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 (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

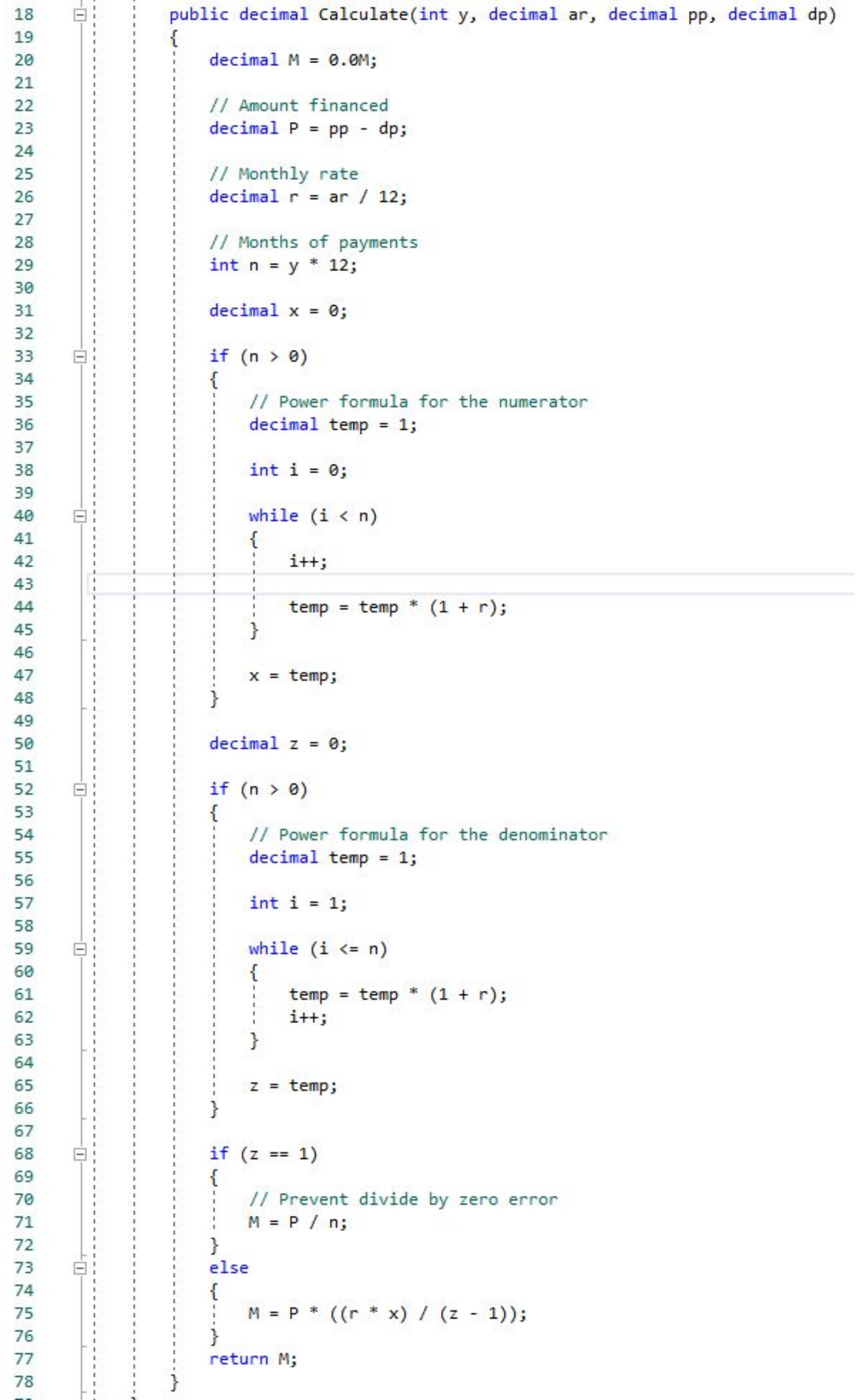
Code Smells

- Are easy to spot

- Do not always indicate a problem

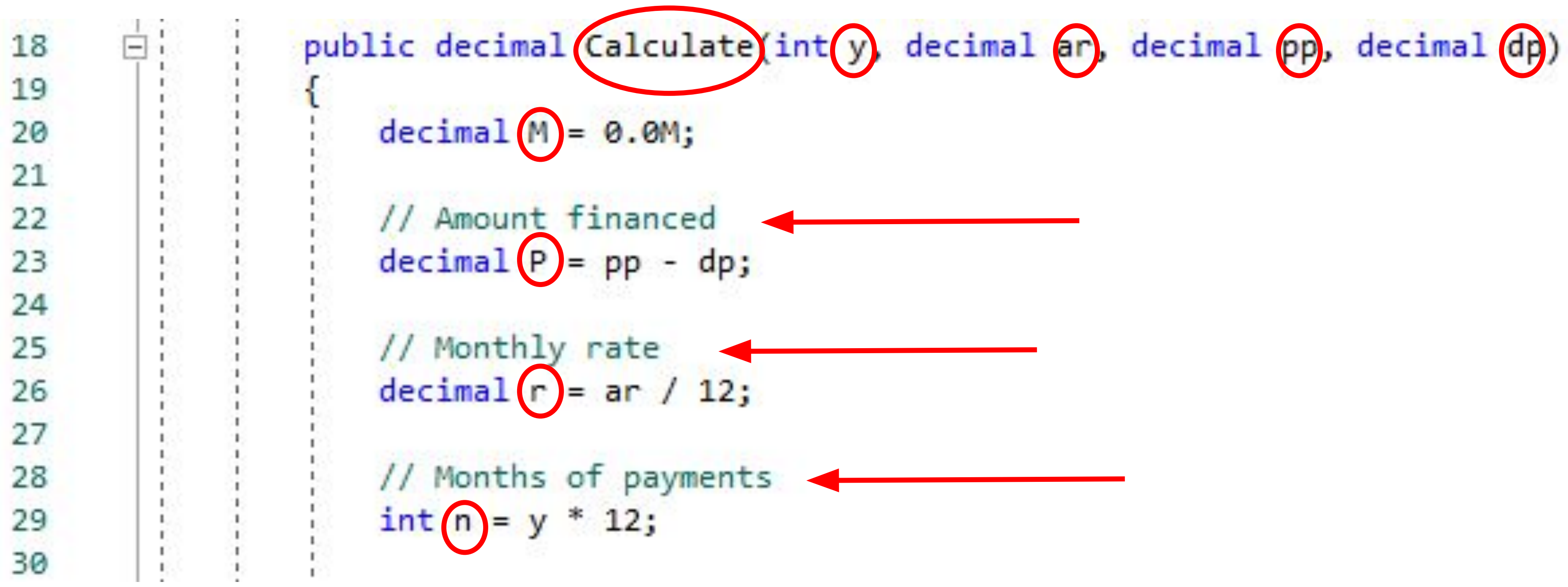
Striped skunk, close by USFWS Mountain-Prairie
Licensed under CC-BY 2.0
Original Source

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


```
public decimal Calculate(int y, decimal ar, decimal pp, decimal dp)
{
    decimal M = 0.0M;

    // Amount financed
    decimal P = pp - dp;

    // Monthly rate
    decimal r = ar / 12;

    // Months of payments
    int n = y * 12;

    decimal x = 0;

    if (n > 0)
    {
        // Power formula for the numerator
        decimal temp = 1;

        int i = 0;

        while (i < n)
        {
            i++;

            temp = temp * (1 + r);
        }

        x = temp;
    }

    decimal z = 0;

    if (n > 0)
    {
        // Power formula for the denominator
        decimal temp = 1;

        int i = 1;

        while (i <= n)
        {
            temp = temp * (1 + r);
            i++;
        }

        z = temp;
    }

    if (z == 1)
    {
        // Prevent divide by zero error
        M = P / n;
    }
    else
    {
        M = P * ((r * x) / (z - 1));
    }
    return M;
}
```

