

本日の内容

clickEvent リスナ

デバッグ実行

前回ではボタンを配置し電卓のレイアウトを作成しましたが
ボタンを押しても何も処理が行われませんでした。

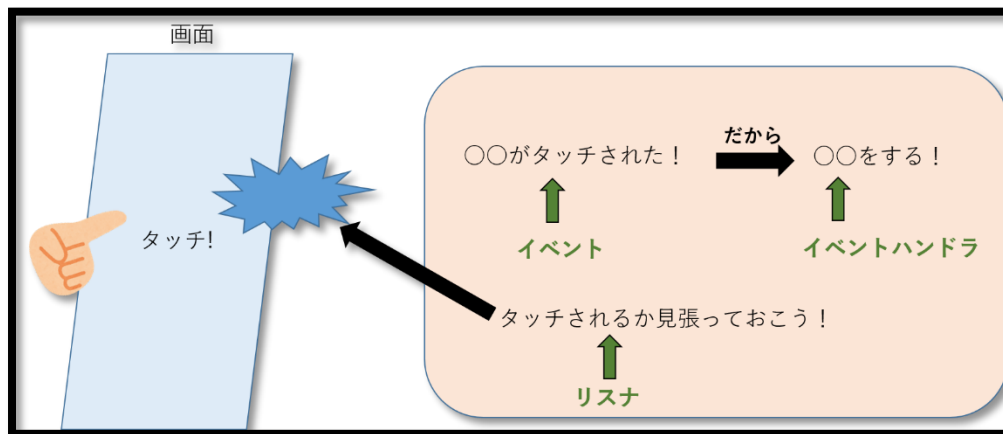
今回は、ボタンを押したときに処理を行う方法を学んでいきましょう。

■「イベント」と「イベントハンドラ」と「リスナ」

Android では、ボタンをタップ、アイコンをドラッグなど、ユーザが画面に対して何かの操作を行います
この操作の事を「イベント」

そのイベントに対応して行う処理のことを「イベントハンドラ」

イベントの検出を行っているものを「リスナ」と呼びます。



前回、何も反応しなかったのはこのリスナの設定がされていなかった為です。

このイベント、イベントハンドラ、リスナという考え方は Android だけでなく
iOS やデスクトップアプリなど、ユーザ操作に応じて処理を行う際の共通した考え方です。

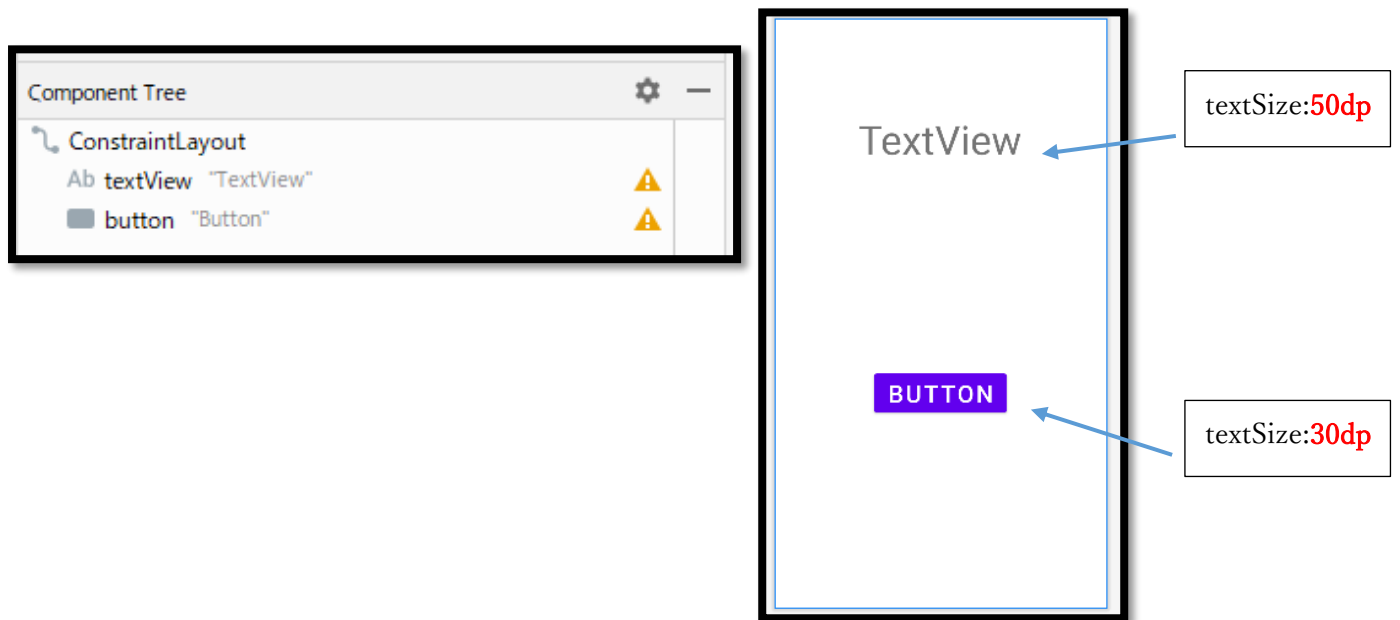
スマートフォンアプリ開発演習

まずは土台となるレイアウトを作成します。

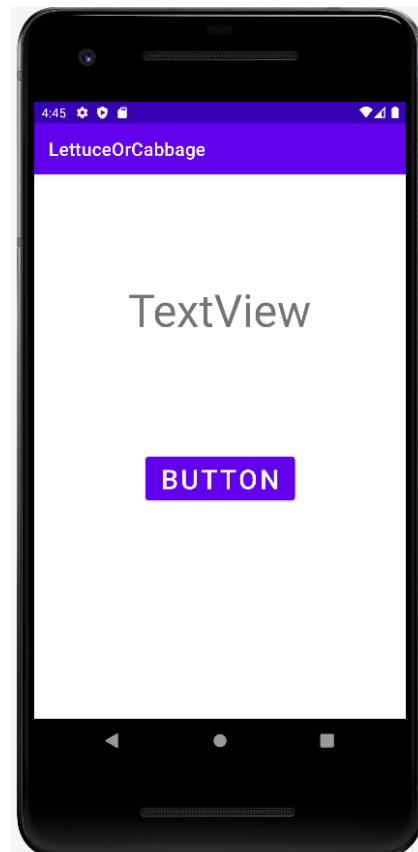
アプリ名：LettuceOrCabbage

テンプレート：EmptyActivity

レイアウトはデフォルトの「constraintlayout」で以下のように配置してください



このまま実行しボタンをクリックしても
何も処理は行われません。



■
MaiActivity.java にインナークラス(クラス内にクラスを作成すること)を追加します

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
  
    /**  
     * ボタンをクリックした時に呼ばれるクラス  
     */  
    private class SelectListener implements View.OnClickListener {  
  
    }  
}
```

赤字になっているのはインポート出来ていない為です。

「Alt」 + 「Enter」 キーを押すと自動でインポート可能です

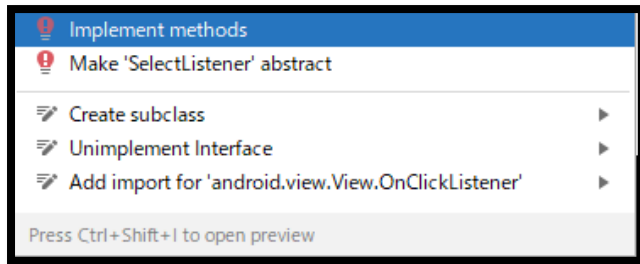
```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
  
