Refactoring

Saving Your Code From Itself

Lorien Rensing

lorien.rensing@gmail.com

https://github.com/makerlorien

@makerlorien

What is refactoring?

Refactoring (noun): a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.

- Martin Fowler, Refactoring: Improving the Design of Existing Code

Refactoring (noun): a small, behavior preserving change.

- Dr. Rebecca Parsons (CTO, ThoughtWorks), The Evolution of Evolutionary Architecture

Refactoring (verb): to restructure software by applying a series of refactorings without changing its observable behavior.

- Martin Fowler, Refactoring: Improving the Design of Existing Code

"Make it work.
Make it right.
Make it fast."
- Kent Beck

Code Smells

Code Smell: A code smell is a surface indication that usually corresponds to a deeper problem in the system.

- Martin Fowler, https://martinfowler.com/bliki/CodeSmell.html

Code Smells

- Are easy to spot

- Do not always indicate a problem

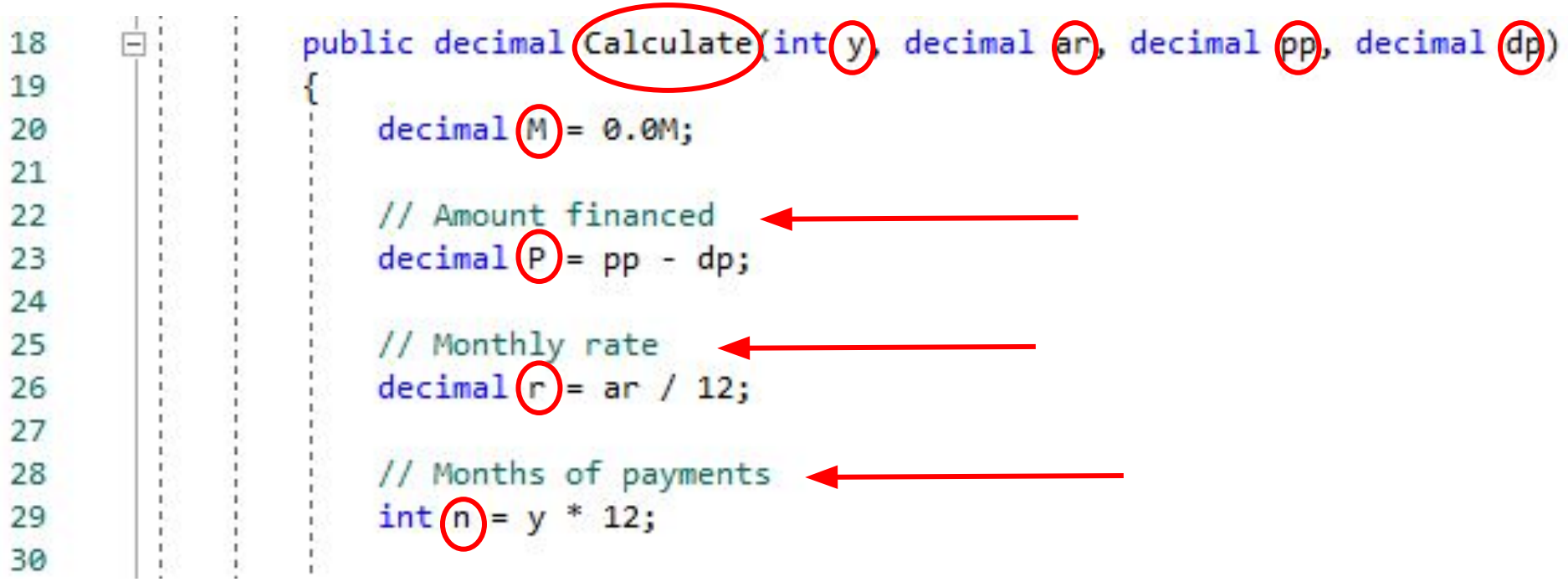
Striped skunk, close by USFWS Mountain-Prairie
Licensed under CC-BY 2.0
Original Source

Blue cheese by Jeremy Keith
Licensed under CC-BY 2.0
Original Source

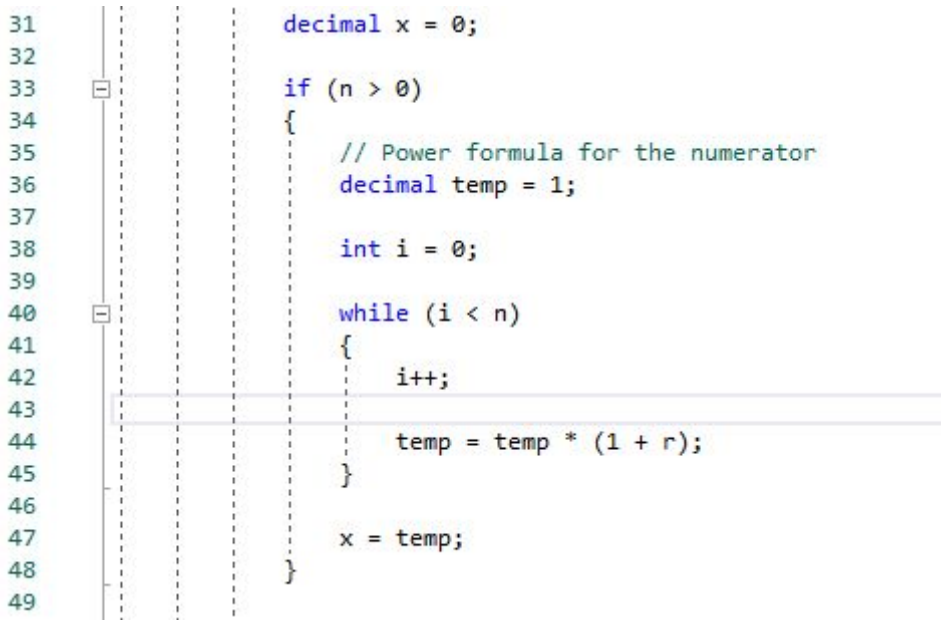
Code smell: Long method

```
18      public decimal Calculate(int y, decimal ar, decimal pp, decimal dp)
19      {
20          decimal M = 0.0M;
21
22          // Amount financed
23          decimal P = pp - dp;
24
25          // Monthly rate
26          decimal r = ar / 12;
27
28          // Months of payments
29          int n = y * 12;
30
31          decimal x = 0;
32
33          if (n > 0)
34          {
35              // Power formula for the numerator
36              decimal temp = 1;
37
38              int i = 0;
39
40              while (i < n)
41              {
42                  i++;
43
44                  temp = temp * (1 + r);
45              }
46
47              x = temp;
48          }
49
50          decimal z = 0;
51
52          if (n > 0)
53          {
54              // Power formula for the denominator
55              decimal temp = 1;
56
57              int i = 1;
58
59              while (i <= n)
60              {
61                  temp = temp * (1 + r);
62                  i++;
63              }
64
65              z = temp;
66          }
67
68          if (z == 1)
69          {
70              // Prevent divide by zero error
71              M = P / n;
72          }
73          else
74          {
75              M = P * ((r * x) / (z - 1));
76          }
77          return M;
78      }
```