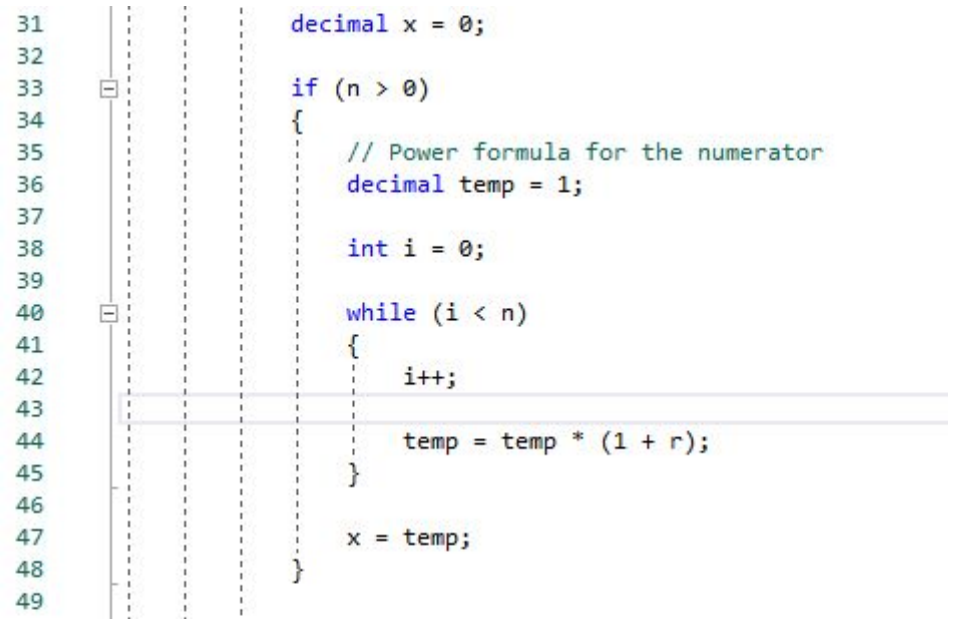
Code smell: Long method

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

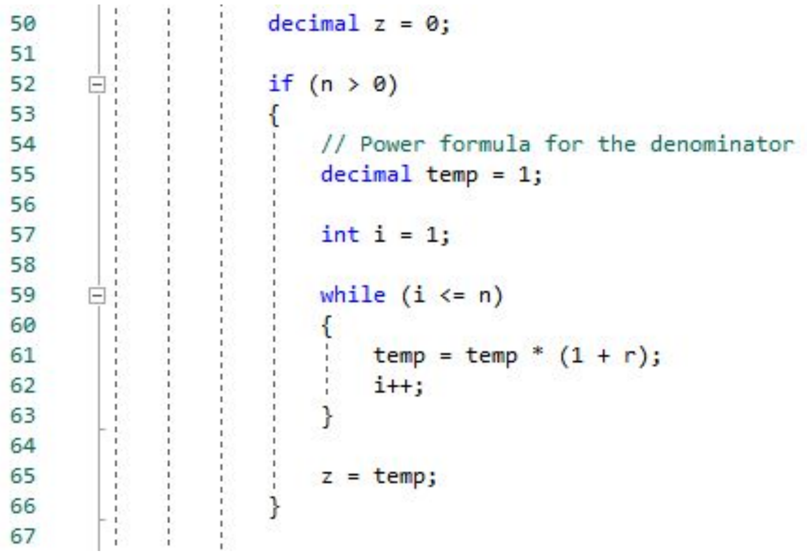


```
public decimal Calculate(int y, decimal ar, decimal pp, decimal dp)
{
    decimal M = 0.0M;

    // Amount financed
    decimal P = pp - dp;

    // Monthly rate
    decimal r = ar / 12;

    // Months of payments
    int n = y * 12;

```

Code smells: comments; uncommunicative names



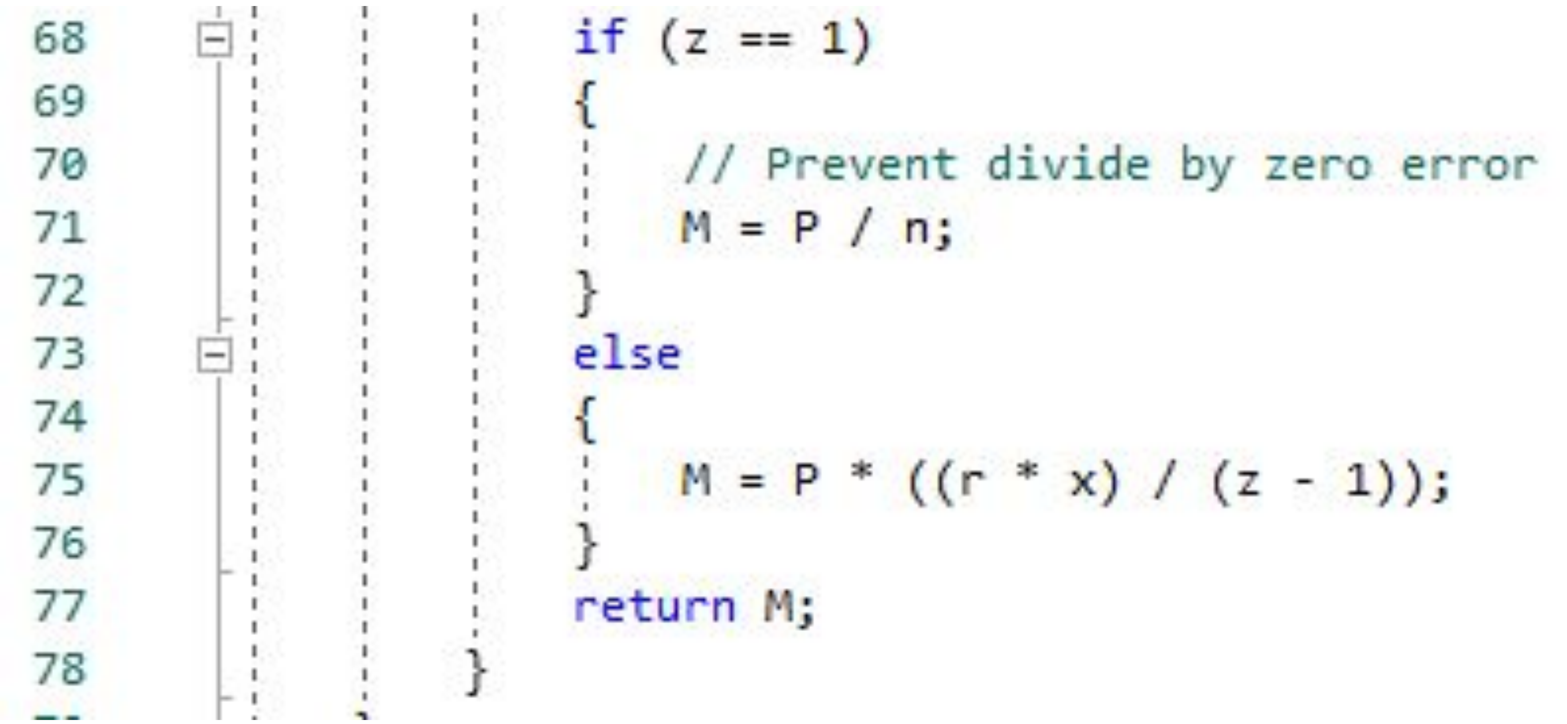
```
decimal x = 0;

if (n > 0)
{
    // Power formula for the numerator
    decimal temp = 1;

    int i = 0;

    while (i < n)
    {
        i++;

        temp = temp * (1 + r);
    }

    x = temp;
}
```



```
decimal z = 0;

if (n > 0)
{
    // Power formula for the denominator
    decimal temp = 1;

    int i = 1;

    while (i <= n)
    {
        temp = temp * (1 + r);
        i++;
    }

    z = temp;
}
```

