Assginment 1 – Software Design and Modelling

Volodymyr Karpenko Claudio Maggioni

October 17, 2022

1 Project selection process

We need to find a project that is a single unit in terms of compilation modules[1] self contained and with as little external dependencies as possible to ease the analysis project. Additionally, it would be nice if we choose a project that we already know as library clients.

1.1 Projects Considered

We considered the following GitHub repositories:

vavr-io/vavr a Java library for functional programming, discarded as the project is less than 20K LOC and doesn't meet the selection criteria;

bitcoin4j/bitcoin4j a Java implementation of the bitcoin protocol, discarded as the project is distributed in several subprojects;

FasterXML/jackson-core a Java JSON serialization and deserialization library. We chose this library because it meets the selection criteria, it doesn't rely on external components for its execution, and its project structure uses a single Maven module for its sources and thus easy to analyze.

1.2 The Jackson Core Library

As already mentioned, `Jackson` is a library that offers serialization and deseralization capabilities in JSON format. The library is highly extensible and customizable through a robust but flexible API and module suite that allows to change the serialization and deserialization rules, or in the case of the `jackson-dataformat-xml` module, to allow to target XML instead of JSON.

The chosen repository contains only the *core* module of Jackson. The *core* module implements the necessary library abstractions and interfaces to allow other modules to be plugged-in. Additionally, the *core* module implements the tokenizer and low-level abstractions to work with the JSON format.

We chose to analyze version 2.13.4 of the module (corresponding to the code under the git tag `jackson-core-2.13.4`) because it is the latest stable version available at the time of writing.

[1] A problem for Pattern4J as compiled `.class` files are distributed across several directories and would have to be merged manually for analyzing them

2 Analysis

We use *Pattern4* as a pattern detection tool. This tool needs compiled `.class` files in order to perform analysis. Therefore, as `jackson-core` is a standard Maven project, we compile the sources using the command `mvn clean compile`. The `pom.xml` of the library specifies Java 1.6 as a compilation target, which is not supported by JDK 17 or above. We used JDK 11 instead, as it is the previous LTS version.

An XML dump of the *Pattern4j* analysis results are included in the submission as the file `analysis.xml`.

2.1 Comments

- Lots of false positives for the Singleton pattern. Example, com.fasterxml.jackson.core.sym.Name1 has a package private constructor and a public static final instance of it, but reading the documentation the class represents (short) JSON string literals and therefore is clearly initialized by client code.

  sym.Name1, JsonLocation, DefaultIndenter, util.DefaultPrettyPrinter$FixedSpaceInde
  not a singleton (detected cause of "convenient" default instance given as static final field), the constructor is not used but the class is extensible

  JsonPointer, filter.TokenFilter like above, but constructors are protected

  JsonpCharacterEscapes, util.DefaultPrettyPrinter$NopIndenter, Version a singleton but with a public constructor that is never called in the module code, may be called in tests

  io.JsonStringEncoder like above, but the class is final

  util.InternCache, io.CharTypes$AltEscapes actual singleton, thread-unsafe initialization

  io.ContentReference like above, but constructor is protected

- TBD