Android 6.0 Marshmallow アプリ開発入門 Vol.2

IT研究所

第7章　ダイアログ

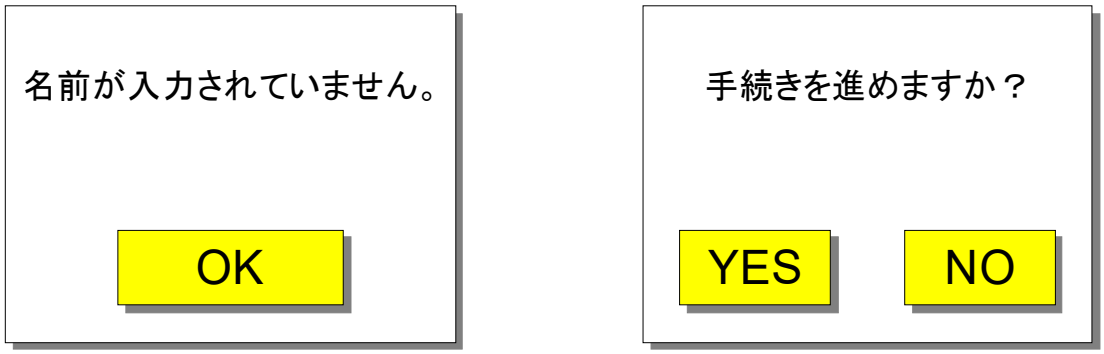
第8章　メニュー

第9章　ライフサイクル

第7章
ダイアログ

7.1 ダイアログとは

- ユーザに情報を提示し
  必要に応じてユーザに応答してもらうもの。



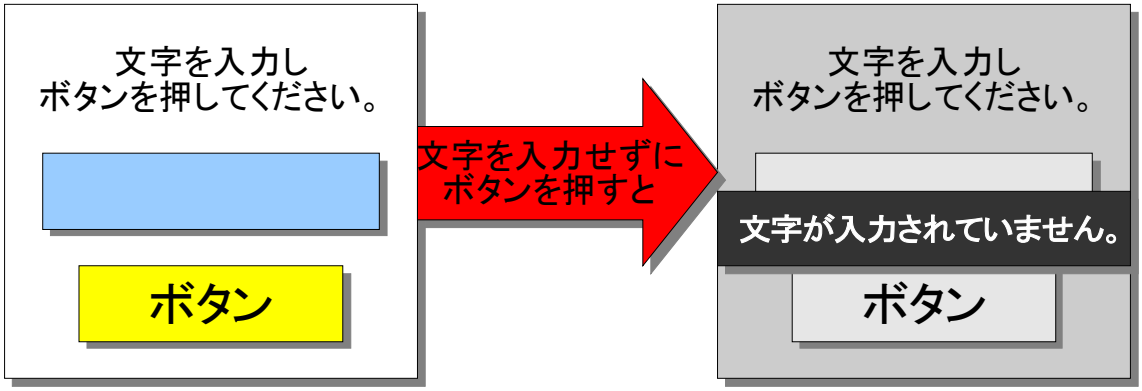
ダイアログとは「対話」という意味であり、ユーザに情報を提示し、必要に応じてユーザに応答してもらうものです。

例えば、必要に応じてユーザに入力を求めたり、何らかの通知を行うことに使用されます。

7.2 AlertDialogの利用方法1

アプリケーション名：AlertDialogApp01

動作概要



◆作成手順

①新規アプリケーション「AlertDialogApp01」を作成する。

②「activity_main.xml」で「TextView（Plain TextView）」と「EditText」と「Button」を配置する。

③「MainActivity.java」でボタンをクリックした時の処理を設定する。

●Buider クラス

Builder クラスは、AlertDialog クラス内で次のように定義されている。

```
public class AlertDialog {
        public static class Builder {
                ・・・・・・・・・・・
        }
}
```

●Bulder クラスのコンストラクタ

【書式】
public Builder (Context context)

【引数】
AlertDialogを表示するContext のオブジェクトを指定する。
（ActivityはContextのサブクラスであるため、Activityを設定することができる。）

【機能】
AlertDialogを生成する。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.alertdialogapp01;

import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        EditText editText = (EditText)findViewById(R.id.editText);

        String str = editText.getText().toString();

        if (!str.equals("")) {
            Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
            toast.show();
        } else {
            AlertDialog.Builder builder = new AlertDialog.Builder(this);

            builder.setTitle("警告");
            builder.setMessage("文字が入力されていません。");
            builder.show();
        }
    }
}
```

※一部抜粋

★setTitle メソッド（AlertDialog.Builder クラスに定義）

【書式】
public void setTitle(CharSequence str)

【機能】
ダイアログにタイトルを設定する。

★setMessage メソッド（AlertDialog.Builder クラスに定義）

【書式】
public void setMessage(CharSequence str)

【機能】
ダイアログに表示するメッセージを設定する。

★show メソッド（AlertDialog.Builder クラスに定義）

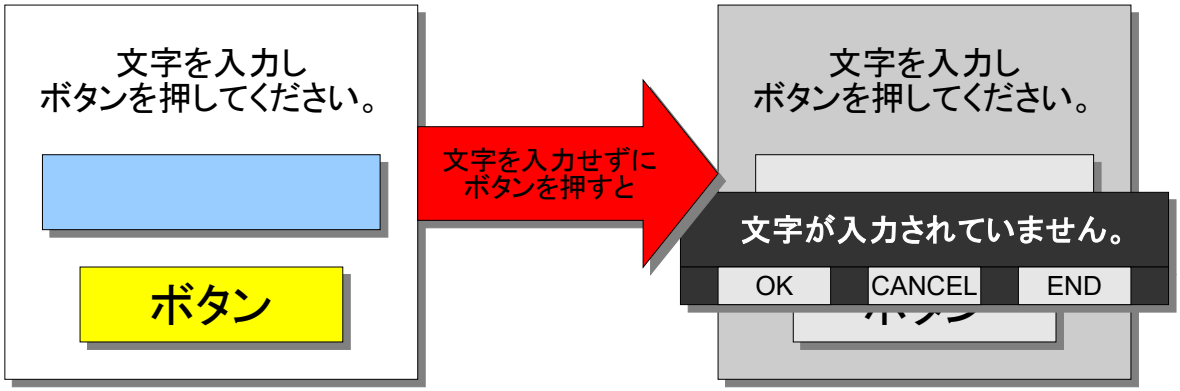
【書式】
public void show()

【機能】
ダイアログを画面に表示する。

7.3 AlertDialogの利用方法2

アプリケーション名：AlertDialogApp01

動作概要



◆作成手順

①「AlertDialogApp01」を変更する。

●DialogInterface インターフェース

DialogInterface インターフェースは、次のように定義されている。

```
public interface DialogInterface {
        public static final int BUTTON_POSITIVE = -1;
        public static final int BUTTON_NEGATIVE = -2;
        public static final int BUTTON_NEUTRAL = -3;

        public static interface OnClickListener {
                ・・・・・・
        }
}
```

※BUTTON_POSITIVEはポジティブ・ボタン用の定数。
※BUTTON_NEGATIVEはネガティブ・ボタン用の定数。
※BUTTON_NEUTRALはニュートラル・ボタン用の定数。

●DialogInterface.OnClickListener インターフェース

OnClickListener インターフェースは DialogInterface インターフェース内で、次のように定義されている。

