DroidKaigi 2020

俺が今までやらかした失敗事例
やらかしそうになった
ヒヤリハット事例を紹介する

大前良介 (OHMAE Ryosuke)

自己紹介

- 大前良介（OHMAE Ryosuke）
  - https://github.com/ohmae
  - twitter: ryo_mm2d
  - qiita: ryo_mm2d
- ヤフー株式会社 @グランフロント大阪
  - Androidアプリエンジニア
  - Yahoo!天気アプリ担当

過去のDroidKaigi

- DroidKaigi 2018
  - タッチイベントを捕まえよう
- DroidKaigi 2019
  - Chrome Custom Tabsの仕組みから学ぶプロセス間通信
- DroidKaigi 2020 イマココ
  - 俺が今までやらかした失敗事例、やらかしそうになったヒヤリハット事例を紹介する

本発表には以下の成分が含まれます

- 脈絡のない失敗事例の列挙
  - Android開発で発生する問題あるある
  - 合計〇〇円分の損失を出しました、などの内容は含まれません
- 突如始まるエモい話

予防線

- 本発表は個人の見解であって、所属組織を代表するものではありません
- あくまであるある事例の紹介
  - やらかしたのか、ヒヤリハットですんだのかも秘密

察してください（重要）

startActivity
でクラッシュ

```
android.content.ActivityNotFoundException:
 No Activity found to handle Intent { xxxx }
```

ActivityNotFoundException

- Intentを受け取れるActivityが見つからない
  - 外部特定アプリを起動するIntentで仕様がかわった
  - 暗黙的Intentを投げたが通常あるであろうアプリがインストールされていない
  - システム設定を呼び出したが、特定メーカーのカスタム設定画面は呼び出せなかった
- ほとんどの環境、テスト環境でも問題無い

対策

- try/catchしましょう
  - 特に外部アプリを起動するときは必ずcatchするクセを

```
fun Context.startActivitySafely(intent: Intent) {
    runCatching { startActivity(intent) }
}
```

```
android.os.TransactionTooLargeException
        at android.os.BinderProxy.transactNative(Native Method)
        at android.os.BinderProxy.transact(Binder.java:496)
```

TransactionTooLargeException

- トランザクションデータが大きすぎる
  - プロセス内のトランザクションバッファ1MB
  - プロセス内の合計が超えるとアウト
  - IntentのExtra/onSaveInstanceStateのBundle

- 正直制限が厳しい
  - 誰だよこんなデータ突っ込んだの

（レガシーコードあるある）

対策

- IntentやsavedStateに大きなデータを置かない、キーとなるパラメータだけを置く
  - Activity内　：ViewModel
  - Activity間　：キャッシュ・永続化データ・再取得
  - プロセス間　：AIDL

消えた
ウィジェット

アップデートとともに消えるウィジェット

- ウィジェットが消えた！
- 表示できませんに変わった！

※OSやホームアプリによって症状が違う

原因

- リファクタリングでWidgetProviderのComponentNameが変わった
  - WidgetProviderのComponentNameに変化があると設置済みのウィジェットと紐付けができなくなる

原因

- リファクタリングでWidgetProviderのComponentNameが変わった
  - WidgetProviderのComponentNameに変化があると設置済みのウィジェットと紐付けができなくなる

起動できない
ショートカット

ホーム画面に作成したショートカットが！

- 消えた！
- タップしてもアプリが起動しない！

※追従してくれる場合もある

原因

- リファクタリングでActivityの
  ComponentNameが変わった
  - アプリドロワーはIntentFilterから列挙
  - ショートカットはComponentNameで管理