    /**  
     * ボタンをクリックした時に呼ばれるクラス  
     */  
    private class SelectListener implements View.OnClickListener {  
  
    }  
}
```

インポートは出来ましたが、赤波線が増えてしまいました。

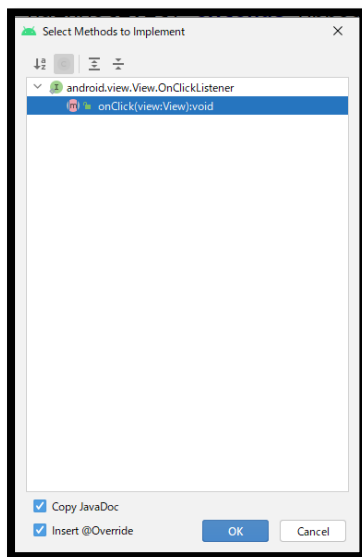
これは「インターフェイス」が実装されていない為です。全て手打ちで書くことも出来ませんがスペルミスなどが多いので、AndroidStudio の機能を活用していきましょう。

スマートフォンアプリ開発演習

赤波線に対してクリックした状態で「Alt」＋「Enter」キーを押すと出てくる
「Implement methods」を選択します



すると必要なインターフェイスがリストで出てきますのでOKを押下



```
/**
 * ボタンをクリックした時に呼ばれるクラス
 */
private class SelectListener implements View.OnClickListener {

    @Override
    public void onClick(View view) {

    }

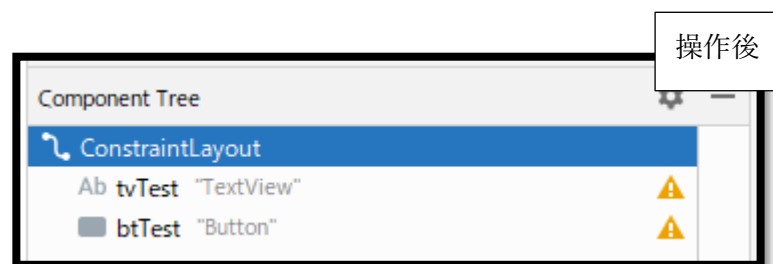
}
```

必要な onClick メソッド(イベントハンドラメソッド)が自動で実装されました！
とても便利な機能なので覚えていきましょう。

これで検知する為のクラスを生成しましたが、
どのボタンに対して検知するか等、識別しやすいよう先に
各ウィジェットに id を命名しておきましょう。

TextView の id: tvTest

Button の id: btTest



スマートフォンアプリ開発演習

■各ウィジェットを扱えるように変数として宣言

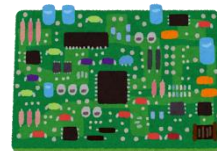
Android java では java 内の Button 変数と xml 内でのレイアウトとしての Button タグが存在しています。

xml はただの図面

java では部品としての変数となっており、この2つを繋ぐ処理が必要になります。

まずは各ウィジェットの変数を作成し

```
public class MainActivity extends AppCompatActivity {  
    TextView textView; // TextViewの変数  
    Button button;     // ボタンの変数  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```



Java の部品



xml のレイアウト

setContentView でレイアウトを読み込んでから、変数に関連付ける処理を行います。

```
public class MainActivity extends AppCompatActivity {  
    TextView textView; // TextViewの変数  
    Button button;     // ボタンの変数  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        // ウィジェットの変数とレイアウトのウィジェットを関連付ける  
        textView = findViewById(R.id.tvTest);  
        button = findViewById(R.id.btTest);  
    }  
}
```

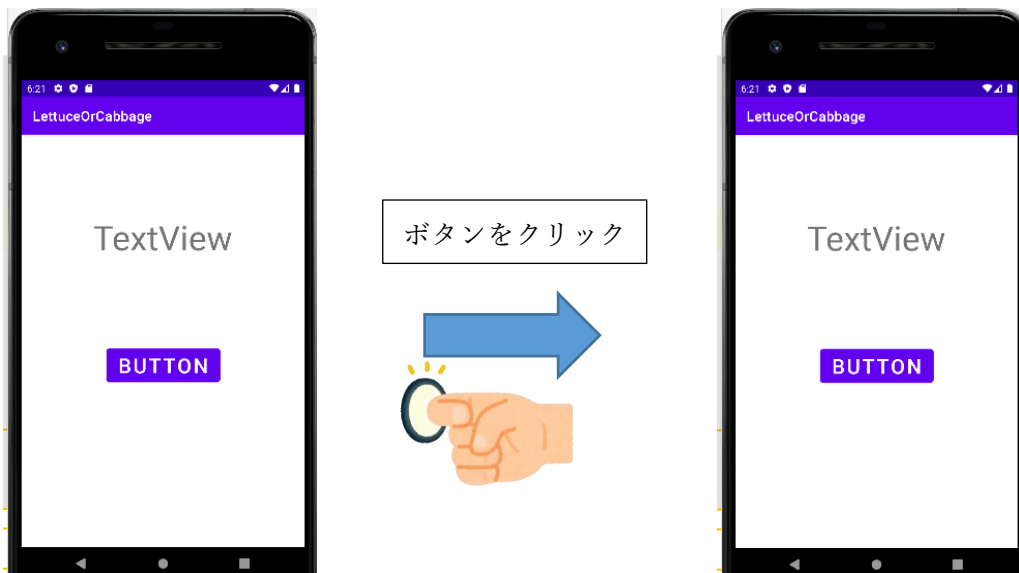
スマートフォンアプリ開発演習

■ イベントハンドラメソッドに処理を記述

今回のイベント処理としては、「**ボタンが押された場合、TextView の表示テキストを書き換える動作**」にします
では、onClick()メソッド内に処理を記述していきます

```
/**
 * ボタンをクリックした時に呼ばれるクラス
 */
private class SelectListener implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        // どのウィジェットが押されたかidを元に分岐
        switch (view.getId()){
            case R.id.btTest:
                // textViewの表示テキストを書き換える
                textView.setText("Hello");
                break;
        }
    }
}
```

この状態で実行してみましょう！



クリックしても何も変わりません、何故なのでしょう？

スマートフォンアプリ開発演習

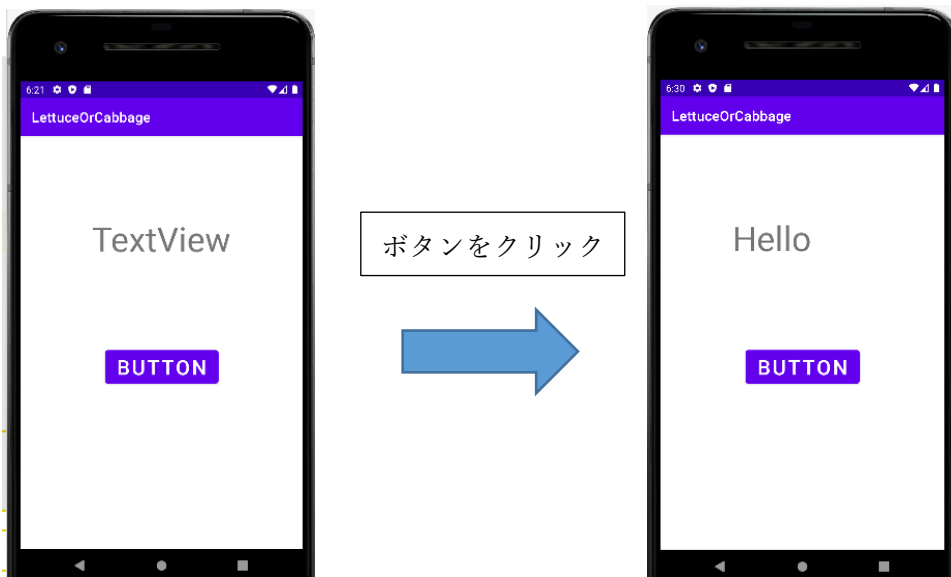
イベントを検知する SelectListener クラスは作成しましたが、
「このボタンのイベントを検知する」という登録を行っていません。