```
public static interface OnClickListener {
        public abstract void onClick(DialogInterface dialog, int which);
}
```

※onClickメソッドの第1引数には、クリックが発生したDialogInterfaceを実装したクラスのオブジェクトが渡される。
※onClickメソッドの第2引数には、クリックが発生したボタンの種類が渡される。

●イベントリスナ登録

ダイアログ内で使用するボタンのイベントリスナ登録を行うには、下記のメソッドを使用して行う。

★setPositiviButton メソッド

【書式】
public void setPositiveButton(文字列, DialogInterface.OnClickListener インターフェースを実装したオブジェクト)

【機能】
ポジティブ・ボタンをDialogInterface.OnClickListener にイベントリスナ登録を行う。

★setNegativeButton メソッド

【書式】
public void setNegativeButton(文字列, DialogInterface.OnClickListener インターフェースを実装したオブジェクト)

【機能】
ネガティブ・ボタンをDialogInterface.OnClickListener にイベントリスナ登録を行う。

★setNeutralButton メソッド

【書式】
public void setNeutralButton(文字列, DialogInterface.OnClickListener インターフェースを実装したオブジェクト)

【機能】
ニュートラル・ボタンをDialogInterface.OnClickListener にイベントリスナ登録を行う。

●プログラム例

ファイル名 : MainActivity.java

```
package com.example.alertdialogapp01;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener, DialogInterface.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }
```

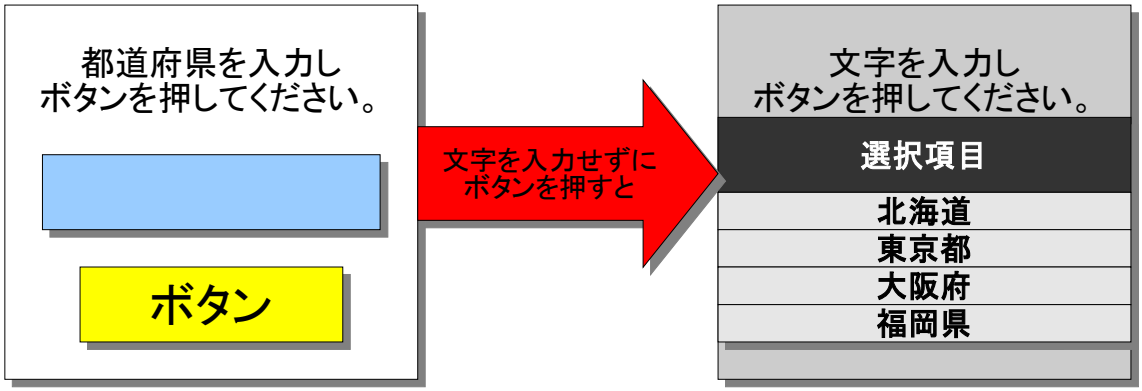
```
    @Override
    public void onClick(View view) {
        EditText editText = (EditText)findViewById(R.id.editText);

        String str = editText.getText().toString();

        if (!str.equals("")) {
            Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
            toast.show();
        } else {
            AlertDialog.Builder builder = new AlertDialog.Builder(this);

            builder.setTitle("警告");
            builder.setMessage("文字が入力されていません。");

            builder.setPositiveButton("OK", this);
            builder.setNegativeButton("CANCEL", this);
            builder.setNeutralButton("END", this);

            builder.show();
        }
    }

    @Override
    public void onClick(DialogInterface dialog, int which) {
        String str = "";

        switch (which) {
            case DialogInterface.BUTTON_POSITIVE : str = "OK"; break;
            case DialogInterface.BUTTON_NEGATIVE : str = "CANCEL"; break;
            case DialogInterface.BUTTON_NEUTRAL : str = "END"; break;
        }

        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

※一部抜粋

7.4 AlertDialogの利用方法3

アプリケーション名：AlertDialogApp02

動作概要



◆作成手順

①新規アプリケーション「AlertDialogApp02」を作成する。

②「activity_main.xml」で「TextView」と「EditText」と「Button」を配置する。

③「strings.xml」で string-array の項目を設定する。

ファイル名：strings.xml

```
<resources>

    <string-array name="prefecture_list">
        <item >北海道</item>
        <item >東京都</item>
        <item >大阪府</item>
        <item >福岡県</item>
    </string-array>

</resources>
```

④「MainActivity.java」でボタンをクリックした時の処理を設定する。

●イベントリスナ登録

ダイアログ内で使用するリスト選択時のイベントリスナ登録を行うには、setItems メソッドを使用して行う。

【書式】
public void setItems(CharSequence[] array, DialogInterface.OnClickListener　インターフェースを実装したオブジェクト)

【機能】
ダイアログにリストを設定し、DialogInterface.OnClickListener にイベントリスナ登録を行う。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.alertdialogapp02;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.res.Resources;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener, DialogInterface.OnClickListener {
    private String[] prefectures;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);

        Resources resource = getResources();
        prefectures = resource.getStringArray(R.array.prefecture_list);
    }

    @Override
    public void onClick(View view) {
        EditText editText = (EditText)findViewById(R.id.editText);

        String str = editText.getText().toString();

        if (!str.equals("")) {
            Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
            toast.show();
        } else {
            AlertDialog.Builder builder = new AlertDialog.Builder(this);

            builder.setTitle("選択項目");
            builder.setItems(prefectures, this);

            builder.show();
        }
    }

    @Override
    public void onClick(DialogInterface dialog, int which) {
        EditText editText = (EditText)findViewById(R.id.editText);
        editText.setText(prefectures[which]);
    }
}
```

※一部抜粋

★getResources メソッド（Activity クラスで定義）

【書式】
public Resources getResources()

【戻り値】
android.content.res.Resources クラスのオブジェクト

【機能】
リソースを取得する。

★getStringArray メソッド（Resources クラスに定義）

【書式】
public Strings[] getStringArray(int id)

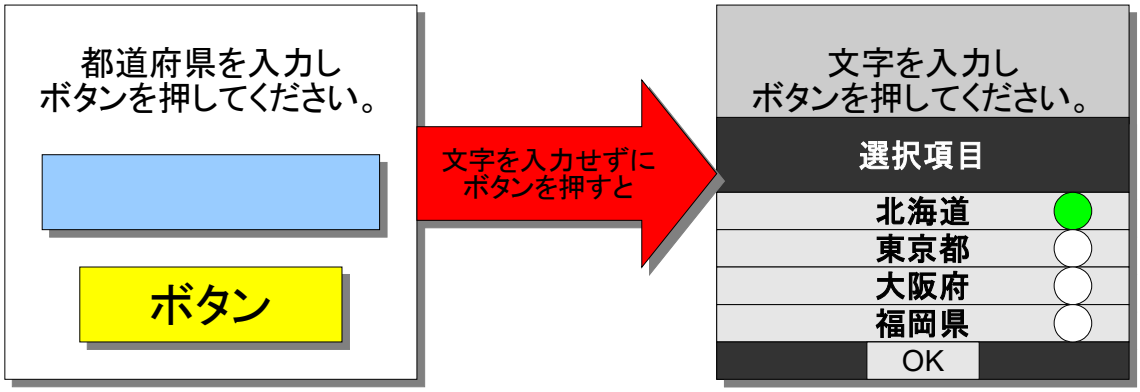
【戻り値】
文字列配列

【機能】
指定したリソースIDから文字列配列を取得する。

7.5 AlertDialogの利用方法4

アプリケーション名：AlertDialogApp02

動作概要



◆作成手順

①「AlertDialogApp02」を変更する。

●イベントリスナ登録

ダイアログ内で使用するリストのラジオボタン選択時のイベントリスナ登録を行うには setSingleChoiceItems メソッドを使用して行う。

【書式】
public void setSingleChoiceItems(CharSequence[] array, int checkedItem,
DialogInterface.OnClickListener インターフェースを実装したオブジェクト)

※第2引数のcheckedItemには、デフォルトで表示される項目の番号を設定する。

【機能】
ダイアログにリストのラジオボタンを設定し、DialogInterface.OnClickListener にイベントリスナ登録を行う。

●プログラム例

ファイル名：MainActivity.java

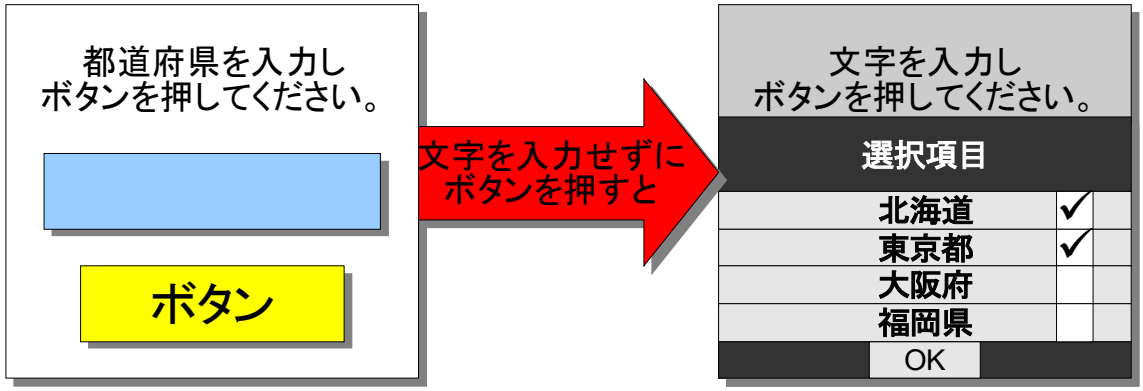
```
package com.example.alertdialogapp02;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.res.Resources;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener, DialogInterface.OnClickListener {
    private int selectNo;
    private String[] prefectures;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);

