本日の内容

TouchEvent

今回マニュアル感がとても強いです。

Androidアプリ内で出来ることを増やす目的で授業を行います。

TouchEvent(公式ドキュメント)

<https://developer.android.com/training/gestures/movement?hl=ja>

■Touchとは

Androidアプリ内に作成する、メニューボタンのようなものです。

使用する場合このオーバーフローメニューに対してもレイアウトファイルが必要になります。

■プロジェクト作成

- Empty Activity

- プロジェクト名：TryTouchEvent

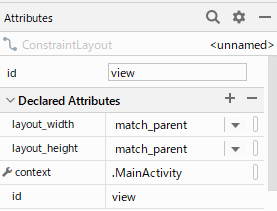
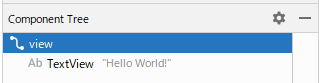
＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

■activity\_main.xml

今回、画面全体に対してタッチされたかの判定を取得したいため、

画面全体が該当するルートタグ(ConstraintLayout)に対して、タッチイベントの検知を行います。

PG上で識別できるようidを付与しておきましょう



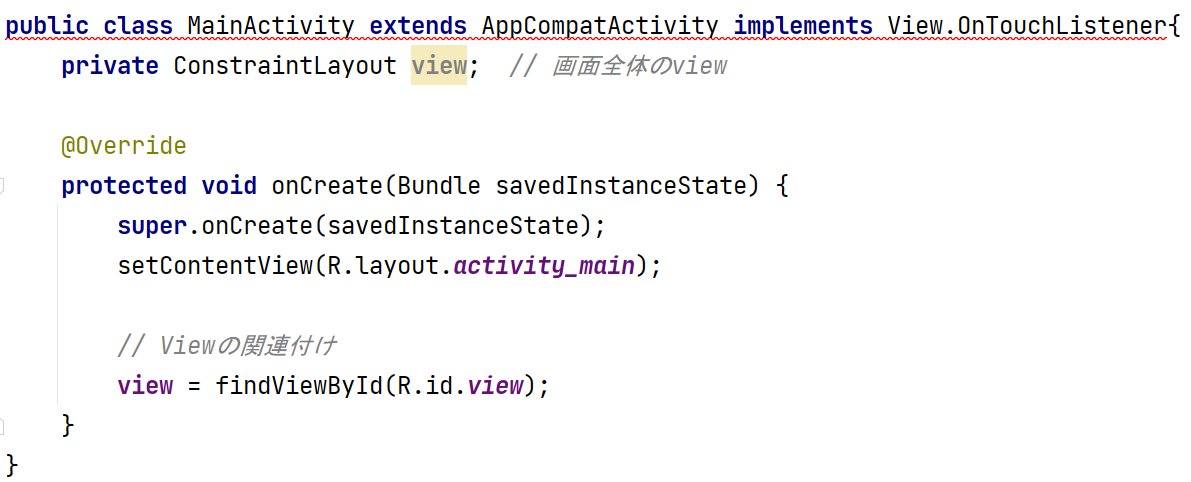
■MainActivity.java

先程のConstraintLayoutに対して処理を行えるよう、「変数の定義」と「viewとの関連付け」を行います



■View.OnTouchListener

タッチしたかを判定するインターフェイスを実装します。

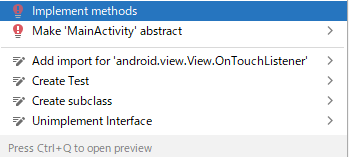
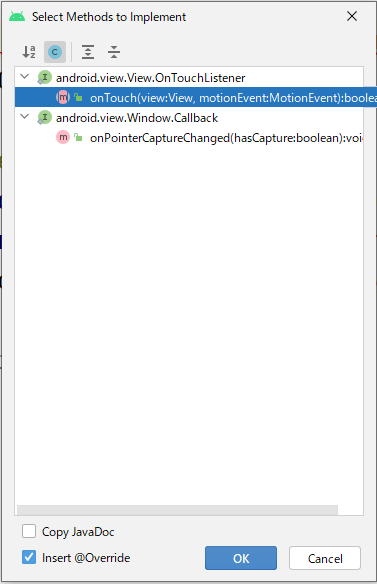


