アプリケーションドキュメンテーション

        - `onConfigurationChanged`メソッド
        - `adjustViewsForOrientation`メソッド
    - 12. `UrlMappingHelper.java`
        - パッケージ宣言
        - インポート宣言
        - クラス宣言
        - インスタンス変数
        - 静的初期化ブロック
        - プライベートコンストラクタ
        - `getUrl`メソッド
    - 13. `activity_file_list.xml`
        - ヘッダー部分
        - 動画ボタン部分
        - 設定画面表示コンテナ
        - ボトムナビゲーション部分

デモの手順

アプリケーションのデモを行う手順は以下の通りです。

1. 初めに、Webページの4つのDeepLinkのうちいずれかをタップし、APKファイルをダウンロードします。ここでは未インストール時にポップアップが表示されることを確認します。APKファイルは、Android Studioのプロジェクトからもダウンロードできます。事前に、デバイズ側の「不明なアプリのインストール」設定を許可しておいてください。
2. apkファイルからアプリをインストール後、Webページの4つのDeepLinkをタップし、それぞれが正常に動作することを確認します。
3. 各ボタンをタップして、正常に動作することを確認します。
4. デバイズを横画面にし、適切に画面が表示されていることを確認します。

ソースコードの概要説明

今回の成果物となるソースコードは、大きく12つです。各ファイルの詳細を以下に示します。ファイル名の後に、Android Studio内でどこのディレクトリに保存されているかを記載しています。

1. `index.html`(Webページ用)

このHTMLファイルはアプリケーションのデモ用のWebページを表しています。4つのDeepLink（`myapp://dda.io/view/file/123`、`myapp://dda.io/view/file/456`、`myapp://dda.io/view/file/789`、`myapp://dda.io/view/folders/123`）がそれぞれ異なるリンクとして記述されています。これらのリンクはアプリケーションがインストールされていない場合に、APKファイルのダウンロードリンクを表示するポップアップを表示します。

2. `activity_main.xml`(src/main/res/layout)

このXMLファイルはアプリケーションのメイン画面のレイアウトを定義しています。ヘッダー(戻るボタンとカテゴリ名)、動画名、設定の各部分がレイアウトに含まれています。

3. `bottom_nav_menu_main.xml`と`bottom_nav_menu_file_list.xml`(src/main/res/menu)

アプリのボトムナビゲーションメニューの項目を定義するXMLファイル。ビューアと設定(file_listでは一覧と設定)の2つのメニューアイテムが含まれています。

4. `fragment_settings.xml`(src/main/res/layout)

アプリ内の「設定」セクションのレイアウトを定義するXMLファイル。アプリのバージョン情報とフリーテキスト入力欄を表示するためのレイアウトが含まれています。

5. `MainActivity.java`(src/main/java/com/example/app)

このJavaファイルはアプリケーションのメインアクティビティを定義しています。ここではBottomNavigationViewを用いて`FileViewerFragment`や`SettingsFragment`への遷移を制御しています。また、アプリがバックグラウンドに移行してから10分後にアプリを自動的に終了する機能も実装されています。

6. `SettingsFragment.java`(src/main/java/com/example/app)

アプリの設定画面を表示するフラグメントのJavaファイル。レイアウトは`fragment_settings.xml`によって定義されています。

7. `AndroidManifest.xml`(src/main/res)

このXMLファイルはアプリケーションの基本的な設定とDeepLinkの設定を含んでいます。`<data>`タグ内の属性`android:host`、`android:scheme`、`android:pathPrefix`によりDeepLinkの形式が定義されています。

8. `YouTube.html`と`example_com.html`(src/main/assets)

これらのHTMLファイルはそれぞれ、YouTubeの動画と`example.com`のWebページを表示するためのiframeを含んでいます。これらのファイルはアプリケーション内部でローカルに保存され、DeepLinkからのパラメー

タにより特定のURLをWebViewにロードします。

9. `FileListActivity.java`(src/main/java/com/example/app)

