Refactoring Fundamentals

Method Refactorings

Steve Smith
Ardalis.com
@ardalis

pluralsight
hardcore developer training

In This Course

- ~~What is Refactoring?~~
- ~~Why do it?~~
- ~~What's the process?~~
- ~~What are some tools that can assist with it?~~
- ~~What is a *Code Smell*?~~
- ~~*What are some examples of Code Smells?*~~
- What are some common refactorings?
- How does one apply them correctly?

Method-Related Refactorings

Keep Your Methods

- **Short**
- **Clear**
- **Well-Named**
- **Focused**

- ***More method refactorings in the next module.***

- Extract Method
- Rename Method
- Inline Method
- Introduce Explaining Variable
- Inline Temp
- Replace Temp with Query
- Split Temporary Variable
- Parameterize Method
- Replace Parameter with Explicit Methods
- Add Parameter
- Remove Parameter
- Separate Query from Modifier

Extract Method

You have several lines of code that can be grouped together and given an intention-revealing name

- **Identify the code to move**
- **Create a new method with an intention-revealing name**
- **Copy the code from the source method to the new method**
- **Convert any needed local variable references to parameters**
  - Temporary variables used only in this code can be declared in the new method
- **Identify any modified local variables**
  - Consider whether the extracted method needs to return this variable
  - If there are several such variables, consider extracting a smaller method
- ***Compile***
- **Replace extracted code in source with call to new method**
- ***Compile and Test***

Guest Opinion: Scott Hanselman

Rename Method

The name of a method does not reveal its purpose.

- **Check whether the method is implemented by a sub- or superclass**
  - If so, repeat these steps with each implementation
- **Create a new method with the new name (and identical parameters)**
- **Copy (don't cut) the old body of code over into the new method**
- ***Compile***
- **Change the body of the old method to call the new method**
  - You can skip this step if the method is only called in a few places
- ***Compile and test***
- **Find all references to the old name and change them to use the new name**
  - One approach: comment out the old method and let the compiler find the references
- ***Compile and test***
- **Remove the old method (or mark as Obsolete)**
- ***Compile and test***

Guest Opinion: Scott Hanselman

Extracting and Renaming Methods

Demo

Inline Method

A method's body is just as clear as its name.

- **Confirm no subclasses override the method**
  - They can't override it if it isn't there!
- **Find all calls to the method**
  - Lean on the compiler
- **Replace each call with the method body**
- ***Compile and test***
- **Remove the method definition**

Indirection can be helpful, but needless indirection is irritating.

Martin Fowler

Inline Method

```
public void UpdateQuality()
{
   if(NameContainsBackstagePasses(Items[i].Name)
    {
    }
}

public void NameContainsBackstagePasses(string name)
{
   return name.Contains(“Backstage passes”);
}

public void UpdateQuality()
{
   if(Items[i].Name.Contains(“Backstage passes”))
    {
    }
}
```

Introduce Explaining Variable

You have a complicated expression that would be more clear if some or all of it were given an explanatory name.

- Declare a variable and set it to the result of part of the complex expression
- Replace the result part of the expression with the value of the temp
- *Compile and test*
- Repeat as needed for other parts of the expression

Introduce Explaining Variable

```
public decimal CalculateConeVolume()
{
   return Math.Pi * radius * radius * height / 3;
}


public void CalculateConeVolume()
{
    decimal coneOpeningArea = Math.Pi * radius * radius;
    return coneOpeningArea * height / 3;
}
```

$$V = \pi r^2 H/3$$
$$A = \pi r^2$$

Inline Temp

You have a temp that is assigned to once with a simple expression, and the temp is getting in the way of other refactorings.

- Confirm the temp is only assigned once
- Find all references to the temp and replace them with the right side of the temp assignment operation
- *Compile and test (after each)*
- Remove the declaration and assignment of the temp
- *Compile and test*

Most of the time Inline Temp is used as part of Replace Temp with Query…

Martin Fowler

Replace Temp with Query

You are using a temporary variable to hold the result of an expression.