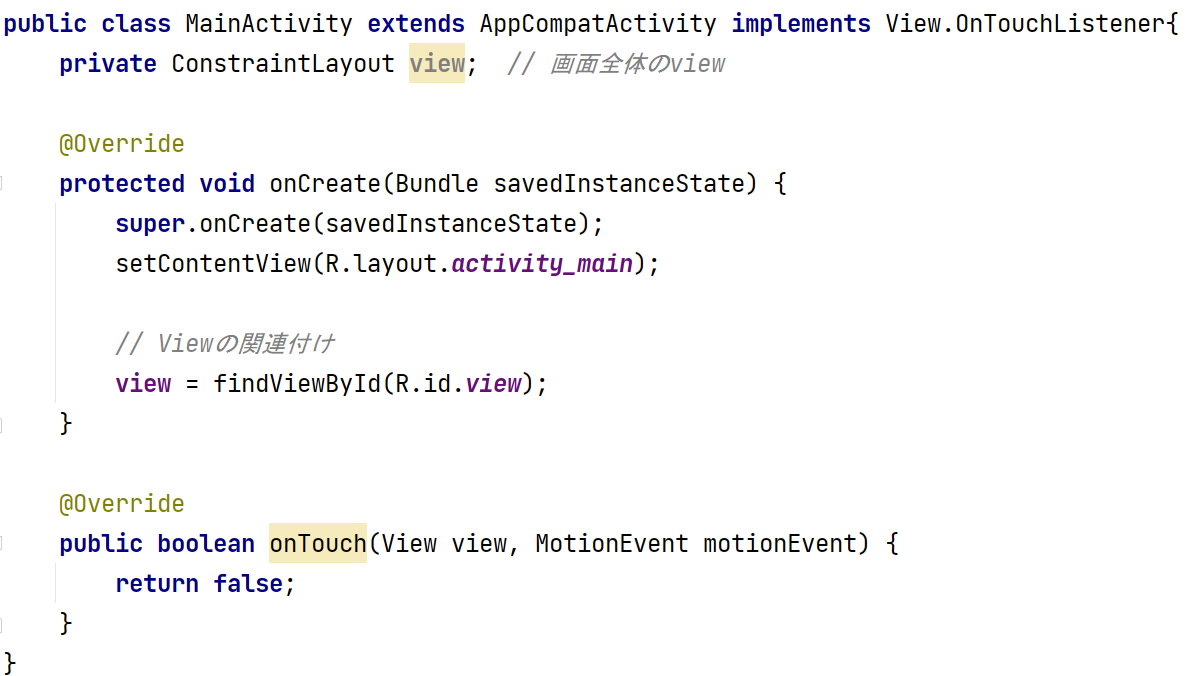
しかし、赤波線が出ていますね、これは必要なメソッドが実装されていないからです。

View.OnTouchListenerをクリックした状態で「Alt」+ 「Enter」で

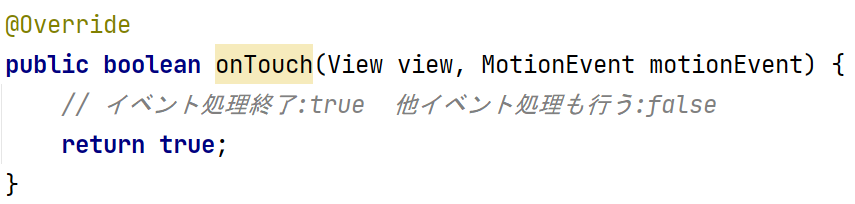
必要なメソッドを自動的に実装させましょう

今回はonTouchメソッドを実装します。



実装されたonTouchメソッド内の戻り値をtrueに変更しておきましょう。



onTouchメソッドを実装しましたが、このままでは何をタッチしたら

このメソッドを呼ぶのかが決めていません。

画面(ConstraintLayout)をタッチしたら、onTouchメソッドが呼ばれるようにしましょう

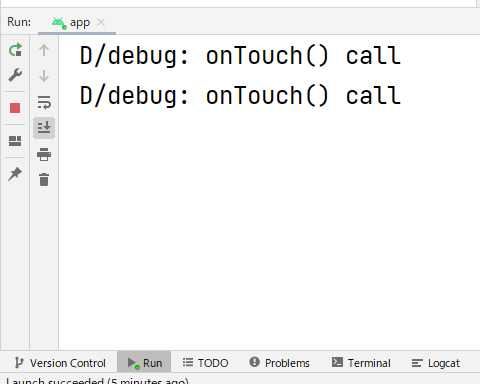
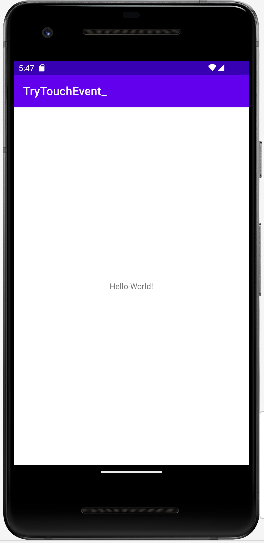