        Resources resource = getResources();
        prefectures = resource.getStringArray(R.array.prefecture_list);
    }

    public void onClick(View view) {
        EditText editText = (EditText)findViewById(R.id.editText);

        String str = editText.getText().toString();

        if (!str.equals("")) {
            Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
            toast.show();
        } else {
            AlertDialog.Builder builder = new AlertDialog.Builder(this);

            builder.setTitle("選択項目");
            builder.setSingleChoiceItems(prefectures, selectNo, this);
            builder.setPositiveButton("OK", this);

            builder.show();
        }
    }

    @Override
    public void onClick(DialogInterface dialog, int which) {
        if (which == DialogInterface.BUTTON_POSITIVE) {
            EditText editText = (EditText) findViewById(R.id.editText);
            editText.setText(prefectures[selectNo]);
        } else {
            selectNo = which;
        }
    }
}
```

※一部抜粋

7.6 AlertDialogの利用方法5

アプリケーション名：AlertDialogApp02

動作概要



◆作成手順

①「AlertDialogApp02」を変更する。

●DialogInterface.OnMultiChoiceClickListener インターフェース

OnMultiChoiceClickListener インターフェースは DialogInterface インターフェース内で、次のように定義されている。

```
public static interface OnClickListener {
        public abstract void onClick(DialogInterface dialog, int which, boolean isChecked);
}
```

※onClickメソッドの第1引数には、クリックが発生したDialogInterfaceを実装したクラスのオブジェクトが渡される。
※onClickメソッドの第2引数には、クリックが発生した部品の項目を示す整数値が渡される。
※onClickメソッドの第3引数には、クリックが発生したボタンの選択状態を示す真偽値が渡される。

●イベントリスナ登録

ダイアログ内で使用するリストのチェックボックス選択時のイベントリスナ登録を行うには、setMultiChoiceItems メソッドを使用して行う。

【書式】
public void setMultiChoiceItems(CharSequence[] array, boolean[] checkedItems,
DialogInterface.OnMultiChoiceClickListener インターフェースを実装したオブジェクト)

※第2引数のcheckedItemsには、デフォルトで表示される各項目の選択状態をまとめたboolean型配列を設定する。

【機能】
ダイアログにリストのチェックボックスを設定し、DialogInterface.OnMultiChoiceClickListener にイベントリスナ登録を行う。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.alertdialogapp02;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.res.Resources;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity
    implements OnClickListener, DialogInterface.OnClickListener, DialogInterface.OnMultiChoiceClickListener {
  private String[] prefectures;
  private boolean[] itemChecker = {false, false, false, false};

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button button = (Button)findViewById(R.id.button);
    button.setOnClickListener(this);
    Resources resource = getResources();
    prefectures = resource.getStringArray(R.array.prefecture_list);
  }

  @Override
  public void onClick(View view) {
    EditText editText = (EditText)findViewById(R.id.editText);
    String str = editText.getText().toString();

    if (!str.equals("")) {
      Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
      toast.show();
    } else {
      AlertDialog.Builder builder = new AlertDialog.Builder(this);
      builder.setTitle("選択項目");
      builder.setMultiChoiceItems(prefectures, itemChecker, this);
      builder.setPositiveButton("OK", this);
      builder.show();
    }
  }

  @Override
  public void onClick(DialogInterface dialog, int which) {
    String str = "";
    EditText editText = (EditText) findViewById(R.id.editText);

    for (int i = 0; i < itemChecker.length; i++) {
      if (itemChecker[i]) {
        str += str.equals("") ? prefectures[i] : "   " + prefectures[i];
      }
    }

    editText.setText(str);
  }

  @Override
  public void onClick(DialogInterface dialog, int which, boolean isChecked) {
    itemChecker[which] = isChecked;
  }
}
```

※一部抜粋

7.7 ハンドラとは

- 何らかの処理要求が発生したときに起動されるプログラム。

Android の Activity は描画処理を行うための UI（User Interface）スレッドしかもたないシングルスレッド設計です。負荷が高い処理、待ち時間が発生する処理（ネットワーク通信をはじめとした非同期通信など）は UI スレッドを圧迫し、アプリケーションのレスポンスに影響します。UI スレッドの負荷を下げるには別スレッドを作成し、仕事を分散させるマルチスレッド処理の実装が必要です。これに対応する仕組みがハンドラです。

ハンドラはプログラム中で関数やサブルーチンなどの形で実装され、メモリ上に展開されるが、通常のプログラムの流れには組み込まれず、普段は待機しています。そのハンドラが対応すべき処理要求が発生するとプログラムの流れを中断してハンドラが呼び出され、要求された処理を実行します。対応付けられた事象の種類により「イベントハンドラ」「割り込みハンドラ」などの種類があります。

◆Handler クラス

Handler クラスは android.os パッケージに定義されている。

◆Handler クラスの主なメソッド

★handleMessage メソッド

【書式】
public void handleMessage(Message msg)

【機能】
メッセージを受信すると起動するメソッド。

★sendEmptyMessageDelayed メソッド

【書式】
public final boolean sendEmptyMessageDelayed(int what, long time)
※timeが1000の場合、1秒を設定できる。

【戻り値】
メッセージがメッセージキューに置かれたらtrueを返す。

【機能】
指定した時間ごとにwhatの値だけを持つメッセージを送信するメソッド。

★sendMessage メソッド

【書式】
public boolean sendMessage(Message msg)

【戻り値】
メッセージがメッセージキューに置かれたらtrueを返す。

【機能】
メッセージを送信する。

★sendEmptyMessage メソッド

【書式】
public boolean sendEmptyMessage(int what)