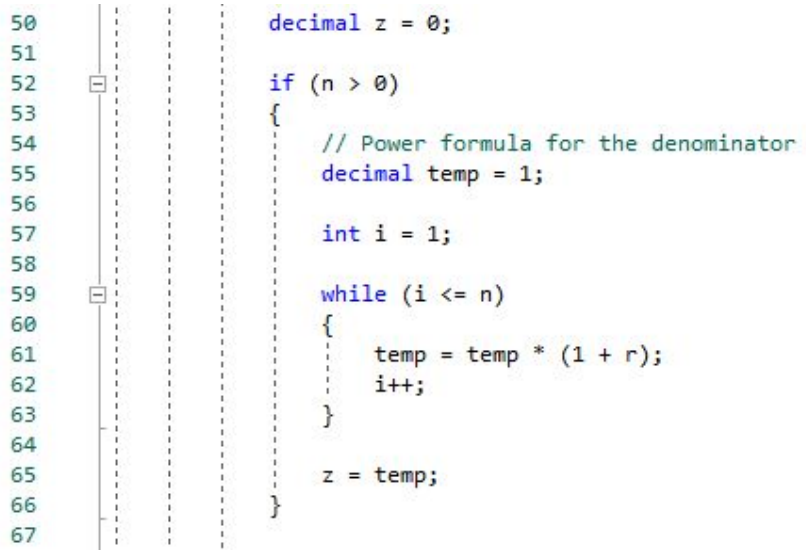
"Measuring software productivity by lines of code is like measuring progress on an airplane by how much it weighs."- Bill Gates



Code smells: comments; uncommunicative names



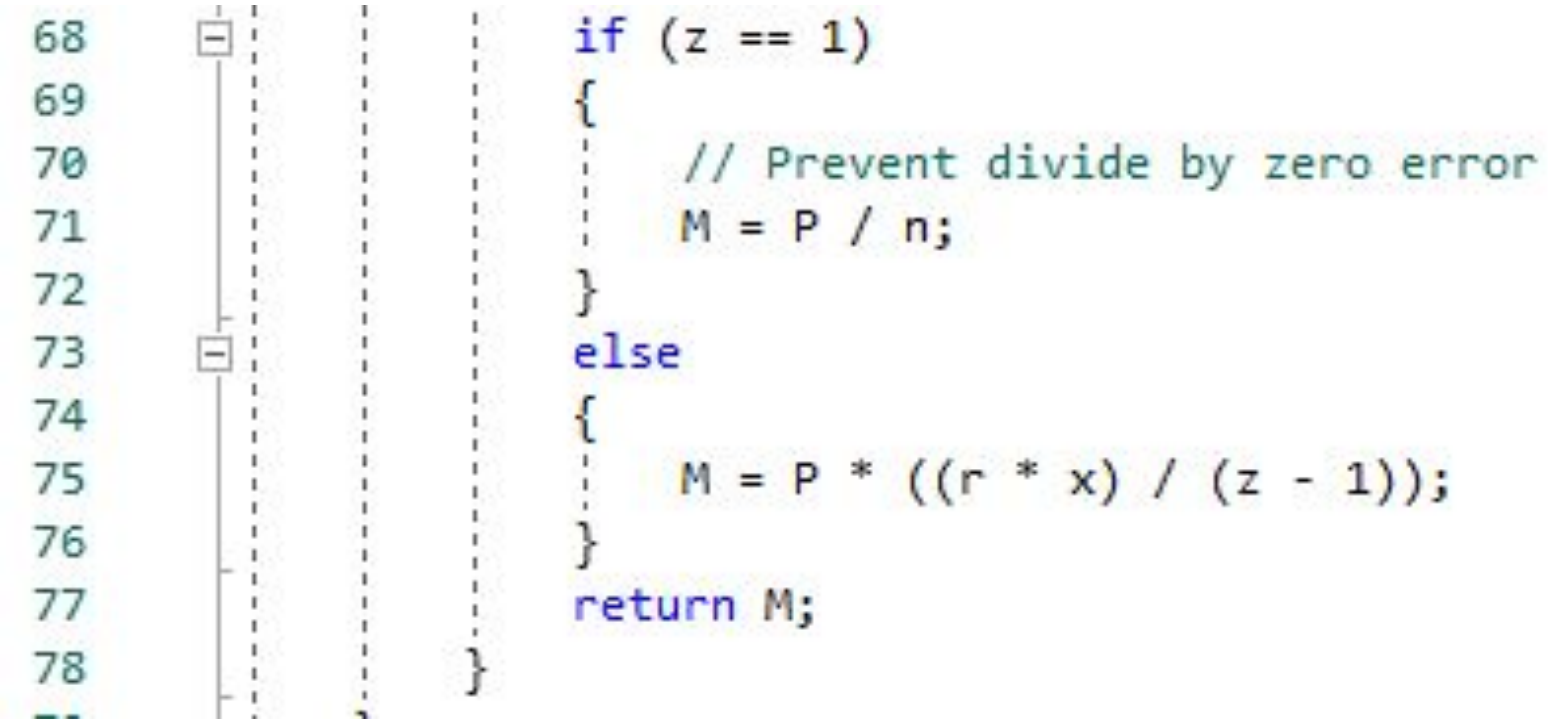
```
31          decimal x = 0;
32
33          if (n > 0)
34          {
35              // Power formula for the numerator
36              decimal temp = 1;
37
38              int i = 0;
39
40              while (i < n)
41              {
42                  i++;
43
44                  temp = temp * (1 + r);
45              }
46
47              x = temp;
48          }
49
```



```
50          decimal z = 0;
51
52          if (n > 0)
53          {
54              // Power formula for the denominator
55              decimal temp = 1;
56
57              int i = 1;
58
59              while (i <= n)
60              {
61                  temp = temp * (1 + r);
62                  i++;
63              }
64
65              z = temp;
66          }
67
```