このJavaファイルは、アプリケーション内でファイルのリストを表示するアクティビティを定義しています。このアクティビティは、ユーザーが利用可能なファイルやフォルダを一覧表示し、選択することで詳細を閲覧できるように設計されています。

10. `FileViewerFragment.java`(src/main/java/com/example/app)

`FileViewerFragment.java`は、WebViewを利用して特定のURLを表示するフラグメントを定義しています。DeepLinkからのパラメータを取得し、対応するURLをWebViewにロードします。

11. `UrlMappingHelper.java`(src/main/java/com/example/app)

このJavaファイルは、DeepLinkのパラメータとそれに対応するURLをマッピングするヘルパークラスです。特定のキーに対応するURLを返す`getUrl`メソッドが実装されています。

12. `activity_file_list.xml`(src/main/res/layout)

このXMLファイルは、`FileListActivity.java`で使用されるレイアウトを定義しています。ヘッダー部分、動画ボタン部分、設定画面を表示するためのコンテナ、およびボトムナビゲーション部分を含んでいます。各部分は、ユーザーがファイルやフォルダを効率的に閲覧・選択できるようにデザインされています。

ソースコードの詳細説明

1. index.html

ヘッダー部分

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Deep Link</title>
```

この部分は、HTML文書の宣言（<!DOCTYPE html>）と基本的なメタデータ（文字コード、ページタイトル）を設定する<head>セクションを示しています。

CSSスタイルシート

```
<style>
    #app-banner {
        display: none;
        position: fixed;
        bottom: 0;
        width: 100%;
        background-color: lightgrey;
        padding: 10px;
        text-align: center;
    }
</style>
```

この部分は、IDが"app-banner"の要素のスタイルを定義する内部CSSスタイルシートです。初期状態では、アプリインストールバナーは非表示に設定されています。

本文とディープリンク

```
</head>
<body>
    <p><a class="deep-link" href="myapp://dda.io/view/file/123">example.comのWebページ</a></p>
    ...
```

ここは、HTMLの本文部分を示しており、<body>タグで開始されます。特定のディープリンクを指す複数のリンクが含まれています。

アプリインストールバナー

```
<div id="app-banner">
    <p>Install our app for a better experience!</p>
    <a href="https://github.com/shuka733/Deeplinktest/raw/main/app-debug.apk"
download>Install App</a>
</div>
```

このセクションは、アプリをインストールするためのプロモーションバナーを表示する`<div>`要素を示しています。バナーにはアプリのapkファイルへの直接ダウンロードリンクが含まれています。

JavaScriptコード

```
<script>
    var deepLinks = document.querySelectorAll('.deep-link');
    ...
</script>
```

この部分は、ディープリンクのクリックイベントをハンドルするためのJavaScriptコードを示しています。クリックされたディープリンクを開き、一定時間後にアプリが未インストールである場合にインストールバナーを表示する機能が定義されています。

2. activity_main.xml

このレイアウトは、アプリケーションのメイン画面の基盤となるレイアウトです。内部の各要素はこのレイアウトの上に配置されます。

ヘッダーコンテナ

```
<LinearLayout
    android:id="@+id/header"
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:background="#FF5733"
    android:orientation="horizontal"
    android:gravity="center_vertical"
    android:weightSum="3"
    android:layout_alignParentTop="true">
    ...
</LinearLayout>
```

この部分はアプリケーションのヘッダーを表示するためのレイアウトです。ヘッダー内には、左側に戻るボタン、中央にカテゴリ名表示部分、そして右側にダミーのビューが配置されています。

戻るボタン

```
<TextView
    android:id="@+id/back"
    ...
    android:text="@string/nav_back"
    android:textSize="16sp"
    ...
    android:gravity="center|start" />
```

この部分は、ヘッダーの左端に表示される戻るボタンのテキストビューです。

カテゴリ名表示部分