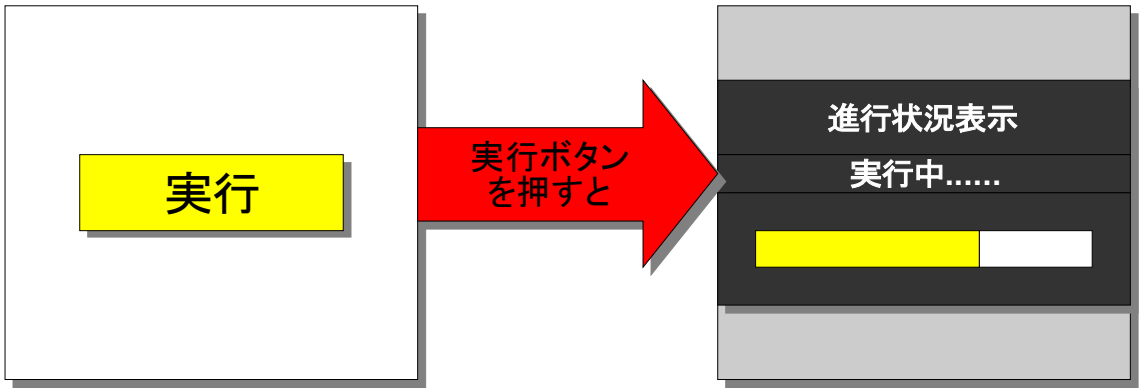
【戻り値】
メッセージがメッセージキューに置かれたらtrueを返す。

【機能】
whatの値だけを持つメッセージを送信する。

7.8 ProgressDialogの利用方法

アプリケーション名：ProgressDialogApp

動作概要



◆作成手順

①新規アプリケーション「ProgressDialogApp」を作成する。

②「activity_main.xml」で「Button」を配置する。

③「MainActivity.java」でボタンをクリックした時の処理を設定する。

●ProgressDialog クラス

ProgressDialog クラスは、android.app パッケージで定義されている。

●ProgressDialog クラスのコンストラクタ

【書式】
public ProgressDialog (Context context)

【引数】
ProgressDialogを表示するContextのオブジェクトを指定する。
（ActivityはContextのサブクラスであるため、Activityを設定することができる。）

【機能】
ProgressDialogを生成する。

●ProgressDialog クラスの主なメソッド

メソッド	説明
setTitle	【書式】 public void setTitle(CharSequence message) 【機能】 タイトルにmessageを設定する。
setMessage	【書式】 public void setMessage(CharSequence message) 【機能】 メッセージにmessageを設定する。
setProgressStyle	【書式】 public void setProgressStyle(int style) ※styleにSTYLE_SPINNERを設定すると、円形のプログレスバーとなる。 ※styleにSTYLE_HORIZONTALを設定すると、横長バーのプログレスバーとなる。 【機能】 表示スタイルを設定する。
setProgress	【書式】 public void setProgress(int value) 【機能】 プログレスバーに値を設定する。
getProgress	【書式】 public int getProgress() 【機能】 プログレスバーの値を取得する。
setMax	【書式】 public void setMax(int value) 【機能】 プログレスバーに最大値を設定する。
getMax	【書式】 public int getMax() 【機能】 プログレスバーの最大値を取得する。
incrementProgressBy	【書式】 public void incrementProgressBy(int diff) 【機能】 プログレスバーの値をdiffの値だけ増やす。
dismiss	【書式】 public void dismiss() 【機能】 プログレスバーを終了する。

●プログラム例

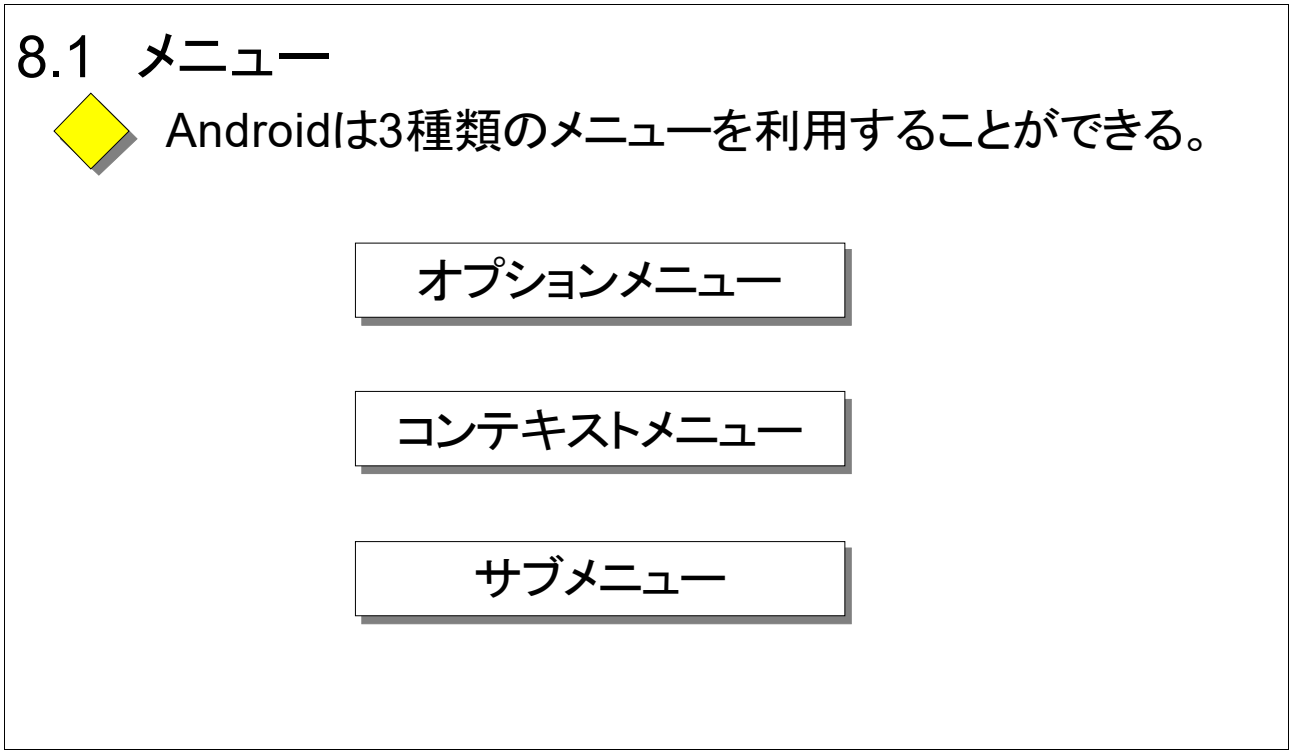
ファイル名 : MainActivity.java

```
package com.example.progressdialogapp;

import android.app.Activity;
import android.app.ProgressDialog;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity implements OnClickListener {
    private ProgressDialog dialog;
    private Operator operator;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);

        operator = new Operator();
    }

    @Override
    public void onClick(View view) {
        dialog = new ProgressDialog(this);
        dialog.setTitle("進行状況表示");
        dialog.setMessage("実行中...");
        dialog.setMax(100);
        dialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);

        dialog.show();
        operator.sendEmptyMessage(0);
    }

    public class Operator extends Handler {
        public void handleMessage(Message msg) {
            super.handleMessage(msg);

            int value = dialog.getProgress();
            int max = dialog.getMax();

            if (value >= max) {
                dialog.dismiss();
                dialog = null;
            } else {
                dialog.incrementProgressBy(5);
                sendEmptyMessageDelayed(0, 100);
            }
        }
    }
}
```