これでタッチを検知出来るようになったので、ログ処理を記述して

画面タッチが認識されているか確認してみましょう！



下部のRunタブを開いた状態で、Androidの画面を1回クリックしてみましょう

何故か2回反応していますね。これは一体・・・・？

■onTouchで渡される引数

先程の謎の鍵は引数にあります。

public boolean onTouch(View view, MotionEvent motionEvent) { }

第1引数：View view,

これは、タッチされたviewが入っています。

第2引数： MotionEvent motionEvent

ここにはどのような操作をしたかが入っています。

私たちはタップしただけですが、

Androidには

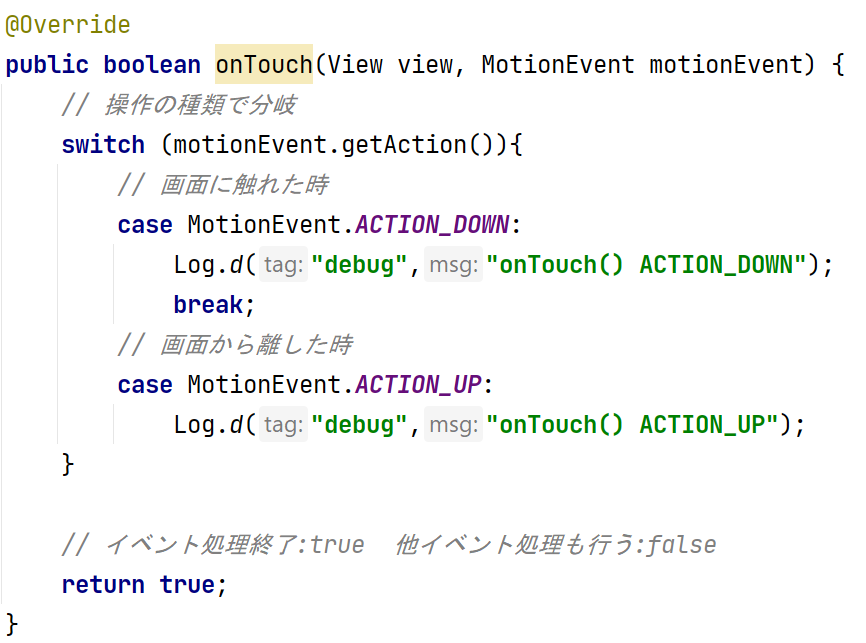
・画面に指が触れたアクション

・画面から指を離したアクション

この2種類の処理が行われていたのです。

なので触れた時と離れた時で分岐をさせる必要があります。

どのアクションだったかは[motionEvent.getAction()]で取得可能です。

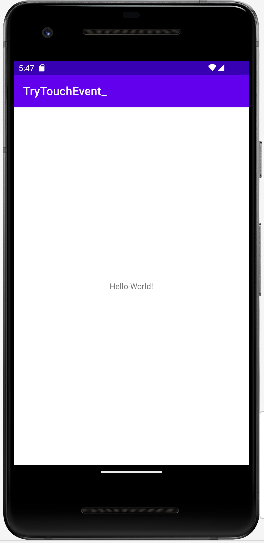
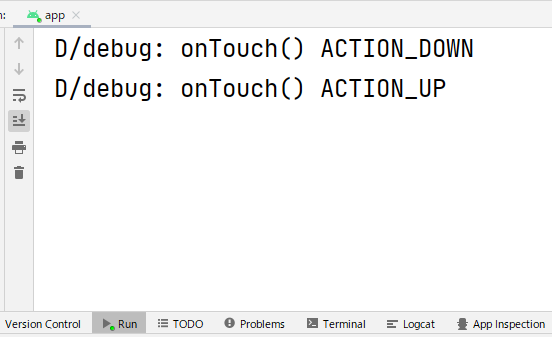


