Finding and documenting design patterns

Software Design and Modelling, Assignment 1

Luca Di Bello

October 11, 2024

Contents

1 Introduction

In this assingment we are going to use the pattern detection tool pattern4j to automatically detect the usage of design patterns in a chosen open-source Java project and document the result in a report. The tool is able to scan Java bytecode and detect the usage of 13 design patterns.

1.1 Project requirements

The assignment requires to choose a Java open-source project available on GitHub, which satisfies the following requirements:

- At least 100 starts
- At least 100 forks
- At least 10 open issues
- At least 50'000 lines of Java code (comments included)

In order to find a valid project, the GitHub search feature allows to filter repositories based on different criteria. For example, to find a Java project that satisfies the requirements listed above, the following query can be used:

```
stars:>100 forks:>100 language:java
```

On the other hand, in order to count the lines of code of a project without having to clone it locally, it is possible to use the following web application: codetabs.com/count-loc.

1.2 Project selection

In order to learn more about the design patterns used in famous Java projects possibly used in the industry, I looked for active projects, with a large community and a good number of stars and forks. In order to find projects

In order to learn more about design patterns, I decided to look for a project that is activelly used in the industry, with a large community and a good number of stars and forks. After some research I first selected a small set of projects that satisfied the requirements:

- projectlombok/lombok: Library containing a set of useful Java annotations to reduce boilerplate code in Java applications. It has 30.7k stars, 3.7k forks, and used actively in many famous projects. It has around 98k lines of Java code (comments excluded).

  This project unfortunately was discarded as it was using the `Ant` build system, which I am not familiar with. To avoid potential issues and to be able to focus on the main task of the assignment, I decided to look for other projects using `Maven` or `Gradle`.

- ReactiveX/RxJava: Library for composing asynchronous and event-based programs using observable sequences. It provides many bindings for different languages and platforms. It has 19.9k stars, 2.9k forks, and has about 395k lines of Java code (comments excluded).

  Initially, I started this assignment with this project in mind but, unfortunately, I encountered major problems during the compilation of the project which made me discard it. The project was using the `Gradle` build system but the build process terminated with an unexpected error that I was not able to debug:

```
FAILURE: Build failed with an exception.

* What went wrong:
Could not open settings generic class cache for settings file '/Users
    /lucadibello/Developer/RxJava/settings.gradle' (/Users/lucadibello
    /.gradle/caches/7.6.4/scripts/7qrwiegia3vs821g20py0pu68).
> BUG! exception in phase 'semantic-analysis' in source unit '\
    _BuildScript\_' Unsupported class file major version 67
```

  Listing 1: Error during compilation of RxJava project

- benmanes/caffeine: A high-performance caching library for Java used in many famous projects such as *Apache Kafka*, *Cassandra*, *Neo4J* and many others. It has 15.8k starts, 1.6k forks and has about 100k lines of Java code.

  This project completely aligns with the project requirements and my personal objectives for this assignment. For these reasons, I decided to use this project for the assignment. Additionally, the project uses the `Gradle` build system, which I am more familiar with.

1.3 High-level overview of the project structure

TODO: Provide a high-level overview of the project structure, what are the main packages and explain that many lines of code are related to the tests and for this reason I will also consider the test code in the design pattern analysis.

This project, apart from offering an high-performance caching library, offers also a set of utilities and tools that can be used in different contexts. The main packages of the project are:

- *com.github.benmanes.caffeine.cache*: Contains the main classes of the caching library.

- *com.github.benmanes.caffeine.cache.guava*: Contains classes that provide compatibility with the Google Guava caching library.
- *com.github.benmanes.caffeine.cache.jcache*: Contains classes that provide statistics about the cache.

2 Project

3 System Design

4 Evaluation

5 Conclusions and Future Work

References