特定のボタンに対してイベント検知の登録を onCreate() メソッド内で行いましょう。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // ウィジェットの変数とレイアウトのウィジェットを関連付ける
    textView = findViewById(R.id.tvTest);
    button = findViewById(R.id.btTest);
    // 特定のボタンに対しての検知登録
    SelectListener selectListener = new SelectListener();
    button.setOnClickListener(selectListener);
}
```



この状態で再度実行してみましょう！



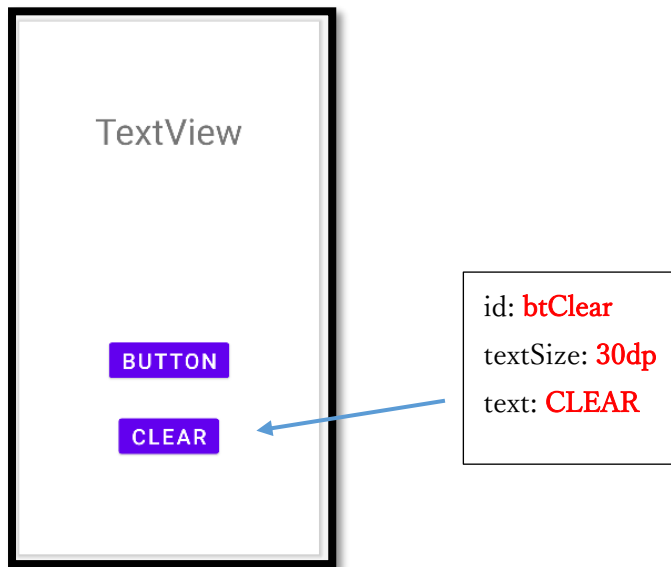
ボタンのクリックイベントを検知し
テキストを書き換える処理が行えました！

スマートフォンアプリ開発演習

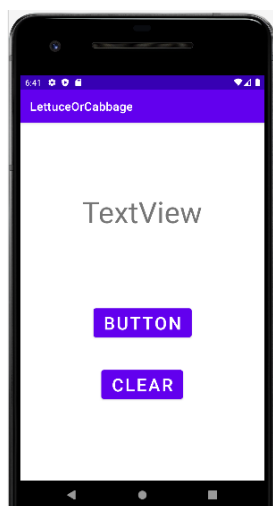
■ ボタンをもう1つ追加してみる

更にボタンを追加し拡張しやすいように練習していきましょう

以下のようにボタンを1つ追加してください。



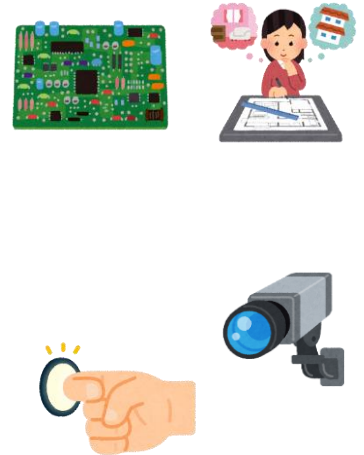
実行し、ボタンが追加された画面になっているか確認しましょう



■追加したボタンの処理を記述する

まずは変数として宣言し、レイアウトとの関連付けも行いましょう。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // ウィジェットの変数とレイアウトのウィジェットを関連付ける
    textView = findViewById(R.id.tvTest);
    button = findViewById(R.id.btTest);
    button2 = findViewById(R.id.btClear);
    // 特定のボタンに対しての検知登録
    SelectListener selectListener = new SelectListener();
    button.setOnClickListener(selectListener);
    button2.setOnClickListener(selectListener);
}
```



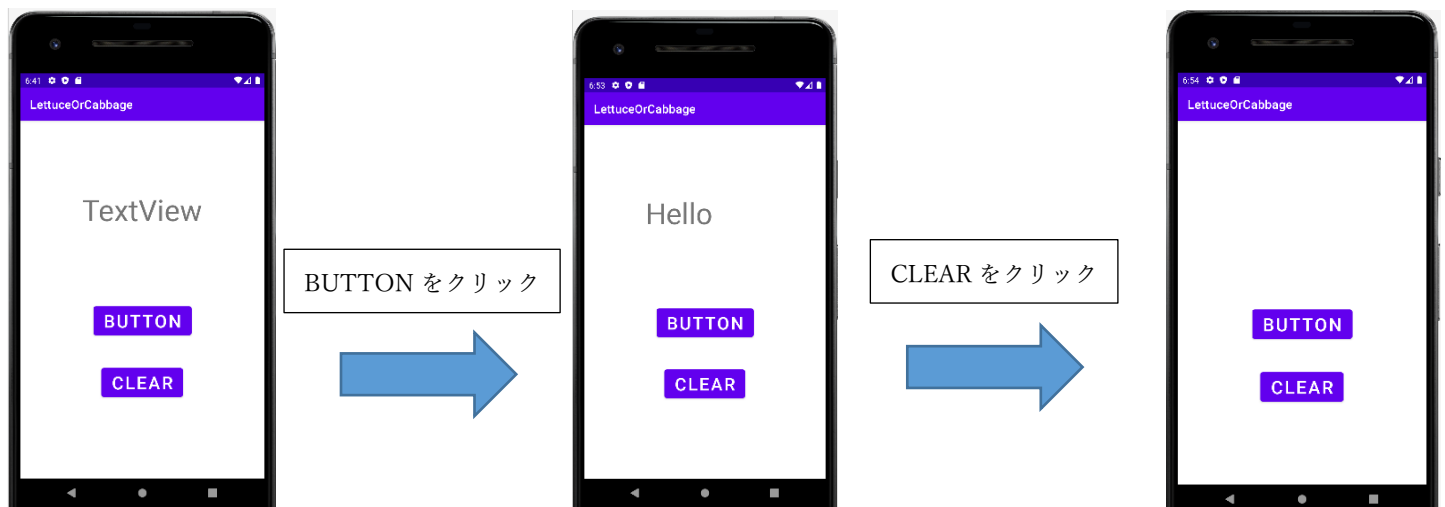
今回はテキストを初期化するようなボタンにしたいと思います。

クリアボタンが押下された場合、“”(空文字)を表示するように記述していきましょう。

```
/**
 * ボタンをクリックした時に呼ばれるクラス
 */
private class SelectListener implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        // どのウィジェットが押されたかidを元に分岐
        switch (view.getId()){
            case R.id.btTest:
                // textViewの表示テキストを書き換える
                textView.setText("Hello");
                break;
            case R.id.btClear:
                // textViewの表示テキストを空文字に書き換える
                textView.setText("");
                break;
        }
    }
}
```

