Radboud University

Dr. Daniel Strüber
Dr. Carlos Diego Damasceno

Radboud University
Institute for Computing
and Information Sciences

Autumn term 2022/23
Published: 10.10.2022
To be submitted:
17.10.2022 10:30

Software Product Lines

Assignment 6

Tasks 1-3 are designed as live exercises for the tutorial session – for this, we will split up in breakup groups. It's not required to submit anything for these tasks.

Task 4 is to be completed offline and to be submitted until the deadline.

Task 1: Pointcuts

Define pointcuts for capturing each of the following (sets of) join points:

(a) Executions of the method **send** in the class **Client** with the parameter **text**
(b) Calls of a method with the name **send**
(c) Calls of a **public** method
(d) Calls of method **send** in the class **Client** that do not come from the Client itself
(e) Each read access to the **Socket** in the **Server**
(f) Each creation of a **Connection** object
(g) Each method call from within the method **handleIncomingMessages** of the class **Server**, except for calls to objects of the class **Encrypter.**

Task 2: Obliviousness

Does the obliviousness principle also apply to collaborations? What are the benefits and drawbacks of this principle? How could one avoid the fragile pointcut problem?

Task 3: Refactoring towards aspects

Implement lines 3, 4, 7, 9, 10, 11 of the following programs as aspects. At the end, there should be a Java class that consist of the base code, and aspects that restore the behavior of the original code.

```
1  public class Foo {
2      public static void main (String parameter, Locker locker) {
3          int lockId = locker.lock();
4          try {
5              int  i  = parameter.length() ;
6              firstOperation(i);
7              log("first Operation executed with parameter " + i);
8              secondOperation(i);
9          } finally {
10             locker.unlock(lockId);
11         }
12     }
13     //..
14
```

Task 4: Aspect-oriented implementation

Evolve your chat product line towards aspect-oriented programming. Your solution should have at least a base program and one feature (of your choice) implemented as an aspect. The base program may include feature-specific code of the other features.

Hints:

- This week, we need to face the limitations of the used tooling: According to the FeatureIDE website, the AspectJ integration of the most recent version is buggy.
  We were able to complete the steps from the tutorial* without errors and, therefore, expect that you can get a "minimal" solution to work that supports one (possibly simple) feature. But there is a possibility that this might not work out.
- If you cannot get a working solution after reasonable effort, write up a brief report of the problems you encountered and what you tried to address the problems. Your submission can then be counted as successfully completed.

Submit a zip archive with:

- Your chat product line implementation from task 4 *or* a report of the problems you've encountered and steps taken to address them.

* https://github.com/FeatureIDE/FeatureIDE/wiki/Tutorial#AspectJ