475 Software Engineering for Industry : Topic 1 : Legacy Code

Assessed - work in a group of 3.

In this exercise we will look at the techniques for working with legacy code, and making changes to it in a rigorous and reliable manner. We will do this through a combination of reading, practical work, writing and discussion.

Reading

Start by having a look at the ideas of Technical Debt and the Design Stamina Hypothesis, both usefully written up by Martin Fowler on his website.

The following articles by Michael Feathers give some more detail and background on some of the techniques presented in class for working with code where little effort has been spent on testability.
http://www.netobjectives.com/system/files/WorkingEffectivelyWithLegacyCode.pdf
http://www.informit.com/articles/article.aspx?p=359417&seqNum=3

You might also look at this famous paper *Big Ball of Mud* by Foote and Yoder. It is quite long so we suggest looking through the first 13 pages. http://joeyoder.com/PDFs/mud.pdf

Have a search on the internet for more people talking about working with legacy code, in academic articles, technical blogs, and perhaps videos of conference talks.

Coding

For the practical part of the exercise we'll work with an open source codebase that we found on the internet. It is an application to arrange photos into albums and slideshows. It is written in Java. It has no tests. We want to introduce some tests around the code to help us understand it, and give us confidence that we won't break it if we make changes. We'll probably need to refactor the code to allow us to test it. **Look at page 2 of this spec for details of how to get the code etc.**

You should find a class called `JPhotoFrame`. If you run the `main()` method in this class you should be able to launch the application to see what it looks like.

For this exercise, we want to write a test that demonstrates what the application does when you click on *Start Slideshow* in the View menu.

As a starting point, find the string *Start Slideshow* in the code. Use the IDE's search function you should find the string in the `JPhotoMenu` class. Line 57 defines a constant for the string "Start Slideshow". If we trace uses of this constant*, we find one in the `JPhotoFrame` class on line 582. This is in the middle of the (massive) `actionPerformed()` method.

* Alt-F7 in IntelliJ, or Alt-Shift-G in Eclipse (key combinations may vary on Linux/Windows/Mac).

Add some Tests

Create a JUnit test for `JPhotoFrame` (typically we name this either `JPhotoFrameTest` or `TestJPhotoFrame`). Write some tests to demonstrate what happens when you click View Slideshow. You probably want to refactor a bit. It will probably be easier to test if we extract the behaviour we are interested in out of the `actionPerformed()` method. You will probably need to refactor further to break dependencies to help with sensing.

Add some Behaviour

Once you are happy that you have enough tests in place that you understand and can verify the behaviour of the existing code, try adding a feature (and a test for it, of course). Can you add another menu option that allows a preview slideshow that cycles through the photos more quickly, with a shorter interval between them?

Questions to Consider

What makes this code hard to test? (or is it easy?)
What do you want to test and how can you test it?
What are the tradeoffs you need to make?
Do you think the design of this code might be different if it had been developed test-first?

Getting The Code

You can find the skeleton code on GitLab. Go to https://gitlab.doc.ic.ac.uk/475/jphotoalbum and click ***Fork Repository*** to create your own copy of the repository under your own GitLab account (only one person from your group needs to do this, then they can add the other group members as developers on the project). Clone your repo locally to start work.

Importing the code into an IDE…

When you clone the code it should already have project files included for Eclipse and IntelliJ IDEA. In Eclipse you should be able to *Import… => Existing Projects into Workspace*. In IntelliJ you should be able to *Open…* the project. But you may want to re-import the project, or set it up in a different way. Feel free to do that if it is easier. We would not suggest trying to use the provided build.xml file with Ant.

Because of its legacy dependencies, this codebase may well not build with OpenJDK - but it should build using an Oracle JDK.

Writing

Once you have made some progress with the coding part, write up a summary of your thoughts on working with legacy code. The submission should be 2 pages (no covers or contents pages please).

One page one, address one of the following questions:

- If no-one likes working with legacy code, why do so many companies have so much of it?
- To what degree do different types of tests help us to work with legacy code?
- If a company has a lot of legacy code and systems, should they update and replace them?

This is not a long essay, just a short statement of your thoughts, with supporting evidence and references. Try to focus on the key points and be interesting!

The ideal length is around **300 words**. References can be extra, but everything must fit on one page. Do not use a tiny font and tiny margins, it is better to write less and trim out anything unnecessary.

On page two, give an overview of what you did in the photo album exercise, what decisions you made and why. There's no need to write in detail, but maybe a flow chart, some box-and-line diagrams, or some code snippets might be useful. Include the URL of your GitLab repository for the exercise code.

Submission As a group, submit a pdf (`topic1.pdf`) of your 2-page write-up via CATE.

Deadline Monday 21st Jan, 9am.

Note that the deadline is Monday at 9am - i.e. in the **morning**. We suggest you think of it as midnight Sunday.

Discussion

During the class on Tuesday 22nd Jan, we will discuss your thoughts and experiences. We will ask some groups to briefly present the code that they wrote for the exercise, and others to describe their thoughts. We hope for a good discussion amongst the class.

Schedule

Tuesday 15th Jan exercise released
Friday 18th Jan (9-11am) - lab session (lab 219)
Monday 21st Jan (9am) - deadline for submission to CATE.
Tuesday 22nd Jan (11am-1pm) - discussion class (lecture theatre 340).

Assessment

The code you write and page 2 of the submission are not graded, but you must demonstrate that you have done something reasonable for that part in order to have page 1 graded.

On page 1 we are looking for you to express your thoughts and ideas based on your reading, experience and discussions, backed up by evidence. The grading scheme is as follows:

F - E
Little or no understanding of the given topic demonstrated.

D
Shows an incorrect or flawed understanding of how or why to apply the given tools/techniques.

C
Shows a reasonable, but limited, understanding of the application of ideas and techniques covered, and the context in which they apply.

B
Shows a good understanding of how to apply these techniques and the problems that they solve. Arguments are well presented and backed up by references.

A
Displays a broad understanding of the use of these techniques, comparing different approaches and the forces that might make them suitable for different situations, displaying evidence of further independent reading and thought, beyond what was suggested and covered in the class.

A*
Gives an excellent and insightful commentary, comparing different views, approaches and/or tools, and displaying evidence of further independent reading and thought. Demonstrates critical thinking and considered opinion, but backed up by references and practical experience.