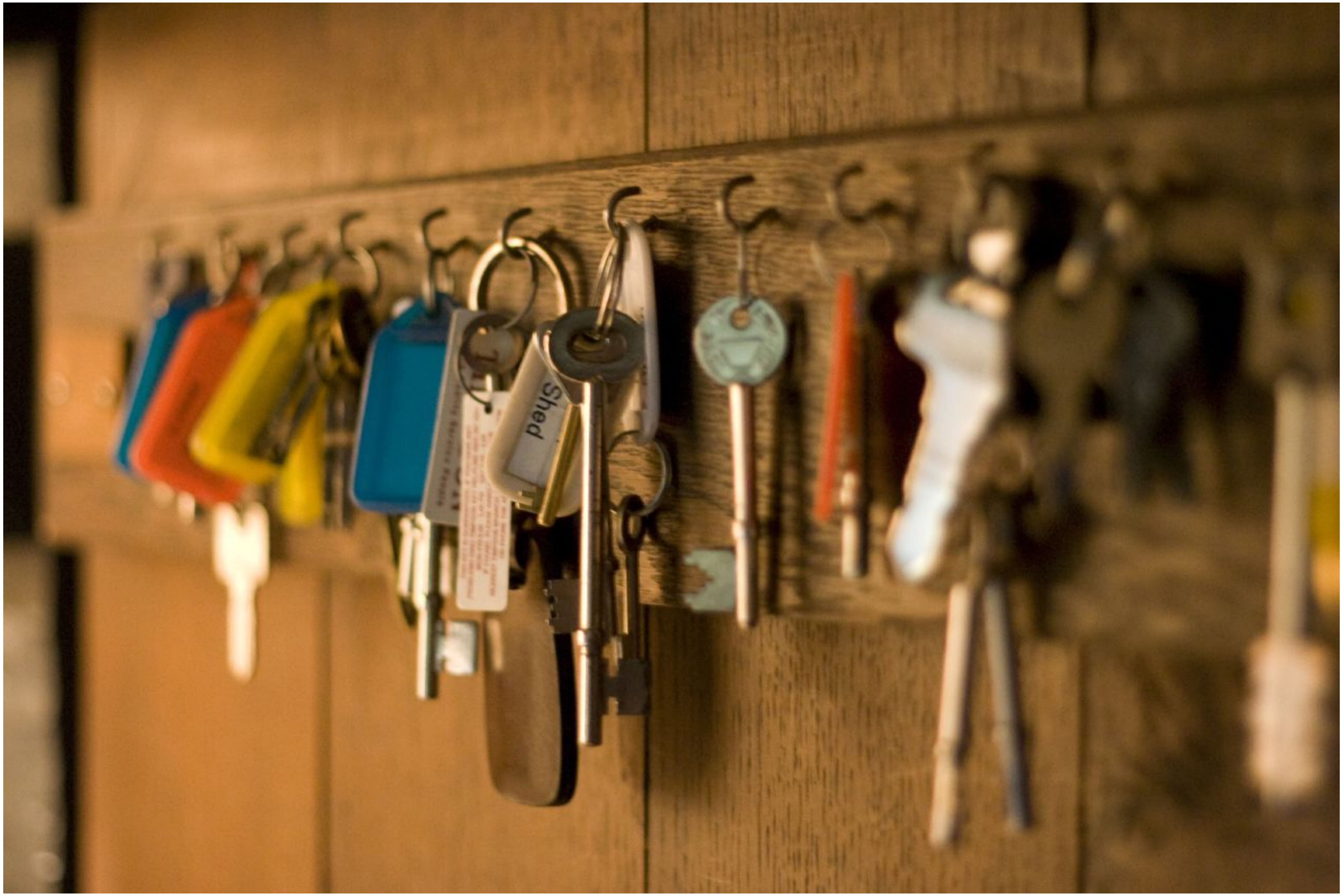
Code smell - duplicated code

Don't
Repeat
Yourself



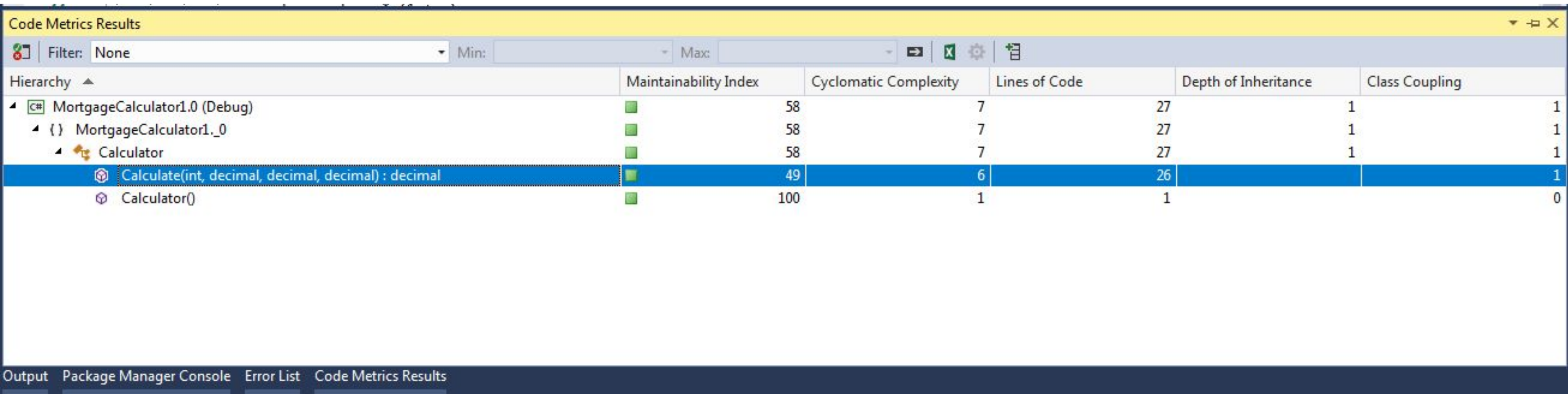
```
            if (z == 1)
            {
                // Prevent divide by zero error
                M = P / n;
            }
            else
            {
                M = P * ((r * x) / (z - 1));
            }
            return M;
        }
```

Code smell - poorly organized code



Keys by Jessica Paterson
Licensed under CC-BY 2.0
Original Source

Visual Studio Code Metrics


Code Metrics Results

Filter: None Min: Max:

Hierarchy	Maintainability Index	Cyclomatic Complexity	Lines of Code	Depth of Inheritance	Class Coupling
MortgageCalculator1.0 (Debug)	🟩 58	7	27	1	1
MortgageCalculator1._0	🟩 58	7	27	1	1
Calculator	🟩 58	7	27	1	1
Calculate(int, decimal, decimal, decimal) : decimal	🟩 49	6	26		1
Calculator()	🟩 100	1	1		0

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

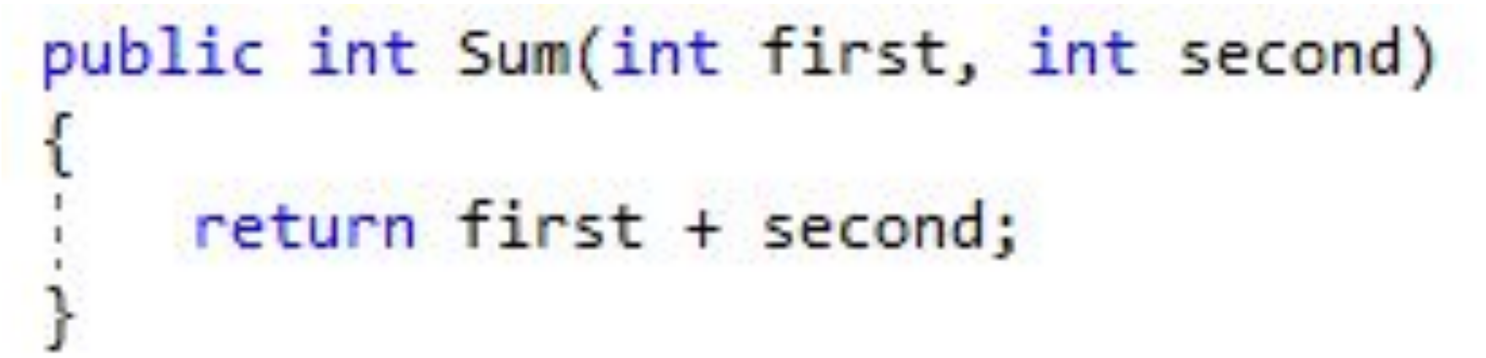
Maintainable Software...

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

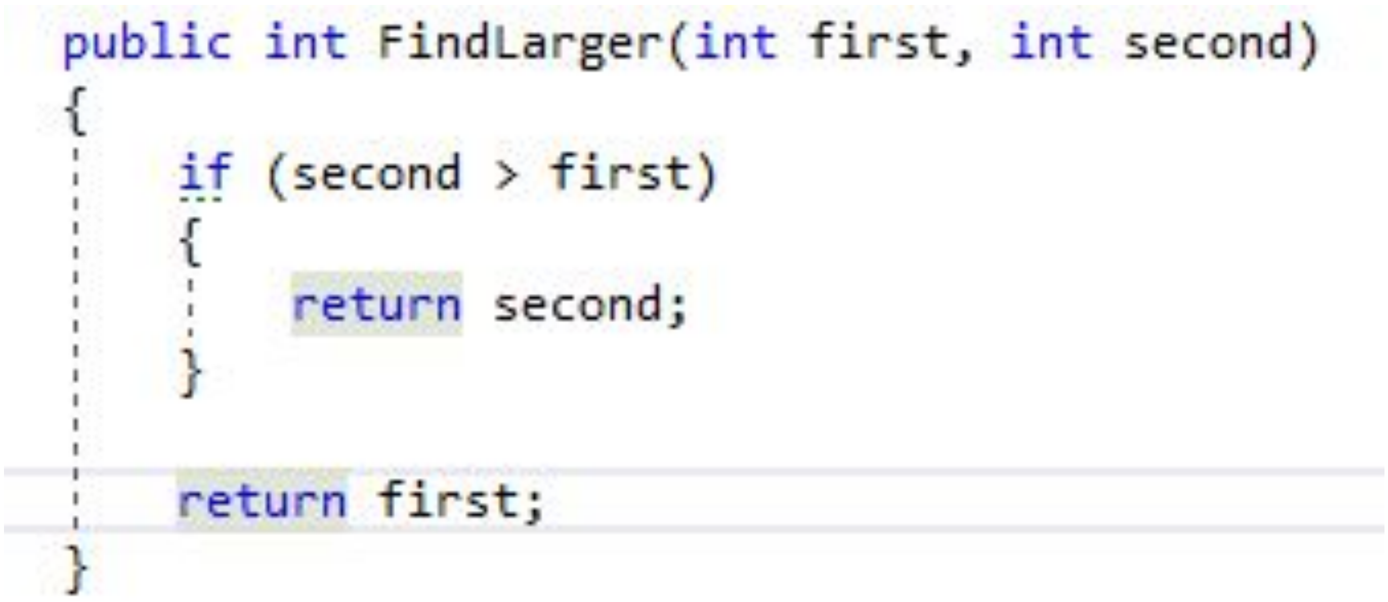
Cyclomatic Complexity

Cyclomatic Complexity: The number of distinct paths through a method

Low Cyclomatic Complexity (1)

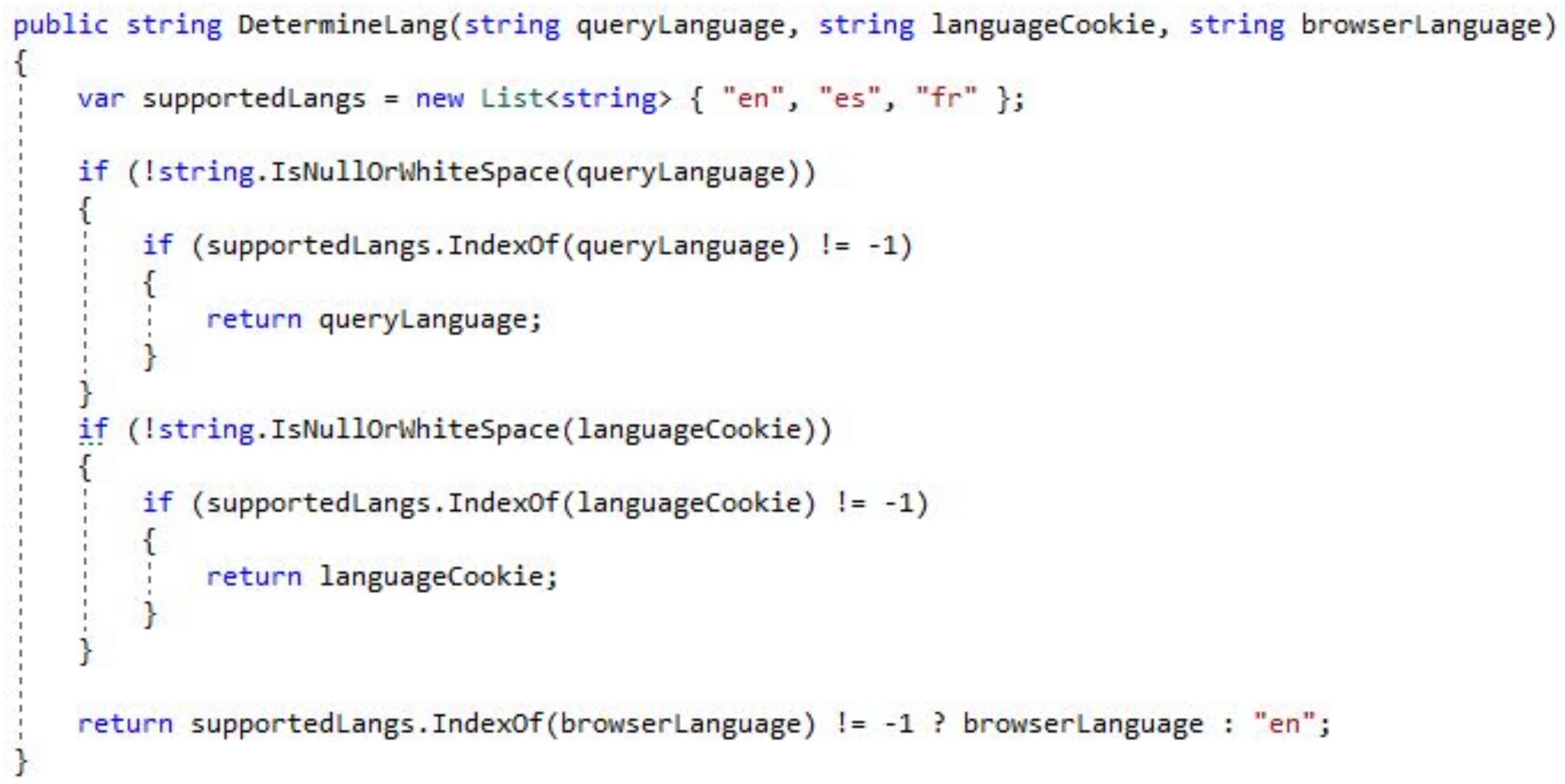

```
public int Sum(int first, int second)
{
    return first + second;
}
```