```
<TextView
    android:id="@+id/fileViewer"
    ...
    android:text=""
    android:textSize="20sp"
    android:gravity="center" />
```

ヘッダー中央に表示されるテキストビューです。動的にカテゴリ名やその他の情報を表示するための部分です。

パラメータ表示部分

```
<TextView
    android:id="@+id/parameter"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:layout_below="@id/header"
    android:text=""
    android:textSize="20sp"
    android:gravity="center" />
```

この部分は、DeepLinkから受け取ったパラメータを表示するためのテキストビューです。

フラグメント表示コンテナ

```
<FrameLayout
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_above="@+id/bottomNavigationView"
    android:layout_below="@id/parameter" />
```

この部分は、異なるフラグメントを表示するためのコンテナとして機能するフレームレイアウトです。

WebView部分

```
<WebView
    android:id="@+id/webview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_above="@+id/bottomNavigationView"
    android:layout_below="@id/parameter" />
```

この部分は、Webページを表示するためのWebViewです。DeepLinkからのパラメータに応じて、異なるURLがロードされます。

ボトムナビゲーション部分

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottomNavigationView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
        android:layout_alignParentBottom="true"
        app:menu="@menu/bottom_nav_menu_main" />
```

この部分は、アプリケーションの下部に配置されるボトムナビゲーションビューです。"ビューア"と"設定"という2つのメニューアイテムを持っています。

3. bottom_nav_menu_main.xml

このXMLファイルは、アプリケーションのボトムナビゲーションメニューのアイテムを定義しています。

メニューアイテム: ビューア

```
<item
    android:id="@+id/viewer"
    android:title="@string/nav_viewer"
    app:showAsAction="ifRoom" />
```

- android:id: このアイテムのIDとしてviewerを指定しています。これにより、Javaコード側からこのアイテムを識別・操作できます。
- android:title: このアイテムの表示名としてリソース文字列@string/nav_viewer（"ビューア"）を使用しています。
- app:showAsAction: このアイテムがアクションバーの中に表示される条件を指定しています。ifRoomはスペースがある場合にのみアクションバーに表示することを意味します。ボトムナビゲーションにおいては通常は無視されますが、一般的なメニューにおいてはこの属性が役立ちます。

メニューアイテム: 設定

```
<item
    android:id="@+id/settings"
    android:title="@string/nav_settings"
    app:showAsAction="ifRoom" />
```

- android:id: このアイテムのIDとしてsettingsを指定しています。
- android:title: このアイテムの表示名としてリソース文字列@string/nav_settings（"設定"）を使用しています。
- app:showAsAction: 同様に、このアイテムがアクションバーの中に表示される条件を指定しています。

4. fragment_settings.xml

このXMLファイルは、アプリケーションの設定画面のレイアウトを定義しています。

ルート要素: LinearLayout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">
```

- android:orientation: 子要素を垂直方向に並べるためにverticalを指定しています。
- android:padding: すべての側面に16dpのパディングを追加しています。これにより、子要素が端に接触しないようになります。

テキストビュー: バージョン情報

```
<TextView
    android:id="@+id/versionInfo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/version_info"
    android:textSize="20sp" />
```

- android:id: このテキストビューのIDとしてversionInfoを指定しています。
- android:text: テキストビューの内容としてリソース文字列@string/version_infoを表示しています。
- android:textSize: テキストのサイズとして20spを指定しています。

エディットテキスト: フリーテキスト入力

```
<EditText
    android:id="@+id/editableText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter text here..."
    android:layout_marginTop="16dp"/>
```

- android:id: このエディットテキストのIDとしてeditableTextを指定しています。
- android:hint: ユーザーが何も入力していないときに表示されるヒントテキストとして"Enter text here..."を指定しています。
- android:layout_marginTop: 上側に16dpのマージンを追加しています。

このレイアウトは設定画面を表しており、バージョン情報を表示するテキストビューと、フリーテキストを入力するためのエディットテキストが含まれています。

5. MainActivity.java

パッケージ宣言

