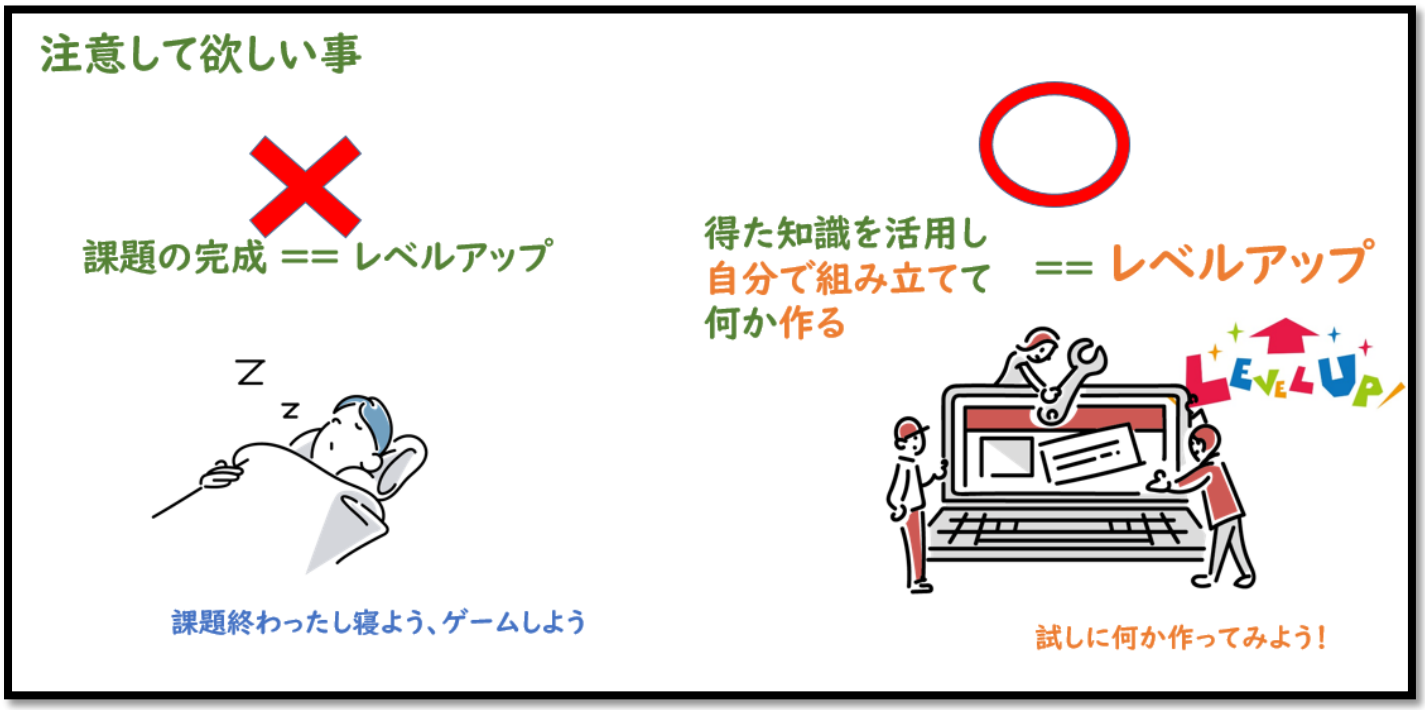
本日の内容

・SoundPool
・画面の向きの固定

公式ドキュメント
https://developer.android.com/reference/android/media/SoundPool

今回は音(SE)を実装する方法を学んでいきます。



■SoundPool とは
音データを再生するクラスの 1 つ
他には MediaPlayer , AudioTrack などがある
・5 秒程度のデータを再生出来るので SE の再生に向いている
mp3,ogg,wav などの拡張子を再生可能