Low Cyclomatic Complexity (2)



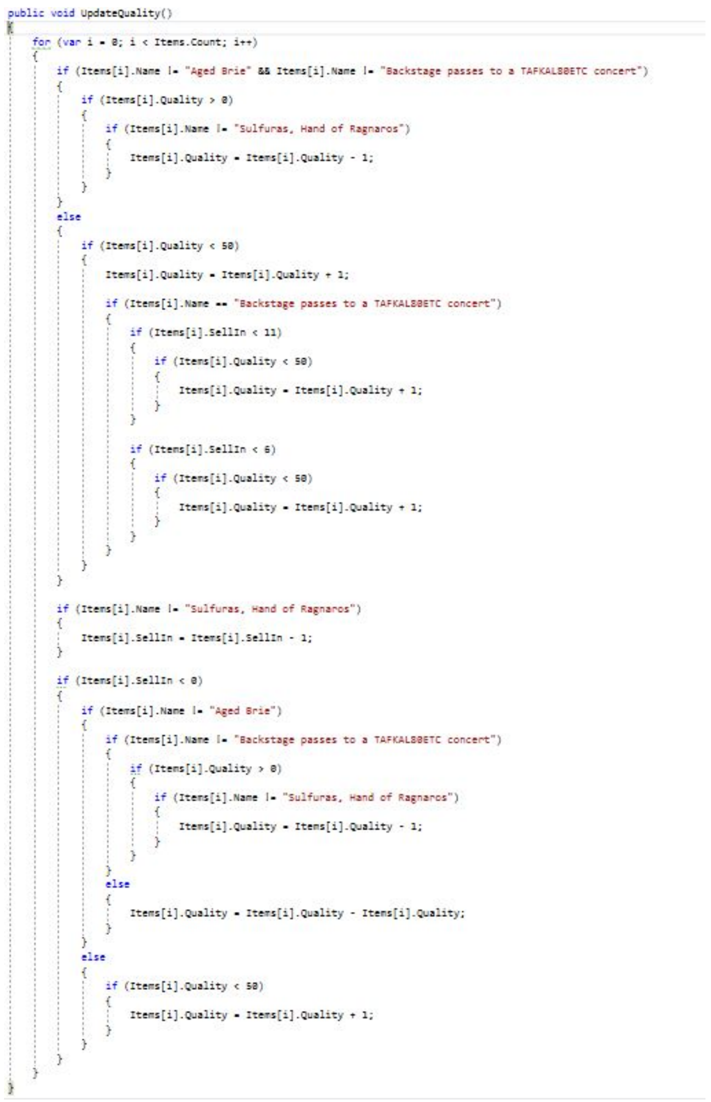
```
public int FindLarger(int first, int second)
{
    if (second > first)
    {
        return second;
    }

    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

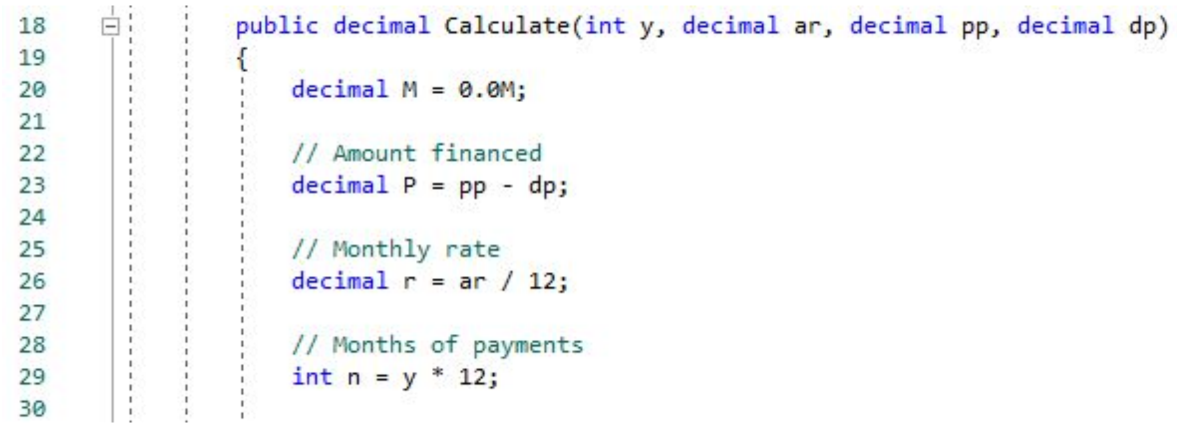
More lines of code

==

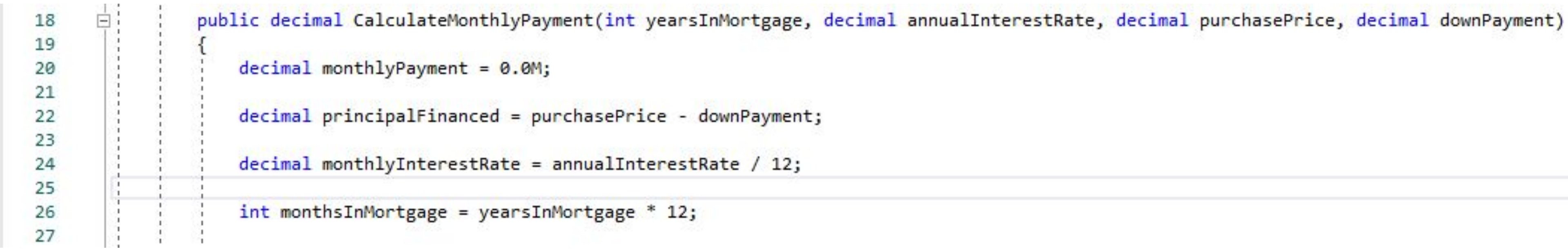
More code to maintain

Original Code Metrics

	Maintainability Index	Cyclomatic Complexity	Lines of Code
Original	49	6	26



```
public decimal Calculate(int y, decimal ar, decimal pp, decimal dp)
{
    decimal M = 0.0M;

    // Amount financed
    decimal P = pp - dp;

    // Monthly rate
    decimal r = ar / 12;

    // Months of payments
    int n = y * 12;

```

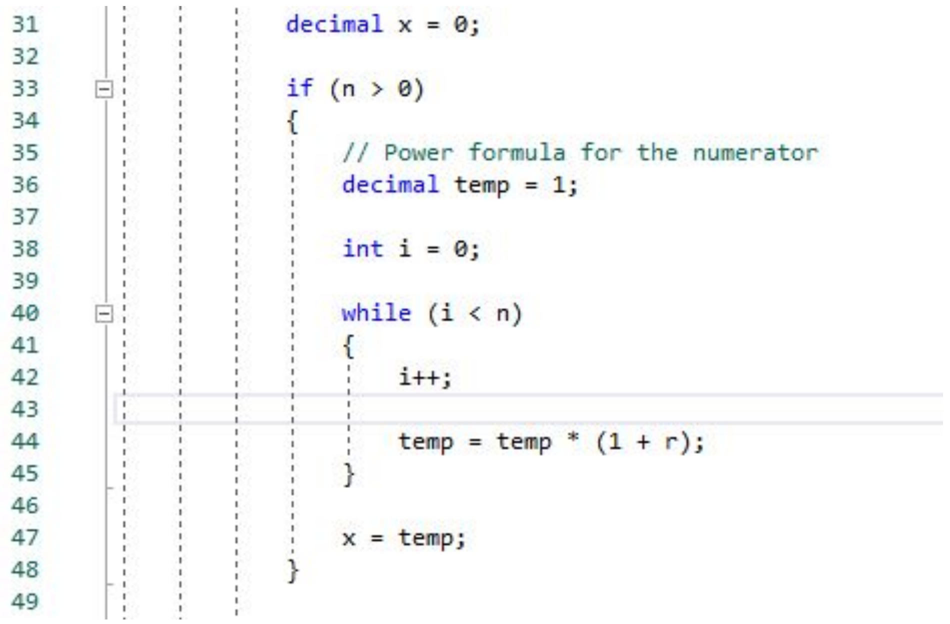