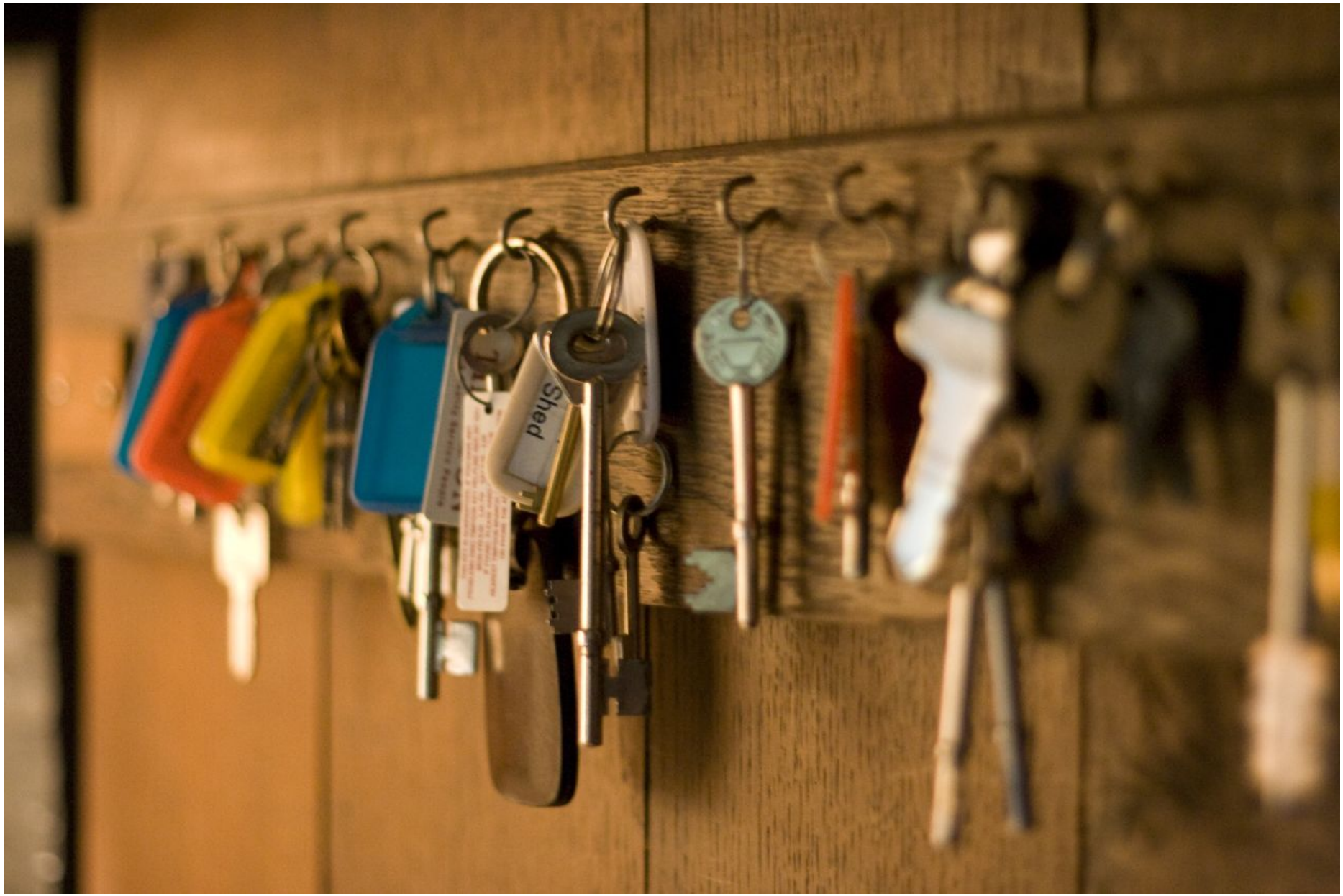
Code smell - duplicated code

Don't
Repeat
Yourself

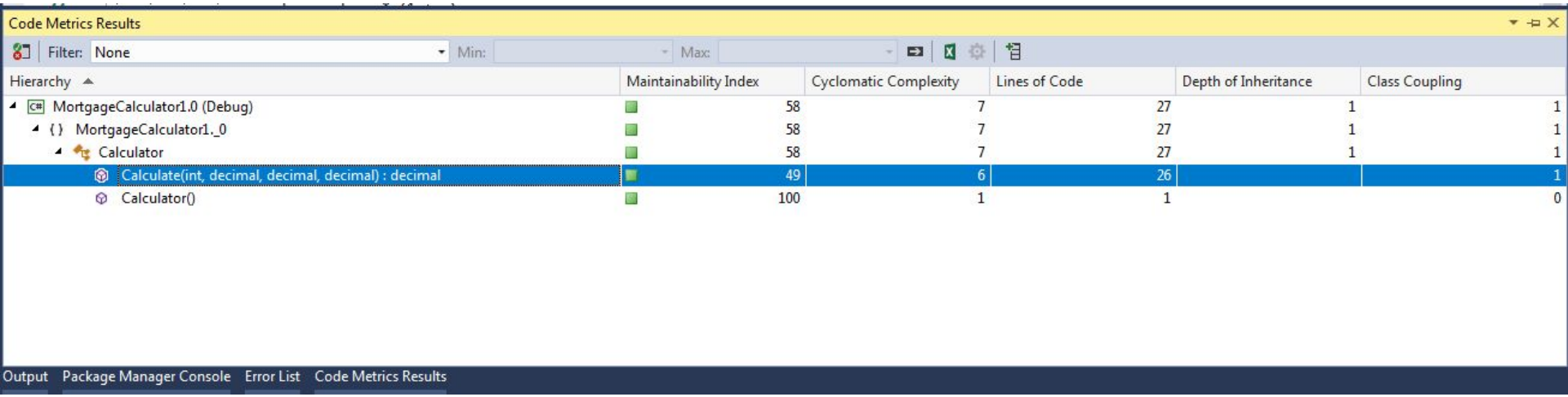


```
68          if (z == 1)
69          {
70              // Prevent divide by zero error
71              M = P / n;
72          }
73          else
74          {
75              M = P * ((r * x) / (z - 1));
76          }
77          return M;
78      }
```

Code smell - poorly organized code



Visual Studio Code Metrics



Maintainability Index

Maintainability Index: The relative ease of maintaining a given section of code

Microsoft's Scoring System

0 - 9: RED

10 - 19: YELLOW

20 - 100: GREEN

Maintainable Software...

- Lets you fix bugs more easily
- Lets you add features more easily
- Lets you spend more time on new work

Cyclomatic Complexity

Cyclomatic Complexity: The number of distinct paths through a method

Low Cyclomatic Complexity (1)

```
public int Sum(int first, int second)
{
    return first + second;
}
```

Low Cyclomatic Complexity (2)

```
public int FindLarger(int first, int second)
{
    if (second > first)
    {
        return second;
    }

    return first;
}
```

Moderate Cyclomatic Complexity (6)

```
public string DetermineLang(string queryLanguage, string languageCookie, string browserLanguage)
{
    var supportedLangs = new List<string> { "en", "es", "fr" };

    if (!string.IsNullOrWhiteSpace(queryLanguage))
    {
        if (supportedLangs.IndexOf(queryLanguage) != -1)
        {
            return queryLanguage;
        }
    }
    if (!string.IsNullOrWhiteSpace(languageCookie))
    {
        if (supportedLangs.IndexOf(languageCookie) != -1)
        {
            return languageCookie;
        }
    }

    return supportedLangs.IndexOf(browserLanguage) != -1 ? browserLanguage : "en";
}
```

High Cyclomatic Complexity (21)

```
public void UpdateQuality()
{
    for (var i = 0; i < Items.Count; i++)
    {
        if (Items[i].Name != "Aged Brie" && Items[i].Name != "Backstage passes to a TAFKAL80ETC concert")
        {
            if (Items[i].Quality > 0)
            {
