# 技術者なら知っておきたい！Androidアプリ開発

# 講座の流れ

1. 基本概念と基本構造
2. Activity とイベントハンドリング
3. SharedPreferences とデータ永続化
4. Fragment と画面間通信
5. 簡単なアプリ作成体験

# ◆Activity(アクティビティ)

- ユーザー操作を処理するためのUI（画面）
- アクティビティのライフサイクル

※詳細は、公式ドキュメント（アクティビティのライフサイクル）参照

https://developer.android.com/guide/components/activities/activity-lifecycle?hl=ja

# ◆UI(ユーザーインターフェース)

- アクティビティー内で利用するためのUI（部品）
  - ViewやFragmentでユーザー操作をつなぐ
  - XMLファイルで画面構成を定義して見た目を作る

※他に Intent、Service、Broadcast-Receiver、Content-Provider、などがある

**基本概念**

# ◆プロジェクト構造

```
├ build.gradle                    # ビルド設定（モジュール）
├ app
   ├ build.gradle                 # ビルド設定（プロジェクト）
   ├ src
      ├ main
         ├ java                   # ソースファイル（Javaファイル）
         ├ res                    # リソースファイル（XMLファイル、画像ファイル）
         ├ AndroidManifest.xml    # アプリ設定ファイル
```

# ◆res/ ディレクトリ

- アプリリソースを管理する
  - ➢ layout/　　：レイアウト用のXMLファイル
  - ➢ drawable/　：アイコン、背景、などのシェープ
    - ※ mipmap/ を使うこともある
  - ➢ values/　　：文字列、色、などのスタイル

**基本構造**

# 基本構造

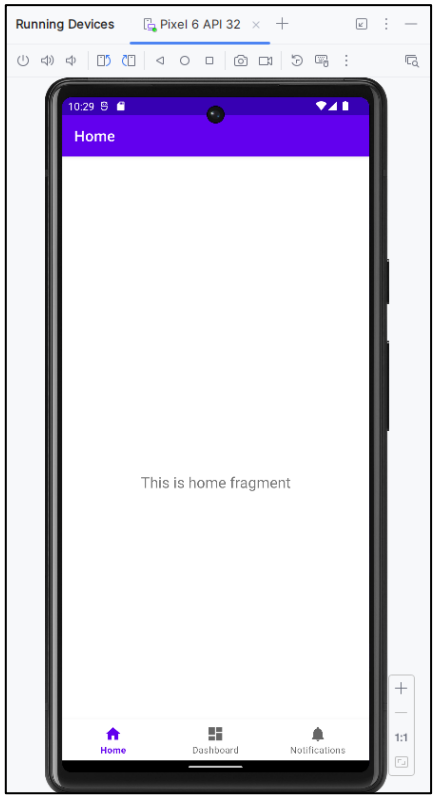
## ◆AndroidManifest.xml

- アプリケーション設定を記述する
  - ➢アプリ内で利用する基本概念の実現方式
    アクティビティ、サービス、インテント、など
  - ➢アプリ内で利用する依存機能へのパーミッション
    インターネット許可、カメラ許可、など

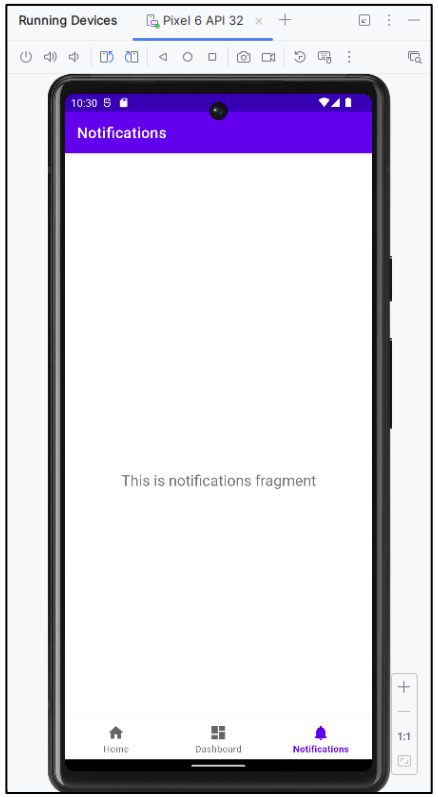
## ◆build.gradle (プロジェクト)