```
package com.example.app;
```

この行は、このJavaクラスが所属するパッケージを宣言しています。`com.example.app`はアプリケーションの名前空間を示しており、このパッケージの中には`MainActivity`の他にもいくつかのクラスやリソースが含まれている可能性があります。

インポート宣言

```
import android.content.Intent;
import android.content.res.Configuration;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import androidx.appcompat.app.AppCompatActivity;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import android.net.Uri;
```

以下の部分は、このクラスで使用する外部のクラスやインターフェースをインポートするための宣言です。

クラス宣言

```
public class MainActivity extends AppCompatActivity {
```

この行は、`MainActivity`という名前の新しいクラスを宣言しています。

インスタンス変数

```
private Handler handler;
private Runnable shutdownRunnable;
```

これらはクラスのプライベートインスタンス変数です。

`onCreate`メソッド

```
@Override
protected void onCreate(Bundle savedInstanceState) {
```

```
    ...
}
```

onCreateはアクティビティライフサイクルにおける最初に呼ばれるメソッドです。

5.1. 基本的な初期化

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
```

上記のコードで、親クラスのonCreateメソッドを呼び出し、UIのレイアウトを設定しています。

5.2. アクションバーの非表示設定

```
if (getSupportActionBar() != null) {
    getSupportActionBar().hide();
}
```

アプリケーションのアクションバーを非表示にしています。

5.3. シャットダウンの準備

```
handler = new Handler();
shutdownRunnable = new Runnable() {
    @Override
    public void run() {
        finishAndRemoveTask();
    }
};
```

アプリがバックグラウンドに移行した後のシャットダウンのための準備をしています。

5.4. ボトムナビゲーションの初期化とイベントリスナー

```
BottomNavigationView bottomNav = findViewById(R.id.bottomNavigationView);
bottomNav.setOnNavigationItemSelectedListener(item -> {
    ...
});
```

ボトムナビゲーションのインスタンスを取得し、各アイテムが選択されたときの動作を定義しています。

5.5. 画面の向きに応じてビューの調整

```
adjustViewsForOrientation(getResources().getConfiguration().orientation);
```

画面の向きに応じてビューの表示を調整するメソッドを呼び出しています。

アクティビティのライフサイクルメソッド

5.6. onConfigurationChanged

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    adjustViewsForOrientation(newConfig.orientation);
}
```

デバイスの画面の向きが変わったときに呼ばれるメソッドです。画面の向きに応じてビューを調整します。

5.7. onPause()

```
@Override
protected void onPause() {
    super.onPause();
    handler.postDelayed(shutdownRunnable, 10 * 60 * 1000);
}
```

アクティビティがバックグラウンドに移動するときに呼ばれるメソッドです。

5.8. onResume()

```
@Override
protected void onResume() {
    super.onResume();
    handler.removeCallbacks(shutdownRunnable);
}
```

アクティビティがフォアグラウンドに戻るときに呼ばれるメソッドです。

6. SettingsFragment

SettingsFragmentは、設定画面を表示するためのフラグメントであり、UIのレイアウトはfragment_settings.xmlファイルに定義されています。このクラスはシンプルですが、アプリケーションの設定や情報を表示するための基本的な機能を持っています。

クラスの概要

```
public class SettingsFragment extends Fragment {
```

このクラスは、Fragmentクラスを拡張することで、アプリケーション内の設定画面部分を表しています。

コンストラクタ

```
// コンストラクタ
public SettingsFragment() {
    // Required empty public constructor
}
```

SettingsFragmentの公開コンストラクタです。フラグメントのインスタンス生成時に呼び出されるものの、特に何もしていません。フラグメントには公開のデフォルトコンストラクタが必要とされるため、これが用意されています。

onCreateView メソッド

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // このフラグメントのレイアウトをインフレートする
    return inflater.inflate(R.layout.fragment_settings, container, false);
}
```