※一部抜粋

メモ

第8章
メニュー



Android では、３種類のメニューを利用することができます。

◆Android のメニュー

①オプションメニュー

Activity に対するメニューアイテムのセットである。これは [MENU] キーを押すことで表示される。

②コンテキストメニュー

パソコンで右クリックで開くような、いわゆるポップアップメニューである。

モバイルアプリケーションの場合は、ビューをタッチしたときなどに出すメニューのことである。

③サブメニュー

「オプションメニュー」か「コンテキストメニュー」の子として作成されるサブメニューである。

サブメニューの下にさらにサブメニューを作ることはできない。

8.2 メニューの利用

Activityクラスには、メニュー関連のメソッドがある。

- onCreateOptionsMenuメソッド
- onPrepareOptionsMenuメソッド
- onMenuOpenedメソッド
- onMenuItemSelectedメソッド
- onOptionsItemSelectedメソッド
- onOptionsItemClosedメソッド
- onCreateContextMenuメソッド
- onContextItemSelectedメソッド
- onContextItemClosedメソッド

Activity クラスには、メニュー関連のメソッドが、いくつか定義されています。これらのメソッドを利用することにより、メニューを扱うプログラムを構築することができます。

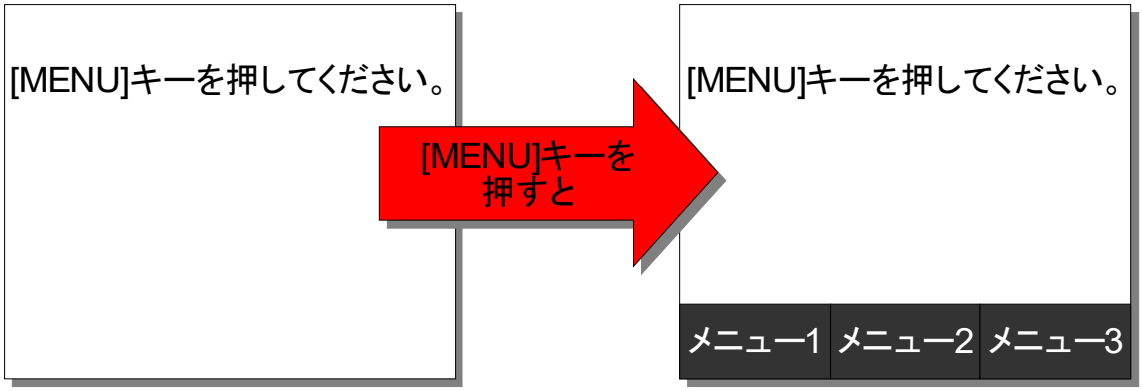
◆Activity クラスに定義されているメニュー関連のメソッド

メソッド	説明
onCreateOptionsMenu	【書式】 public boolean onCreateOptionsMenu(Menu menu) 【機能】 オプションメニューが最初に呼び出される時に1度だけ呼び出される。
onPrepareOptionsMenu	【書式】 public boolean onPrepareOptionsMenu(Menu menu) 【機能】 オプションメニューが表示される度に呼び出される。
onMenuOpened	【書式】 public boolean onPrepareOptionsMenu(Menu menu) 【機能】 メニューを開いた時に呼び出される。
onMenuItemSelected	【書式】 public boolean onMenuItemSelected(int id, MenuItem item) 【機能】 メニュー項目を選択した時に呼び出される。
onOptionsItemSelected	【書式】 public boolean onOptionsItemSelected(MenuItem item) 【機能】 オプションメニュー項目を選択した時に呼び出される。
onOptionsMenuClosed	【書式】 public void onOptionsMenuClosed(Menu menu) 【機能】 オプションメニュー項目を閉じた時に呼び出される。
onCreateContextMenu	【書式】 public void onCreateContextMenu(Menu menu, View view, ContextMenuInfo menuInfo) 【機能】 コンテキストメニューが最初に呼び出される時に1度だけ呼び出される。
onContextItemSelected	【書式】 public boolean onContextItemSelected(MenuItem item) 【機能】 コンテキストメニュー項目を選択した時に呼び出される。
onContextMenuClosed	【書式】 public void onContextMenuClosed(Menu menu) 【機能】 コンテキストメニュー項目を閉じた時に呼び出される。

8.3 オプションメニューの利用方法

アプリケーション名：OptionMenuApp

動作概要



◆作成手順

①新規アプリケーション「OptionMenuApp」を作成する。

②「activity_main.xml」で「TextView（Plain TextView）」を配置する。

③「res」フォルダ内の「menu」フォルダに「option_menu.xml」ファイルを作成し、下記のように設定する。

ファイル名：option_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/item1" android:title="メニュー1"></item>
    <item android:id="@+id/item2" android:title="メニュー2"></item>
    <item android:id="@+id/item3" android:title="メニュー3"></item>
</menu>
```

④「MainActivity.java」で[MENU]キーを押した時のイベント処理を行う。

●onCreateOptionsMenu メソッド内のプログラムを、次のように設定する。

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.option_menu, menu);

    return true;
}
```

※「getMenuInflater().inflate(R.menu.option_menu, menu);」の部分は

「option_menu.xml」に定義した項目からオプションメニューを設定するというプログラムである。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.optionmenuapp;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends Activity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.option_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
            String str = "";

            switch (item.getItemId()) {
                case R.id.item1 : str = "メニュー1が選択されました。"; break;
                case R.id.item2 : str = "メニュー2が選択されました。"; break;
                case R.id.item3 : str = "メニュー3が選択されました。"; break;
            }

            Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
            toast.show();

            return super.onOptionsItemSelected(item);
    }
}
```

★getItemId メソッド（MenuItem クラスに定義）

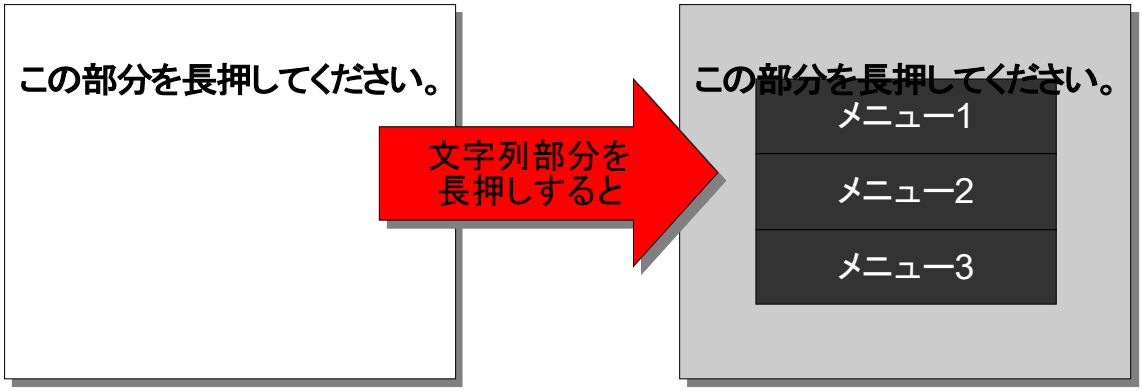
【書式】
public int getItemId()

【機能】
メニュー項目のリソースIDを取得する。

8.4 コンテキストメニューの利用方法

アプリケーション名：ContextMenuApp

動作概要



◆作成手順

①新規アプリケーション「ContextMenuApp」を作成する。

②「activity_main.xml」で「TextView（Plain TextView）」を配置する。

③「res」フォルダ内の「menu」フォルダに「context_menu.xml」ファイルを作成し、下記のように設定する。

ファイル名：context_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/item1" android:title="メニュー1"></item>
    <item android:id="@+id/item2" android:title="メニュー2"></item>
    <item android:id="@+id/item3" android:title="メニュー3"></item>
</menu>
```