- プロジェクトやアプリのビルド設定を記述する
  - ➢ビルド定義
    Javaバージョン、Androidバージョン、マニフェスト、
    デバッグビルド関係、署名関係、など
  - ➢ビルド依存関係
    依存ライブラリ、ライブラリ取得先リポジトリ、など

# Ex01（Bottom Navigation Views）








**基本概念と基本構造**

# ◆Activityとイベントハンドリング

- UI要素（ボタンなど）に対してイベントリスナーを設定します。これにより、ユーザーが特定の操作を行ったときに、プログラムを実行できます。

# ◆イベントリスナー

- イベントを監視するためのオブジェクト
  - クリックイベント（View.OnClickListener）など

※詳細は、公式ドキュメント（入力イベントの概要）参照
https://developer.android.com/develop/ui/views/touch-and-input/input-events?hl=ja

**イベントハンドリング**

# ◆イベントハンドリング

- UI要素（ボタンなど）のオブジェクトを取得
  - ➢レイアウトID（android:id）を利用

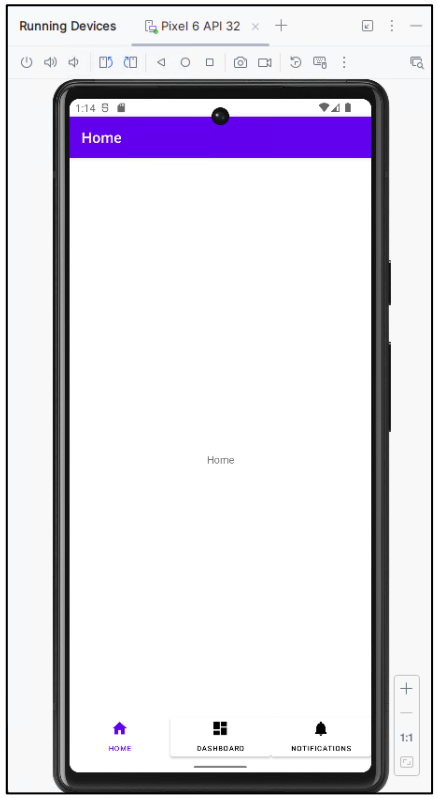
```
final ActionBar actionBar = getSupportActionBar();
final Button button1 = ((Button) findViewById(R.id.button1));
final Button button2 = ((Button) findViewById(R.id.button2));
final Button button3 = ((Button) findViewById(R.id.button3));
final TextView textView1 = ((TextView) findViewById(R.id.textView1));
```

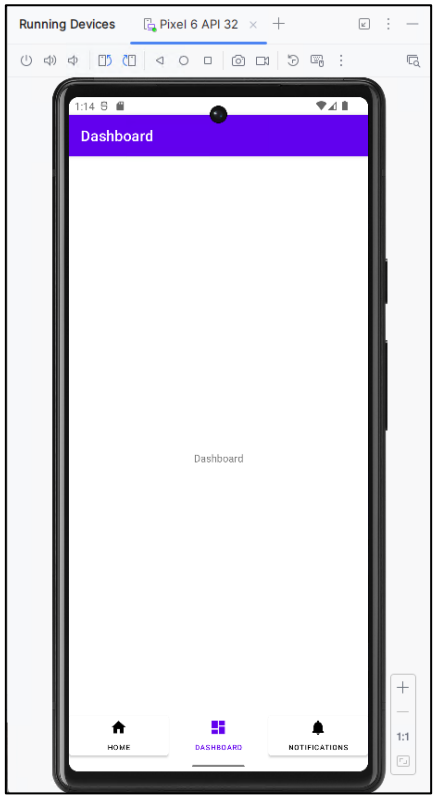
- イベントハンドリングにリスナーを設定

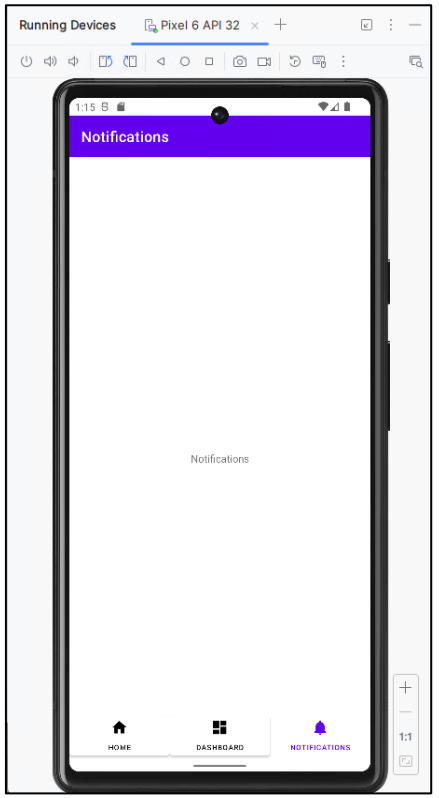
```
button1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Toast.makeText(MainActivity.this,"button1 clicked",Toast.LENGTH_SHORT).show();
        button1.setEnabled(false);
        button2.setEnabled(true);
        button3.setEnabled(true);
        textView1.setText(button1.getText());
        actionBar.setTitle(button1.getText());
    }
});
```