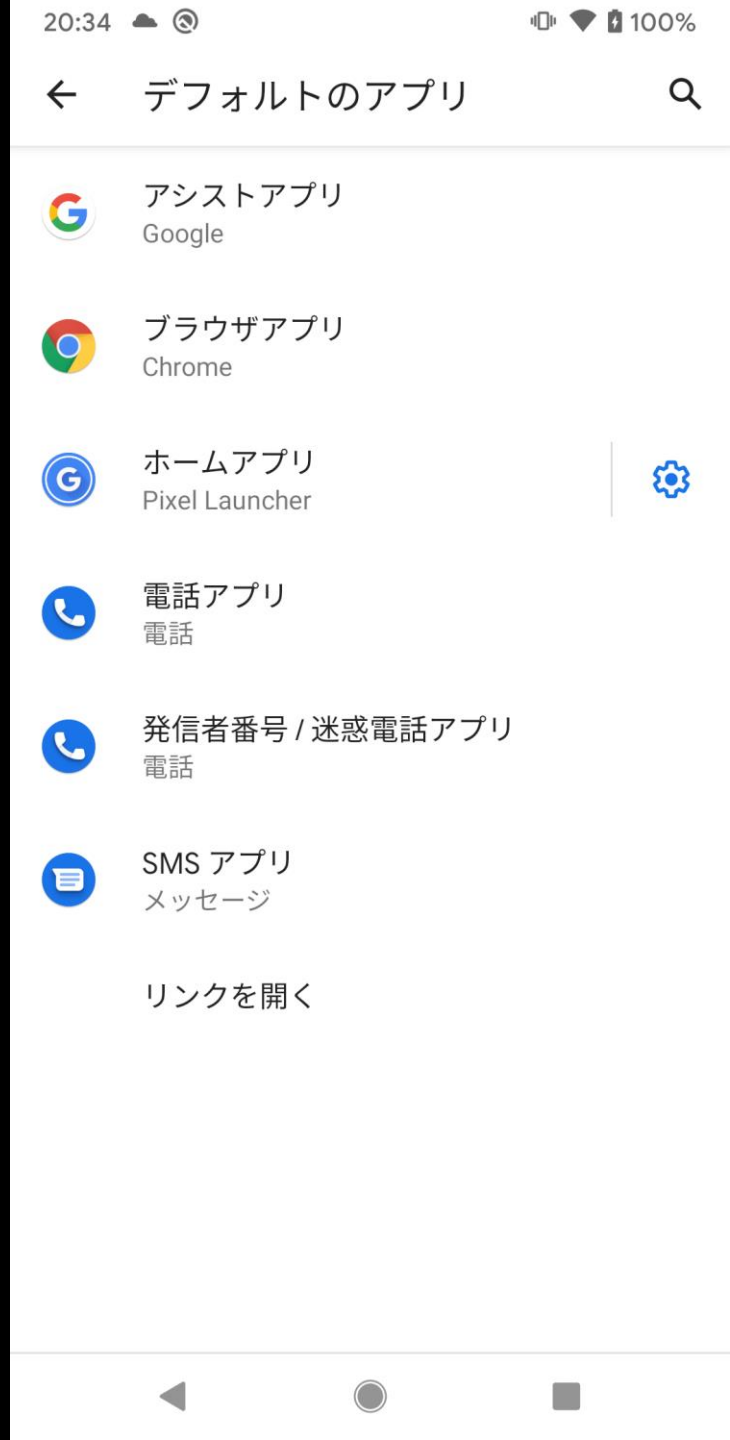
- ComponentNameが変わると紐付けが外れる

外された
デフォルトアプリ

デフォルトアプリ

- セレクターを経由しないで起動する
  - ブラウザーやホームアプリにとって重要
  - アプリ側からデフォルトにすることは不可

- せっかく設定してもらえたのに
  はずれた!?



原因

- リファクタリングでActivityの CompnentNameが（ry
  - デフォルトアプリもCompnentNameで識別
- ComponentNameが変わると紐付けが外れる

AndroidManifest

- アプリの外部仕様の宣言
  - データを扱うのはアプリ外
  - 削除と追加はできてもマイグレーションは不可能
- Manifestへの変更はプロトコルの変更である
  - 過去・未来・外部アプリからの影響を考慮する

対策

- Manifestの変更を極力回避する
  - 変更しなくてよいように新規作成、追加時に検討する
- リファクタリング・アーキテクチャ変更
  - Activity：activity-alias
  - その他：ロジックの乗らない踏み台となる層を残す

```
class OldService: NewService()
class OldReceiver: NewReceiver()
```

Android 4.x

Android 4.3以下で ClassNotFoundException

- Objects

Android 4.3以下で ClassNotFoundException

- Objects
  - Java7で追加されたクラス

```
Objects.requireNonNull(hoge)
```

Android 4.3以下で ClassNotFoundException

- Objects
  - Java7で追加されたクラス

```
Objects.requireNonNull(hoge)
```

  - デバッグビルドだと動く

Android 4.3以下で ClassNotFoundException

- Objects
  - Java7で追加されたクラス

```
Objects.requireNonNull(hoge)
```

  - デバッグビルドだと動く
  - リリースビルドするとClassNotFoundException

Android 4.3以下で ClassNotFoundException

- Objects
  - Java7で追加されたクラス

```
Objects.requireNonNull(hoge)
```

  - デバッグビルドだと動く
  - リリースビルドするとClassNotFoundException

なんでや！

Android 4.3以下でClassCastException

```
Caused by: java.lang.ClassCastException:
android.content.res.XmlBlock$Parser cannot be cast to
java.lang.AutoCloseable
```

```
context.resources.getXml(resId).use {
    loadFromResource(context, it, target)
}
```

Android 4.3以下でClassCastException

```
Caused by: java.lang.ClassCastException:
android.content.res.XmlBlock$Parser cannot be cast to
java.lang.AutoCloseable
```

```
context.resources.getXml(resId).use {
    loadFromResource(context, it, target)
}
```

API 19

```
public interface XmlResourceParser extends XmlPullParser,
AttributeSet, AutoCloseable {
    String getAttributeNamespace (int index);
    public void close();
}
```

API 19

```
public interface XmlResourceParser extends XmlPullParser,
AttributeSet, AutoCloseable {
    String getAttributeNamespace (int index);
    public void close();
}
```

API 18

```
public interface XmlResourceParser extends XmlPullParser,
AttributeSet {
    public void close();
}
```

Kotlinの罠

- jdk7の意味

```
implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
```

Kotlinの罠

- jdk7の意味

```
implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
```

https://github.com/JetBrains/kotlin/blob/master/libraries/stdlib/jdk7/src/kotlin/AutoCloseable.kt

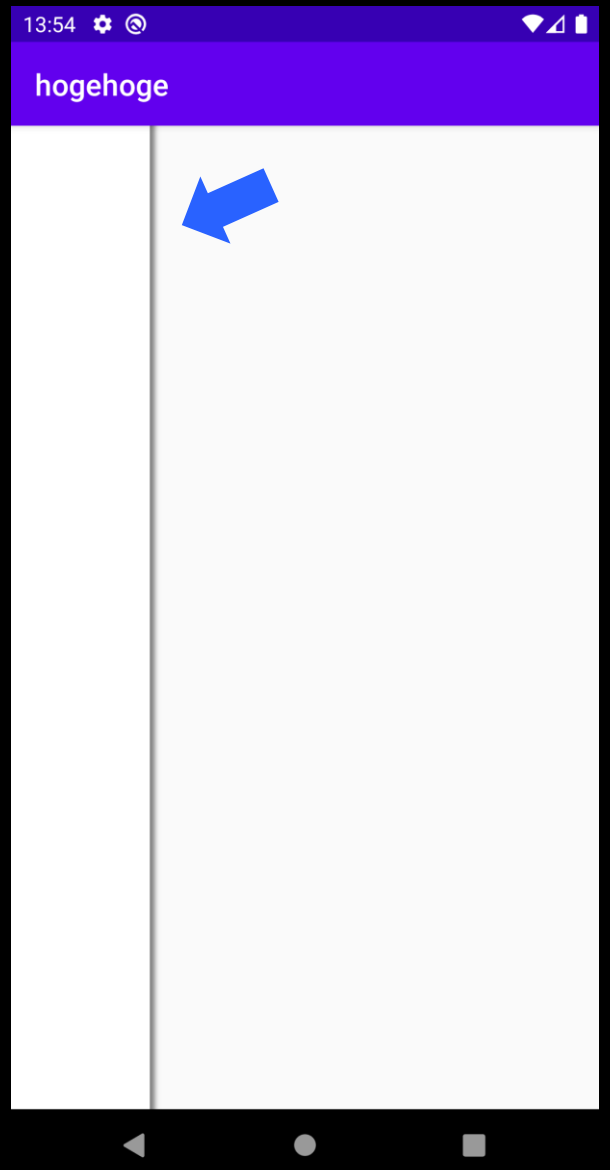
```
public inline fun <T : AutoCloseable?, R> T.use(block: (T) -> R): R {
    var exception: Throwable? = null
    try {
        return block(this)
    } catch (e: Throwable) {
        exception = e
        throw e
    } finally {
        this.closeFinally(exception)
    }
}
```

