、

```
R.id.名前
```

という形の式を評価すると、その値として、その中に書かれた名前を持つウィジェットの ID が得られます。たとえば、

```
R.id.namako
```

という式を評価することによって得られる整数は、namako という名前を持つテキストビューの ID です。

3.1.4　ウィジェットの取得

Java のソースの中でウィジェットを取得したいときは、アクティビティーが持っている、

```
View findViewById(int id)
```

というメソッドを使います。このメソッドは、引数としてウィジェットの ID を受け取って、その ID によって識別されるウィジェットを戻り値として返します。ただし、戻り値の型は、ウィジェットの種類ごとのクラスではなくて、

```
android.view.View
```

というスーパークラスですので、戻り値をダウンキャストすることが必要になります。

ウィジェットのクラス名は、ウィジェットを作る要素の要素型名と同じです。たとえば、テキストビューのクラス名は、

```
android.widget.TextView
```

ですので、

```
final TextView tv = (TextView) findViewById(R.id.textview);
```

という宣言を書くことによって、textview という定数であらわされる ID を持つテキストビューを、tv という変数に設定することができます。

3.1.5　テキストビューの上に表示される文字列の変更

テキストビューの上に表示される文字列は、アプリケーションの実行中に変更することが可能です。

テキストビューの上に表示される文字列を変更したいときは、テキストビューが持っている、

- `void setText(CharSequence text)`
- `void setText(int resid)`

というメソッドを使います。これらのメソッドのうちのどちらかを呼び出して、文字列、または文字列のリソース ID を引数として渡すと、テキストビューは、引数で指定された文字列を自分の上に表示します。たとえば、テキストビューが tv という変数に設定されているとするとき、

```
tv.setText("私は文字列です。");
```

という式文を実行することによって、そのテキストビューの上に、「私は文字列です。」という文字列を表示することができます。

3.2　ボタン

3.2.1　ボタンの作り方

ボタン (button) を作りたいときは、レイアウト XML の中に、Button という要素型の要素を書きます。たとえば、

```
<Button android:id="@+id/dosomething"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="何かする"
    />
```

という Button 要素を書くことによって、「何かする」という文字列が表示されたボタンを作ることができます。

3.2.2 イベントを処理する方法

ユーザーによる操作などによって生じる、ボタンがクリックされたというような出来事は、「イベント」(event) と呼ばれます。

ウィジェットの上でイベントが発生したときに、そのイベントに応じた処理をするためには、「イベントリスナー」(event listener) と呼ばれるオブジェクトを作って、それをウィジェットに設定しておく必要があります。イベントリスナーというのは、イベントが発生したときに呼び出されるメソッドを持っているオブジェクトのことです。

3.2.3 イベントリスナーの作り方

イベントリスナーを作りたいときは、イベントリスナーのインターフェースを実装したクラスを定義して、そのクラスのインスタンスを生成します。

ウィジェットがクリックされたというイベントを処理するイベントリスナーは、

```
android.view.View.OnClickListener
```

というインターフェースを実装することによって作ります。このインターフェースが定義しているのは、

```
void onClick(View v)
```

というメソッドです。ウィジェットがクリックされたというイベントが発生すると、このメソッドが呼び出されて、クリックされたウィジェットが引数として渡されます。たとえば、

```
public void onClick(View v) {
    final TextView tv = (TextView) findViewById(R.id.textview);
    tv.setText("ウィジェットがクリックされました。");
}
```

と `onClick` を定義しておいたとすると、ウィジェットがクリックされたとき、`textview` という定数であらわされる ID を持つテキストビューの上に、「ウィジェットがクリックされました。」という文字列が表示されることになります。この例から分かるとおり、`findViewById` というメソッドは、`onClick` の中から呼び出すことも可能です。

3.2.4 イベントリスナーの設定

ウィジェットの上でイベントが発生したときに、そのイベントを処理するためには、イベントリスナーをそのウィジェットに設定する必要があります。

クリックされたというイベントを処理するイベントリスナーをウィジェットに設定したいときは、そのウィジェットが持っている、

```
void setOnClickListener(OnClickListener l)
```

というメソッドを呼び出して、イベントリスナーを引数として渡します。

3.2.5 ボタンを使ったアプリケーションの例

それでは、ボタンを使った Android アプリケーションの例として、ボタンがクリックされるたびに、テキストビューに表示された数字が 1 ずつ増加していく、というものを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`increase`
Application name	ボタンをクリックすると増加
Package name	`org.example.increase`
Create Activity	`IncreaseActivity`

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例 `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/number"
```

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="0"
    />
<Button android:id="@+id/increase"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="増加"
    />
</LinearLayout>
```

次に、IncreaseActivity.javaを次のように書き換えてください。

プログラムの例 IncreaseActivity.java

```
package org.example.increase;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;
import android.widget.Button;

public class IncreaseActivity extends Activity {
    private int n = 0;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button increase = (Button) findViewById(R.id.increase);
        increase.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                increaseNumber();
            }
        });
    }

    private void increaseNumber() {
        final TextView number = (TextView) findViewById(R.id.number);
        number.setText(Integer.toString(++n));
    }
}
```

これで完成です。実行して、ボタンをクリックしてみてください。クリックするたびに、テキストビューに表示された数字が1ずつ増加していくはずです。

3.3 エディットテキスト

3.3.1 エディットテキストの作り方

文字列を編集するために使われるウィジェットは、「エディットテキスト」(edit text)と呼ばれます。

エディットテキストを作りたいときは、レイアウトXMLの中に、EditTextという要素型の要素を書きます。

エディットテキストを作る場合には、その使い方に応じて、さまざまな属性に対して属性値を設定する必要があります。エディットテキストに関連する属性としては、次のようなものがあります。

android:scrollbars	この属性に対してverticalという属性値を設定することによって、垂直方向のスクロールバーを表示することができます。
android:gravity	この属性に対してtopという属性値を設定することによって、表示される文字列の垂直方向の位置を上に寄せることができます。

android:singleLine	この属性に対して true という属性値を設定することによって、1 行だけしか入力できないようにすることができます。
android:numeric	この属性に対して decimal という属性値を設定することによって、10 進数の数字と小数点だけしか入力できないようにすることができます。
android:maxLength	この属性に対してプラスの整数を設定すると、それが、入力することのできる最大の文字数になります。

3.3.2 文字列の設定と取得

テキストビューと同じように、エディットテキストに対しても、

- void setText(CharSequence text)
- void setText(int resid)

というメソッドを使うことによって、文字列を設定することができます。

エディットテキストから文字列を取得したいときは、それが持っている、

```
Editable getText()
```

というメソッドを呼び出します。ただし、このメソッドが戻り値として返すオブジェクトは、String クラスのインスタンスではありません。それを String クラスのインスタンスに変換したいときは、それが持っている、

```
String toString()
```

というメソッドを呼び出します。

3.3.3 エディットテキストを使ったアプリケーションの例 (1)

それでは、エディットテキストを使った Android アプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	edittext
Application name	エディットテキスト
Package name	org.example.edittext
Create Activity	EditTextActivity

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例 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/edittext1"
    android:layout_width="fill_parent"
    android:layout_height="140sp"
    android:scrollbars="vertical"
    android:gravity="top"
    />
<Button android:id="@+id/forward"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="転送"
    />
<EditText android:id="@+id/edittext2"
    android:layout_width="fill_parent"
    android:layout_height="140sp"
    android:scrollbars="vertical"
    android:gravity="top"
    />
</LinearLayout>
```

次に、EditTextActivity.java を次のように書き換えてください。

プログラムの例 EditTextActivity.java

```
package org.example.edittext;

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

public class EditTextActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button forward = (Button) findViewById(R.id.forward);
        forward.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                forwardText();
            }
        });
    }

    private void forwardText() {
        final EditText edittext1 = (EditText) findViewById(R.id.edittext1);
        final EditText edittext2 = (EditText) findViewById(R.id.edittext2);
        edittext2.setText(edittext1.getText().toString());
    }
}
```

これで完成です。実行すると、二つのエディットテキストとひとつボタンが表示されますので、上のエディットテキストに文字列を入力して、ボタンをクリックしてみてください。そうすると、上のエディットテキストに入力した文字列が、下のエディットテキストに転送されるはずです。

3.3.4 エディットテキストを使ったアプリケーションの例 (2)

エディットテキストを使った Android アプリケーションを、もうひとつ作ってみましょう。今度は、平方根を求めるアプリケーションです。

先ほど作ったアプリケーションでは、どんな文字列でも入力することができるエディットテキストを使いましたが、今度のアプリケーションでは、10 進数の数字または小数点から構成される 1 行の文字列だけを入力することのできるエディットテキストを使うことにします。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`sqrt`
Application name	平方根
Package name	`org.example.sqrt`
Create Activity	`SqrtActivity`

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例 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/number"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:singleLine="true"
    android:numeric="decimal"
    />
```

```
<Button android:id="@+id/sqrt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="平方根"
    />
<TextView android:id="@+id/result"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、SqrtActivity.javaを次のように書き換えてください。

プログラムの例 SqrtActivity.java

```
package org.example.sqrt;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.widget.TextView;

public class SqrtActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button sqrt = (Button) findViewById(R.id.sqrt);
        sqrt.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showSqrt();
            }
        });
    }

    private void showSqrt() {
        final EditText number = (EditText) findViewById(R.id.number);
        double a = Double.parseDouble(number.getText().toString());
        final TextView result = (TextView) findViewById(R.id.result);
        result.setText(a + "の平方根は" + Math.sqrt(a) + "です。");
    }
}
```

これで完成です。実行して、エディットテキストに数値を入力して、ボタンをクリックしてみてください。入力した数値の平方根が表示されるはずです。

3.4 チェックボックス

3.4.1 チェックボックスの作り方

チェックボックス (check box) を作りたいときは、レイアウト XML の中に、CheckBoxという要素型の要素を書きます。たとえば、

```
<CheckBox android:id="@+id/luxury"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="豪華なオプション"
    />
```

というCheckBox要素を書くことによって、「豪華なオプション」という文字列が表示されたチェックボックスを作ることができます。

3.4.2　チェックの判定

チェックボックスにチェックが入っているかどうかを判定したいときは、チェックボックスが持っている、

```
boolean isChecked()
```

というメソッドを使います。このメソッドは、チェックボックスにチェックが入っているならば真を返し、入っていないならば偽を返します。

3.4.3　チェックボックスを使ったアプリケーションの例

それでは、チェックボックスを使った Android アプリケーションの例として、文字列の長さを求めるというものを作ってみましょう。このアプリケーションは、文字列の長さ、つまり文字列を構成している文字の個数を求めるわけですが、チェックボックスにチェックが入っている場合は、空白を除いた文字の個数を求めます。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`length`
Application name	文字列の長さ
Package name	`org.example.length`
Create Activity	`LengthActivity`

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例　`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/edittext"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:singleLine="true"
    />
<CheckBox android:id="@+id/except"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="空白は数えない。"
    />
<Button android:id="@+id/length"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="長さ"
    />
<TextView android:id="@+id/result"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、`LengthActivity.java` を次のように書き換えてください。

プログラムの例　`LengthActivity.java`

```
package org.example.length;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.CheckBox;
import android.widget.Button;
import android.widget.TextView;
```

```
public class LengthActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button length = (Button) findViewById(R.id.length);
        length.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                lengthOfString();
            }
        });
    }

    private void lengthOfString() {
        final EditText edittext = (EditText) findViewById(R.id.edittext);
        final String s = edittext.getText().toString();
        final CheckBox except = (CheckBox) findViewById(R.id.except);
        final TextView result = (TextView) findViewById(R.id.result);
        if (except.isChecked())
            result.setText(Integer.toString(lengthExceptSpace(s)));
        else
            result.setText(Integer.toString(s.length()));
    }

    private int lengthExceptSpace(String s) {
        int len = 0;
        for (int i=0; i<s.length(); i++)
            if (s.charAt(i) != ' ')
                len++;
        return len;
    }
}
```

これで完成です。空白を含む文字列を入力して、ボタンをクリックしてみてください。チェックボックスにチェックが入っていない場合は、空白も含めて文字を数えた結果が表示されて、チェックが入っている場合は、空白以外の文字を数えた結果が表示されます。

3.5 ラジオボタン

3.5.1 ラジオボタンの作り方

ラジオボタン (radio button) を作りたいときは、レイアウト XML の中に、`RadioButton` という要素型の要素を書きます。たとえば、

```
<RadioButton android:id="@+id/underground"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="アングラ放送局"
    />
```

という `RadioButton` 要素を書くことによって、「アングラ放送局」という文字列が表示されたラジオボタンを作ることができます。

3.5.2 ラジオグループ

ラジオボタンというのは、チェックボックスと同様に、チェックが入っている状態と入っていない状態という二つの状態を持つことのできるウィジェットです。

チェックボックスとラジオボタンとの相違点は、前者が単独で使われるものなのに対して、後者はグループで使われるものだという点にあります。ひとつのグループに所属しているそれぞれのラジオボタンは互いに連動していて、ひとつをクリックすると、それまでチェックが入っていたラジオボタンはチェックが外れます。そのようにして、グループを構成しているラジオボタンは、常にひとつのラジオボタンだけにチェックが入っているように管理されます。

Android では、「ラジオグループ」(radio group) というウィジェットを使うことによって、ラ

ジオボタンのグループを作ることができます。

3.5.3 ラジオグループの作り方

ラジオグループを作りたいときは、レイアウト XML の中に、RadioGroup という要素型の要素を書きます。

RadioGroup 要素は、android:orientation という属性を持っています。これは、グループに所属するラジオボタンをどの方向に並べるかということを意味する属性です。この属性に対して設定することのできるのは、次のどちらかの文字列です。

horizontal	水平方向に並べる。デフォルト。
vertical	垂直方向に並べる。

たとえば、

```
<RadioGroup android:id="@+id/emptygroup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    >
</RadioGroup>
```

という RadioGroup 要素を書くことによって、空のラジオグループを作ることができます。

RadioGroup 要素は、その子供として RadioButton 要素を何個でも書くことができます。子供のラジオボタンは、親のラジオグループに所属することになります。

3.5.4 ラジオグループの初期設定

ラジオグループは、初期状態では、それを構成しているラジオボタンのうちのどれにもチェックが入っていない状態になっています。ですから、ラジオグループを使う場合は、初期設定として、ラジオボタンの一つにチェックを入れるという処理が必要になります。

ラジオボタンにチェックを入れたいときは、ラジオグループが持っている、

```
void check(int id)
```

というメソッドを使います。このメソッドは、ラジオボタンの ID を引数として受け取って、そのラジオボタンにチェックを入れます。たとえば、group という変数がラジオグループを指しているとするとき、

```
group.check(R.id.underground)
```

という式を評価することによって、underground という定数であらわされる ID を持つラジオボタンにチェックを入れることができます。

3.5.5 選択されたラジオボタンの ID

ラジオグループを構成しているラジオボタンのうちのどれが選択されているのか（どれにチェックが入っているのか）ということを調べたいときは、ラジオグループが持っている、

```
int getCheckedRadioButtonId()
```

というメソッドを使います。このメソッドは、選択されているラジオボタンの ID を返します。

3.5.6 ラジオボタンを使ったアプリケーションの例

それでは、ラジオボタンを使った Android アプリケーションの例として、挨拶を表示するというものを作ってみましょう。これは、ボタンをクリックすると挨拶を表示するというアプリケーションです。このアプリケーションは、ラジオボタンで、朝、昼、晩のどれかから時間帯を選択することができて、選択された時間帯に応じた挨拶を表示します。

まず最初に、次のようなプロジェクトを作成してください。

Project name	greet
Application name	挨拶
Package name	org.example.greet
Create Activity	GreetActivity

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例 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"
    >
<RadioGroup android:id="@+id/time"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    >
    <RadioButton android:id="@+id/morning"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="朝"
    />
    <RadioButton android:id="@+id/afternoon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="昼"
    />
    <RadioButton android:id="@+id/night"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="晩"
    />
</RadioGroup>
<Button android:id="@+id/greet"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="挨拶"
    />
<TextView android:id="@+id/result"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、GreetActivity.java を次のように書き換えてください。

プログラムの例 GreetActivity.java

```
package org.example.greet;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.RadioGroup;
import android.widget.Button;
import android.widget.TextView;

public class GreetActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final RadioGroup time = (RadioGroup) findViewById(R.id.time);
        time.check(R.id.morning);
        final Button greet = (Button) findViewById(R.id.greet);
        greet.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                setGreet();
            }
```

```
        });
    }

    private void setGreet() {
        final RadioGroup time = (RadioGroup) findViewById(R.id.time);
        final TextView result = (TextView) findViewById(R.id.result);
        switch (time.getCheckedRadioButtonId()) {
        case R.id.morning:
            result.setText("おはようございます。");
            break;
        case R.id.afternoon:
            result.setText("こんにちは。");
            break;
        case R.id.night:
            result.setText("こんばんは。");
            break;
        }
    }
}
```

これで完成です。ラジオボタンで時間帯を選択して、ボタンをクリックしてみてください。選択した時間帯に応じた挨拶が表示されるはずです。

3.6 リストビュー

3.6.1 リストビューの基礎

Android では、いくつかのオブジェクトの中から、どれかをユーザーに選択してほしいときに、「リストビュー」(list view) と呼ばれるウィジェットを使うことができます。リストビューは、

```
android.widget.ListView
```

というクラスのオブジェクトです。

リストビューは、任意のクラスのインスタンスから構成されるリストを画面に表示することができます。ユーザーは、項目のひとつをクリックすることによって、それに対応するオブジェクトを選択することができます。

リストビューによって表示されるリストのそれぞれの項目の上には、その項目のオブジェクトが持っている、

```
String toString()
```

というメソッドが返した文字列が表示されます。ですから、望ましい文字列が表示されるようにするためには、項目のオブジェクトのクラスで、このメソッドをオーバーライドしておく必要があります。

3.6.2 リストビューのためのアダプター

リストビューを使うためには、リストの項目となるオブジェクトとリストビューとのあいだの橋渡しをする、「アダプター」(adapter) と呼ばれるオブジェクトを作る必要があります。

アダプターは、

```
android.widget.BaseAdapter
```

というクラスのオブジェクトですが、ここでは、そのサブクラスとして定義されている、

```
android.widget.ArrayAdapter
```

というクラスを使って、アダプターの作り方を説明したいと思います。

`ArrayAdapter` は、1 個の型引数を受け取る総称型ですので、そのインスタンスを生成する段階で、ひとつの型を引数として渡す必要があります。渡さないといけないのは、アダプターを構成する項目（リストを構成する項目と同じ）の型です。たとえば、`Person` というクラスのインスタンスから構成されるアダプターを作りたいとするならば、

```
ArrayAdapter<Person> adapter =
    new ArrayAdapter<Person>(this,
        android.R.layout.simple_list_item_1);
```

というように、型として Person を渡します。ちなみに、

```
simple_list_item_1
```

というのは、リストの項目のために最初から定義されているレイアウトのひとつです。
次に、アダプターが持っている add というメソッドを呼び出して、アダプターに項目を追加します。Person クラスのインスタンスを項目とするアダプターの場合、

```
adapter.add(new Person("夏目なつみ"));
```

というように書くことによって、Person クラスのインスタンスを項目としてアダプターに追加することができます。
そして、リストビューにアダプターを設定すれば、アダプターに追加された項目から構成されるリストが、リストビューによって表示されることになります。リストビューに対してアダプターを設定したいときは、リストビューが持っている、

```
void setAdapter(ListAdapter adapter)
```

というメソッドを呼び出します。

3.6.3 リストビューのイベントリスナー

リストビューに対してイベントリスナーを設定しておくことによって、リストビューによって表示されているリストの項目がクリックされたときに何らかの動作を実行する、ということができます。
リストの項目がクリックされたときに何らかの動作を実行するイベントリスナーは、

```
android.widget.AdapterView.OnItemClickListener
```

というインターフェースを実装したクラスのインスタンスです。そして、このインターフェースを実装するために定義する必要があるのは、

```
void onItemClick(AdapterView<?> parent, View view,
    int position, long id)
```

というメソッドです。

3.6.4 クリックされた項目の取得

ところで、onItemClick の中で、クリックされた項目を取得したいときは、どうすればいいのでしょうか。
onItemClick は 4 個の引数を受け取るわけですが、クリックされた項目は、それらの引数のうちの 1 個目と 3 個目を使うことによって取得することができます。1 個目の引数は、イベントを発生させたリストビューで、2 個目の引数は、クリックされた項目の番号です（番号は、追加した順番のとおりに、0、1、2、……と与えられます）。
クリックされた項目を取得したいときは、まず、リストビューが持っている、

```
ListAdapter getAdapter()
```

というメソッドを呼び出すことによって、そのリストビューに設定されているアダプターを取得します。そして次に、アダプターが持っている、

```
Object getItem(int position)
```

というメソッドを呼び出して、クリックされた項目の番号を引数として渡すことによって、クリックされた項目を取得します。

3.6.5 リストビューを使ったアプリケーションの例

それでは、リストビューを使ったアプリケーションを作ってみましょう。
まず最初に、次のようなプロジェクトを作成してください。

Project name	`listview`
Application name	リストビュー
Package name	`org.example.listview`
Create Activity	`ListViewActivity`

次に、main.xml を次のように書き換えてください。

レイアウト XML の例 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"
    >
<ListView android:id="@+id/listview"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<TextView android:id="@+id/selecteditem"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、ListViewActivity.java を次のように書き換えてください。

プログラムの例 ListViewActivity.java