**イベントハンドリング**

# Ex02（Empty Views）




Activity とイベントハンドリング

# ◆SharedPreferencesの概要

- SharedPreferencesは、キー・バリューのペアで扱えるデータをOSのアプリ専用領域に格納
- 格納時のファイル名は指定可能

（例）getSharedPreferences(ファイル名, モード)

※本研修では指定しません

# ◆SharedPreferencesの安全性

- 他アプリケーションからアクセスできないように「Context.MODE_PRIVATE」での利用が一般的
- ファイル内容が暗号化されないため保存内容を難読化して運用するなど検討が必要
  - EncryptedSharedPreferencesクラス利用など

**データ永続化**

# ◆SharedPreferencesの利用

- Activityから操作するのが基本
  - ➢データ取得

```
SharedPreferences sp = this.getPreferences(Context.MODE_PRIVATE);
String userId = sp.getString(USER_ID, STRING_EMPTY);
```

  - ➢データ登録

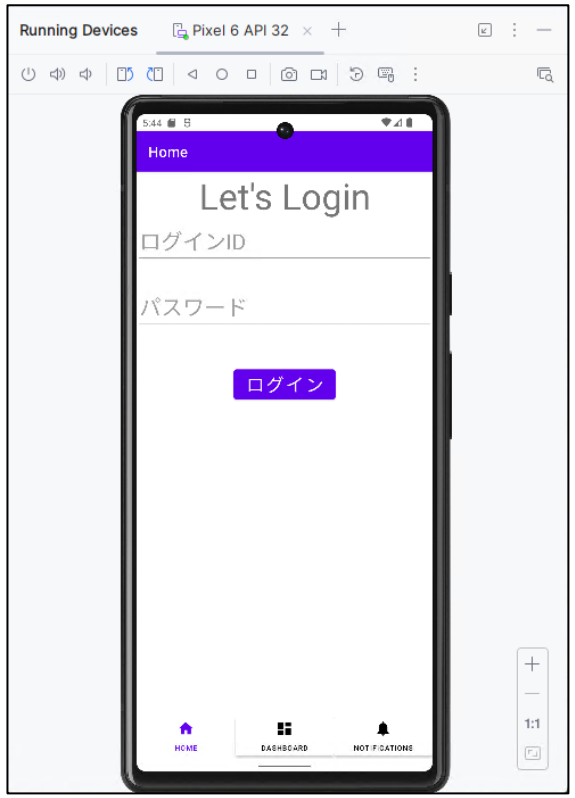
```
SharedPreferences sp = this.getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sp.edit();
editor.putString(USER_ID, "SAMPLE");
editor.commit();
```

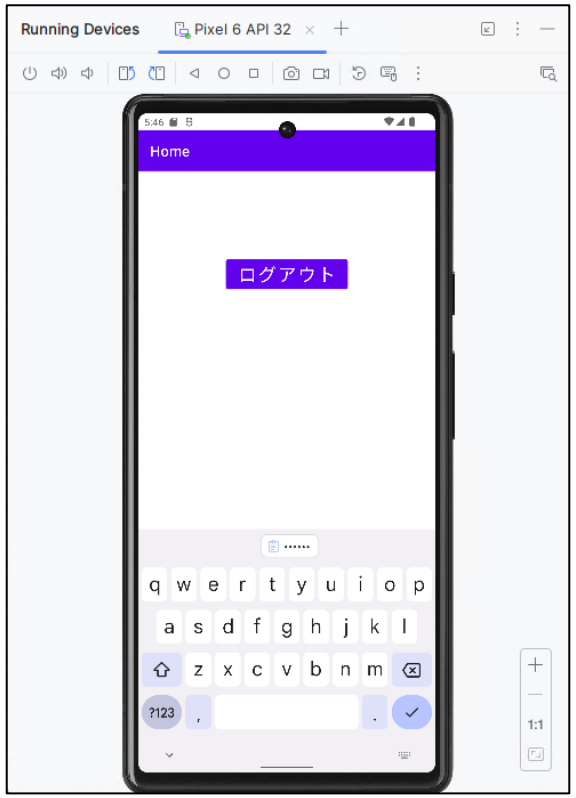
  - ➢データ削除

```
SharedPreferences sp = this.getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sp.edit();
editor.remove(USER_ID);
editor.commit();
```

