

# Clean Code

Jitendra Yashwantrao



# What is Clean Code?

- Easy to understand
- Easy to maintain
- Testable



# What are Code Smells?

- Long Method
- Feature Envy
- Comments (Deodorants)
- Speculative Generality
- Divergent Change
- Shotgun Surgery
- Inappropriate Intimacy
- Long Parameter List
- Utility Classes - Missing Domain Object
- Temporary Variables
- Primitive Obsession
- Switch Case - Polymorphism



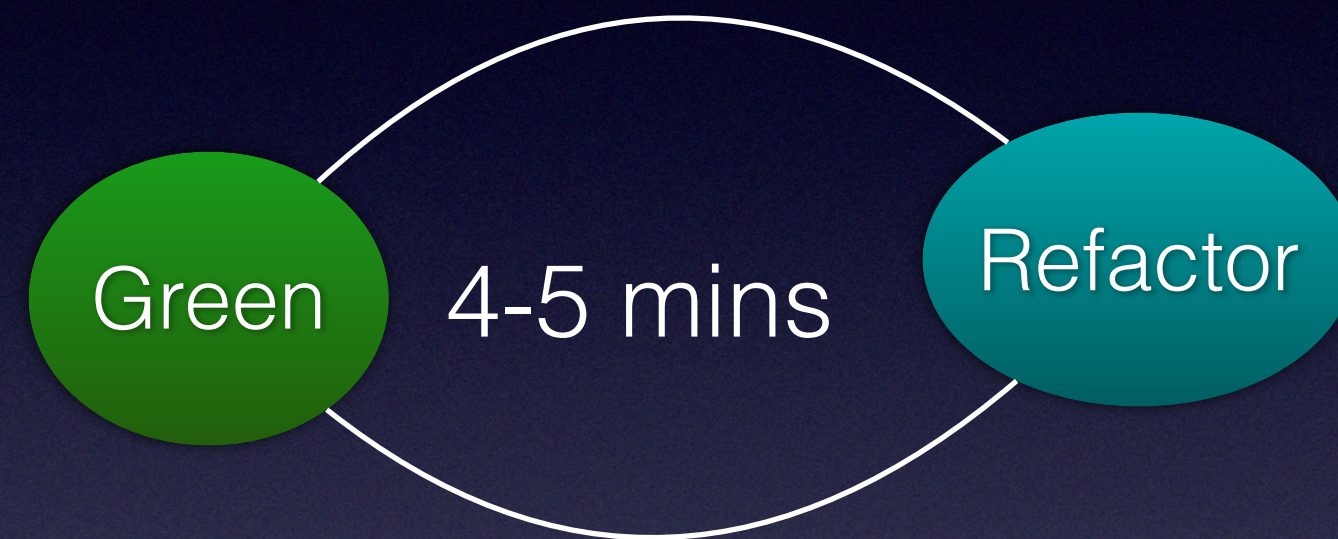
# Refactoring

- Applying change to internal structure of code to make it easy to understand and cheaper to modify without changing its observable behaviour
- Boy Scout Rule - Leave the place (code) cleaner than found
- Cover and Modify
- Refactoring Cycle



# Coding Hats

## Refactoring Hat



- Only refactor to improve design and understanding
- Don't add new test unless necessary
- Don't fix bugs

## Coding Hat



- Add test for functionality / bug
- Add new feature to code / fix bug
- Don't refactor



# Long Method

- Method with more than 10 lines of code or page size lines of code.
- Hard to understand and maintain
- ExtractMethods out
- Decompose Conditions

```
func (c Customer) Statement() string {
    var totalAmount float64
    frequentRenterPoints := 0
    result := "Rental Record for " + c.getName() + "\n"
    for _, each := range c.rentals {
        var thisAmount float64
        //determine amounts for each line
        switch each.getMovie().getPriceCode() {
        case Movie_REGULAR:
            thisAmount += 2
            if each.getDaysRented() > 2 {
                thisAmount += (float64(each.getDaysRented()) - 2) * 1.5
            }
            break
        case Movie_NEW_RELEASE:
            thisAmount += float64(each.getDaysRented()) * 3
            break
        case Movie_CHILDRENS:
            thisAmount += 1.5
            if each.getDaysRented() > 3 {
                thisAmount += (float64(each.getDaysRented()) - 3) * 1.5
            }
            break
        }
        // add frequent renter points
        frequentRenterPoints++
        // add bonus for a two day new release rental
        if each.getMovie().getPriceCode() == Movie_NEW_RELEASE && each.getDaysRented() > 1 {
            frequentRenterPoints++
        }

        //show figures for this rental
        result += "\t" + each.getMovie().getTitle() + "\t" +
            fmt.Sprintf("%f", thisAmount) + "\n"
        totalAmount += thisAmount
    }
    //add footer lines result
    result += "Amount owed is " + fmt.Sprintf("%f", totalAmount) + "\n"
}
```



