

1ヶ月で

ANDROIDカメラアプリ開発

@Trunk
酒本伸也

2016/3/20

本日の流れ

- レイアウトの基本知識
- Activityのライフサイクル
- イベントハンドリング
- Widgetの状態管理
- (カメラ呼び出し)
- (画像受け取り)
- Aviary設定

ACTIVITY

ACTIVITY ≡ 画面

- ActivityはUIを持つ画面には必須
- 画面はxmlで作る
 - 動的に生成したいものはコードで定義
- 画面を構成するxmlは
レイアウトとウィジェットで成り立つ

LAYOUT

LAYOUT

- WidgetやViewを一定の規則に基づき配置するためのもの
- 一つのxmlにレイアウトは複数あっても良い
- 親のレイアウトは必ず一つ

いくつか紹介

- LinearLayout
 - 上から下、左から右など、順序に詰めて配置するレイアウト
- RelativeLayout
 - 配置した他のコンポーネントとの位置関係を定義しながら配置するレイアウト
- FrameLayout
 - 左上を起点として、子要素であるときも前面に表示されるレイアウト

XMLの設定

- setContentView(Int id)

```
9 public class MainActivity extends AppCompatActivity {  
10  
11     @Override  
12     protected void onCreate(Bundle savedInstanceState) {  
13         super.onCreate(savedInstanceState);  
14         setContentView(R.layout.activity_main);  
15         Log.d("Trunk", "onCreate");  
16     }  
17 }
```


- ステータスバー
- アクションバー
(ツールバー)
- ナビゲーションバー



My Application

Hello world!

```
MainActivity.java x AndroidManifest.xml x activity_main.xml x MyApplication x  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    tools:context=".MainActivity">  
  
    <TextView  
        android:text="@string/hello_world"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"/>  
  
</RelativeLayout>
```