対策

- ~~Android 4.xのサポートを切る~~
  - サポートを維持するコストとリスクを把握しましょう
- Java7/8の機能を使うときは要注意
  - 使えないクラスを使っても警告が出ない場合がある

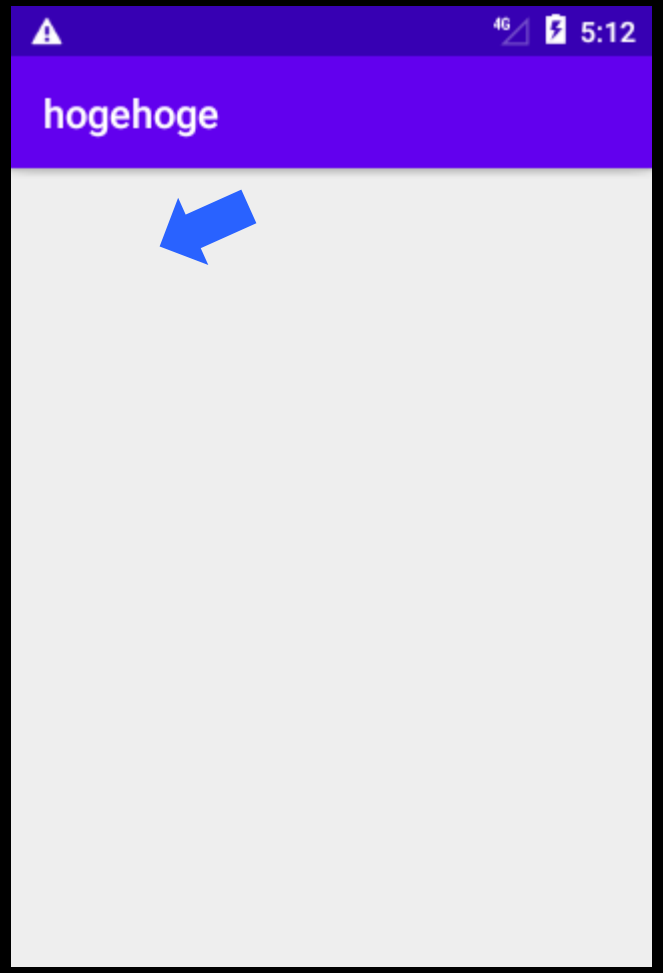
```
compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}
```