```
public decimal CalculateMonthlyPayment(int yearsInMortgage, decimal annualInterestRate, decimal purchasePrice, decimal downPayment)
{
    decimal monthlyPayment = 0.0M;

    decimal principalFinanced = purchasePrice - downPayment;

    decimal monthlyInterestRate = annualInterestRate / 12;

    int monthsInMortgage = yearsInMortgage * 12;

```

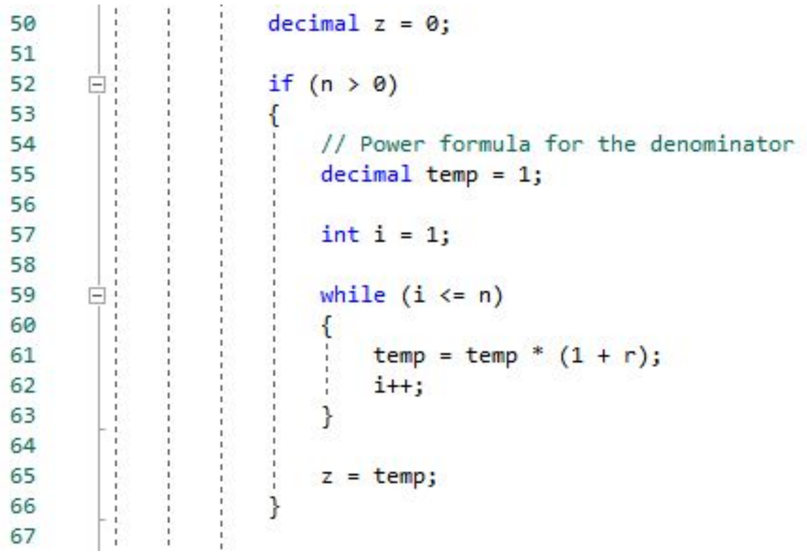
Step 1: Rename variables and methods

Code Metrics - After Step 1

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

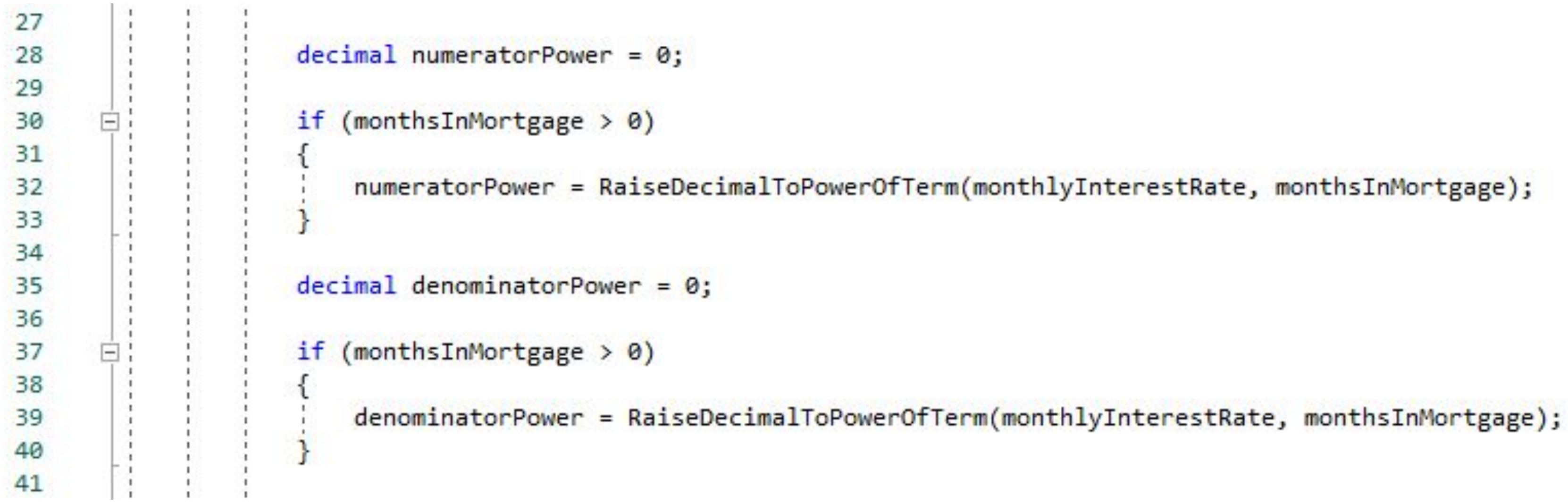


```
decimal x = 0;

if (n > 0)
{
    // Power formula for the numerator
    decimal temp = 1;

    int i = 0;

    while (i < n)
    {
        i++;

        temp = temp * (1 + r);
    }

    x = temp;
}

```

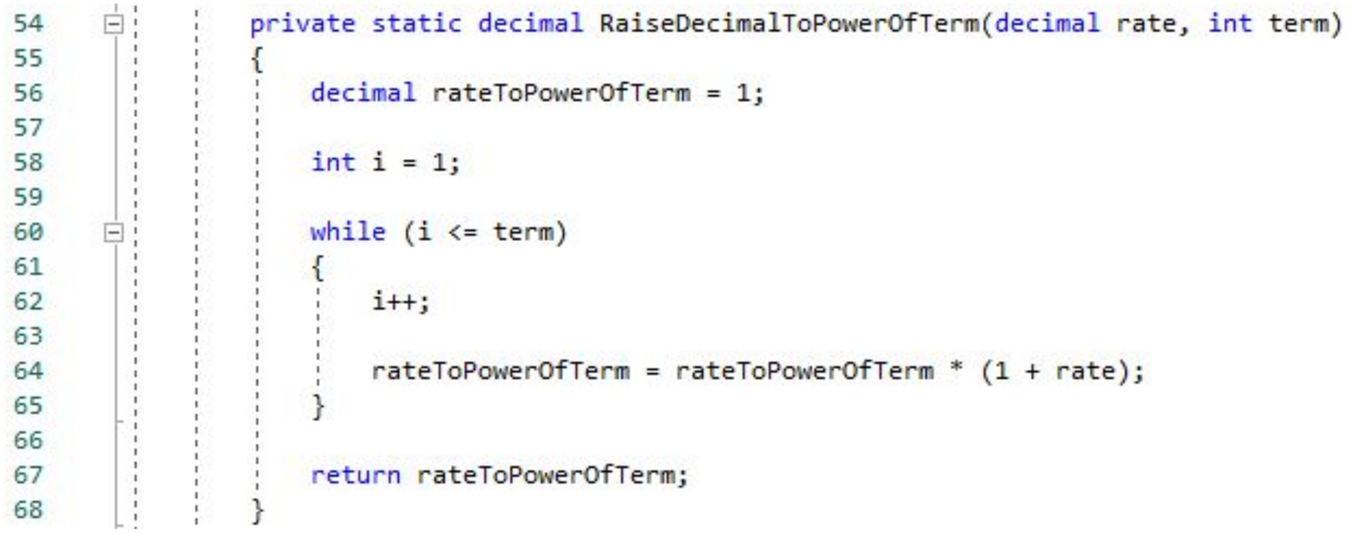


```
decimal z = 0;

if (n > 0)
{
    // Power formula for the denominator
    decimal temp = 1;

    int i = 1;

    while (i <= n)
    {
        temp = temp * (1 + r);
        i++;
    }

    z = temp;
}

```

Step 2: Extract Duplicate Code



```
decimal numeratorPower = 0;

if (monthsInMortgage > 0)
{
    numeratorPower = RaiseDecimalToPowerOfTerm(monthlyInterestRate, monthsInMortgage);
}

decimal denominatorPower = 0;

if (monthsInMortgage > 0)
{
    denominatorPower = RaiseDecimalToPowerOfTerm(monthlyInterestRate, monthsInMortgage);
}

```



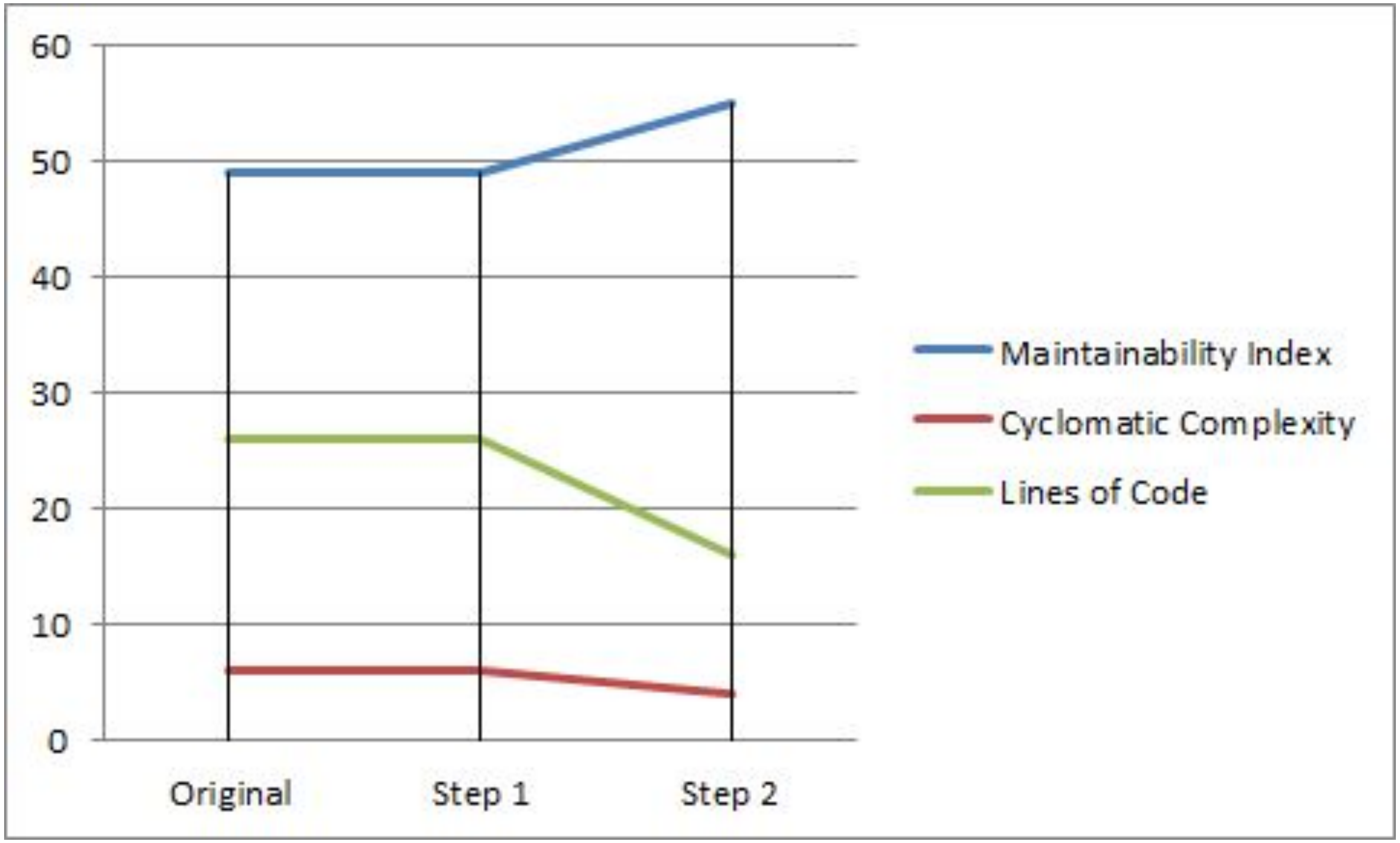
```
private static decimal RaiseDecimalToPowerOfTerm(decimal rate, int term)
{
    decimal rateToPowerOfTerm = 1;