④「MainActivity.java」で TextView の部分を長押しした時のイベント処理を行う。

●onCreateContextMenu メソッド内のプログラムを、次のように設定する。

```
public void onCreateContextMenu(Menu menu, View view, ContextMenuInfo menuInfo) {
    getMenuInflater().inflate(R.menu.context_menu, menu);
    return true;
}
```

```
TextView text = (TextView)findViewById(R.id.textView1);
registerForContextMenu(text);
```

●onCreate メソッド内で、TextView をコンテキストメニューに組み込む。

※「registForContextMenu(text);」の部分は

TextView を長押ししたらメニューが呼び出されるように登録するプログラムである。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.contextmenuapp;

import android.app.Activity;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TextView text = (TextView)findViewById(R.id.textView);
        registerForContextMenu(text);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public void onCreateContextMenu(
                        ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        getMenuInflater().inflate(R.menu.context_menu, menu);
    }

    @Override
    public boolean onContextItemSelected(MenuItem item) {
        String str = "";

        switch (item.getItemId()) {
            case R.id.item1 : str = "メニュー1が選択されました。"; break;
            case R.id.item2 : str = "メニュー2が選択されました。"; break;
            case R.id.item3 : str = "メニュー3が選択されました。"; break;
        }

        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        toast.show();

        return super.onContextItemSelected(item);
    }
}
```

★registForContextMenu メソッド（Activity クラスに定義）

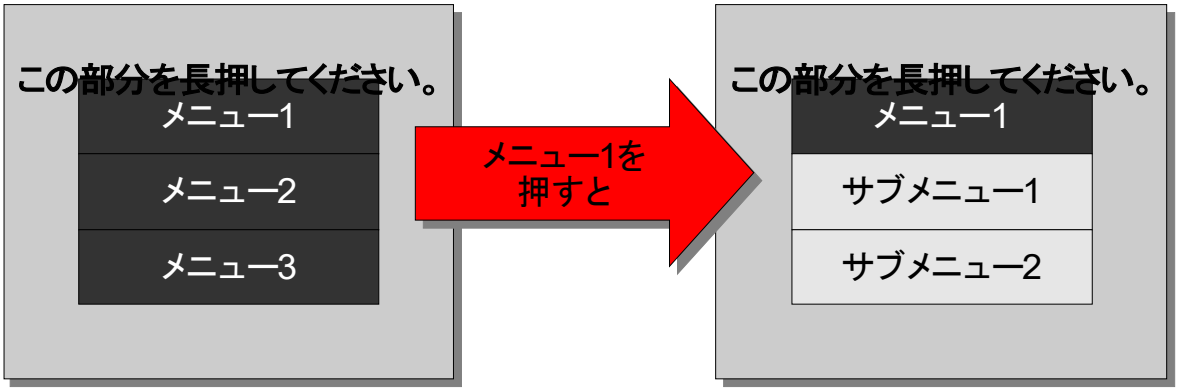
【書式】
public void registForContextMenu(View view)

【機能】
viewに対してコンテキストメニューを登録する。

8.5 サブメニューの利用方法

アプリケーション名：ContextMenuApp

動作概要



◆作成手順

①「ContextMenuApp」を変更する。

●「res」フォルダ内の「menu」フォルダの「context_menu.xml」ファイルを下記のように設定する。

ファイル名：context_menu.xml

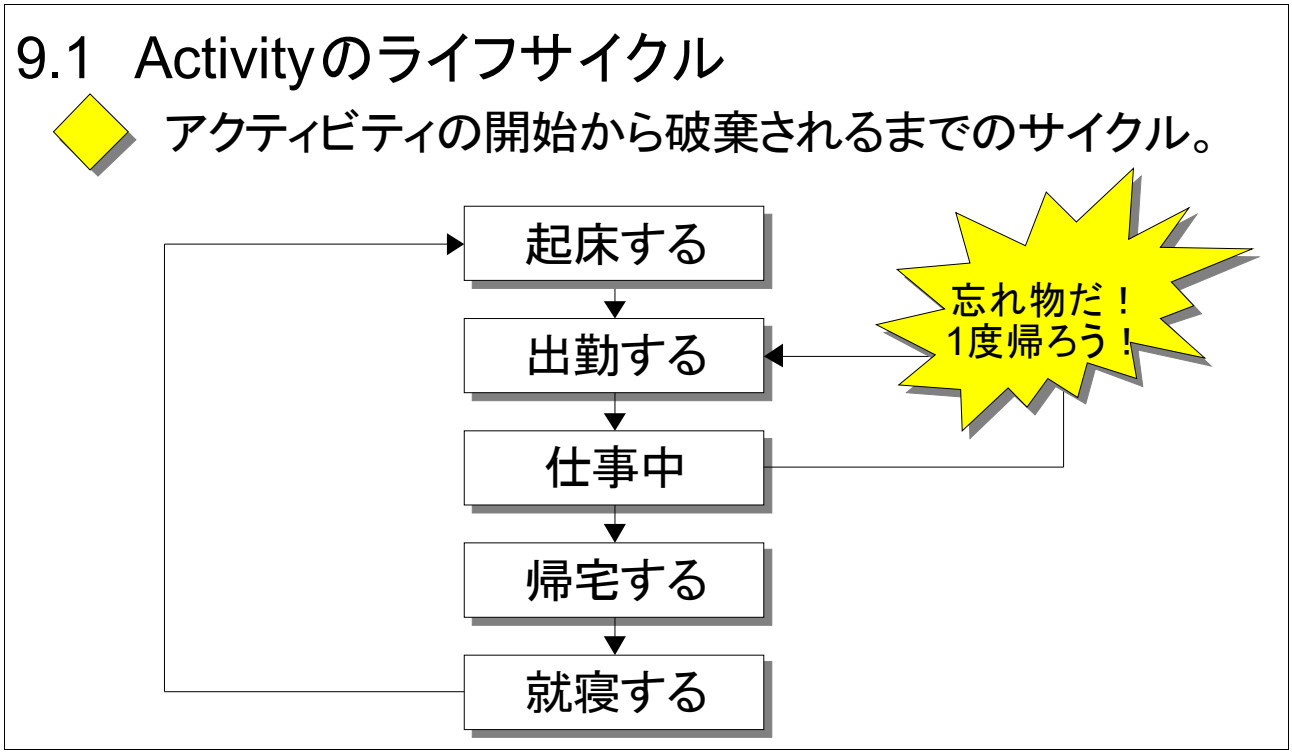
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/item1" android:title="メニュー1">
        <menu>
            <item android:id="@+id/item4" android:title="サブメニュー1"/>
            <item android:id="@+id/item5" android:title="サブメニュー2"/>
        </menu>
    </item>
    <item android:id="@+id/item2" android:title="メニュー2"></item>
    <item android:id="@+id/item3" android:title="メニュー3"></item>
</menu>
```

●「MainActivity.java」の onContextItemSelected メソッド内の switch ブロックを下記のように変更する。

```
switch (item.getItemId()) {
    case R.id.item1 : str = " メニュー1が選択されました。"; break;
    case R.id.item2 : str = " メニュー2が選択されました。"; break;
    case R.id.item3 : str = " メニュー3が選択されました。"; break;
    case R.id.item4 : str = " サブメニュー1が選択されました。"; break;
    case R.id.item5 : str = " サブメニュー2が選択されました。"; break;
}
```