                if (Items[i].Name != "Sulfuras, Hand of Ragnaros")
                {
                    Items[i].Quality = Items[i].Quality - 1;
                }
            }
        }
        else
        {
            if (Items[i].Quality < 50)
            {
                Items[i].Quality = Items[i].Quality + 1;

                if (Items[i].Name == "Backstage passes to a TAFKAL80ETC concert")
                {
                    if (Items[i].SellIn < 11)
                    {
                        if (Items[i].Quality < 50)
                        {
                            Items[i].Quality = Items[i].Quality + 1;
                        }
                    }

                    if (Items[i].SellIn < 6)
                    {
                        if (Items[i].Quality < 50)
                        {
                            Items[i].Quality = Items[i].Quality + 1;
                        }
                    }
                }
            }
        }

        if (Items[i].Name != "Sulfuras, Hand of Ragnaros")
        {
            Items[i].SellIn = Items[i].SellIn - 1;
        }

        if (Items[i].SellIn < 0)
        {
            if (Items[i].Name != "Aged Brie")
            {
                if (Items[i].Name != "Backstage passes to a TAFKAL80ETC concert")
                {
                    if (Items[i].Quality > 0)
                    {
                        if (Items[i].Name != "Sulfuras, Hand of Ragnaros")
                        {
                            Items[i].Quality = Items[i].Quality - 1;
                        }
                    }
                }
                else
                {
                    Items[i].Quality = Items[i].Quality - Items[i].Quality;
                }
            }
            else
            {
                if (Items[i].Quality < 50)
                {
                    Items[i].Quality = Items[i].Quality + 1;
                }
            }
        }
    }
}
```

Lines of Code

Lines of Code refers only to lines of

executable code

Lines of Code does **NOT** refer to...