    int i = 1;

    while (i <= term)
    {
        i++;

        rateToPowerOfTerm = rateToPowerOfTerm * (1 + rate);
    }

    return rateToPowerOfTerm;
}
```

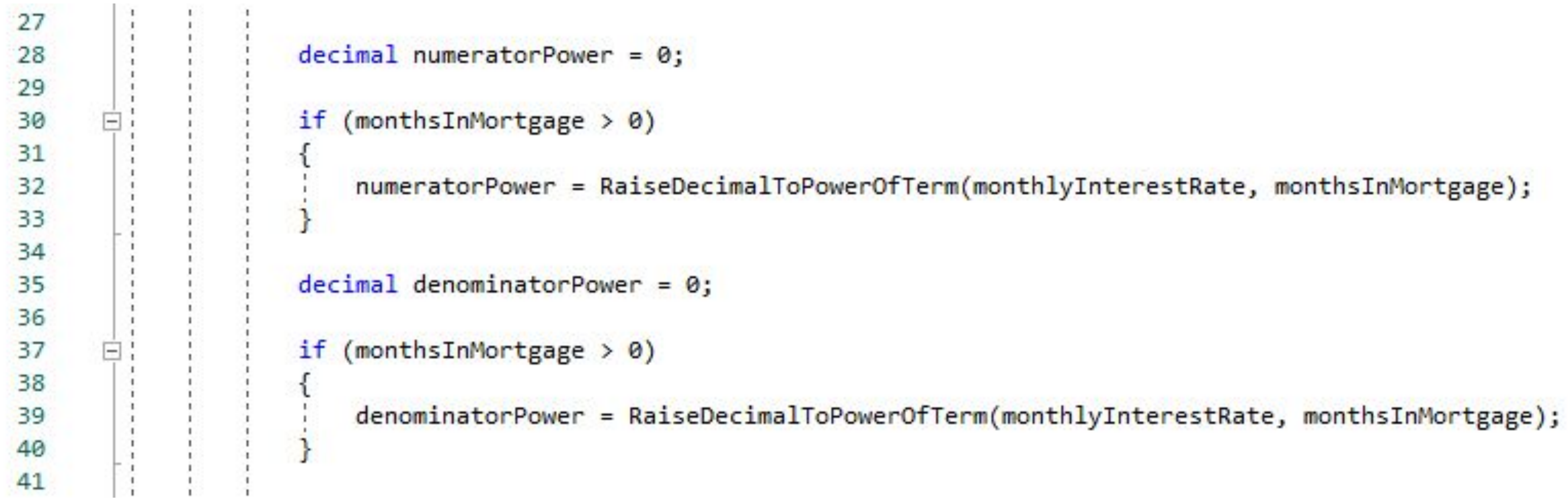
Step 2: Extract duplicate code

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

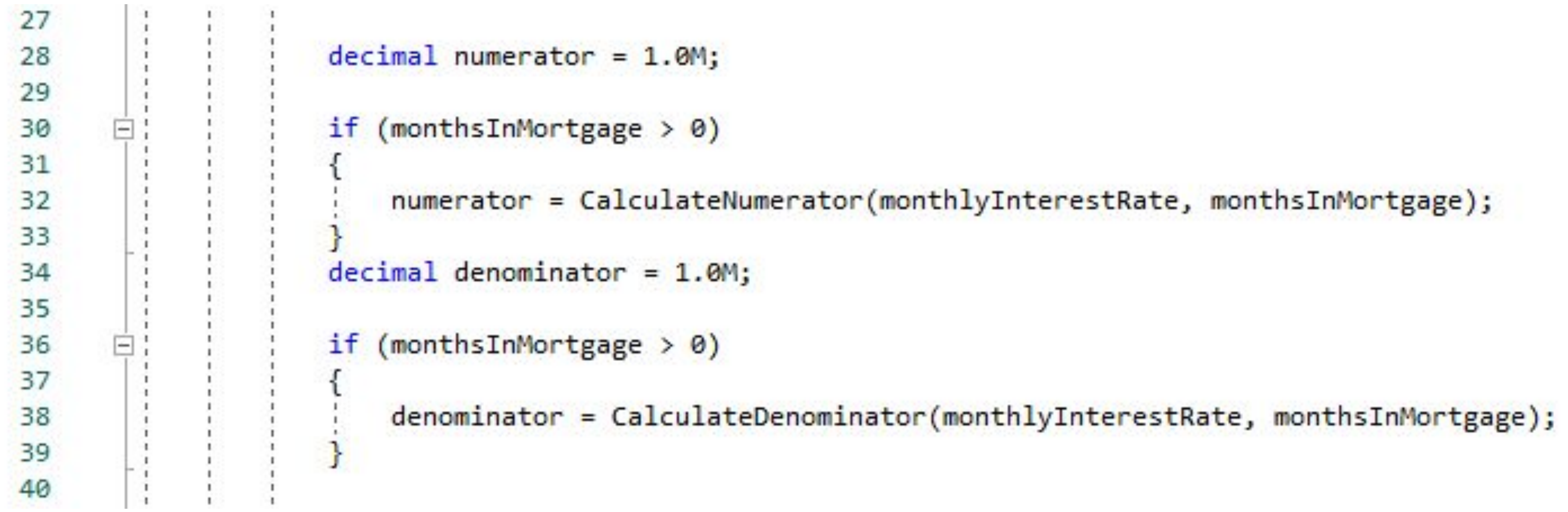
Code Metrics - After Step 2



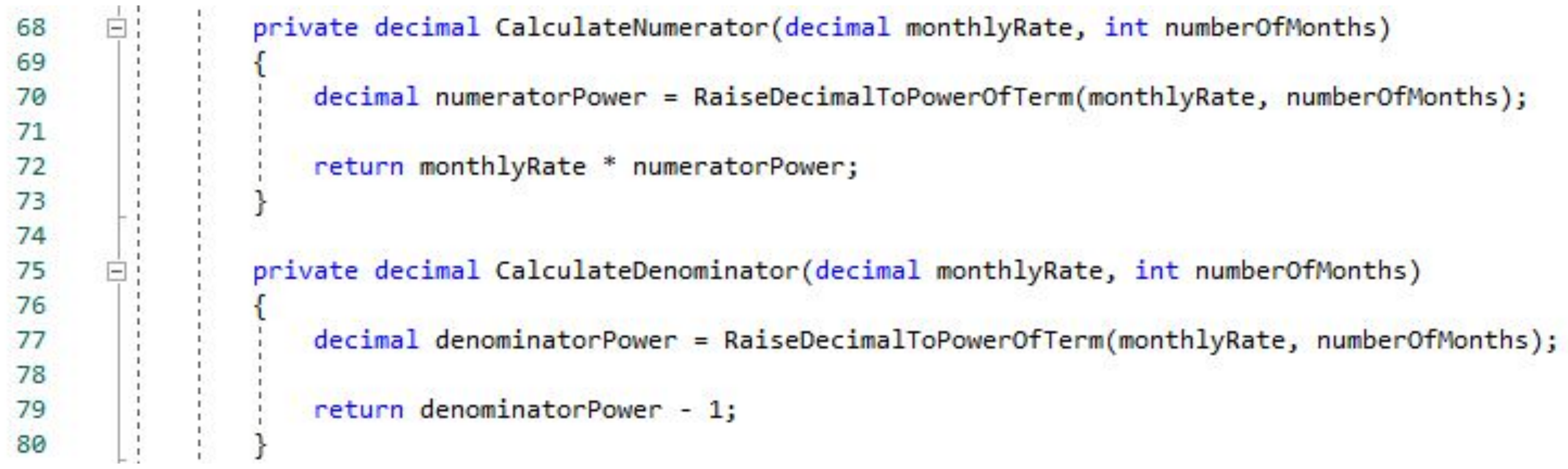


```
decimal numeratorPower = 0;

if (monthsInMortgage > 0)
{
    numeratorPower = RaiseDecimalToPowerOfTerm(monthlyInterestRate, monthsInMortgage);
}

decimal denominatorPower = 0;

if (monthsInMortgage > 0)
{
    denominatorPower = RaiseDecimalToPowerOfTerm(monthlyInterestRate, monthsInMortgage);
}

```

Step 3: Extract Methods