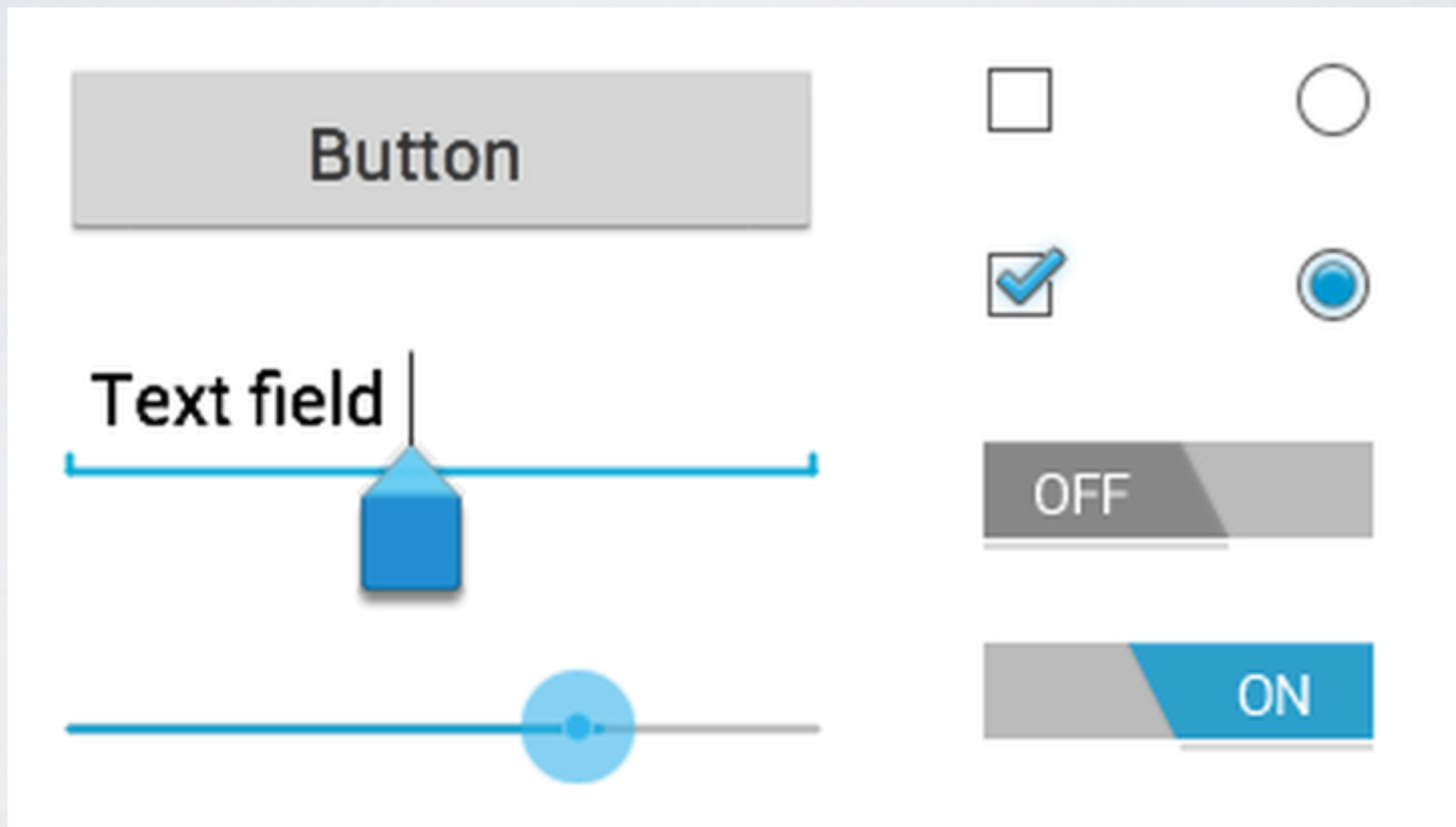
My Application

Hello world!

```
MainActivity.java x AndroidManifest.xml x activity_main.xml x MyApplication x  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    tools:context=".MainActivity">  
    <TextView  
        android:text="@string/hello_world"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"/>  
</RelativeLayout>
```

WIDGET

WIDGET



高頻出

- Button
 - ボタンを配置したいときに使う
- TextView
 - 決められた文字を配置したいときに使う
- ImageView
 - 画像を表示したいときに使う