スマートフォンアプリ開発演習

■プロジェクト作成

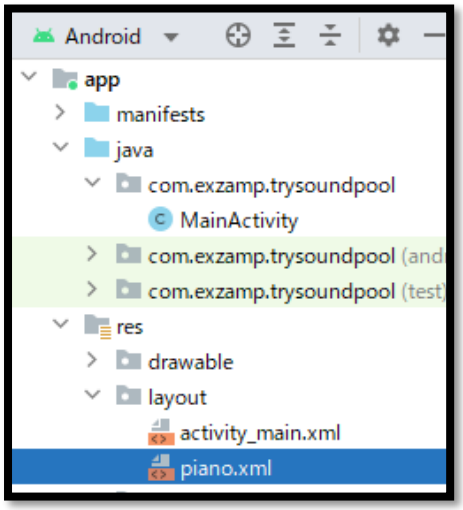
- Empty Activity
- プロジェクト名：TrySoundPool

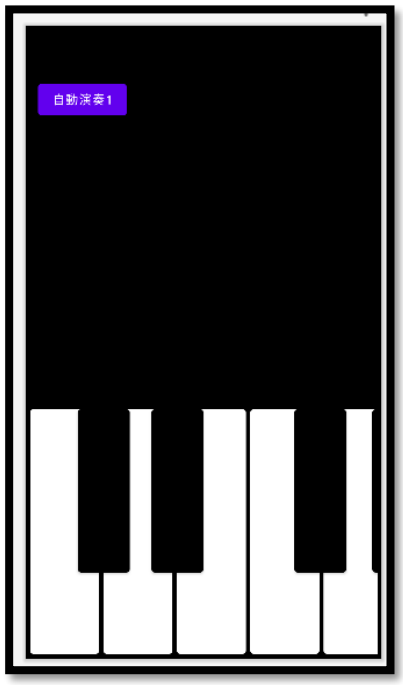
==========

■xml ファイルの配置

今回使用する xml は 1 から作成すると難易度が高い為、今回は授業用に準備した xml を活用します。
自分でカスタマイズや自作する場合はタグなど意味を読み解いて調べて試しご自身の技術に変換してください。

授業フォルダ内の piano.xml を res>layout フォルダ内にコピーしましょう。





■画面を横に固定

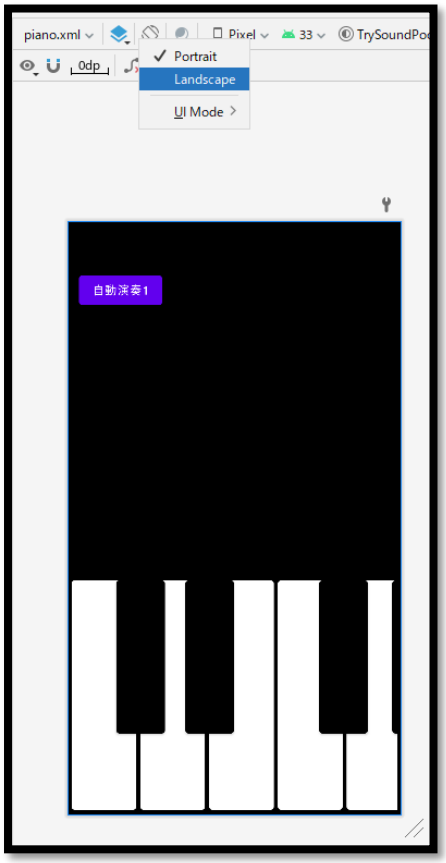
ピアノの画面は用意できましたが縦画面だと使い難そうです。
Android アプリでは 1 画面ごとに縦/横画面で
固定する事が可能です。
次ではその設定を行っていきます。

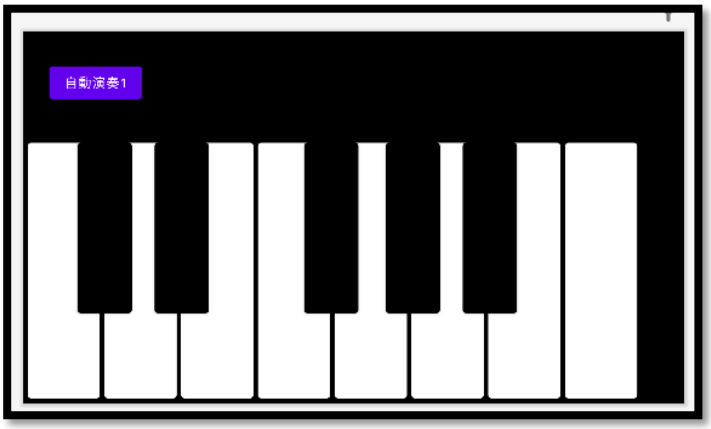
AndroidManifest.xml

```
<activity
    android:name=".MainActivity"
    android:screenOrientation="landscape"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.
        <category android:name="android.inten
    </intent-filter>
```