- Whitespace
- Comments
- Curly braces on their own line

More lines of code

==

More code to maintain

Original Code Metrics

	Maintainability Index	Cyclomatic Complexity	Lines of Code
Original	49	6	26

```
18      public decimal Calculate(int y, decimal ar, decimal pp, decimal dp)
19      {
20          decimal M = 0.0M;
21
22          // Amount financed
23          decimal P = pp - dp;
24
25          // Monthly rate
26          decimal r = ar / 12;
27
28          // Months of payments
29          int n = y * 12;
30
```

```
18      public decimal CalculateMonthlyPayment(int yearsInMortgage, decimal annualInterestRate, decimal purchasePrice, decimal downPayment)
19      {
20          decimal monthlyPayment = 0.0M;
21
22          decimal principalFinanced = purchasePrice - downPayment;
23
24          decimal monthlyInterestRate = annualInterestRate / 12;
25
26          int monthsInMortgage = yearsInMortgage * 12;
27
```

Step 1: Rename variables and methods

Code Metrics - After Step 1

	Maintainability Index	Cyclomatic Complexity	Lines of Code
Original	49	6	26
Step 1	49	6	26

```
31          decimal x = 0;
32
33          if (n > 0)
34          {
35              // Power formula for the numerator
36              decimal temp = 1;
37
38              int i = 0;
39
40              while (i < n)
41              {
42                  i++;
43
44                  temp = temp * (1 + r);
45              }
46
47              x = temp;
48          }
49
```

```
50          decimal z = 0;
51
52          if (n > 0)
53          {
54              // Power formula for the denominator
55              decimal temp = 1;
56
57              int i = 1;
58
59              while (i <= n)
60              {
61                  temp = temp * (1 + r);
62                  i++;
63              }
64
65              z = temp;
66          }
67
```

Step 2: Extract Duplicate Code

```
27
28          decimal numeratorPower = 0;
29
30          if (monthsInMortgage > 0)
31          {
32              numeratorPower = RaiseDecimalToPowerOfTerm(monthlyInterestRate, monthsInMortgage);
33          }
34
35          decimal denominatorPower = 0;
36
37          if (monthsInMortgage > 0)
38          {
39              denominatorPower = RaiseDecimalToPowerOfTerm(monthlyInterestRate, monthsInMortgage);
40          }
41
```

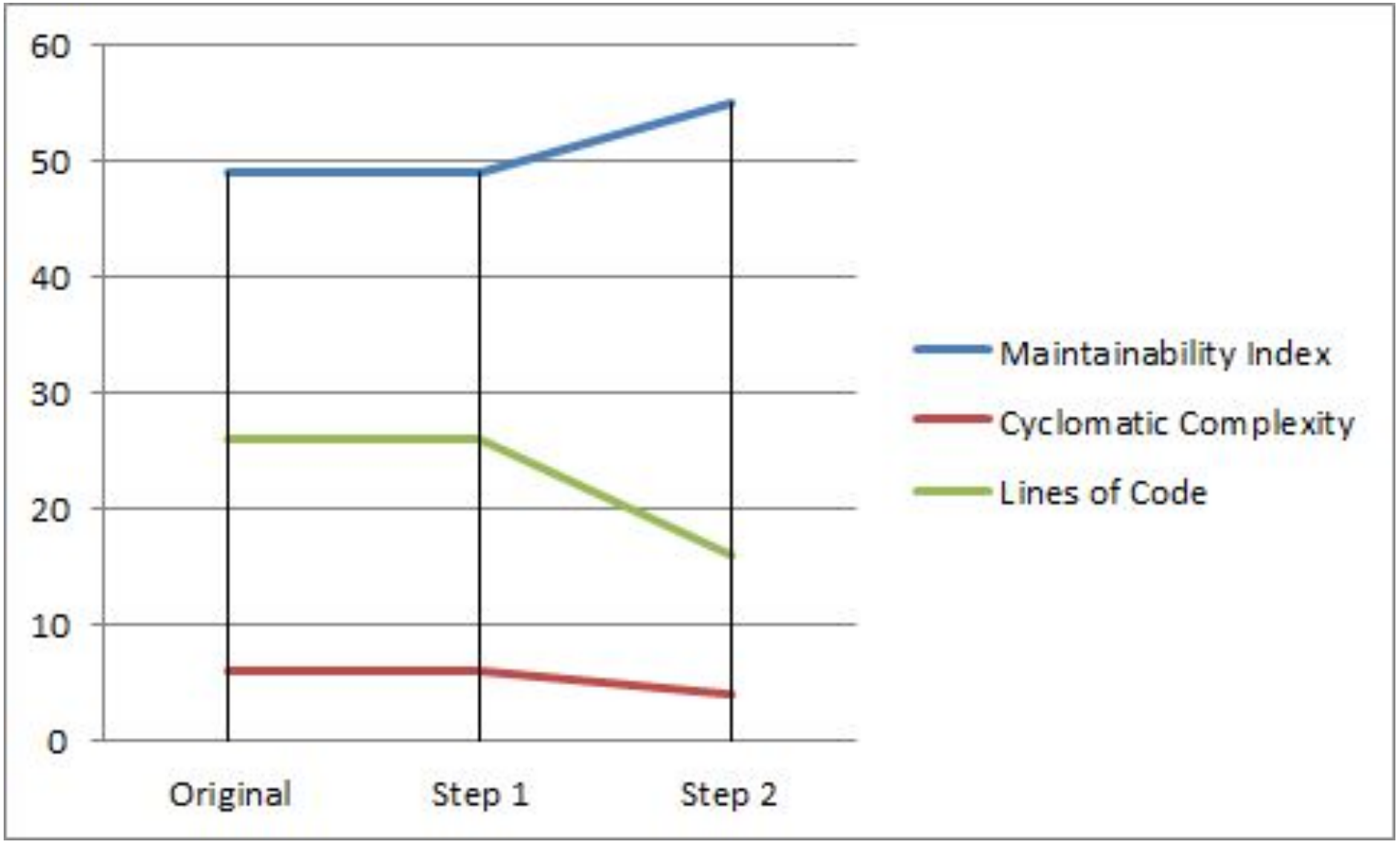
```
54      private static decimal RaiseDecimalToPowerOfTerm(decimal rate, int term)
55      {
56          decimal rateToPowerOfTerm = 1;
57
58          int i = 1;
59
60          while (i <= term)
61          {
62              i++;
63
64              rateToPowerOfTerm = rateToPowerOfTerm * (1 + rate);
65          }
66
67          return rateToPowerOfTerm;
68      }
```