```
package org.example.listview;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.AdapterView;
import android.widget.TextView;

public class ListViewActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayAdapter<Dish> adapter =
            new ArrayAdapter<Dish>(this,
                android.R.layout.simple_list_item_1);
        adapter.add(new Dish("焼きそば", 480));
        adapter.add(new Dish("カツ丼", 540));
        adapter.add(new Dish("カレーうどん", 460));
        adapter.add(new Dish("オムライス", 520));
        final ListView listview = (ListView) findViewById(R.id.listview);
        listview.setAdapter(adapter);
        listview.setOnItemClickListener(
            new AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> parent,
                        View view, int position, long id) {
                    setSelectedItem(parent, position);
                }
            });
    }

    private void setSelectedItem(AdapterView<?> parent, int position) {
        Dish dish = (Dish) parent.getAdapter().getItem(position);
        final TextView selecteditem =
            (TextView) findViewById(R.id.selecteditem);
        selecteditem.setText(
            "オーダー = " + dish.getName() + "\n" +
            "金額 = " + dish.getPrice() + "円");
    }
}
```

```
class Dish {
    String name;
    int price;
    Dish(String aname, int aprice) {
        name = aname;
        price = aprice;
    }
    public String toString() {
        return name + "（" + price + "円）";
    }
    public String getName() {
        return name;
    }
    public int getPrice() {
        return price;
    }
}
```

これで完成です。実行すると、リストビューによってリストが表示されるはずですので、項目のどれかをクリックしてみてください。そうすると、クリックされた項目についてのメッセージが表示されるはずです。

3.7 スピナー

3.7.1 スピナーの基礎

Android は、「スピナー」(spinner) と呼ばれるウィジェットを持っています。スピナーは、リストビューと同じように、いくつかのオブジェクトの中から、どれかをユーザーに選択してほしいときに使うことのできるものです。スピナーは、

```
android.widget.Spinner
```

というクラスのオブジェクトです。

リストビューは、選択の対象となるリストの項目を画面の上にすべて表示するわけですが、それに対してスピナーは、現時点で選択されているひとつの項目しか表示しません。選択されている項目を変更したいときは、スピナーの右端にある逆三角形のボタンをクリックします。そうすると、「ドロップダウンリスト」(drop-down list) と呼ばれるリストが表示されますので、そのリストの項目のうちで、選択されているものとは異なるものをクリックすれば、その項目が選択されることになります。現時点で選択されている項目を変更しないでリストを閉じたいときは、現時点で選択されている項目をクリックするか、[戻る] ボタンをクリックします。

リストビューの場合と同じように、スピナーの上や、逆三角形のボタンをクリックしたときに表示されるリストのそれぞれの項目の上に、望ましい文字列が表示されるようにするためには、その項目のオブジェクトのクラスで、

```
String toString()
```

というメソッドをオーバーライドしておく必要があります。

3.7.2 スピナーのためのアダプター

スピナーを使うためには、リストビューの場合と同じように、リストの項目となるオブジェクトとスピナーとのあいだの橋渡しをしてもらうために、アダプターを作る必要があります。

スピナーのためのアダプターの作り方は、リストビューの場合とほとんど同じです。たとえば、

```
ArrayAdapter<Person> adapter =
    new ArrayAdapter<Person>(this,
        android.R.layout.simple_spinner_item);
```

と書くことによって、Person というクラスのインスタンスから構成されるアダプターを作ることができます。ちなみに、

```
simple_spinner_item
```

というのは、スピナーの上に表示される項目のために最初から定義されているレイアウトです。

アダプターに対しては、スピナーの上に表示される項目のためのレイアウトだけではなくて、ドロップダウンリストの項目のためのレイアウトも、設定する必要があります。
ドロップダウンリストの項目のためのレイアウトをアダプターに設定したいときは、

```
void setDropDownViewResource(int resource)
```

というメソッドを使います。たとえば、

```
adapter.setDropDownViewResource(
    android.R.layout.simple_spinner_dropdown_item);
```

と書くことによって、

```
simple_spinner_dropdown_item
```

という、ドロップダウンリストの項目のために最初から定義されているレイアウトをアダプターに設定することができます。
アダプターに項目を追加する方法や、スピナーに対してアダプターを設定する方法は、リストビューの場合と同じです。

3.7.3　スピナーのイベントリスナー

スピナーに対してイベントリスナーを設定しておくことによって、スピナーによって選択されている項目が変更されたときに何らかの動作を実行する、ということができます。
選択されている項目が変更されたときに何らかの動作を実行するイベントリスナーは、

```
android.widget.AdapterView.OnItemSelectedListener
```

というインターフェースを実装したクラスのインスタンスです。そして、このインターフェースを実装するために定義する必要があるのは、

- `void onItemSelected(AdapterView<?> parent, View view, int position, long id)`
- `void onNothingSelected(AdapterView<?> parent)`

という二つのメソッドです。
選択されている項目が変更されたときに実行したい動作は、`onItemSelected` の中に書きます。このメソッドが受け取る引数の使い方は、第 3.6 節で説明した、`onItemClick` の引数の使い方と同じです。なお、このメソッドは、選択されている項目が変更されたときだけではなくて、1 個以上の項目を持つアダプターがスピナーに設定されたときにも呼び出されます。

3.7.4　デフォルトの項目の設定

スピナーは、アダプターが設定された直後は、0 という番号を持つ項目、つまりアダプターに最初に追加した項目が選択されている状態になっています。ユーザーに項目を選択してもらう前に、デフォルトの項目として 0 番以外の項目を設定しておきたいときは、

```
void setSelection(int position)
```

というメソッドを使います。このメソッドを呼び出して、引数として項目の番号を渡すと、スピナーは、その項目が選択されている状態に設定されます。

3.7.5　スピナーを使ったアプリケーションの例

それでは、スピナーを使ったアプリケーションを作ってみましょう。
まず最初に、次のようなプロジェクトを作成してください。

Project name	`spinner`
Application name	スピナー
Package name	`org.example.spinner`
Create Activity	`SpinnerActivity`

次に、`main.xml` を次のように書き換えてください。

レイアウト XML の例　`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"
    >
<Spinner android:id="@+id/spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<TextView android:id="@+id/selecteditem"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、SpinnerActivity.javaを次のように書き換えてください。

プログラムの例 SpinnerActivity.java

```
package org.example.spinner;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.AdapterView;
import android.widget.TextView;

public class SpinnerActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayAdapter<Man> adapter =
            new ArrayAdapter<Man>(this,
                android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(
            android.R.layout.simple_spinner_dropdown_item);
        adapter.add(new Man("源義経", 526));
        adapter.add(new Man("足利尊氏", 876));
        adapter.add(new Man("織田信長", 1200));
        adapter.add(new Man("土方歳三", 361));
        final Spinner spinner = (Spinner) findViewById(R.id.spinner);
        spinner.setAdapter(adapter);
        spinner.setSelection(2);
        spinner.setOnItemSelectedListener(
            new AdapterView.OnItemSelectedListener() {
                @Override
                public void onItemSelected(AdapterView<?> parent,
                        View view, int position, long id) {
                    setSelectedItem(parent, position);
                }
                public void onNothingSelected(AdapterView<?> parent) {
                }
            });
    }

    private void setSelectedItem(AdapterView<?> parent, int position) {
        Man man = (Man) parent.getAdapter().getItem(position);
        final TextView selecteditem =
            (TextView) findViewById(R.id.selecteditem);
        selecteditem.setText(
            "本命の彼氏 = " + man.getName() + "\n" +
            "年収 = " + man.getIncome() + "万円");
    }
}
```

```
class Man {
    String name;
    int income;
    Man(String aname, int aincome) {
        name = aname;
        income = aincome;
    }
    public String toString() {
        return name + "（年収" + income + "万円）";
    }
    public String getName() {
        return name;
    }
    public int getIncome() {
        return income;
    }
}
```

これで完成です。実行すると、スピナーが表示されて、それによって選択されているオブジェクトについてのメッセージが、その下に表示されるはずですので、スピナーの右端にある逆三角形のボタンをクリックしてみてください。そうすると、ドロップダウンリストが表示されるはずですので、選択されている項目以外の項目をクリックしてみてください。そうすると、スピナーで選択されている項目が変更されて、その下のメッセージも変化するはずです。

3.8 レイアウト

3.8.1 レイアウトとは何か

Android では、画面の上にウィジェットを配置するために使われるオブジェクトのことを「レイアウト」(layout) と呼びます。

レイアウトには、リニアレイアウト (linear layout)、テーブルレイアウト (table layout)、絶対レイアウト (absolute layout) など、いくつかの種類があります。

一つのレイアウトは、画面の上に一つの長方形の領域を作って、その中にウィジェットを配置します。

3.8.2 レイアウトの作り方

レイアウトは、レイアウト XML の中に、レイアウトの要素を書くことによって作ることができます。レイアウトの要素の要素型は、レイアウトの種類ごとに決まっています。

レイアウトを使ってウィジェットを配置したいときは、そのレイアウトの要素の子供として、それらのウィジェットの要素を書きます。つまり、レイアウトは、自分の子供として書かれているウィジェットを自分の内部に配置することになるわけです。

レイアウトの要素の子供としては、ウィジェットの要素だけではなくて、レイアウトの要素を書くことも可能です。レイアウトは、自分の子供になっているレイアウトも、ウィジェットと同じように自分の内部に配置します。

3.8.3 レイアウトの大きさ

レイアウトを作るときは、ウィジェットを作るときと同じように、その大きさを決めるための次の二つの属性に値を設定する必要があります。

`android:layout_width` 横の長さ。
`android:layout_height` 縦の長さ。

ウィジェットの場合と同じように、これらの属性には、具体的な長さを設定することもできますし、次の単語を設定することもできます。

`fill_parent` 親の長さに合わせる。
`wrap_content` 内容の長さに合わせる。

3.8.4 リニアレイアウト

リニアレイアウトは、ウィジェットを一列に並べるレイアウトです。並べる方向は、垂直または水平のどちらかです。

プロジェクトを作成したときに自動的に生成されるレイアウト XML は、リニアレイアウトを使っています。

リニアレイアウトは、`LinearLayout` という要素型の要素を書くことによって作ることができます。

`LinearLayout` 要素は、`android:orientation` という属性を持っています。これは、ウィジェットを配置する方向を指定するための属性で、次のどちらかの値を設定することができます。

`horizontal`	水平方向。デフォルト。
`vertical`	垂直方向。

リニアレイアウトを構成するウィジェットは、要素を書いた順番のとおりに、方向が水平の場合は左から右へ、方向が垂直の場合は上から下へ並びます。

リニアレイアウトは、これまでに作成したアプリケーションでも使われていますが、これまでは、垂直方向にウィジェットを並べるという使い方しかしていませんでした。そこでここでは、水平方向にもウィジェットを並べるアプリケーションを作ってみることにしましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`linear`
Application name	リニアレイアウト
Package name	`org.example.linear`
Create Activity	`LinearActivity`

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例　`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:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ボタン 1"
    />
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ボタン 2"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ボタン 3"
        />
</LinearLayout>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ボタン 4"
    />
</LinearLayout>
```

これで完成です。ボタン 2 とボタン 3 は、水平方向に並んでいるはずです。

3.8.5　テーブルレイアウト

テーブルレイアウトは、水平と垂直の両方向にウィジェットを整列させるレイアウト、つまり、二次元の升目の上にウィジェットを並べることによって表の形を作るレイアウトです。

テーブルレイアウトにおいて、水平方向のウィジェットの並びは「行」(row) と呼ばれ、垂直方向のウィジェットの並びは「列」(column) と呼ばれます。

行は、TableRow という要素型の要素を書くことによって作ることができます。この要素の子供として、ウィジェットを作る要素を何個か書くと、それらのウィジェットから構成される行ができます。行を構成するウィジェットは、要素を書いた順番のとおりに、左から右へ並びます。

通常、レイアウト XML でウィジェットを作るときは、

```
android:layout_width
android:layout_height
```

という属性に属性値を設定する必要があるわけですが、行を構成するウィジェットを作る場合、これらの属性に属性値を設定する必要はありません。

テーブルレイアウトは、TableLayout という要素型の要素を書くことによって作ることができます。この要素型の要素の子供として、何個かの TableRow 要素を書けば、それらの TableRow 要素によって作られた行から構成されるテーブルレイアウトができます。テーブルレイアウトを構成する行は、要素を書いた順番のとおりに、上から下へ並びます。

TableLayout 要素は、android:stretchColumns という属性を持っています。これは、特定の列の幅を可能な限り左右に広げる属性です。属性値として列の番号を設定すると、その番号の列の横幅が可能な限り拡張されます（番号をコンマで区切ることによって、複数の番号を設定することもできます)。列の番号は、左から右へ向かって、0、1、2、……というように与えられています。

それでは、テーブルレイアウトを使ったアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	table
Application name	テーブルレイアウト
Package name	org.example.table
Create Activity	TableActivity

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例　main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1"
    >
<TableRow>
    <Button android:text="ボタン 1"/>
    <Button android:text="ボタン 2"/>
    <Button android:text="ボタン 3"/>
</TableRow>
<TableRow>
    <Button android:text="ボタン 4"/>
    <Button android:text="ボタン 5"/>
    <Button android:text="ボタン 6"/>
</TableRow>
</TableLayout>
```

これで完成です。6 個のボタンが、2 行 3 列の表の形で並んでいるはずです。また、1 番目の列（中央の列）の幅が、可能な限り左右に拡張されているはずです。

3.8.6　絶対レイアウト

絶対レイアウトは、個々のウィジェットの絶対的な位置を指定することによってウィジェットを配置するレイアウトです。

ウィジェットの絶対的な位置は、座標を使って指定します。画面の座標系は、画面の左上の隅が原点で、x 軸は右向き、y 軸は下向きです。

絶対レイアウトは、AbsoluteLayout という要素型の要素を書くことによって作ることができます。

ウィジェットの座標は、AbsoluteLayout 要素が持っている次の属性に設定します。

`android:layout_x` x 座標。

`android:layout_y` y 座標。

それでは、絶対レイアウトを使ったアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`absolute`
Application name	絶対レイアウト
Package name	`org.example.absolute`
Create Activity	`AbsoluteActivity`

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例　`main.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="0px"
    android:layout_y="0px"
    android:text="ボタン 1"/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="150px"
    android:layout_y="60px"
    android:text="ボタン 2"/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="100px"
    android:layout_y="200px"
    android:text="ボタン 3"/>
</AbsoluteLayout>
```

これで完成です。3 個のボタンが、座標で指定されたそれぞれの位置に配置されているはずです。

3.9 トースト

3.9.1 トーストの基礎

Android には、一定の時間だけメッセージを画面に表示する、「トースト」(toast) と呼ばれるウィジェットがあります。

トーストを生成したいときは、

```
android.widget.Toast
```

というクラスが持っている、

```
static Toast makeText(Context context, CharSequence text, int duration)
```

というクラスメソッドを呼び出します。そうすると、トーストが生成されて、それがこのメソッドの戻り値になります。このメソッドに渡す引数の 1 個目は this の値、2 個目は表示するメッ

セージ、3 個目は表示の持続時間です。表示の持続時間の指定には、通常、 Toast クラスの中で定義されている次の定数が使われます。

LENGTH_LONG	長い持続時間。
LENGTH_SHORT	短い持続時間。

トーストにメッセージを表示させたいときは、トーストが持っている、 show というメソッドを呼び出します。

3.9.2 トーストを使ったアプリケーションの例

まず最初に、次のようなプロジェクトを作成してください。

Project name	toast
Application name	トースト
Package name	org.example.toast
Create Activity	ToastActivity

次に、 main.xml を次のように書き換えてください。

レイアウト XML の例　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/showtoast"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="トーストを表示する"
    />
</LinearLayout>
```

次に、 ToastActivity.java を次のように書き換えてください。

プログラムの例　ToastActivity.java

```
package org.example.toast;

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

public class ToastActivity extends Activity {
    private Toast toast;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        toast = Toast.makeText(this, "私はトーストです。",
            Toast.LENGTH_SHORT);
        final Button showtoast = (Button) findViewById(R.id.showtoast);
        showtoast.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                toast.show();
            }
        });
    }
}
```

これで完成です。実行して、ボタンをクリックしてみてください。トーストによってメッセージが表示されるはずです。

3.10 アラートダイアログ

3.10.1 アラートダイアログの基礎

GUI を持つアプリケーションが、人間と対話をするために画面に一時的に表示するウィンドウは、「ダイアログ」(dialog) と呼ばれます。

Android では、ダイアログは、

```
android.app.Dialog
```

というクラスから生成されます。しかし、このクラスを使ってダイアログを表示するというのは、けっこう面倒ですので、Android には、ダイアログを簡単に生成することができるように、

```
android.app.AlertDialog.Builder
```

というクラスが準備されています。このクラスを使って生成されたダイアログは、「アラートダイアログ」(alert dialog) と呼ばれます。

3.10.2 肯定ボタンと中立ボタンと否定ボタン

アラートダイアログの上には、最大三つまで、ボタンを左右に並べて表示することができます。

アラートダイアログの上に表示することのできる三つのボタンのそれぞれは、次のような名前で呼ばれます。

- 肯定ボタン (positive button)
- 中立ボタン (neutral button)
- 否定ボタン (negative button)

3.10.3 アラートダイアログの表示

アラートダイアログを表示したいときは、まず、`AlertDialog.Builder` クラスのオブジェクトを `new` で生成します。このとき、`this` の値を引数としてコンストラクタに渡します。

次に、生成されたオブジェクトが持っている次のようなメソッドを呼び出して、それに対する設定をします。

`setTitle`	タイトルとして引数を設定する。
`setMessage`	表示するメッセージとして引数を設定する。
`setPositiveButton`	肯定ボタンに対して、その上に表示する文字列と、イベントリスナーを設定する。
`setNeutralButton`	中立ボタンに対して、その上に表示する文字列と、イベントリスナーを設定する。
`setNegativeButton`	否定ボタンに対して、その上に表示する文字列と、イベントリスナーを設定する。
`show`	アラートダイアログを表示する。

`setPositiveButton`、`setNeutralButton`、`setNegativeButton` のそれぞれは、それを呼び出せば対応するボタンが表示されますが、呼び出さなければ、それに対応するボタンは表示されません。ですから、それらのうちで、表示したいボタンに対応するものだけを呼び出せばいいわけです。

ボタンの設定をするメソッドに対しては、2 個の引数を渡す必要があります。1 個目はボタンの上に表示する文字列で、2 個目はイベントリスナーです。ボタンがクリックされたときはダイアログを閉じるだけでいいという場合は、2 個目の引数として `null` を渡します。

アラートダイアログに対する設定のメソッドは、メソッド呼び出しをカスケードにすることができますので、

```
new AlertDialog.Builder(this)
    .setTitle("タイトル")
    .setMessage("表示するメッセージ")
    .setPositiveButton("ボタンの上に表示するテキスト", null)
```

```
        .show()
```

というような形の式を書くことによって、アラートダイアログを表示することができます。

3.10.4　アラートダイアログを表示するアプリケーションの例 (1)

それでは、アラートダイアログを表示するアプリケーションを作ってみましょう。
まず最初に、次のようなプロジェクトを作成してください。

Project name	`alert`
Application name	アラートダイアログ
Package name	`org.example.alert`
Create Activity	`AlertActivity`

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例　`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/alert"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="アラートダイアログを表示する"
    />
</LinearLayout>
```

次に、`AlertActivity.java`を次のように書き換えてください。

プログラムの例　`AlertActivity.java`

```
package org.example.alert;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.app.AlertDialog;

public class AlertActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button alert =
            (Button) findViewById(R.id.alert);
        alert.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showAlertDialog();
            }
        });
    }

    private void showAlertDialog() {
        new AlertDialog.Builder(this)
            .setTitle("お知らせ")
            .setMessage("私はアラートダイアログです。")
            .setPositiveButton("閉じる", null)
            .show();
    }
}
```

これで完成です。実行して、ボタンをクリックしてみてください。アラートダイアログが表示されるはずです。

3.10.5 ダイアログのボタンに設定するイベントリスナー

ダイアログのボタンにイベントリスナーを設定しておくと、そのボタンがクリックされたときに、イベントリスナーの中のメソッドが実行されます。

ダイアログのボタンに設定するイベントリスナーは、

```
android.content.DialogInterface.OnClickListener
```

というインターフェースを実装したクラスから生成されるオブジェクトです。このインターフェースを実装するためには、

```
void onClick(DialogInterface dialog, int which)
```

というメソッドを定義する必要があります。ダイアログのボタンがクリックされると、このメソッドが呼び出されます。

3.10.6 アラートダイアログを表示するアプリケーションの例 (2)

それでは、アラートダイアログのボタンに対してイベントリスナーを設定するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	banana
Application name	バナナ
Package name	org.example.banana
Create Activity	BananaActivity

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例 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/banana"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="バナナについての意見を求める"
    />
<TextView android:id="@+id/selectedfaction"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、 BananaActivity.java を次のように書き換えてください。

プログラムの例 BananaActivity.java

```
package org.example.banana;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.widget.TextView;

public class BananaActivity extends Activity {
    /** Called when the activity is first created. */
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button banana = (Button) findViewById(R.id.banana);
        banana.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                bananaResearch();
            }
        });
    }

    private void bananaResearch() {
        new AlertDialog.Builder(this)
            .setTitle("バナナに関する調査")
            .setMessage("バナナはおやつに入りますか？")
            .setPositiveButton("入る",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                            int which) {
                        setFaction("バナナはおやつ派");
                    }
                }
            )
            .setNeutralButton("どちらでもない",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                            int which) {
                        setFaction("バナナはバナナ派");
                    }
                }
            )
            .setNegativeButton("入らない",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                            int which) {
                        setFaction("バナナはおやつじゃない派");
                    }
                }
            )
            .show();
    }

    private void setFaction(String faction) {
        final TextView selectedfaction =
            (TextView) findViewById(R.id.selectedfaction);
        selectedfaction.setText(
            "あなたは「" + faction + "」です。");
    }
}
```

これで完成です。実行して、ボタンをクリックしてみてください。そうすると、三つのボタンを持つアラートダイアログが表示されるはずですので、それらのボタンのうちのどれかをクリックしてみてください。そうすると、そのボタンに対応したメッセージが表示されるはずです。