スマートフォンアプリ開発演習

この状態で実行し、動作を確認してみましょう！



ここまでの内容を応用して次はちょっとしたクイズアプリを作ってみましょう！

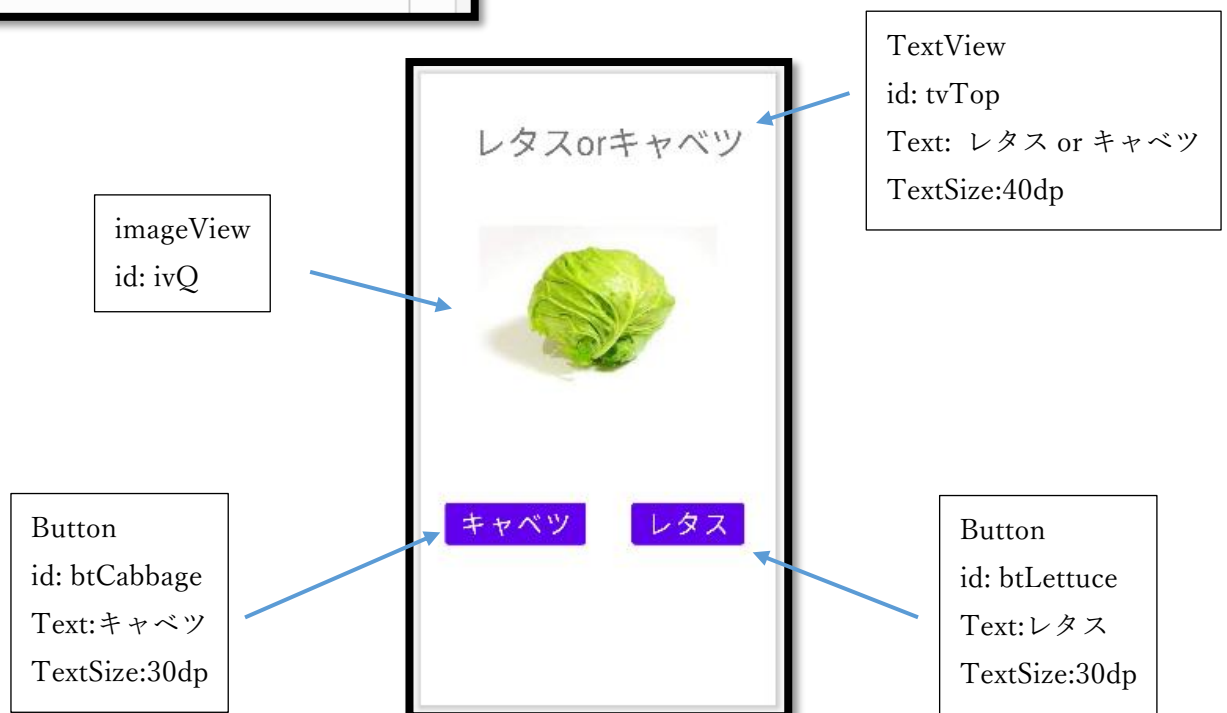
スマートフォンアプリ開発演習

■キャベツ or レタスゲーム

画像がキャベツなのかレタスなのか当てるクイズゲームアプリを作成します。

プロジェクトはそのままに、レイアウトを新規に作成し以下のようなレイアウトを作成してください。

ファイル名: maingame.xml



レイアウトを記述出来たら、setContentView()の内容を書き替えて、ちゃんと表示されているか実行して確認してみましょう。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.maingame);
    // ウィジェットの変数とレイアウトのウィジェットを関連付ける
    textView = findViewById(R.id.tvTest);
}
```

起動すらしなくなりました・・・？

何故・・・？

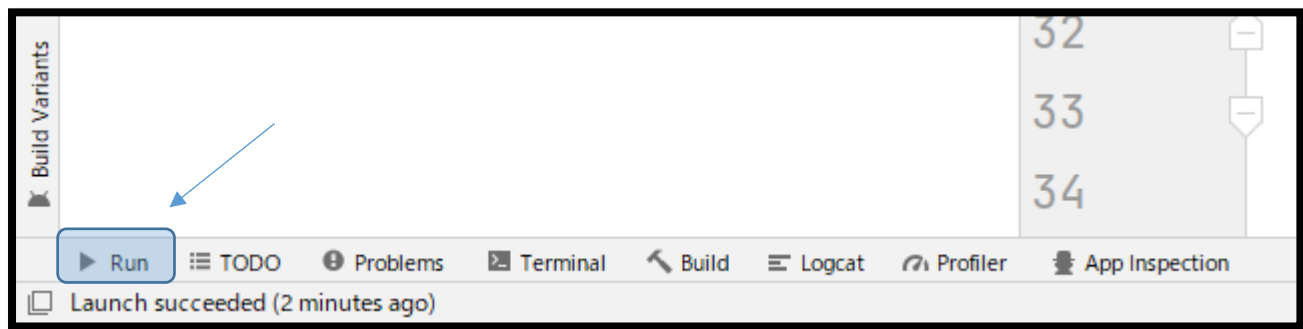
スマートフォンアプリ開発演習

■エラーログでエラー内容を確認しよう

起動しなくなったのはアプリ起動時にエラーが出ている為です。

そのエラーの確認方法から学んでいきましょう。

画面左下にある[Run]のタブをクリックし、実行時のログを確認してみましょう



すると、「どの行で」「どんなエラーが発生した」を確認できます。

```
Caused by: java.lang.NullPointerException: Attempt to invoke virtual method 'void android.widget.Button.  
    at com.exzamp.lettuceorcabbage.MainActivity.onCreate(MainActivity.java:26)  
    at android.app.Activity.performCreate(Activity.java:8000)  
    at android.app.Activity.performCreate(Activity.java:7984)  
    at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1309)  
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3422) <8 more...> <1 internal
```

該当行を確認してみましょう

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.maingame);  
    // ウィジェットの変数とレイアウトのウィジェットを関連付ける  
    textView = findViewById(R.id.tvTest);  
    button = findViewById(R.id.btTest);  
    button2 = findViewById(R.id.btClear);  
    // 特定のボタンに対しての検知登録  
    SelectListener selectListener = new SelectListener();  
    button.setOnClickListener(selectListener);  
    button2.setOnClickListener(selectListener);  
}
```

button 変数の `setOnClickListener` のメソッドを実行しようとするすると `NullPointerException` のエラーになっていました。関連付けているはずなのに `NullPointerException` になるのは変ですね。

