Android (Kotlin) Conventions

Structure

• Kotlin structure

```Kotlin
vn.asiantech.project
    - api
    - models
    - managers
    - utils
    - fragments
    - service
    - interfaces
    - ui
        + screen_name_1
        + screen_name_2
    - views
```

• Resource structure

Name	Path	Description
XML Layouts	`res/layout/`	This is where we put our XML layout files.
XML Menus	`res/menu/`	This is where we put our AppBar menu actions.
Drawables	`res/drawable`	This is where we put images and XML drawables.
Colors	`res/values/colors.xml`	This is where we put color definitions.
Dimensions	`res/values/dimens.xml`	This is where we put dimension values.
String	`res/values/strings.xml`	This is where we put strings.
Styles	`res/values/styles.xml`	This is where we put style values.

Class

- First letter of classes is UpperCase.
- Name of class only accept in range **[A-Z], [a-z]** and follow **Camel Case**.
- Classes name must be **noun**.
- For classes that extend an Android component, the name of the class should end with the name of the component; for example: SignInActivity, SignInFragment, ImageUploaderService, ChangePasswordDialog.
- Open brace ‘ { ‘ appears at the end of the same line as the declaration statement.
- Closing brace ‘ } ‘ starts a line by itself indented to match its corresponding opening statement, except when it is a null statement the ‘ } ‘ should appear immediately after the ‘ { ‘
- Write KotlinDoc for each class, Kotlin doc must define what are classes working for.

See Example

```kotlin
/**
* Copyright © AsianTech Co., Ltd
* Created by toannt on 06/09/2017.
* SomeClass show information of user
*/
class SomeClass {
...
}
```

- Class member ordering There is no single correct solution for this but using a **logical** and **consistent** order will improve code learnability and readability. It is recommendable to use the following order:

    1. Constants
    2. Fields
    3. Constructor
    4. Override method and callbacks
    5. Public method
    6. Internal method
    7. Private method
    8. Inner classes and interfaces

`( TODO: inner class can be change position and some positions others)` **Example**:

```kotlin
class MainActivity : Activity() {

  var mTitle : String
  var mTextViewTitle : TextView

  Override fun onCreate() {
        . . .
  }

  internal fun methodName () : Int {
        . . .
  }

  private fun setUpView() {
        . . .
  }
}
```

If your class is extending an **Android components** such as an Activity or a Fragment, it is a good practice to order the override methods so that they **match the component's lifecycle**. For example, if you have an Activity that implements **onCreate()**, **onDestroy()**, **onPause()** and **onResume()**, then the correct order is:

```kotlin
class MainActivity : Activity(){
 // Order matches Activity lifecycle
 Override fun onCreate() {}

 Override fun onResume() {}

 Override fun onPause() {}

 Override fun onDestroy() {}
}
```

Method

- Short method content Method content should be short and focus on the feature of method. Avoid repeating code.
- Code line not over 80 characters. Except for inputting text or URL.
- Method name must start with verb is first letter.
- First letter of method name is **LOWERCASE**.
- @SuppressWarnings: The ***@SuppserssWarnings*** annotation should only be used under circumstances

where it is impossible to eliminate a warning. If a warning passes this ”impossible to eliminate” test, the ***@SuppressWarnings*** annotation must be used, so as to ensure that all warnings reflect actual problems in the code.

Example

- Write document for methods if it’s necessary.

```kotlin
/**
* Format date
*
* @param dateFormat Date need format
* @return the date formatted
* @throw ParseException exception
*/
fun formatDate(dateFormat: String): String{
 …
}
```

- Limit method block line less than **300** lines, limit method arguments less than **5**. Separate code if function has long line method.
- If statement convention.

GOOD

```kotlin
 if (…) {
  doSomething()
 }
```

BAD

```kotlin
if (…)
doSomething()
```

- Use `// TODO` for dummy data or need to change later.

```kotlin
// TODO Remove after api finish
```

- Catch exception.