消えた9-patch



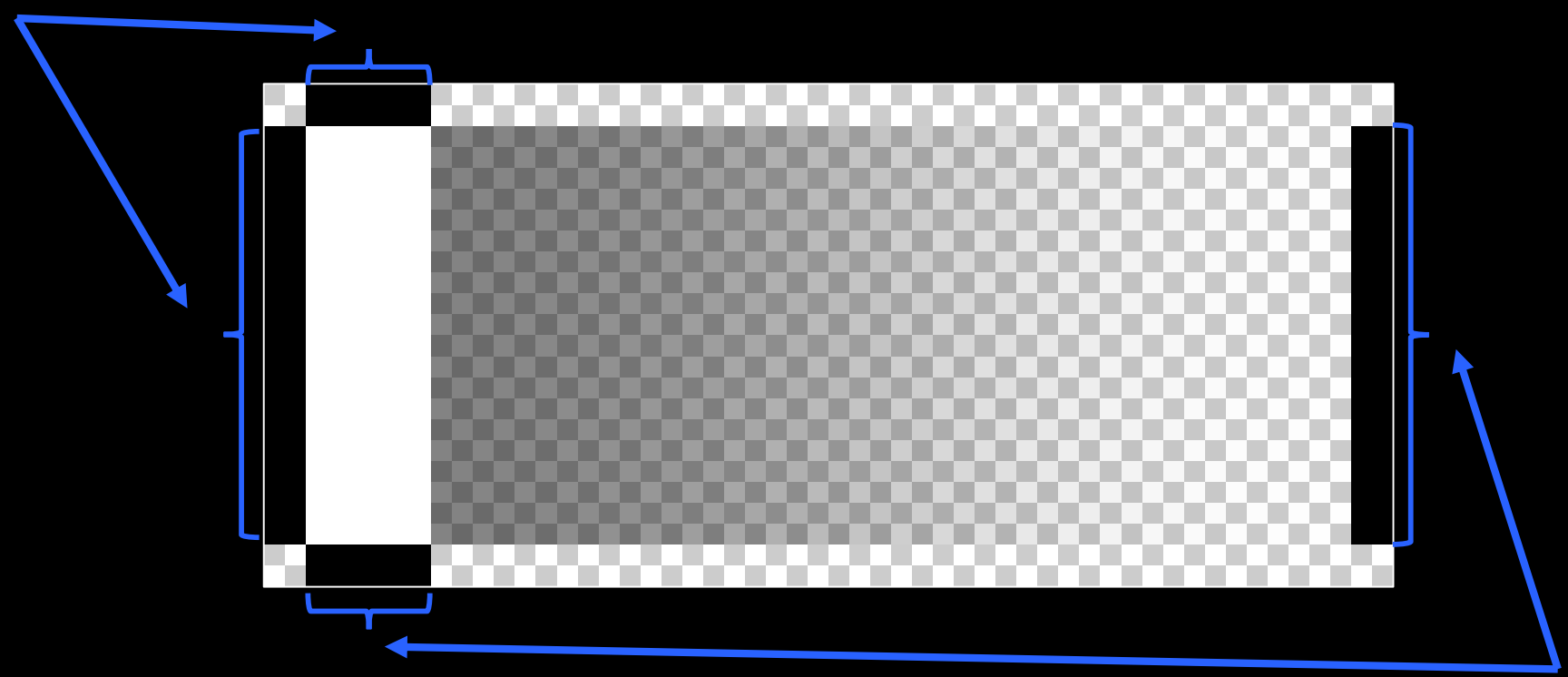
9-patchで作った境界線

見えなくなる端末がある？



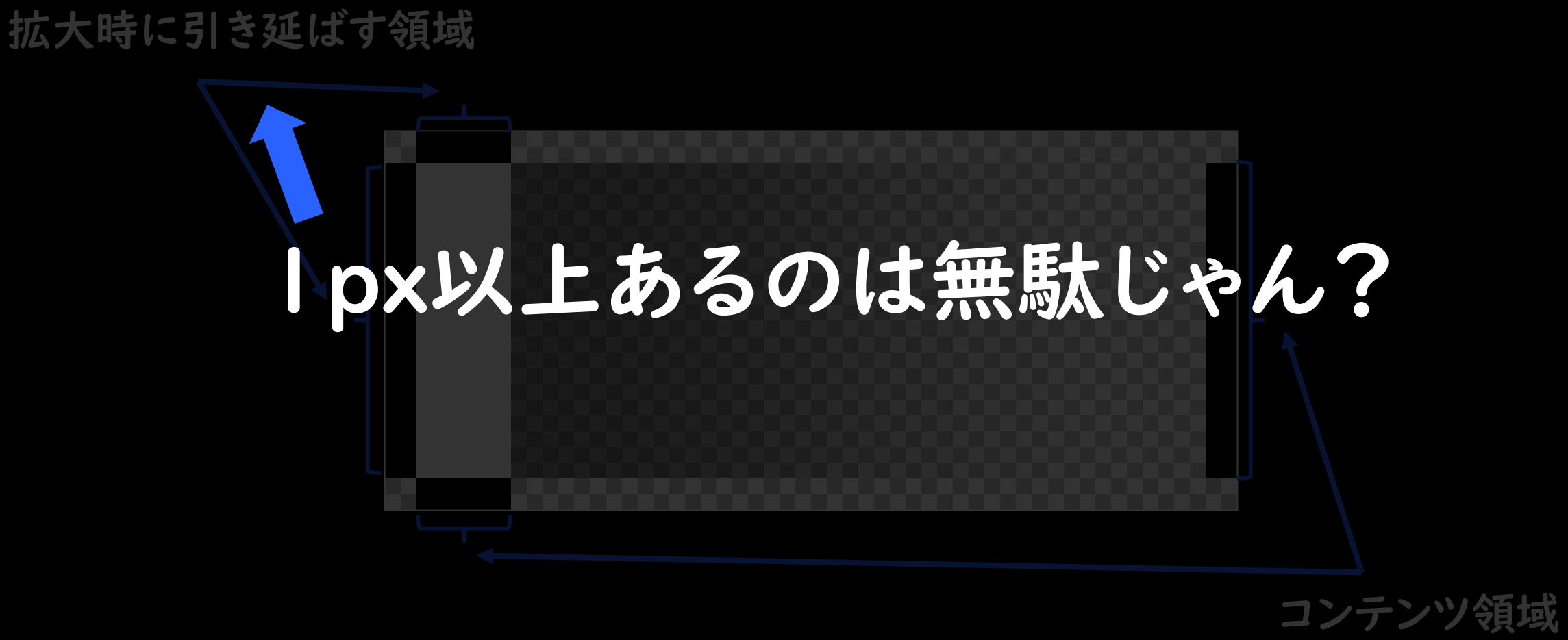
9-patchとは

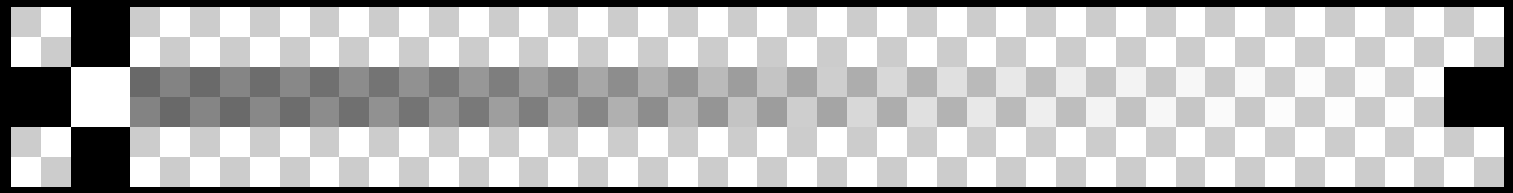
拡大時に引き延ばす領域



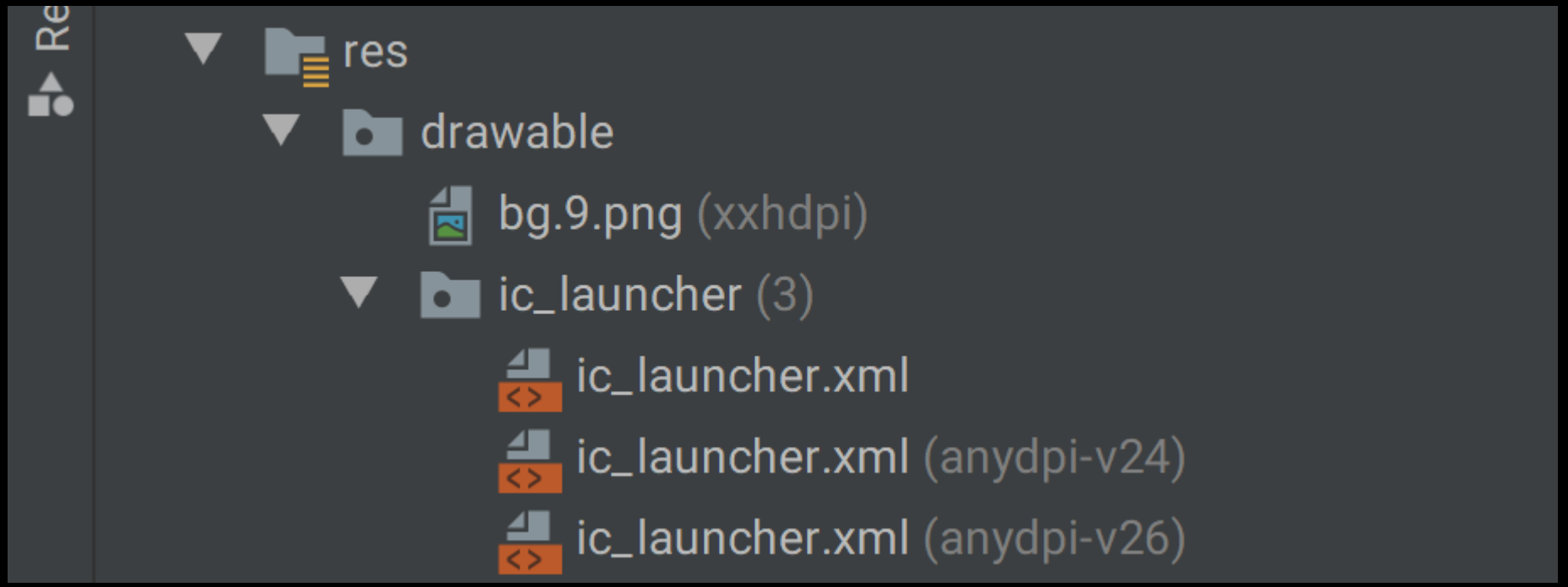
コンテンツ領域

9-patchとは

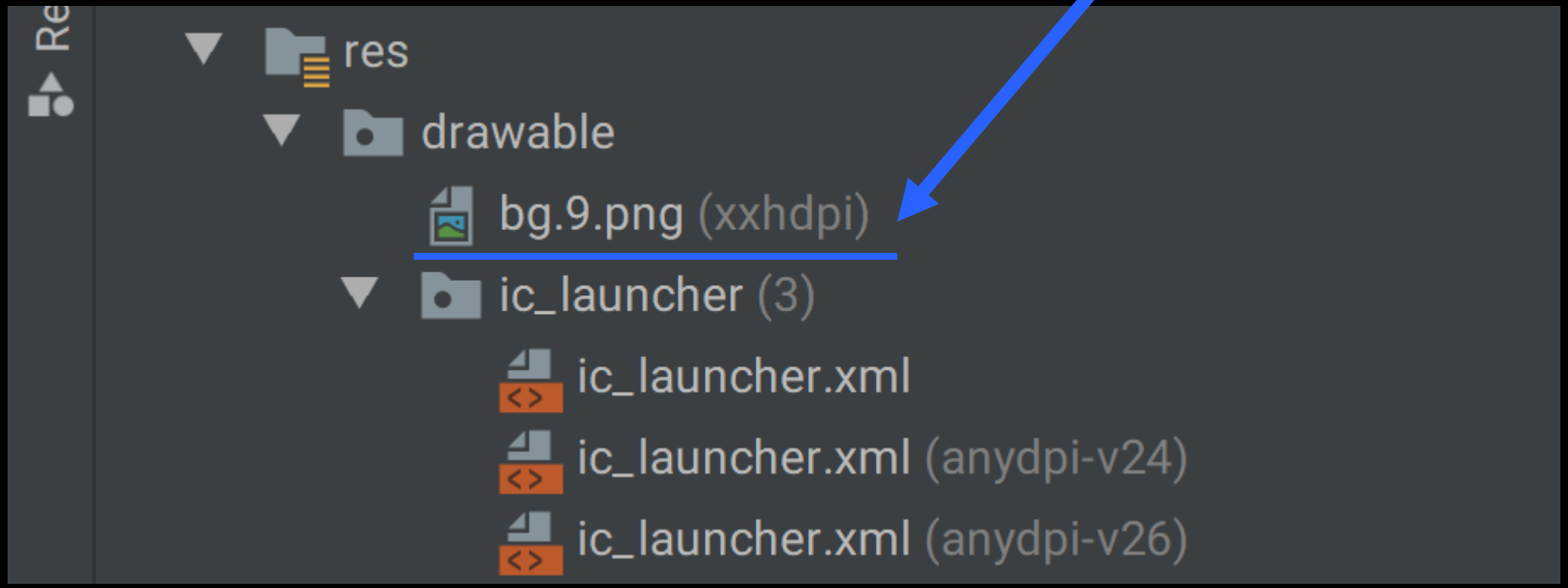




一切無駄のない9-patch
ふつくしい……



あっ



なにが起こっていたか

- 9-patchを無駄なく1pxの画像にした

なにが起こっていたか

- 9-patchを無駄なく1pxの画像にした
- xxhdpiしか用意していなかった
  - 縮小方向なら品質的な問題も起こりにくい

なにが起こっていたか

- 9-patchを無駄なく1pxの画像にした
- xxhdpiしか用意していなかった
  - 縮小方向なら品質的な問題も起こりにくい

- 1pxの画像を縮小すると？

  ＼(^o^)／

対策

- 解像度ごとのリソースを用意する
- mdpiに縮小しても問題無い画像を利用する

アップデートで
クラッシュ
その1

getStringExtraでClassNotFoundException？

```
Caused by: java.lang.RuntimeException: Parcelable encountered
ClassNotFoundException reading a Serializable object (name = xxx)
    at android.os.Parcel.readSerializable(Parcel.java:2378)
    at android.os.Parcel.readValue(Parcel.java:2197)
    at android.os.Parcel.readArrayMapInternal(Parcel.java:2479)
    at android.os.BaseBundle.unparcel(BaseBundle.java:221)
    at android.os.BaseBundle.getString(BaseBundle.java:918)