3.10.7 リストを持つアラートダイアログ

AlertDialog.Builderを使うことによって、リストを持つアラートダイアログを表示するということも可能です。

リストを持つアラートダイアログを表示したいときは、setItemsというメソッドを使って、アラートダイアログにリストを設定します。

setItemsは、2個の引数を受け取ります。1個目は、リストを構成するそれぞれの項目に表示される文字列から構成される配列で、2個目は、リストの項目がクリックされたときに呼び出されるメソッドを持つイベントリスナーです。イベントリスナーの作り方は、アラートダイアログのボタンに設定するイベントリスナーの作り方と同じです。

リストの項目がクリックされると、イベントリスナーが持っている onClick というメソッドが呼び出されるわけですが、このメソッドが受け取る 2 個目の引数は、クリックされた項目の番号です。項目の番号は、 setItems に 1 個目の引数として渡した配列の添字と一致します。

3.10.8 アラートダイアログを表示するアプリケーションの例 (3)

それでは、リストを持つアラートダイアログを表示するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	listalert
Application name	リストのアラートダイアログ
Package name	org.example.listalert
Create Activity	ListAlertActivity

次に、レイアウト XML を次のように書き換えてください。

レイアウト XML の例 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/tour"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="旅行の行先についての意見を求める"
    />
<TextView android:id="@+id/selecteddestination"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、 ListAlertActivity.java を次のように書き換えてください。

プログラムの例 ListAlertActivity.java

```
package org.example.listalert;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.widget.TextView;

public class ListAlertActivity extends Activity {
    private static final String[] items = new String[] {
        "京都", "奈良", "長崎", "鹿児島"
    };

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button tour = (Button) findViewById(R.id.tour);
        tour.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                tourResearch();
            }
        });
    }
```

```
    private void tourResearch() {
        new AlertDialog.Builder(this)
            .setTitle("希望する旅行の行先は？")
            .setItems(items,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                            int which) {
                        setDestination(which);
                    }
                }
            )
            .show();
    }

    private void setDestination(int which) {
        final TextView selecteddestination =
            (TextView) findViewById(R.id.selecteddestination);
        selecteddestination.setText(
            "あなたが希望する旅行の行先は" +
            items[which] + "です。");
    }
}
```

これで完成です。実行して、ボタンをクリックしてみてください。そうすると、四つの項目から構成されるリストを持つアラートダイアログが表示されるはずですので、それらの項目のうちのどれかをクリックしてみてください。そうすると、その項目に対応したメッセージが表示されるはずです。

3.11 メニュー

3.11.1 メニューの基礎

Android を搭載した機器の上には、「メニューボタン」(menu button) と呼ばれる、「MENU」と書かれたボタンがあります。Android アプリケーションは、このボタンが押されたときに、メニューを表示することができます。

メニューは、Menu というクラスのオブジェクトです。

メニューボタンが押されると、Activity クラスで定義されている、

```
boolean onCreateOptionsMenu(Menu menu)
```

というメソッドが呼び出されます。メニューボタンが押されたときにメニューが表示されるようにするためには、このメソッドをオーバーライドする必要があります。

このメソッドが受け取る引数は、表示されることになるメニューのオブジェクトです。

このメソッドは、戻り値として、メニューを表示してほしい場合は真、表示しなくていい場合は偽を返すように定義します。

3.11.2 メニューの項目の追加

メニューは、いくつかの項目から構成されます。メニューを表示するためには、それに対して項目を追加する必要があります。

メニューが持っている、

```
MenuItem add(int groupId, int itemId, int order, CharSequence title)
```

というメソッドを呼び出すことによって、メニューに 1 個の項目を追加することができます。このメソッドに渡す引数は、1 個目がグループの ID、2 個目が項目の ID、3 個目がメニューの中での順番、4 個目がタイトルです。ID や順番を設定する必要がない場合は、Menu クラスで定義されている NONE という定数を使います。

たとえば、menu という変数にメニューのオブジェクトが設定されているとすれば、

```
menu.add(2, 7, Menu.NONE, "オムライス");
```

と書くことによって、グループの ID として 2、項目の ID として 7、そしてタイトルとして「オムライス」が設定された項目をメニューに追加することができます。

add は、MenuItem というクラスのオブジェクトを戻り値として返します。この戻り値は、メニューに追加された項目のオブジェクトです。

3.11.3 ショートカットキーの設定

メニューの項目のオブジェクトは、

```
MenuItem setShortcut(char numericChar, char alphaChar)
```

というメソッドを持っています。これは、メニューの項目にショートカットキーを設定するメソッドです。このメソッドを呼び出して、引数として 2 個の文字を渡すと、1 個目の文字がテンキー用のショートカットキーとして、2 個目の文字がフルキーボード用のショートカットキーとして設定されます。

3.11.4 メニューの項目が選択されたときの動作

メニューの項目が選択されると、Activity クラスで定義されている、

```
boolean onOptionsItemSelected(MenuItem item)
```

というメソッドが呼び出されます。ですから、このメソッドをオーバーライドすることによって、メニューの項目が選択されたときに実行される動作を記述することができます。

このメソッドが受け取る引数は、選択されたメニューの項目のオブジェクトです。

このメソッドは、戻り値として、メニューの項目が選択されたときの動作の継続を許可する場合は偽、このメソッドで動作を完結させる場合は真を返すように定義します。

3.11.5 メニューの項目に設定されたデータの取得

次のメソッドを呼び出すことによって、メニューの項目から、そこに設定されているグループの ID、項目の ID、そしてタイトルを取得することができます。

- `int getGroupId()`
- `int getItemId()`
- `CharSequence getTitle()`

3.11.6 メニューを持つアプリケーションの例

それでは、メニューを持つアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`menu`
Application name	メニュー
Package name	`org.example.menu`
Create Activity	`MenuActivity`

次に、MenuActivity.java を次のように書き換えてください。

プログラムの例 MenuActivity.java

```
package org.example.menu;

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

public class MenuActivity extends Activity {
    private static final int WASHOKU = 0;
    private static final int KATSUDON = 0;
    private static final int SUKIYAKI = 1;
    private static final int UDON = 2;
    private static final int NIKUJAGA = 3;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        menu.add(WASHOKU, KATSUDON, Menu.NONE, "カツ丼")
            .setShortcut('0', 'k');
        menu.add(WASHOKU, SUKIYAKI, Menu.NONE, "すき焼き")
            .setShortcut('1', 's');
        menu.add(WASHOKU, UDON, Menu.NONE, "うどん")
            .setShortcut('2', 'u');
        menu.add(WASHOKU, NIKUJAGA, Menu.NONE, "肉じゃが")
            .setShortcut('3', 'n');
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getGroupId()) {
        case WASHOKU:
            String itemid = Integer.toString(item.getItemId());
            String title = item.getTitle().toString();
            showAlertDialog("項目の ID = " + itemid + "\n" +
                            "タイトル = " + title);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    private void showAlertDialog(String message) {
        new AlertDialog.Builder(this)
            .setTitle("選択された項目")
            .setMessage(message)
            .setPositiveButton("閉じる", null)
            .show();
    }
}
```

これで完成です。実行して、メニューボタンを押して、メニューの項目を選択してみてください。

3.12　イメージビュー

3.12.1　イメージビューの基礎

Android には、ビットマップ画像をその上に表示することのできる、「イメージビュー」(image view) と呼ばれるウィジェットがあります。

Android が扱うことのできるビットマップ画像の形式は、次の 4 種類です。

- Windows bitmap (BMP)
- JPEG
- PNG
- GIF

ただし、使用が推奨されているのは PNG だけで、それ以外は非推奨です。

3.12.2　リソースとしてのビットマップ画像

Android では、`res/drawable-hdpi`、`res/drawable-mdpi`、`res/drawable-ldpi` という、画面の解像度に対応する三つのディレクトリのそれぞれの下にビットマップ画像 (bitmap image) のファイルを置くことによって、それをリソースとして扱うことができます。

ビットマップ画像を参照する記述は、

@drawable/ファイル名から拡張子を除いたもの

と書きます。たとえば、ビットマップ画像のファイル名が sample.png だとするならば、

@drawable/sample

と書きます。

3.12.3 イメージビューの作り方

イメージビューは、ImageView という要素型の要素をレイアウト XML の中に書くことによって作ることができます。

ImageView 要素は、android:src という属性を持っています。ビットマップ画像を参照する記述をこの属性に設定しておくと、そのビットマップ画像がイメージビューの上に表示されることになります。

3.12.4 ビットマップ画像を表示するアプリケーションの例

それでは、イメージビューを使うことによってビットマップ画像を表示するアプリケーションを作ってみましょう。

次のようなプロジェクトを作成してください。

Project name	imageview
Application name	イメージビュー
Package name	org.example.imageview
Create Activity	ImageViewActivity

次に、res/drawable-hdpi、res/drawable-mdpi、res/drawable-ldpi という三つのディレクトリのそれぞれの下に、sample.xxx という名前を持つビットマップ画像のファイル（拡張子は、png、jpg、gif など）を置いてください。

次に、main.xml を次のように書き換えてください。

レイアウト XML の例　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"
    >
<ImageView
    android:src="@drawable/sample"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

これで完成です。実行すると、sample.xxx に格納されているビットマップ画像が表示されるはずです。

第 4 章　インテント

4.1 インテントの基礎

4.1.1 インテントとは何か

この章では、「インテント」(intent) と呼ばれるオブジェクトについて説明したいと思います。

インテントというのは、実行されるべき動作をあらわすオブジェクトのことです。ちなみに、intent は、「意図」とか「趣旨」というような意味を持っている動詞です。

Android アプリケーションを構成している 4 種類のコンポーネントのうちで、アクティビティー、サービス、ブロードキャストレシーバーの 3 種類は、動作の主体となるコンポーネントです。これらの 3 種類のコンポーネントを起動するためには、インテントが必要になります。

4.1.2　インテントの生成

インテントは、Intent というクラスから生成されるオブジェクトです。

Intent クラスにはいくつかのコンストラクタがありますが、通常は、

```
Intent(Context packageContext, Class<?> cls)
```

というものを使います。このコンストラクタには、1 個目の引数として、コンポーネントを起動するオブジェクト（通常は this の値）、2 個目の引数として、起動されるコンポーネントを生成するクラスのクラスオブジェクトを渡します。たとえば、

```
new Intent(this, SecondActivity.class)
```

と書くことによって、SecondActivity というクラスのオブジェクトを起動するためのインテントを生成することができます。

4.1.3　アクティビティーの起動

これまでは、1 個の画面だけから構成される Android アプリケーションしか作ってこなかったわけですが、Android アプリケーションは、画面を何個でも持つことができます。

複数の画面を扱う Android アプリケーションを作るためには、それぞれの画面ごとに、それを生成するアクティビティーを作る必要があります。

Android アプリケーションは、画面を現在のものから別のものに切り替えるとき、その画面を生成するアクティビティーを起動します。

アクティビティーからアクティビティーを起動したいときは、通常、アクティビティーが持っている、

```
void startActivityForResult(Intent intent, int requestCode)
```

というメソッドを呼び出します。このメソッドには、1 個目の引数として、アクティビティーを起動するためのインテント、2 個目の引数として、「リクエストコード」(request code) と呼ばれる整数を渡します。リクエストコードというのは、起動したアクティビティーを識別するために使われるもので、通常は 0 以上の整数を使います。

4.1.4　Android マニフェストへの記述の追加

コンポーネントをアプリケーションに追加する場合には、そのコンポーネントについての記述を Android マニフェストに書き加える必要があります。

Android マニフェストは、簡単に書けば、

```
<manifest>
    <application>
    </application>
</manifest>
```

という構造になっています。コンポーネントについて記述した要素は、application 要素の子供として追加します。

アクティビティーは、activity という要素型の要素によって記述されます。

たとえば、SecondActivity というクラスから生成されるアクティビティーについての記述は、

```
<activity android:name=".SecondActivity"
          android:label="@string/app_name"/>
```

というように書きます。

4.1.5　第二の画面を表示するアプリケーションの例

それでは、第二のアクティビティーを動作させる Android アプリケーション、言い換えれば第二の画面を表示する Android アプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`intent`
Application name	インテント
Package name	`org.example.intent`
Create Activity	`IntentActivity`

次に、main.xml を次のように書き換えてください。

レイアウト XML の例 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/tosecond"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="第二の画面に進む"
    />
</LinearLayout>
```

次に、第二のレイアウト XML を書きます。まず、res/layout を右クリックしてください。するとメニューが表示されますので、

[New]→[File]

を選択してください。そして、表示されたダイアログに second.xml というファイル名を入力して、[Finish] をクリックしてください。そうすると、second.xml という名前の新しいファイルができますので、そのファイルに、次のレイアウト XML を入力してください。

レイアウト XML の例 second.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/textview"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="私は第二の画面です。"
    />
</LinearLayout>
```

次に、第二のアクティビティーを生成するクラスを定義します。まず、

src/org.example.intent

を右クリックしてください。するとメニューが表示されますので、

[New]→[Class]

を選択してください。そして、表示されたダイアログに、

Name SecondActivity

Superclass android.app.Activity

と入力して、[Finish] をクリックしてください。そうすると、

SecondActivity.java

という名前の新しいファイルができますので、そのファイルの内容を次のように書き換えてください。

プログラムの例 SecondActivity.java

```
package org.example.intent;

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

public class SecondActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.second);
    }
}
```

次に、IntentActivity.java を次のように書き換えてください。

プログラムの例　IntentActivity.java

```
package org.example.intent;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.content.Intent;

public class IntentActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button tosecond = (Button) findViewById(R.id.tosecond);
        tosecond.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                startSecondActivity();
            }
        });
    }

    private void startSecondActivity() {
        Intent i = new Intent(this, SecondActivity.class);
        startActivityForResult(i, 0);
    }
}
```

最後に、Android マニフェストの application 要素の子供として、

```
<activity android:name=".SecondActivity"
          android:label="@string/app_name"/>
```

という要素を書き加えてください。Android マニフェストは、プロジェクトのフォルダのすぐ下にある、AndroidManifest.xml という名前のファイルです。

さて、第二の画面を表示する Android アプリケーションは、これで完成です。実行してみてください。そうすると、[第二の画面に進む] というボタンが表示されます。そのボタンをクリックすると、第二のアクティビティーが起動して、第二の画面が表示されます。

最初の画面に戻りたいときは、Android エミュレーターの上にある矢印のボタン（[戻る] ボタン）をクリックしてください。そうすると、第二のアクティビティーが終了して、最初の画面に戻ります。

4.2　拡張データ

4.2.1　拡張データの登録

インテントというのはメッセージのオブジェクトですので、それに対してさまざまなデータを登録することができます。

たとえば、インテントが持っている putExtra というメソッドを呼び出すことによって、「拡張データ」(extended data) と呼ばれるデータをそのインテントに登録することができます。

拡張データは、キーと値のペアという形のデータです。インテントを受け取ったオブジェクトは、キーを指定することによって、それに対応づけられている値を取得することができます。

キーとしては、アプリケーションのパッケージ名を接頭辞として持つ文字列を使います。たとえば、

```
org.example.namako
```

というパッケージ名のアプリケーションでは、

```
org.example.namako.umiushi
```

というような文字列を、拡張データのキーとして使います。

putExtra には、2 個の引数を渡します。そうすると、1 個目をキー、2 個目を値とする拡張データがインテントに登録されます。2 個目の引数としては、真偽値、数値、文字列、あるいはそれらの配列など、さまざまな型のデータを渡すことができます。

4.2.2 インテントの取得

インテントが持っているデータをインテントから取得するためには、それに先立って、そのインテント自体を取得する必要があります。

アクティビティーは、

```
Intent getIntent()
```

というメソッドを持っています。このメソッドは、アクティビティーが受け取ったインテントを戻り値として返します。

4.2.3 拡張データのマップの取得

インテントから、そこに登録されている拡張データの値を取得するためには、拡張データのマップ（Bundle クラスのオブジェクト）をインテントから取得して、そのマップから拡張データの値を取得する、という 2 段階の処理をする必要があります。

拡張データのマップは、インテントが持っている、

```
Bundle getExtras()
```

というメソッドを呼び出すことによって取得することができます。このメソッドは、拡張データのマップをインテントから取得して、それを戻り値として返します。インテントに拡張データが登録されていない場合は、 null を返します。

4.2.4 拡張データの値の取得

拡張データの値は、拡張データのマップが持っている次のようなメソッドを使うことによって取得することができます。

- `boolean getBoolean(String key)`
- `int getInt(String key)`
- `String getString(String key)`
- `boolean[] getBooleanArray(String key)`
- `int[] getIntArray(String key)`
- `String[] getStringArray(String key)`

これらのメソッドは、拡張データのキーを引数として受け取って、そのキーに対応する値を戻り値として返します。

4.2.5 アクティビティーにデータを渡すアプリケーションの例

それでは、インテントに拡張データを登録することによって、アクティビティーからアクティビティーへデータを渡すアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	putextra
Application name	拡張データの登録
Package name	org.example.putextra
Create Activity	PutExtraActivity

次に、 main.xml を次のように書き換えてください。

レイアウト XML の例 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/edittext1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<EditText android:id="@+id/edittext2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<EditText android:id="@+id/edittext3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<Button android:id="@+id/tosecond"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="第二の画面に進む"
    />
</LinearLayout>
```

次に、res/layout の下に second.xml というファイルを作って、そのファイルに次のレイアウト XML を入力してください。

レイアウト XML の例　second.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/textview1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<TextView android:id="@+id/textview2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<TextView android:id="@+id/textview3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、Activity をスーパークラスとする、SecondActivity という名前の新しいクラスを作って、SecondActivity.java の内容を次のように書き換えてください。

プログラムの例　SecondActivity.java

```
package org.example.putextra;

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

public class SecondActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second);
        final TextView textview1 = (TextView) findViewById(R.id.textview1);
        final TextView textview2 = (TextView) findViewById(R.id.textview2);
        final TextView textview3 = (TextView) findViewById(R.id.textview3);
        Bundle extras = getIntent().getExtras();
        if (extras != null) {
```

```
            String s1 = extras.getString("org.example.putextra.s1");
            String s2 = extras.getString("org.example.putextra.s2");
            String s3 = extras.getString("org.example.putextra.s3");
            textview1.setText(s1);
            textview2.setText(s2);
            textview3.setText(s3);
        }
    }
}
```

次に、PutExtraActivity.javaを次のように書き換えてください。

プログラムの例 PutExtraActivity.java

```
package org.example.putextra;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.content.Intent;

public class PutExtraActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button tosecond = (Button) findViewById(R.id.tosecond);
        tosecond.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                startSecondActivity();
            }
        });
    }

    private void startSecondActivity() {
        final EditText edittext1 = (EditText) findViewById(R.id.edittext1);
        final EditText edittext2 = (EditText) findViewById(R.id.edittext2);
        final EditText edittext3 = (EditText) findViewById(R.id.edittext3);
        String s1 = edittext1.getText().toString();
        String s2 = edittext2.getText().toString();
        String s3 = edittext3.getText().toString();
        Intent i = new Intent(this, SecondActivity.class);
        i.putExtra("org.example.putextra.s1", s1);
        i.putExtra("org.example.putextra.s2", s2);
        i.putExtra("org.example.putextra.s3", s3);
        startActivityForResult(i, 0);
    }
}
```

最後に、Androidマニフェストのapplication要素の子供として、

```
<activity android:name=".SecondActivity"
          android:label="@string/app_name"/>
```

という要素を書き加えてください。

さて、アクティビティーからアクティビティーへデータを渡すAndroidアプリケーションは、これで完成です。実行してみてください。

最初に表示される画面は、三つのエディットテキストと、[第二の画面に進む]というボタンから構成されています。それぞれのエディットテキストに文字列を入力して、ボタンをクリックしてみてください。そうすると、画面が切り替わって、最初の画面で入力した文字列がその上に表示されます。

4.3　アクティビティーの結果

4.3.1　この節について

アクティビティーは、自分を起動したアクティビティーからデータを受け取ることができるだけではなくて、自分を起動したアクティビティーに対して、自分が動作した結果として、さまざまなデータを返すことができます。

この節では、アクティビティーによって起動されたアクティビティーが、自分を起動したアクティビティーに結果を返したいときにはどうすればいいのかということ、そして、アクティビティーを起動したアクティビティーが自分が起動したアクティビティーから結果を受け取りたいときにはどうすればいいのか、ということについて説明したいと思います。

4.3.2　自分自身によるアクティビティーの終了

アクティビティーの動作は、Android エミュレーターの上にある [戻る] ボタンをクリックすることによって終了させることができるのですが、自分で自分を終了させるアクティビティーを作ることも可能です。

アクティビティーは、

```
void finish()
```

というメソッドを呼び出すことによって、自分自身を終了させることができます。

4.3.3　結果の設定

アクティビティーは、

- `void setResult(int resultCode)`
- `void setResult(int resultCode, Intent data)`

という二つのメソッドのうちのどちらかを呼び出すことによって、自分を起動したアクティビティーに対して返す結果を設定することができます。

`setResult` を使うことによって設定することのできる結果は、「結果コード」(result code) と呼ばれる 1 個の整数と、1 個のインテントです。引数が 1 個だけの `setResult` は結果コードだけを設定して、引数が 2 個の `setResult` は両方を設定します。

結果コードは、通常、ユーザーの意思が「OK」なのか「キャンセル」なのか、言い換えれば自分の処理を有効なものとみなすべきかどうか、ということを、自分を起動したアクティビティーに知らせるために使われます。`Activity` クラスは、結果コードとして使うために、

```
Activity.RESULT_OK
Activity.RESULT_CANCELED
```

という定数を定義しています。

結果としてインテントを設定したいときは、まず、

```
Intent i = new Intent();
```

というように、引数を受け取らないコンストラクタでインテントを作ったのち、`putExtra` でデータをそれに登録して、それを `setResult` に渡します。

4.3.4　アクティビティーの結果の処理

アクティビティーが、自分が起動したアクティビティーの結果を処理したいときは、