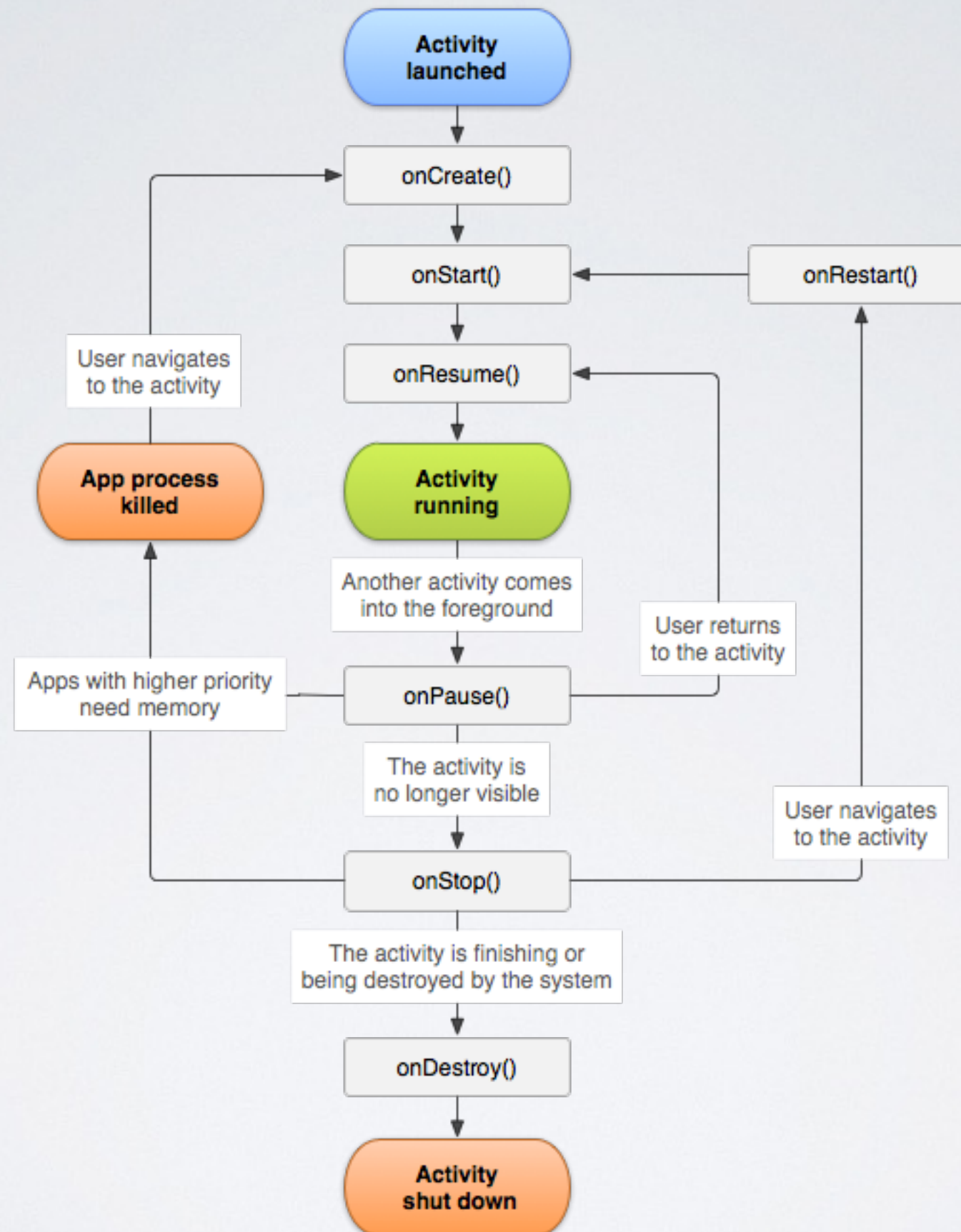
実習

- 自分が作りたいアプリのレイアウトを考え
実装してみる

ライフサイクル

ライフサイクル

- Activity同士の円滑なやり取りを助長する
 - 複数のActivity
- ライフサイクル ≡ 状態遷移