実行してみると

押したときにACTION\_DOWN

離したときにACTION\_UP

のログが出力されていますね！



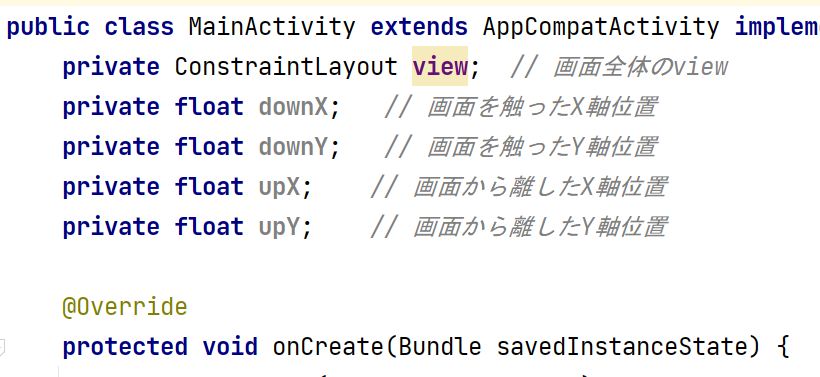
■座標の取得

今回はフリック操作で方向を検知したいのですが、

始点と終点がわからないと方向が求められません。

それぞれの座標を取得していきます。

まずは座標値を格納する変数の定義から。



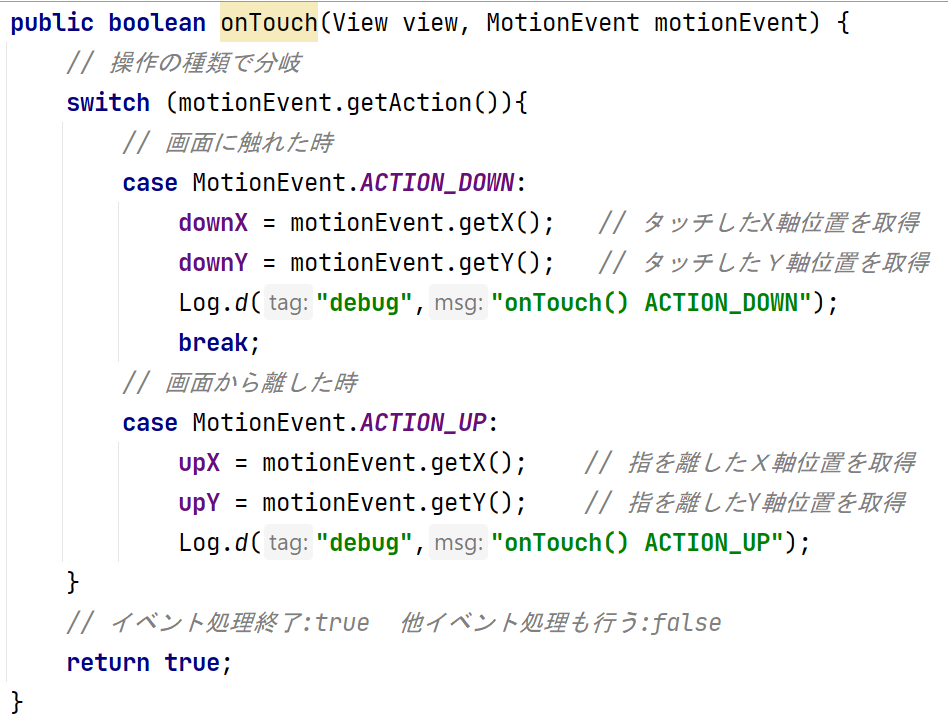
次にそれぞれの座標を取得します。

取得する方法は

motionEvent.getX()でX軸

motionEvent.getY()でY軸

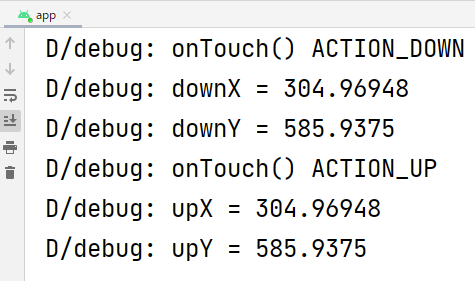
上記に値を取得します。その際,float型で取得されます。



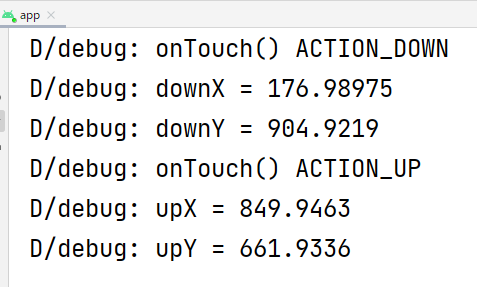
開発者も座標がわかるようにログの出力するように追記しておきましょう。



タップした時のログ

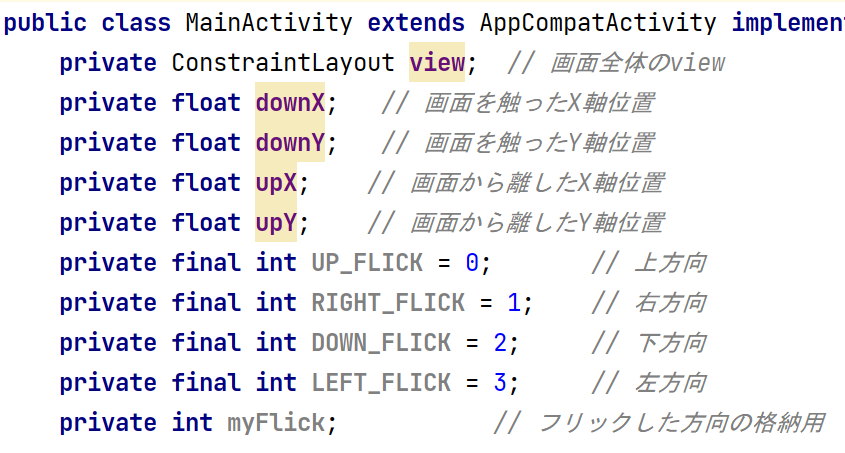


フリックor スライドした時のログ