```
void onActivityResult(int requestCode, int resultCode, Intent data)
```

というメソッドをオーバーライドします。これは、自分が起動したアクティビティーの動作が終了したときに自動的に呼び出されるメソッドです。

`onActivityResult` が受け取る引数の 1 個目は、リクエストコード、つまり、アクティビティーを識別するための整数です。終了したアクティビティーごとに異なる処理をしたい場合、この引数を使うことによって、終了したアクティビティーを識別することができます。

引数の 2 個目は、結果コード、つまり `setResult` によって設定された整数です。

引数の 3 個目は、`setResult` によって設定されたインテントです。

4.3.5 アクティビティーから結果を受け取るアプリケーションの例

それでは、アクティビティーを起動して、そのアクティビティーから結果を受け取るアクティビティーを持つ Android アプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`result`
Application name	結果
Package name	`org.example.result`
Create Activity	`ResultActivity`

次に、`main.xml` を次のように書き換えてください。

レイアウト XML の例 `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/tosecond"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="第二の画面に進む"
    />
<Button android:id="@+id/tothird"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="第三の画面に進む"
    />
<TextView android:id="@+id/textview"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、`res/layout` の下に `second.xml` というファイルを作って、そのファイルに次のレイアウト XML を入力してください。

レイアウト XML の例 `second.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/second_ok"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="OK"
    />
<Button android:id="@+id/second_cancel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="キャンセル"
    />
</LinearLayout>
```

次に、`res/layout` の下に `third.xml` というファイルを作って、そのファイルに次のレイアウト XML を入力してください。

レイアウト XML の例 `third.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/edittext"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<Button android:id="@+id/third_ok"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="OK"
    />
<Button android:id="@+id/third_cancel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="キャンセル"
    />
</LinearLayout>
```

次に、Activity をスーパークラスとする、SecondActivity という名前の新しいクラスを作って、SecondActivity.java の内容を次のように書き換えてください。

プログラムの例　SecondActivity.java

```
package org.example.result;

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

public class SecondActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second);
        final Button ok = (Button) findViewById(R.id.second_ok);
        ok.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                setResult(Activity.RESULT_OK);
                finish();
            }
        });
        final Button cancel = (Button) findViewById(R.id.second_cancel);
        cancel.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                setResult(Activity.RESULT_CANCELED);
                finish();
            }
        });
    }
}
```

次に、Activity をスーパークラスとする、ThirdActivity という名前の新しいクラスを作って、ThirdActivity.java の内容を次のように書き換えてください。

プログラムの例　ThirdActivity.java

```
package org.example.result;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.content.Intent;
```

```
public class ThirdActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.third);
        final Button third_ok = (Button) findViewById(R.id.third_ok);
        third_ok.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                setResultOfString();
                finish();
            }
        });
        final Button third_cancel = (Button) findViewById(R.id.third_cancel);
        third_cancel.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                setResult(Activity.RESULT_CANCELED);
                finish();
            }
        });
    }

    private void setResultOfString() {
        final EditText edittext = (EditText) findViewById(R.id.edittext);
        String s = edittext.getText().toString();
        Intent i = new Intent();
        i.putExtra("org.example.result.s", s);
        setResult(Activity.RESULT_OK, i);
    }
}
```

次に、ResultActivity.javaを次のように書き換えてください。

プログラムの例 ResultActivity.java

```
package org.example.result;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.content.Intent;

public class ResultActivity extends Activity {
    private static final int SECOND_ACTIVITY = 2;
    private static final int THIRD_ACTIVITY = 3;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button tosecond = (Button) findViewById(R.id.tosecond);
        tosecond.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                startSecondActivity();
            }
        });
        final Button tothird = (Button) findViewById(R.id.tothird);
        tothird.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                startThirdActivity();
            }
        });
    }

    @Override
```

```
    public void onActivityResult(
            int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        setMessage("");
        String message = "";
        switch (requestCode) {
        case SECOND_ACTIVITY:
            message = "第二の画面:";
            switch (resultCode) {
            case Activity.RESULT_OK:
                message += "RESULT_OK";
                break;
            case Activity.RESULT_CANCELED:
                message += "RESULT_CANCELED";
                break;
            }
            break;
        case THIRD_ACTIVITY:
            message = "第三の画面:";
            switch (resultCode) {
            case Activity.RESULT_OK:
                message += "RESULT_OK:";
                Bundle extras = data.getExtras();
                message += extras.getString("org.example.result.s");
                break;
            case Activity.RESULT_CANCELED:
                message += "RESULT_CANCELED";
                break;
            }
            break;
        }
        setMessage(message);
    }

    private void startSecondActivity() {
        Intent i = new Intent(this, SecondActivity.class);
        startActivityForResult(i, SECOND_ACTIVITY);
    }

    private void startThirdActivity() {
        Intent i = new Intent(this, ThirdActivity.class);
        startActivityForResult(i, THIRD_ACTIVITY);
    }

    private void setMessage(String message) {
        final TextView textview = (TextView) findViewById(R.id.textview);
        textview.setText(message);
    }
}
```

最後に、Android マニフェストの application 要素の子供として、

```
<activity android:name=".SecondActivity"
          android:label="@string/app_name"/>
<activity android:name=".ThirdActivity"
          android:label="@string/app_name"/>
```

という二つの要素を書き加えてください。

さて、アクティビティーから結果を受け取るアクティビティーを持つ Android アプリケーションは、これで完成です。実行してみてください。

最初に表示される画面は、二つのボタンとひとつのテキストビューから構成されています。[第二の画面に進む] というボタンをクリックすると第二のアクティビティーが起動して、[第三の画面に進む] というボタンをクリックすると第三のアクティビティーが起動します。テキストビューには、第二または第三のアクティビティーが終了したのちに、それぞれのアクティビティーが返した結果が表示されます。

第二と第三のアクティビティーは、それぞれ、[OK] と [キャンセル] という二つのボタンを持っ

ています。[OK] は、結果コードとして RESULT_OK を設定してアクティビティーを終了させます。[キャンセル] は、結果コードとして RESULT_CANCELED を設定してアクティビティーを終了させます。

第三のアクティビティーは、1 個のエディットテキストを持っています。このエディットテキストに文字列を入力して [OK] をクリックすると、その文字列が登録されたインテントが結果として設定されます。

4.4 暗黙的インテント

4.4.1 明示的インテントと暗黙的インテント

この章では、これまで、自分と同じアプリケーションの中にあるコンポーネントを起動する方法について説明してきたわけですが、インテントを使うことによって起動することができるのは、同じアプリケーションの中にあるコンポーネントだけではありません。インテントを使うことによって、別のアプリケーションの中にあるコンポーネントを起動するということ、つまり別のアプリケーションを起動するということも可能です。

別のアプリケーションの中にあるコンポーネントを起動したいときは、「暗黙的インテント」(implicit intent) と呼ばれるものが使われます。

暗黙的インテントというのは、起動する対象となるコンポーネントのクラスが特定されていないインテントのことです。それに対して、起動する対象となるコンポーネントのクラスが特定されているインテントは、「明示的インテント」(explicit intent) と呼ばれます。

暗黙的インテントは、起動する対象となるコンポーネントのクラスが特定されていないわけですが、その代わりとして、次のような情報を持つことによって、起動する対象となるコンポーネントを選択することができます。

アクション (action)	実行されるべき動作をあらわす文字列。標準的なアクションは、Intent クラスの中で定数として定義されている。
カテゴリー (category)	実行されるべきコンポーネントのカテゴリーをあらわす文字列。標準的なカテゴリーは、 Intent クラスの中で定数として定義されている。
データ (data)	データの URI と、そのデータの MIME タイプ。

たとえば、

```
android.intent.action.VIEW （定数は ACTION_VIEW）
```

というアクションと、http または https というスキーマを持つ URI が与えられたインテントを使うことによって、ブラウザーを起動して、URI で指定されたページを表示させる、ということができます。

4.4.2 アクションと URI を持つインテントを生成するコンストラクタ

アクションと URI を持つインテントは、

```
Intent(String action, Uri uri)
```

というコンストラクタを使うことによって生成することができます。このコンストラクタは、1 個目の引数としてアクションを受け取って、2 個目の引数として URI を受け取ります。

4.4.3 暗黙的インテントによってアクティビティーを起動するメソッド

暗黙的インテントを使ってアクティビティーを起動したいときは、

```
void startActivity(Intent intent)
```

というメソッドを使います。

4.4.4 アプリケーションを起動するアプリケーションの例

それでは、暗黙的インテントを使うことによって、アプリケーションを起動するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	start
Package name	org.example.start

Activity name `StartActivity`
Application name アプリケーションの起動

次に、`main.xml` を次のように書き換えてください。

レイアウト XML の例 `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"
    >
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1"
    >
<TableRow>
    <TextView android:text="アクション"/>
    <EditText android:id="@+id/action"/>
</TableRow>
<TableRow>
    <TextView android:text="URI"/>
    <EditText android:id="@+id/uri"/>
</TableRow>
</TableLayout>
    <Button android:id="@+id/start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="起動"
        />
</LinearLayout>
```

次に、`StartActivity.java` を次のように書き換えてください。

プログラムの例 `StartActivity.java`

```
package org.example.start;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.content.Intent;
import android.net.Uri;

public class StartActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button start = (Button) findViewById(R.id.start);
        start.setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                startApplication();
            }
        });
    }

    private void startApplication() {
        final EditText action = (EditText) findViewById(R.id.action);
        final EditText uri = (EditText) findViewById(R.id.uri);
        Intent i = new Intent(action.getText().toString(),
            Uri.parse(uri.getText().toString()));
        startActivity(i);
    }
```

```
}
```

これで完成です。実行すると、2 個のエディットテキストと 1 個のボタンが表示されるはずですので、上のエディットテキストに、

```
android.intent.action.VIEW
```

というアクションを入力して、下のエディットテキストに、

```
http://ja.wikipedia.org/
```

というような、http または https というスキーマを持つ URI を入力して、ボタンをクリックしてみてください。そうすると、ブラウザーが起動して、入力した URI で指定されたページが表示されるはずです。

同じように、上のエディットテキストに、

```
android.intent.action.DIAL
```

というアクションを入力して、下のエディットテキストに、

```
tel:1123581321
```

というような、tel というスキーマを持つ URI を入力して、ボタンをクリックすると、ダイアラーが起動します。

4.5 インテントフィルター

4.5.1 インテントフィルターの基礎

コンポーネントのうちで、コンテントプロバイダーを除いたもの、つまり、アクティビティー、サービス、ブロードキャストレシーバーの 3 種類のコンポーネントは、「インテントフィルター」(intent filter) と呼ばれるものを持つことができます。

インテントフィルターというのは、それを持っているコンポーネントはどのようなインテントを受け取ることができるのかという条件を規定しているオブジェクトのことです。コンポーネントは、インテントフィルターで規定されている条件を満足するインテントが送られてきた場合、そのインテントを受け取って処理します。

インテントフィルターは、

```
android.content.IntentFilter
```

というクラスのインスタンスですが、通常は、Android マニフェストの中に intent-filter という要素を書くことによって作ります。

intent-filter 要素は、コンポーネントの要素（たとえば activity）の子供として書きます。そして、intent-filter 要素の子供として、アクションを指定する action 要素、カテゴリーを指定する category 要素、データを指定する data 要素を書きます。

4.5.2 別のアプリケーションから起動できるアプリケーションの例

それでは、インテントフィルターを作ることによって、別のアプリケーションから起動することのできるアクティビティーを持つアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	message
Application name	メッセージの表示
Package name	org.example.message
Create Activity	MessageActivity

次に、main.xml を次のように書き換えてください。

レイアウト XML の例　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/message"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、MessageActivity.java を次のように書き換えてください。

プログラムの例　MessageActivity.java

```
package org.example.message;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.net.Uri;

public class MessageActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Uri uri = getIntent().getData();
        if (uri != null) {
            final TextView message = (TextView) findViewById(R.id.message);
            message.setText(uri.toString());
        }
    }
}
```

最後に、Android マニフェストの中にある activity 要素の子供として、

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:scheme="message" />
</intent-filter>
```

という要素を書き加えてください。

これで完成です。このアプリケーションのアクティビティーは、普通に起動した場合は何もしませんが、message というスキーマを持つ URI が与えられたインテントで起動すると、その URI を表示するという処理を実行します。

第 4.4 節で作った、「アプリケーションの起動」というアプリケーションを使って試してみましょう。「アプリケーションの起動」を起動して、上のエディットテキストに、

```
android.intent.action.VIEW
```

というアクションを入力して、下のエディットテキストに、

```
message:IAmVeryHungry
```

というような、message というスキーマを持つ URI を入力して、ボタンをクリックしてみてください。そうすると、「メッセージの表示」が起動して、入力した URI が表示されるはずです。

4.6　サービス

4.6.1　サービスの基礎

この節では、「サービス」(service) と呼ばれるオブジェクトについて説明したいと思います。

第 1.6 節で説明したように、サービスは、Android アプリケーションを構築するための 4 種類のコンポーネントのひとつです。

サービスは、アクティビティーと同じように、動作の主体となるオブジェクトですが、アクティビティーとは違って、画面を表示する機能がありません。ですから、サービスは、バックグラウンドで（画面の背後で）実行されることになります。

サービスは、

```
android.app.Service
```

という抽象クラスのサブクラスから作られるオブジェクトです。

4.6.2 サービスの起動

サービスを起動するメソッドとしては、

- `ComponentName startService(Intent service)`
- `boolean bindService(Intent service, ServiceConnection conn, int flags)`

という二つのものがあります。

`startService` を使ってサービスを起動する方法は、アクティビティーを起動する方法とほとんど同じです。たとえば、サービスのクラス名が `MyService` だとするならば、

```
Intent i = new Intent(this, MyService.class);
startService(i);
```

と書くことによって、そのクラスのサービスを起動することができます。

`bindService` は、「バインド」(bind) と呼ばれる方法でサービスを起動する場合に使われるメソッドです。

4.6.3 サービスが起動されたときに呼び出されるメソッド

`Service` クラスで実装されている、

```
int onStartCommand(Intent intent, int flags, int startId)
```

というメソッドは、`startService` でサービスが起動されたときに呼び出されます。ですから、このメソッドをサブクラスでオーバーライドすることによって、サービスが起動されたときに実行したい動作を記述することができます。

`onStart` が受け取る 1 個目の引数は、サービスを起動したインテントです。

4.6.4 バインドによってサービスが起動されたときに呼び出されるメソッド

`Service` クラスには、

```
IBinder onBind(Intent intent)
```

という抽象メソッドがあります。これを実装したメソッドは、バインドによってサービスが起動されたときに呼び出されることになります。

`startService` を使ってサービスを起動する場合、`onBind` は不要なメソッドですので、`null` を返すだけの動作を実装します。

4.6.5 サービスについての記述

サービスを持つアプリケーションを作る場合には、Android マニフェストの `application` 要素の子供として、そのサービスについての記述を書き加える必要があります。

サービスについての記述は、

```
<service android:name=". クラス名" />
```

という要素です。「クラス名」のところには、`Service` クラスのサブクラスの名前を書きます。

4.6.6 サービスを起動するアプリケーションの例

それでは、サービスを起動するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`service`
Application name	サービス
Package name	`org.example.service`
Create Activity	`ServiceActivity`

次に、`main.xml` を次のように書き換えてください。

レイアウト XML の例 `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/edittext"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<Button android:id="@+id/start"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="サービスの起動"
    />
</LinearLayout>
```

次に、Service をスーパークラスとする、MessageService という名前の新しいクラスを作って、MessageService.java の内容を次のように書き換えてください。

プログラムの例 MessageService.java

```
package org.example.service;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

public class MessageService extends Service
        implements Runnable {
    private Toast toast;
    Thread thread = null;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startID) {
        Toast.makeText(this, "Service Start",
            Toast.LENGTH_SHORT).show();
        toast = Toast.makeText(this,
            intent.getExtras().getString("org.example.service.s"),
            Toast.LENGTH_SHORT);
        thread = new Thread(this);
        thread.start();
        return super.onStartCommand(intent, flags, startID);
    }

    @Override
    public void run() {
        try {
            while (true) {
                Thread.sleep(5000);
                toast.show();
            }
        } catch (InterruptedException e) {
        }
    }
}
```

次に、ServiceActivity.java を次のように書き換えてください。

プログラムの例 ServiceActivity.java

```
package org.example.service;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.content.Intent;

public class ServiceActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button start = (Button) findViewById(R.id.start);
        start.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                startService();
            }
        });
    }

    public void startService() {
        final EditText edittext = (EditText) findViewById(R.id.edittext);
        String s = edittext.getText().toString();
        Intent i = new Intent(this, MessageService.class);
        i.putExtra("org.example.service.s", s);
        startService(i);
    }
}
```

最後に、Android マニフェストの application 要素の子供として、

```
<service android:name=".MessageService" />
```

という要素を書き加えてください。

これで完成です。実行すると、1 個のエディットテキストと、[サービスを起動] というボタンが表示されますので、エディットテキストに文字列を入力して、ボタンをクリックしてみてください。そうすると、まず “Service Start” とトーストで表示されたのち、エディットテキストに入力された文字列が 5 秒ごとにトーストで表示されるはずです。

4.7 ブロードキャスト

4.7.1 ブロードキャストの基礎

これまでに、インテントを使ってコンポーネントを起動するためのメソッドとして、

- `void startActivityForResult(Intent intent, int requestCode)`
- `void startActivity(Intent intent)`
- `ComponentName startService(Intent service)`

という三つものを紹介しましたが、これらのメソッドは、もしもインテントがコンポーネントを起動することができなかった場合、例外を発生させます。このことは、これらのメソッドが、あくまで何らかの動作を期待して呼び出されるものだということを意味しています。

これらのメソッドとは対照的なメソッドとして、

```
void sendBroadcast(Intent intent)
```

というものがあります。このメソッドは、引数で受け取ったインテントを送り出すという動作をするのですが、そのインテントがいかなるコンポーネントも起動しなかったとしても、例外は発生させません。このメソッドの目的は、何らかの動作を引き起こすことではなくて、何らかの事象を広範囲に告知することなのです。

sendBroadcast がインテントを送り出すことを、インテントを「ブロードキャストする」(broadcast) と言います。ちなみに、broadcast は、「(種などを) まく」とか「言いふらす」とか「放送

する」というような意味を持っている動詞です。
　ブロードキャストされるインテントは、URI を持っていなくてもかまいません。Intent クラスでは、アクションは持っているけれども URI は持っていないインテントを生成する、

```
Intent(String action)
```

というコンストラクタが定義されています。

4.7.2　ブロードキャストレシーバー

　動作の主体となる、アクティビティー、サービス、ブロードキャストレシーバーという 3 種類のコンポーネントのうちで、ブロードキャストを受け取ることができるのは、ブロードキャストレシーバーだけです。
　ブロードキャストレシーバーは、サービスと同じように、画面を表示する機能を持っていません。ただし、やはりサービスと同じように、トーストなどを使うことによってユーザーに何かを知らせることは可能です。
　ブロードキャストレシーバーは、時間のかかる処理を実行することができません。ブロードキャストされたインテントを受け取って時間のかかる処理を実行したいときは、ブロードキャストレシーバーでそのインテントを受け取ったのち、サービスを起動する必要があります。

4.7.3　ブロードキャストレシーバーの作り方

　ブロードキャストレシーバーを作りたいときは、

```
android.content.BroadcastReceiver
```

という抽象クラスのサブクラスを定義して、

```
void onReceive(Context context, Intent intent)
```

という抽象メソッドを実装します。このメソッドは、ブロードキャストレシーバーがインテントを受け取ったときに呼び出されます。このメソッドの二つ目の引数は、ブロードキャストレシーバーが受け取ったインテントです。

4.7.4　ブロードキャストレシーバーについての記述

　ブロードキャストレシーバーを持つアプリケーションを作る場合には、Android マニフェストの application 要素の子供として、そのブロードキャストレシーバーについての記述を書き加える必要があります。
　ブロードキャストレシーバーについての記述は、receiver という要素型の要素です。receiver 要素は、

```
<receiver android:name=". クラス名">
    <intent-filter>
        <action android:name="アクション"/>
    </intent-filter>
</receiver>
```

というように書きます。「クラス名」のところには、BroadcastReceiver クラスのサブクラスの名前を書いて、「アクション」のところには、受け取りたいインテントが持っているアクションを書きます。

4.7.5　ブロードキャストされたインテントを受け取るアプリケーションの例

　それでは、ブロードキャストレシーバーを使うことによって、ブロードキャストされたインテントを受け取るアプリケーションを作ってみましょう。
　まず最初に、次のようなプロジェクトを作成してください。

Project name	receiver
Application name	ブロードキャストレシーバー
Package name	org.example.receiver
Create Activity	チェックをはずす。

　次に、BroadcastReceiver をスーパークラスとする、Receiver という名前の新しいクラスを作って、Receiver.java の内容を次のように書き換えてください。

プログラムの例 Receiver.java

```
package org.example.receiver;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.widget.Toast;

public class Receiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle extras = intent.getExtras();
        if (extras != null) {
            String message =
                extras.getString("org.example.broadcast.message");
            Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
        }
    }
}
```

最後に、Android マニフェストの application 要素の子供として、

```
<receiver android:name=".Receiver">
    <intent-filter>
        <action android:name="org.example.broadcast.SHOW_MESSAGE"/>
    </intent-filter>
</receiver>
```

という要素を書き加えてください。

これで完成ですので、Eclipse のメニューで、[Run]→[Run] を選択してください。ただし、このアプリケーションを起動するためには、そのためのインテントをブロードキャストする必要があります。

4.7.6 インテントをブロードキャストするアプリケーションの例

それでは、インテントをブロードキャストすることによって、先ほど作ったアプリケーションを起動するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	broadcast
Application name	ブロードキャスト
Package name	org.example.broadcast
Create Activity	BroadcastActivity

次に、main.xml を次のように書き換えてください。

レイアウト XML の例 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/edittext"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<Button android:id="@+id/broadcast"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ブロードキャスト"
    />
</LinearLayout>
```

次に、BroadcastActivity.java を次のように書き換えてください。

プログラムの例　BroadcastActivity.java