MainActivity に対して、画面を横(landscape)で固定する設定です
他にも、縦固定や全画面表示などもあります。(必要になった際に調べてみましょう)

しかし、作業上では縦のままで見にくいので、作業上も横に変更しておきましょう。





スマートフォンアプリ開発演習

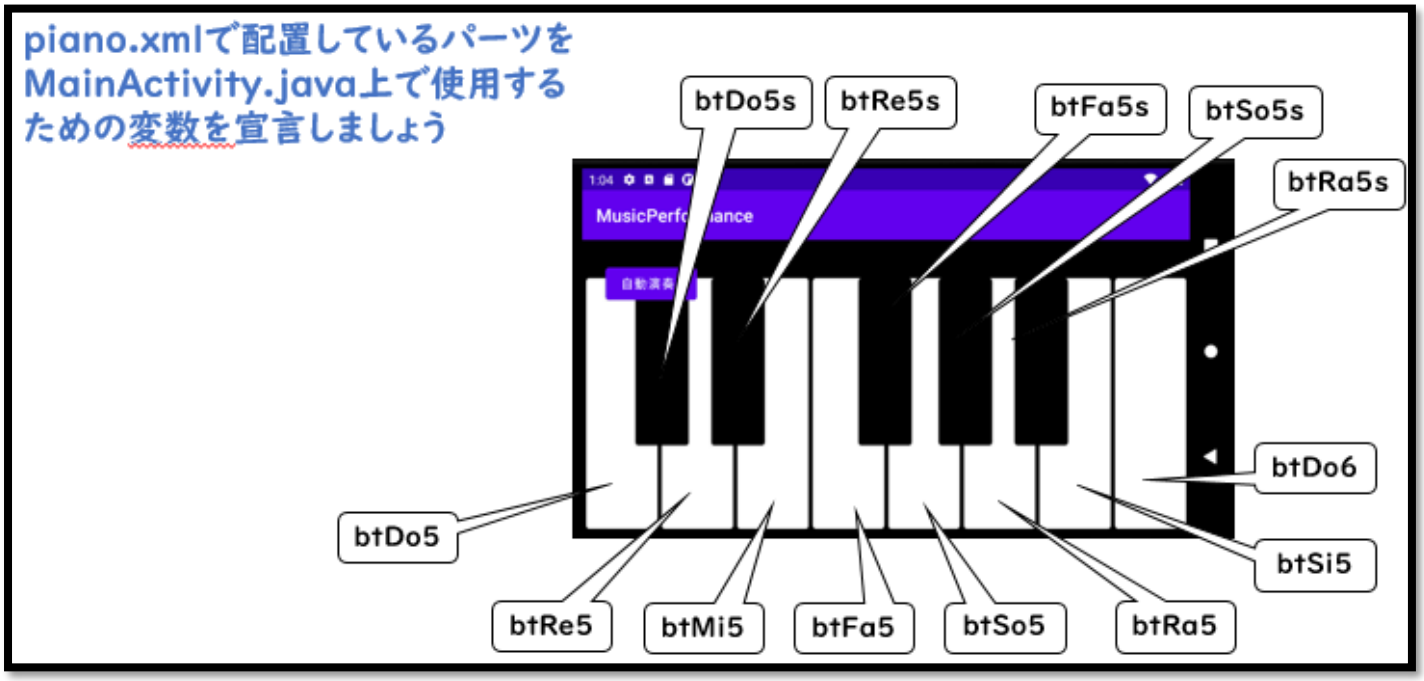
■読み込む xml ファイルを指定
そのまま実行しても表示されるのは Hello wold のレイアウトの activity_main.xml になります。
先程コピーした piano.xml で読み込むためには指定する必要があります。

MainActivity.java

```java
public class MainActivity extends AppCompatActivity {

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

■レイアウトと変数の関連付け



先に各ボタンの id を配列でまとめておきます。

```
public class MainActivity extends AppCompatActivity {
    // 各鍵盤ボタンのidを格納した配列
    private int btIds[] = {
            R.id.btDo5,R.id.btDo5s,R.id.btRe5,
            R.id.btRe5s,R.id.btMi5,R.id.btFa5,
            R.id.btFa5s,R.id.btSo5,R.id.btSo5s,
            R.id.btRa5,R.id.btRa5s,
            R.id.btSi5,R.id.btDo6
    };