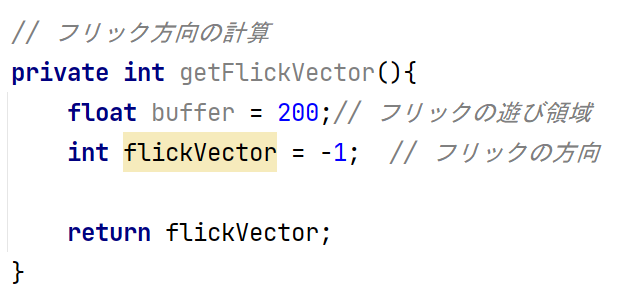


この座標を元に、上下左右どの方向にフリックしたのかを計算し分岐する必要があります。

先に管理しやすいよう、方向の変数と格納用の変数を定義しておきましょう。



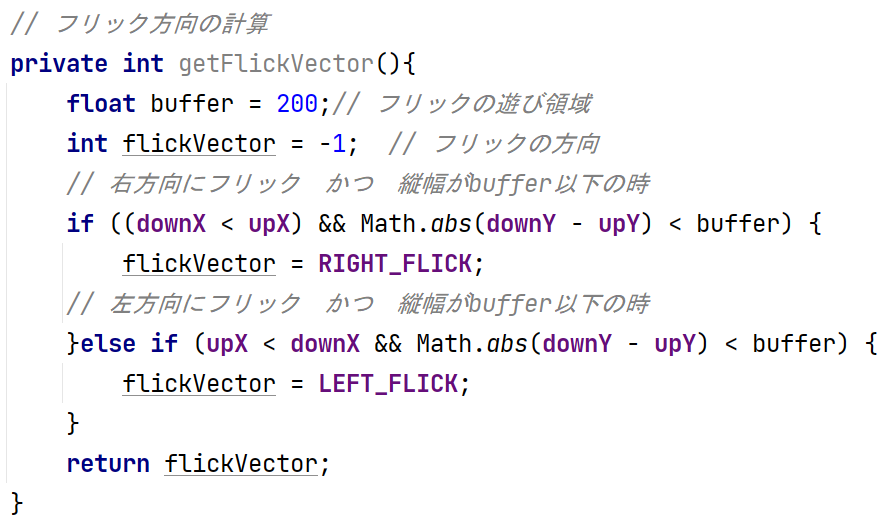
フリック方向を計算するためのメソッドを定義していきましょう。



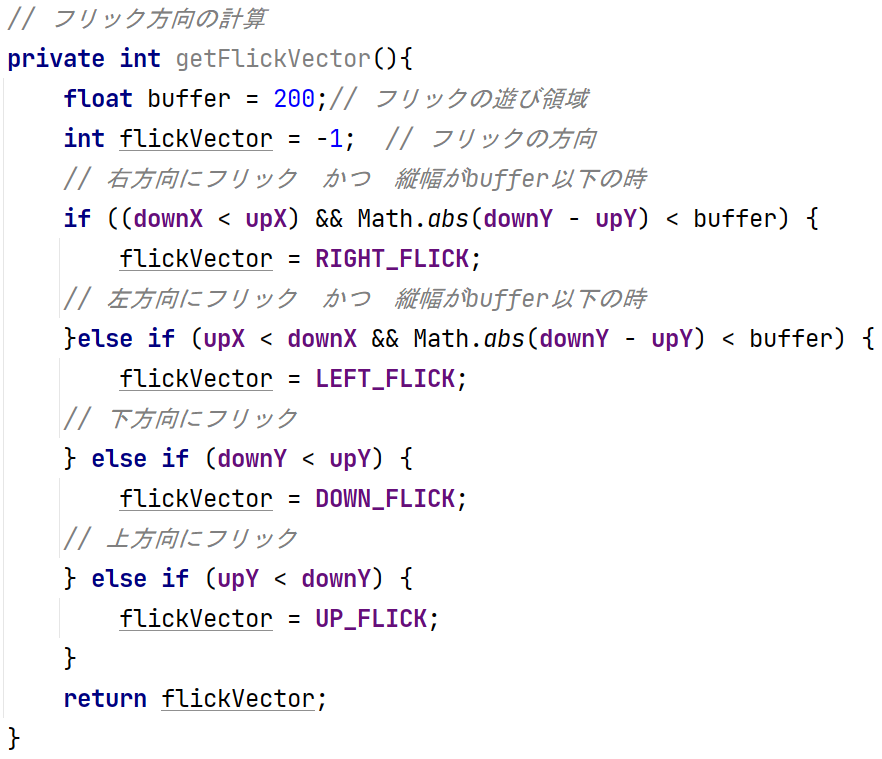
まずは左右方向の確認を行います。

平行に左右へフリックするとは限らない為、ある程度上下にぶれても判定できるようにします。

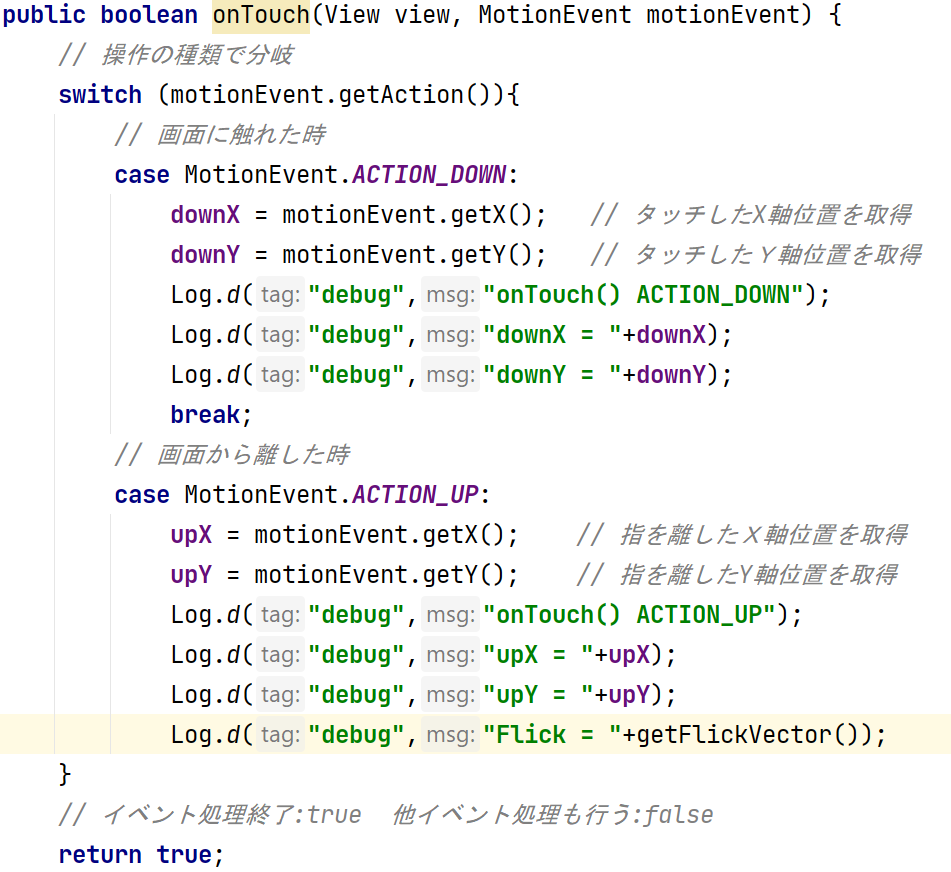
上下の判定はMath.abs()を活用し、２つの値の差を取得します。

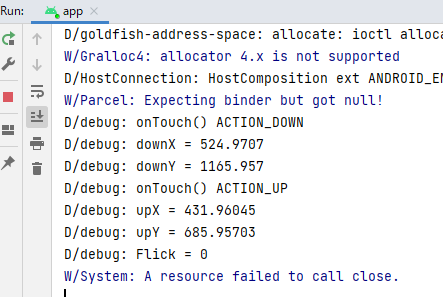


上下方向の判定も追加します。



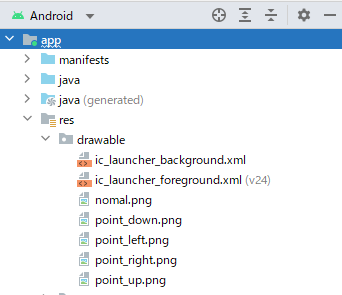
方向を判別出来ているか確認の為のログを出力させてみましょう





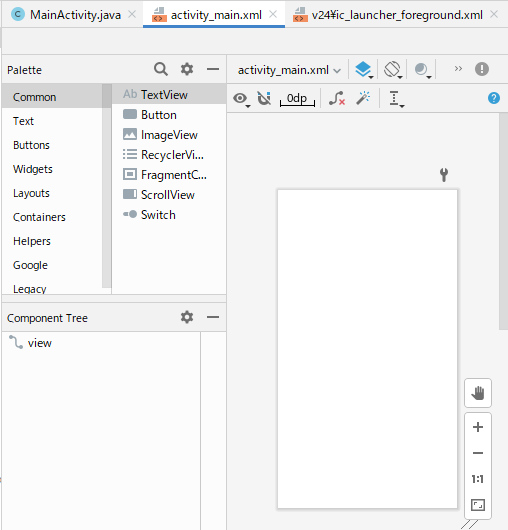