このonCreateViewメソッドは、フラグメントのUI部分が初めて描画される際にシステムから呼び出されます。メソッド内では、LayoutInflaterを使用して、fragment_settings.xmlというリソースファイルに定義されたレイアウトをインフレート（ロード）しています。その後、このインフレートされたビューがメソッドから返され、フラグメントのUIとして表示されます。

7. activity_file_list.xml

ヘッダー部分

```
<LinearLayout
    android:id="@+id/header"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#E0E0E0"
    android:orientation="vertical"
    android:padding="16dp"
    android:layout_alignParentTop="true"
    android:gravity="center_horizontal">

    <TextView
        android:id="@+id/parameterTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:textSize="20sp"
        android:textColor="#000000" />
</LinearLayout>
```

この部分はアプリケーションのヘッダーを表示するためのレイアウトです。

動画ボタン部分

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/header"
    android:layout_above="@+id/bottomNavigationView">

    <LinearLayout
        android:id="@+id/buttonContainer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <!-- 以下、ボタンの例 -->
        <Button
            android:id="@+id/btnExampleCom"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp" />
    </LinearLayout>
</ScrollView>
```

この部分には、動画の一覧を表示するためのボタンが配置されます。

設定画面表示用のコンテナ

```
<FrameLayout
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/header"
    android:layout_above="@+id/bottomNavigationView" />
```

この部分は、設定画面を表示するためのフラグメントを配置するコンテナとして機能します。

ボトムナビゲーション部分

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottomNavigationView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    app:menu="@menu/bottom_nav_menu_file_list" />
```

この部分は、アプリケーションの下部に配置されるボトムナビゲーションビューです。

8. YouTube.html と example_com.html

これらのHTMLファイルは、アプリケーション内部でローカルに保存されているもので、アプリのWebViewで表示する際のコンテンツとして利用されます。

ドキュメントの宣言とヘッダー部分

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>YouTube Video</title>
```

この部分は、HTML文書の宣言（`<!DOCTYPE html>`）と、基本的なメタデータ（文字コード、ページタイトル）を設定する`<head>`セクションです。

スタイルの定義

```
<style>
    html, body {
        height: 100%;
        margin: 0;
        padding: 0;
        background-color: #000;
    }
    .video-container {
        position: relative;
        height: 100%;
    }
    .video-container iframe {
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
    }
</style>
```

この内部CSSスタイルシートは、ページ全体の背景色やiframeの動画の位置、サイズを定義しています。具体的には、動画がページ全体をカバーするように設定されています。

本文と動画の埋め込み

```
<body>
<div class="video-container">
```

```
    <iframe src="https://www.youtube.com/embed/bjmBJ1Fl0cs?autoplay=1"
frameborder="0" allowfullscreen></iframe>
</div>
</body>
```

この部分は、本文（`<body>`）の開始で、YouTubeの動画を埋め込む`<iframe>`が含まれています。動画は自動再生モードで、フルスクリーンモードも許可されています。

`example_com.html`について

`example_com.html`は、基本的な構造とスタイルが`YouTube.html`と同一です。唯一の違いは、`<iframe>`の`src`属性が`https://www.youtube.com/embed/bjmBJ1Fl0cs?autoplay=1`から`http://example.com`に変更されている点です。このことから、`example_com.html`は`http://example.com`のページ内容を埋め込むためのものとして機能します。

9. FileListActivity.java

パッケージ宣言

```
package com.example.app;
```

このJavaクラスが所属するパッケージを宣言しています。com.example.appはアプリケーションの名前空間を示しており、このパッケージの中にはFileListActivityの他にもいくつかのクラスやリソースが含まれている可能性があります。

インポート宣言