# Temp Variables

- Variable that get modified by multiple methods or multiple places
- Each method or piece of code should be responsible for one and one thing only

```
17
18 func (c Customer) Statement() string {
19
20     var totalAmount float64      Jitendra-Yashwantrao, 4 months ago • Initial commit
21     frequentRenterPoints := 0
22     result := "Rental Record for " + c.getName() + "\n"
23     for _, rental := range c.rentals {
24         frequentRenterPoints += renterPointFor(rental)
25     }
26     for _, rental := range c.rentals {
27         thisAmount := amountFor(rental)
28         result += "\t" + rental.getMovie().getTitle() + "\t" +
29             fmt.Sprintf("%f", thisAmount) + "\n"
30         totalAmount += thisAmount
31     }
32
33     result += "\tAmount owed is " + fmt.Sprintf("%f", totalAmount) + "\n"
34     result += "\tYou earned " + fmt.Sprintf("%v", frequentRenterPoints) + " frequent re
35     return result
36 }
37
```



# Feature Envy

- Methods that make extensive use of another class may belong in another class. Consider moving this method to the class it is so envious of.

```
17
18 ✓ func (c Customer) Statement() string {
19
20     var totalAmount float64
21     result := "Rental Record for " + c.getName() + "\n"
22
23     for _, rental := range c.rentals {
24         result += "\t" + rental.getMovie().getTitle() + "\t" +
25             fmt.Sprintf("%f", amountFor(rental)) + "\n"
26     }
27
28     totalAmount = totalAmountForRentals(c.rentals)
29
30     result += "\tAmount owed is " + fmt.Sprintf("%f", totalAmount) + "\n"
31     result += "\tYou earned " + fmt.Sprintf("%v", totalRenterPointsFor(c.rentals)) + "\n"
32     return result
33 }
34
35 > func totalRenterPointsFor(rentals []Rental) int { ...
41 }
42
43 > func totalAmountForRentals(rentals []Rental) float64 { ...
49 }
50 | You, seconds ago • Uncommitted changes
51 > func renterPointFor(rental Rental) int { ...
57 }
58
59 > func amountFor(rental Rental) float64 { ...
80 }
81
```



# Comments (Deodorants)

- Added with good intentions but hide code that needs improvement
- Simplifying code expression
- Extracting section of code to new method