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

次に各ボタンの変数を配列でまとめて定義してしましましょう

```
public class MainActivity extends AppCompatActivity {
    // 各鍵盤ボタンのidを格納した配列
    private int btIds[] = {
            R.id.btDo5,R.id.btDo5s,R.id.btRe5,
            R.id.btRe5s,R.id.btMi5,R.id.btFa5,
            R.id.btFa5s,R.id.btSo5,R.id.btSo5s,
            R.id.btRa5,R.id.btRa5s,
            R.id.btSi5,R.id.btDo6
    };
    // 各鍵盤ボタンの変数を定義
    private Button kenban[] = new Button[btIds.length];

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

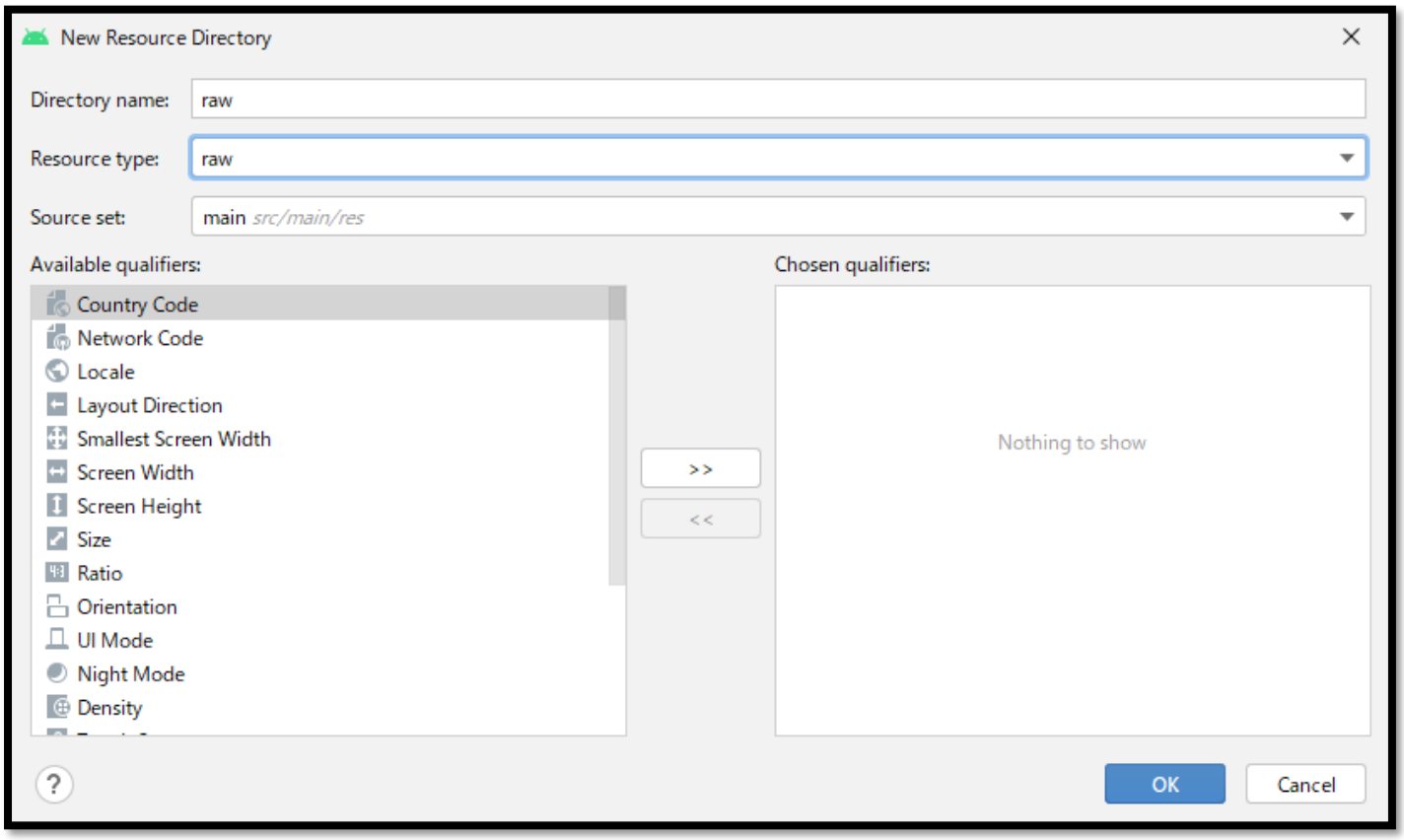
続いて関連付けも行いますが、配列で定義しているので for 文でまとめて行うことも可能です

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.piano);
    for(int i=0; i<kenban.length;i++){
        // レイアウトと関連付け
        kenban[i] = findViewById(btIds[i]);
    }
}
```