    at android.content.Intent.getStringExtra(Intent.java:4816)
    at ...
```

getStringExtraでClassNotFoundException？

Enum?

```
Caused by: java.lang.RuntimeException: Parcelable encountered
ClassNotFoundException reading a Serializable object (name = xxx)
    at android.os.Parcel.readSerializable(Parcel.java:2378)
    at android.os.Parcel.readValue(Parcel.java:2197)
    at android.os.Parcel.readArrayMapInternal(Parcel.java:2479)
    at android.os.BaseBundle.unparcel(BaseBundle.java:221)
    at android.os.BaseBundle.getString(BaseBundle.java:918)
    at android.content.Intent.getStringExtra(Intent.java:4816)
    at ...
```

```
val intent = Intent(ACTION_HOGE)
intent.putExtra(EXTRA_KEY_FUGA, VALUE_FUGA)
val pendingIntent
    = PendingIntent.getBroadcast(this, 1, intent, 0)
```

```
val value = intent.getStringExtra(EXTRA_KEY_FUGA)
```

Stringを設定

```
val intent = Intent(ACTION_HOGE)
intent.putExtra(EXTRA_KEY_FUGA, VALUE_FUGA)
val pendingIntent
    = PendingIntent.getBroadcast(this, 1, intent, 0)
```

```
val value = intent.getStringExtra(EXTRA_KEY_FUGA)
```