方向も判定できるようになっていますね

次に、CPUとバトル出来るように画像を追加していきましょう



■Activity\_main.xml

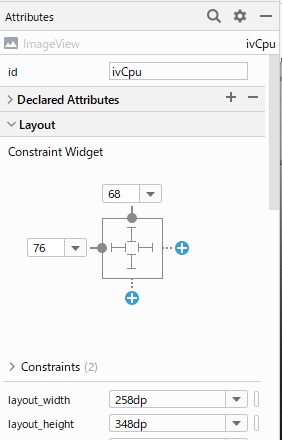
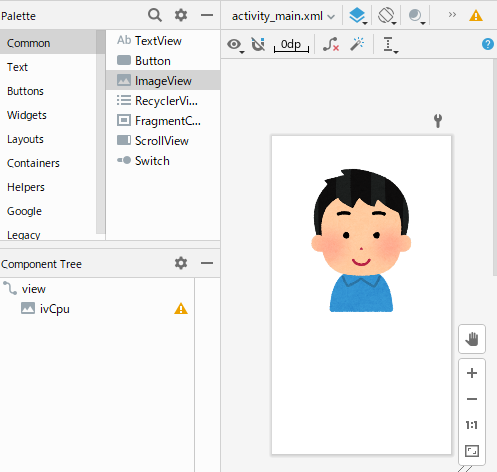
いったんtextViewを削除します



ImageViewを配置しましょう

Id: ivCpu に変更

位置制約も忘れずに設定



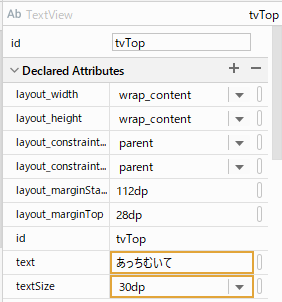
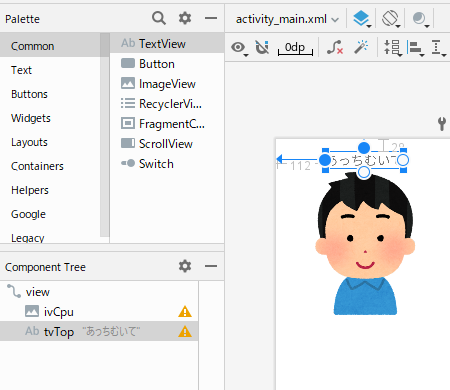
トップの文字用のTextViewを配置

Id : tvTop

TextSize : 30dp

Text : あっちむいて

位置制約も忘れずに



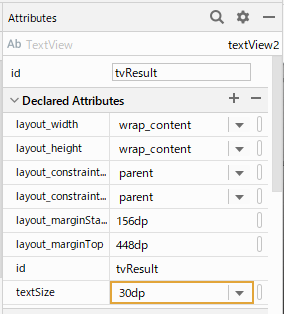
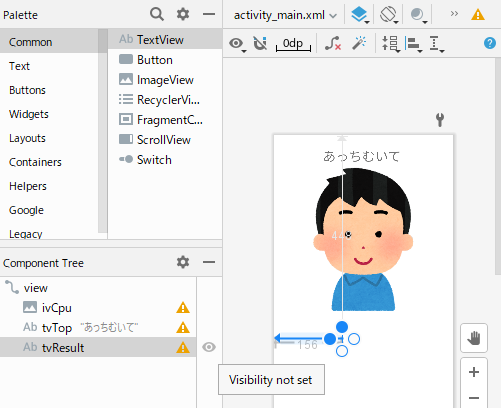
結果表示用のTextViewも配置

Id : tvResult

TextSize : 30dp

Text :