GOOD

```kotlin
try {
    …
} catch (et1 : ExceptionType1) {
    …
} catch (et2 : ExceptionType2) {
    …
}
```

BAD

```kotlin
try {
    …
} catch (e : Exception) {
    …
}
```

Variable:

`TODO: Update more in future` * All letters of constant name is UPPERCASE

```kotlin
companion object {
    const val GOOGLE_HOME_URL = "http://www.google.com"
}
```

XML Conventions

- View ID is followed by camel-case, example:

```kotlin
btnLogin, tvCaption
```

- Prefix for id in XML

Name	Prefix	Name	Prefix
`Button`	`btn`	`Datepicker`	`datePicker`
`EditText`	`edt`	`Timepicker`	`timePicker`
`TextView`	`tv`	`Videoview`	`videoView`
`Checkbox`	`chk`	`Seekbar`	`seekBar`
`RadioButton`	`rb`	`RatingBar`	`ratingBar`
`ToggleButton`	`tb`	`Processbar`	`processBar`
`Spinner`	`spn`	`ImageView`	`img`
`Menu`	`menu`	`ImageButton`	`imgBtn`
`ListView`	`lv`	`RecyclerView`	`recycleView`
`GalleryView`	`gv`	`CardView`	`cardView`
`LinearLayout`	`ll`	`ScrollView`	`scrollView`
`RelativeLayout`	`rl`	`ToolBar`	`toolbar`
`Calendar`	`calendar`		

- XML File Name:

    1. Item for list view: `item_list _*`
    2. Item for grid view: `item_grid _*`
    3. Custom for view: `custom_*`
    4. Custom for dialog: `dialog_*`

Example:

```Markup
@+id/tvSubjectQuestions
@+id/ivTakeCamera
```

- Image name
    1. Icon: `ic_*`
    2. Background: `bg_*`
    3. Image: `img_*`

- Other naming
    1. Model (example: Student, Teacher)
    2. Activity: `*Activity` (example: UserActivity)
    3. Fragment: `*Fragment` (example: HomeFragment)
    4. Apdater: `*Adapter` (example: PageAdapter)

- Do not make a deep hierarchy of `ViewGroups` . here
- Also keep dimens.xml DRY, define generic constants. here
- Use multiple style files to avoid a single huge one. here
- When an XML element doesn't have any contents, you must use self closing tags.

GOOD

Markup
```
<TextView
    android: id="@+id/text_view_profile"
    android: layout_width="wrap_content"
    android: layout_height="wrap_content" />
```

BAD

Markup
```
<!-- Don\'t do this! -->
<TextView
    android: id="@+id/text_view_profile"
    android: layout_width="wrap_content"
    android: layout_height="wrap_content" >
</TextView>
```

Environments

- Android Studio. download
- Setup Kotlin plugin.

Framework Conventions (TODO: Will update later)

Security Conventions

- Remember cancel Thread, Asyntask, …if go out other screen or turn back home screen.
- Remember enable minifyEnabled function in build.gradle while on release mode and config proguard

carefully.

```
    buildTypes {
        release {
            debuggable false
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt' ), 'progu
            applicationVariants. all { variant ->
            appendVersionNameVersionCode(variant)
            }
        }
    }
```

- If you guys create keystore by yourself, let talk to PM or TL about that, and having backup that keystore carefully and remember information of keystore (keyAlias, keyPassword).
- Don't push keystore file and information of keystore to GitHub, keep it in local as in local.properties file.
- More practice:

    1. Proguard configure
    2. Gradle configure

local.properties file

```
sdk. dir=/Users/Administrator/Library/Android/sdk
keyAlias=sampletext1
keyPassword= sampletext2
```

- Verify input validation carefully.
- Verify Runtime Permission function available in Android 6.0 and above carefully. More details
- Avoid leak memory. More details
- Avoid out of memory. More details
- Avoid run time exception

Code management practices and Dependency management practices

- Introduce Gradle Tool

References

- Android practices Github
- KotlinLang
- Android Developer Page
- Kotlin Code Coventions Page

Code review Checklist

- Pass Circle CI or Travis CI with checktyle findbug lint tools: 70% follow convention.
- Pass Reviewer: 30% follow convention.
- Github

More References

- Android Boilerplate
- Android folder structure