Stringを設定

```
val intent = Intent(ACTION_HOGE)
intent.putExtra(EXTRA_KEY_FUGA, VALUE_FUGA)
val pendingIntent
    = PendingIntent.getBroadcast(this, 1, intent, 0)
```

Stringを読み出し

```
val value = intent.getStringExtra(EXTRA_KEY_FUGA)
```

どこに問題が？

前バージョン

```
val intent = Intent(ACTION_HOGE)
intent.putExtra(EXTRA_KEY_HOGE, HogeEnum.HOGE)
val pendingIntent
    = PendingIntent.getBroadcast(this, 1, intent, 0)
```

```
val value = intent.getSerializableExtra(EXTRA_KEY_HOGE)
```

前バージョン

Enumを設定

```
val intent = Intent(ACTION_HOGE)
intent.putExtra(EXTRA_KEY_HOGE, HogeEnum.HOGE)
val pendingIntent
    = PendingIntent.getBroadcast(this, 1, intent, 0)
```

```
val value = intent.getSerializableExtra(EXTRA_KEY_HOGE)
```

前バージョン

Enumを設定

```
val intent = Intent(ACTION_HOGE)
intent.putExtra(EXTRA_KEY_HOGE, HogeEnum.HOGE)
val pendingIntent
    = PendingIntent.getBroadcast(this, 1, intent, 0)
```

Serializableとして読み出し

```
val value = intent.getSerializableExtra(EXTRA_KEY_HOGE)
```

前バージョン

これが見つからない？

Enumを設定

```
val intent = Intent(ACTION_HOGE)
intent.putExtra(EXTRA_KEY_HOGE, HogeEnum.HOGE)
val pendingIntent
    = PendingIntent.getBroadcast(this, 1, intent, 0)
```

Serializableとして読み出し

```
val value = intent.getSerializableExtra(EXTRA_KEY_HOGE)
```

前バージョン

これが見つからない？

Enumを設定

```
val intent = Intent(ACTION_HOGE)
intent.putExtra(EXTRA_KEY_HOGE, HogeEnum.HOGE)
val pendingIntent
    = PendingIntent.getBroadcast(this, 1, intent, 0)
```

Serializableとして読み出し

```
val value = intent.getSerializableExtra(EXTRA_KEY_HOGE)
```

Keyも違うよ？

前バージョンのPendingIntentを
新バージョンで受け取った？

…だとしても読み出ししていないよ？

ソースを追ってみる

Intent

```
public @Nullable String getStringExtra(String name) {
    return mExtras == null ? null : mExtras.getString(name);
}
```

ソースを追ってみる

Intent

```
public @Nullable String getStringExtra(String name) {
    return mExtras == null ? null : mExtras.getString(name);
}
```

Bundle

```
@Nullable
public String getString(@Nullable String key) {
    unparcel();
    final Object o = mMap.get(key);
    try {
        return (String) o;
    } catch (ClassCastException e) {
        typeWarning(key, o, "String", e);
        return null;
    }
}
```

ソースを追ってみる

Bundle

```
void unparcel() {
    synchronized (this) {
        final Parcel source = mParcelledData;
        if (source != null) {
            initializeFromParcelLocked(source, true, mParcelledByNative);
        } else {
            if (DEBUG) {
                Log.d(TAG, "unparcel "
                        + Integer.toHexString(System.identityHashCode(this))
                        + ": no parcelled data");
            }
        }
    }
}
```

IntentのgetXXXの挙動

- Bundle（Extras）から読み出す

IntentのgetXXXの挙動

- Bundle（Extras）から読み出す
- Bundleは内部データがunparcel前であれば、
  内部データ全体をunparcelする

IntentのgetXXXの挙動

- Bundle（Extras）から読み出す
- Bundleは内部データがunparcel前であれば、内部データ全体をunparcelする

- Extraの中に一つでもunparcel/deserializeできないデータが含まれているとException

対策

- PendingIntentを含む、アプリ外へ出て行くIntentに独自クラスを入れることを原則禁止
- 先にActionなどで分岐
- アプリ外から受け取るIntentのExtraに不用意にアクセスしない
  - 読み出す場合はtry/catch必須

try/catch付きの拡張関数を用意するとか

```
fun Intent.getBooleanExtraSafely(key: String, default: Boolean): Boolean =
    runCatching { getBooleanExtra(key, default) }.getOrDefault(default)

fun Intent.getIntExtraSafely(key: String, default: Int = 0): Int =
    runCatching { getIntExtra(key, default) }.getOrDefault(default)
```

可能性の話

- 悪意あるIntentを投げることもできてしまう
  - 適当なSerializableをIntentにいれて投げるだけ
  - 対策をしていなければクラッシュ
  - 意図せず加害者になる可能性も

可能性の話

- 悪意あるIntentを投げることもできてしまう
  - 適当なSerializableをIntentにいれて投げるだけ
  - 対策をしていなければクラッシュ
  - 意図せず加害者になる可能性も

アップデートで
クラッシュ
その2

SharedPreferencesで ClassCastException !?

```
java.lang.ClassCastException: java.lang.Boolean cannot
be cast to java.lang.Integer
    at android.app.SharedPreferencesImpl.getInt(
SharedPreferencesImpl.java:302)
```

何が起こったか？

過去バージョン

```
sharedPreferences
    .getBoolean("HOGE_HOGE", false)
```

何が起こったか？

過去バージョン

```
sharedPreferences
        .getBoolean("HOGE_HOGE", false)
```

削除

何が起こったか？

過去バージョン

```
sharedPreferences
        .getBoolean("HOGE_HOGE", false)
```

削除

```
sharedPreferences
        .getInt("HOGE_HOGE", 0)
```

新バージョン

何が起こったか？

過去バージョン

```
sharedPreferences
        .getBoolean("HOGE_HOGE", false)
```

削除

同じKey!

```
sharedPreferences
        .getInt("HOGE_HOGE", 0)
```

新バージョン

要するに

- 同バージョン内では整合性がとれている
  - ユニットテストなどでは問題なし

要するに

- 同バージョン内では整合性がとれている
  - ユニットテストなどでは問題なし
- 前バージョンには存在しないキー
  - アップデートを含むシナリオテストでも問題なし

要するに

- 同バージョン内では整合性がとれている
  - ユニットテストなどでは問題なし
- 前バージョンには存在しないキー
  - アップデートを含むシナリオテストでも問題なし
- 一般ユーザー：削除前からのユーザー多数