Step 2: Extract duplicate code

Code Metrics - After Step 2

	Maintainability Index	Cyclomatic Complexity	Lines of Code
Original	49	6	26
Step 1	49	6	26
Step 2	55	4	16

Code Metrics - After Step 2



```
27
28          decimal numeratorPower = 0;
29
30          if (monthsInMortgage > 0)
31          {
32              numeratorPower = RaiseDecimalToPowerOfTerm(monthlyInterestRate, monthsInMortgage);
33          }
34
35          decimal denominatorPower = 0;
36
37          if (monthsInMortgage > 0)
38          {
39              denominatorPower = RaiseDecimalToPowerOfTerm(monthlyInterestRate, monthsInMortgage);
40          }
41
```

Step 3: Extract Methods

```
27
28              decimal numerator = 1.0M;
29
30              if (monthsInMortgage > 0)
31              {
32                  numerator = CalculateNumerator(monthlyInterestRate, monthsInMortgage);
33              }
34              decimal denominator = 1.0M;
35
36              if (monthsInMortgage > 0)
37              {
38                  denominator = CalculateDenominator(monthlyInterestRate, monthsInMortgage);
39              }
40
```

```
68          private decimal CalculateNumerator(decimal monthlyRate, int numberOfMonths)
69          {
70              decimal numeratorPower = RaiseDecimalToPowerOfTerm(monthlyRate, numberOfMonths);
71
72              return monthlyRate * numeratorPower;
73          }
74
75          private decimal CalculateDenominator(decimal monthlyRate, int numberOfMonths)
76          {
77              decimal denominatorPower = RaiseDecimalToPowerOfTerm(monthlyRate, numberOfMonths);
78
79              return denominatorPower - 1;
80          }
```

Step 3: Extract Methods

Code Metrics - After Step 3

	Maintainability Index	Cyclomatic Complexity	Lines of Code
Original	49	6	26
Step 1	49	6	26
Step 2	55	4	16
Step 3	55	4	16

```
68          if (z == 1)
69          {
70              // Prevent divide by zero error
71              M = P / n;
72          }
73          else
74          {
75              M = P * ((r * x) / (z - 1));
76          }
77          return M;
78      }
```