```
package org.example.broadcast;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.content.Intent;

public class BroadcastActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button broadcast = (Button) findViewById(R.id.broadcast);
        broadcast.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                sendBroadcast();
            }
        });
    }

    private void sendBroadcast() {
        final EditText edittext = (EditText) findViewById(R.id.edittext);
        String message = edittext.getText().toString();
        Intent i = new Intent("org.example.broadcast.SHOW_MESSAGE");
        i.putExtra("org.example.broadcast.message", message);
        sendBroadcast(i);
    }
}
```

これで完成です。実行してみてください。そうすると、1 個のエディットテキストと 1 個のボタンが表示されるはずですので、エディットテキストに文字列を入力してボタンをクリックしてみてください。そうすると、その文字列を持つインテントがブロードキャストされて、先ほど作ったアプリケーションが、そのインテントをブロードキャストレシーバーで受け取って、文字列をトーストで表示するはずです。

4.7.7　Android によるブロードキャスト

インテントをブロードキャストするのはアプリケーションだけではありません。Android 自身も、さまざまな事象を広範囲に告知するためにインテントをブロードキャストします。たとえば、Android は、システム設定のタイムゾーンが変更されると、

```
android.intent.action.TIMEZONE_CHANGED
```

というアクションを持つインテントをブロードキャストします。

Android アプリケーションは、Android がブロードキャストしたインテントを受け取ることも可能です。ただし、そのためには、それを許可する記述を Android マニフェストの中に書く必要があります。

Android がブロードキャストしたインテントの受け取りを許可する記述というのは、receiver 要素の開始タグの中に書く属性指定で、属性名は android:permission です。たとえば、システム設定のタイムゾーンが変更されたときにブロードキャストされるインテントを受け取りたいときは、receiver 要素の開始タグの中に、

```
android:permission="android.permission.RECEIVE_SMS"
```

という属性指定を書きます。

4.7.8 Android 内の事象によって起動されるアプリケーションの例

それでは、Android がブロードキャストしたインテントを受け取るアプリケーションの例として、システム設定のタイムゾーンが変更されたときに起動するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`timezone`
Application name	タイムゾーン
Package name	`org.example.timezone`
Create Activity	チェックをはずす。

次に、`BroadcastReceiver` をスーパークラスとする、`TimezoneReceiver` という名前の新しいクラスを作って、`TimezoneReceiver.java` の内容を次のように書き換えてください。

プログラムの例　`TimezoneReceiver.java`

```
package org.example.timezone;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class TimezoneReceiver extends BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent) {
             String message = "タイムゾーンが変更されました。";
             Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
        }
}
```

最後に、Android マニフェストの `application` 要素の子供として、

```
  <receiver android:name=".TimezoneReceiver"
            android:permission="android.permission.RECEIVE_SMS">
      <intent-filter>
          <action android:name="android.intent.action.TIMEZONE_CHANGED"/>
      </intent-filter>
  </receiver>
```

という要素を書き加えてください。

これで完成ですので、Eclipse のメニューで、[Run]→[Run] を選択してください。そして、Anodoid に付属している Settings というアプリケーションを使って、システム設定のタイムゾーンを変更してみてください。そうすると、「タイムゾーンが変更されました。」というメッセージがトーストで表示されるはずです。

第 5 章　グラフィックス

5.1　ビュー

5.1.1　ビューの基礎

この章では、Android の画面にグラフィックスを描画する方法について説明したいと思います。

Android の画面にグラフィックスを描画するためには、「ビュー」(view) と呼ばれるものを作る必要があります。ビューというのは、画面の上に表示することのできる長方形の平面のことです。テキストビューやボタンなどのウィジェットも、ビューの一種です。

5.1.2　ビューの作り方

ビューは、`android.view.View` というクラスを継承するクラスから生成されるオブジェクトです。

その上にグラフィックスを描画することのできるビューを作るためには、`View` クラスを継承する独自のクラスを定義する必要があります。

Viewクラスのコンストラクタには、android.content.Contextという抽象クラスを継承しているクラスのオブジェクト（アクティビティーはその一種）を引数として渡す必要があります。ですから、Viewクラスのサブクラスを定義する場合には、

```
class MyView extends View {
    MyView(Context context) {
        super(context);
    }
}
```

というように、Contextクラスを継承しているクラスのオブジェクトを引数として受け取るコンストラクタを定義する必要があります。

5.1.3 アクティビティーに対するビューの設定

ビューを画面に表示するためには、そのビューをアクティビティーに設定する必要があります。
アクティビティーに対してビューを設定したいときは、

```
void setContentView(View view)
```

というメソッドを使います。たとえば、

```
setContentView(new MyView(this));
```

と書くことによって、MyViewというクラスからビューを生成して、それをアクティビティーに設定することができます。

5.1.4 色をあらわす整数

Androidでは、何かに色を設定するメソッドは、その色をあらわす整数を引数として受け取ります。
色をあらわす整数は、android.graphics.Colorというクラスの中で定義されている、次のようなクラスメソッドを使うことによって求めることができます。

- `static int argb(int alpha, int red, int green, int blue)`
- `static int rgb(int red, int green, int blue)`
- `static int parseColor(String colorString)`

argbは、アルファ値、赤、緑、青のそれぞれをあらわす0から255までの整数から、色をあらわす整数を求めるメソッドです。「アルファ値」(alpha value)というのは、不透明度のことです。0は完全に透明、255は完全に不透明という意味になります。
rgbは、アルファ値が255の色、つまり完全に不透明な色をあらわす整数を求めるメソッドです。
parseColorは、#AARRGGBBまたは#RRGGBBという形式の文字列から、色をあらわす整数を求めるメソッドです。
色をあらわす整数は、Colorクラスの中で定義されている次の定数を使うことによって求めることもできます。

```
BLACK   CYAN     GRAY    LTGRAY    RED           WHITE
BLUE    DKGRAY   GREEN   MAGENTA   TRANSPARENT   YELLOW
```

getResourcesとgetColorという二つのメソッドを使うことによって、リソースとして定義された色から、その色をあらわす整数を求める、ということもできます。たとえば、

```
getResources().getColor(R.color.usuhanada)
```

と書くことによって、usuhanadaという名前で定義された色のリソースから、その色をあらわす整数を求めることができます。

5.1.5 ビューの背景色

ビューの背景色は、ビューのクラスのコンストラクタまたはメソッドの中で、

```
void setBackgroundColor(int color)
```

というメソッドを呼び出すことによって設定することができます。
たとえば、

```
setBackgroundColor(Color.rgb(255, 128, 0));
```

と書くことによって、ビューの背景色をオレンジ色に設定することができます。

5.1.6 独自のビューを表示するアプリケーションの例

それでは、独自のビューを表示するアプリケーションを作ってみましょう。
次のようなプロジェクトを作成してください。

Project name	`view`
Application name	独自のビュー
Package name	`org.example.view`
Create Activity	`ViewActivity`

次に、`ViewActivity.java`を次のように書き換えてください。

プログラムの例 `ViewActivity.java`

```
package org.example.view;

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

public class ViewActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new MyView(this));
    }
}

class MyView extends View {
    MyView(Context context) {
        super(context);
        setBackgroundColor(Color.rgb(192, 255, 64));
    }
}
```

これで完成です。実行すると、背景色が黄緑色に設定されたビューが表示されるはずです。

5.2 キャンバス

5.2.1 キャンバスの基礎

ビューの上にグラフィックスを描画するためには、「キャンバス」(canvas) と呼ばれるオブジェクトが必要になります。

キャンバスというのは、`android.graphics.Canvas`というクラスのオブジェクトのことです。このオブジェクトは、ビューの上にグラフィックスを描画するために使うことのできるさまざまなメソッドを持っています。

5.2.2 グラフィックスの描画

`View`クラスの中では、

```
void onDraw(Canvas canvas)
```

というメソッドが定義されています。これは、ビューの上にグラフィックスを描画する必要が生じたときに自動的に呼び出されるメソッドです。ということは、`View`を継承するクラスを定義するときに、このメソッドをオーバーライドして、その中でグラフィックスの描画を記述しておけば、その記述は、ビューの上にグラフィックスを描画する必要が生じるたびに実行される、ということになります。

onDraw は、キャンバスを引数として受け取ります。グラフィックスを描画したいときは、onDraw が引数として受け取ったキャンバスが持っているメソッドを使うことになります。

5.2.3　座標系

グラフィックスを描画するときには、点の位置を指定するために、何らかの座標系が使われます。

Android で使われる座標系は、次のようなものです。

- 原点は画面の左上の隅。
- x 軸は右向き。
- y 軸は下向き。
- 距離の単位はピクセル。

5.2.4　ペイント

キャンバスは、ビューの上にグラフィックスを描画するためのさまざまなメソッドを持っています。たとえば、キャンバスが持っている、

```
void drawCircle(float cx, float cy, float radius, Paint paint)
```

というメソッドを呼び出すことによって、ビューの上に円を描画することができます。このメソッドが受け取る引数は、1 個目が中心の x 座標、2 個目が中心の y 座標、3 個目が半径です。そしてこのメソッドは、4 個目の引数として、「ペイント」(paint) と呼ばれるオブジェクトを受け取ります。

ペイントというのは、android.graphics.Paint というクラスのオブジェクトのことです。ペイントは、グラフィックスを描画するときに使われるスタイルと色を保持することができます。たとえば、

```
void setColor(int color)
```

というメソッドを呼び出すことによって、グラフィックスを描画するときに使う色をペイントに設定することができます。

5.2.5　グラフィックスを描画するアプリケーションの例

それでは、グラフィックスを描画するアプリケーションを作ってみましょう。

次のようなプロジェクトを作成してください。

Project name	canvas
Application name	キャンバス
Package name	org.example.canvas
Create Activity	CanvasActivity

次に、CanvasActivity.java を次のように書き換えてください。

プログラムの例　CanvasActivity.java

```
package org.example.canvas;

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

public class CanvasActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new CanvasView(this));
    }
}
```

```
class CanvasView extends View {
    CanvasView(Context context) {
        super(context);
        setBackgroundColor(Color.WHITE);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        paint.setColor(Color.rgb(255, 0, 0));
        canvas.drawCircle(100, 100, 50, paint);
        paint.setColor(Color.argb(128, 0, 0, 255));
        canvas.drawCircle(140, 100, 50, paint);
    }
}
```

これで完成です。実行すると、赤色の円と、青色で半透明の円が描画されるはずです。

5.2.6 ビューの大きさ

ビューの上にグラフィックスを描画する場合に、そのビューの大きさを意識してグラフィックスの位置や大きさを決定したい、ということがしばしばあります。Android では、

- `int getWidth()`
- `int getHeight()`

というメソッドを呼び出すことによって、ビューの大きさを求めることができます。`getWidth` は横の長さを返すメソッドで、`getHeight` は縦の長さを返すメソッドです（単位はどちらもピクセル）。

5.2.7 ビューの大きさを意識して動作するアプリケーションの例

それでは、`getWidth` と `getHeight` を使って、ビューの大きさを意識して動作するアプリケーションを作ってみましょう。

次のようなプロジェクトを作成してください。

Project name	`size`
Application name	ビューの大きさ
Package name	`org.example.size`
Create Activity	`SizeActivity`

次に、`SizeActivity.java` を次のように書き換えてください。

プログラムの例 `SizeActivity.java`

```
package org.example.size;

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

public class SizeActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new CanvasView(this));
    }
}

class CanvasView extends View {
    private static final int RADIUS = 30;
```

```
    CanvasView(Context context) {
        super(context);
        setBackgroundColor(Color.WHITE);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        int width = getWidth();
        int height = getHeight();
        Paint paint = new Paint();
        paint.setColor(Color.rgb(0, 128, 0));
        canvas.drawCircle(0, 0, RADIUS, paint);
        canvas.drawCircle(width, 0, RADIUS, paint);
        canvas.drawCircle(0, height, RADIUS, paint);
        canvas.drawCircle(width, height, RADIUS, paint);
    }
}
```

これで完成です。実行すると、ビューの左上、右上、左下、右下のそれぞれを中心とする 4 個の円が描画されるはずです。

5.3 形状

5.3.1 形状を描画するメソッド

キャンバスは、形状を描画するメソッドとして、次のようなものを持っています。

drawCircle	円を描画します。
drawRect	長方形を描画します。
drawRoundRect	角の丸い長方形を描画します。
drawOval	楕円を描画します。
drawLine	直線を描画します。

これらのメソッドのほかに、形状を描画するメソッドとしては、drawPath というのもあるのですが、これについては第 5.4 節で説明することにしたいと思います。

5.3.2 描画のスタイル

ペイントは、描画のスタイルというものを保持しています。描画のスタイルというのは、形状を描画するときに、それを塗りつぶすのか、それとも輪郭の線だけを描画するのか、という状態のことです。デフォルトでは、塗りつぶすだけの状態になっています。

描画のスタイルは、

```
void setStyle(Paint.Style style)
```

というメソッドを呼び出すことによって設定することができます。

setStyle には、描画のスタイルをあらわすオブジェクトを引数として渡します。描画のスタイルをあらわすオブジェクトは、Paint.Style というクラスの中で、次のような定数として定義されています。

FILL	塗りつぶすだけ。デフォルト。
STROKE	輪郭の線だけを描画する。
FILL_AND_STROKE	塗りつぶして、さらに輪郭の線も描画する。

5.3.3 線の幅

ペイントは、形状を描画するときに使われる線の幅を保持しています。

線の幅は、

```
void setStrokeWidth(float width)
```

というメソッドを呼び出すことによって設定することができます。

5.3.4 長方形のクラス

drawRect や drawRoundRect や drawOval は、長方形をあらわす、RectF というクラスのオブジェクトを引数として受け取ります。

RectF クラスのオブジェクトを作るときは、普通、コンストラクタに 4 個の不動小数点数を渡します。それらの 4 個の不動小数点数は、1 個目が長方形の左端の x 座標、2 個目が長方形の上端の y 座標、3 個目が長方形の右端の x 座標、4 個目が長方形の下端の y 座標になります。

5.3.5 形状を描画するアプリケーションの例

それでは、形状を描画するアプリケーションを作ってみましょう。
次のようなプロジェクトを作成してください。

Project name	shape
Application name	形状の描画
Package name	org.example.shape
Create Activity	ShapeActivity

次に、ShapeActivity.java を次のように書き換えてください。

プログラムの例 ShapeActivity.java

```
package org.example.shape;

import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.graphics.Color;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.RectF;

public class ShapeActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new CanvasView(this));
    }
}

class CanvasView extends View {
    CanvasView(Context context) {
        super(context);
        setBackgroundColor(Color.WHITE);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        paint.setColor(Color.rgb(0, 128, 255));
        canvas.drawCircle(50, 50, 30, paint);
        canvas.drawRect(new RectF(20, 100, 100, 150), paint);
        canvas.drawRoundRect(
            new RectF(20, 170, 100, 220), 20, 15, paint);
        canvas.drawOval(new RectF(20, 240, 100, 290), paint);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(6);
        canvas.drawCircle(150, 50, 30, paint);
        canvas.drawRect(new RectF(120, 100, 200, 150), paint);
        canvas.drawRoundRect(
            new RectF(120, 170, 200, 220), 20, 15, paint);
        canvas.drawOval(new RectF(120, 240, 200, 290), paint);
        canvas.drawLine(20, 310, 200, 360, paint);
    }
```

```
}
```

これで完成です。実行すると、円、長方形、角の丸い長方形、楕円、直線が描画されるはずです。

5.4 パス

5.4.1 パスの基礎

第5.3節で説明したように、キャンバスは、形状を描画するためのさまざまなメソッドを持っているわけですが、そのようなメソッドのひとつに、

```
void drawPath(Path path, Paint paint)
```

というものがあります。これは、「パス」(path) と呼ばれる形状を描画するメソッドです。

パスというのは、直線または曲線の任意の組み合わせによって構築される形状のことです。Android では、パスは、`Path` というクラスのオブジェクトによってあらわされます。

パスは、次のような手順によって描画することができます。

(1) `Path` クラスのオブジェクトを生成します。コンストラクタに引数を渡す必要はありませんので、

```
Path path = new Path();
```

というように書きます。生成された直後のオブジェクトは、線をまったく含まない、空のパスをあらわしています。

(2) `Path` クラスのオブジェクトが持っているメソッドを呼び出すことによって、直線や曲線をパスに追加します。

(3) `drawPath` を呼び出すことによって、パスを描画します。

5.4.2 カレントポイントの移動

パスは、仮想的なペンを使って紙の上に線を描くという考え方にもとづいて構築されます。ペンが存在している位置は、「カレントポイント」(current point) と呼ばれます。

パスに対して何も追加しないでカレントポイントを移動させたいときは、

- `void moveTo(float x, float y)`
- `void rMoveTo(float dx, float dy)`

というメソッドのどちらかを使います。`moveTo` のほうは引数を絶対座標として扱うメソッドで、`rMoveTo` のほうは引数を相対座標として扱うメソッドです。

「絶対座標」(absolute coordinate) というのは、座標系にもとづく絶対的な座標系のことです。それに対して、「相対座標」(relative coordinate) というのは、カレントポイントからの距離によって位置を指定する、相対的な座標のことです。

`moveTo` と `rMoveTo` のように、パスを構築するためのメソッドの多くは、絶対座標を扱うものと相対座標を扱うものとで、ペアになっています。

生成された直後の `Path` クラスのオブジェクトは、カレントポイントを持っていませんので、パスに線を追加するためには、それに先立って `moveTo` を呼び出しておく必要があります。

5.4.3 直線の追加

パスに直線を追加したいときは、

- `void lineTo(float x, float y)`
- `void rLineTo(float dx, float dy)`

というメソッドのどちらかを使います。`lineTo` が絶対座標を扱うメソッドで、`rLineTo` が相対座標を扱うメソッドです。

これらのメソッドは、カレントポイントから出発して引数で指定された位置で終わる直線をパスに追加します。そして、引数で指定された位置へカレントポイントを移動させます。

5.4.4 曲線の追加

パスに曲線（3 次ベジェ曲線、cubic Bézier curve）を追加したいときは、

- `void cubicTo(`
  `float x1, float y1, float x2, float y2, float x3, float y3)`
- `void rCubicTo(`
  `float x1, float y1, float x2, float y2, float x3, float y3)`

というメソッドのどちらかを使います。`cubicTo` が絶対座標を扱うメソッドで、`rCubicTo` が相対座標を扱うメソッドです。

これらのメソッドは、カレントポイントから出発して (x_3, y_3) で終わる曲線をパスに追加します。そして、カレントポイントを (x_3, y_3) へ移動させます。曲線の形は、(x_1, y_1) と (x_2, y_2) によって制御されます。

5.4.5 形状を閉じるメソッド

閉じた形状、つまり線が環状になっている形状を作りたいときは、

```
void close()
```

というメソッドを使います。形状の最初の位置（最後に `movoTo` または `rMoveTo` で移動したときの移動先）とカレントポイントとが同じではない場合は、それらをつなぐ直線をパスに追加したのちに形状を閉じます。

5.4.6 パスを描画するアプリケーションの例

それでは、パスを描画するアプリケーションを作ってみましょう。

次のようなプロジェクトを作成してください。

Project name	`path`
Application name	パスの描画
Package name	`org.example.path`
Create Activity	`PathActivity`

次に、`PathActivity.java` を次のように書き換えてください。

プログラムの例 `PathActivity.java`

```
package org.example.path;

import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.graphics.Color;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Path;

public class PathActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new CanvasView(this));
    }
}

class CanvasView extends View {
    CanvasView(Context context) {
        super(context);
        setBackgroundColor(Color.WHITE);
    }

    @Override
    protected void onDraw(Canvas canvas) {
```

```
        Paint paint = new Paint();
        paint.setColor(Color.rgb(255, 128, 0));
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(10);
        Path path = new Path();
        path.moveTo(100, 150);
        path.rLineTo(50, -100);
        path.rLineTo(50, 100);
        path.close();
        path.moveTo(150, 250);
        path.rCubicTo(60, -80, 100, 60, 0, 90);
        path.rCubicTo(-100, -30, -60, -170, 0, -90);
        path.close();
        canvas.drawPath(path, paint);
    }
}
```

これで完成です。実行すると、三角形とハート形が描画されるはずです。

5.5 テキスト

5.5.1 テキストの描画の基礎

キャンバスが持っているメソッドのひとつに、

```
void drawText(String text, float x, float y, Paint paint)
```

というものがあります。これは、テキストを描画するメソッドです。1 個目の引数は描画するテキスト、2 個目と 3 個目の引数は描画する位置を指定する x 座標と y 座標です。

5.5.2 テキストの大きさ

描画するテキストの大きさを指定したいときは、

```
void setTextSize(float textSize)
```

というメソッドを使って、ペイントに対してテキストの大きさを設定します。

5.5.3 書体

テキストを描画するときに使われる書体 (typeface) を指定したいときは、

```
Typeface setTypeface(Typeface typeface)
```

というメソッドを使って、ペイントに対して書体を設定します。

このメソッドには、引数として、書体のオブジェクトを渡します。書体のオブジェクトというのは、

```
android.graphics.Typeface
```

というクラスのオブジェクトのことです。このクラスでは、次のような書体のオブジェクトが定数として定義されています。

DEFAULT	デフォルトの書体。
SERIF	セリフ（ひげ飾りのある書体)。
SANS_SERIF	サンセリフ（ひげ飾りのない書体)。
MONOSPACE	横幅が一定の書体。

5.5.4 書体のスタイル

書体のオブジェクトには、デフォルトではノーマルなスタイルが設定されていますが、Typeface クラスが持っている、

```
static Typeface create(Typeface family, int style)
```

という静的メソッドを呼び出すことによって、太字またはイタリックが設定された書体を生成するということも可能です。

このメソッドは、1 個目の引数として受け取ったオブジェクトによってあらわされている書体と、2 個目の引数として受け取った整数によってあらわされているスタイルを持つ書体のオブジェクトを生成して、それを戻り値として返します。

スタイルをあらわす整数は、Typeface クラスで定義されている、次の定数を使って記述します。

NORMAL	ノーマルなスタイル。
BOLD	太字。
ITALIC	イタリック。
BOLD_ITALIC	太字でかつイタリック。

5.5.5 テキストを描画するアプリケーションの例

それでは、テキストを描画するアプリケーションを作ってみましょう。

次のようなプロジェクトを作成してください。

Project name	text
Application name	テキストの描画
Package name	org.example.text
Create Activity	TextActivity

次に、TextActivity.java を次のように書き換えてください。

プログラムの例 TextActivity.java

