ASM Linter - Software and Architecture Design Specification

Design Purpose

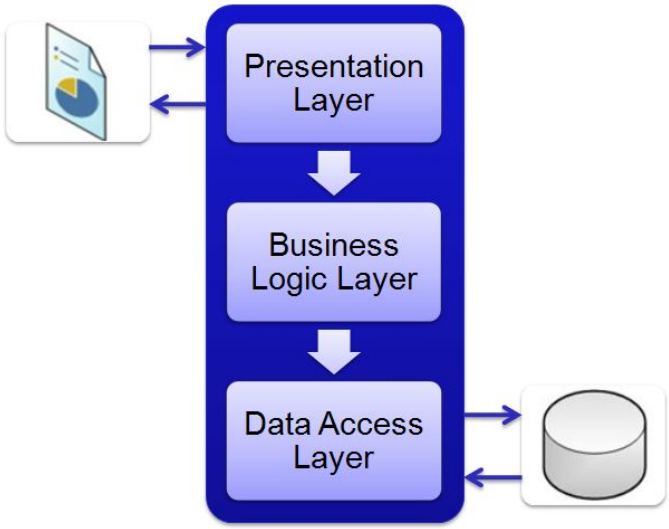
The linter was originally designed for the use of our client Dr. Yiji who is a professor of CSSE374 at Rose-Hulman Institute of Technology. In CSSE375, the goal is to refactor the codebase for readability and the work of future features easier. Our main goals were to:

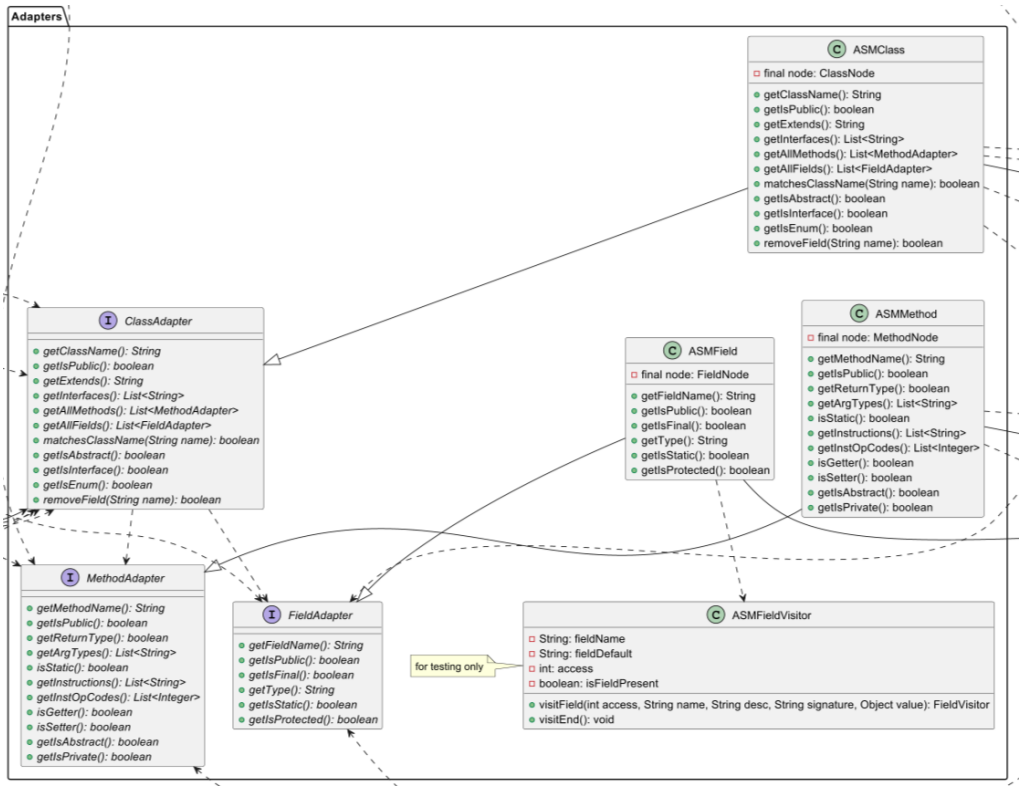
- Refactor Bad Code Smells
- Maintain a functional product with flexible design for future iterations
- Quickly create new features to gain feedback

Though linters are very common and necessary for almost all text editor products, a linter which verifies/checks design patterns is not seen on the market, so we are working in the novel domain. In doing so, we create a mainly greenfield system.

Design

The design of this software follows the separation of concerns principle, splitting the design into 3 layers: data, domain, and presentation. The data layer handles system files and external libraries, the domain layer handles all core functionality of the software, including checks and the linter object itself, and the presentation layer contains anything involved in UI or displaying things to the user.





The software works by loading Java class files into ASM objects which are stored into adapters. These adapters are wrappers of the ASM library so that the dependence on the library is not too strong and can be swapped out later if needed. These adapters are then sent through Check objects that the user chooses to run. Each check object has an independent attribute they check for in the provided classes, such as badly implemented design patterns or incorrect naming conventions. Each Check returns an object containing the faults it found in the code, and they are then compiled and passed to the presentation layer where they are displayed to the user.

Quality Attributes

For this iteration, the quality attributes are categorized as nonfunctional requirements below:
Category: Performance, Usability

- Learnability: New users will be able to configure and use the software within 30 minutes
- Latency: System will be designed to evaluate one design at a time. The linter will return an output in less than 15 seconds.

Primary Functionality

As a design linter, the functional goal of our program is to flag bad code smells, stylistic design errors, suspicious constructs when given a Java Project. The main goal of this program is to help software engineers create good Java design by flagging suspicious areas for reconsideration. Additionally the program could be used as a supplementary tool for Software Design graders - primarily to direct a graders attention to areas of suspicion.

Additional Features added:

- Additional checks
- Interactive User Interface
- Compile a list of design patterns identified
- External configuration file
- Logging feature for saving linter output

Architectural Concerns

- Breaking up the features equally amongst our team
- Displaying and saving the issues our linter detects
- Designing the User Interface

Constraints

Technical:

- We are required to use Java
- We are required to use GitHub
- We are required to keep documentation

Non-Technical:

- Team members must contribute equally
- Document all features/sources/refactorings
- Flexible to using another library in place of ASM