- **Confirm the temp is only assigned once**
- **Extract the right-hand side of the assignment into a method**
  - Ensure the extracted method is free of side effects
- ***Compile and test***
- **Replace references to the temp with calls to the new method**
- ***Compile and test***
- **Delete the temp and its assignment**
- ***Compile and test***

Replace Temp with Query

Demo

Split Temporary Variable

You have a temporary variable assigned to more than once, but it is not a loop variable nor a collecting variable.

- Change the name of the temp at its declaration and first assignment
- Confirm the new temp is not assigned elsewhere
- Change all references of the temp up to its second assignment to use the new name
- Declare the original temp at its second assignment
- *Compile and test*
- Repeat until all temps are only assigned where declared

Parameterize Method

Several methods do similar things but with different values contained in the method body.

- **Create a parameterized method that can be substituted for each repetitive method**
- ***Compile***
- **Replace one old method with a call to the new method**
- ***Compile and test***
- **Repeat for each method, testing after each substitution**
- **Remove the original methods if they are no longer used**

Replace Parameter with Explicit Methods

You have a method that runs different code depending on the value of an enumerated parameter.

- Create a separate, explicit method for each value of the parameter
- In the original parameterized method, replace conditional bodies with calls to explicit methods
- *Compile and test* after each conditional body change
- Replace each caller of the original method with a call to the appropriate new method
- *Compile and test*
- When all callers are changed, remove the original method

Replace Parameter with Explicit Method

```
public void UpdateValue(string property, int value)
{
   if(property==“height”)
    {
      this._height = value;
      return;
    }
    if(property=“width”)
    {
      this._width = value;
      return;
    }
    // throw exception if you get here
}
```

Replace Parameter with Explicit Method

```
public void UpdateHeight(int value)
{
    this._height = value;
}

public void UpdateWidth(int value)
{
    this._width = value;
}
```

Add Parameter

A method needs more information from its caller.

- **Check to see if the signature is implemented in sub or superclasses**
  - Make a note to adjust these implementations as well
- **Declare a new method with the additional parameter(s)**
- **Copy the old method body into the new method**
- ***Compile***
- **Change the body of the old method so that it calls the new one**
- ***Compile and test***
- **Find all references to the old method and change them to the new one**
- ***Compile and test* after each replacement**
- **Remove the old method**
- ***Compile and test***

Remove Parameter

A parameter is no longer used by the method body.

- **Check to see if a superclass or subclass implements this signature**
  - If the subclass or superclass uses the parameter, STOP. Don't do this refactoring.
- **Declare a new method without the parameter**
- **Copy the old body of code into the new method's body**
- ***Compile***
- **Change the body of the old method to call the new one**
- ***Compile and test***
- **Change each reference to the old method to call the new one**
- ***Compile and test* after each change**
- **Remove the old method**
- ***Compile and test***

Guest Opinion: Scott Hanselman

Separate Query from Modifier

You have a method that returns a value
but also changes the state of an object.

- Create a query that returns the same value as the original method
- Modify the original method so that it returns the result of a call to the new query method
- *Compile and test*
- For each call, replace the single call to the original method with a call to the query. Add a call to the original method before the line that calls the query.
- *Compile and test* after each change
- Update the original method to return void and remove all return statements from it
- *Compile and test*

Summary

- Extract Method
- Rename Method
- Inline Method
- Introduce Explaining Variable
- Inline Temp
- Replace Temp with Query
- Split Temporary Variable
- Parameterize Method
- Replace Parameter with Explicit Method
- Add Parameter
- Remove Parameter
- Separate Query from Modifier

References

Books

Refactoring http://amzn.to/110tscA

Web

Refactoring Catalog http://www.refactoring.com/catalog/

Thanks!

Steve Smith

Ardalis.com

Twitter: @ardalis

To Teach Is To Learn Twice

pluralsight
hardcore developer training