メモ

第9章
ライフサイクル



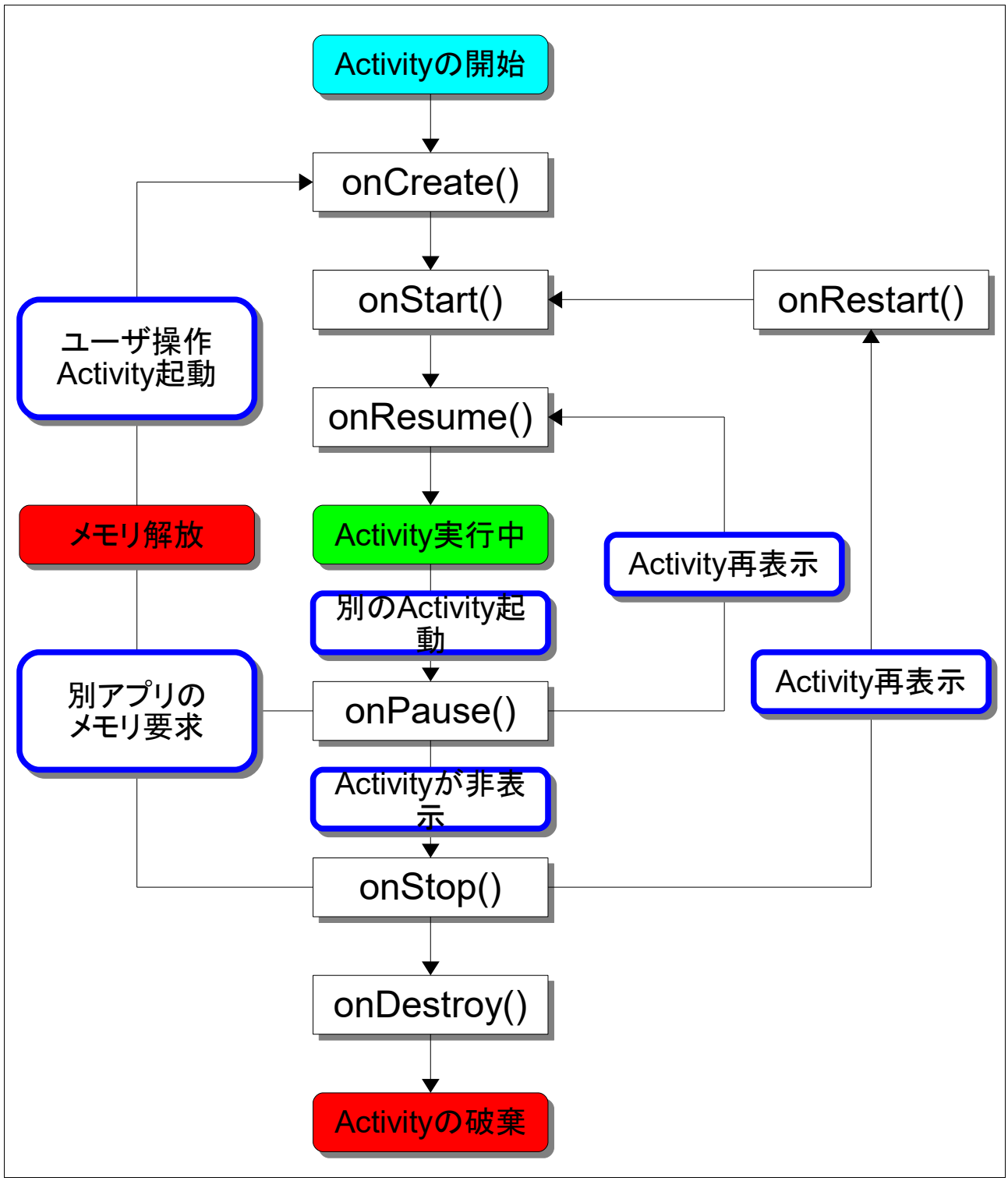
Android の画面からアプリケーションが起動されると最初のアクティビティが開始し画面に表示されます。同じアプリケーションの別のアクティビティが表示されれば最初のアクティビティは隠れたり、別のアプリケーションが起動して違う画面が表示されることもあります。このように 1 つ 1 つのアクティビティは表示されたり隠れたりといったことを繰り返します。このようなアクティビティが開始されて、そして破棄されるまでをアクティビティのライフサイクルと呼んでいます。

◆Activity のライフサイクル

●Activity のライフサイクルメソッド

種類	呼び出されるタイミング
onCreate	該当するアクティビティが作成された時に呼び出される。
onStart	画面が表示される直前に呼び出される。
onRestart	停止状態から実行中になる直前に呼び出される。
onResume	ユーザとのやり取り(入力の受付)を開始する直前に呼び出される。
onPause	画面が非表示、あるいは別のアクティビティが実行状態になった時に呼び出される。
onStop	画面が長時間、非表示になった時に呼び出される。
onDestroy	該当するアクティビティが破棄される直前に呼び出される。

●Activity のライフタイム



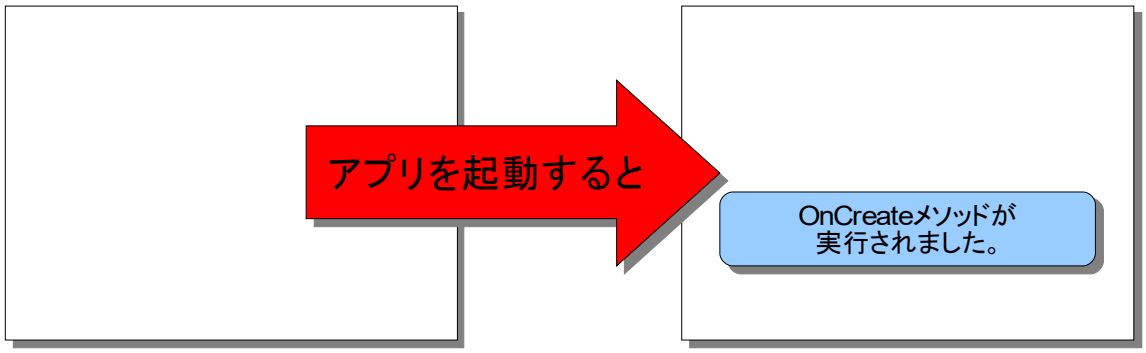
●画面遷移によるライフサイクル

アプリケーション起動	開始→onCreate()→onStart()→onResume()
アプリケーション非表示	実行中→onPause()→onStop()
アプリケーション再起動	開始→onRestart()→onStart()→onResume()
アプリケーション終了	開始→onPause()→onStop()→onDestroy()→終了

9.2 ライフサイクルの利用方法

アプリケーション名：LifeCycleApp

動作概要



◆作成手順

①新規アプリケーション「LifeCycleApp」を作成する。

②「MainActivity.java」でプログラムを設定する。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.lifecycleapp;

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

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Toast.makeText(this, "onCreateメソッドが呼び出されました。", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onStart() {
        super.onStart();
        Toast.makeText(this, "onStartメソッドが呼び出されました。", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onRestart() {
        super.onRestart();
        Toast.makeText(this, "onRestartメソッドが呼び出されました。", Toast.LENGTH_SHORT).show();
    }
```

