Commenting Your Code:

The A290/A590 Course Standard

Contents

Introduction: Java versus XML

Part of learning to write program code involves learning how to comment the code effectively and meaningfully. This document is our attempt to provide you with guidelines not only for this class, but also for future projects.

The first question we must address when discussing code commenting is, *why comment?* Mainly, comments are added to increase human understanding of the program code. First and foremost, comments help you, the programmer, remember what you've done. This is especially important when debugging your code. Well-placed comments can be very helpful in determining whether the code actually does what you think it does. Comments that provide an “outline” of the algorithm can remind you what is happening at each stage of the program and also show us, the instructors, that you do know what's going on, even if it doesn't work correctly.

Comments will also be useful when others are trying to determine what your code does. Specifically, if you come to office hours with a question, comments can help us help you better. Also, if anyone (including you) sees your code at a much later date, the comments will provide insight into design decisions that you may not remember or that you may not be available to explain. Finally, if you find yourself as part of a programming team in your professional life, your comments will help your teammates understand your code if they find themselves having to maintain or update it. Similarly, their comments will aid you in similar tasks.

Okay, so we agree that commenting code is a Good and Useful Practice. Now, *why must we have a standard?* In terms of this course and beyond, a standard gives us a way to evaluate your comments as part of the grading for each assignment. Also, we know that standards are used extensively in industry, and we want you to gain experience working within prescribed parameters. In practice, you will likely be required to follow much more stringent coding standards than those given in this course.

In this document, each section will address both Java standards and XML standards.

BE SURE to read carefully as they can be very different.

File Headers

Java

The top of each file will contain a short header comment. This comment will give the following information:

- The name of the file
- A *brief* statement of the contents of the file.
- The author's name
- The date the file was created
- The name of the person to last modify the file
- The date the file was last modified
- The assignment to which it belongs
- The Project, Application, or other files to which it is related or to which it belongs

For example:

```
/*AndroidCallingMainActivity.java
 *
 *Contains the Java code for the
 *Primary Activity in this project
 *
 *Created by: Jeff Whitmer
 *Created on: 1/17/17
 *Last Modified by: Jeff Whitmer
 *Last Modified on: 1/23/17
 *Assignment/Project: A290 Android Development
 *Part of: Android Calling, refers to content_android_calling_main.xml layout file
 **/
```

Notice that the “comment” syntax for Java is an opening “/*” then a “*” beginning each line, and then a closing “**/”. Our SDK should actually do all this for you, if you properly start with “/*”.

As long as the prescribed information is present, you may choose the specific format for comment symbols, spacing, etc. However, we would encourage you to using something like the above to be sure you do not accidentally omit something.

XML

It is crucial to remember that the very beginning of your XML file defines how the entire file will be interpreted. Therefore, your file should always begin with something like this:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

    tools:context=".MainActivity" >
```

So where do we put our File Header? Fortunately, XML allows us the flexibility that we also have in XHTML, so we can put our heading comment NEAR the very top. DO NOT PUT IT FIRST. The “version=” line must be first. Once this is clear, the contents of the File Header is the same as above. Note the Header is at the end of the element.

```
<!-- content_android_calling_main.xml
Primary Layout file for this Project
Created by: Jeff Whitmer
Created on: 1/17/17
Last Modified by: Jeff Whitmer
Last Modified on: 1/24/17
Assignment/Project: A290 Android Development
Part of: Android Calling Project, refers to AndroidCallingMainActivity.java
-->

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
```

If you only have a single line of comments, you can write it as

```
<!-- This line has been modified by renaming the @string to hello_universe -->
```

But if you have more than one line, it is best to put the two tags for the comment element on their own lines.

Function Headers

Java

Each function declaration and definition will be preceded by a short comment stating the following:

- what the function does (not how it does it)
- the role of each input value
- any assumptions the function makes about the input values
- any guarantees the function makes about its output value

For example:

A function with no parameters:

```
/*  yesOrNo returns 1 if the answer given is yes,
*                  0 if the answer given is no,
*             and -1 if the answer is neither.
**/

int yesOrNo();
```

A function with parameters:

```
/*   processInput reads the command contained in cmdfile
*    and makes appropriate changes to the input data.
*    It returns 0 if no errors are encountered.  Otherwise,
*    1 is returned with an appropriate error message in err.
**/

int processInput(char *cmdfile, char *err);
```

XML

They may not be functions per se, but when you add a control or other, similar element, you need to comment it as well.

For example:

```
<!—
Drop Down List that allow choice of a Programming Language
There are three options: C#, CSS, and Java
Available options are preset and not interactive
Choice sets that items valus as “Selected”
-->

[XML markup unary or binary element(s) would be here]
```

Remember, the goal is information more than explanation. But it needs to be thorough enough that someone who has never worked with your mark-up or code will be able to quickly understand what you did and how to modify or maintain it.

In-line Comments

Java

Comments that appear interleaved with the code should adhere to the following guidelines:

- They should provide an outline of the function.
- They should explain what the code is doing without restating the code itself.
- They are necessary only when it is not obvious what the code is doing. To determine “obviousness”, ask yourself the question “If I had never seen this code before, or even hadn't seen it for a week, how long would it take me to determine what this section does?” If the answer is anything longer than 1 minute, include a comment.
- They should appear right at the beginning after the opening brace or at the closing brace of long blocks to help the reader determine which opening brace it is paired up with.

XML

Comments that appear interleaved with the mark-up should adhere to the following guidelines:

- They should provide an outline of the element or feature.
- They should explain what the mark-up or element is doing without restating the mark-up or element itself.

- They are necessary only when it is not obvious what the mark-up or element is doing. To determine “obviousness”, ask yourself the question “If I had never seen this mark-up or element before, or even hadn't seen it for a week, how long would it take me to determine what this section does?” If the answer is anything longer than 1 minute, include a comment.
- They should appear immediately before the feature or element to help the reader determine which feature or element it is paired up with.

Revising Given Code

If we provide you with code that must be modified, comments relating to the modified code should be updated to reflect the changes. The updates will involve adding the following information to the appropriate headers, above and beyond the basics you need to include in the File Header:

- the date of modification
- the name of the person who performed the modifications
- a brief statement of the revisions

The role and importance of comments in your A290/A590 assignment and Project scores

The comment portion of each assignment grade will be determined by the presence of your comments and your adherence to the standard. Although the number of points ascribed to “comments” may not seem that large, your ability to comment effectively and consistently is a skill you will need for future projects. This is especially true if you will be involved with any large software development projects. Comments will not only allow you to understand your own code better, they will allow those who follow you to understand it as well. Thus, it is in your best interest to include comments according to the standard. Just to be clear, points related to comments are significant enough (up to 20% of an assignment score) that if you fail to follow these guidelines, it could impact you by as much as a full letter grade in the course.

Acknowledgements:
Revised from CS 225, University of Illinois, Urbana-Champaign, W. Terry, 1997.
Revised from Electrical Engineering website, Iowa State University, 2008
Last update: 1/22/2017