  ＼(^o^)／

SharedPreferences

- 自由度が高すぎ、型安全でない

```
private Map<String, Object> mMap;

public String getString(String key, @Nullable String defValue) {
    synchronized (mLock) {
        awaitLoadedLocked();
        String v = (String)mMap.get(key);
        return v != null ? v : defValue;
    }
}
```

SharedPreferences

- Keyを適切に管理する責任はアプリ側にある
- 内容はアプリをアップデートしても残り続ける
- 無法地帯化しやすい

対策

1. KeyをEnumで管理

```
class Preferences<K>(
    context: Context,
    kClass: KClass<K>
) where K : Enum<*>,
        K : Key {
    private val sharedPreferences: SharedPreferences =
        context.getSharedPreferences(kClass.simpleName, MODE_PRIVATE)

    fun writeBoolean(key: K, value: Boolean) {
        if (BuildConfig.DEBUG) key.checkSuffix(value)
        sharedPreferences.edit().putBoolean(key.name, value).apply()
    }

    fun readBoolean(key: K, defaultValue: Boolean): Boolean = ...
    fun writeInt(key: K, value: Int): Unit = ...
    fun readInt(key: K, defaultValue: Int) = ...
```

```
interface Key {
    enum class Main : Key {
        PREFERENCES_VERSION_INT,
        KEY_BOOLEAN,
        KEY_INT,
        KEY_LONG,
        KEY_FLOAT,
        @Deprecated("removed:v1.1.1")
        KEY_STRING,
        ...
    }
    enum class Temp : Key {
        PREFERENCES_VERSION_INT,
        ...
```

2. Key名に型名をつける

カッコ悪い……

```
interface Key {
    enum class Main : Key {
        PREFERENCES_VERSION_INT,
        KEY_BOOLEAN,
        KEY_INT,
        KEY_LONG,
        KEY_FLOAT,
        @Deprecated("removed:v1.1.1")
        KEY_STRING,
        ...
    }
    enum class Temp : Key {
        PREFERENCES_VERSION_INT,
        ...
```

```
interface Key {
    enum class Main : Key {
        PREFERENCES_VERSION_INT,
        KEY_BOOLEAN,
        KEY_INT,
        KEY_LONG,
        KEY_FLOAT,
        @Deprecated("removed:v1.1.1")
        KEY_STRING,
        ...
    }
    enum class Temp : Key {
        PREFERENCES_VERSION_INT,
        ...
```

2. Key名に型名をつける

カッコ悪い……

3. 使わなくなったキーを削除しないことをルール化

```
interface Key {
    enum class Main : Key {
        PREFERENCES_VERSION_INT,
        KEY_BOOLEAN,
        KEY_INT,
        KEY_LONG,
        KEY_FLOAT,
        @Deprecated("removed:v1.1.1")
        KEY_STRING,
        ...
    }
    enum class Temp : Key {
        PREFERENCES_VERSION_INT,
        ...
```

2. Key名に型名をつける

カッコ悪い……

3. 使わなくなったキーを削除しないことをルール化

4. 過去を清算しやすく

- バージョンをつける
- 一時的設定値を分離

5. アプリ全体からはKeyを隠蔽

```
class PreferenceService(
    private val main: Preferences<Main>,
    private val temp: Preferences<Temp>
) {
    fun getHoge(): Int =
        main.readInt(Main.KEY_INT, 0)

    fun setHoge(value: Int) =
        main.writeInt(Main.KEY_INT, value)
```

```
fun writeBoolean(key: K, value: Boolean) {
    if (BuildConfig.DEBUG) key.checkSuffix(value)
    sharedPreferences.edit().putBoolean(key.name, value).apply()
}
```

6. 名前ルールは実行時チェック

```
internal fun Enum<*>.checkSuffix(value: Any) {
    when (value) {
        is Boolean -> require(name.endsWith(SUFFIX_BOOLEAN))
        is Int -> require(name.endsWith(SUFFIX_INT))
        is Long -> require(name.endsWith(SUFFIX_LONG))
        is Float -> require(name.endsWith(SUFFIX_FLOAT))
        is String -> require(name.endsWith(SUFFIX_STRING))
    }
}
```

マイグレーションミス

永続化データのマイグレーションミス

- 忘れがち
  - バージョンは一つ一つ上がるわけではない
  - 1→2、2→3のマイグレーションを想定していても1→3が正しく動作しているか？
- 過去の清算も計画的に
  - マイグレーションを実装するときに、いつまでマイグレーションをサポートするかを決めておく
  - 一定以上古い場合は全クリアした方が安全

消えたリソース

shrinkResource

- リソースをID参照していない
  - getIdentifier

```
resources.getIdentifier(
    "ic_launcher", "drawable", packageName)
```

  - パス参照

```
"file:///android_res/drawable/ic_launcher.png"
```

getIdentifierの罠

- 連番のついたリソース

```
(1..9).map {
    resources.getIdentifier(
        "ic_$it", "drawable", packageName)
}
```

  - どのリソースが使われているのか分からない！

対策

- getIdentifierを原則禁止
  - ローカルリソースは必ずIDで参照する
  - 連番リソースもarrayやmapなどで保持する
- パス参照
  - ID参照にする
  - assetsに移動

Gravityの罠

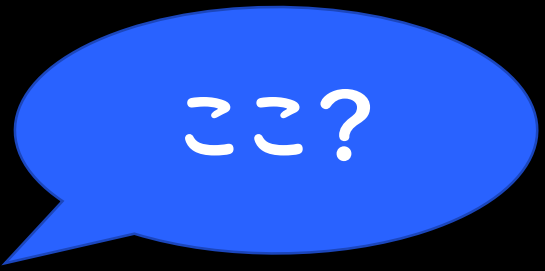
SlideでInvalid slide direction?

```
java.lang.IllegalArgumentException: Invalid slide direction
    at android.transition.Slide.setSlideEdge(Slide.java:165)
    at android.transition.Slide.<init>(Slide.java:112)
    at ...
```