本当に button が Null になっているのか、

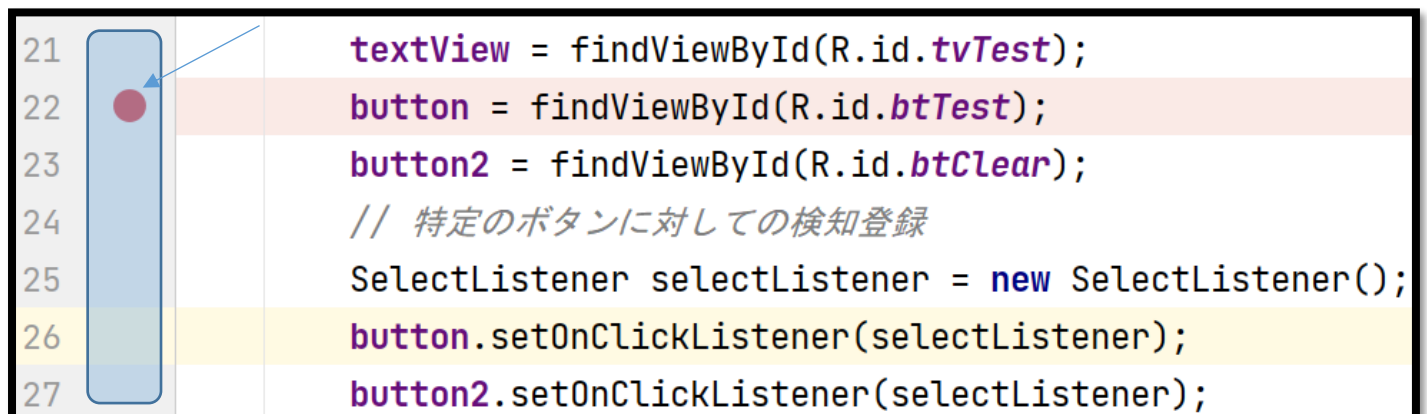
今度は処理の途中で止めて変数の中身を確認するデバッグの方法を学びます。

■ブレークポイントを活用しデバッグ実行

処理を一時停止させたい行に「ブレークポイント」を設定します。

止めたい行の番号横の余白部分をクリックして設定します。

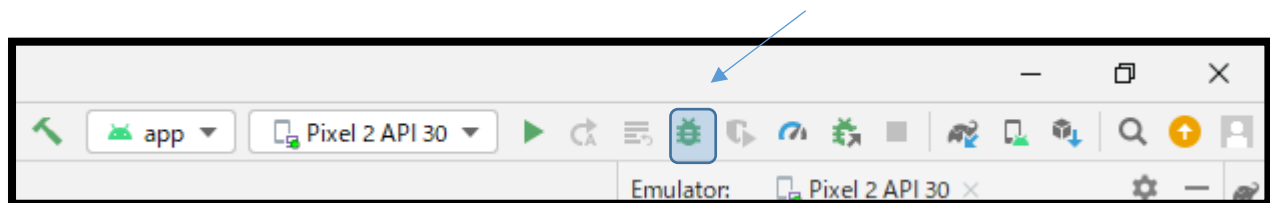
赤い○が表示されれば設定出来ています。



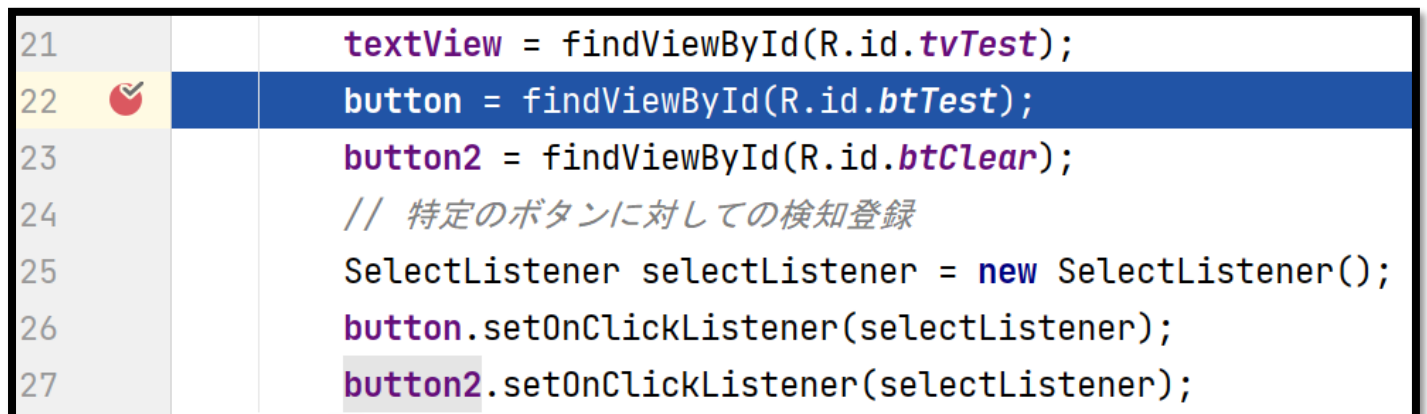
更に通常の実行ではなく、デバッグモードで実行します。

デバッグ実行は虫アイコンが目印です。

ではデバッグ実行してみましょう



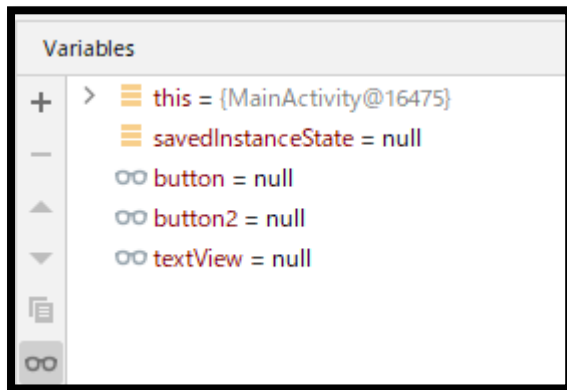
ブレークポイントを設定した行で処理が一時停止しました。



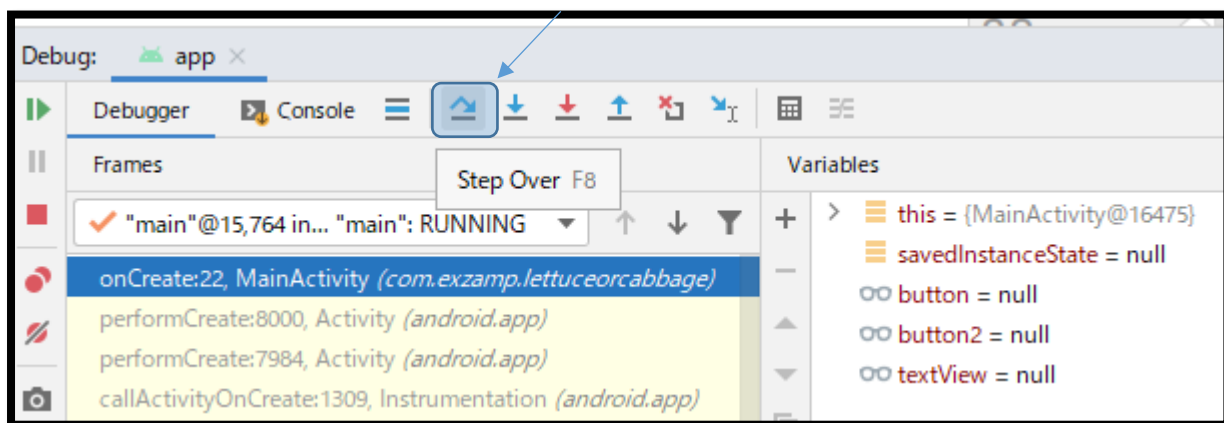
スマートフォンアプリ開発演習

注意としては、この行に入った時点で止まっている為、まだ該当行は処理されていません。

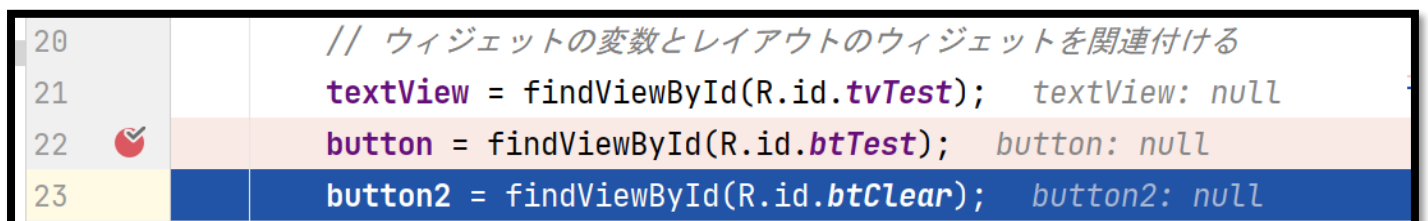
画面下の Variables では、現時点の変数の中身などが確認出来ます。まだ代入前なのでどれも null となっています。



