INF6306 - Patrons pour la compréhension de programme
TP2 - Code Smells and Refactoring

Isabella Ferreira
isabella.ferreira@polymtl.ca

This project should be done in groups.
Deadline: 10th of April 2020 at 11:59pm
Submission: via Moodle.

1 Goal

Code smells are poor design choices to recurring design problems, and they are opposite to design patterns. Typically, they are introduced by inexperienced developers, and represent common pitfalls in software development. To solve this problem, refactorings can be performed. Refactoring is a software maintenance activity that aims to improve the code design, while preserving the behavior. Thus, the goal of this project is to recognize code smells and practice the application of code refactoring.

2 Task

1. Your task is to identify 5 **different** types of code smells in the projects listed in the file *java_projects.txt*.

2. For each code smell found in the previous step, you should **manually** perform a refactoring. You could use any of the refactorings proposed by the catalog of Martin Fowler (`https://refactoring.com/catalog/`). The usage of tools is not allowed in this step!

3. Code metrics such as LOC, complexity, number of classes, number of calls, coupling, etc should be used to compare the quality of the project *before* and *after* the refactoring.

3.1. To extract code metrics from the source code, I recommend using the tool Understand (`https://scitools.com`). There is a free trial of 2 weeks, or you can ask for student access (`https://scitools.com/student/`).

Bonus

1. Analyze the relationship between technical debt and code smells
2. Analyze the relationship between security vulnerabilities and code smells

   This analysis can be done with SonarQube (`https://www.sonarqube.org`).

3 To submit

- You need to submit a report in pdf format including your name and student id.
- For each code smell and refactoring instance, present the name of the project/class/file with the problem, an UML diagram, a screenshot of the code before and after the refactoring, and the answer to the following questions:

  1. What is the name of the code smell that you found and which refactoring type have you performed to remove it? Also, describe step by step how you found the code smell and how you solved it.

  2. How did the code metrics change before and after the refactoring? Do you see any quality improvement captured by the metrics? Explain why.

  3. In a scale from 1 (less severe) to 5 (most severe), what is the severity of the code smell found? Did you have problem identifying and fixing it?

- Bonus: Analyze if smelly code is related to vulnerabilities and/or technical debt. Provide evidence from the tool and an in-depth analysis.

4 Marking scheme (full mark: 100 points)

- 5 points for each code smell correctly identified by the student
- 5 points for each refactoring correctly performed by the student
- 10 points for code metrics comparison, UML diagram, screenshots, and answer to questions

- Bonus: 5% for each additional analysis