```
    @Override
    public void onResume() {
        super.onResume();
        Toast.makeText(this, "onResumeメソッドが呼び出されました。", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onPause() {
        super.onPause();
        Toast.makeText(this, "onPauseメソッドが呼び出されました。", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onStop() {
        super.onStop();
        Toast.makeText(this, "onStopメソッドが呼び出されました。", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "onDestroyメソッドが呼び出されました。", Toast.LENGTH_SHORT).show();
    }
}
```

【実行確認方法】

①アプリケーション起動時

実行開始→onCreate→onStart→onResume の順で実行される。

②アプリケーション非表示（「ホーム」ボタンを押した時）

実行中→onPause→onStop の順で実行される。

③アプリケーションの再起動（「アプリケーション」アイコンをクリックした時）

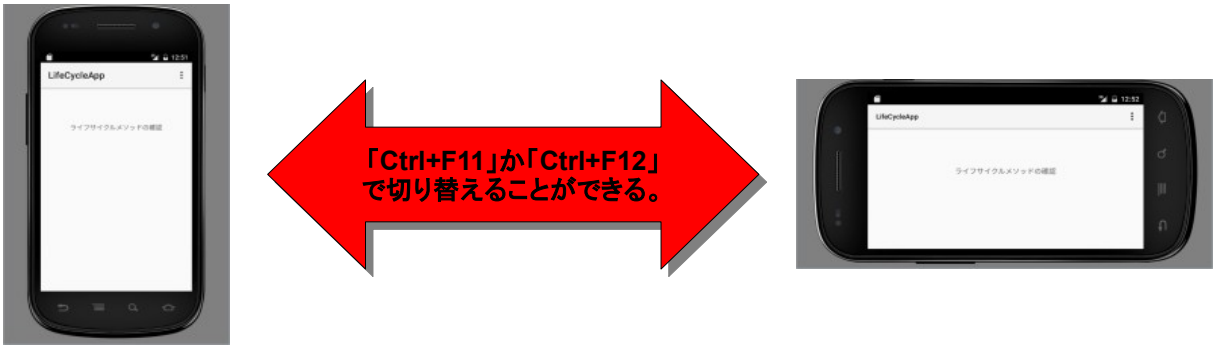
停止中→onRestart→onStart→onResume の順で実行される。

④アプリケーションの終了（「戻る」ボタンを押した時）

実行中→onPause→onStop→onDestroy の順で実行される。

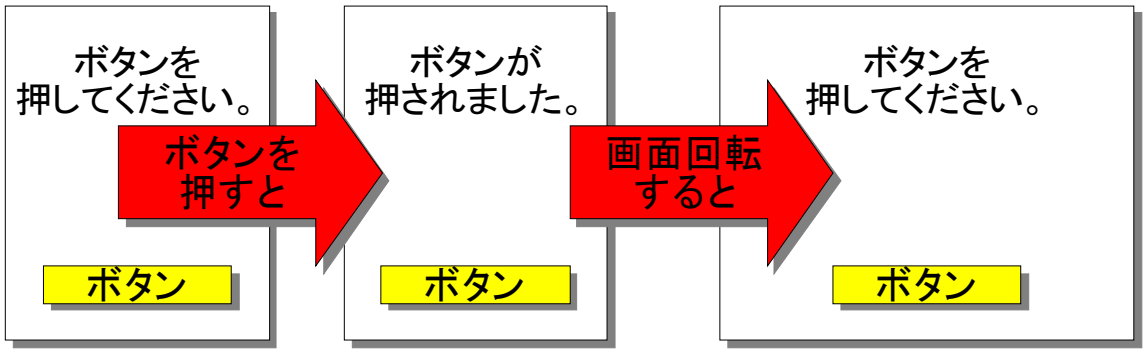
⑤画面回転時

実行中→onPause→onStop→onDestroy→onCreate→onStart→onResume の順で実行される。



9.3 画面回転の利用方法

- アプリケーション名：RollingApp
- 動作概要



◆作成手順

①新規アプリケーション「RollingApp」を作成する。

②「activity_main.xml」で「TextView」と「Button」を配置する。

③「MainActivity.java」でプログラムを設定する。

●プログラム例

ファイル名：MainActivity.java

```
public class MainActivity extends Activity implements OnClickListener {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        TextView text = (TextView)findViewById(R.id.textView);
        text.setText("ボタンが押されました。");
    }
}
```

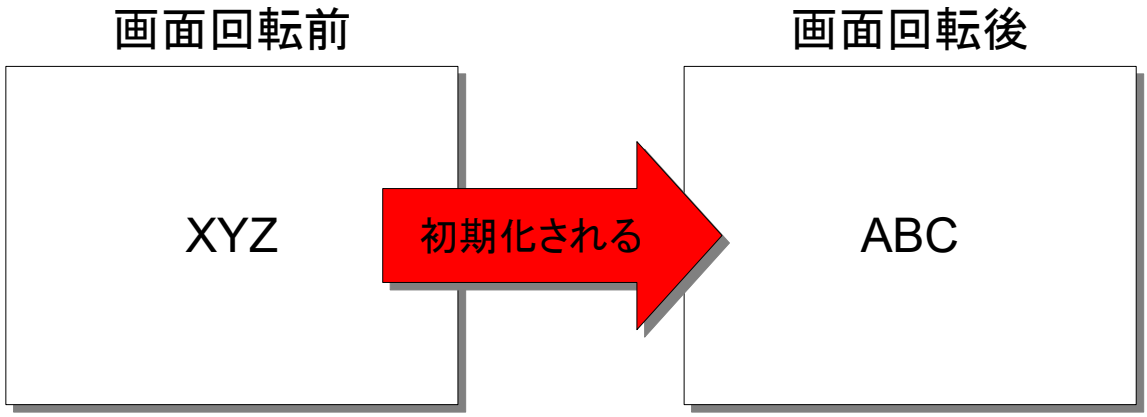
※一部抜粋

【実行確認方法】

「ボタン」を押すとテキスト表示が変わるが、画面回転させるとテキスト表示が初期化される。

9.4 画面回転の対応

画面回転すると部品の表示が初期化される。



Android では、本体の向きなどに応じて自動的に画面が回転して表示されます。画面回転が行われると部品の表示文字が初期化されてしまうため作成したアプリケーションに不具合が生じる可能性も多々あります。

このような不具合が生じる理由は画面回転するとライフサイクルメソッドが

「実行中→onPause→onStop→onDestroy→onCreate→onStart→onResume」

の順で実行されるためです。

onDestroy メソッドが実行されることにより Activity が破棄され、新たに Activity のオブジェクトが生成され onCreate メソッドを実行することにより初期値が設定されてしまいます。

このような不具合を生じさせないためには、２つの方法があります。

◆画面回転による不具合への対応方法

①オリエンテーション（方向づけ）を固定値にする。

●「AndroidManifest.xml」ファイルに「android:screenOrientation="portrait"」指定を追加する。

```
<activity
   android:name=".MainActivity"
   android:label="@string/title_activity_main"
   android:screenOrientation="portrait" >
   ・・・・・
</activity>

※一部抜粋
```

※Orientation が portrait の場合は縦固定で、landscape の場合は横固定に設定できる。

② Activity が破棄される前に、表示文字の値を保存しておき、新たに Activity 生成後、保存値を設定する。

●Activity クラスに定義されているデータ保存及び再設定処理を行うメソッド

メソッド	説明
onSaveInstanceState	【書式】 protected void onSaveInstanceState(Bundle outState) 【機能】 インスタンスが破棄される前にBundleに設定情報を保存する。
onRestoreInstanceState	【書式】 protected void onRestoreInstanceState(Bundle savedInstanceState) 【機能】 インスタンス生成時にBundleの設定情報を設定する。

●プログラム例

「RollingApp」の「MainActivity.java」に下記プログラムを追加記述する。

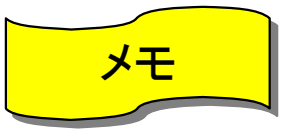
```
@Override
protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        TextView text = (TextView)findViewById(R.id.textView);
        outState.putString("str", (String) text.getText());
}

@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        TextView text = (TextView)findViewById(R.id.textView);
        text.setText(savedInstanceState.getString("str"));
}
```

★putString メソッド（Bundle クラスに定義）

```
【書式】
public void putString(String key, String value)

【機能】
指定したkeyに値valueを関連付けBundleに設定する。
```

★getString メソッド（Bundle クラスに定義）

```
【書式】
public String getString(String key)

【機能】
Bundleから指定したkeyの値を取得する。
```


メモ