では代入している行の処理を行いたいのので StepOver を使用し 1 行分処理を進めます。



1 行分進みましたが、変数 button は null のままです。よく見ると変数 textView も null のままです。



これは一体どういうことでしょうか・・・？

原因を解明する鍵はこれらの行では何をしているかを再確認すると見えてきます。

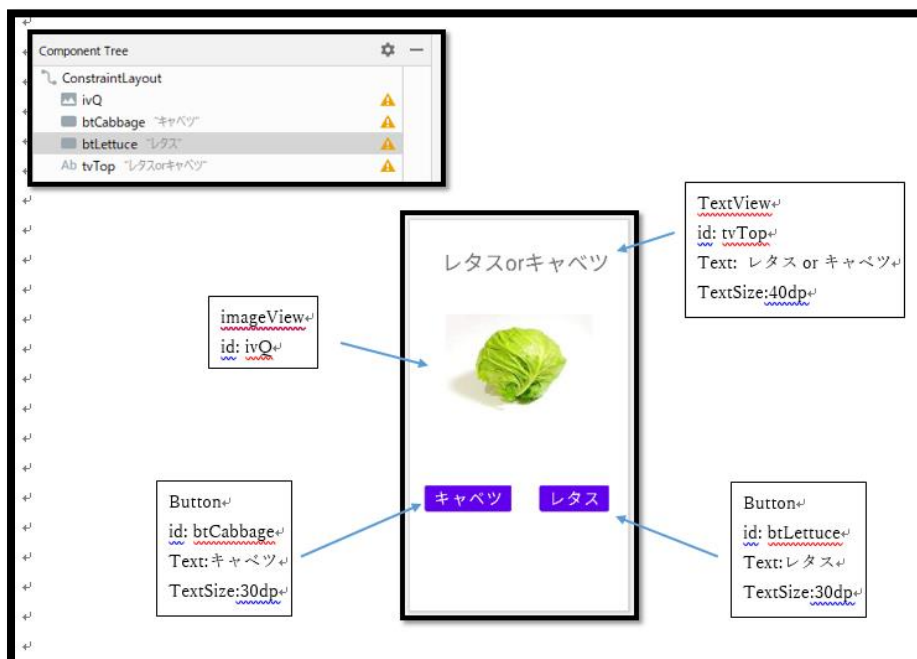
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.maingame);
    // ウィジェットの変数とレイアウトのウィジェットを関連付ける
    textView = findViewById(R.id.tvTest);
    button = findViewById(R.id.btTest);
    button2 = findViewById(R.id.btClear);
}
```

1 つずつ確認していきましょう。

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

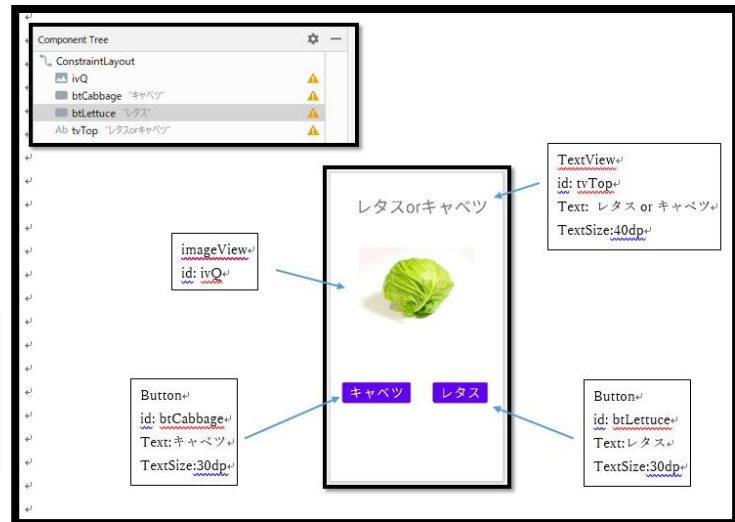
この行で画面のレイアウトを読み込んでいます。

現画面では 4 つのウィジェットがある状態です。



そして次の処理

```
textView = findViewById(R.id.tvTest);
button = findViewById(R.id.btTest);
button2 = findViewById(R.id.btClear);
```



現画面の中に tvTest という id を持った TextView はありません

同様に、「btTest という id の Button」も「btClear という id を持った Button」もありません。

つまり、別画面のウィジェットを読み込もうとしているため、

NullPointerException となっています。

その為、Android では 1 画面に 1Activity を作るのがセオリーとなっています。

■ゲームアプリ用に修正

原因も判明したので、修正を行っていきましょう。

まずは、読み込むウィジェットを maingame.xml の内容に書き換えましょう。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.maingame);
    // ウィジェットの変数とレイアウトのウィジェットを関連付ける
    textView = findViewById(R.id.tvTop);
    button = findViewById(R.id.btCabbage);
    button2 = findViewById(R.id.btLettuce);
    // 特定のボタンに対しての検知登録
    SelectListener selectListener = new SelectListener();
    button.setOnClickListener(selectListener);
    button2.setOnClickListener(selectListener);
}
```


