17-708: Software Product Lines

Assignment 5: Preprocessor Implementation

Task 1

Reimplement the Chat system from Assignment 3 (not Assignment 4) as a product line, using compile-time conditional compilation with a preprocessor. Aim at removing all code of deselected features at compile-time.

When using C/C++/C# code, we recommend using the standard preprocessor cpp. For Java code, we recommend using Munge or Antenna. Among the two, Antenna has the better integration in FeatureIDE. FeatureIDE comes with examples on how to use preprocessors in Java code.

Task 2

Add a new feature *Text Interface* that provides an alternative to the GUI of the original implementation of the chat client. That is, a user should be able to participate in a chat from the command line.

Task 3

Reflect on the implementation, especially the following two aspects:

1. Effort: Was the creation of the product line a difficult task? What were the challenges? Was adding a new feature challenging? How much planning was required? How does it relate to your experience with frameworks (HW4)?

2. Code quality: How would you rate the understandability and maintainability of your implementation? Have you taken any steps toward improving code quality or would there be reasonable steps that you could have taken?

A good reflection document does not only present facts but also interprets them in depth and adds additional value by adding judgments from experience or opinions. Do not merely recite textbook opinions but discuss whether your experience aligns with them. A good reflection document will include concrete statements about lessons learned, with clear supporting evidence, such as examples, to support the claims. For example, "It is easy to understand in isolation." is weakly supported. One could strengthen it with examples from the development experience as follows: "One source of difficulty was the integration of components A and B, because the API for A was similar but slightly different from that of B… In the a larger system, this could have the following consequences.. Once could try to use … instead."

Deadlines, Technicalities, and Hints

Finish the implementation and reflection document by Oct 26, 11:59pm.

Create a branch “preprocessor” in the provided GitHub repository. Provide brief mapping from features to names (e.g. macro names, flags) used in the implementation in a README file in the root of the repository. In the same file, describe which preprocessor you have used.

Add the reflection document to the root of the directory. Both .pdf files and pure text files (e.g. Markdown) are acceptable. Create explicit subsections in that document for the two aspects. The entire reflection document should not exceed 1000 words (soft limit).

At the deadline, we will take a snapshot of your GitHub repository.

As usual, follow reasonable development practices, e.g. using Java naming conventions, making incremental cohesive commits with meaningful commit messages, including documentation or tests where appropriate.

Grading

We expect

- A preprocessor-based product-line implementation of the chat (both client and server if needed), including the new feature
- A README file containing instructions on the preprocessor used and a mapping of features to preprocessor macros.
- A reflection document with two subsections discussing in-depth the stated questions.