```
package org.example.text;

import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.graphics.Color;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Typeface;

public class TextActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new CanvasView(this));
    }
}

class CanvasView extends View {
    CanvasView(Context context) {
        super(context);
        setBackgroundColor(Color.WHITE);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        paint.setColor(Color.rgb(0, 128, 0));
        paint.setTextSize(20);
        paint.setTypeface(Typeface.SERIF);
        canvas.drawText("SERIF", 50, 50, paint);
        paint.setTypeface(Typeface.SANS_SERIF);
        canvas.drawText("SANS_SERIF", 50, 80, paint);
        paint.setTypeface(Typeface.MONOSPACE);
        canvas.drawText("MONOSPACE", 50, 110, paint);
        paint.setTypeface(
            Typeface.create(Typeface.SERIF, Typeface.BOLD));
        canvas.drawText("SERIF BOLD", 50, 140, paint);
        paint.setTypeface(
```

```
            Typeface.create(Typeface.SERIF, Typeface.ITALIC));
        canvas.drawText("SERIF ITALIC", 50, 170, paint);
        paint.setTypeface(
            Typeface.create(Typeface.SERIF, Typeface.BOLD_ITALIC));
        canvas.drawText("SERIF BOLD ITALIC", 50, 200, paint);
    }
}
```

これで完成です。実行すると、さまざまな書体を持つテキストが描画されるはずです。

5.5.6 パスに沿ったテキスト

drawTextは、水平方向に文字を並べたテキストしか描画できませんが、

```
void drawTextOnPath(String text, Path path,
    float hOffset, float vOffset, Paint paint)
```

というメソッドを使うことによって、パスに沿ってテキストを描画する、ということも可能です。このメソッドの1個目の引数は描画するテキスト、2個目の引数はテキストをそれに沿って描画するパスのオブジェクト、3個目の引数はテキストの描画を開始する位置、4個目の引数はテキストとパスとの距離です。

5.5.7 パスに沿ったテキストを描画するアプリケーションの例

それでは、パスに沿ったテキストを描画するアプリケーションを作ってみましょう。
次のようなプロジェクトを作成してください。

Project name	textonpath
Application name	パスに沿ったテキストの描画
Package name	org.example.textonpath
Create Activity	TextOnPathActivity

次に、TextOnPathActivity.javaを次のように書き換えてください。

プログラムの例 TextOnPathActivity.java

```
package org.example.textonpath;

import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.graphics.Color;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.Typeface;

public class TextOnPathActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new CanvasView(this));
    }
}

class CanvasView extends View {
    CanvasView(Context context) {
        super(context);
        setBackgroundColor(Color.WHITE);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        paint.setColor(Color.rgb(128, 0, 255));
```

```
        paint.setTextSize(24);
        paint.setTypeface(Typeface.SERIF);
        Path path = new Path();
        path.moveTo(100, 200);
        path.rCubicTo(-150, -200, 250, -200, 100, 0);
        canvas.drawTextOnPath(
            "Yoda: Do, or do not. There is no try.",
            path, 6, -4, paint);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(3);
        canvas.drawPath(path, paint);
    }
}
```

これで完成です。実行すると、パスに沿ったテキストが描画されるはずです。

5.6 ビットマップ画像

5.6.1 ビットマップ画像のリソース ID を求める式

ビューの上には、ビットマップ画像 (bitmap image) を描画するということも可能です。

ビットマップ画像を描画するためには、それが格納されているファイルを、`res/drawable` というディレクトリの下に置くことによって、それをリソースとして扱うことができるようにしておく必要があります。

`res/drawable` というディレクトリは、プロジェクトを作成したときに自動的には作成されません。その代わりとして、`drawable-hdpi`、`drawable-mdpi`、`drawable-ldpi` という三つのディレクトリが `res` の下に作成されています。これらのディレクトリは、画面の解像度に応じて適切なビットマップ画像が選択されるようにしたいというときに使われるものです。

ビットマップ画像を識別するためのリソース ID は、

```
R.drawable.ファイル名から拡張子を除いたもの
```

という形の式を評価することによって求めることができます。たとえば、ビットマップ画像のファイル名が `sample.png` だとするならば、

```
R.drawable.sample
```

という式の値が、それを識別するリソース ID になります。

5.6.2 ビットマップ画像の取得

Java のソースの中でビットマップ画像を取得したいときは、まず、ビューが持っている、

```
Resources getResources()
```

というメソッドを呼び出して、その戻り値が持っている、

```
Drawable getDrawable(int id)
```

というメソッドを呼び出します。このメソッドに、ビットマップ画像のリソース ID を引数として渡すと、そのビットマップ画像が戻り値として得られます。ただし、その戻り値は、

```
android.graphics.drawable.BitmapDrawable
```

というクラスにキャストする必要があります。

たとえば、`View` のサブクラスで、

```
private BitmapDrawable bitmap;
```

というフィールドを宣言しておいて、コンストラクタの中で、

```
bitmap = (BitmapDrawable) getResources()
            .getDrawable(R.drawable.sample);
```

という文を実行することによって、そのフィールドにビットマップ画像を設定することができます。

5.6.3　位置と大きさの設定

ビットマップ画像をビューの上に描画するためには、あらかじめ、そのビットマップ画像に対して位置と大きさを設定しておく必要があります。

ビットマップ画像に対する位置と大きさの設定には、

```
void setBounds(int left, int top, int right, int bottom)
```

というメソッドを使います。このメソッドに渡す引数は、1 個目が左端の x 座標、2 個目が上端の y 座標、3 個目が右端の x 座標、4 個目が下端の y 座標です。

5.6.4　ビットマップ画像の描画

ビットマップ画像は、それが持っている、

```
void draw(Canvas canvas)
```

というメソッドを呼び出すことによって、ビューの上に描画することができます。

5.6.5　ビットマップ画像を描画するアプリケーションの例

それでは、ビットマップ画像を描画するアプリケーションを作ってみましょう。

次のようなプロジェクトを作成してください。

Project name	`bitmap`
Application name	ビットマップ画像
Package name	`org.example.bitmap`
Create Activity	`BitmapActivity`

次に、`res/drawable` というディレクトリを作って、`sample.xxx` という名前を持つビットマップ画像のファイル（拡張子は、`png`、`jpg`、`gif` など）をその下に置いてください。

次に、`BitmapActivity.java` を次のように書き換えてください。

プログラムの例　`BitmapActivity.java`

```
package org.example.bitmap;

import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.graphics.Color;
import android.graphics.Canvas;
import android.graphics.drawable.BitmapDrawable;

public class BitmapActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new CanvasView(this));
    }
}

class CanvasView extends View {
    private BitmapDrawable bitmap;

    CanvasView(Context context) {
        super(context);
        setBackgroundColor(Color.WHITE);
        bitmap = (BitmapDrawable) getResources()
                    .getDrawable(R.drawable.sample);
        bitmap.setBounds(50, 50, 200, 200);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        bitmap.draw(canvas);
```

```
    }
}
```

これで完成です。実行すると、sample.xxx に格納されているビットマップ画像が描画されるはずです。

5.7 タッチ

5.7.1 タッチの基礎

Android では、ユーザーが指で画面に触れることを、「タッチする」(touch) と言います[1]。
ビューは、タッチによってイベントが発生すると、

```
boolean onTouchEvent(MotionEvent event)
```

というメソッドを呼び出します。ですから、このメソッドを、View を継承するクラスでオーバーライドすることによって、タッチによってイベントが発生したときに実行される動作を記述することができます。

onTouchEvent は、引数として、

```
android.view.MotionEvent
```

というクラスのオブジェクトを受け取ります。このオブジェクトは、発生したイベントについてのさまざまな情報を持っています。

onTouchEvent は、戻り値として、イベントを処理した場合は真、処理しなかった場合は偽を返すように定義します。

5.7.2 アクションの識別

onTouchEvent が受け取った引数は、イベントを発生させたアクション (action) を識別する整数を持っています。この整数は、

```
int getAction()
```

というメソッドを呼び出すことによって取得することができます。
アクションを識別する整数は、MothonEvent クラスの中で定数として定義されています。アクションを識別する整数の定数としては、たとえば次のようなものがあります。

ACTION_DOWN	指が触れるというアクション。
ACTION_UP	指が離れるというアクション。
ACTION_MOVE	触れたままで指が移動するというアクション。

5.7.3 位置の取得

onTouchEvent が受け取った引数は、イベントが発生したときの指の位置をあらわす座標を持っています。x 座標と y 座標のそれぞれは、

- `float getX()`
- `float getY()`

というメソッドを呼び出すことによって取得することができます。

5.7.4 強制的な再描画

何らかのイベントが発生したときにグラフィックスを強制的に再描画したいときは、

```
void invalidate()
```

というメソッドを呼び出します。そうすると、現在のグラフィックスが無効になりますので、onDraw が自動的に呼び出されて、再描画が実行されます。

5.7.5 タッチを処理するアプリケーションの例

それでは、タッチを処理するアプリケーションを作ってみましょう。
次のようなプロジェクトを作成してください。

[1] エミュレーター上では、マウスを使うことによってタッチをシミュレートすることができます。

Project name `touch`
Application name タッチ
Package name `org.example.touch`
Create Activity `TouchActivity`

次に、`TouchActivity.java`を次のように書き換えてください。

プログラムの例 `TouchActivity.java`

```
package org.example.touch;

import java.util.ArrayList;
import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.view.MotionEvent;
import android.graphics.Point;
import android.graphics.Color;
import android.graphics.Canvas;
import android.graphics.Paint;

public class TouchActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new CanvasView(this));
    }
}

class CanvasView extends View {
    private static final int RADIUS = 10;
    private ArrayList<Point> alp;

    CanvasView(Context context) {
        super(context);
        setBackgroundColor(Color.WHITE);
        alp = new ArrayList<Point>();
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        int action = event.getAction();
        if (action == MotionEvent.ACTION_DOWN) {
            Point p = new Point();
            p.x = (int) event.getX();
            p.y = (int) event.getY();
            alp.add(p);
            invalidate();
        }
        return true;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        paint.setColor(Color.rgb(0, 0, 128));
        for (int i = 0; i < alp.size(); i++) {
            Point p = alp.get(i);
            canvas.drawCircle(p.x, p.y, RADIUS, paint);
        }
    }
}
```

これで完成です。実行して、ビューをマウスでクリックしてみてください。そうすると、クリッ

クした位置に、小さな円が描画されるはずです。

5.8 キー

5.8.1 キーの基礎

Android を搭載している機器の表面に取り付けられた各種のボタンは、「キー」(key) と呼ばれます。人間がキーを操作すると、イベントが発生します。

ビューは、人間によってキーが押されたというイベントが発生すると、

```
boolean onKeyDown(int keyCode, KeyEvent event)
```

というメソッドを呼び出します。また、押されていたキーが離されたというイベントが発生すると、

```
boolean onKeyUp(int keyCode, KeyEvent event)
```

というメソッドを呼び出します。ですから、これらのメソッドを、View を継承するクラスでオーバーライドすることによって、キーが操作されたときに実行される動作を記述することができます。

onKeyDown と onKeyUp は、それぞれ、2 個の引数を受け取ります。1 個目は、操作されたキーを識別する、「キーコード」(key code) と呼ばれる整数です。そして 2 個目は、

```
android.view.KeyEvent
```

というクラスのオブジェクトです。このオブジェクトは、発生したイベントについてのさまざまな情報を持っています。

onKeyDown と onKeyUp は、戻り値として、イベントを処理した場合は真、処理しなかった場合は偽を返すように定義します。

5.8.2 キーコード

キーコードは、KeyEvent クラスの中で定数として定義されています。いくつか例を挙げると、次のような定数です。

KEYCODE_0	数字のゼロ。
KEYCODE_A	英字の A。
KEYCODE_SPACE	スペースキー。
KEYCODE_ENTER	エンターキー。
KEYCODE_DPAD_CENTER	センターキー。
KEYCODE_DPAD_DOWN	下向き矢印キー。
KEYCODE_DPAD_LEFT	左向き矢印キー。
KEYCODE_DPAD_RIGHT	右向き矢印キー。
KEYCODE_DPAD_UP	上向き矢印キー。

5.8.3 フォーカス

キーの操作によって発生したイベントに反応するビューは、「フォーカス」(focus) を持っていると言われます。

ビューは、デフォルトではフォーカスを得ることができない状態に設定されています。この状態を変更して、フォーカスを得ることができるようにしたいときは、

- `void setFocusable(boolean focusable)`
- `void setFocusableInTouchMode(boolean focusableInTouchMode)`

という二つのメソッドを呼び出して、どちらのメソッドにも、引数として true を渡します。

5.8.4 キーに対する操作を処理するアプリケーションの例

それでは、キーに対する操作を処理するアプリケーションを作ってみましょう。

次のようなプロジェクトを作成してください。

Project name　　key

Application name キー

Package name org.example.key

Create Activity KeyActivity

次に、KeyActivity.javaを次のように書き換えてください。

プログラムの例 KeyActivity.java

```
package org.example.key;

import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.view.KeyEvent;
import android.graphics.Color;
import android.graphics.Canvas;
import android.graphics.Paint;

public class KeyActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new CanvasView(this));
    }
}

class CanvasView extends View {
    private static final int RADIUS = 20;
    private static final int STEP = 10;
    private int code = 0;
    private int x = 100;
    private int y = 100;

    CanvasView(Context context) {
        super(context);
        setBackgroundColor(Color.WHITE);
        setFocusable(true);
        setFocusableInTouchMode(true);
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        code = keyCode;
        switch (code) {
        case KeyEvent.KEYCODE_DPAD_CENTER:
            x = getWidth() / 2;
            y = getHeight() / 2;
            break;
        case KeyEvent.KEYCODE_DPAD_DOWN:
            y += STEP;
            break;
        case KeyEvent.KEYCODE_DPAD_LEFT:
            x -= STEP;
            break;
        case KeyEvent.KEYCODE_DPAD_RIGHT:
            x += STEP;
            break;
        case KeyEvent.KEYCODE_DPAD_UP:
            y -= STEP;
            break;
        }
        invalidate();
        return true;
    }
```

```
    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        paint.setColor(Color.rgb(255, 0, 128));
        paint.setTextSize(24);
        canvas.drawText("KeyCode=" + code, 10, 50, paint);
        canvas.drawCircle(x, y, RADIUS, paint);
    }
}
```

これで完成です。実行して、キーを押してみてください。そうすると、押されたキーのキーコードがビューの上に描画されるはずです。また、矢印キーとセンターキーを押すことによって、ビューの上に描画された円を移動させることもできるはずです。

5.9 OpenGL ES

5.9.1 OpenGL ES とは何か

Android には、OpenGL ES と呼ばれるライブラリが搭載されています。

OpenGL というのは、Khronos Group という標準化団体によって仕様が策定されている、2 次元または 3 次元のグラフィックスを描画するためのライブラリのことです。そして、OpenGL ES(OpenGL for Embedded Systems) というのは、組み込み機器で使われることを想定した OpenGL のサブセットのことです。

この節では、Android の上で OpenGL ES を使ってグラフィックスを描画する方法について説明したいと思います。ただし、OpenGL ES について詳細には説明しませんので、さらに深く勉強したい人は、OpenGL に関する文献を参照してください。

5.9.2 OpenGL ES によるグラフィックスの描画に必要なオブジェクト

Android の上で OpenGL によるグラフィックスの描画を実行するためには、最低限、次の二つのオブジェクトが必要になります。

- サーフェスビュー (surface view)
- レンダラー (renderer)

サーフェスビューというのは、グラフィックスがその上に描画される「サーフェス」(surface) と呼ばれるものを表示するためのビューのことです。Android には、

```
android.opengl.GLSurfaceView
```

という、OpenGL のためのサーフェスビューを生成するクラスが準備されています。

レンダラーというのは、描画を実行するメソッドを持っているオブジェクトのことです。Android で OpenGL による描画を実行する場合は、レンダラーとして、

```
android.opengl.GLSurfaceView.Renderer
```

というインターフェースを実装したクラスから生成されたオブジェクトを使います。OpenGL のためのサーフェスビューに対して、

```
void setRenderer(GLSurfaceView.Renderer renderer)
```

というメソッドを使ってレンダラーを設定しておくと、そのレンダラーがサーフェスビューの上にグラフィックスを描画することになります。

5.9.3 レンダラーを生成するクラス

`GLSurfaceView.Renderer` というインターフェースでは、次の三つのメソッドが宣言されています。

- `void onSurfaceCreated(GL10 gl, EGLConfig config)`
  サーフェスが生成されたときに呼び出されるメソッド。
- `void onDrawFrame(GL10 gl)`
  描画を実行するメソッド。

- `void onSurfaceChanged(GL10 gl, int width, int height)`
  サーフェスの大きさが変化したときに呼び出されるメソッド。

レンダラーは、これらのメソッドを実装したクラスから生成されます。

5.9.4 OpenGL ESを使ってグラフィックスを描画するアプリケーションの例

それでは、OpenGL ESを使ってグラフィックスを描画するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`opengl`
Application name	OpenGL
Package name	`org.example.opengl`
Create Activity	`OpenGLActivity`

次に、`Triangle`という名前の新しいクラスを作って、`Triangle.java`の内容を次のように書き換えてください。

プログラムの例 `Triangle.java`

```
package org.example.opengl;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;
import javax.microedition.khronos.opengles.GL10;

public class Triangle {
    private final static int VSIZE = 3; // size of vertex
    private final static int CSIZE = 4; // size of color
    private final static int FSIZE = 4; // size of float
    private final static int VERTS = 3; // number of vertices
    private FloatBuffer vfb;
    private FloatBuffer cfb;

    Triangle() {
        float vertices[] = {
            0.2f, 0.2f, 0.0f,
            0.8f, 0.2f, 0.0f,
            0.5f, 0.5f, 0.0f
        };
        float colors[] = {
            1.0f, 1.0f, 1.0f, 1.0f,
            1.0f, 1.0f, 0.0f, 1.0f,
            1.0f, 0.0f, 0.0f, 1.0f
        };
        ByteBuffer vbb =
            ByteBuffer.allocateDirect(VSIZE * FSIZE * VERTS);
        vbb.order(ByteOrder.nativeOrder());
        vfb = vbb.asFloatBuffer();
        vfb.put(vertices);
        vfb.position(0);
        ByteBuffer cbb =
            ByteBuffer.allocateDirect(CSIZE * FSIZE * VERTS);
        cbb.order(ByteOrder.nativeOrder());
        cfb = cbb.asFloatBuffer();
        cfb.put(colors);
        cfb.position(0);
    }

    public void draw(GL10 gl) {
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
        gl.glVertexPointer(VSIZE, GL10.GL_FLOAT, 0, vfb);
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glColorPointer(CSIZE, GL10.GL_FLOAT, 0, cfb);
        gl.glEnableClientState(GL10.GL_COLOR_ARRAY);
```

```
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, VERTS);
    }
}
```

このクラスで定義されている draw というメソッドは、白と黄色と赤で塗られた三角形を描画します。

次に、GLSurfaceView.Renderer というインターフェースを実装する、TriangleRenderer という名前の新しいクラスを作って、TriangleRenderer.java の内容を次のように書き換えてください。

プログラムの例　TriangleRenderer.java

```
package org.example.opengl;

import android.opengl.GLSurfaceView;
import android.opengl.GLU;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

public class TriangleRenderer implements GLSurfaceView.Renderer {
    private Triangle triangle;

    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        gl.glClearColor(0.0f, 0.0f, 0.4f, 1.0f);
        gl.glMatrixMode(GL10.GL_PROJECTION);
        gl.glLoadIdentity();
        GLU.gluOrtho2D(gl, 0.0f, 1.0f, 0.0f, 1.0f);
        triangle = new Triangle();
    }

    public void onDrawFrame(GL10 gl) {
        triangle.draw(gl);
    }

    public void onSurfaceChanged(GL10 gl, int width, int height) {
        gl.glViewport(0, 0, width, height);
    }
}
```

最後に、OpenGLActivity.java を次のように書き換えてください。

プログラムの例　OpenGLActivity.java

```
package org.example.opengl;

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

public class OpenGLActivity extends Activity {
    private GLSurfaceView glSurfaceView;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        glSurfaceView = new GLSurfaceView(this);
        glSurfaceView.setRenderer(new TriangleRenderer());
        setContentView(glSurfaceView);
    }
}
```

これで完成です。実行してみてください。そうすると、白と黄色と赤で塗られた三角形が描画されるはずです。

第 6 章　データの永続化

6.1　ファイルへの書き込み

6.1.1　出力ストリームの生成

Android のアプリケーションは、ファイルに対する読み書きを実行することができます。ただし、Android では、アプリケーションがアクセスすることのできるファイルは、自分がファイルを作る権限を持っているディレクトリに存在するものだけに限定されます。

ファイルへの書き込みを実行したいときは、そのための出力ストリームを生成する必要があります。Android では、

```
android.content.Context
```

というクラスを継承するクラス（Activity もそのひとつ）が持っている、

```
FileOutputStream openFileOutput(String name, int mode)
```

というメソッドを呼び出すことによって、出力ストリームを生成することができます。このメソッドに渡す引数の 1 個目は、そこにデータを書き込むファイルの名前です（パス名ではなくてファイル名ですので、ディレクトリを指定することはできません）。引数の 2 個目は、「ファイル作成モード」(file creation mode) と呼ばれる、書き込みやファイル作成のモードを指定するための整数です。ファイル作成モードは、通常、

```
android.content.Context.MODE_PRIVATE
```

という定数を指定します。

openFileOutput は、引数で指定された名前を持つファイルに対してデータを書き込むための出力ストリームを生成して、それを戻り値として返します。指定されたファイルが存在しない場合は、その名前を持つファイルが新しく作られます。

openFileOutput が戻り値として返すのは、