スマートフォンアプリ開発演習

アプリ自体はエラーで落ちなくなりますが、

変数 `button` と変数 `button2` について、

このままでは、どちらがキャベツボタンでどちらがレタスボタンか判別しにくいですね。

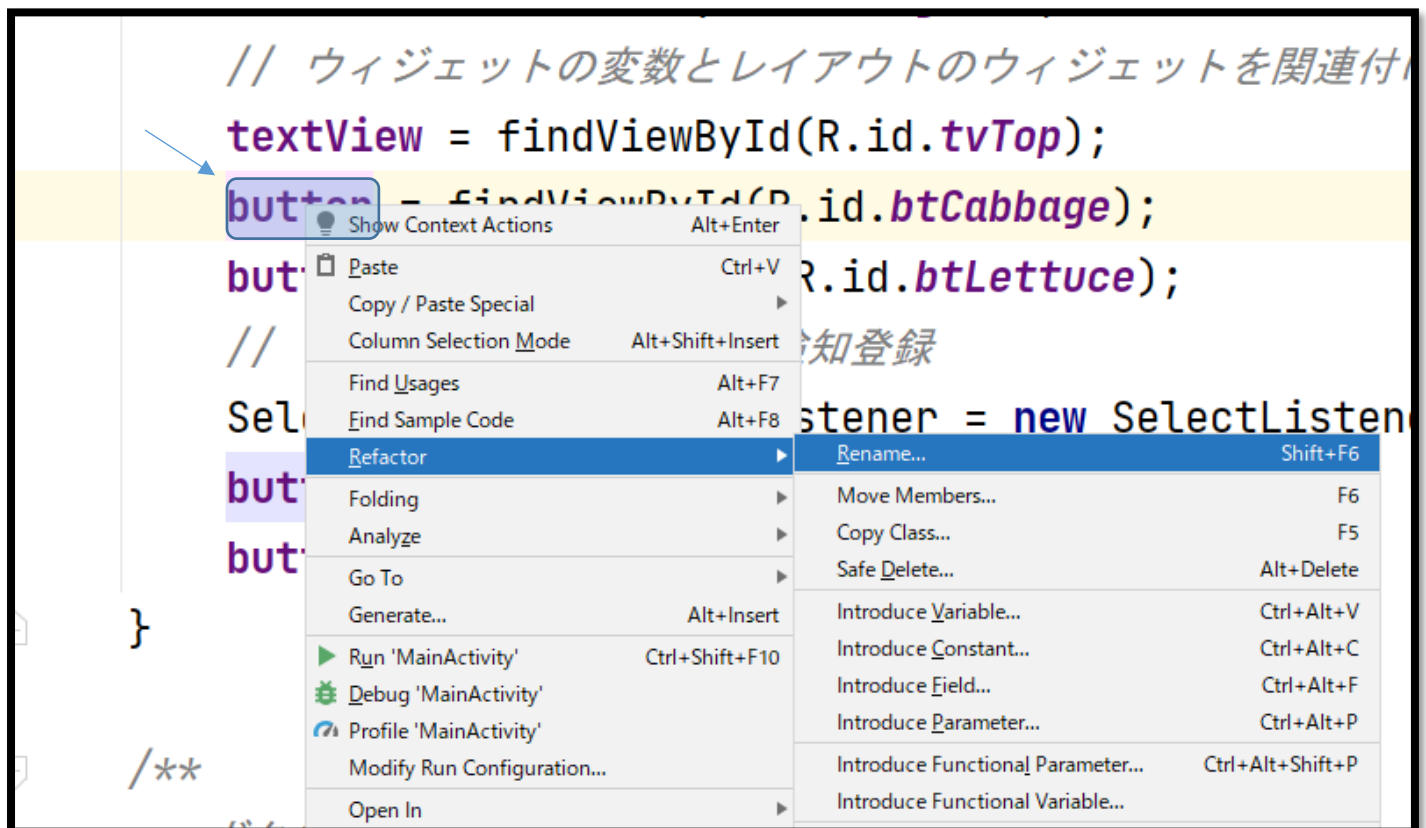
なので、変数名を変更します。

ここでは、複数個所で使われている変数名を一気に変更出来る便利な方法を紹介します。

■Rename で同時編集

変更したい変数を右クリック

Refactor > Rename をクリック



すると変数名を一気に編集できるようになったので `btCabbage` に変更しましょう

```
// ウィジェットの変数とレイアウトのウィジェットを関連付ける
textView = findViewById(R.id.tvTop);
button = findViewById(R.id.btCabbage);
button2 = findViewById(R.id.btLettuce);
// 特定のボタンに対しての検知登録
SelectListener selectListener = new SelectListener();
button.setOnClickListener(selectListener);
button2.setOnClickListener(selectListener);
```



```
// ウィジェットの変数とレイアウトのウィジェットを関連付ける
textView = findViewById(R.id.tvTop);
btCabbage = findViewById(R.id.btCabbage);
button2 = findViewById(R.id.btLettuce);
// 特定のボタンに対しての検知登録
SelectListener selectListener = new SelectListener();
btCabbage.setOnClickListener(selectListener);
button2.setOnClickListener(selectListener);
```

同様に「button2 を btLettuce」に「textView を tvTop」に変更しましょう

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.maingame);
    // ウィジェットの変数とレイアウトのウィジェットを関連付ける
    tvTop = findViewById(R.id.tvTop);
    btCabbage = findViewById(R.id.btCabbage);
    btLettuce = findViewById(R.id.btLettuce);
    // 特定のボタンに対しての検知登録
    SelectListener selectListener = new SelectListener();
    btCabbage.setOnClickListener(selectListener);
    btLettuce.setOnClickListener(selectListener);
}
```

実行してみて問題無く表示されるか確認しておきましょう。



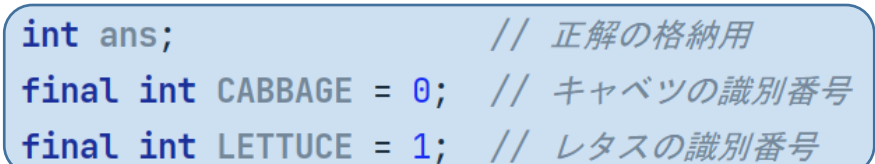
■ ボタン押下時の処理の修正

ボタンイベントについても、前の状態になっているので、ゲームアプリに合わせた内容に修正しましょう

現時点ではキャベツが正解なので、キャベツをクリックした時に正解と表示するようにしたいですが、そのまま固定で実装してしまうと問題の変更や数を増やせないなど汎用性が低くなるので、少し工夫をします。

まずは正解か判定する為の識別用の変数と定数をクラスフィールドに用意しましょう。

```
public class MainActivity extends AppCompatActivity {  
    TextView tvTop;           // 上部のテキスト  
    Button btCabbage;         // キャベツボタン  
    Button btLettuce;         // レタスボタン  
    int ans;                  // 正解の格納用  
    final int CABBAGE = 0;    // キャベツの識別番号  
    final int LETTUCE = 1;    // レタスの識別番号  
}
```



次に暫定で1問目の正解を格納しておきます。

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.maingame);  
    // ウィジェットの変数とレイアウトのウィジェットを関連付ける  
    tvTop = findViewById(R.id.tvTop);  
    btCabbage = findViewById(R.id.btCabbage);  
    btLettuce = findViewById(R.id.btLettuce);  
    // 特定のボタンに対しての検知登録  
    SelectListener selectListener = new SelectListener();  
    btCabbage.setOnClickListener(selectListener);  
    btLettuce.setOnClickListener(selectListener);  
  
    // 1問目の正解を格納  
    ans = CABBAGE;  
}
```

続いて、ボタンが押された時の処理も変更していきましょう。

キャベツボタンとレタスボタンが押された時に分岐出来るように書き換えます。

```
/**
 * ボタンをクリックした時に呼ばれるクラス
 */
private class SelectListener implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        // どのウィジェットが押されたかidを元に分岐
        switch (view.getId()){
            case R.id.btCabbage:
                // textViewの表示テキストを空文字に書き換える
                break;
            case R.id.btLettuce:
                // textViewの表示テキストを空文字に書き換える
                break;
        }
    }
}
```

更に正解している場合は、「正解」

不正解の場合は「不正解」と表示するようにしたいです。

分岐ごとに if 文を作成してもいいのですが、処理が増えると管理が難しくなるため
正解判定の関数を作成してしまいしょう。

スマートフォンアプリ開発演習

まずは正解判定を行い、処理をするメソッドを作成します。

```
/**
 * クイズの正解判定メソッド
 * @param selectBt
 */
private void judge (int selectBt){
    // 正解と回答が一致している場合
    if(ans == selectBt){
        // 正解のテキストに書き換える
        tvTop.setText("正解");
    }else{
        // 不正解のテキストに書き換える
        tvTop.setText("不正解");
    }
}
```

次に作成したメソッドを使用するように処理を追記していきましょう

```
/**
 * ボタンをクリックした時に呼ばれるクラス
 */
private class SelectListener implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        // どのウィジェットが押されたかidを元に分岐
        switch (view.getId()){
            case R.id.btCabbage:
                // textViewの表示テキストを空文字に書き換える
                judge(CABBAGE);
                break;
            case R.id.btLettuce:
                // textViewの表示テキストを空文字に書き換える
                judge(LETTUCE);
                break;
        }
    }
}
```