```
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentTransaction;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.util.HashMap;
import java.util.Map;
```

このクラスで使用する外部のクラスやインターフェースをインポートするための宣言です。

クラス宣言

FileListActivityは、アプリ内のファイル一覧を表示するアクティビティです。このクラスはAndroidのAppCompatActivityを継承しているため、Androidのライフサイクルメソッドや UI関連の機能を利用することができます。

インスタンス変数

```
private LinearLayout buttonContainer;
private TextView headerTextView;
```

buttonContainerは動画のボタン一覧を格納するLinearLayoutを参照します。headerTextViewはヘッダー部分のテキストを表示するTextViewを参照します。

onCreateメソッド

アクティビティのインスタンスが生成される際にシステムから呼び出されるメソッドで、UIのセットアップや初期化を行います。

ボタンリスナーの設定

```
private void setButtonListeners() { ... }
```

このメソッドでは、各動画のボタンに対してクリックリスナーを設定しています。ユーザーがボタンをタップすると、`openFileViewer`メソッドが呼び出され、指定されたパラメータをもとに`MainActivity`が起動します。

ボトムナビゲーションのリスナー設定

```
private void setBottomNavigationListener(BottomNavigationView bottomNav) { ... }
```

ボトムナビゲーションの各アイテムが選択されたときの動作をこのメソッドで定義しています。

ボタンテキストの更新

```
private void updateButtonText() { ... }
```

各動画のボタンのテキストをこのメソッドで更新しています。動画のカテゴリ名と動画名を組み合わせて、ボタンのテキストとして設定しています。

`FileViewer`の起動

```
private void openFileViewer(String parameter) { ... }
```

指定されたパラメータに基づいて`MainActivity`を起動するこのメソッドでは、パラメータは動画のURLを決定するためのキーとして使用されます。

10. SettingsFragment.java

パッケージ宣言

```
package com.example.app;
```

この行は、Javaクラスが所属するパッケージを示しています。`com.example.app`はアプリケーションの名前空間を示しており、このパッケージ内には他のクラスやリソースが存在する可能性があります。

インポート宣言

```
import androidx.fragment.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

これらの行は、`SettingsFragment`クラスで使用する外部のクラスやインターフェースをインポートするための宣言です。特に、`androidx.fragment.app.Fragment`はAndroidのフラグメント関連の基本クラスを表しています。

クラス宣言

```
public class SettingsFragment extends Fragment {
```

この行で、`SettingsFragment`という新しいフラグメントクラスを宣言しています。このクラスは`Fragment`クラスを継承しており、これによりAndroidのフラグメントの基本的な機能を使用できます。

コンストラクタ

```
public SettingsFragment() {
    // Required empty public constructor
}
```

この部分は`SettingsFragment`のデフォルトコンストラクタを定義しています。フラグメントのインスタンスがシステムによって再生成される際には、引数なしの公開コンストラクタが必要です。

`onCreateView` メソッド

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    return inflater.inflate(R.layout.fragment_settings, container, false);
}
```

`onCreateView`はフラグメントのライフサイクルにおいてUIが初めて描画される際にシステムから呼び出されるメソッドです。このメソッド内で、フラグメントのUIがインフレートされ、そのルートビューが返されます。この例では、`fragment_settings.xml`がインフレートされています。

11. FileViewerFragment.java

パッケージ宣言

```
package com.example.app;
```

この行は、このJavaクラスが所属するパッケージを宣言しています。このパッケージ内には、FileViewerFragmentをはじめとする様々なクラスやリソースが含まれる可能性があります。

インポート宣言

各種ライブラリやフレームワークのクラスを利用するためのインポート文です。

クラス宣言

```
public class FileViewerFragment extends Fragment {
```

この行は、FileViewerFragmentという名前の新しいクラスを宣言しています。このクラスはAndroidのFragmentを拡張しています。

onCreateViewメソッド

```
@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
    ...
}
```

このメソッドは、フラグメントのUIを生成するためにシステムによって呼び出されます。ここでは、fragment_file_viewerというレイアウトリソースをインフレートしています。

onViewCreatedメソッド

このメソッドは、フラグメントのビューが作成された後にシステムによって呼び出されます。ここでの主な処理は、WebViewの設定やIntentからのデータ取得、ボタンのクリックリスナー設定などです。

onConfigurationChangedメソッド