```
java.io.FileOutputStream
```

というクラスの出力ストリームです。この出力ストリームを使ってファイルにデータを書き込む方法は、Android アプリケーションではない普通のアプリケーションの場合と同じです。

6.1.2　ファイルにデータを書き込むアプリケーションの例

それでは、ファイルにデータを書き込むアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	filewrite
Application name	ファイルへの書き込み
Package name	org.example.filewrite
Create Activity	FileWriteActivity

次に、main.xml を次のように書き換えてください。

レイアウト XML の例　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"
    >
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1"
    >
<TableRow>
    <TextView android:text="ファイル名"/>
    <EditText android:id="@+id/filename"/>
</TableRow>
<TableRow>
    <TextView android:text="文字列"/>
<EditText android:id="@+id/edittext"
```

```
    android:layout_height="260sp"
    android:scrollbars="vertical"
    android:gravity="top"
    />
</TableRow>
</TableLayout>
<Button android:id="@+id/write"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="書き込む"
    />
</LinearLayout>
```

次に、FileWriteActivity.javaを次のように書き換えてください。

プログラムの例 FileWriteActivity.java

```
package org.example.filewrite;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.content.Context;
import android.util.Log;
import java.io.IOException;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.BufferedWriter;

public class FileWriteActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button forward = (Button) findViewById(R.id.write);
        forward.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                writeText();
            }
        });
    }

    private void writeText() {
        final EditText filename = (EditText) findViewById(R.id.filename);
        final EditText edittext = (EditText) findViewById(R.id.edittext);
        try {
            FileOutputStream fos =
                openFileOutput(filename.getText().toString(),
                    Context.MODE_PRIVATE);
            OutputStreamWriter osw = new OutputStreamWriter(fos);
            BufferedWriter bw = new BufferedWriter(osw);
            bw.write(edittext.getText().toString());
            bw.flush();
            bw.close();
        } catch (IOException e) {
            Log.d("FileWriteActivity", e.getMessage());
        }
    }
}
```

これで完成です。実行すると、2個のエディットテキストと1個のボタンが表示されるはずですので、1個目のエディットテキストにファイル名、2個目のエディットテキストに文字列を入力して、ボタンをクリックしてみてください。そうすると、このアプリケーションは、入力され

た文字列を、指定された名前を持つファイルに書き込みます。

6.1.3　ファイルの場所

Android アプリケーションがファイルを作成する権限を持っているディレクトリというのは、

```
/data/data/パッケージ名/files
```

というディレクトリです。したがって、先ほど作ったアプリケーションは、

```
/data/data/org.example.filewrite/files
```

というディレクトリにファイルを作っているはずです。

それでは、先ほど作ったアプリケーションが本当にファイルにデータを書き込んだかどうか、Android エミュレーターのシェルを使って確認してみましょう。まず、

```
adb shell
```

というコマンドでシェルを起動して、次に、

```
cat /data/data/org.example.filewrite/files/ファイル名
```

というコマンドを入力してください。そうすると、先ほどファイルに書き込んだ文字列が出力されるはずです。

6.2　ファイルからの読み込み

6.2.1　入力ストリームの生成

ファイルからの読み込みを実行したいときは、そのための入力ストリームを生成する必要があります。Android では、`Context` クラスを継承するクラスが持っている、

```
FileInputStream openFileInput(String name)
```

というメソッドを呼び出すことによって、入力ストリームを生成することができます。このメソッドに渡す引数は、そこからデータを読み込むファイルの名前です。

`openFileInput` は、引数で指定された名前を持つファイルからデータを読み込むための入力ストリームを生成して、それを戻り値として返します。

`openFileInput` が戻り値として返すのは、

```
java.io.FileInputStream
```

というクラスの入力ストリームです。この入力ストリームを使ってファイルからデータを読み込む方法は、Android アプリケーションではない普通のアプリケーションの場合と同じです。

6.2.2　ファイルからデータを読み込むアプリケーションの例

それでは、ファイルからデータを読み込むアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`fileread`
Application name	ファイルからの読み込み
Package name	`org.example.fileread`
Create Activity	`FileReadActivity`

次に、`main.xml` を次のように書き換えてください。

レイアウト XML の例　`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"
    >
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1"
```

```
    >
<TableRow>
    <TextView android:text="ファイル名"/>
    <EditText android:id="@+id/filename"/>
</TableRow>
</TableLayout>
<Button android:id="@+id/read"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="読み込む"
    />
<TextView android:id="@+id/result"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、FileReadActivity.javaを次のように書き換えてください。

プログラムの例 FileReadActivity.java

```
package org.example.fileread;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.widget.TextView;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.io.BufferedReader;

public class FileReadActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button forward = (Button) findViewById(R.id.read);
        forward.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                readFile();
            }
        });
    }

    private void readFile() {
        final EditText filename = (EditText) findViewById(R.id.filename);
        StringBuffer sb = new StringBuffer();
        try {
            FileInputStream fis =
                openFileInput(filename.getText().toString());
            InputStreamReader isw = new InputStreamReader(fis);
            BufferedReader br = new BufferedReader(isw);
            String line;
            while ((line = br.readLine()) != null) {
                sb.append(line);
                sb.append(System.getProperty("line.separator"));
            }
            br.close();
        } catch (IOException e) {
            sb.append(e.toString());
        }
        final TextView result = (TextView) findViewById(R.id.result);
        result.setText(sb.toString());
```

```
    }
}
```

アプリケーションはこれで完成です。ただし、このアプリケーションが正しく動作するかどうかを確かめるためには、読み込みの対象となるテキストファイルを作る必要があります。

6.2.3　ファイルの作成

先ほど作ったアプリケーションの動作を確認するために、次のような手順でテキストファイルを作ってください。

(1) 次のコマンドで、シェルを起動します。

```
adb shell
```

(2) 次のコマンドで、カレントディレクトリを変更します。

```
cd data/data/org.example.fileread
```

(3) 次のコマンドで、`files` という名前のディレクトリを作ります。

```
mkdir files
```

(4) 次のコマンドで、カレントディレクトリを変更します。

```
cd files
```

(5) 次のコマンドで、ファイルに格納する文字列の入力を開始します。

```
cat > ファイル名
```

(6) ファイルの内容として、何行かの文字列を入力します。

(7) 入力を終了させるために、コントロールキーを押しながら D のキーを押して、エンターキーを押します。

このような手順でファイルを作成したのち、先ほどのアプリケーションを実行してみてください。ファイル名を入力してボタンをクリックすると、ファイルの内容が表示されるはずです。

6.3　ファイル名のリスト

6.3.1　ファイル名のリストの取得

Android アプリケーションは、ファイルを作成する権限を持っているディレクトリに存在しているファイルの名前のリストを取得することができます。

ファイル名のリストを取得したいときは、`Context` クラスを継承するクラスが持っている、

```
String[] fileList()
```

というメソッドを呼び出します。このメソッドは、ファイル名を要素とする配列を戻り値として返します。

6.3.2　ファイル名のリストを取得するアプリケーションの例

それでは、ファイル名のリストを取得するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`filelist`
Application name	ファイル名のリストの取得
Package name	`org.example.filelist`
Create Activity	`FileListActivity`

次に、`main.xml` を次のように書き換えてください。

レイアウト XML の例　`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/list"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ファイル名のリストの取得"
    />
<TextView android:id="@+id/result"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、FileListActivity.javaを次のように書き換えてください。

プログラムの例 FileListActivity.java

```
package org.example.filelist;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class FileListActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button forward = (Button) findViewById(R.id.list);
        forward.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showFileList();
            }
        });
    }

    private void showFileList() {
        String name[] = fileList();
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < name.length; i++) {
            sb.append(name[i]);
            sb.append(System.getProperty("line.separator"));
        }
        final TextView result = (TextView) findViewById(R.id.result);
        result.setText(sb.toString());
    }
}
```

アプリケーションはこれで完成です。ただし、このアプリケーションが正しく動作するかどうかを確かめるためには、ファイルをいくつか作る必要があります。

このアプリケーションがファイルを作成する権限を持っているのは、

```
/data/data/org.example.filelist/files
```

というディレクトリですので、このディレクトリを作って、そこをカレントディレクトリにして、

```
echo namako > namako.txt
```

というようなコマンドでファイルをいくつか作成してください。

そののち、先ほどのアプリケーションを実行して、ボタンをクリックしてみてください。そうすると、ファイル名のリストが表示されるはずです。

6.4　プリファレンス

6.4.1　プリファレンスの基礎

Android は、通常のファイルに対する読み書きの機能に加えて、「プリファレンス」(preference) と呼ばれる機能を持っています。

プリファレンスというのは、キーと値の組から構成されるデータを永続的に保存する機能のことです。プリファレンスを保存するために使われるファイルは、「プリファレンスファイル」(preference file) と呼ばれます。プリファレンスファイルは、

```
/data/data/パッケージ名/shared_prefs
```

というディレクトリに作られます。

プリファレンスファイルの内容は、`map` という要素型の要素をルート要素とする XML 文書です。キーと値の組は、`map` 要素の子供になっている、`string` という要素型の要素によってあらわされます。`string` 要素は、

```
<string name="キー">値</string>
```

という形で、キーと値の組を記述します。

6.4.2　プリファレンスオブジェクトの取得

プリファレンスを利用するためには、書き込みの場合も読み込みの場合も、

```
android.content.SharedPreferences
```

というインターフェースを実装したクラスのオブジェクト（このオブジェクトを「プリファレンスオブジェクト」(preference object) と呼ぶことにします）を取得する必要があります。プリファレンスオブジェクトは、アクティビティーが持っている、

```
SharedPreferences getSharedPreferences(String name, int mode)
```

というメソッドを呼び出すことによって取得することができます。このメソッドに渡す引数の 1 個目は、プリファレンスファイルの名前です。`.xml` という拡張子が自動的に付加されますので、その左側の部分だけを渡します。そして引数の 2 個目は、ファイル作成モードです。通常、

```
android.content.Context.MODE_PRIVATE
```

という定数の値を渡します。

6.4.3　エディターオブジェクトの生成

プリファレンスファイルに対する書き込みを実行するためには、

```
android.content.SharedPreferences.Editor
```

というインターフェースを実装したクラスのオブジェクト（このオブジェクトを「エディターオブジェクト」(editor object) と呼ぶことにします）を生成する必要があります。

エディターオブジェクトは、プリファレンスオブジェクトが持っている、

```
SharedPreferences.Editor edit()
```

というメソッドを呼び出すことによって生成することができます。

6.4.4　プリファレンスファイルへの書き込み

プリファレンスファイルに対してキーと値の組を書き込みたいときは、まず、エディターオブジェクトに対して、その組を設定します。組の値の型に応じて、次のメソッドのいずれかを呼び出すことによって、エディターオブジェクトに対して組を設定することができます。

- `SharedPreferences.Editor putBoolean(String key, boolean value)`
- `SharedPreferences.Editor putInt(String key, int value)`
- `SharedPreferences.Editor putLong(String key, long value)`
- `SharedPreferences.Editor putFloat(String key, float value)`
- `SharedPreferences.Editor putString(String key, String value)`

これらのメソッドに渡す引数は、1 個目がキーで、2 個目が値です。

エディターオブジェクトに組を設定した段階では、その組は、まだプリファレンスファイルに書き込まれていません。エディターオブジェクトに設定されている組は、

```
boolean commit()
```

というメソッドが呼び出されることによって、プリファレンスファイルに書き込まれます。

6.4.5 プリファレンスファイルからの読み込み

プリファレンスファイルから組を読み込みたいときは、組の値の型に応じて、プリファレンスオブジェクトが持っている次のメソッドのいずれかを呼び出します。

- `boolean getBoolean(String key, boolean defValue)`
- `int getInt(String key, int defValue)`
- `long getLong(String key, long defValue)`
- `float getFloat(String key, float defValue)`
- `String getString(String key, String defValue)`

これらのメソッドに渡す引数は、1 個目がキーで、2 個目は、そのキーを持つ組が存在しなかった場合に返すべき値です。呼び出したメソッドが返す値の型と、キーに対応する値の型とが一致していない場合は、`ClassCastException` という例外が発生します。

6.4.6 プリファレンスを利用するアプリケーションの例

それでは、プリファレンスファイルに対するデータの読み込みと書き込みを実行するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`preference`
Application name	プリファレンス
Package name	`org.example.preference`
Create Activity	`PreferenceActivity`

次に、`main.xml` を次のように書き換えてください。

レイアウト XML の例 `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"
    >
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1"
    >
<TableRow>
    <TextView android:text="キー"/>
    <EditText android:id="@+id/putkey"/>
</TableRow>
<TableRow>
    <TextView android:text="値"/>
    <EditText android:id="@+id/putvalue"/>
</TableRow>
</TableLayout>
<Button android:id="@+id/put"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="書き込み"
    />
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1"
```

```
    >
<TableRow>
    <TextView android:text="キー"/>
    <EditText android:id="@+id/getkey"/>
</TableRow>
</TableLayout>
<Button android:id="@+id/get"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="読み込み"
    />
<TextView android:id="@+id/getvalue"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、PreferenceActivity.java を次のように書き換えてください。

プログラムの例　PreferenceActivity.java

```
package org.example.preference;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.widget.TextView;
import android.content.SharedPreferences;

public class PreferenceActivity extends Activity {
    public static final String FILENAME = "preference";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button put = (Button) findViewById(R.id.put);
        put.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                put();
            }
        });
        final Button get = (Button) findViewById(R.id.get);
        get.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                get();
            }
        });
    }

    private void put() {
        final EditText putkey =
            (EditText) findViewById(R.id.putkey);
        final EditText putvalue =
            (EditText) findViewById(R.id.putvalue);
        String key = putkey.getText().toString();
        String value = putvalue.getText().toString();
        SharedPreferences sp = getSharedPreferences(
            FILENAME, MODE_PRIVATE);
        SharedPreferences.Editor editor = sp.edit();
        editor.putString(key, value);
        editor.commit();
        putkey.setText("");
        putvalue.setText("");
```

```
    }

    private void get() {
        final EditText getkey =
            (EditText) findViewById(R.id.getkey);
        final TextView getvalue =
            (TextView) findViewById(R.id.getvalue);
        String key = getkey.getText().toString();
        SharedPreferences sp = getSharedPreferences(
            FILENAME, MODE_PRIVATE);
        getvalue.setText("値: " + sp.getString(key, "undefined"));
    }
}
```

これで完成です。実行すると、三つのエディットテキストと二つのボタンが表示されるはずです。

1 番目のエディットテキストにキー、2 番目のエディットテキストに値を入力して、「書き込み」ボタンをクリックすると、そのキーと値の組がプリファレンスファイルに書き込まれます。

また、3 番目のエディットテキストにキーを入力して、「読み込み」ボタンをクリックすると、そのキーを持つ組の値が、ボタンの下に表示されます。

6.5 SQLite

6.5.1 この節について

Android には、データベース管理システム (database management system, DBMS) が標準で搭載されています。ですから、Android では、データの永続化の手段として、データベースを手軽に扱うことができます。

Android に標準で搭載されているデータベース管理システムは、SQLite というものです。この節では、Android アプリケーションの中で SQLite を使ってデータベースを操作する方法について説明したいと思います。

6.5.2 データベースヘルパー

Android アプリケーションの中で SQLite を使うためには、「データベースヘルパー」(database helper) と呼ばれるオブジェクトを作る必要があります。

データベースヘルパーは、

```
android.database.sqlite.SQLiteOpenHelper
```

という抽象クラスを継承するクラスから生成されるオブジェクトです。

SQLiteOpenHelper のサブクラスでは、コンストラクタを定義する必要があります。そして、そのためにはデータベースのファイル名とバージョン番号（整数）が必要になります。ですから、あらかじめ、

```
private static final String DATABASE_NAME = "namako.db";
private static final int DATABASE_VERSION = 1;
```

というように、それらを定数として定義しておくとよいでしょう。

コンストラクタは、ファイル名やバージョン番号などを引数にして、

```
SQLiteOpenHelper(Context context, String name,
    SQLiteDatabase.CursorFactory factory, int version)
```

というスーパークラスのコンストラクタを呼び出すように定義します（3 個目の引数は null でかまいません）。つまり、

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

というように書けばいいわけです。

SQLiteOpenHelper のサブクラスを定義するときには、かならず、次の二つの抽象メソッドを実装しないといけません。

- `void onCreate(SQLiteDatabase db)`
- `void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)`

これらのメソッドが受け取る 1 個目の引数は、「データベースオブジェクト」(database object) と呼ばれるオブジェクトです。データベースに対するさまざまな操作は、このオブジェクトが持っているメソッドを呼び出すことによって実行することができます。

`onCreate` は、データベースが最初に生成されるときに呼び出されるメソッドです。このメソッドでは、通常、テーブルを生成する SQL 文を実行します。SQL 文は、データベースオブジェクトが持っている、

```
void execSQL(String sql)
```

というメソッドを使うことによって実行することができます。

`onUpgrade` は、データベースのアップグレードが必要になったときに呼び出されるメソッドです。

データベースに対する操作をしたいときは、まず、

```
DatabaseHelper helper = new DatabaseHelper(this);
```

というようにデータベースヘルパーを生成して、そののち、次の二つのメソッドのうちのどちらかを使って、データベースヘルパーからデータベースオブジェクトを取得します。

- `SQLiteDatabase getReadableDatabase()`
  読み込み専用のデータベースオブジェクトを取得する。
- `SQLiteDatabase getWritableDatabase()`
  読み込みも書き込みもできるデータベースオブジェクトを取得する。

データベースをクローズしたいときは、データベースオブジェクトが持っている次のメソッドを呼び出します。

```
void close()
```

6.5.3　行の挿入

データベースに行を挿入したいときは、データベースオブジェクトが持っている、

```
long insert(String table, String nullColumnHack, ContentValues values)
```

というメソッドを使います。1 個目の引数はテーブルの名前で、3 個目の引数は、挿入する行のデータが設定された、

```
android.content.ContentValues
```

というクラスのオブジェクトです。

`ContentValues` クラスのオブジェクトは、

- `void put(String key, String value)`
- `void put(String key, Integer value)`

などのメソッドを持っています。これらのメソッドを呼び出して、1 個目の引数として列の名前、2 個目の引数として列の値を渡すことによって、`ContentValues` クラスのオブジェクトに行のデータを設定することができます。

6.5.4　カーソル

データベースオブジェクトは、

```
Cursor query(String table, String[] columns, String selection,
    String[] selectionArgs, String groupBy, String having,
    String orderBy)
```

というメソッドを持っています。このメソッドを呼び出すことによって、データベースを検索して、その結果をカーソルとして取得することができます。カーソルは、

```
android.database.Cursor
```

というインターフェースで宣言されている、次のようなメソッドを使うことによって操作することができます。

- `boolean moveToFirst()`
  先頭の行へ移動する。カーソルが空の場合は偽を返す。
- `boolean moveToNext()`
  次の行へ移動する。次の行が存在しない場合は偽を返す。
- `int getColumnIndexOrThrow(String columnName)`
  列の名前からその列のインデックスを取得する。
- `String getString(int columnIndex)`
  現在の行から、インデックスで指定された列の内容を取得して、文字列で返す。
- `int getInt(int columnIndex)`
  現在の行から、インデックスで指定された列の内容を取得して、整数で返す。
- `void close()`
  カーソルをクローズする。

6.5.5 SQLite を使ってデータベースを操作するアプリケーションの例

それでは、SQLite を使ってデータベースを操作するアプリケーションを作ってみましょう。
まず最初に、次のようなプロジェクトを作成してください。

Project name	`sqlite`
Application name	SQLite
Package name	`org.example.sqlite`
Create Activity	`SQLiteActivity`

次に、`main.xml` を次のように書き換えてください。

レイアウト XML の例　`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/memo"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<Button android:id="@+id/insert"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="メモの挿入"
    />
<TextView android:id="@+id/result"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、`SQLiteOpenHelper` をスーパークラスとする、`DatabaseHelper` という名前の新しいクラスを作って、`DatabaseHelper.java` の内容を次のように書き換えてください。

プログラムの例　`DatabaseHelper.java`

```
package org.example.sqlite;

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

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "notepad.db";
    private static final int DATABASE_VERSION = 1;

    public DatabaseHelper(Context context) {
```

```
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(
            "create table if not exists notepad (" +
            "    id integer primary key autoincrement," +
            "    memo text )");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db,
            int oldVersion, int newVersion) {
        db.execSQL("drop table if exists notepad");
        onCreate(db);
    }
}
```

次に、SQLiteActivity.java を次のように書き換えてください。

プログラムの例　SQLiteActivity.java

```
package org.example.sqlite;

import android.app.Activity;
import android.os.Bundle;
import android.database.sqlite.SQLiteDatabase;
import android.database.Cursor;
import android.content.ContentValues;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.widget.TextView;

public class SQLiteActivity extends Activity {
    private DatabaseHelper helper;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button write = (Button) findViewById(R.id.insert);
        write.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                insertRow();
                showTable();
            }
        });
        helper = new DatabaseHelper(this);
        showTable();
    }

    private void insertRow() {
        final EditText ememo = (EditText) findViewById(R.id.memo);
        String memo = ememo.getText().toString();
        ememo.setText("");
        ContentValues values = new ContentValues();
        values.put("memo", memo);
        SQLiteDatabase db = helper.getWritableDatabase();
        db.insert("notepad", null, values);
        db.close();
    }

    private void showTable() {
        SQLiteDatabase db = helper.getReadableDatabase();
```

```
        Cursor c = db.query("notepad", new String[] {"id", "memo"},
            null, null, null, null, null);
        StringBuffer sb = new StringBuffer();
        while (c.moveToNext()) {
            sb.append(c.getInt(0));
            sb.append("|");
            sb.append(c.getString(1));
            sb.append(System.getProperty("line.separator"));
        }
        c.close();
        db.close();
        final TextView result = (TextView) findViewById(R.id.result);
        result.setText(sb.toString());
    }
}
```