■音ファイルの配置
今回は効果音の再生して演奏を実現させるので音ファイルが必要です。
授業フォルダから音ファイルをコピーしプロジェクトへ移しましょう

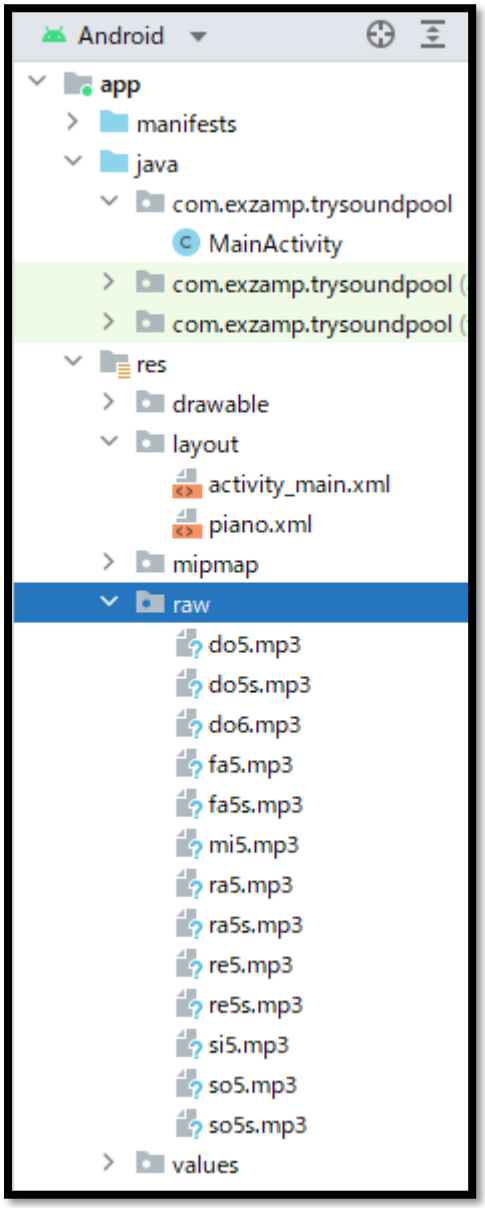
音のファイルは専用のリソースフォルダに配置します。
専用のリソースフォルダ raw を作成しましょう。
res>右クリック>new>Android Resouce Directory

Resouce type を raw>OK



スマートフォンアプリ開発演習

作成した raw フォルダに音ファイルを全てコピーしましょう。



■音を再生するクラスの作成
同じ MainActivity 内に作成しても良いのですが管理が行いやすいように
別に Piano クラスを作成し MainActivity 内で活用しましょう。

New>Java Class
クラス名：Piano

```
public class Piano {


}
```

スマートフォンアプリ開発演習

音を鳴らすために SoundPool の変数 soundPool を定義

```
public class Piano {
    private SoundPool soundPool;     // SEを鳴らす用
}
```

再生する音ファイルも管理しやすいように配列で定義しておきましょう

```
public class Piano {
    private SoundPool soundPool;     // SEを鳴らす用
    // 音ファイルの配列
    private int soundFiles[] = {
            R.raw.do5, R.raw.do5s, R.raw.re5,
            R.raw.re5s, R.raw.mi5, R.raw.fa5,
            R.raw.fa5s, R.raw.so5, R.raw.so5s,
            R.raw.ra5, R.raw.ra5s, R.raw.si5,
            R.raw.do6
    };
}
```