```
decimal numerator = 1.0M;

if (monthsInMortgage > 0)
{
    numerator = CalculateNumerator(monthlyInterestRate, monthsInMortgage);
}
decimal denominator = 1.0M;

if (monthsInMortgage > 0)
{
    denominator = CalculateDenominator(monthlyInterestRate, monthsInMortgage);
}

```

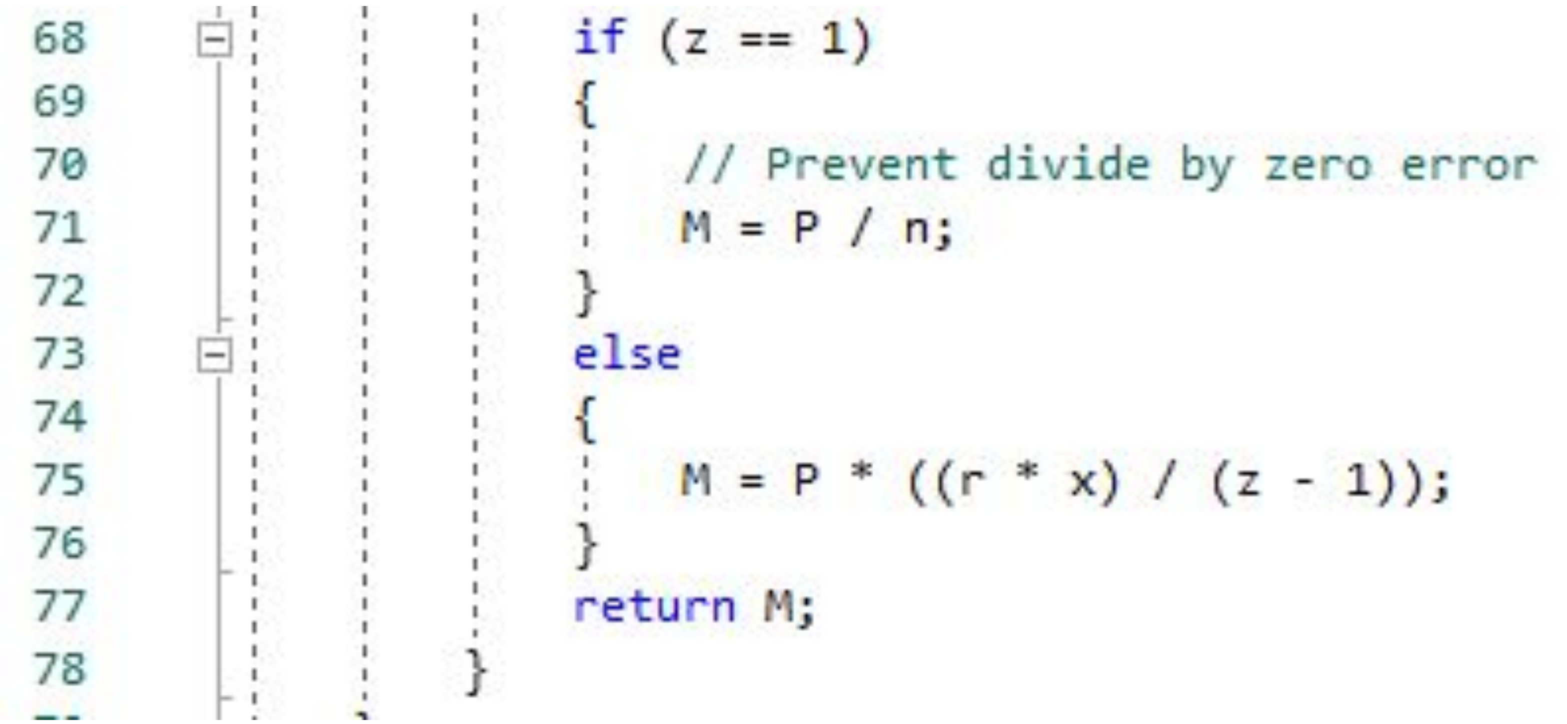


```
private decimal CalculateNumerator(decimal monthlyRate, int numberOfMonths)
{
    decimal numeratorPower = RaiseDecimalToPowerOfTerm(monthlyRate, numberOfMonths);

    return monthlyRate * numeratorPower;
}

private decimal CalculateDenominator(decimal monthlyRate, int numberOfMonths)
{
    decimal denominatorPower = RaiseDecimalToPowerOfTerm(monthlyRate, numberOfMonths);

    return denominatorPower - 1;
}
```

Step 3: Extract Methods

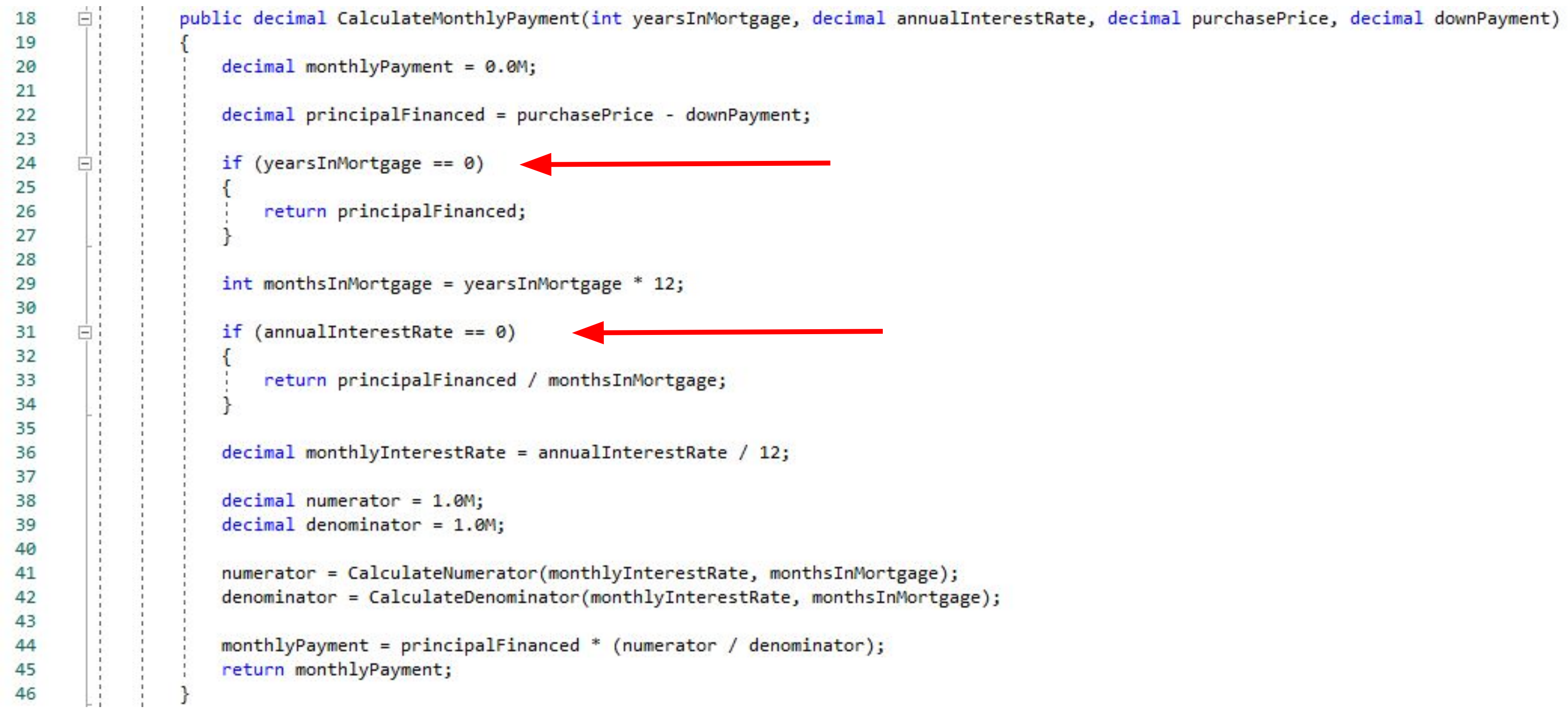
Code Metrics - After Step 3

	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



```
            if (z == 1)
            {
                // Prevent divide by zero error
                M = P / n;
            }
            else
            {
                M = P * ((r * x) / (z - 1));
            }
            return M;
        }
```

Step 4: Re-organize code

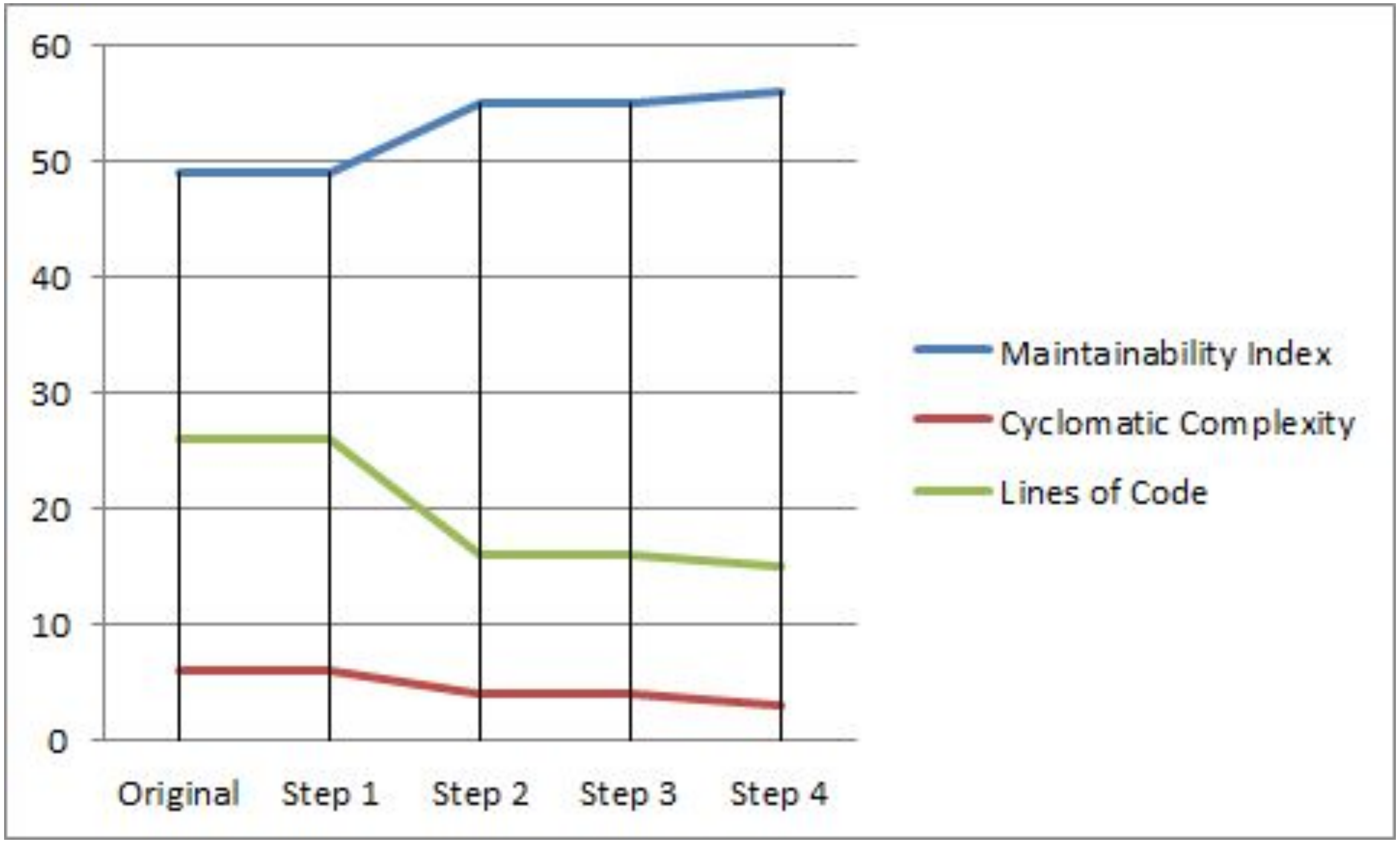