```
@Override
public void onConfigurationChanged(@NonNull Configuration newConfig) {
    ...
}
```

デバイスの設定（例：画面の向き）が変更されたときに呼ばれるメソッドです。このメソッド内で、画面の向きに応じてビューの表示を調整する`adjustViewsForOrientation`メソッドを呼び出しています。

`adjustViewsForOrientation`メソッド

このプライベートメソッドは、画面の向きに応じてビューの表示を調整するためのものです。例えば、画面がランドスケープモードの場合、一部のビューを非表示にしたり、逆にポートレートモードの場合は表示したりしています。

12. UrlMappingHelper.java

パッケージ宣言

```
package com.example.app;
```

この行は、このJavaクラスが所属するパッケージを宣言しています。

インポート宣言

このクラスで使用する外部のクラスやインターフェースをインポートするための宣言です。ここでは、HashMapやMapをインポートしています。

クラス宣言

```
public class UrlMappingHelper {
```

この行は、UrlMappingHelperという名前の新しいクラスを宣言しています。このクラスはDeepLinkのパラメータから対応するURLを取得するためのヘルパークラスとして機能します。

インスタンス変数

```
private static final Map<String, String> URL_MAP = new HashMap<>();
```

この行は、DeepLinkのパラメータと対応するURLをマッピングするための静的なマップを宣言しています。

静的初期化ブロック

```
static {
    ...
}
```

この部分は、静的初期化ブロックとして、クラスがロードされるときに一度だけ実行されるコードブロックです。ここでは、URL_MAPにパラメータと対応するURLのマッピングを追加しています。

プライベートコンストラクタ

```
private UrlMappingHelper() {}
```

この行は、このクラスのインスタンス化を防ぐためのプライベートコンストラクタを宣言しています。

`getUrl`メソッド

```
public static String getUrl(String key) {
    ...
}
```

このメソッドは、指定されたキーに対応するURLを`URL_MAP`から取得して返します。

13. activity_file_list.xml

このレイアウトは、アプリケーション内で利用可能な動画のリストを表示する画面の基盤となるレイアウトです。

ヘッダー部分

```
<LinearLayout
    android:id="@+id/header"
    ...
    android:gravity="center_horizontal">

    <TextView
        android:id="@+id/parameterTextView"
        ...
        android:textColor="#000000" />
</LinearLayout>
```

この部分はアプリの上部に表示され、DeepLinkから受け取ったパラメータを表示するためのテキストビューを含む。

動画ボタン部分

```
<ScrollView
    ...
    android:layout_above="@+id/bottomNavigationView">

    <LinearLayout
        android:id="@+id/buttonContainer"
        ...
        android:orientation="vertical">

        <Button
            android:id="@+id/btnExampleCom"
            ...
            android:layout_marginTop="16dp" />

        <Button
            android:id="@+id/btnYouTube"
            ...
            android:layout_marginTop="16dp" />

        <Button
            android:id="@+id/btnExampleDotCom"
            ...
            android:layout_marginTop="16dp" />
    </LinearLayout>
</ScrollView>
```

利用可能な動画のリストを表示する部分です。各動画を開くためのボタンが垂直に並べられています。

設定画面表示コンテナ

```
<FrameLayout
    android:id="@+id/fragment_container"
    ...
    android:layout_above="@+id/bottomNavigationView" />
```

設定フラグメントを表示するためのコンテナ部分です。

ボトムナビゲーション部分

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottomNavigationView"
    ...
    app:menu="@menu/bottom_nav_menu_file_list" />
```

アプリケーションの下部に配置されるボトムナビゲーションビューです。この画面では、動画リストと設定画面を切り替えるためのメニューアイテムが配置されています。