**データ永続化**

# Ex03（Ex02を改造）




## SharedPreferences とデータ永続化

# ◆Fragment(フラグメント)

画面の一部機能をユーザー操作を含めて処理するためのUI（画面）

- FragmentContainerViewを利用すると動的に操作（追加・削除・置換）することができる
- Fragmentのライフサイクル

※詳細は、公式ドキュメント（フラグメントのライフサイクル）参照

https://developer.android.com/guide/fragments/lifecycle?hl=ja

# ◆プログラムによるFragment操作

## ◆Activityから操作するのが基本

```
FragmentManager manager = getSupportFragmentManager();
FragmentTransaction transaction = manager.beginTransaction();
transaction.replace(containerViewId, fragment);
transaction.addToBackStack( name: null);
transaction.commit();
```

Fragment

# ◆Bundleオブジェクト

異なるアクティビティ間やフラグメント間でデータをやり取りするためのコンテナ

# ◆Fragmentのインスタンス化

- クラス内にnewInstanceメソッドを準備
- インスタンス変数データはここで受け取る

※引数ありコンストラクタなどから操作しないのが基本

```
public static DashBoardFragment newInstance(String param1, String param2) {
    DashBoardFragment fragment = new DashBoardFragment();
    Bundle args = new Bundle();
    args.putString(ARG_PARAM1, param1);
    args.putString(ARG_PARAM2, param2);
    fragment.setArguments(args);
    return fragment;
}
```

**画面間通信**

# ◆インスタンス変数データの格納

- Fragmentライフサイクルの流れを意識する
  - ➢ onCreateメソッドをオーバーライド
  - ➢ Bundleオブジェクトからデータ取得
    - ✓ getArguments()メソッドを利用する
  - ➢ インスタンス変数へ格納

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (getArguments() != null) {
        mParam1 = getArguments().getString(ARG_PARAM1);
        mParam2 = getArguments().getString(ARG_PARAM2);
    }
}
```

**画面間通信**

# ◆Fragment操作中のActivity取得

- Fragmentライフサイクルの流れを意識する
  - ➢ onAttachメソッドをオーバーライド
  - ➢ キャストしてインスタンス変数へ格納

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof MainActivity) {
        this.activity = ((MainActivity)context);
    }
}
```