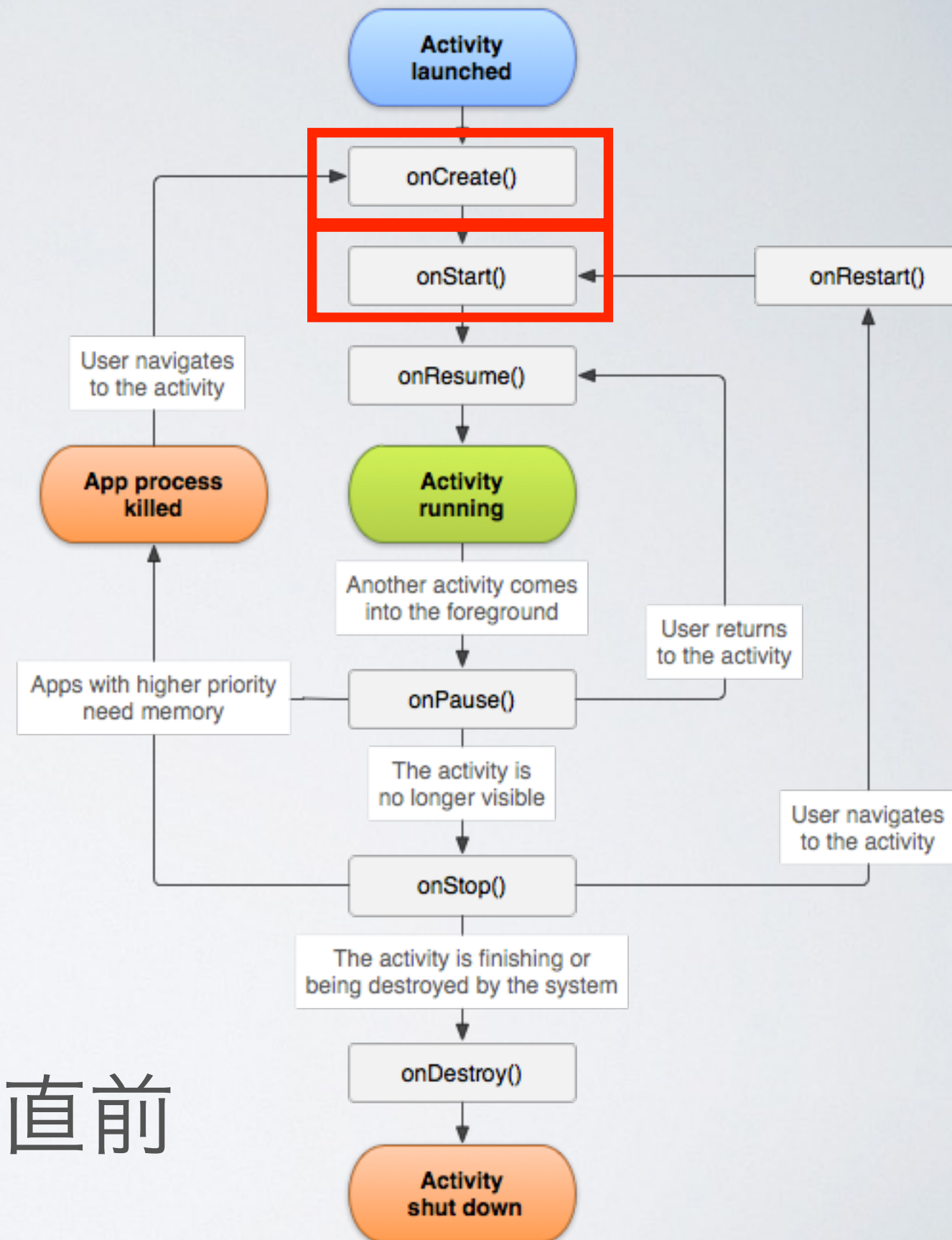


- onCreate

- Activityが初めて作られたとき

- onStart

- Activityがユーザに見えるようになる直前

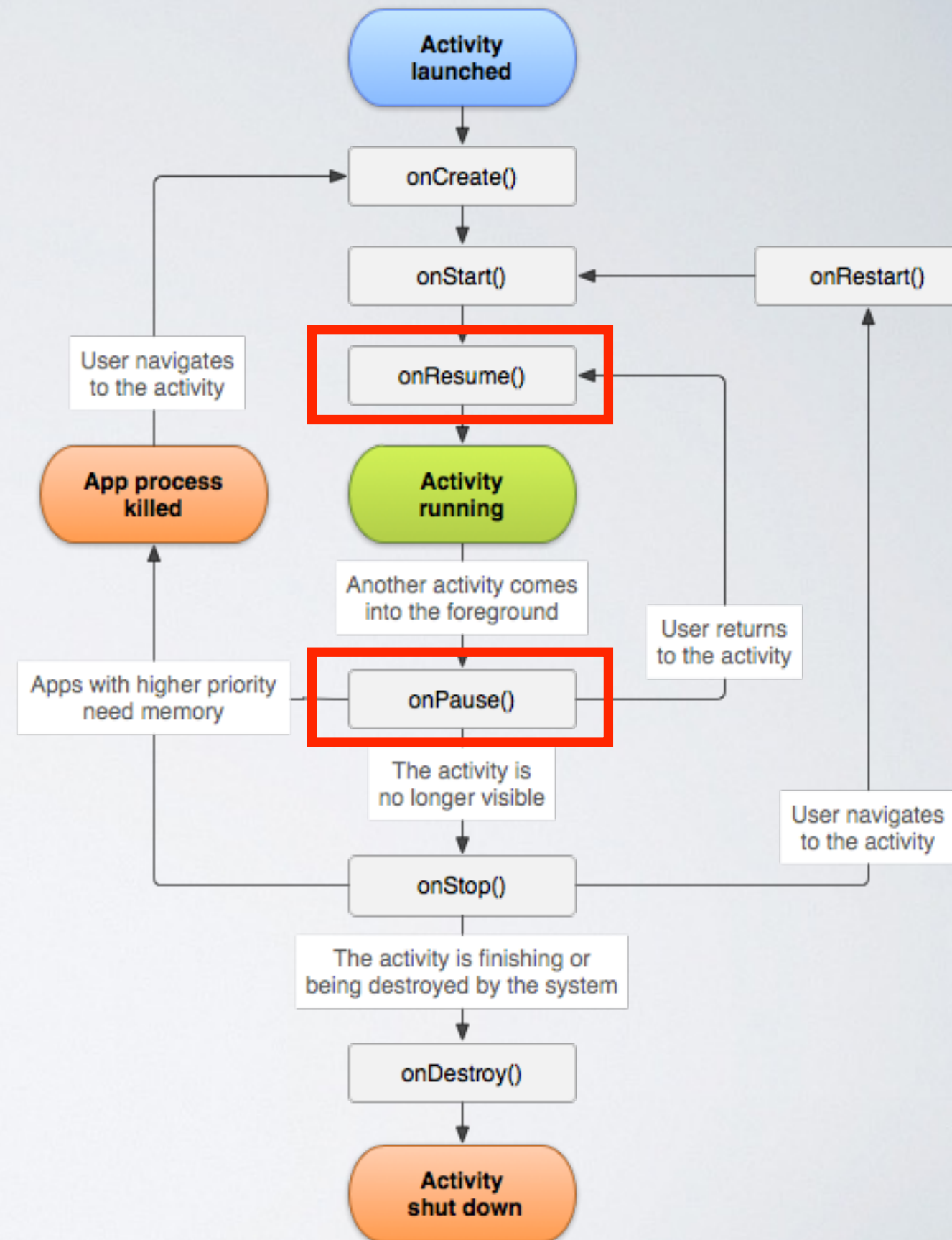


- onResume

- Activityが
表示されたとき

