Coding-Convention-Report
For
Vaccination App
Group#9

Mentor: Dr. Sarmad Ali
Client: Dr. Noman Javed

Group Members:

1. Muhammad Imtiaz (15026420)
2. Muhammad Haroon (14031184)
3. Zainab Malik (15026433)
4. Bilal Yousaf (14031236)
5. Atiq Zaman (15026391)

Contents

1. Introduction

The purpose of this document is to define the coding convention of the language in which we are developing our project. If we use standards in our product then it is easy for the other developers to understand and change out products.

Why coding convention is necessary?

● By using standard Java language conventions, you make your code easier to read for yourself and for
other programmers.
● More than 80% of lifetime cost of a system goes to its maintenance. By using standard coding
convention we can make this task easy for other programmers.
● Hardly any software is maintained by its original author for whole life so, it helps the
Incoming employees.
● It improves the readability of the software and It indicates that **less time** is spent to figure out what
the code does

Acknowledgments

This document reflects the Java language coding standards presented in the Java Language Specification.

2. Android app Files

There are two types of files in an android.

- Java files
- XML files

3. Java Coding Conventions

3.1 File Names

This section lists commonly used file suffixes and names.

3.2 File Suffixes

Java uses the following file suffixes:
File Type --------> Suffix
Java source --------> filename.java
Java byte code --------> filename. Class

3.3 Class Names

Class name should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread
etc.

3.4 File Organization

A file consists of sections that should be separated by blank lines and can give comment which defines
functionality of that section. Files longer than 2000 lines should be avoided.

3.5 Java Source Files

Each Java source file contains a single public class or interface. We can add private classes and interfaces
are associated with a public class in the same source file.

3.6 Beginning Comments

All source files should begin with a c-style comment that lists the programmers, the date, a
copyright notice, and also a brief description of the purpose of the program.

3.7 Package Statement

The first non-comment line of Java source files is a package statement.

3.8 Import Statement

After Package Statement, import statements can follow.

3.9 Class and Interface Declarations

After import statements we can declare class or interface.

3.10 Variable Declaration

First we declare public variables, then protected, and then private variables.

3.11 Constructor

After variable declaration we should make constructor of class.

3.12 Methods

After constructor we should give declaration of methods.

3.13 Indentation

Four spaces should be used as the unit of indentation. Tabs must be set exactly every 8 spaces.

3.14 Line Length

Avoid lines longer than 80 characters, since they're not handled well by many tools.

3.15 Wrapping Lines

When an expression will not fit on a single line, break it according to these general principles:
• Break after a comma.
• Break before an operator.
• Prefer higher-level breaks to lower-level breaks.
• Align the new line with the beginning of the expression at the same level on the previous line.

3.16 Comments

Java programs can have two kinds of comments: implementation comments and documentation
comments.

3.17 Implementation Comment

Programs can have four styles of implementation comments
- Block
- single-line
- Trailing
- end-of-line

3.18 Block Comments

Block comments are used to provide descriptions of files, methods, data structures and
algorithms. In Block comments we comment multiple lines. Block comments should be used at
the beginning of each file and before each method to describe the functionality of each method.
Block comment starts with /* and ends with */.

3.19 Single-Line Comments

Short comments can appear on a single line indented to the level of the code that follows. If a
comment can't be written in a single line, it should follow the block comment format. Single line
comment also starts with /* and ends with */ but it is written on single line. It is use to describe
the working of that line.

3.20 Trailing Comments

Very short comments can appear on the same line as the code they describe, but should be
shifted far enough to separate them from the statements. If more than one short comment

appears in code, they should all be indented to the same tab setting. Avoid too many trailing
comments. . Trailing comment also starts with /* and ends with */ but it is written on single line
after the code of that line. It is use to describe the working of that line.

3.21 ▯ End-Of-Line Comments

End of line comment can comment out a complete line or only a partial line. It is possible to give
multiple line comments but it is not recommended.

3.22 Declarations

One variable declaration per line is recommended because in this why we can easily comment about
that variable.

3.23 Initialization

Try to initialize local variables where they're declared but if value to be initialized depends on some kind
of computation then you can initialize it later.

3.24 Naming Conventions

Naming conventions make programs more understandable by making them easier to read.

3.25 ▯ Class name

Class name should start with uppercase letter.

3.26 ▯ Interface name

Interface name should start with uppercase letter.

3.27 ▯ Method name

Method name should start with lowercase letter.

3.28 Variable name

Variable name should start with lower case letter.

3.29 Package name

Package name should consist of lower case letters.

3.30 Constants name

Constant name should consist of uppercase letters.

4. Standard Java Annotations

Annotation @Override is used which actually tells that we are going to give the implementation of that method that is inherited from parent class.

5. Conditional Statements

In our project we use the **if, if-else if, if-else** statements

6. Loops

In our project loops are also used. Mostly we used **while** and **for** loops

While Loop:

```
while(c.moveToNext())   {
        buffer.append("uob: "+c.getString(0)+"\n");
    }
```

For Loop:

```
for(int i=0;i<checkedStd.size();i++) {
            mSelectedStd.add(mDatabase.getSelectedStudent(checkedStd.get(i)));
        }
```

7. Exceptions

- Exceptions are handled in the code.
- Specific exceptions are catch before the generic exceptions.

```
try{

}catch (Exception e){

}
```

8. Log Class

To print the errors messages or other information the methods of class Log have been used.

For example:

- Log.i(String tag, String msg);
- Log.d(String tag, String msg)

9. XML Styling Conventions

9.1 Layouts

In android in xml files there are layouts every activity has at least one layout. Mostly we have used the relative layout and constraint layout.

Here are the names of layouts:

- Linear layout
- Relative layout
- Constraint layout

9.2 Views and widgets:

Views and widgets are also included in the code:

- TextView
- ScrollView
- ImageView
- ImageButton
- ToggleButton
- Button
- CheckBox

9.3 ID

Each view or widget has some id .Which is used to use that view or widget.

9.4 Closing tags

Self-closing tags are used in the xml code
Example of self-closing tag:

```
<TextView

    android:layout_width="wrap_content"/>
```

< ----