07-120 Introduction to Software Construction

Fall 2023

Michael Hilton and Mayank Goel

Administrivia

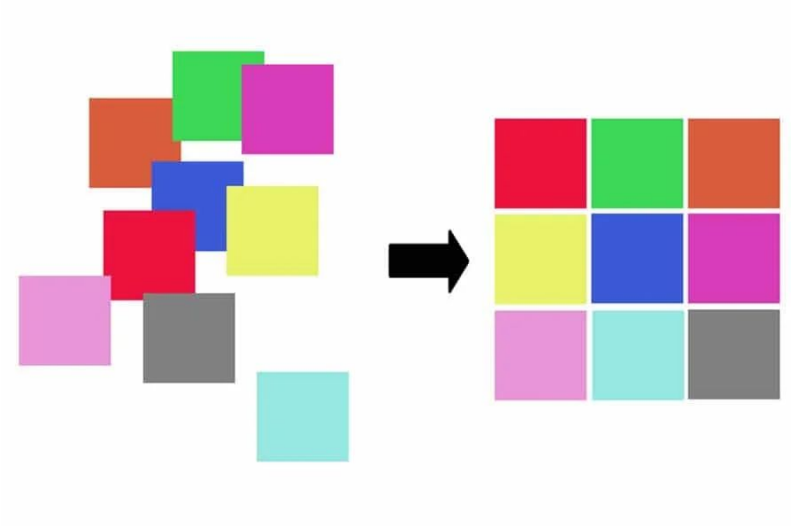
1. Make sure to install Processing IDE on your computers.
2.

Refactoring

What is Refactoring?

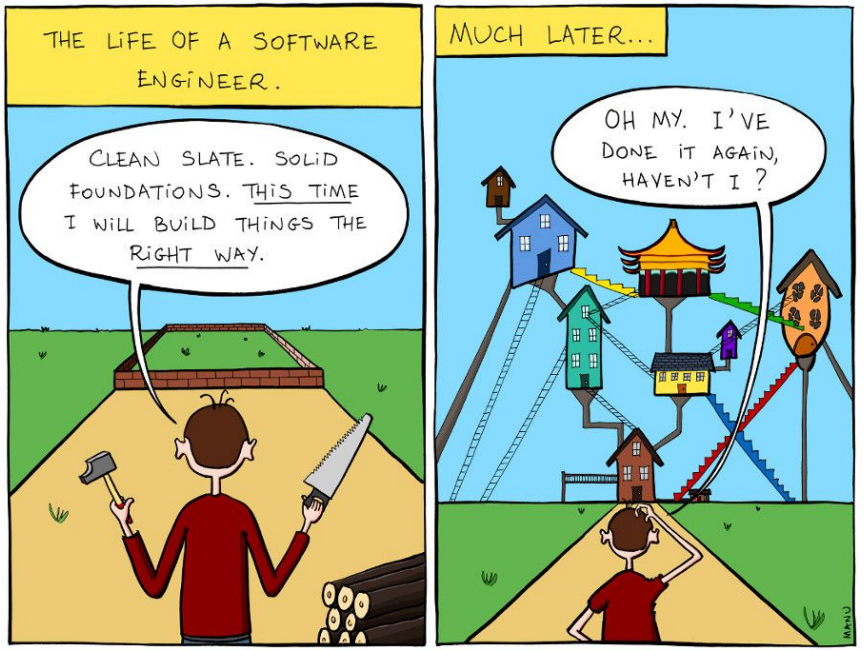
The process of restructuring existing computer code without changing its external behavior.

Any examples?



Why should we refactor?

- Improve/maintain code quality
- Readability
- Understandability
- Collaboration
- Change in requirements
- Better design
- *Google / Stack Overflow / LLM*



Signs that refactoring is needed

- Code Smells
  - certain patterns or characteristics in the source code that may indicate the presence of deeper problems
  - Same as bugs?

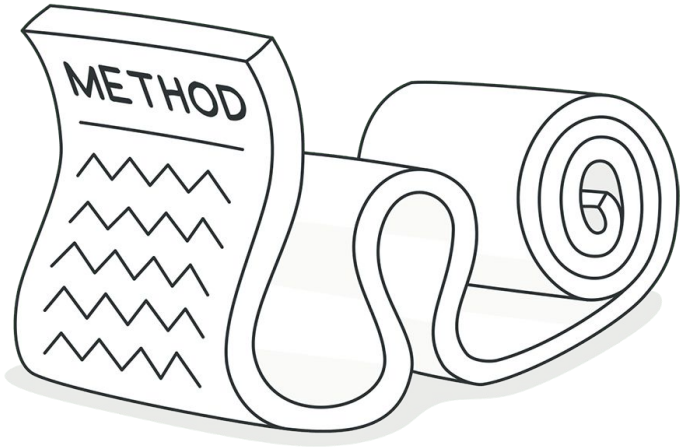
Signs that refactoring is needed: Code Smells

