Android Development Standard

CONTENT OWNER:
Sutrix Media
All future revisions to this document shall be approved by the content owner prior to release.
The information contained herein is PROPRIETARY to The Sutrixmedia Joint Stock Company
and shall not be reproduced or disclosed in whole or in part or used for any purpose except when the user possesses direct,
written authorization from
The Sutrixmedia Joint Stock Company

Preface

❖ Signature

Originator By: Le Duc Tuan	**Date: 04/05/2015**
Prepared By: Le Duc Tuan	**Date: 04/05/2015**
Approved By: Hai Nguyen	**Date: 04/05/2015**
Reviewed By: Hai Nguyen	
Distributed To: Android team	

Preface

❖ Signature Revision History

***A – Added, M – Modified, D - Deleted**

Version	Date	A*, M, D	Change Description	Author	Approved By
1.0	04/05/2015	A	• Create document	Le Duc Tuan	

Preface

- Convention Used in this document

I. Goals

Goals

- Working in the same bases and conventions

- Ease of project backup

- Ease of maintenance

II. Language - English

Language - English

Conventions:

- All class, method and variable names are in English and follow below rules
- Give simplest and shortest names as possible
- Avoid using plural (use collection type instead)
- Words are lowercase when alone, otherwise use CamelCase
- Give the generic term as possible, avoid gender and false specialization
- Avoid confusing acronyms, use full text to describe the meaning

Examples (Do # Don't):

- performSearch() # ~~recherche()~~
- createUserList() # ~~createListOfUsers()~~
- mCatList # ~~mCats~~
- person, personAge, personWithName
- PersonClass # ~~ManClass,WomanClass~~
- HTTP, URL # ~~AAA (Batter Type/Codon Lysine, etc.)~~

III. Project Name

Project Name

Conventions:

- The project must follow client requirement first, otherwise use this convention
- Each project must be identified by a prefix and postfix (if required) as the rules follow

  [Company Name]Project Name[Version]
- No duplications with another project

Examples (Do # Don't):

- DigitasYouPlusV4 # ~~YouPlus~~
- KeyneosoftMyTao # ~~MyTao~~
- KeyneosoftAG2R # ~~AG2R~~

IV. Naming Convention

Naming Conventions

- This section will describe the naming convention of following
  - Application package name
  - Classes & members (variables)
  - Accessor members
  - Methods & parameters
  - Implementation
  - Listeners
  - Constants
  - Layouts
  - Values
  - Identifiers

Application Package Name

Conventions:

- Use the package name assigned by client or use this convention
- All lowercase and as follow

  [com.][company name.][project name]
- Package separated by application functions
- Any other packages that is commonly used throughout the project must be put into [common] package

Examples:

Do

- com.ag2r (company name is omitted)
  - com.ag2r.common.gcm
  - com.ag2r.about
- com.digitas.nissaneu.mobile.yp.android
- com.sutrixmedia.application

Don't

- ~~com.facebook.katana~~ (duplicated with facebook application)
- ~~sutrixmedia.application~~ (missing com.)

Classes & Members

Conventions: Class name

- Start with company name/project abbreviation and use CamelCase

- End with class type or functional usage

  [company/project name][class type]

Examples:

Do

- AG2RSessionFragment
- AG2RLoginResult
- TAOWebviewFragment

Don't

- ~~HomeFragment~~
- ~~MainActivity~~
- ~~MainView~~
- ~~UpdateUserInfo~~

Classes & Members (cont.)

Conventions: Class member name (global)

- Start with “m” and use CamelCase
- Avoid using numbers, add functional usage or purpose of the variable instead
- For boolean type, the prefix followed by “is” and end with an adjective

Examples:

Do

- mProduct
- mProductList
- mIsProductChecked

Don’t

- boolean: ~~checked~~ (not specific and missing “m” and “is” for the prefix)
- ~~mProduct3~~ (no numbers, purpose instead)
- ~~mainProduct~~ (missing “m” for the prefix)

Accessor Members

Conventions:

- Remove “m” for both getters and setters
- Setter:
  - Followed by variable name
  - Use CamelCase
- Getter:
  - Followed by variable name
  - Use CamelCase
  - For boolean type, “is” followed by the variable name

