Software engineering 2,
assignment 3:

CODE INSPECTION

Julián David Gallego García

Politecnico di Milano. January 5, 2016


POLITECNICO
MILANO

CONTENTS

1. Introduction

1.1. Purpose

This document has the purpose to present the code inspection of some methods from a specific class of a version of glassfish. The main objective of the inspection is to improve the coding skills of the developers and to improve the quality of this initial development phase code. For this task to each group was assigned some code to analyze applying code inspection techniques, to help the inspection was given a checklist to analyze the code, from there each group has present a report of the problems of the code.

1.2. Scope

This document has the goal to present the inspection of a given code, to improve coding skills and to improve the quality of the inspected code, applying the inspections techniques and checklist given to do it.

1.3. Reference documents

- Assignment document: Code inspection.pdf
- Glashfish javadoc of this version: http://glassfish.pompel.me/
- Methods assigned to each group: http://assignment.pompel.me/

1.4. Overview

The document is organized in:

- Section 1: brief description of the idea, goals and objective of the document.
- Section 2: Describes the code that is going to be inspected, saying the methods and the class in which belong to. Using the checklist given for the examination, it is reported all the problems found in the code.
- Section 3: Describes the problems that are out of the checklist for the code inspection.

2. Code inspection

2.1. Classes and Methods

All methods assigned belong to the same class MappingReferenceKeyElementImpl.

2.1.1. Methods

- **Name:** removeColumnPairs(ArrayList pairNames)
  Start Line:347 from MappingReferenceKeyElementImpl.java

- **Name:** addColumnPairs(ColumnPairElement [ ] pairs)
  Start Line:408 from MappingReferenceKeyElementImpl.java

- **Name:** getColumnPair(DBIdentifier name)
  Start Line:538 from MappingReferenceKeyElementImpl.java

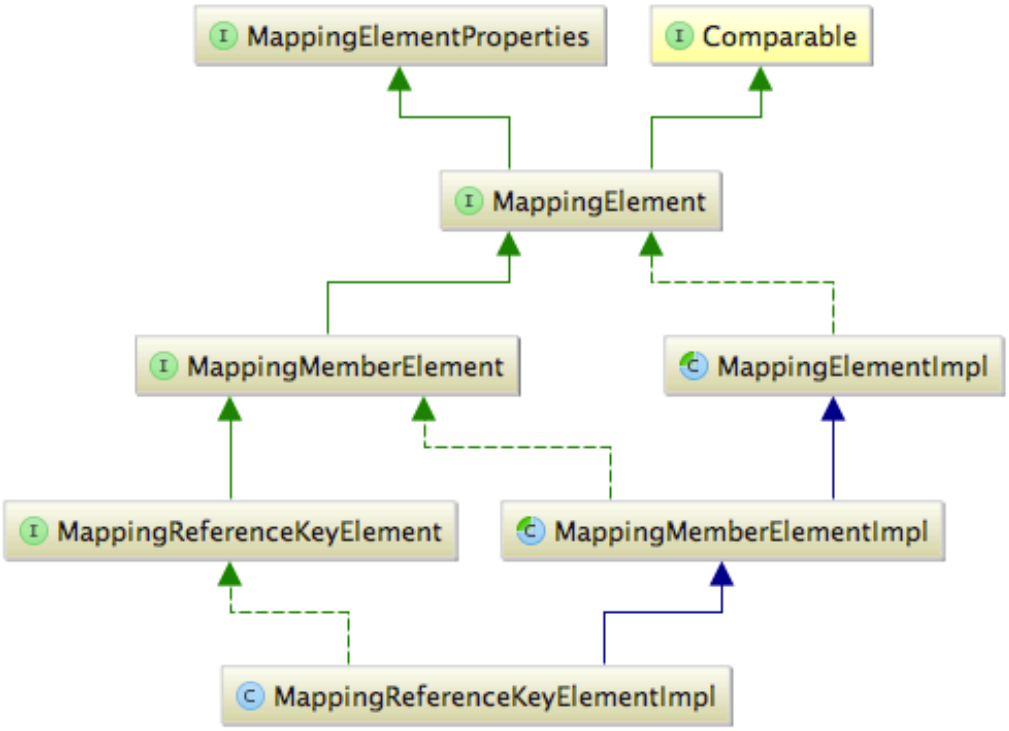

Figure 1. Inheritance diagram.

2.2. Checklist

2.2.1. Naming conventions

Everything is fine.

2.2.2. Indention

All the methods were written using tabs.

2.2.3. Braces

Consistent Allman style.
The single if statement is not surrounded by braces at line 373.
The single if statement is not surrounded by braces at line 375.
The single if statement is not surrounded by braces at line 555.

2.2.4. File Organization

Everything is fine.

2.2.5. Wrapping lines

Everything is fine.

2.2.6. Comments

Everything is fine.

2.2.7. Java source files

Everything is fine.

2.2.8. Package and import statements

Everything is fine.

2.2.9. Class and interface declarations

Everything is fine.

2.2.10. Initialization and declarations

Variable *count* depends on a computation can't be initialize immediately at line 351
Variable *i* at line 351 can be declared in the 'for' loop at line 353.
Variable *count* depends on a computation can't be initialize immediately at line 441
Variable *i* at line 441 can be declared in the 'for' loop at line 443.
Variable *count* depends on a computation can't be initialize immediately at line 540
Variable *i* at line 549 can be declared in the 'for' loop at line 551.

2.2.11. Method calls

Everything is fine.

2.2.12. Arrays

Everything is fine.

2.2.13. Object comparison

Everything is fine.

2.2.14. Output Format

Everything is fine.

2.2.15. Computation, comparisons and assignments

Everything is fine.

2.2.16. Exceptions

Everything is fine.

2.2.17. Flow of control

Everything is fine.

2.2.18. Files

Everything is fine, no files.

3. Other problems

Variable ‘remove1’ and ‘remove2’ initializer ‘null’ is redundant at line 362
‘else’ statement with empty body at line 427

4. Software used

The following software were used to redact and do the content of this document:

- NetBeans: for inspecting the code.
- Microsoft Word: this editor was used to redact the document.