Style guide

1. Introduction

This document serves as the definition of our group's coding standard for the source code in our mobile application.

2. Source File Basics

2.1 File name

All Java files are named using Pascal case.

Example: MainActivity.java

All XML files are named using snake case.

Example: fragment_main.xml

2.2 Special Characters

2.2.1 Whitespace Characters

Tab characters are used for indentation.

2.2.2 Special escape sequences

For any character that has a special escape sequence (\n, \b, \t, \\), we use that sequence rather than other forms.

3. Source file structure

3.1 UI elements

3.1.1 Activity Source Files

All activities are stored within the activity's folder

3.1.2 Fragment Source Files

All fragments are stored within the fragment's folder

3.2 Database Source Files

All files containing functions that modify or access the database are stored within the database folder

3.3 XML Files

All XML files are stored within the res folder.

4. Formatting

4.1 Braces

4.1.1 Braces are used where optional

4.1.2 Nonempty blocks

Braces follow the Kernighan and Ritchie style ("Egyptian brackets") for nonempty blocks and block-like constructs

Example:

```
public static void main() {
    System.out.println("Hello world");
}
```

4.2 Block indentation: +1 tab

4.3 One statement per line

4.4 Column limit: 100

4.5 Line-wrapping

Use line-wrapping to avoid overflowing the column limit.

4.6 Whitespace

4.6.1 Vertical Whitespace

Put whitespace after initializing a group of variables and at the end of a function.

4.6.2 Horizontal Whitespace

Put whitespace before and after operators (+, - , =, !=, &&, >), but not dot separator and two colons.

Put whitespace after commas.

4.6.3 Horizontal alignment: never required

4.7 Specific constructs

4.7.1 Variable declarations

4.7.1.1 One variable per declaration

Every variable declaration declares only one variable.

Exception: Multiple variable declaration are acceptable in the header of a *for* loop.

4.7.1.1 Declared when needed

Local variables are declared close to when they are first used.

4.7.2 Arrays

4.7.2.1 Array initializers can be "block-like"

Any array initializer may optionally be formatted as if it were a "block-like Construct"

4.7.2.2 No C-style array declarations

The square brackets form a part of the type, not the variable: String[] args, not String args[].

4.7.3 Switch statements

4.7.3.1 Fall-through: commented

4.7.3.2 The *default* case can be present.

4.7.4 Comments

4.7.4.1 Block comment style

They may be in /* … */ style or // … style. For multi-line /* … */ comments, subsequent lines must start with * aligned with the * on the previous line.

5. Naming

5.2 Rules by identifier type

5.2.1 Class names

Class names are written in UpperCamelCase.

5.2.2 Method names

Method names are written in lowerCamelCase

5.2.3 Constant names

Constant names use CONSTANT_CASE : all uppercase letters, with each word separated from the next by a single underscore.

5.2.5 Non-constant field names

Non-constant field names are written in lowerCamelCase.

5.2.6 Parameter names

Parameter names are written in lowerCamelCase

5.2.7 Local variable names

Local variable names are written in lowerCamelCase.

6. Programming Practices

6.1 @*Override* : always used

A method is marked with the @*Override* annotation whenever it is legal.

6.2 Caught exceptions: not ignored