Step 4: Re-organize code

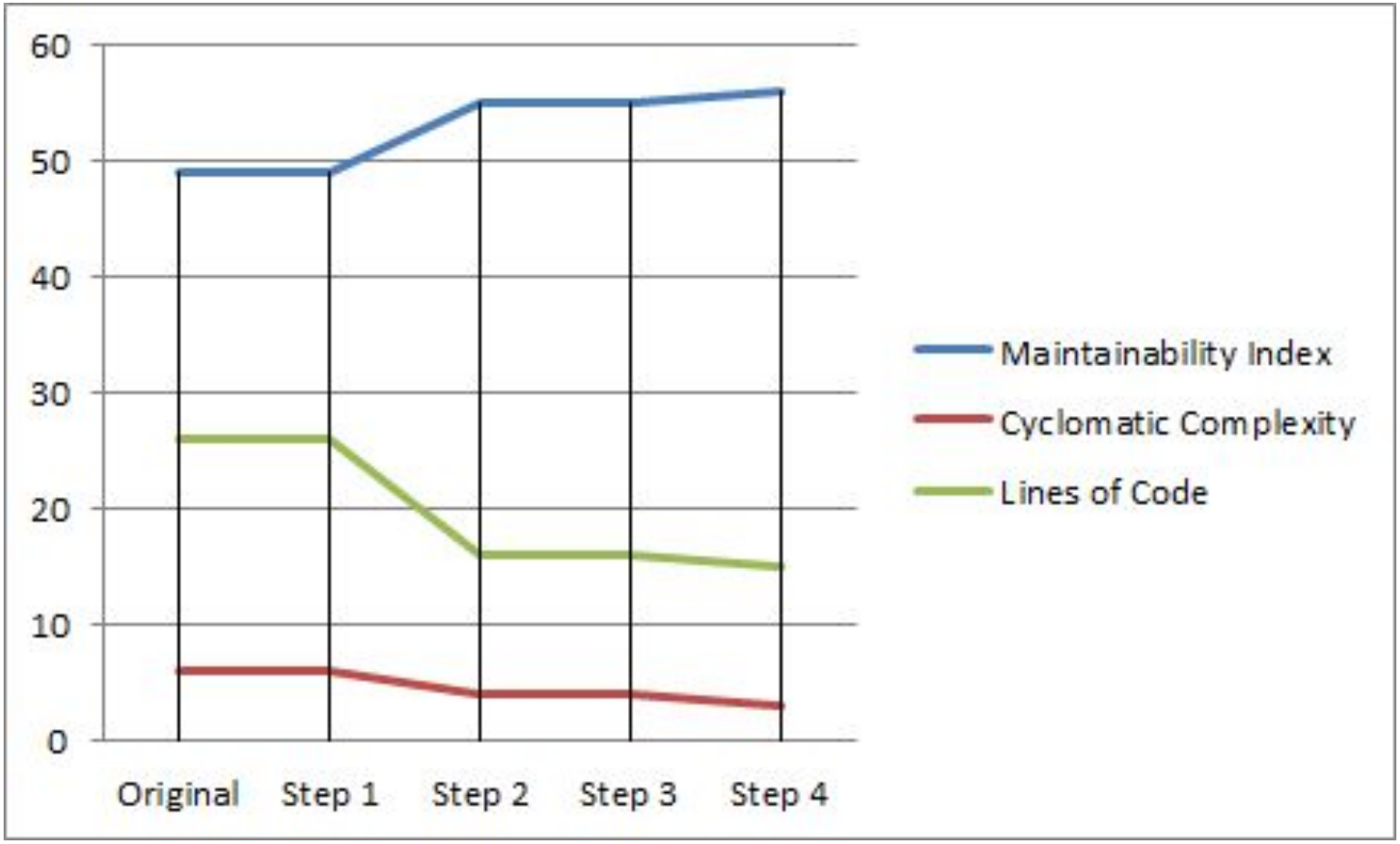
```
18      public decimal CalculateMonthlyPayment(int yearsInMortgage, decimal annualInterestRate, decimal purchasePrice, decimal downPayment)
19      {
20          decimal monthlyPayment = 0.0M;
21
22          decimal principalFinanced = purchasePrice - downPayment;
23
24          if (yearsInMortgage == 0)
25          {
26              return principalFinanced;
27          }
28
29          int monthsInMortgage = yearsInMortgage * 12;
30
31          if (annualInterestRate == 0)
32          {
33              return principalFinanced / monthsInMortgage;
34          }
35
36          decimal monthlyInterestRate = annualInterestRate / 12;
37
38          decimal numerator = 1.0M;
39          decimal denominator = 1.0M;
40
41          numerator = CalculateNumerator(monthlyInterestRate, monthsInMortgage);
42          denominator = CalculateDenominator(monthlyInterestRate, monthsInMortgage);
43
44          monthlyPayment = principalFinanced * (numerator / denominator);
45          return monthlyPayment;
46      }
```

Step 4: Reorganize code

Code Metrics - After Step 4

	Maintainability Index	Cyclomatic Complexity	Lines of Code
Original	49	6	26
Step 1	49	6	26
Step 2	55	4	16
Step 3	55	4	16
Step 4	56	3	15

