What is Refactoring?

- Software restructuring and refactoring are used to improve the quality of the software. [5]
- The Techniques that reduce software complexity and improve quality in object-oriented software development.

```java
class A {
    int m1(int a, int b) {
        return a+b;
    }
}
```

After “Rename”

```java
class Tools {
    int addTwo(int a, int b) {
        return a+b;
    }
}
```
Why Teaching Refactoring

- Software engineering courses are significantly lagging behind in providing training on these areas. [2]
- Students are trained on developing new small program from scratch, but not taught how to understand and change exist code. [2]
- We should focus on problem solving, object-oriented concepts, and testing rather than the syntax of the implementation language. [1]

An Innovative Pedagogical to Teaching Refactoring [3, 4]

- The following approach are introduced in CS1 and CS2 students to learn.
- Three introductory lessons for introduction, includes assignments in first year.
- And a continuation lesson for teaching further refactoring techniques.
Concepts In Four Lessons

- Self-documenting code
- encapsulation
- proper use of constant and variables
- good methods
- conditionals
- good classes
- proper inheritance, and others

Lesson One: Self-Documenting Code and Functional Testing

- Reading, reviewing, commenting, and formatting code are not part of the process to refactoring but essential to refactoring.
- Students are required to read and understand poor code in lesson one.
- And then providing meaningful names by the refactoring "Rename".
- Functional testing are introduced in lesson earlier than other CS course.
Lesson One: Self-Documenting Code and Functional Testing (cont.)

- At the end of lesson one, a fairly complex program of 200-1000 lines of code is given as a homework assignment.
- Codes are lightly commented, poorly formatted, and unrefactored.
- Students must resolve these code segments in later homework assignment.
- Version Control System (VCS), and IDE (Eclipse, JBuilder) are also introduced.

Lesson Two: Encapsulation and Unit Testing

- Encapsulation is one of the OO principle that denies access to object internal state.
- In order to work toward more complex refactorings, the stricter form of access is taught in lesson two.
- Unit testing is also introduced in lesson two, students build their own test classes and methods to test refactored code.
Lesson Three: Constant and Variables

- This lesson introduces refactoring techniques that help students to use constant and variables properly.
- For example, students typically do not use named constants effectively.
- A refactoring “Replace Magic Number with Constant” is used to change any literal number to a meaningful named constant.

Lesson Four: Extracting Methods

- Lesson four’s objective is try to write well-formed OO methods
- A well-formed OO methods must be.
  - Acceptably cohesive
  - Low in complexity
  - Appropriately sized;
  - Well documented.
Pedagogical Experience

- A large on-going code example is used to demonstrate refactoring process in lesson one.
- In some exercise, students exchange their code with classmates for peer evaluation.
- A lesson learned from teaching refactoring is to cover the conditional refactoring before applying the criteria should be taught first.
- Another lesson learned is the student who were practicing software developer and write less granular method shows that these concept help improve software quality.

Lesson++;

- Teaching composite refactoring
  - Object-Oriented programming skill.
  - Design patterns.
  - Language neutral.
- My experience in learning refactoring
  - To program like an Art ?!
  - Its difficult to tell the beauty of code when you can not see beyond.
Conclusion

- Refactoring help students to improve their programming style and code.
- Students were impressed with the benefits of integrating the concepts to their application.
- Additionally, students confessed they saw no real benefit to documenting code until lesson four.
- Teaching refactoring is natural fit for software engineering course and also a good practice for software development.

Reference