これで実行してみましょう！

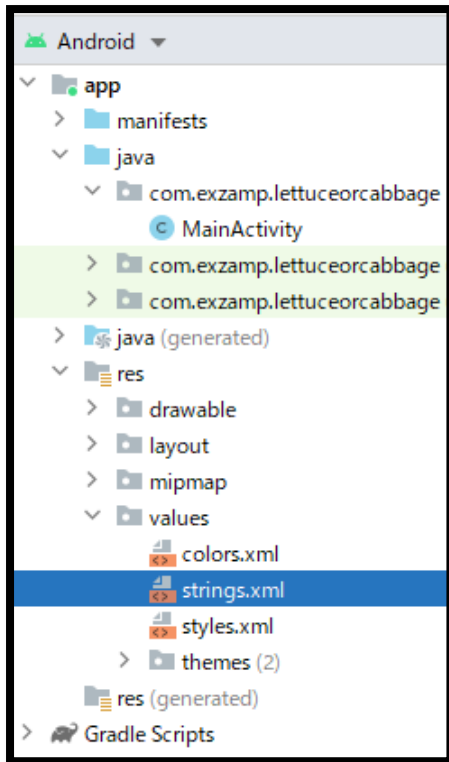


■テキストを string.xml から読み込む

どの言語でも共通していますが、出来るだけ文字列や数値は直接使わない方が良いです。
変数のように使用している箇所全てを変更出来るようにした方が良い為です。
Android では定数以外にも xml ファイルで定義し活用する方法があります。
その方法を学びましょう。

スマートフォンアプリ開発演習

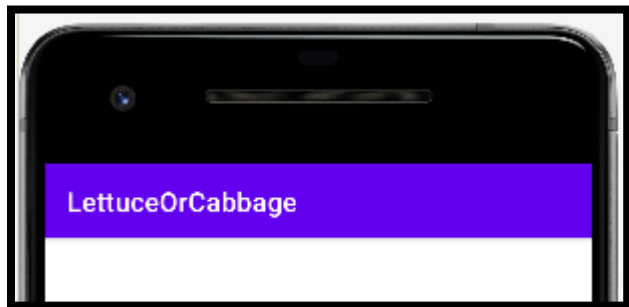
res フォルダ > value フォルダの中に strings.xml というファイルが配備されています。



Strings.xml の中身を確認してみましょう

```
<resources>
    <string name="app_name">LettuceOrCabbage</string>
</resources>
```

見覚えのある文字列が定義されていますね



ここで定義されているのは、

アプリの名前を app_name で定義しています。

どのように正解と不正解の文字列も同じように定義してみましょう。

スマートフォンアプリ開発演習

strings.xml

```
<resources>
    <string name="app_name">LettuceOrCabbage</string>
    <string name="judge_success">正解</string>
    <string name="judge_failure">不正解</string>
</resources>
```

定義が出来たので使用しているプログラムも修正しておきましょう。

```
/**
 * クイズの正解判定メソッド
 * @param selectBt
 */
private void judge (int selectBt){
    // 正解と回答が一致している場合
    if(ans == selectBt){
        // 正解のテキストに書き換える
        tvTop.setText(R.string.judge_success);
    }else{
        // 不正解のテキストに書き換える
        tvTop.setText(R.string.judge_failure);
    }
}
```

再度実行して、問題無く表示されていることを確認しましょう



スマートフォンアプリ開発演習

■ ボタンのグレースアウト

現状のままでは、同じ問題で何度でも回答出来てしまいます。

ボタンを押したら、もう一方は反応しないようにすることが出来ます。

その方法を学んでいきましょう。

有効か無効にするには

ボタン変数.setEnabled(boolean 型の値)

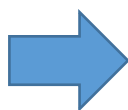
True :有効状態

False:無効状態

上記で変更が可能です。

では実際に処理を追加してみましょう

```
private class SelectListener implements View.OnClickListener {  
    @Override  
    public void onClick(View view) {  
        // どのウィジェットが押されたかidを元に分岐  
        switch (view.getId()){  
            case R.id.btCabbage:  
                // 正解判定  
                judge(CABBAGE);  
                // レタスボタンをグレースアウト  
                btLettuce.setEnabled(false);  
                break;  
            case R.id.btLettuce:  
                // 正解判定  
                judge(LETTUCE);  
                // キャベツボタンをグレースアウト  
                btCabbage.setEnabled(false);  
                break;  
        }  
    }  
}
```



■ImageView の画像の差し替え

次の問題であったり、全て終わった際に画像を変える必要が出てくると思います。

画像も新規で ImageView を定義するのではなく、読み込む画像ファイルをプログラム上で差し替える処理で実現が可能です。

今回は全問終わったことを想定して、おわりの画像に差し替える処理で動きを学んでいきましょう。

差し替える場合は

ImageView 変数. setImageResource(リソース ID)

まずはプログラムで扱えるように ImageView の変数の準備から行いましょう。

```
public class MainActivity extends AppCompatActivity {
    TextView tvTop;           // 上部のテキスト
    Button btCabbage;         // キャベツボタン
    Button btLettuce;         // レタスボタン
    ImageView ivQ;            // 問題画像表示用
    int ans;                  // 正解の格納用
    final int CABBAGE = 0;    // キャベツの識別番号
    final int LETTUCE = 1;    // レタスの識別番号

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

続いて、レイアウトとの関連付けも行います。

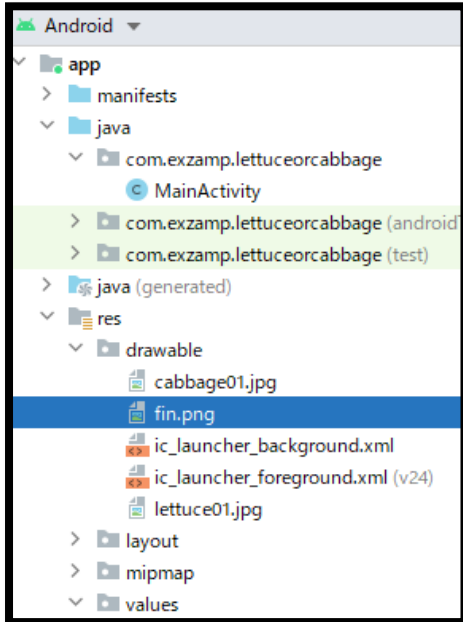
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.maingame);
    // ウィジェットの変数とレイアウトのウィジェットを関連付ける
    tvTop = findViewById(R.id.tvTop);
    btCabbage = findViewById(R.id.btCabbage);
    btLettuce = findViewById(R.id.btLettuce);
    ivQ = findViewById(R.id.ivQ);
    // 特定のボタンに対しての検知登録
    SelectListener selectListener = new SelectListener();
    btCabbage.setOnClickListener(selectListener);
    btLettuce.setOnClickListener(selectListener);

    // 1問目の正解を格納
    ans = CABBAGE;
}
```

スマートフォンアプリ開発演習

現時点では1問しかないので1問が終わったタイミングで
画像を差し替える処理を追加します。

fin.png ファイルを res > drawable 内に配置しておいてください



問題の回答後に画像を差し替える処理を追加します。

```
@Override
public void onClick(View view) {
    // どのウィジェットが押されたかidを元に分岐
    switch (view.getId()){
        case R.id.btCabbage:
            // 正解判定
            judge(CABBAGE);
            // レタスボタンをグレーアウト
            btLettuce.setEnabled(false);
            break;
        case R.id.btLettuce:
            // 正解判定
            judge(LETTUCE);
            // キャベツボタンをグレーアウト
            btCabbage.setEnabled(false);
            break;
    }
    // 全問終了の画像に差し替え
    ivQ.setImageResource(R.drawable.fin);
}
```

スマートフォンアプリ開発演習

実行し、動作を確認してみましょう。

