Refactoring

Lecturer: Adel Vahdati

Refactoring: Definition

- A change made to the internal structure of software to make it
  - easier to understand, and
  - cheaper to modify.
- The observable behavior of the software should not be changed.

Refactoring: Why?

- Refactoring Improves the Design of Software
- Refactoring Makes Software Easier to Understand
- Refactoring Helps You Find Bugs
- Refactoring Helps You Program Faster

Refactoring: When?

- Refactor the third time you do something similar (The Rule of Three)
- Refactor When You Add Function
- Refactor When You Need to Fix a Bug
- Refactor As You Do a Code Review

Symptoms of Bad Code

- **1. Duplicated Code**
- **2. Long Method**
- **3. Large Class**
- **4. Long Parameter List**
- **5. Divergent Change:** When one class is commonly changed in different ways for different reasons.
- **6. Shotgun Surgery:** When every time you make a kind of change, you have to make a lot of little changes to a lot of different classes.
- **7. Feature Envy:** A method that seems more interested in a class other than the one it actually is in.
- **8. Data Clumps:** Bunches of data that regularly appear together.

Symptoms of Bad Code (2)

- **9. Primitive Obsession:** Excessive use of primitives, due to reluctance to use small objects for small tasks.
- **10. Switch Statements**
- **11. Parallel Inheritance Hierarchies:** Where every time you make a subclass of one class, you also have to make a subclass of another.
- **12. Lazy Class:** A class that isn't doing enough to justify its maintenance.
- **13. Speculative Generality:** Classes and features have been added just because a need for them may arise someday.
- **14. Temporary Field:** An instance variable that is set only in certain circumstances.
- **15. Message Chains:** Transitive visibility chains.

Symptoms of Bad Code (3)

- **16. Middle Man:** Excessive delegation.
- **17. Inappropriate Intimacy:** Excessive interaction and coupling.
- **18. Alternative Classes with Different Interfaces:** Classes that do the same thing but have different interfaces for what they do.
- **19. Incomplete Library Class**
- **20. Data Class:** Classes that have fields, getting and setting methods for the fields, and nothing else.
- **21. Refused Bequest:** When subclasses do not fulfill the commitments of their superclasses.
- **22. Comments:** When comments are used to compensate for bad code.

Refactoring Patterns: Categories

- **Composing Methods:** Packaging code properly
- **Moving Features Between Objects:** Reassigning responsibilities
- **Organizing Data:** Making data easier to work with
- **Simplifying Conditional Expressions:** Making conditional logic less error-prone
- **Making Method Calls Simpler:** Making interfaces easy to understand and use
- **Dealing with Generalization:** Moving features around a hierarchy of inheritance
- **Big Refactorings:** Large-scale changes to code

References

- Fowler, M., Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.
- Fowler, M., Catalog of Refactorings, Published online at: http://refactoring.com/catalog/, December 2013 (last visited on: 1 December 2014).
- Ramsin, Raman. "Home." Department of Computer Science and Engineering, Sharif University of Technology. Accessed February 15, 2025. https://sharif.edu/~ramsin/index.htm.