Piano インスタンスの初期設定を行えるようコンストラクタの Piano メソッドを定義

```
public class Piano {
    private SoundPool soundPool;     // SEを鳴らす用
    // 音ファイルの配列
    private int soundFiles[] = {
            R.raw.do5, R.raw.do5s, R.raw.re5,
            R.raw.re5s, R.raw.mi5, R.raw.fa5,
            R.raw.fa5s, R.raw.so5, R.raw.so5s,
            R.raw.ra5, R.raw.ra5s, R.raw.si5,
            R.raw.do6
    };

    // コンストラクタ
    public Piano(Context context){

    }
}
```

スマートフォンアプリ開発演習

インスタンス生成する処理を追記

SoundPool(int maxStreams, int streamType, int srcQuality)

maxStreams：プールする最大の数

streamType：Stream のタイプ（通常は STREAM_MUSIC を利用する）

srcQuality：サンプリングレートのクオリティ（デフォルトは 0）

```
// コンストラクタ
public Piano(Context context){
    // SoundPoolのインスタンスを生成
    soundPool = new SoundPool(soundFiles.length, AudioManager.STREAM_MUSIC, srcQuality: 0);
}
```

音ファイルの情報と関連付ける

構文

soundPool 変数.load(context , 音ファイル , 優先度)

```
// コンストラクタ
public Piano(Context context){
    // SoundPoolのインスタンスを生成
    soundPool = new SoundPool(soundFiles.length, AudioManager.STREAM_MUSIC, srcQuality: 0);
    for(int i=0; i<soundFiles.length;i++){
        // 音を読み込ませていく
        soundPool.load(context,soundFiles[i], priority: 1 );
    }
}
```

指定した音ファイルを再生するメソッド play を定義

```
// コンストラクタ
public Piano(Context context){
    // SoundPoolのインスタンスを生成
    soundPool = new SoundPool(soundFiles.length, AudioManager.STREAM_MUSIC, srcQuality: 0)
    for(int i=0; i<soundFiles.length;i++){
        // 音を読み込ませていく
        soundPool.load(context,soundFiles[i], priority: 1 );
    }
}
// 音を鳴らす為のメソッド
public void play(int soundId){

}
```

引数の値によって分岐できるよう処理を追記
引数には何番目の鍵盤が押されたかを受け取る(配列は 0 番目からだが、play は 1 から添え字が始まる)

構文

soundPool 変数.play(何番目の音を再生するか , 左音量 , 右音量 , 優先度 , ループ回数 , 速度)

```
// 音を鳴らす為のメソッド
public void play(int soundId){
    soundPool.play( soundID: soundId+1, leftVolume: 1.0f, rightVolume: 1.0f, priority: 0, loop: 0, rate: 1.0f);
}
```

これでピアノ音を再生する Piano クラスが完成しました。
あとは MainActivity に Piano クラスを使用する処理を記述すれば演奏可能です。

■クリックリスナーのインターフェイスを実装
implements View.OnClickListener を追記

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
    // 各鍵盤ボタンのidを格納した配列
    private int btIds[] = {
            R.id.btDo5,R.id.btDo5s,R.id.btRe5,
            R.id.btRe5s,R.id.btMi5,R.id.btFa5,
            R.id.btFa5s,R.id.btSo5,R.id.btSo5s,
            R.id.btRa5,R.id.btRa5s,
            R.id.btSi5,R.id.btDo6
```

onClick メソッドを実装

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.piano);
    for(int i=0; i<kenban.length;i++){
        // レイアウトと関連付け
        kenban[i] = findViewById(btIds[i]);
    }
}
@Override
public void onClick(View view) {

}
```

このままでは、ボタンクリックの検知が出来ませんのでリスナーを登録しましょう

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.piano);
    for(int i=0; i<kenban.length;i++){
        // レイアウトと関連付け
        kenban[i] = findViewById(btIds[i]);
        //  リスナーを登録
        kenban[i].setOnClickListener(this);
    }
}
```

■Piano クラスを使用して音を鳴らす

クリックイベント処理の準備も出来ましたので音を鳴らす処理を追記していきましょう

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
    // 各鍵盤ボタンのidを格納した配列
    private int btIds[] = {
            R.id.btDo5,R.id.btDo5s,R.id.btRe5,
            R.id.btRe5s,R.id.btMi5,R.id.btFa5,
            R.id.btFa5s,R.id.btSo5,R.id.btSo5s,
            R.id.btRa5,R.id.btRa5s,
            R.id.btSi5,R.id.btDo6
    };
    // 各鍵盤ボタンの変数を定義
    private Button kenban[] = new Button[btIds.length];

    Piano mPiano;   // ピアノクラス
```

