Coding Convention

Submitted to:

Dr. Adil Raja

Faculty of Engineering and Computer Science

Namal College, Mianwali.

Submitted by:

Group #03

M.Ikram Ullah Khan (UoB # 15026431)

Muhammad Usman (UoB # 15026436)

Naeem Ullah Khan

Shafi Ullah Khan

Abdul Kareem

Arshad Majeed

Students of Computer Science department Year-3

Namal College, Mianwali.

November 03, 2017.

Contents

Introduction 3
Android app Files 3
Java Coding Conventions 3
File Name 3
Package Import 4
Indentation 4
Comments 4
Line Length 5
Fields 5
Field Naming Convention 5
Methods 6
Standard Java Annotations 6
Conditional Statements 7
Loops 7
Exceptions 8
Log Class 8
Order of Class Members 8
XML Styling Conventions 9
Layouts 9
Views and widgets: 9
Closing tags 10
Dimension and Text Size 10
ID 10

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. This documents follows the standards that java and xml follows in android.

Android app Files

There are two types of files in an android.

- Java files
- XML files

Java files contains the main functionality of our app while in xml files the design and layout of app is maintained. First we talk about the java coding conventions and then xml.

Java Coding Conventions

File Name

In our android app project

- The class name is the same as the file name of java file
- The first letter of file is capital

For example:

File Name:

AddTeacher.java

Class Signature:

```
public class AddTeacher extends AppCompatActivity {
    // Class content
}
```

Package Import

There are two ways to import class from package.

Let we need a class **TextView** from package widget so we can write

I. import android.widget.*;
II. import android.widget.TextView;

We used the (II) way because it is more efficient way because in (I) way we import all the class from the package but in (II) way we import specific class.

Indentation

For the Indentation of code four (4) spaces or tabs are used and empty lines are also included just to increase the readability of code.

Comments

- Copyright statement at the top of every file then the package and import statements is written.
- Then the class is declared.
- Then in Javadoc comments the description about a class is written.
- Similarly the Javadoc comment is written for the every class and methods which describes about its functionality in a single line
- Single line comments "//" and multiple line comments are used to explain the methods, class and attributes and their role in the project.

Line Length

Mostly we also tried to keep the length of line short so that we don’t have to scroll horizontally the page except some important lines and URL.

Fields

Fields in the project are mostly defined either at the top of class or just above the methods that use them.

Field Naming Convention

Name of fields follow the naming rules given below and also have some meaningful name

- Fields that are not public and non-static start with **m**.
- Fields that are static start with **s**.
- Fields that are public static final are in capital letters with underscores.
- Other fields start with lower case letter and camel case convention is used
-

For example:

Button saveButton;

Toolbar mToolBar;

Methods

We follow the given bellow rules for methods to use them in project.

- Each method have some proper meaningful name
- Each method start with the lower case letter and then follow camel case convention.

Standard Java Annotations

Mostly in Android three type of annotations are used. In our project @Override is used which actually tells that we are going to give the implementation of that method that is inherited from parent class.

For Example:

```
protected void onCreate(Bundle savedInstanceState) {
        // some piece of code


        }
```

Conditional Statements

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

For example:

```
if(name.isEmpty() || email.isEmpty()){
            Toast.makeText(AddTeacher.this, "Enter Name And Email", Toast.LENGTH_SHORT).show();
          }else{
            insertData(name,email,degree);
          }
```

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)));
      }
```

Exceptions

- Exceptions are not ignored.
- Generic exceptions are not catch before the specific exceptions.

```
try{
        //some piece of code
    }catch (Exception e){
        //some piece of code
    }
```

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)

Order of Class Members

It is not strictly a rule but by mostly class members are in this order if they exist:

- Constants and fields
- Constructor
- Override methods
- Public methods
- Private methods
- Inner class

XML Styling Conventions

Layouts

In android in xml files there are layouts every activity has at least one layout. We could have layouts within the layouts according to our needs. Mostly we have used the relative layout and constraint layout.

Here are the names of layouts:

- Linear layout
- Relative layout
- Constraint layout

Views and widgets:

Views and widgets that we have used mostly are:

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

Closing tags

There are two ways for closing tags one is self-closing tag and in another way we us tag name and then use "/" in tag. We used self-closing tag.

Example of self-closing tag:

```
<TextView

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

Example of closing tag by using its name:

```
<TextView

    android:layout_width="wrap_content">

</TextView>
```

Dimension and Text Size

For dimensions it is used **dp** and for text size **sp**

ID

Each view or widget has some id, and id name is in lower case order.