Examples:

Do

- setProduct(), getProduct()
- setProductList(), getProductList()
- setProductChecked(), isProductChecked()

Don’t

- ~~setproduct(), getproduct(), SetProduct()~~
- ~~productChecked()~~
- ~~isProduct(), isMask()~~

Methods & Parameters

Conventions: Methods (instance & class)

- Start with a lowercase verb/“is”
- Use CamelCase
- Avoid abbreviation
- Give simplest and shortest method name as possible

Examples:

Do

- performSearch(), buildWhereClause()
- searchProductById(), findViewById()

Don’t

- ~~checkProduct()~~
- ~~setproduct(), getproduct(), SetProduct()~~
- ~~isProduct(), isMask()~~

Methods & Parameters (cont.)

Conventions: Parameters

- Use CamelCase
- Avoid abbreviation
- Give simplest and shortest name as possible

Examples:

Do

- product, productList
- name, id, isVisible

Don't

- ~~Product, mProduct~~
- ~~Isvisbile~~

Implementation

Conventions: Local variables

- Start with “t”
- Use CamelCase
- Avoid abbreviation
- Give simplest and shortest name as possible

Examples:

Do

- tProduct, tProductList
- tName, tId, tIsVisible

Don’t

- ~~product, Product, mProduct~~
- ~~TIsvisbile~~

Implementation (cont.)

Conventions: Loop variables

- Use single letter
- The common is to use “i - j - k - l” with the increment order
- Should not be more than 4 nested loops, otherwise the implementation logic is too complicated and might be wrong

Examples:

Do

- for (int i …) , for (int j …)

Don’t

- ~~for (int ii …)~~
- ~~for (int j …), for (int i …)~~
- ~~for (int ij …), for (int ik …)~~

Listeners

Conventions:

- Same rule as class naming convention
- Post fix with “Listener” (optional)
- Methods in the listener follow same rule as method naming convention
- The name should explain what the listener does (action/behavior)
- Make it a separated interface

Examples:

Do

- AG2RLocationUpdatedListener
- YouPlusNotificationReceivedListener
- WebserviceResultHandler

Don’t

- ~~MyListener~~

Constants

Conventions:

- Start with company name/project abbreviation
- All uppercases
- Each term separated by an "_"
- Declared as "static final" variables

Examples:

Do

- AG2R_MIN_WIDTH
- YOUPLUS_MAX_COUNT
- AG2R_PREF_PROGRAM_ID

Don't

- ~~ag2r_min_width~~
- ~~YOUPLUSMAXCOUNT~~
- ~~Ag2rPrefProgramId~~

Layouts

Conventions:

- Start with company name/project abbreviation
- Layout bind with Activity use prefix “activity”, bind with Fragment use “fragment”, otherwise use “layout”
- Followed by name of binding class or layout purpose
- All lowercase
- Each term separated by an “_”

Examples:

Do

- ag2r_activity_home.xml
- ag2r_fragment_session_result.xml
- ag2r_layout_home_header.xml

Don’t

- ~~home_activity.xml~~
- ~~fragment_about.xml~~
- ~~layoutListviewFooter.xml~~

Values

Conventions:

- Start with company name/project abbreviation
- Name itself should explain the purpose
- All lowercase
- Each term separated by an “_”
- End with “values”

Examples:

Do

- ag2r_string_values.xml
- ag2r_array_values.xml
- ag2r_dimension_values.xml

Don’t

- ~~strings.xml~~
- ~~arrays.xml~~
- ~~dimensions.xml~~

Identifiers

Conventions:

- Start with layout file name (exclude the company name/project abbreviation)
- Followed by 2 letters describing the view (tv - TextView ,bt - Button, et - EditText, rl - RelativeLayout)
- End with purpose (action/behavior)
- All lowercase
- Each term separated by an "_"
- Absolutely no textView1, textView2, etc.

[activity/fragment/layout]_screen_view_function]

Examples:

Do

- activity_home_bt_login
- fragment_login_bt_signin
- layout_main_header_tv_username

Don't

- ~~buttonLogin~~
- ~~username_txt~~
- ~~view_title~~
- ~~textView1~~

Thank you