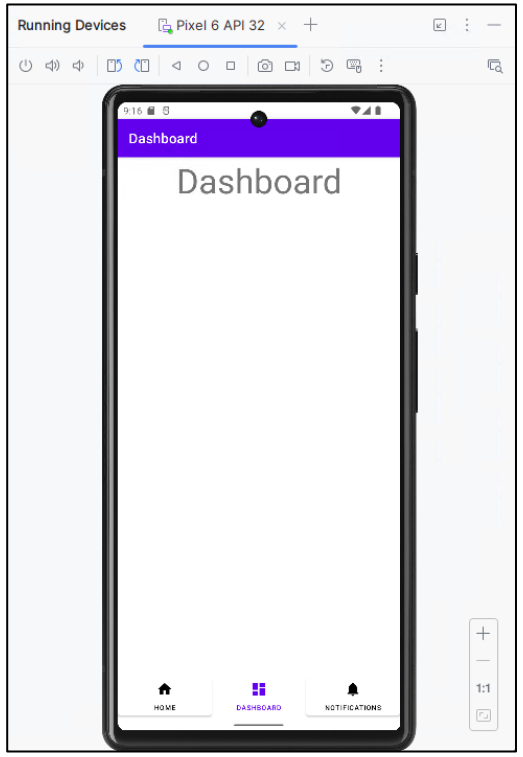
# ◆Fragment内のView操作

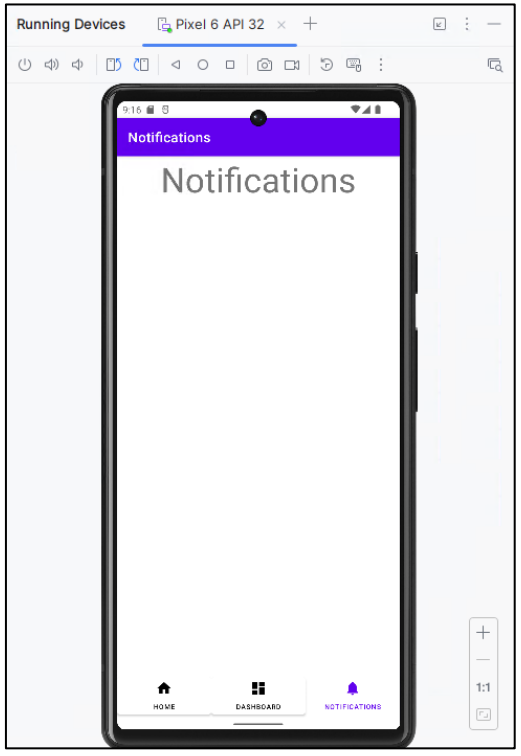
- Fragmentライフサイクルの流れを意識する
  - ➢ onViewCreated(View, Bundle)メソッドをオーバーライド

```
@Override
public void onViewCreated(View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
```

画面間通信

# Ex04（Ex03を改造）


# Fragmentと画面間通信

# ◆Ex05（Ex04を改造）

1. ログインチェック処理をonCreate(Bundle)メソッド内へ移動
2. 未ログイン時にBASHBOARDボタンとNOTIFICATIONSボタンを押下したとき、ログインフラグメントを表示




**簡単なアプリ作成体験**

# ◆Ex06（Ex05を改造）

1. BASHBOARD画面に標準アイコン画像、タイトルを２列グリッドで表示
2. NOTIFICATIONS画面に日付、タイトル、コンテントをリスト表示




**簡単なアプリ作成体験**

# ◆Ex06（Ex05を改造）・・参考

- BASHBOARD画面に標準アイコン画像、タイトルを２列グリッドで表示
  - 関連ファイル
    - DashBoardFragment.java　　…フラグメント描画処理
      - RecyclerViewオブジェクトを設定

```
@Override
public void onViewCreated(View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    final RecyclerView listView = ((RecyclerView) view.findViewById(R.id.main_dashboard_list));
    listView.setLayoutManager(new GridLayoutManager(view.getContext(), spanCount: 2));
    listView.setAdapter(new DashboardListAdapter(DashboardListTestData.get()));

}
```

    - DashboardListAdapter.java　　…グリッド描画処理
    - DashboardListItem.java　　…グリッド１項目データ格納用
    - DashboardListTestData.java　　…グリッドで扱うテストデータ
    - fragment_dash_board.xml　　…フラグメントのレイアウト
      - ✓ RecyclerViewにより、グリッド描画位置を定義
    - fragment_dashboard_list_item.xml　　…グリッドの１項目レイアウト

**簡単なアプリ作成体験**

# ◆Ex06（Ex05を改造）・・参考

- NOTIFICATIONS画面に日付、タイトル、コンテントをリスト表示
  - 関連ファイル
    - NotificationsFragment.java　　…フラグメント描画処理
      - ✓ RecyclerViewオブジェクトを設定
        ※DashBoardFragmentと同じ
    - NotificationListAdapter.java　　…グリッド描画処理
    - NotificationListItem.java　　…グリッド１項目データ格納用
    - NotificationListTestData.java　　…グリッドで扱うテストデータ
    - fragment_notifications.xml　　…フラグメントのレイアウト
      - ✓ RecyclerViewにより、グリッド描画位置を定義
    - fragment_notification_list_item0.xml…グリッドの１項目レイアウト
      - ✓ 奇数列のレイアウト
    - fragment_notification_list_item1.xml …グリッドの１項目レイアウト
      - ✓ 偶数列のレイアウト

**簡単なアプリ作成体験**

# 5.まとめ

◆ アクティビティとフラグメントの違いを知っていますか。
◆ イベントリスナーの設定方法を知っていますか。
◆ 変数とSharedPreferencesの違いを知っていますか。
◆ フラグメント切り替えで画面内容が変更できることを知っていますか。
◆ アクティビティのライフサイクルについて知っていますか。
◆ フラグメントのライフサイクルについて知っていますか。 。
◆ 画面間通信で注意することを説明できますか。
◆ 基本的なプロジェクト構造と階層ごとの役割を説明できますか。