スマートフォンアプリ開発演習

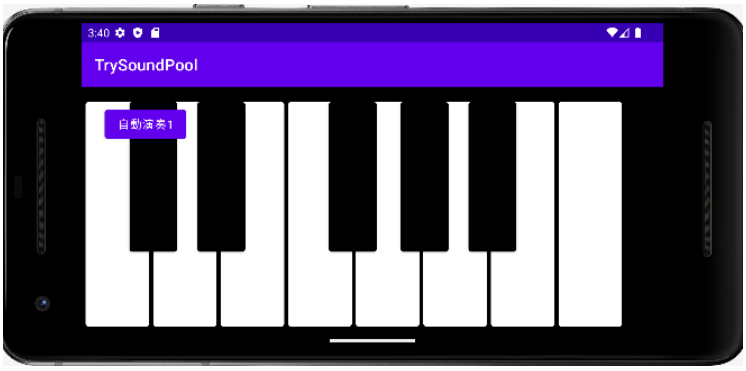
Piano クラスのインスタンスを生成

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.piano);
    for(int i=0; i<kenban.length;i++){
        // レイアウトと関連付け
        kenban[i] = findViewById(btIds[i]);
        //  リスナーを登録
        kenban[i].setOnClickListener(this);
    }
    // インスタンス生成
    mPiano = new Piano(getApplicationContext());
}
```

何番目の鍵盤が押されたを特定し鳴らす音を指定する

```
@Override
public void onClick(View view) {
    for (int i = 0; i < btIds.length; i++) {
        // 何番目の鍵盤が押されたのかチェック
        if (view.getId() == btIds[i]) {
            // 押された鍵盤の音を鳴らす
            mPiano.play(i);
        }
    }
}
```

音が鳴るか確認してみましょう！



スマートフォンアプリ開発演習

■自動演奏
自動演奏を押下でメロディが流れるように追記してみましょう！

ボタンの変数定義と関連付け、リスナー登録を行います。

```
public class MainActivity extends AppCompatActivity implements
    // 各鍵盤ボタンのidを格納した配列
    private int btIds[] = {
            R.id.btDo5,R.id.btDo5s,R.id.btRe5,
            R.id.btRe5s,R.id.btMi5,R.id.btFa5,
            R.id.btFa5s,R.id.btSo5,R.id.btSo5s,
            R.id.btRa5,R.id.btRa5s,
            R.id.btSi5,R.id.btDo6
    };
    // 各鍵盤ボタンの変数を定義
    private Button kenban[] = new Button[btIds.length];

    Piano mPiano;   // ピアノクラス
    private Button btMelody;    // 自動演奏ボタン
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.piano);
    for(int i=0; i<kenban.length;i++){
        // レイアウトと関連付け
        kenban[i] = findViewById(btIds[i]);
        //  リスナーを登録
        kenban[i].setOnClickListener(this);
    }
    // インスタンス生成
    mPiano = new Piano(getApplicationContext());
    // 関連付け
    btMelody= findViewById(R.id.btAutoMusic1);
    // リスナー登録
    btMelody.setOnClickListener(this);
}
```

Piano クラスにメロディとして音を鳴らすメソッドを定義

スマートフォンアプリ開発演習

Piano.java

```
public void playMelody(){
    // 鳴らす音の配列
    int melody[] = {5,5,5,5,3,7,5};
    try {
        for (int i = 0; i < melody.length; i++) {
            play(melody[i]);
            // 100ミリ秒待機させて音の長さを調整
            Thread.sleep( millis: 100);
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

自動演奏ボタンの時と処理を分岐

MainActivity.java

```
@Override
public void onClick(View view) {
    if(view.getId() == R.id.btAutoMusic1) {
        mPiano.playMelody();
    }else {
        for (int i = 0; i < btIds.length; i++) {
            // 何番目の鍵盤が押されたのかチェック
            if (view.getId() == btIds[i]) {
                // 押された鍵盤の音を鳴らす
                mPiano.play(i);
            }
        }
    }
}
```

メロディが鳴るか確認してみましょう！

スマートフォンアプリ開発演習