これで完成です。実行してみてください。エディットテキストに文字列を入力して、「メモの挿入」というボタンをクリックすると、入力した文字列を含む行がデータベースに挿入されます。

6.6 組み込みコンテントプロバイダー

6.6.1 組み込みコンテントプロバイダーの基礎

Android アプリケーションは、「コンテントプロバイダー」(ontent provider) と呼ばれるオブジェクトを経由することによって、別のアプリケーションが公開しているファイルやデータベースに対して、データの参照や追加などの操作を実行することができます。

Android 自身にも、いくつかのコンテントプロバイダーが組み込まれています。そのようなコンテントプロバイダーは、「組み込みコンテントプロバイダー」(built-in content provider) と呼ばれます。組み込みコンテントプロバイダーを使って Android が公開しているのは、通話ログ、コンタクトリスト、ブラウザーのブックマーク、システム設定値などです。

この節では、組み込みコンテントプロバイダーを利用してブックマークを取得するアプリケーションと、ブックマークを保存するアプリケーションの作り方について説明したいと思います。

6.6.2 ブックマークに対する操作の許可

ブックマークの取得を実行するアプリケーションを作るためには、Android マニフェストの中に、ブックマークの取得を許可する記述を書いておく必要があります。同じように、ブックマークの変更を実行するアプリケーションを作るためには、Android マニフェストの中に、ブックマークの変更を許可する記述を書いておく必要があります。

ブックマークの取得を許可する記述というのは、

```
<uses-permission
  android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
```

という要素で、ブックマークの変更を許可する記述というのは、

```
<uses-permission
    android:name="com.android.browser.permission.WRITE_HISTORY_BOOKMARKS"/>
```

という要素です。これらの要素は、Android マニフェストのルート要素の子供として書きます。

6.6.3 ブックマークの URI と列の名前

ブックマークを取得するためには、ブックマークの URI と列の名前が必要になります。

ブックマークの URI と列の名前は、次のように定数として定義されています。

ブックマークの URI	`Browser.BOOKMARKS_URI`
タイトルの列の名前	`Browser.BookmarkColumns.TITLE`
URL の列の名前	`Browser.BookmarkColumns.URL`

6.6.4 ブックマークを取得する方法

ブラウザーのブックマークを取得したいときに、まずしなければならないことは、「コンテントリゾルバー」(content resolver) と呼ばれるオブジェクトを取得することです。コンテントリゾ

ルバーは、アクティビティーやサービスが持っている、

```
ContentResolver getContentResolver()
```

というメソッドを呼び出すことによって取得することができます。
次にしなければならないことは、コンテントリゾルバーに対して検索を実行して、その結果から構成されるカーソルを取得することです。検索は、コンテントリゾルバーが持っている、

```
Cursor query(Uri uri, String[] projection, String selection,
    String[] selectionArgs, String sortOrder)
```

というメソッドを呼び出すことによって実行することができます。たとえば、

```
Cursor c = getContentResolver().query(
    Browser.BOOKMARKS_URI,
    new String[] {
        Browser.BookmarkColumns.URL,
        Browser.BookmarkColumns.TITLE
    },
    null, null, null);
```

というように getContentResolver と query を呼び出すことによって、すべてのタイトルと URL から構成されるカーソルを取得することができます。

6.6.5　ブックマークを取得するアプリケーションの例

それでは、ブラウザーのブックマークを取得するアプリケーションを作ってみましょう。
まず最初に、次のようなプロジェクトを作成してください。

Project name	getbm
Application name	ブックマークの取得
Package name	org.example.getbm
Create Activity	GetBookmarkActivity

次に、main.xml を次のように書き換えてください。

レイアウト XML の例　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"
    >
<ListView android:id="@+id/listview"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、GetBookmarkActivity.java を次のように書き換えてください。

プログラムの例　GetBookmarkActivity.java

```
package org.example.getbm;

import android.app.Activity;
import android.os.Bundle;
import android.provider.Browser;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.database.Cursor;

public class GetBookmarkActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        getBookmark();
```

```
    }

    private void getBookmark() {
        Cursor c = getContentResolver().query(
            Browser.BOOKMARKS_URI,
            new String[] {
                Browser.BookmarkColumns.URL,
                Browser.BookmarkColumns.TITLE
            },
            null, null, null);
        if (c.moveToFirst()) {
            int ititle = c.getColumnIndexOrThrow(
                Browser.BookmarkColumns.TITLE);
            int iurl = c.getColumnIndexOrThrow(
                Browser.BookmarkColumns.URL);
            ArrayAdapter<String> adapter =
                new ArrayAdapter<String>(this,
                    android.R.layout.simple_list_item_1);
            final ListView listview = (ListView) findViewById(R.id.listview);
            do {
                adapter.add("[" + c.getString(ititle) + "]" +
                    c.getString(iurl));
            } while (c.moveToNext());
            listview.setAdapter(adapter);
        }
        c.close();
    }
}
```

最後に、Android マニフェストの `manifest` 要素の子供として、

```
<uses-permission
    android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
```

という要素を書き加えてください。

これで完成です。実行してみてください。そうすると、ブックマークがリストビューによって表示されるはずです。

6.6.6 ブックマークを保存する方法

ブックマークを保存する方法は、ブックマークを取得する方法よりも簡単で、

```
android.provider.Browser
```

というクラスの中で定義されている、

```
void saveBookmark(Context c, String title, String url)
```

という静的メソッドを呼び出すだけです。

6.6.7 ブックマークを保存するアプリケーションの例

それでは、ブックマークを保存するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`savebm`
Application name	ブックマークの保存
Package name	`org.example.savebm`
Create Activity	`SaveBookmarkActivity`

次に、`main.xml` を次のように書き換えてください。

レイアウト XML の例 `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"
```

```
    >
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1"
    >
<TableRow>
    <TextView android:text="タイトル"/>
    <EditText android:id="@+id/title"/>
</TableRow>
<TableRow>
    <TextView android:text="URL"/>
    <EditText android:id="@+id/url"/>
</TableRow>
</TableLayout>
<Button android:id="@+id/save"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="保存"
    />
</LinearLayout>
```

次に、SaveBookmarkActivity.java を次のように書き換えてください。

プログラムの例　SaveBookmarkActivity.java

```
package org.example.savebm;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.provider.Browser;
import android.widget.Button;
import android.widget.EditText;

public class SaveBookmarkActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button save = (Button) findViewById(R.id.save);
        save.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                saveBookmark();
            }
        });
    }

    private void saveBookmark() {
        EditText title = (EditText) findViewById(R.id.title);
        EditText url = (EditText) findViewById(R.id.url);
        Browser.saveBookmark(this,
            title.getText().toString(), url.getText().toString());
    }
}
```

最後に、Android マニフェストの manifest 要素の子供として、

```
<uses-permission
        android:name="com.android.browser.permission.WRITE_HISTORY_BOOKMARKS"/>
```

という要素を書き加えてください。

これで完成です。実行すると、二つのエディットテキストとひとつのボタンが表示されますので、上のエディットテキストにタイトル、下のエディットテキストに URL を入力して、ボタンをクリックしてください。そうすると、確認のためのダイアログが表示されますので、問題がな

ければ [OK] のボタンをクリックしてください。そうすると、入力したタイトルと URL がブックマークに保存されるはずですので、保存されたかどうかを、先ほどのアプリケーションかまたはブラウザーを使って確認してください。

第 7 章　ネットワーク

7.1　HTTP

7.1.1　インターネットへの接続の許可

Android アプリケーションは、パソコン上の OS のアプリケーションと同じように、インターネット上のサービスを自由に利用することができます。

ただし、インターネットを利用する Android アプリケーションを作るためには、Android マニフェストの中に、インターネットへの接続を許可するという記述を書いておく必要があります。

インターネットへの接続を許可する記述というのは、

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

という要素です。これを、Android マニフェストのルート要素の子供として書いておくと、インターネットへの接続が可能になります。

7.1.2　URL クラスのオブジェクトの生成

この節では、HTTP を使ってインターネットからデータを取得する方法について説明したいと思います。

プロトコルとして HTTP を使って通信をするアプリケーションは、直接的にソケットを使うことによって作ることも可能ですが、Java のクラスライブラリーが持っている、HTTP による接続のためのクラスを使えば、さらに簡単に作ることができます。

HTTP による接続のためのクラスを使うためには、まず、アクセスするリソースの URL をあらわす、

```
java.net.URL
```

というクラスのオブジェクトを生成する必要があります。このクラスのオブジェクトは、文字列の URL をコンストラクタに渡すことによって生成することができます。

7.1.3　HttpURLConnection クラスのオブジェクトの取得

HTTP による接続のためのクラスというのは、

```
java.net.HttpURLConnection
```

というクラスのことです。このクラスのオブジェクトは、URL クラスのオブジェクトが持っている、

```
URLConnection openConnection() throws IOException
```

というメソッドを呼び出すことによって取得することができます。ただし、このメソッドの戻り値は、

```
java.net.URLConnection
```

という型になっていますので、その戻り値を HttpURLConnection にキャストする必要があります。

7.1.4　HTTP によるアクセスの設定

HttpURLConnection クラスのオブジェクトを使って HTTP でリソースにアクセスするためには、あらかじめ、

```
void setRequestMethod(String method) throws ProtocolException
```

というメソッドを使って、リクエストのメソッドをそのオブジェクトに設定する必要があります。このメソッドに渡す引数は、"GET" や "POST" のような、メソッドをあらわす文字列です。

また、

```
void setConnectTimeout(int timeout)
```

というメソッドを呼び出すことによって、接続を確立するときのタイムアウトを設定することができます。同じように、

```
void setReadTimeout(int timeout)
```

というメソッドを呼び出すことによって、データを読み込むときのタイムアウトを設定することもできます。これらのメソッドに渡す引数は、ミリ秒を単位とするタイムアウトまでの時間です。

7.1.5 接続と切断

HTTP による接続の確立は、

```
void connect() throws IOException
```

というメソッドを呼び出すことによって実行されます。そして切断は、

```
void disconnect()
```

というメソッドを呼び出すことによって実行されます。

7.1.6 入力ストリームの取得

接続先からデータを受信するためには、接続をあらわすオブジェクトから入力ストリームを取得する必要があります。入力ストリームは、

```
InputStream getInputStream() throws IOException
```

というメソッドを呼び出すことによって取得することができます。

7.1.7 HTTP でデータを取得するアプリケーションの例

それでは、HTTP を使ってインターネットからデータを取得するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`http`
Application name	HTTP
Package name	`org.example.http`
Create Activity	`HttpActivity`

次に、`main.xml` を次のように書き換えてください。

レイアウト XML の例 `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/url"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:singleLine="true"
    />
<Button android:id="@+id/get"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="データを取得"
    />
<TextView android:id="@+id/result"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、`Http` という名前の新しいクラスを定義します。まず、

```
src/org.example.http
```

を右クリックすることによって表示されるメニューの中にある、

[New]→[Class]

という項目を選択してください。そうすると、New Java Class というダイアログが表示されますので、その中の Name の項目に Http と入力して、[Finish] をクリックしてください。そうすると、Http.java という名前の新しいファイルができますので、そのファイルの内容を次のように書き換えてください。

プログラムの例 Http.java

```
package org.example.http;

import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.net.URL;
import java.net.HttpURLConnection;

public class Http {
    public static String getHttp(String urlString) {
        StringBuffer sb = new StringBuffer();
        try {
            URL url = new URL(urlString);
            HttpURLConnection huc =
                (HttpURLConnection) url.openConnection();
            huc.setRequestMethod("GET");
            huc.setConnectTimeout(10000);
            huc.setReadTimeout(50000);
            huc.connect();
            BufferedReader br =
                new BufferedReader(
                    new InputStreamReader(huc.getInputStream()));
            String line;
            while ((line = br.readLine()) != null)
                sb.append(line);
            br.close();
            huc.disconnect();
        } catch (Exception e) {
            sb.append(e.toString());
        }
        return sb.toString();
    }
}
```

次に、HttpActivity.java を次のように書き換えてください。

プログラムの例 HttpActivity.java

```
package org.example.http;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.widget.TextView;

public class HttpActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button get = (Button) findViewById(R.id.get);
        get.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showHttp();
            }
```

```
        });
    }

    private void showHttp() {
        EditText url = (EditText) findViewById(R.id.url);
        TextView result = (TextView) findViewById(R.id.result);
        result.setText(Http.getHttp(url.getText().toString()));
    }
}
```

最後に、Android マニフェストの manifest 要素の子供として、

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

という要素を書き加えてください。

これで完成です。実行してみてください。そして、http で始まる URL をエディットテキストに入力して、ボタンをクリックしてください。そうすると、URL で参照されるリソースの最初の部分が表示されるはずです。

7.2 SMTP

7.2.1 ソケットの生成とクローズ

この節では、ソケットを使って SMTP のクライアントを作る方法について説明したいと思います。

Java では、ソケットは、

```
java.net.Socket
```

というクラスのインスタンスです。このクラスのインスタンスを生成する場合は、通常、

```
Socket(String host, int port)
    throws UnknownHostException, IOException
```

というコンストラクタを使います。このコンストラクタに渡す引数は、1 個目が接続先のホスト名で、2 個目がポート番号です。

ソケットによる通信が終了したときは、そのソケットをクローズする必要があります。ソケットは、それが持っている、

```
void close() throws IOException
```

というメソッドを呼び出すことによって、クローズすることができます。

7.2.2 ストリームの取得

ソケットを使ってデータの送受信をするためには、ソケットから入力ストリームと出力ストリームを取得する必要があります。入力ストリームと出力ストリームのそれぞれは、

- `InputStream getInputStream() throws IOException`
- `OutputStream getOutputStream() throws IOException`

というメソッドを呼び出すことによって取得することができます。

7.2.3 SMTP でメッセージを送信するアプリケーションの例

それでは、SMTP を使ってメッセージを送信するアプリケーションを作ってみましょう。

まず最初に、次のようなプロジェクトを作成してください。

Project name	`smtp`
Application name	SMTP
Package name	`org.example.smtp`
Create Activity	`SmtpActivity`

次に、main.xml を次のように書き換えてください。

レイアウト XML の例 `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"
    >
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1"
    >
<TableRow>
    <TextView android:text="サーバー"/>
    <EditText android:id="@+id/server"/>
</TableRow>
<TableRow>
    <TextView android:text="送信者"/>
    <EditText android:id="@+id/from"/>
</TableRow>
<TableRow>
    <TextView android:text="受信者"/>
    <EditText android:id="@+id/to"/>
</TableRow>
<TableRow>
    <TextView android:text="件名"/>
    <EditText android:id="@+id/subject"/>
</TableRow>
<TableRow>
    <TextView android:text="本文"/>
    <EditText android:id="@+id/body"/>
</TableRow>
</TableLayout>
<Button android:id="@+id/send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="送信"
    />
<TextView android:id="@+id/log"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

次に、Smtpという名前の新しいクラスを作って、Smtp.javaの内容を次のように書き換えてください。

プログラムの例 Smtp.java

```
package org.example.smtp;

import java.io.*;
import java.net.*;

class Smtp {
    static StringBuffer sb;
    static Socket socket;
    static BufferedReader reader;
    static BufferedWriter writer;

    static void open(String host)
            throws IOException {
        socket = new Socket(host, 25);
        try {
            reader = new BufferedReader(
                new InputStreamReader(socket.getInputStream()));
            writer = new BufferedWriter(
                new OutputStreamWriter(
                    socket.getOutputStream(), "ISO-2022-JP"));
```

```
        } catch (IOException e) {
            socket.close();
            throw e;
        }
    }

    static void close() throws IOException {
        reader.close();
        writer.close();
        socket.close();
    }

    static void send(String s) throws IOException {
        writer.write(s + "\r\n");
        writer.flush();
    }

    static String receive() throws IOException {
        String line = reader.readLine();
        sb.append(line);
        sb.append(System.getProperty("line.separator"));
        return line;
    }

    static String getLocalHostName()
            throws UnknownHostException {
        String name = "localhost";
        name = InetAddress.getLocalHost().getHostName();
        return name;
    }

    static void sendReceive(String s)
            throws IOException {
        send(s);
        String reply = receive();
        if (reply.startsWith("4") || reply.startsWith("5"))
            throw new IOException("error replied: " + reply);
    }

    public static String sendSmtp(String server, String from,
            String to, String subject, String body) {
        sb = new StringBuffer();
        try {
            open(server);
            sendReceive("helo " + getLocalHostName());
            sendReceive("mail from:<" + from + ">");
            sendReceive("rcpt to:<" + to + ">");
            sendReceive("data");
            send("To: " + to);
            send("From: " + from);
            send("Subject: " + subject);
            send("MIME-Version: 1.0");
            send("Content-Type: text/plain; charset=ISO-2022-JP");
            send("");
            send(body);
            sendReceive(".");
            sendReceive("quit");
            close();
        } catch (IOException e) {
            sb.append(e.toString());
        }
        return sb.toString();
    }
}
```

次に、SmtpActivity.javaを次のように書き換えてください。

プログラムの例 SmtpActivity.java

```
package org.example.smtp;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.widget.TextView;

public class SmtpActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button send = (Button) findViewById(R.id.send);
        send.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                sendSmtp();
            }
        });
    }

    private void sendSmtp() {
        EditText server = (EditText) findViewById(R.id.server);
        EditText from = (EditText) findViewById(R.id.from);
        EditText to = (EditText) findViewById(R.id.to);
        EditText subject = (EditText) findViewById(R.id.subject);
        EditText body = (EditText) findViewById(R.id.body);
        TextView log = (TextView) findViewById(R.id.log);
        log.setText(Smtp.sendSmtp(
            server.getText().toString(),
            from.getText().toString(),
            to.getText().toString(),
            subject.getText().toString(),
            body.getText().toString()
        ));
    }
}
```

最後に、Android マニフェストの manifest 要素の子供として、

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

という要素を書き加えてください。

これで完成です。実行してみてください。そして、SMTP サーバーが動作しているホストのホスト名、送信者のメールアドレス、受信者のメールアドレス、件名、本文を入力して、ボタンをクリックしてください。そうすると、SMTP サーバーに対してメッセージが送信されるはずです。もしも、

```
java.net.UnknownHostException
```

という例外が発生する場合は、ホスト名の代わりに IP アドレスを入力して試してみてください。

参考文献

[Ableson,2009] Frank Ableson, Charlie Collins and Robi Sen, *Unlocking Android: A Developer's Guide*, Manning, 2009, ISBN 978-1-933988-67-2. 邦訳（安生真、土肥拓生、谷沢智史）、『コードからわかる Android プログラミングのしくみ：開発で困ったときの解決アプローチ』、日経 BP 社、2010、ISBN 978-4-8222-8409-1。

[Burnette,2008] Ed Burnette, *Hello, Android: Introducing Google's Mobile Development Platform*, Pragmatic Bookshelf, 2008, ISBN 978-1-934356-17-3. 邦訳（長尾高弘）、『初めての

Android』、オライリー・ジャパン、2009、ISBN 978-4-87311-409-5。

[DiMarzio,2008] Jerome DiMarzio, *Android: A Programmer's Guide*, McGraw-Hill, 2008, ISBN 978-0-07-159988-7. 邦訳（土肥拓生）、『初めての Google Android プログラミング：サンプルで学ぶ必須作法と基本手順』、日経 BP 社、2009、ISBN 978-4-8222-8371-1。

[Meier,2009] Reto Meier, *Professional Android Application Development*, Wiley Publishing, 2009, ISBN 978-0-470-34471-2.

[Murphy,2009] Mark L. Murphy, *The Busy Coder's Guide to Android Development*, CommonsWare, 2009, ISBN 978-0-9816780-0-9.

[Rogers,2009] Rick Rogers, John Lombardo, Zigurd Mednieks and Blake Meike, *Android Application Development: Programming with the Google SDK*, O'Reilly, 2009, ISBN 978-0-596-52147-9.

[木南,2008] 木南英夫、「Google Android アプリ開発入門」、『組込みプレス』、Vol. 12、技術評論社、2008、ISBN 978-4-7741-3571-7。

[木南,2009] 木南英夫、『Google Android アプリケーション開発入門：画面作成からデバイス制御まで——基本機能の全容』、日経 BP 社、2009、ISBN 978-4-8222-8390-2。

[江川,2008] 江川崇、竹端進、山田暁通、麻野耕一、山岡敏夫、藤井大助、窪田康大、藤田泰介、佐野徹郎、『Google Android 完全解説』、アスキー、2008、ISBN 978-4-7561-5130-8。

[江川,2009] 江川崇、藤井大助、麻野耕一、藤田泰介、山田暁通、山岡敏夫、佐野徹郎、竹端進、『Google Android プログラミング入門』、アスキー、2009、ISBN 978-4-04-867956-5。

[柴田,2009] 柴田文彦、藤枝崇史、伊原頌二、『入門 Google Android プログラミング』、インプレスジャパン、2009、ISBN 978-4-8443-2771-4。

[嶋,2008] 嶋是一、『Google Android 入門：携帯電話開発の新技術』、技術評論社、2008、ISBN 978-4-7741-3462-8。

[嶋,2009] 嶋是一、中村秀樹、江川崇、杉本礼彦、安生真、内嶋教人、今村謙之、山崎淳、木南英夫、安藤恐竜、「Android アプリ開発「匠」への道」、『Software Design』、2009 年 3 月号、技術評論社、2009。

[西沢,2009] 西沢直木、『SQLite 入門：すぐに使える軽快・軽量データベースエンジン・第 2 版』、翔泳社、2009、ISBN 978-4-7981-1944-1。

[布留川,2009] 布留川英一、『Android 1.5 プログラミングバイブル』、ソシム、2009、ISBN 978-4-88337-663-6。

[柳井,2009] 柳井政和、『Google Android アプリ開発ガイド』、秀和システム、2009、ISBN 978-4-7980-2300-7。

[若林,2009] 若林登、『はじめての Android プログラミング』、工学社、2009、ISBN 978-4-7775-1435-9。

索引