Coding standards document

May 11, 2018

1 Style,Structure and layout

1.1 Variables, Methods and attributes:

• Should be short and descriptive.

• Should begin with lower-case letter.

• Upper-case letter should be used immediately after the first name description of the variable.

• Should use a single upper-case letter at the start of each new word within the name.

• Should not be a single character e.g. string x

1.2 Accessor and Mutators

• Should have names based on the attribute to which they provide access.

• Should begin with the prefix get or set followed by the name of the attribute beginning with a capital letter e.g. getItem

1.3 Final variables

• Should make use of final variables instead of magic numbers to avoid making it hard to identify which numbers should be changed if the use of the program needs to be altered.

• Should have all upper-case names to make them clearly identifiable.

1.4 Indentation and Layout

• Blocks that are nested more should be indented more.

• Lines should be kept to a sensible length to make the code easier to read and print.

• Single blank lines should be used to separate methods and to emphasise blocks of code.

1.5 Bracketing

• Brackets will be used to clearly show the blocks of code they encapsulate.

• Closing curly brackets should be placed on the line after the last line of the code they enclose, at the same level of indentation as the start of the header on which the block begins.

• Curly brackets should always be used for if-, while- and for-, even when not strictly needed such as when the body only contains a single statement.

1.6 Exceptions

• Exceptions should be used where necessary.

• Instead of throwing basic Exception classes, sub-classes should be made with more meaningful names.

• Appropriate catch statements should be put in place to allow the program to recover if need be.

1.7 Imports

• Should avoid using import *(importing every package), rather each class should be specifically imported as required.

• Any unused imports should be removed if they are no longer needed to make it clear which classes will be used.

2 Naming Conventions

This section will cover examples referring to some of the conventional rules mentioned above.

2.1 Classes

• Class names should begin with a capital letter and each new word within the name should begin with a capital letter e.g "ClassName".

2.2 Exception Classes

• Exception classes should follow the same rule as normal classes, but should end with the word Exception e.g "ClassNameException ".

2.3 Methods

• Methods should begin with a lower case letter and each new word within the name should begin with a capital letter e.g "methodName()".

2.4 Variables and Attributes

• Both variables and attributes should being with a lower case letter and each new word within the name should begin with a capital letter exactly the same as methods e.g "variableName".

2.5 Packages

• Package names should all be in lower case e.g "demo.package".

3 XML

3.1 Tags

• XML tags should be ordered as follows: "xmlns" first, then id,then `layout_width` and `layout_height` alphabetically.

• Add a space between the closing slash and the final attribute.

E.g. `android:textSize="10dp" />`

• Self closing tags should be used when an XML element doesn't have any contents, you should use self closing tags.

3.2 Layout XML ID Naming / Java Class Widget Declaration Variable

This section provides a guide on how to use elements and prefixes.

Element	**Prefix And Example**
TextView	tv e.g tvText
ImageView	iv e.g ivImage
Button	btn e.g btnSubmit
EditText	et e.g etWords

4 Documentation

4.1 Comments

• Use in-line commenting to help the next developer who might be editing your code.

• Inline comments should appear on the line above the code you are commenting.

• Comment XML View elements using `<!  -- Comment -->`.

• Comments should be added within the body of a method if they are used within that method.

4.2 Resource Files

• Resources file names should be written in `lowercase_underscore` e.g `ic_star.png`

4.3 Layout Files

• Layout files should match the name of the Android components that they are intended for but moving the top level component name to the beginning.

• Example, if you are creating a layout for the SignInActivity, the name of the layout file should be `activity_sign_in.xml`

4.4 Version control

• No commented out code must be committed unless you have a very good reason that is clearly described in a comment by the code you are committing.

4.5 Repository definitions

• All the pictures should be in a folder called Pictures.

• All the files should belong to a specific folder.

• Folder names should be descriptive enough to show what they contain.

5 Code quality

5.1 To ensure code quality:

• Android Studio code style will be used.

• Coding standard rules mentioned above will be used for resources naming conventions, formatting and also to ensure version control.

• Testing will be conducted using JUnit to run unit tests.

• Lint tool will be used to identify bugs.

• FindBugs tool will be used to spot issues related to performance and correctness of code.

6 Repository

6.1 Branching

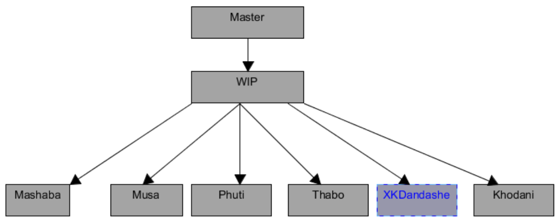

Figure 1: Branching

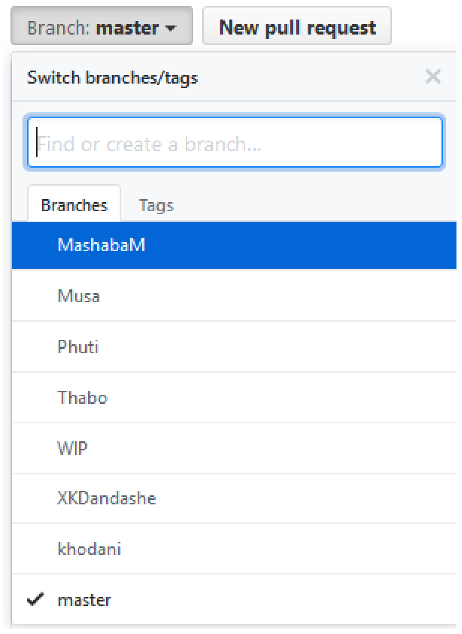

Figure 2: Git structure

6.2 Merging

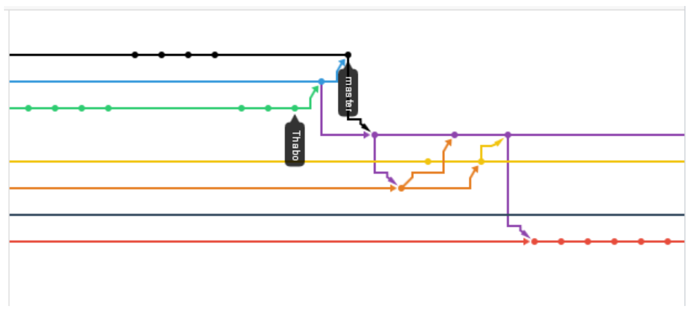

Figure 3: Merging