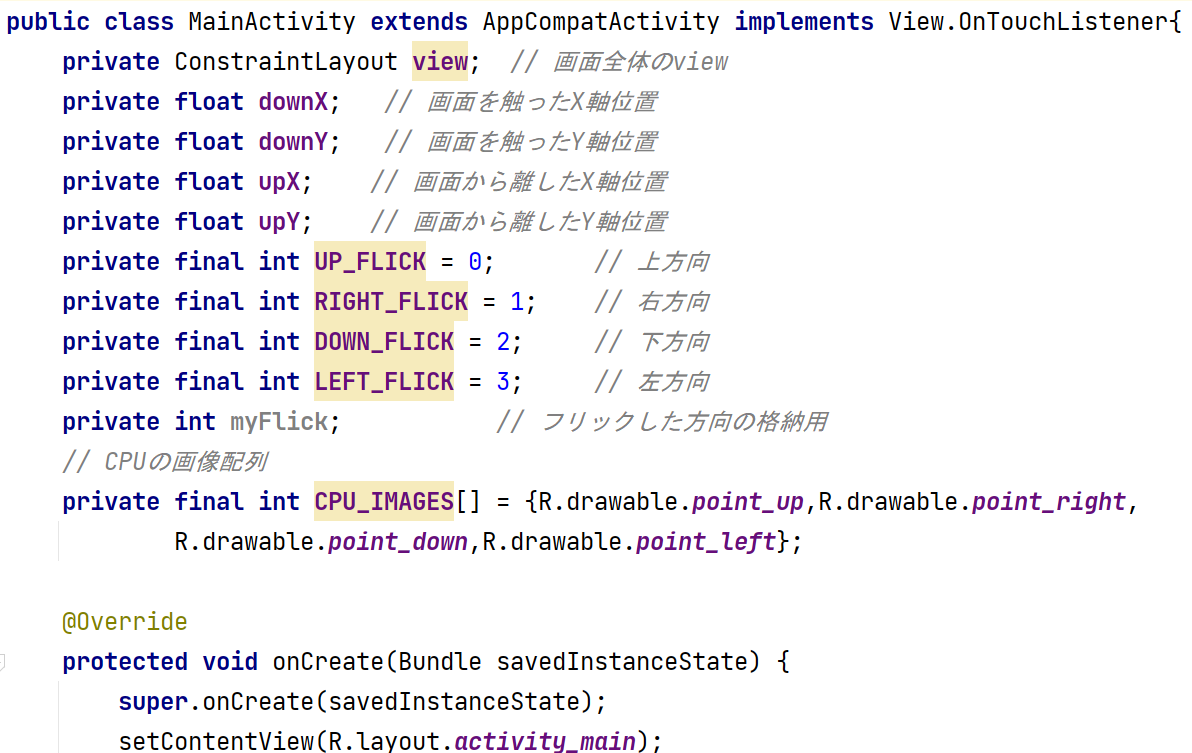
位置制約も忘れずに



レイアウトの準備ができたので、

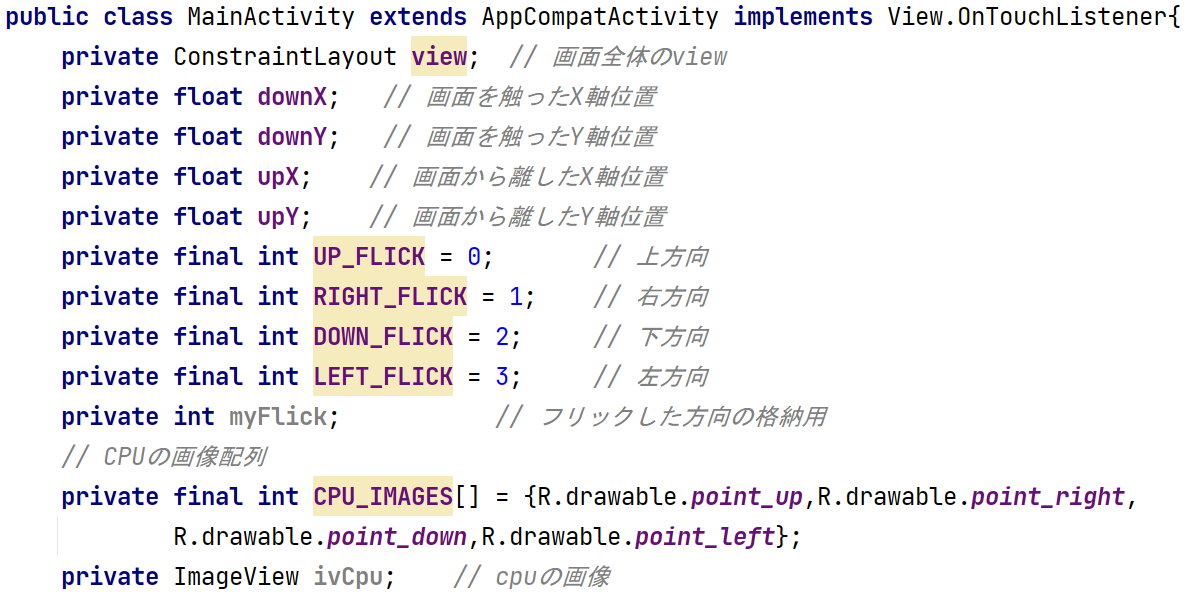
フリック後に画像と結果を表示するように処理を追加していきましょう

表示させる画像idの配列を作成し取り出しやすくしておきましょう



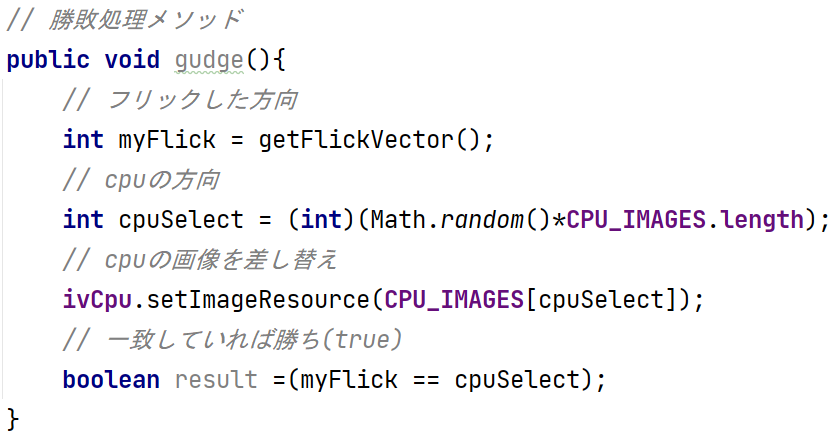
CPUの画像を差し替えれるように

ImageViewの宣言と関連付けを行います