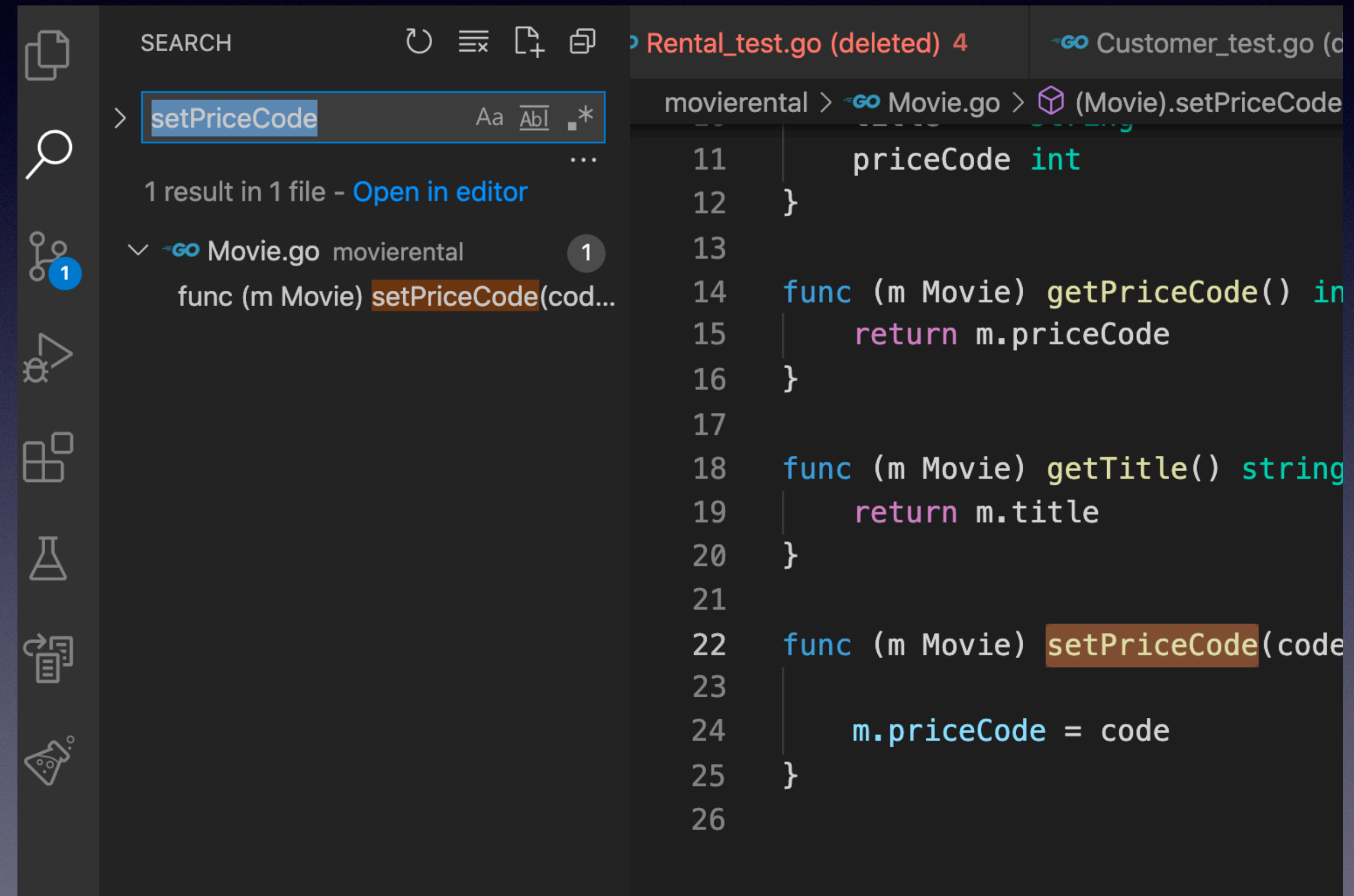
```
func (c Customer) Statement() string {
    var totalAmount float64
    frequentRenterPoints := 0
    result := "Rental Record for " + c.getName() + "\n"
    for _, each := range c.rentals {
        var thisAmount float64
        //determine amounts for each line
        switch each.getMovie().getPriceCode() {
        case Movie_REGULAR:
            thisAmount += 2
            if each.getDaysRented() > 2 {
                thisAmount += (float64(each.getDaysRented()) - 2) * 1.5
            }
            break
        case Movie_NEW_RELEASE:
            thisAmount += float64(each.getDaysRented()) * 3
            break
        case Movie_CHILDRENS:
            thisAmount += 1.5
            if each.getDaysRented() > 3 {
                thisAmount += (float64(each.getDaysRented()) - 3) * 1.5
            }
            break
        }
        // add frequent renter points
        frequentRenterPoints++
        // add bonus for a two day new release rental
        if each.getMovie().getPriceCode() == Movie_NEW_RELEASE && each.getDaysRented() > 1 {
            frequentRenterPoints++
        }

        //show figures for this rental
        result += "\t" + each.getMovie().getTitle() + "\t" +
            fmt.Sprintf("%f", thisAmount) + "\n"
        totalAmount += thisAmount
    }
    //add footer lines result
    result += "Amount owed is " + fmt.Sprintf("%f", totalAmount) + "\n"
}
```



# Speculative Generality

- Code that present for future use
- Unused methods, fields, parameters



The screenshot shows an IDE interface with a search bar at the top. The search bar contains the text 'setPriceCode' and shows '1 result in 1 file - Open in editor'. Below the search bar, a dropdown menu shows the search results: 'Movie.go movierental' with a count of '1'. The search results show the function signature 'func (m Movie) setPriceCode(cod...'. The main editor window displays the implementation of the 'setPriceCode' function in a Go file named 'Rental\_test.go (deleted) 4'. The function signature is 'func (m Movie) setPriceCode(code int) int'. The function body is as follows:

```
11 priceCode int
12 }
13
14 func (m Movie) getPriceCode() int {
15     return m.priceCode
16 }
17
18 func (m Movie) getTitle() string {
19     return m.title
20 }
21
22 func (m Movie) setPriceCode(code int) int {
23
24     m.priceCode = code
25 }
26
```



# Divergent Change

- To introduce one change, requires to change multiple unrelated method within same class



# Shotgun Surgery

- To introduce one change, requires to change multiple classes
- Violation of SRP principle



# Inappropriate Intimacy

- One class using other class details
- Bad coupling
- Bad Responsibility assignment
- <https://blog.devgenius.io/code-smell-64-inappropriate-intimacy-f1b064984094>

```
1  class Candidate {  
2  
3  void printJobAddress(Job job) {  
4  
5      System.out.println("This is your position address");  
6  
7      System.out.println(job.address().street());  
8      System.out.println(job.address().city());  
9      System.out.println(job.address().zipCode());  
10  
11     if (job.address().country() == job.country()) {  
12         System.out.println("It is a local job");  
13     }  
14 }
```



# Primitive Obsession

- Using primitive types like string where small object expected (password, currency, url)



# References

- <https://martinfowler.com/articles/refactoring-video-store-js/>
- <https://refactoring.guru/smells/long-method>
- <https://blog.codinghorror.com/code-smells/>
-



Thank You.