- onPause

- 別のActivityが
表示されるとき



- onStop

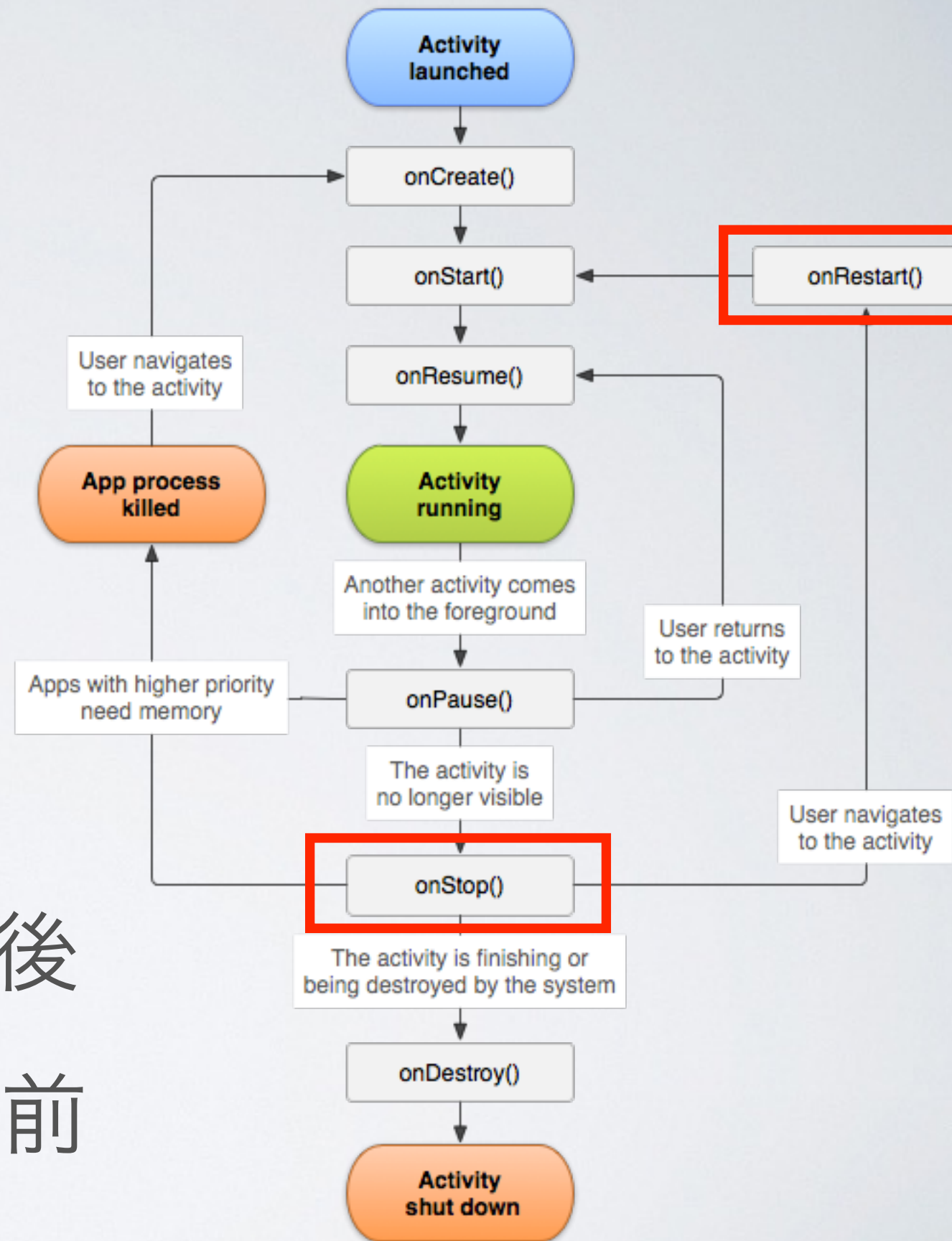
- Activityが

見えなくなった時

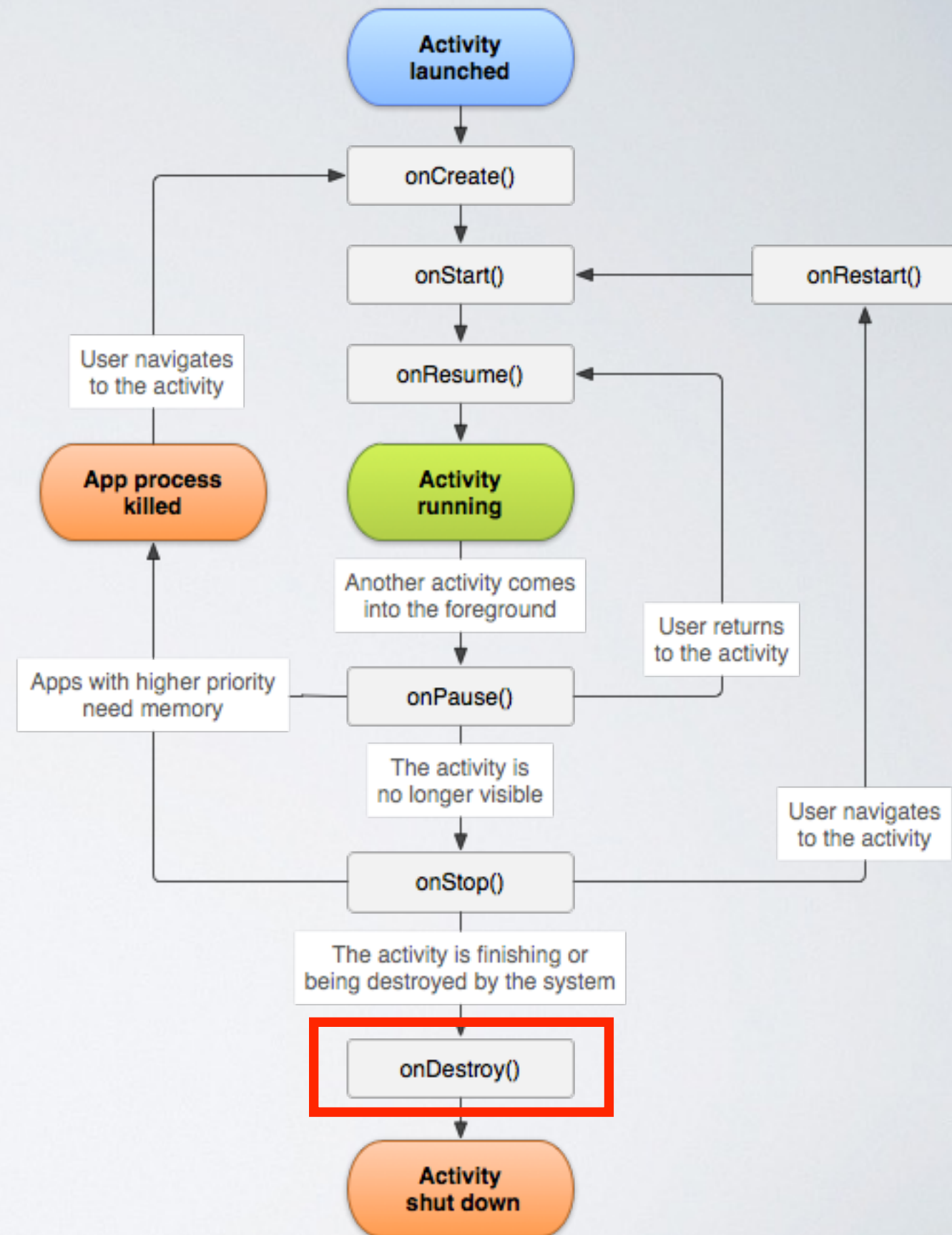
- onRestart

- Activityが停止した後

再度開始される直前



- onDestroy
- Activityが
破棄される直前
- Activityが受け取る
最後の呼び出し



詳細

- <http://dev.classmethod.jp/smartphone/android/android-tips-21-activity-lifecycle/>