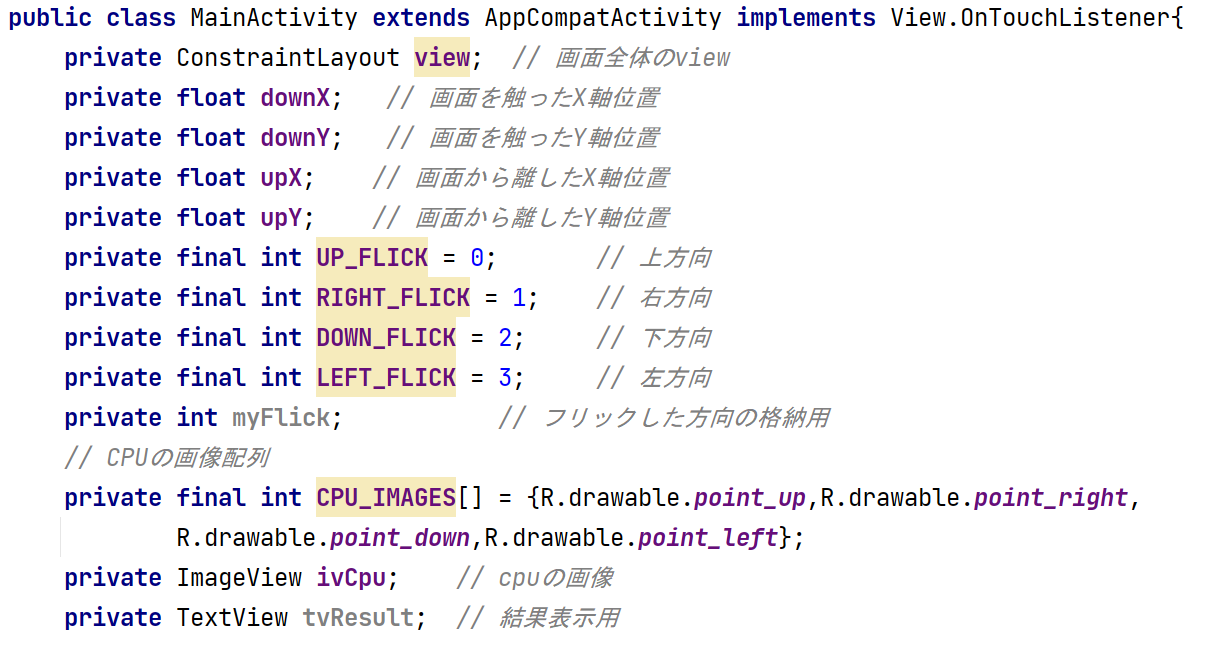


勝敗判定メソッドを作成し、CPUの内容によって画像を差し替えましょう。

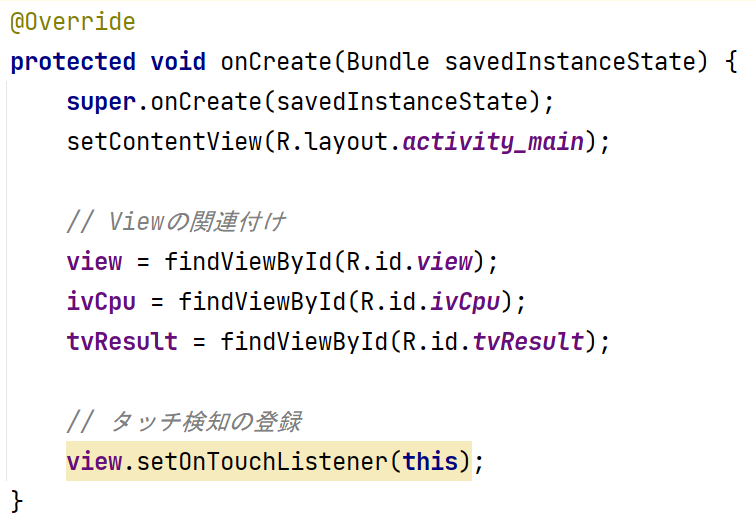


しかし、このままでは勝敗がわからないので、TextViewに結果を表示するよう処理を追加します。

TextViewの宣言



Viewとの関連付け



結果をTextViewに表示させます。



最後に勝敗勝利をどのタイミングで行うかですが、

フリックして、指を離したタイミングで行うようにしてみましょう