- Duplicate code
  - The same or very similar code appears in multiple places
  - But, why would it happen in the first place?
  - Why is it bad?
    - Makes the code lengthy and bulky
    - Hampers understandability
    - *What if the duplicated code was buggy?*

```c
int calculate_square1(int x) {
    return x * x;
}

int calculate_square2(int y) {
    return y * y;
}

int main() {
    int num1 = 5;
    int num2 = 8;

    int result1 = calculate_square1(num1);
    int result2 = calculate_square2(num2);

    printf("Square of num1: %d\n", result1);
    printf("Square of num2: %d\n", result2);

    return 0;
}
```

Signs that refactoring is needed: Code Smells

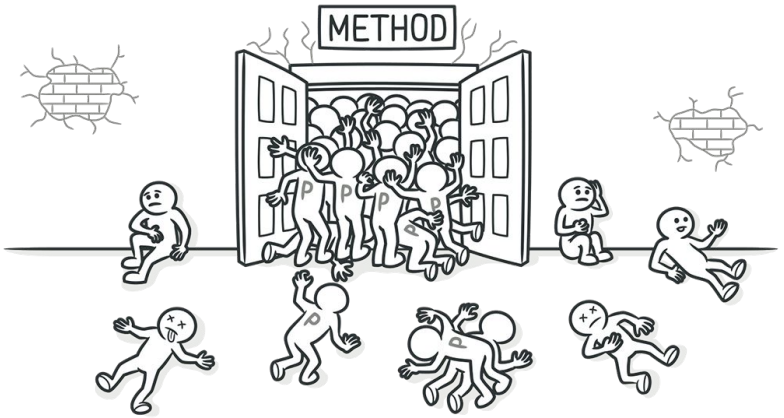
- Long Function / Method
  - “any method longer than ten lines should make you start asking questions.”
  - Why is it bad?
    - Decreases readability
    - Hampers understandability
    - Maybe indicate that a function is doing too much



https://refactoring.guru/smells/long-method

Signs that refactoring is needed: Code Smells

- Long Parameter List
  - A method has many parameters
  - “More than three or four parameters for a method.”
  - Why is it bad?
    - Hampers understandability
    - Hard to maintain
    - Maybe indicate that a function is doing too much



https://refactoring.guru/smells/long-method

Signs that refactoring is needed: Code Smells

- Shotgun Surgery
  - Making any modifications requires that you make many small changes to many different classes.

  - Why fix?
    - Better organization
    - Less code duplication
    - Easier maintenance

```
class SavingsAccount {

  withdraw(amount) {
    if(this.balance < MIN_BALANCE) {
      this.notifyAccountHolder(WITHDRAWAL_MIN_BALANCE);
      return;
    }
    // implementation
  }

  transfer(amount) {
    if(this.balance < MIN_BALANCE) {
      this.notifyAccountHolder(TRANSFER_MIN_BALANCE);
      return;
    }
    // implementation
  }

  processFees(fee) {
    this.balance = this.balance - fee;

    if(this.balance < MIN_BALANCE) {
      this.notifyAccountHolder(MIN_BALANCE_WARNING);
    }
  }
}
```

https://medium.com/thinkster-io/code-smell-shotgun-surgery-66a3f5ed60ea

Basic Principles of Refactoring

- Small incremental changes and keep it simple
  - Maintains stability and reduces the risk of introducing bugs
- Maintain working code
  - Each small change should follow by running tests to ensure that the code still functions correctly.
- Refactor with a purpose
  - Identify why you are refactoring *right now*. E.g., improving readability, reducing duplication, etc.
- Version control
- Refactoring as a continuous process

Consolidate Conditional Expression

```
double disabilityAmount() {
  if (seniority < 2) {
    return 0;
  }
  if (monthsDisabled > 12) {
    return 0;
  }
  if (isPartTime) {
    return 0;
  }
  // Compute the disability amount.
  // ...
}
```

Solution?

Decompose Conditional

```
if (date.before(SUMMER_START) || date.after(SUMMER_END)) {
  charge = quantity * winterRate + winterServiceCharge;
}
else {
  charge = quantity * summerRate;
}
```

Solution?

Add comments? *This is actually a code smell!*

Extract Method

```
void calculateInterest()
{
    //code to calculate interest
    //........
    //........

    printf("Name: %s\n",name);
    printf("Address: %s\n",address);
    printf("Interest owed: %d\n", interest);

    //code to save the calculated interest in the customer's database
    //..........
    //..........
}
```

Solution?

Best Practices and Things to Keep in Mind

- Use descriptive variable names
- Use comments judiciously. Mostly use to explain **why** and not **how**
- Revisit your contracts and tests
- Use diagrams to inform refactoring
- Revisit visualizations and diagrams after refactoring
- When not to refactor?

Best Practices and Things to Keep in Mind

- Use descriptive variable names
- Use comments judiciously. Mostly use to explain **why** and not **how**
- Revisit your contracts and tests
- Use diagrams to inform refactoring
- Revisit visualizations and diagrams after refactoring
- When not to refactor?
  - When the code is not working
  - To fix an error (conditions apply)