```
public decimal CalculateMonthlyPayment(int yearsInMortgage, decimal annualInterestRate, decimal purchasePrice, decimal downPayment)
{
    decimal monthlyPayment = 0.0M;

    decimal principalFinanced = purchasePrice - downPayment;

    if (yearsInMortgage == 0)
    {
        return principalFinanced;
    }

    int monthsInMortgage = yearsInMortgage * 12;

    if (annualInterestRate == 0)
    {
        return principalFinanced / monthsInMortgage;
    }

    decimal monthlyInterestRate = annualInterestRate / 12;

    decimal numerator = 1.0M;
    decimal denominator = 1.0M;

    numerator = CalculateNumerator(monthlyInterestRate, monthsInMortgage);
    denominator = CalculateDenominator(monthlyInterestRate, monthsInMortgage);

    monthlyPayment = principalFinanced * (numerator / denominator);
    return monthlyPayment;
}
```

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

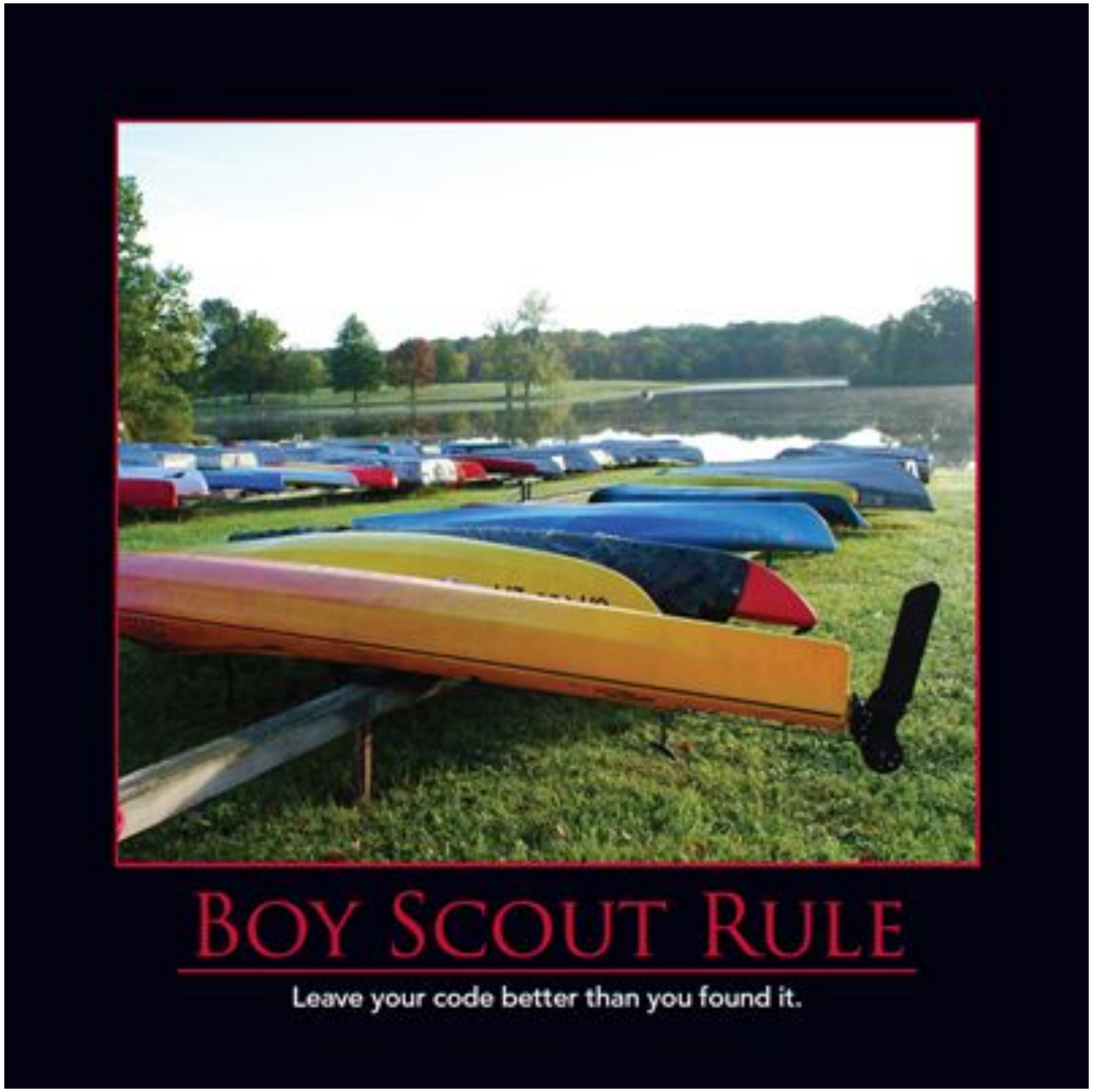
Code Metrics - Trends



When should I refactor?

Opportunistic Refactoring

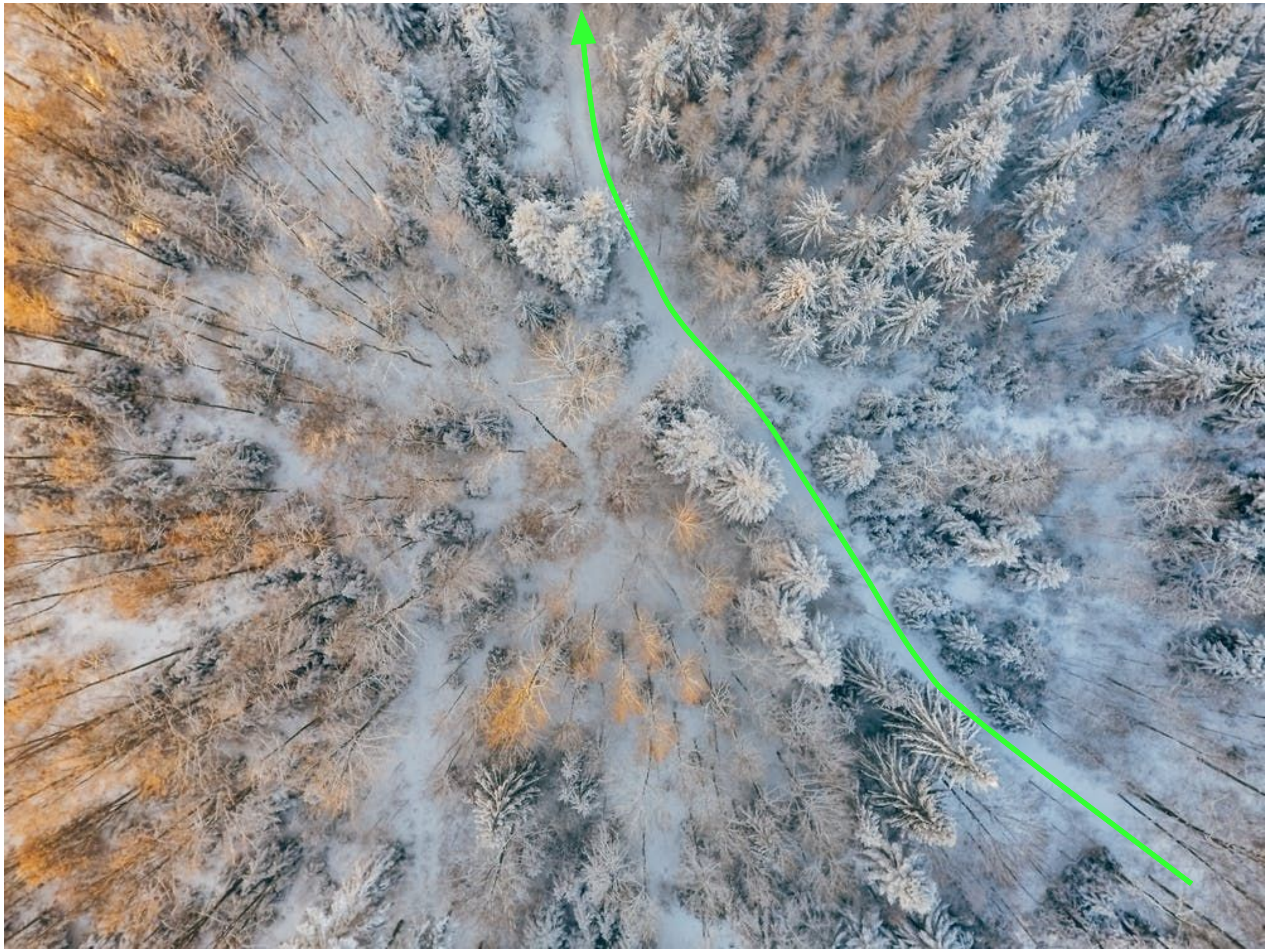
Leave It Better Than You Found It



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 Resources

https://martinfowler.com/

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

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

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

Lorien Rensing

lorien.rensing@gmail.com

https://github.com/makerlorien

@makerlorien