Observers & adapters in action - Implementing a logging library
Announcements - A4

● Due **Saturday, March 4, at 4 pm**
  ○ Submit your work early, and make sure your code compiles
  ○ Do NOT change the testing code (i.e. anything under `src/test`) or interfaces
  ○ Do NOT have dependencies on the testing code (i.e. anything under `src/test`)
  ○ If Eclipse/IntelliJ indicate that there are compile warnings, fix them!
  ○ The deadline is strict (as the auto-marker rolls back commits that were made after the deadline)

● There will be no exceptions for A4

● You are expected to work on this assignment gradually (and commit your work as you go along).

● **If your first commit is in the last 24 hours before the deadline, I will not look at your code, no matter what the issue is.**
Case Study

● The goals of this case study are to:
  ○ Learn about common software engineering challenges.
  ○ Apply some of the techniques we learned to help us with these challenges.
  ○ Realize that professional libraries are not as simple as they seem
    ■ Requirements evolve as developers push tools to their limits.
    ■ Extensible and flexible design is extremely important
Case Study

- You’ve probably done some logging in your life
  - Ex: Print statement to verify that your code is behaving as expected
- `System.out.println` only gets you so far...
  - Need to evolve our requirements
  - Eventually, requirements become demanding enough to justify creating a library.
OK, let’s go through the example ...

Note: We will only go through steps 1 through 9
Case Study

● Are we done?
  ○ You are never done working on a library,
  ○ But you have to stop at some point

● What are we missing?

● What can we improve?
Further improvements?

● User story:
  ○ As a programmer, I would like to define a custom logging level, PRODUCTION-DEBUG, that is between DEBUG and INFO

● Can we make our library more flexible?

● Any ideas?
  ○ Maybe StackOverflow has the answer
Further improvements?

● **User story:**
  ○ As a programmer, I would like to use an appender that writes messages over the network, without having it slowing down my application

● **Solution:** Do all logging in a [separate thread](#)
  ○ Possibly using a thread-pool
Further improvements?

- Problem: **flushing the buffer** on each log message is inefficient.
  - Flush the buffer = Write to disk
  - Slower than memory IO

- Solution: Flush buffer in a **shutdown hook**

- Problem: Log file is not updated in real time. Difficult to use it for debugging.

What to do? Decide on a trade-off
How Much Should We Improve?

It’s easy to think of more use cases and features, but ...

- We can’t keep improving our software forever, we need to release!
  - Get users
  - Collect feedback
  - Figure out which improvements are important to your users (and which aren’t)