REFACTORING

EXERCISES

#exercises   #java   #refactoring   #oop

REFACTORING

1. SMELLS AND REFACTORING

In these exercises, you will try to identify Code Smells ↗ in some sample code, and apply one or more Refactoring Techniques ↗ to remove them:

- Some problems might not be explicit **smells**.

- Make sure you **read** each explanation about the **smell** and **refactoring before** applying the solution.

- The "Refactoring: improving the design of existing code" ↗ book has a more detailed explanation for each smell and refactoring.

- If you don't understand what a **smell** represents or how a **refactoring** fixes a problem, ask your teacher.

Important: Code refactoring is the process of restructuring existing computer code without changing its external behavior. It's important to understand that, although automatic tools exist, the purpose of refactoring code is **not to automatize** the restructuring process but to **systematize** it.

In small examples, you might not fully comprehend the importance of refactoring, but in large projects, having step-by-step instructions is of paramount importance.

2. DOWNLOAD AND IMPORT

Download the 📄 sample code and open it in IntelliJ.

3. FIND SMELLS AND REFACTOR

- For each package, try finding any code smells (or other kinds of problems) and remove them using refactoring techniques.

- Don't forget to discuss your findings with your teacher.

- Verify if your code still works by running the included tests.

- Tests might need to be changed to accommodate some modifications in the main code.

- The idea of these exercises is to use refactoring techniques. Don't just write the correct code. In a much larger system, rewriting without using proper refactoring techniques might introduce bugs.