イベントハンドリング

イベントハンドリング

- イベント
 - ユーザの何かしらの行動や
内部の処理の特定のタイミング
- イベントを適切なタイミングでキャッチし
適切な処理を行う。

ボタンをクリック

- ButtonのWidgetを設置
- Buttonがクリックされた時に行う処理を記述
- Buttonのクリックイベントをハンドリング

BUTTONのWIDGETを設置

- xml

```
activity_main.xml x
1  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:tools="http://schemas.android.com/tools"
3      android:id="@+id/layout_main"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:paddingLeft="16dp"
7      android:paddingRight="16dp"
8      android:paddingTop="16dp"
9      android:paddingBottom="16dp"
10     tools:context=".MainActivity">
11
12     <Button
13         android:id="@+id/button"
14         android:layout_centerInParent="true"
15         android:text="@string/hello_world"
16         android:layout_width="wrap_content"
17         android:layout_height="wrap_content"/>
18
19 </RelativeLayout>
20
```


BUTTONがクリックされた時 に行う処理を記述

- `setOnClickListener`に実装を渡してあげる
 - `Listener = Interface`


```
10
11 public class MainActivity extends AppCompatActivity implements View.OnClickListener {
12
13     public final static String TAG = "Trunk";
14
15     private MyCustomClick mMyCustomClick = new MyCustomClick();
16     private View.OnClickListener mOnClickListener = new View.OnClickListener() {
17         @Override
18         public void onClick(View v) {
19             Log.d(TAG, "mOnClickListener: onClick");
20         }
21     };
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_main);
27
28         Button button = (Button)findViewById(R.id.button);
29
30         button.setOnClickListener(mMyCustomClick);
31         button.setOnClickListener(new MyCustomClick());
32         button.setOnClickListener(mOnClickListener);
33
34         button.setOnClickListener(new View.OnClickListener() {
35             @Override
36             public void onClick(View v) {
37                 Log.d(TAG, "View.OnClickListener: onClick");
38             }
39         });
40
41         button.setOnClickListener(this);
42
43     }
44
```

実装

ハンドリング

```
10
11 public class MainActivity extends AppCompatActivity implements View.OnClickListener {
12
13     public final static String TAG = "Trunk";
14
15     private MyCustomClick mMyCustomClick = new MyCustomClick();
16     private View.OnClickListener mOnClickListener = new View.OnClickListener() {
17         @Override
18         public void onClick(View v) {
19             Log.d(TAG, "mOnClickListener: onClick");
20         }
21     };
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_main);
27
28         Button button = (Button)findViewById(R.id.button);
29
30         button.setOnClickListener(mMyCustomClick);
31         button.setOnClickListener(new MyCustomClick());
32         button.setOnClickListener(mOnClickListener);
33
34         button.setOnClickListener(new View.OnClickListener() {
35             @Override
36             public void onClick(View v) {
37                 Log.d(TAG, "View.OnClickListener: onClick");
38             }
39         });
40
41         button.setOnClickListener(this);
42
43     }
44
```

ハンドリング
+
実装

```
10
11 public class MainActivity extends AppCompatActivity implements View.OnClickListener {
12
13     public final static String TAG = "Trunk";
14
15     private MyCustomClick mMyCustomClick = new MyCustomClick();
16     private View.OnClickListener mOnClickListener = new View.OnClickListener() {
17         @Override
18         public void onClick(View v) {
19             Log.d(TAG, "mOnClickListener: onClick");
20         }
21     };
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_main);
27
28         Button button = (Button)findViewById(R.id.button);
29
30         button.setOnClickListener(mMyCustomClick);
31         button.setOnClickListener(new MyCustomClick());
32         button.setOnClickListener(mOnClickListener);
33
34         button.setOnClickListener(new View.OnClickListener() {
35             @Override
36             public void onClick(View v) {
37                 Log.d(TAG, "View.OnClickListener: onClick");
38             }
39         });
40
41         button.setOnClickListener(this);
42
43     }
44
```

実装?

