COMP-533 Model-Driven Software Development

Assignment 1

Logging Implementation

(15% of final grade)

September 24, 2018

Introduction

Logging is probably one of the most common recurring development concerns. During development, logging the execution of important operations, or the sending of remote messages, or the modification of state, or the handling of exceptions, can significantly streamline understanding, testing and debugging. Nowadays, many deployed systems also make heavy use of logging for auditing purpose.

The goal of assignment 1 is to get a feeling of the connection between low-level design models and code (task 1), and then to gain experience in using aspect-oriented technology at the programming level to modularize the logging functionality in a reusable way.

Task 1: Installing TouchCORE and Java Code Generation

Download *TouchCORE*, the concern-oriented modelling tool used for this assignment from *myCourses* (there is a TouchCORE version that runs on any platform that supports Java 7, and one specifically bundled for Macs). Download the *BankWithLogging* concern from *myCourses*.

Open TouchCORE. You will see a circular menu in the top right corner of the screen. Click on the "Open Concern" button (button with a folder icon)[1]. Then navigate to the *BankWithLogging* concern folder and select the `BankWithLogging.core` file. You will now see the main concern edit screen, with one *BankWithLogging* feature (a white rectangle) in the centre. Click-and-hold on the feature, then select "Open Realizing Design Model" from the popup menu that appears. This opens the design class diagram of the *BankWithLogging* application.

Optional: You can explore the class diagram (panning is done by right-clicking and dragging on the background). You can look at the behaviour of the public operations of the design classes by click-and-holding on a public operation and then choosing the "Go to Message View" button (icon that looks like a sequence diagram without "*"). Hit the "Back" button (icon that looks like a left arrow) at the top left of the screen to navigate from the sequence diagram back to the class diagram.

To generate the Java code of the BankWithLogging application, click on the "Weave All" button (icon that looks like a net). This will generate a new class diagram called "woven_BankWithLogging". Now click on the "Generate" button (icon that looks like a gear). Select "Java" from the popup menu. Then choose a folder into which you would like to generate the application code.

Compile the code, then run it. You should see a simple GUI that allows you to create customers, create accounts for the customers, and deposit and withdraw money from the accounts. Whenever a customer or account object is instantiated, or deposit or withdraw is executed, a log message is displayed in the console.

[1] Clicking-and-holding on the center of the menu will display explaining text for each button.

Task 2: Install AspectJ and Separating Logging

For task 2 and 3 you are required to use *AspectJ*, an aspect-oriented extension of Java. *AspectJ* is an Eclipse project, but you don't need to use Eclipse to run *AspectJ*. Nevertheless, the *AspectJ* integration with Eclipse is very well done, so IMHO the easiest way is to get *AspectJ* running is to download the latest Eclipse for Java Developers (Eclipse 4.9 Photon), and then use the Help->Install New Software... menu and the update site http://download.eclipse.org/tools/ajdt/48/dev/update to install the *AspectJ* compiler and all the related Eclipse IDE plugins.

For task 2 you are asked to refactor the *BankWithLogging* Java code. You are to use *AspectJ* to modularize the logging functionality in such a way that the *Account* and *Customer* classes in the end only contain business logic, and do no longer make explicit logging-related calls. The *Logging* module does not have to be reusable, i.e., the *Logging* code is allowed to directly refer to the *Account* and *Customer* classes.

Make sure that the logging output printed to the console looks exactly like before. When done, keep a copy of your code (for submission on *myCourses*).

Task 3: Reusable Logging

For task 3 you are asked to elaborate a *reusable Logging* module. It should provide *generic logging functionality* for recording constructor executions and operation executions. The *Logging* module is not allowed to have any dependencies on the *Account* and *Customer* classes.

Then refactor the *BankWithLogging* code to use the generic logging module. For this, you need to write a *bank-specific composition specification* that tells the system that you want to log the creation of customers and accounts, as well as the withdraw and deposit operation executions. You can declare additional aspects to do this, or you can add annotations to the business classes, if necessary. You are not allowed, though, to make explicit calls from business classes to the generic logging module.

Again, make sure that when you run your code the logging output printed to the console looks exactly like before. When done, keep a copy of your code (for submission on *myCourses*).

Task 4: Demonstrating Reuse

Write some other Java code (or take some existing Java code you have previously worked on), and configure it to reuse the *Logging* module you developed in task 3. The application can be super simple. The aim of this task is simply to demonstrate that your *Logging* concern is indeed reusable.

Hand-In

The assignment is due Wednesday October 10th. Late hand-ins are not accepted.

Remember that you are supposed to work in groups of 3 students on this assignment, but that you are not allowed to work with the same students for assignment 2 or for the project. Please sign up to one of the groups named "Assignment 1 Group x", and then submit your hand-in through *myCourses*.

- Please create a folder for task 2 containing the complete code for task 2 (aspect(s) and Java code).
- Create a folder for task 3 containing the complete code for task 3 (aspect(s) and Java code), and a "readme" file explaining how a user of your *Logging* module has to customize (i.e., specify the application-specific composition specification) your reusable module to his needs.
- Create a folder for task 4 containing the complete code for task 4 (aspect(s) and Java code), together with a "readme" file explaining how to compile and run your code.

Compress the 3 folders into one zip file and submit it on myCourses.