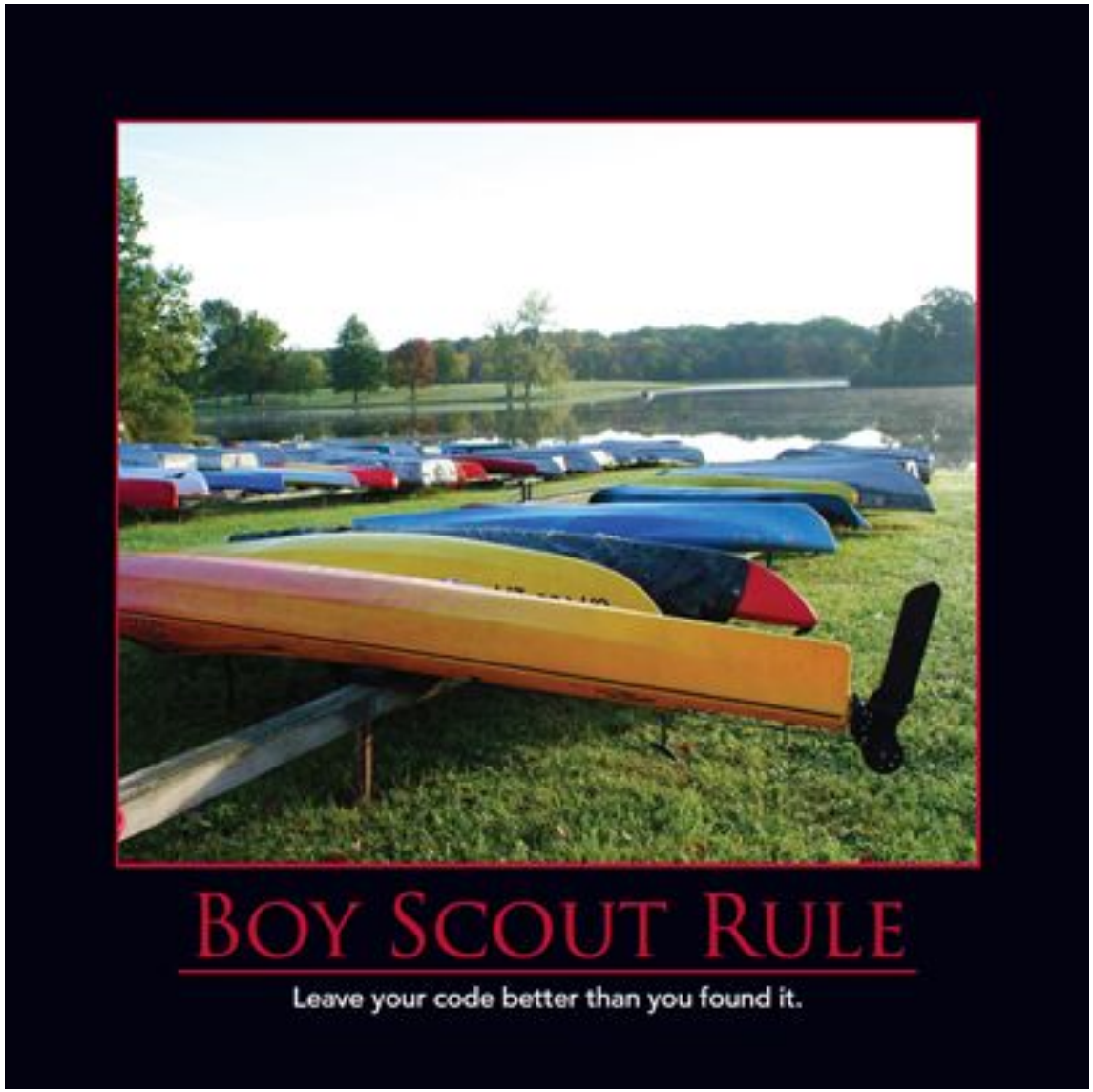
Code Metrics - Trends



When should I refactor?

Opportunistic Refactoring

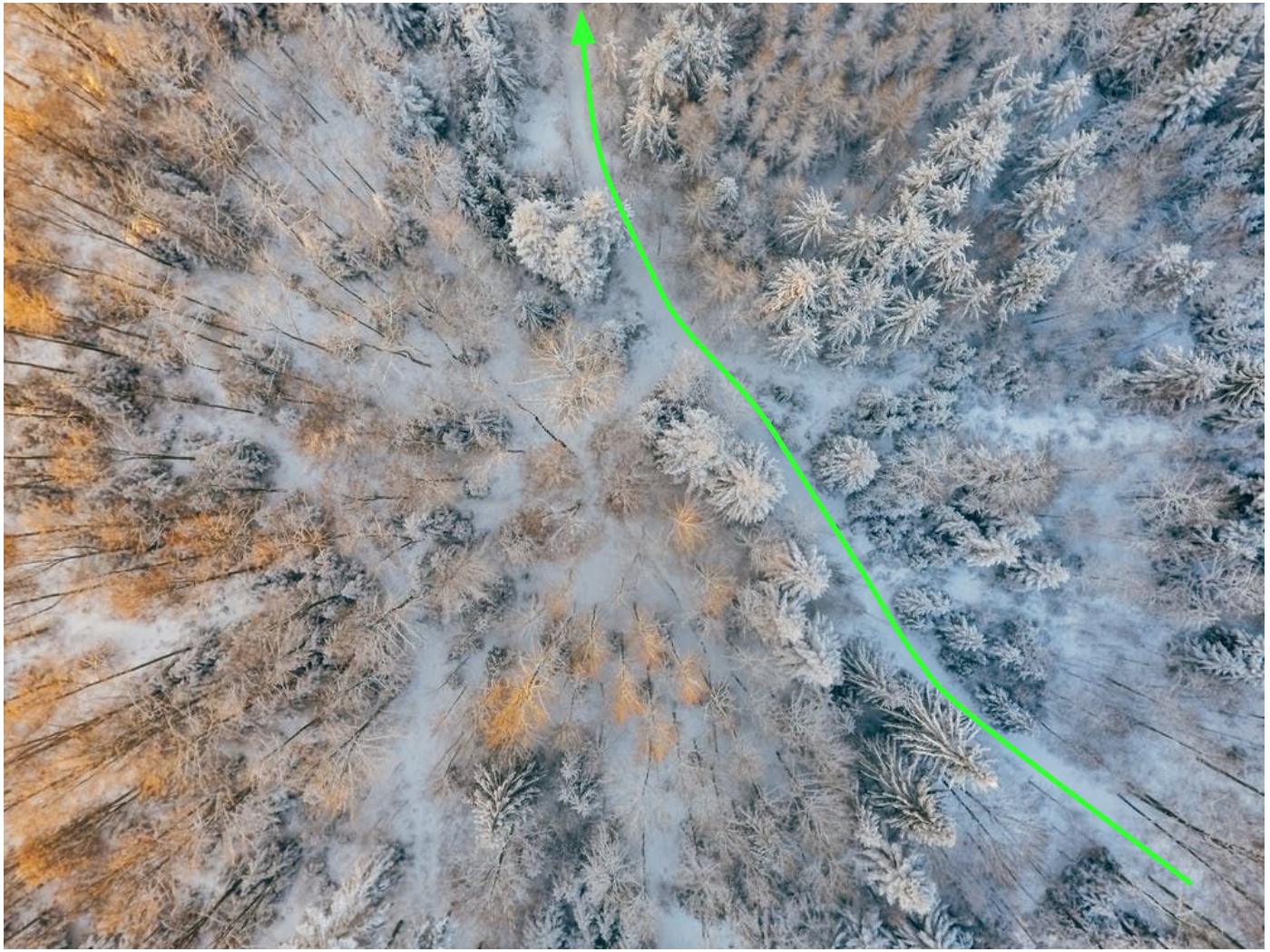
Leave It Better Than You Found It



What should I do if I don't have unit tests?

Write tests for new code

Find spots you can test existing code



Smoke tests to cover existing code

Questions?

Other Refactoring Resources

https://martinfowler.com/

https://martinfowler.com/articles/refactoring-2nd-ed.html

https://www.sandimetz.com/99bottles/

https://ardalis.com/when-should-you-refactor

https://www.pluralsight.com/courses/refactoring-fundamentals

Other Tools

NDepend - in depth metrics for .NET

https://www.ndepend.com/

JetBrains - IDEs with built in code metrics, refactoring tools

https://www.jetbrains.com/

Lorien Rensing

lorien.rensing@gmail.com

https://github.com/makerlorien

@makerlorien