ハンドリング

別クラスで実装

- MyCustomClick.class

```
public class MyCustomClick implements View.OnClickListener{  
    @Override  
    public void onClick(View v) {  
        Log.d(MainActivity.TAG, "MyCustomClick onClick");  
    }  
}
```



```
10
11 public class MainActivity extends AppCompatActivity implements View.OnClickListener {
12
13     public final static String TAG = "Trunk";
14
15     private MyCustomClick mMyCustomClick = new MyCustomClick();
16     private View.OnClickListener mOnClickListener = new View.OnClickListener() {
17         @Override
18         public void onClick(View v) {
19             Log.d(TAG, "mOnClickListener: onClick");
20         }
21     };
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_main);
27
28         Button button = (Button)findViewById(R.id.button);
29
30         button.setOnClickListener(mMyCustomClick);
31         button.setOnClickListener(new MyCustomClick());
32         button.setOnClickListener(mOnClickListener);
33
34         button.setOnClickListener(new View.OnClickListener() {
35             @Override
36             public void onClick(View v) {
37                 Log.d(TAG, "View.OnClickListener: onClick");
38             }
39         });
40
41         button.setOnClickListener(this);
42     }
43
44
```

実装?

ハンドリング

同じクラスで実装

- 実装が記述されているのでthisが使える。

```
66  
67  
68       
69     @Override  
70     public void onClick(View v) {  
71         Log.d(TAG, "MainActivity: onClick");  
72     }  
73 }
```

```
10
11 public class MainActivity extends AppCompatActivity implements View.OnClickListener {
12
13     public final static String TAG = "Trunk";
14
15     private MyCustomClick mMyCustomClick = new MyCustomClick();
16     private View.OnClickListener mOnClickListener = new View.OnClickListener() {
17         @Override
18         public void onClick(View v) {
19             Log.d(TAG, "mOnClickListener: onClick");
20         }
21     };
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_main);
27
28         Button button = (Button)findViewById(R.id.button);
29
30         button.setOnClickListener(mMyCustomClick);
31         button.setOnClickListener(new MyCustomClick());
32         button.setOnClickListener(mOnClickListener);
33
34         button.setOnClickListener(new View.OnClickListener() {
35             @Override
36             public void onClick(View v) {
37                 Log.d(TAG, "View.OnClickListener: onClick");
38             }
39         });
40
41         button.setOnClickListener(this);
42
43     }
44
```

実装します宣言

SET_ON_CLICK以外のリスナー

- setOn○○○で用意されている。
- 登録/解除が必要なものは
add/remove
register/unregister
など、命名規則で判別できる

実習

- 自分が作りたいアプリのいくつかのwidgetでイベントハンドリングを設定する

WIDGETの状態

BUTTONがクリックされた時

- なんどもクリックできる
- クリックできたかどうかわからない
- クリック可能な状態か不明

なんどもクリックできる

- メールを送るなど、複数回されては困るものがある
 - View#setEnabled(boolean)
 - Viewの状態を、有効/無効にする

クリックできたかどうか
わからない

クリック可能な状態か不明

- ユーザにフィードバックを与える

```
<Button  
    android:id="@+id/button"  
    android:layout_centerInParent="true"  
    android:text="@string/hello_world"  
    android:background="@drawable/button_state"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"/>
```

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <selector xmlns:android="http://schemas.android.com/apk/res/android">
3      <item
4          android:state_enabled="false"
5          <shape
6              android:shape="rectangle"
7              <solid android:color="#00ff00" />
8          </shape>
9      </item>
10     <item
11         android:state_pressed="true"
12         android:state_enabled="true"
13         <shape
14             android:shape="rectangle"
15             <solid android:color="#ff0000" />
16         </shape>
17     </item>
18     <item
19         android:state_pressed="false"
20         android:state_enabled="true"
21         <shape
22             android:shape="rectangle"
23             <solid android:color="#0000ff" />
24         </shape>
25     </item>
26 </selector>
```

無効な状態

有効で
押されている状態

有効で
押されていない状態

AVIARY

AVIARY

- エディター機能は別SDK
 - <https://creativesdk.adobe.com/docs/android/#/articles/imageediting/index.html>
 - 一緒にやっていきましょう。