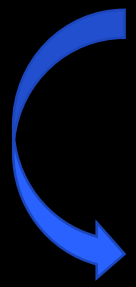
何が起こっている？

```
if (animate) {
    val transition = Slide(Gravity.END)
        .setDuration(150L)
        .setInterpolator(DecelerateInterpolator())
    fragment.enterTransition = transition
}
activity.supportFragmentManager.beginTransaction()
    .replace(R.id.server_detail_container, fragment)
    .commitAllowingStateLoss()
```

何が起こっている？

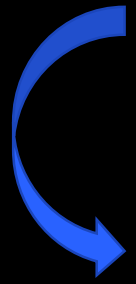



```
if (animate) {
    val transition = Slide(Gravity.END)
        .setDuration(150L)
        .setInterpolator(DecelerateInterpolator())
    fragment.enterTransition = transition
}
activity.supportFragmentManager.beginTransaction()
    .replace(R.id.server_detail_container, fragment)
    .commitAllowingStateLoss()
```



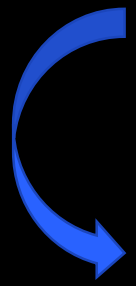
```java
public Slide(@GravityFlag int slideEdge) {
    setSlideEdge(slideEdge);
}

public void setSlideEdge(@GravityFlag int slideEdge) {
    switch (slideEdge) {
        case Gravity.LEFT:
            mSlideCalculator = sCalculateLeft; break;
        case Gravity.TOP:
            mSlideCalculator = sCalculateTop; break;
        case Gravity.RIGHT:
            mSlideCalculator = sCalculateRight; break;
        case Gravity.BOTTOM:
            mSlideCalculator = sCalculateBottom; break;
        case Gravity.START:
            mSlideCalculator = sCalculateStart; break;
        case Gravity.END:
            mSlideCalculator = sCalculateEnd; break;
        default:
            throw new IllegalArgumentException("Invalid slide direction");
```

```java
public Slide(@GravityFlag int slideEdge) {
    setSlideEdge(slideEdge);
}
```



```java
public void setSlideEdge(@GravityFlag int slideEdge) {
    switch (slideEdge) {
        case Gravity.LEFT:
            mSlideCalculator = sCalculateLeft; break;
        case Gravity.TOP:
            mSlideCalculator = sCalculateTop; break;
        case Gravity.RIGHT:
            mSlideCalculator = sCalculateRight; break;
        case Gravity.BOTTOM:
            mSlideCalculator = sCalculateBottom; break;
        case Gravity.START:
            mSlideCalculator = sCalculateStart; break;
        case Gravity.END:
            mSlideCalculator = sCalculateEnd; break;
        default:
            throw new IllegalArgumentException("Invalid slide direction");
```

問題無い？



```java
public Slide(@GravityFlag int slideEdge) {
    setSlideEdge(slideEdge);
}

public void setSlideEdge(@GravityFlag int slideEdge) {
    switch (slideEdge) {
        case Gravity.LEFT:
            mSlideCalculator = sCalculateLeft; break;
        case Gravity.TOP:
            mSlideCalculator = sCalculateTop; break;
        case Gravity.RIGHT:
            mSlideCalculator = sCalculateRight; break;
        case Gravity.BOTTOM:
            mSlideCalculator = sCalculateBottom; break;
        case Gravity.START:
            mSlideCalculator = sCalculateStart; break;
        case Gravity.END:
            mSlideCalculator = sCalculateEnd; break;
        default:
            throw new IllegalArgumentException("Invalid slide direction");
```

問題無い？

API Level？

```
public void setSlideEdge(int slideEdge) {
    switch (slideEdge) {
        case Gravity.LEFT:
            mSlideCalculator = sCalculateLeft;
            break;
        case Gravity.TOP:
            mSlideCalculator = sCalculateTop;
            break;
        case Gravity.RIGHT:
            mSlideCalculator = sCalculateRight;
            break;
        case Gravity.BOTTOM:
            mSlideCalculator = sCalculateBottom;
            break;
        default:
            throw new IllegalArgumentException("Invalid slide direction");
    }
```

```
public void setSlideEdge(int slideEdge) {
    switch (slideEdge) {
        case Gravity.LEFT:
            mSlideCalculator = sCalculateLeft;
            break;
        case Gravity.TOP:
            mSlideCalculator = sCalculateTop;
            break;
        case Gravity.RIGHT:
            mSlideCalculator = sCalculateRight;
            break;
        case Gravity.BOTTOM:
            mSlideCalculator = sCalculateBottom;
            break;
        default:
            throw new IllegalArgumentException("Invalid slide direction");
    }
```

API Level 21

時系列

- GravityにSTART/ENDが追加された　：API 14
- LayoutでStart/End指定が追加された：API 17
- android.transion.Slideが追加された：API 21
- SlideがSTART/ENDに対応した　　：API 22

対策

- Jetpackを使いましょう
- SDKにも実装ミスがありうることを頭の片隅に

おまけ：start/endとleft/rightは違うよ

- 当たり前だけどね
  - コード上から操作するときも、start/endとleft/rightの使い分けきちんとできてる？
  - setPaddingRelative
  - setPadding
  - setCompoundDrawablesRelative
  - setCompoundDrawables

エモい話

エモい話

- やらかさない人はいない
  - 人間はミスを犯すもの
  - 開発はルーチンワークではなく、チャレンジの連続
- 開発は必ずしも万全の状態で行えない
  - 時間が無い・人員が足りない
  - スキルが足りない・秘伝のタレ
- 失敗を過度に恐れない、失敗から学べばよい
  - どんな凄腕エンジニアもはじめは初心者

エモいまとめ

- 慎重になるべき部分を見極める
  - 問題が出やすい部分
  - 問題が出たときに取り返しがつかない部分
- 特に注意するべき部分
  - 外部とのプロトコル
  - 永続化データ

エモいまとめ

- コードを改善すればミスも減る
  - コードの状態は伝搬する
    - 悪いコードは悪いコードを増やす
    - よいコードはよいコードを増やす
  - リファクタリングはエンジニアの当然の義務

- 過去を清算できる仕組みを仕込む
  - いつサポートを切るか？

エモいまとめ

- 失敗してしまったら
  - 何が問題だったか、現実的な再発防止策を考える
  - 「再発防止策を行わない」という結論も重要

- 失敗を乗り越えたエンジニアは強い！
  - 人の失敗事例は他山の石とする

キサマ等がやらかしそうに
なったミスは
既に私が1年前に
通過したミスだッッッ！

ありがとうございました